A comparison of the String Gradient Weighted Moving Finite Element method and a Parabolic Moving Mesh Partial Differential Equation method for solutions of Partial Differential Equations

A. Wacher
BCAM - Basque Center for Applied Mathematics
Mazarredo 14, E48009 Bilbao, Basque Country - Spain.
E-mail: awacher@bcamath.org
and
Department of Mathematical Sciences, Durham University
Science Laboratories, South Rd, Durham DH1 3LE,
United Kingdom. E-mail: abigail.wacher@durham.ac.uk

May 30, 2012

Abstract

We compare numerical experiments from the String Gradient Weighted Moving Finite Element Method and a Parabolic Moving Mesh Partial Differential Equation method, applied to three benchmark problems based on two different partial differential equations. Both methods are described in detail and we highlight some strengths and weaknesses of each method via the numerical comparisons. The two equations used in the benchmark problems are the viscous Burgers' equation and the porous medium equation, both in one dimension. Simulations are made for the two methods for: a) a travelling wave solution for the viscous Burgers' equation, b) the Barenblatt self-similar analytical solution of the porous medium equation, and c) a waiting-time

solution for the porous medium equation. Simulations are carried out for varying mesh sizes, and the numerical solutions are compared by computing errors in two ways. In the case of an analytic solution being available, the errors in the numerical solutions are computed directly from the analytic solution. In the case of no availability of an analytic solution, an approximation to the error is computed using a very fine mesh numerical solution, as the reference solution.

Keywords Moving meshes, weighted moving finite elements, moving mesh partial differential equations, numerical solutions of partial differential equations, porous medium equation, waiting-time solutions, viscous Burgers' equation **Subject code 65M50, 35R37**

1 Introduction

In this paper we investigate two moving mesh methods and their application to unsteady flow problems. One method is called String Gradient Weighted Moving Finite Elements (SGWMFE) in [5], and the other method is called Parabolic Moving Mesh Partial Differential Equations (MMPDE) in [14]. Classic problems often used for testing moving mesh methods or shock capturing methods are: Burgers' equation and Sod's shock tube problem, with initial conditions which result in steep moving fronts. Moving mesh methods have proved useful for solving problems dealing with moving complex structures, such as the classic problems just mentioned, problems which are difficult to solve using conventional numerical methods.

The SGWMFE method which we will describe in the first section was developed in [4, 10] and later extended for use in multiple dimensions in [5]. The SGWMFE formulation for systems of partial differential equations was originally proposed by [3] as an alternative formulation of the Gradient Weighted Moving Finite Element (GWMFE) method, which was developed in detail in [1] and [2] by Carlson and Miller for one and two-dimensional problems. [1] report on the design and implementation of a robust and versatile GWMFE code in one dimension applied to various PDEs and PDE systems. Sample problems for which the code was tested in that paper are: 1) a convection-diffusion boundary layer problem, 2) the viscous Burgers' equation, plus a strong nonlinear source term and also with no diffusion, 3) drift-diffusion equations for semiconductors, 4) Sod's shock tube problem, and 5) a steady-state convection problem. There they found that GWMFE efficiently pro-

duces accurate results for problems which form steep moving fronts. The corresponding two-dimensional paper [2] does the same as the one-dimensional paper including the additional application problems: 1) Non-linear arsenic diffusion, 2) The BuckleyLeverett "black oil" equations, and 3) motion by mean curvature which was implemented in the one-dimensional case in the paper [3] by Miller. The results therein show that the method is intended for "problems with sharp moving fronts where one needs to *resolve* the fine-scale structure of the front to compute the correct answer" greatly improving on the first Moving Finite Element (MFE) method developed originally by K. Miller and R.N. Miller in [8] in one dimension, and by K. Miller in [9] for two dimensions. In the thesis [10], the SGWMFE method was implemented for several sample problems, some of which were implemented in one dimension and some in two dimensions. Sample problems for which the code was tested in that thesis are: 1) Sod's shock tube problem, 2) the porous medium equation, 3) a two phase oil reservoir model, and 3) the shallow water equations. The article [5] which develops the generalised SGWMFE method includes results from the implementation of the method in two dimensions for 1) the porous medium equation, 2) the shallow water equations, and 3) the Gray Scott equations.

Some early papers using moving mesh methods, which apply equidistribution ideas to solve one-dimensional time-dependent partial differential equations, are found in [27], [28], and [29]. Over the last two decades, moving mesh methods based on applying an Equidistribution Principle are often called Moving Mesh Partial Differential Equation (MMPDE) methods. To the best of our knowledge, the concept and name of "MMPDE" was first proposed in [14]. The MMPDE method defines a PDE for the time-dependent grid points in addition to the PDEs which have already needed to be solved for a given problem. These equations are usually discretised in space using standard finite difference or finite element techniques. The idea used for MMPDE methods to equidistribute the nodes using a weight function stems originally from De Boor's Equidistribution principle (EP), in [30]. Given a function defined on a discrete set of points, the idea of equidistribution to define an MMPDE is to distribute a particular quantity equally over all intervals over the given set of points. The quantity to be equidistributed by using the MMPDE method is commonly a monitor function which traditionally aimed to put more nodes in places where the gradients are large, though more recently various other types of monitor functions have been studied. The idea of equidistribution using a monitor function is used much before

the term MMPDEs was used. For example, in [27], the author uses various monitor functions, including the arc length and local truncation error functions, for the "selection of equidistributing meshes for two-point boundary value problems". In general, the equidistribution of a monitor function is used because one of the goals of a moving mesh method is to place nodes in regions where errors are expected to be large, that is, in regions where more nodes are needed in order to resolve the problem, though there is much debate as to where nodes should be placed. For example when there is a shock or near shock in a solution, nodes are needed at the lip and tale of the shock or near shock, otherwise the difference between the numerical approximation to the solution and the actual solution could be very large.

There are various beliefs of where the nodes should be placed, one of the classical beliefs suggests that nodes should be placed where the solution has greatest curvature, and another classical belief suggests to put nodes where the gradient is the largest. Methods such as GWMFE aim to put nodes where the curvature is largest, and MMPDE methods with arc length monitor functions aim to put nodes where the gradient is largest. In this paper we aim to compare the SGWMFE/GWMFE method and an MMPDE method with a classical monitor function applied to partial differential equations which have steep moving fronts or boundaries. One benchmark problem is of capturing a moving shock for the viscous Burgurs' equation. A second benchmark problem is to resolve the porous medium equation for which we can compare the numerical results with an analytic solution. The third benchmark problem is a waiting-time solution to the porous medium equation, where the waiting-time is defined as the time at which the boundary begins to move. As research progresses on the two methods discussed, we believe it is important to compare the methodologies and highlight strengths and weaknesses of these two methods, as a way of identifying challenges to be resolved by the methods in further research.

This paper is constructed as follows. After the introductory section, in the second section we develop the SGWMFE method for a scalar equation in one dimension. In the third section we develop the MMPDE method for a scalar equation in one dimension for three different monitor functions. Section four outlines the two model problems: viscous Burgers' equation and the porous medium equation, and discusses: a) a travelling wave solution for the viscous Burgers' equation, b) the Barenblatt self-similar analytical solution of the porous medium equation, and c) a waiting-time solution for the porous medium equation. The fifth section then discusses the figures

therein which show comparison error plots of the SGWMFE method, and the MMPDE method with the different monitor functions. Simulations are carried out for varying mesh sizes, and the numerical solutions are compared by computing errors in two ways. In the case of an analytic solution being available, the errors in the numerical solutions are computed directly from the analytic solution. In the case of no availability of an analytic solution, an approximation to the error is computed using a very fine mesh numerical solution, as the reference solution. The paper concludes with a final section summarising the results.

2 String Gradient Weighted Moving Finite Elements

What follows is the SGWMFE formulation for a scalar equation in one dimension. We note that in the case of a single PDE the SGWMFE and GWMFE reduce to the same set of equations, however here we use the SGWMFE approach to developing the system of equations following [10]. The theory is presented in such a way that it should be clear how SGWMFE is generalised to systems of equations with any number of components, in multiple dimensions, however see [10, 5] for the detailed extensions.

2.1 Formulation of the SGWMFE method

Given a partial differential equation as in (1), SGWMFE treats the solution graph for the equation as an evolving one-dimensional manifold $(x, u(x,t))$. To begin, consider the example of a PDE,

$$u_t = L(u) \tag{1}$$

for the unknown function $u(x,t)$ on a one-dimensional spatial interval Ω. L is a general first or second order nonlinear differential operator in space.

Consider a re-parameterisation with a moving coordinate $x(\tau,t)$, where τ is a one-dimensional parameter whose domain is arbitrary but bounded. Under the re-parameterisation the solution manifold becomes an evolving parameterised one-dimensional manifold immersed in two dimensions with the position vector

$$\mathbf{u}(\tau,t) = (x(\tau,t), u(\tau,t)),$$

for the evolving re-parameterised points of the solution graph.

At each parameterised point on the evolving manifold, we split the velocity vector $\dot{\mathbf{u}} = (\dot{x}, \dot{u})$ into its tangential, $[\dot{\mathbf{u}}]_T$, and its normal, $[\dot{\mathbf{u}}]_N$, parts. Noting that solving for the tangential part $[\dot{\mathbf{u}}]_T$ makes no changes to the solution manifold since any points along the manifold moving tangentially stay on the manifold, thus maintaining it the same solution manifold (not necessarily the graph of a function). The original PDE (1), is written in the following vector form:

$$\begin{pmatrix} 0 \\ u_t \end{pmatrix} = \begin{pmatrix} 0 \\ L(u) \end{pmatrix}. \tag{2}$$

Taking the normal component of equation (2) results in the same vector regardless of the parameters used to describe the velocity of the solution manifold, that is

$$\begin{pmatrix} \dot{x} \\ \dot{u} \end{pmatrix}_N = \begin{pmatrix} 0 \\ u_t \end{pmatrix}_N. \tag{3}$$

For a proof of equation (3) see [10]. The equation for the normal velocity is then written

$$\begin{pmatrix} \dot{x} \\ \dot{u} \end{pmatrix}_N = \begin{pmatrix} 0 \\ L \end{pmatrix}_N, \tag{4}$$

which is a system of two PDEs for the two unknown variables $x(\tau, t), u(\tau, t)$. Equation (4) is only a parameterisation of equation (1) as long as the solution manifold is the graph of a function.

It is convenient to use here the projection matrix P which projects any given vector, $\mathbf{F}$, into its normal part, $[\mathbf{F}]_N$, at a given point on the solution manifold, $(x, u(x))$. $[\mathbf{F}]_N$ is obtained by subtracting, from $\mathbf{F}$, the tangential component $[\mathbf{F}]_T$, where the tangential component is given by

$$[\mathbf{F}]_T = \mathbf{t}\mathbf{t}^{\mathrm{T}}\mathbf{F},$$

where $\mathbf{t} = (1, u_x)/\sqrt{1 + u_x^2}$ is the unit tangent vector to the manifold at this point. Hence

$$[\mathbf{F}]_N = \mathbf{F} - [\mathbf{F}]_T = (\mathbf{I} - \mathbf{t}\mathbf{t}^{\mathrm{T}})\mathbf{F} = P\mathbf{F}, \tag{5}$$

where

$$P = \frac{1}{1 + u_x^2} \begin{pmatrix} u_x^2 & -u_x \\ -u_x & 1 \end{pmatrix}.$$

Equation (4) is then discretized by letting the SGWMFE approximation be an evolving, piecewise linear manifold with its two-dimensional nodal

positions $\mathbf{u}_j = (x_j(t), u_j(t))$ as unknowns. Multiplying equation (5) by the well known nodal "hat" basis function α^j, and integrating over the spatial domain gives:

$$\int \begin{pmatrix} \dot{x} \\ \dot{u} \end{pmatrix}_N \alpha^j \mathrm{d}s = \int \begin{pmatrix} 0 \\ L \end{pmatrix}_N \alpha^j \mathrm{d}s, \tag{6}$$

at each node j.

2.2 Time derivative terms

Using equation (6), the integrals of the time derivatives (the left hand side of the PDE system) in $cell_i$ contribute to the i^{th} node by:

$$\int_{cell_i} \begin{pmatrix} 0 \\ u_t \end{pmatrix}_N \alpha^i \mathrm{d}s = \int_{cell_i} P \begin{pmatrix} \dot{x} \\ \dot{u} \end{pmatrix} \alpha^i \mathrm{d}s.$$

The integral is then obtained by using Simpson's quadrature rule found in many introductory mathematical and physical books such as [11],

$$\int_{cell_i} P\dot{\mathbf{u}}\alpha^i \mathrm{d}s = P(\frac{1}{3}\dot{\mathbf{u}}_i + \frac{1}{6}\dot{\mathbf{u}}_{i-1})\Delta s_i, \tag{7}$$

where

$$\Delta s_i = \| \mathbf{u}_i - \mathbf{u}_{i-1} \|_2 = \sqrt{(x_i - x_{i-1})^2 + (u_i - u_{i-1})^2},$$

and noting that the term u_x is constant on each i^{th} cell. This implies that $\mathrm{d}s = \sqrt{1 + u_x^2}\mathrm{d}x$ and the elements of the matrix P are also constant on the i^{th} cell. Further, since α^i and $\dot{\mathbf{u}}$ are both linear functions on the cell, then the integrand above is at most a quadratic polynomial on the i^{th} cell, which is the interval of integration. Since Simpson's rule is exact for polynomials of up to third order the expression for the integral in equation (7) is exact.

Once the ODE system has been constructed we use the numerical integration code developed in [1]. The integration method used is the Backward Differentiation Formula 2 (BDF2), an implicit ODE solver with adaptive time stepping. For details of the implementation of this code see [1].

2.3 Flux terms

Consider restricting the non-linear operator L to have the particular form of a flux function:

$$u_t = -f_x(u, v).$$

For a scalar function $f(x, u(x))$ it will be useful to denote by f_i its value at the i^{th} node and by $[f]_i$, the average over the cell. Thus $f_i = f(x_i, u_i)$ and $[f]_i = \frac{1}{\Delta x_i}\int_{cell_i} f \mathrm{d}x$. Using this notation and noting that $\mathrm{d}s = \sqrt{1+u_x^2}\mathrm{d}x$, the integral contributions from the flux term in $cell_i$ onto the i^{th} node can be written

$$\begin{aligned}
\int_{cell_i} P \begin{pmatrix} 0 \\ -f_x \end{pmatrix} \alpha^i \mathrm{d}s &= P \begin{pmatrix} 0 \\ 1 \end{pmatrix} \int_{cell_i} (-f_x \alpha^i) \mathrm{d}s && (8)\\
&= \begin{pmatrix} -u_x \\ 1 \end{pmatrix} \frac{\int_{cell_i} (-f_x \alpha^i) \mathrm{d}x}{\sqrt{1+u_x^2}} && (9)\\
&= \begin{pmatrix} -u_x \\ 1 \end{pmatrix} \frac{[f]_i - f_i}{\sqrt{1+u_x^2}}. && (10)
\end{aligned}$$

2.4 Constant coefficient diffusion terms

Now consider a term with a non-linear operator L to have an artificial diffusivity term:

$$u_t = \epsilon u_{xx},$$

where ϵ determines the magnitude of the artificial diffusion (here it is a constant). Using piecewise linear basis functions means that diffusive terms vanish in the interior of any cell but are undefined at nodes. One way of dealing with this problem is to use the mathematical technique of mollification as in [1]. The first derivative, while being constant over the main body of the cell, is assumed to vary smoothly between cell values in a small neighbourhood of width 2δ at each node, see Figure 1. Then take the limit $\delta \to 0$. Thus in any integral involving diffusion terms it is only necessary to take into account the small neighbourhoods near each node since u_{xx} is still identically zero for most part of each cell. The use of the mollification technique is presented below using the same principles as in [1] and [3]. For further reading on the use of mollification for MFE and GWMFE see [12].

Denote the value of u_x on $cell_i$ as m_i, then mollify by defining a variable $\sigma(x)$ and then, for instance at the right end of the cell, map a neighbourhood of width 2δ of x_i to $-1 \le \sigma(x) \le 1$, map u_x to $m_i \le u_x \le m_{i+1}$. Then the integral contribution on the i^{th} node due to diffusive terms is:

$$\epsilon \int_{-\delta}^{\delta} P \begin{pmatrix} 0 \\ u_{xx} \end{pmatrix} \alpha^i \mathrm{d}s = \epsilon \int_{-\delta}^{\delta} \frac{1}{\sqrt{1+u_x^2}} \begin{pmatrix} -u_x u_{xx} \\ u_{xx} \end{pmatrix} \mathrm{d}x, \qquad (11)$$

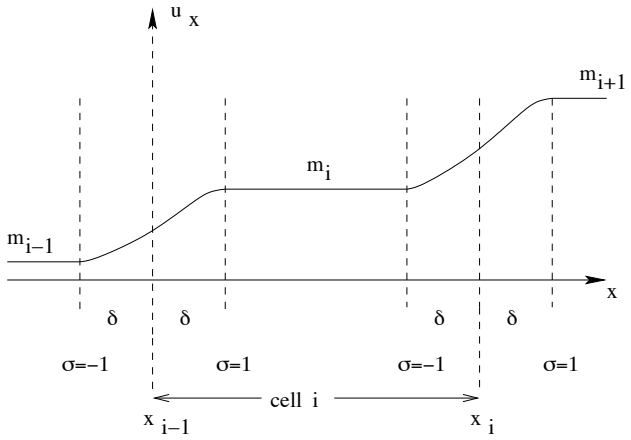

Figure 1: Mollification: the value of u_x on $cell_i$ is m_i, the value of u_x on $cell_{i-1}$ is m_{i-1} and the value of u_x on $cell_{i+1}$ is m_{i+1}. The value of u_x is assumed to vary smoothly in a small neighbourhood of each end of the cell of width 2δ and then $\delta \to 0$.

where α^i has been replaced by 1 since the integral is taken over the infinitesimal neighbourhood of x_i which is the point where $\alpha^i = 1$. The integral is now rewritten using the mapping

$$u_x = m_c + \Delta m_c \sigma(x), \tag{12}$$

where $m_c = \dfrac{m_i + m_{i+1}}{2}$, and $\Delta m_c = \dfrac{m_{i+1} - m_i}{2}$. With the first component of equation (11) in mind, let

$$I_A = \int_{-\delta}^{\delta} \frac{u_x u_{xx}}{\sqrt{1 + u_x^2}} \mathrm{d}x. \tag{13}$$

Noting that when describing the mollification of u_x it is important to understand that the function u_x that is being mollified is the approximation of the actual unknown variable. Substituting equation (12) into (13), so that

$$\mathrm{d}u_x = u_{xx}\mathrm{d}x = \frac{\mathrm{d}}{\mathrm{d}\sigma} u_x \, \mathrm{d}\sigma,$$

then under change of variable for the integral, the neighbourhood size falls

out, and by letting

$$\begin{aligned} a &= 1 + m_c^2, \\ b &= 2m_i \Delta m_c, \\ c &= {\Delta m_c}^2, \end{aligned}$$

results in the following simplified expression for I_A in terms of σ:

$$I_A = \frac{b}{2} \int_{-1}^{1} \frac{\mathrm{d}\sigma}{\sqrt{a + b\sigma + c\sigma^2}} + c \int_{-1}^{1} \frac{\sigma \mathrm{d}\sigma}{\sqrt{a + b\sigma + c\sigma^2}}. \tag{14}$$

Similarly, applying the same technique to the second component leads to a similar integral expression, see [10].

Despite knowing analytic expressions for these integrals, the expressions for the integrals can be subject to severe roundoff errors and great care has to be taken in their evaluation, as in [1] and [2] where they develop formulas for these integrals to avoid roundoff error.

2.5 Non-uniform diffusion terms in conservative form

Now consider the semi-linear diffusion terms from a non-linear operator L of the form $(au_x)_x$, where $a = a(x, u)$. The contribution from this term in the i^{th} cell onto its i^{th} node can be written

$$\begin{aligned} \int P \begin{pmatrix} 0 \\ a(x,u)u_x \end{pmatrix}_x \alpha^i \mathrm{d}s &= P_i \begin{pmatrix} 0 \\ m_i \end{pmatrix} \int_{cell_i \setminus nbd_i} a_x \alpha^i \mathrm{d}s \\ &+ \int_{nbd_i} P \begin{pmatrix} 0 \\ a_x u_x \end{pmatrix} \alpha^i \mathrm{d}s \\ &+ a_i \int_{nbd_i} P \begin{pmatrix} 0 \\ u_{xx} \end{pmatrix} \alpha^i \mathrm{d}s \\ &+ P_{i+1} \begin{pmatrix} 0 \\ m_{i+1} \end{pmatrix} \int_{cell_{i+1} \setminus nbd_i} a_x \alpha^i \mathrm{d}s, \end{aligned} \tag{15}$$

where m_i and m_{i+1} are the values of u_x on $cell_i$ and $cell_{i+1}$ respectively.

The second term on the right hand side of equation (15) is identically zero since the integrand is bounded in the infinitesimal neighbourhood of the i^{th} node. Note that in the third term on the right hand side of equation (15),

the value a_i has been factored out. This is because $a(x,u)$ is replaced by a_i, its value near the i^{th} node, where u_{xx} has its infinitesimal support. See Section 4.5 of [1] for a similar procedure. The integrand in this same term is obtained from the theory for the constant coefficient Laplacian terms. The integrands in the other two terms on the right hand side of equation (15), the first and fourth terms, can be derived using integration by parts as was done for the flux terms, leading to the first and third terms in equation (16). As before let m_i be the value of u_x on $cell_i$, and let m_{i+1} be the value of u_x on $cell_{i+1}$.

$$\begin{aligned}\int P \begin{pmatrix} 0 \\ a(x,u)u_x \end{pmatrix}_x \alpha^i \mathrm{d}s &= m_i \begin{pmatrix} -m_i \\ 1 \end{pmatrix} \frac{\int_{cell_i \setminus nbd_i} a_x \alpha^i \mathrm{d}x}{\sqrt{1+m_i^2}} \\ &+ a_i \int_{nbd_i} P \begin{pmatrix} 0 \\ u_{xx} \end{pmatrix} \alpha^i \mathrm{d}s \qquad (16) \\ &+ m_{i+1} \begin{pmatrix} -m_{i+1} \\ 1 \end{pmatrix} \frac{\int_{cell_{i+1} \setminus nbd_i} a_x \alpha^i \mathrm{d}x}{\sqrt{1+m_{i+1}^2}},\end{aligned}$$

where $\int_{cell_i \setminus nbd_i} a_x \alpha^i \mathrm{d}x = a_i - [a]_i$ and $\int_{cell_{i+1} \setminus nbd_i} a_x \alpha^i \mathrm{d}x = [a]_{i+1} - a_i$, using integration by parts as in the flux terms in equations (8) to (10). The second term in equation (16) is the Laplacian term with constant coefficient a_i, identical to what has been derived in the previous section.

2.6 Regularization

Regularization of the mass matrix (resulting from the MFE, GWMFE or SGWMFE discretizations) is used to avoid the mass matrix from becoming ill-conditioned from possible degenerate nodes or cells. The first type of possible degeneracy is that discussed in [13], for the MFE method where the same phenomena occurs, whereby the i^{th} block of the mass matrix is analyzed, and it is found that when the slopes of two adjacent cells are equal, then the first row of the mass matrix contains only zero elements. This type of degeneracy is called parallelism, that is when two cells are collinear, the central node joining the two cells is unnecessary and as a result the matrix is singular when this happens.

The other type of degeneracy, also discussed in [13], is the "shock" type degeneracy which happens when the cell locations in the MFE method become identical. It can be seen by looking at the mass matrix, if two nodes are in the same place then the elements of the mass matrix corresponding to those nodes will be identical. The result then is that two blocks in the mass matrix become identical thus making the matrix rank deficient.

The common approach to avoid the mass matrix becoming ill-conditioned is to add regularization terms. See [1] and [3] for regularization of GWMFE in one dimension, and [10] for SGWMFE. The terms added are much smaller than the error tolerance used to solve the discretized ODE system, however the terms are of the same form as the terms in the mass matrix so that when there is a degenerate node the regularization terms in that row dominate so as to make the system nonsingular.

The regularization coefficients we use, added to the diagonal terms of the mass matrix, are of the form C/x_i or $C/\sqrt{1+x_i^2}$, where C is chosen so that it is well below the truncation error, thus not affecting the accuracy of the solutions beyond the tolerance required. For all experiments in this paper a local truncation error tolerance on the residuals was set to 10^{-8}, but not all the regularization parameters used are the same. For the Porous Medium Equation with the Barenblatt solutions no regularization was needed and thus none was used. For the waiting-time solutions the regularization terms used were of the form $5(10)^{-10}/x_i$, and for the viscous Burgers' equation a regularization term of the form $10^{-13}/\sqrt{1+x_i^2}$ was used.

The different regularization forms used are different in this paper due to the codes that were used. At the time the work for this paper was carried out, we used the SGWMFE code developed for [10] for the implementation of the porous medium equation waiting-time solutions. Since that SGWMFE code was used, the regularization terms for that code which are of the form C/x_i were left in that form. In the case of the implementation for Burgurs' equation we adjusted the GWMFE code developed for [1] since the SGWMFE is equivalent to the GWMFE method for one PDE in one spatial dimension. Since the GWMFE code was used, the regularization terms implemented in that code, which were of the form $C/\sqrt{1+x_i^2}$, are used for the implementation of Burgurs' equation. In the case of the Barenblatt solutions to the porous medium equation, no regularization terms were required to obtain the numerical results at the set error tolerance, and so no regularization was used.

2.7 Summary

The equations for SGWMFE/GWMFE were presented in this section with a projection matrix. An advantage that has been identified previously is that the equations resulting from the formulation using this projection matrix make a more elegant extension to larger numbers of equations than is the case for the original formulation of the GWMFE method, though both approaches naturally reduce to the same set of equations for scalar PDEs.

3 Moving Mesh Partial Differential Equations

3.1 The equidistribution principle and MMPDEs in one dimension

In one space dimension, good grids can be constructed using the equidistribution principle. Let $x = x(\xi)$ be a strictly increasing map from the *computational domain* $[0, 1]$ onto the *physical domain* $[a, b]$. It equidistributes the monitor function $g = g(x) > 0$ if for every $\xi \in [0, 1]$

$$\int_a^{x(\xi)} g(s)\,\mathrm{d}s = \xi \int_a^b g(s)\,\mathrm{d}s. \tag{17}$$

Differentiation of (17) twice with respect to ξ gives the equivalent formulation,

$$(g\,x_\xi)_\xi = 0, \quad x(0) = a, x(1) = b. \tag{18}$$

If the monitor function g is some measure of the local computational effort required and x equidistributes g, then more grid points are concentrated where needed. As a standing assumption, we let $g = g(x, t)$ be continuous on the space-time domain $[a, b] \times [0, T]$, strictly positive with $g_0 = \min_{x,t} g(x, t)$, and $\int g\,\mathrm{d}x = 1$.

By solving the *physical* PDE and (18) simultaneously at every time step, the equidistribution principle can be used to generate a moving mesh. This solution process would be relatively expensive and the mesh obtained unsmooth which, apart from requiring small time steps, can lead to a deterioration in the convergence rate. Several authors (see [14, 15] and references therein) introduced relaxations of this process by introducing mesh speed in different ways. A very general approach which is also easily generalized to

higher dimensions was introduced in [16]. By moving the mesh in the steepest descent direction of a *mesh functional*, parabolic *moving mesh partial differential equations*(MMPDEs) are obtained which can provide an efficient and stable moving mesh and a reliable moving mesh method.

With the right choice of parameters, one obtains the MMPDE

$$x_t = \frac{1}{\tau}(g x_\xi)_\xi, \quad x(0) = a, x(1) = b,$$

where $\tau > 0$ can be seen as a time scale or a relaxation parameter. In [14], this MMPDE is labeled MMPDE5. It is shown in [17] that its solution approximately equidistributes the monitor function in a sense that is made precise.

3.2 Monitor functions based on geometric properties

So far we have only assumed that we have a monitor function which somehow measures the local computational effort required. We now introduce several choices, which have been successfully used in the past and also a few new ideas. Two general classes are considered. In this section, we introduce monitor functions based on geometric properties of the solution like the arc length or the curvature. We call these *geometric* monitor functions.

For better readability, in the following presentation the scaling $\int_0^1 g\,dx = 1$ is not included. Furthermore, if a monitor function consists of a convex combination of two other monitors g_1 and g_2, i.e., $g = \alpha g_1 + (1-\alpha)g_2$, then we implicitly assume that they are scaled to satisfy $\int g_i \,\mathrm{d}x = 1$. For example, we write $g = \alpha e(u) + (1-\alpha) f(u)$ instead of $g = \alpha e(u)/\int e(u)\,\mathrm{d}x + (1-\alpha) f(u)/\int f(u)\,\mathrm{d}x$.

One of the most popular monitor functions is the arc length,

$$g_{\mathrm{AL}} = \sqrt{1 + u_x^2}.$$

Its main characteristic is its robustness, its wide applicability and interpolation error constants which are likely to be independent of perturbation parameters or similar. These constants usually hold, however, only for first order convergence.

Intuitively, when using piecewise linear splines for the approximation space, the first choice would be a monitor function based on the second derivative. Although great accuracy can be achieved, equally great care has

to be taken as Blom and Verwer [18] show in numerous experiments. In [17], a monitor function is introduced, which is constructed as a combination of the arc length and jumps in the gradient, which are closely related to the second derivative. Let the linear spline u have values u_i at grid points x_i then the *jump* monitor function is defined as the linear spline of

$$g_{\mathrm{JMP},i} = \alpha\, g_{\mathrm{AL}}(x_i) + (1-\alpha)\left|[u_x]_{x=x_i}\right|$$

where $[u_x]$ denotes the jump in the gradient. Small choices of α turned out to produce similar problems as those observed in [18]. A good value for most problems is $\alpha = 0.7$.

Finally, let us also note that for equations like the porous medium equation, monitor functions can be constructed which take into account specific features of the equation, say, the conservation of mass. For an example see [6].

3.3 Basic Implementation of the MMPDE method

The implementation of a moving mesh method based on the MMPDEs described in Section 3.1, and in [17], involves solving a coupled system of partial differential equations of which at least one is nonlinear, as is the coupling between the two.

In this section, an implementation of a finite element moving mesh method based on a decoupling of the physical and the moving mesh PDEs is described. For the physical PDE, only second order parabolic problems with Dirichlet boundary conditions are considered.

The physical model problem, defined on a physical domain Ω_T with physical boundary $\partial\Omega$, which encompasses all examples considered in this paper is

$$\begin{aligned} u_t - \nabla\cdot\mathbf{F}(u) - \nabla\cdot(A(u,x,t)\nabla u) &= f(t) \quad \text{in } \Omega_T, \\ u &= u_0(x) \quad \text{at } t=0, \\ u &= u_1(t) \quad \text{on } \partial\Omega. \end{aligned}$$

3.3.1 The Lagrangian formulation

Following the notation in [17], we define the following functional B in order to simplify the notation for the semi-discrete finite element method defined later in equation (19):

$$B(u,t;v,w) = \int_\Omega \mathbf{F}'(u)\cdot\nabla v + (\nabla v)^\top A(u,x,t)\nabla w \,\mathrm{d}x$$

which is bilinear in (v,w). Assume that for every $t \in [0,T]$ we are given a grid with grid points $(x_i(t); i = 0,\ldots,M)$ and let $(\Phi_i(t); i = 0,\ldots,M)$ be the associated nodal basis of piecewise linear functions. Suppose that the first $\tilde{M}$ nodes are the interior nodes. Then the finite element test and solution spaces are respectively defined as

$$\begin{aligned} V_h(t) &= \operatorname{span}\{\Phi_i(t) : i = 0,\ldots,\tilde{M}\}, \\ S_h(t) &= \sum_{j=\tilde{M}+1}^{M} u_1(x_j,t)\Phi_j(t) + V_h(t), \end{aligned}$$

and the semi-discrete finite element method reads: *For all* $0 < t \le T$ *find* $u_h(t) \in S_h(t)$ *such that for all* $\varphi \in V_h(t)$,

$$(u_{h,t},\varphi)_{L^2(\Omega)} + B(u_h,t;u_h,\varphi) = (f,\varphi)_{L^2(\Omega)}. \tag{19}$$

Consider for a moment the time discretization of (19) by the implicit Euler method. Assume a partition $0 = t_0 < t_1 < \cdots < t_N = T$ of the time-interval $[0,T]$ is given, and set $k_n = t_n - t_{n-1}$. For $f \equiv 0$ the implicit Euler method reads

$$(u_h(t_n),\varphi)_{L^2(\Omega)} + k_n B(u_h(t_n),t_n;u_h(t_n),\varphi) = (u_h(t_{n-1}),\varphi)_{L^2(\Omega)},$$

for all $\varphi \in V_h(t_n)$. If the mesh is not constant (e.g. moving), then the term $(u_h(t_{n-1}),\varphi)$ on the right hand side has to be calculated by a projection of $u_h(t_{n-1})$ onto the space $S_h(t_n)$ with the new mesh.

An alternative way, a discrete equivalent of the so-called Lagrangian formulation of the PDE,

$$u_t - L(t)u = \frac{\mathrm{d}u}{\mathrm{d}t} - (\nabla u)^\top x_t - L(t)u = f(t),$$

is far more efficient even in the one-dimensional case. Here, $\mathrm{d}u/\mathrm{d}t$ stands for the derivative of $u(x(\xi,t),t)$ with respect to t. This form of the PDE is analyzed in greater generality in [7] and [19].

Based on this formulation, we can write an alternative formulation of the semi-discrete Galerkin finite element method (19). Equation (19) is equivalent to

$$\sum_i u_i' \left(\Phi_i, \varphi\right)_{L^2(\Omega)} - \left(\frac{\partial x}{\partial t} \cdot \nabla u_h, \varphi\right)_{L^2(\Omega)} + B(t, u_h; u_h, \varphi) = (f, \varphi)_{L^2(\Omega)}. \quad (20)$$

The form (20) of the semi-discrete finite element method can be easily discretized in time by any ODE solver. One possible choice is described in Section 3.3.2.

3.3.2 Discretization in time

To enable adaptive time-stepping for the physical PDE, we use the second order *singly diagonally implicit* Runge-Kutta (SDIRK2) method. This method was proposed for moving mesh equations in [20] and [21]. Details about the derivation and stability properties can be found in [22].

Suppose the SDIRK2 method is employed to integrate the system

$$\dot{u} = f(t, u),$$

where $f : \mathbf{R} \times \mathbf{R}^m \to \mathbf{R}^m$ on the grid $t_0 < t_1 < t_2 < \ldots$. Then, with $k_n = t_n - t_{n-1}$ and $\gamma = (2 - \sqrt{2})/2$, the method is given by

$$\begin{aligned} v_1 &= f(t_{n-1} + \gamma k_n, u(t_{n-1}) + \gamma k_n v_1), \\ v_2 &= f(t_{n-1} + k_n, u(t_{n-1}) + (1-\gamma) k_n v_1 + \gamma k_n v_2), \\ u(t_n) &= u(t_{n-1}) + k_n((1-\gamma) v_1 + \gamma v_2). \end{aligned} \quad (21)$$

To obtain a local estimate of the error, the second-order SDIRK2 scheme can be combined with an appropriate first-order scheme. To maximize computational efficiency, we use

$$\hat{u}(t_n) = u(t_{n-1}) + k_n v_1,$$

where v_1 is that calculated in (21). For details about the time step control, see [20] or [17]. In this section, we briefly review a method of decoupling the physical from the moving mesh equations which reduces the effort for solving the MMPDE significantly.

For evolving the mesh, we use the implicit Euler method in time. For evolving the physical PDE, we use the SDIRK2 scheme (21). Suppose we

have computed $\mathbf{x}(t_{n-1})$ and $u_h(t_{n-1})$. We use $u_h(t_{n-1})$ as a first approximation for $u_h(t_n)$ to compute an approximation of the mesh $\mathbf{x}(t_n)$. We then use this approximation to the new mesh to compute an approximation of $u_h(t_n)$. This procedure is iterated as often as necessary to improve the approximations for $\mathbf{x}(t_n)$ and $u_h(t_n)$.

The arising nonlinear systems are solved either by a Newton method or by a fixed point iteration. If the nonlinear iteration does not converge, the stepsize is decreased until it is successful.

3.4 Artificial smoothing of the monitor functions

The necessity of spatial smoothing is discussed in [15] in great detail. Smoothing the monitor function, e.g. by some local averaging procedure, can greatly improve performance and even accuracy. The reason is mainly that the iterations converge faster so that bigger time-steps can be taken, while at the same time a slightly displaced mesh will only insignificantly decrease accuracy. In fact, a smoother mesh might even improve it. Spatial smoothing has been fully studied and we shall not discuss it further. Apart from explaining analytically why spatial smoothing is important, the analysis in [17] suggests that smoothness in time is just as important and might bring additional stability and performance. We impose this in two ways.

- At every time-step we take a weighted average between the computed monitor function and that from the last time-step, i.e., we use $g = \mathrm{TMP\_SM}\, g_{\mathrm{old}} + (1 - \mathrm{TMP\_SM}) g_{\mathrm{new}}$.
- Becket et al. [20] suggest using 4 mesh iterations in the alternating solution procedure. Instead we use 8 relaxed iterations, i.e., we choose a relaxation parameter $\mathrm{MRELX} \in (0, 1]$ and take $x_{\mathrm{new}} = x_{\mathrm{old}} + \mathrm{MRELX} \times d$ if d is the usual iteration step.

An intensive benchmark was carried out, producing solutions for several monitor functions and a wide variety of choices of MRELX and TMP_SM. For a very large range of smoothing parameter choices, the error changes on such a small scale that we can practically choose the parameters solely based on performance considerations. For all of the experiments in this work, we use: $\mathrm{TMP\_SM} = 0.3$, $\mathrm{MRELX} = 0.6$.

3.5 Modifications to the MMPDE method for the solution of the porous medium equation

The first modification is the implementation of the boundary movement. To achieve this, we simply add a subroutine to the code which determines the interface movement by approximating equation (23), which is defined in the following section, by the trapezium rule method. This is done every time before a mesh iteration in the alternating solution procedure. The inner derivatives are determined by a linear extrapolation method. The extrapolation values are evaluated at the element centers.

For a simple solution, such as the self-similar test solution in closed form, we could simply use any of the monitor functions presented in Section 3.2. To be able to resolve the boundary movement, especially when it should be zero, we modify the *jump* monitor function. We define

$$g_{\mathrm{PM}} = 0.6 \times g_{\mathrm{JMP}} + 0.4 \times \left(\frac{x - (s_+ + s_-)/2}{s_+ - s_-} \right)^6 \frac{\max |u_x|^2}{1/M + |u_x|^2}.$$

Again, some additional scaling procedures are not considered in this definition. The *PM* monitor function has no specific interpretation. It is constructed to create a strong concentration of grid points at the boundary if the solution should be flat there, compared to other parts of the domain.

The two modifications described so far are sufficient to solve for *easy* solutions, like the Barenblatt solutions defined later in equation (4.2.2), or the waiting-time solution with zero initial interface speed.

4 Model problems

4.1 The viscous Burgers' equation

We consider the viscous Burgers' equation for the scalar unknown variable u, in one dimensional space, as follows:

$$u_t + uu_x - \nu u_{xx} = 0, \quad \text{in } [0,1] \times [0,T],$$

where the spatial variable $x \in [0,1]$, and the time variable $t \in [0,T]$. T is a final time of computation chosen between 0 and 2, and ν is the viscosity coefficient which is chosen to be $\nu = 10^{-3}$ in all the simulations carried out in this paper.

The boundary conditions are:

$$\begin{aligned} u(0,t) &= a(t), \\ u(1,t) &= b(t), \end{aligned}$$

and the initial condition is:

$$u(x,0) = u_0(x).$$

For the simulations carried out in this paper we consider the case where $a(t) = b(t) = 0$ and

$$u_0(x) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x).$$

4.2 The porous medium equation

We consider the following special case of the porous medium equation for the scalar unknown variable u, in one dimensional space, as follows:

$$u_t = (uu_x)_x \quad \text{in } \mathbf{R} \times [0,T], \tag{22}$$

where the spatial variable $x \in \mathbf{R}$, and the time variable $t \in [0,T]$. T is a final time of computation (chosen between 0 and 10 for the computations in this paper), and the initial condition is:

$$u(x,0) = u_0(x) \geq 0,$$

where u_0 has compact support. It arises as a model for many physical phenomena, such as the spreading of a thin film of liquid under gravity or the percolation of gas through a porous medium. For further information see [23] and references therein. Here, we summarize those results which are used in this paper.

The problems modelled in this paper using the porous medium equation are all moving boundary problems on an interval $(s_-(t), s_+(t))$, where $x = s_-(t)$ is the left boundary coordinate as a function of time, and $x = s_+(t)$ is the right boundary coordinate as a function of time. In each benchmark case of the porous medium equation which is to be solved, we impose the following boundary conditions:

$$\begin{aligned} u(s_-(t),t) &= 0, \\ u(s_+(t),t) &= 0. \end{aligned}$$

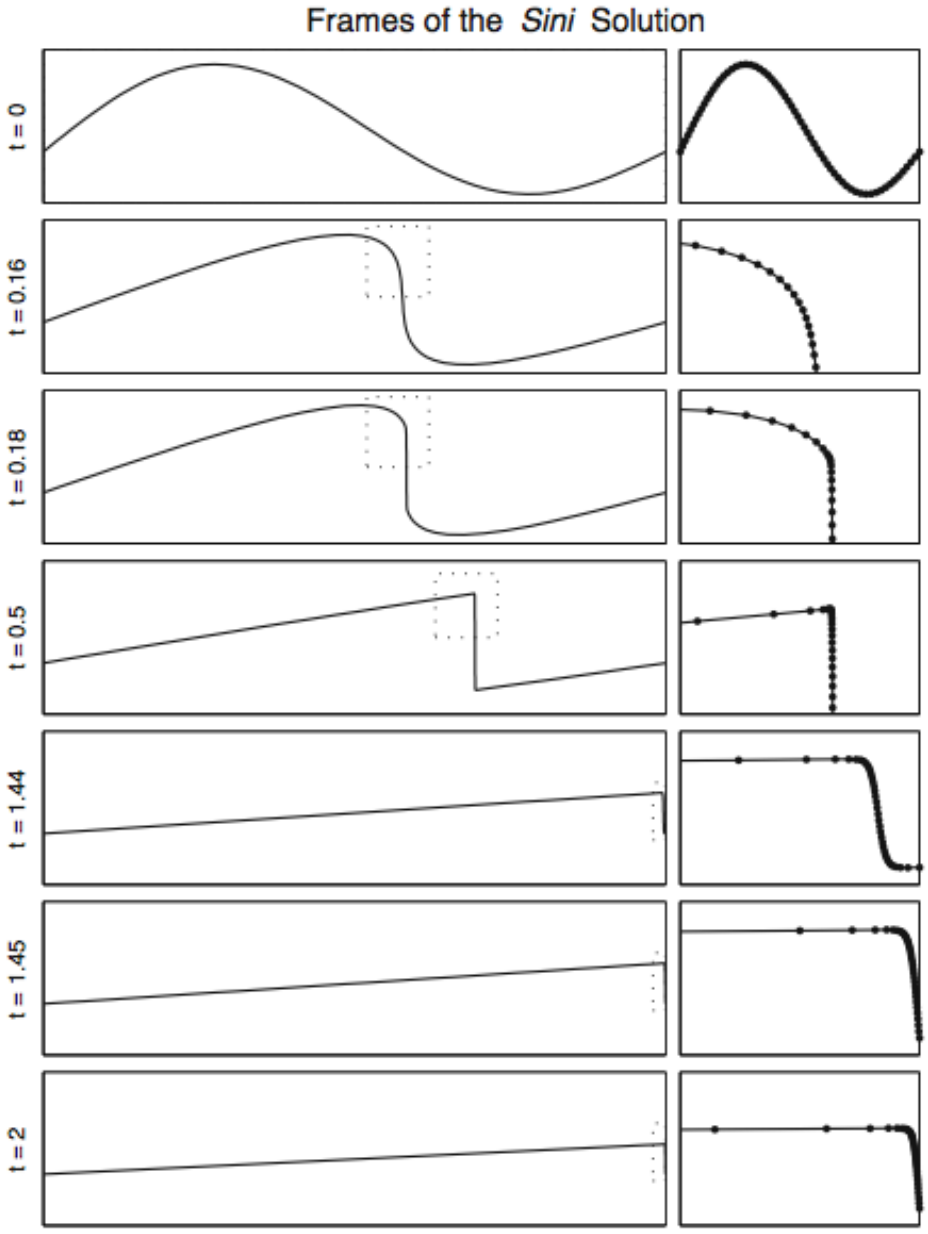

Figure 2: Frames at different times of an MMPDE numerical solution using the arc length monitor function. The solutions are of the viscous Burgers' equation with $\nu = 10^{-3}$. The figures to the right show the zoomed in plot, of the solution inside the dotted line boxes, highlighted in the corresponding figures to the left.

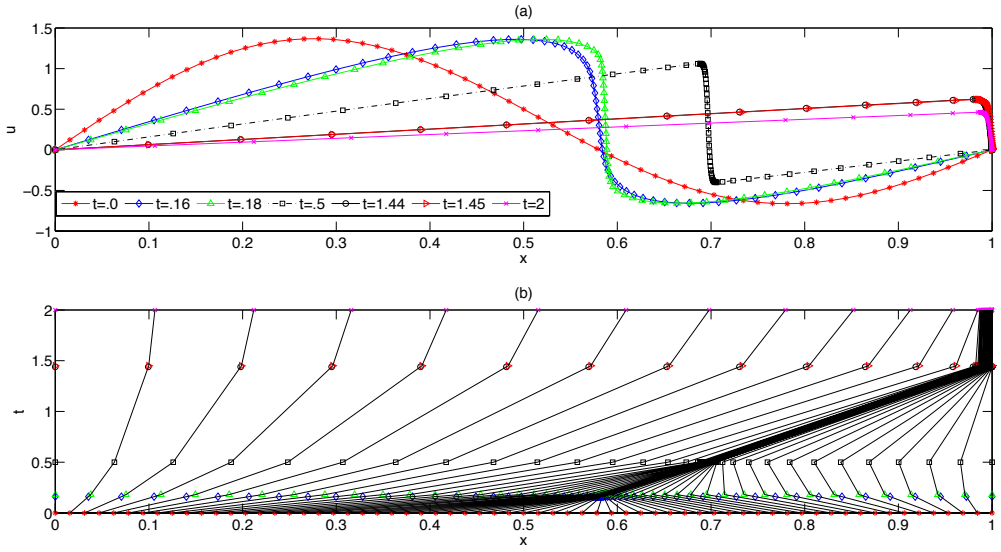

Figure 3: (a) SGWMFE numerical solutions of the viscous Burgers' equation with $\nu = 10^{-3}$. Solutions are shown at the times corresponding to the MMPDE solution time frames shown in Figure 2. (b) SGWMFE mesh evolution plots are shown as a function of time, at the corresponding solutions times shown in (a).

4.2.1 Boundary velocity solution

The porous medium equation (22) has a unique weak solution, i.e., a solution in the distributional sense. If the support of the initial condition u_0 is compact then the support of $u(\cdot, t)$ is compact for all t. If the solution u is positive in the interval $(s_-(t), s_+(t))$ and zero outside, then the interfaces $s_\pm$ move at speeds

$$\frac{\mathrm{d}s_-}{\mathrm{d}t} = -\lim_{x \to s_- +} u_x(x) \quad \text{and} \quad \frac{\mathrm{d}s_+}{\mathrm{d}t} = -\lim_{x \to s_+ -} u_x(x). \tag{23}$$

Note that for the MMPDE method equation (23) is used to determine the boundary movement of the computational domain. For the SGWMFE method, this condition is not necessary as the boundary movement is part of the solution of the SGWMFE method. The SGWMFE methods result in a set of two PDEs at each node, one for the value of u and one for the positions of the nodes in the x-axis. For the SGWMFE method the only condition that is necessary to be applied at the boundary is the fixed boundary condition for u and the nodes are allowed to freely move in the x-axis. An alternative way to determine the interface is to use the fact that the mass $\int_{\mathbf{R}} u(x)\,dx$ and the center of gravity $\int_{\mathbf{R}} xu(x)\,dx$ are conserved by the solution of (22). This approach turned out to be very unstable when used on top of the MMPDE method.

Next, we review some classes of solutions that we will test our methods on.

4.2.2 Barenblatt analytic solutions

The first set of solutions we will test the methods on is a family of similarity solutions, the Barenblatt solutions,

$$u(x,t) = \begin{cases} \frac{1}{6}t^{-1/3}(a^2 - x^2t^{-2/3}), & |x| \le at^{1/3}, \\ 0 & |x| \ge at^{1/3}, \end{cases}$$

where a is a positive constant. This analytic solution is plotted for several times in Figure 4.

4.2.3 Waiting-time solutions

The porous medium equation possesses solutions for which an interface can remain fixed for a finite time and then start moving. The time t^* for which the interface remains stationary is called the waiting-time. Solutions which exhibit such a behaviour are called waiting-time solutions. The following facts are due to [24]. Suppose the initial condition has an interface at $x = x_0$ and the solution is positive to its left. Let $\alpha = \lim_{x\to x_0-} u_0(x)/(x-x_0)^2$, $\beta = \sup_{x<x_0} u_0(x)/(x-x_0)^2$, and let $t_\gamma = 1/(6\gamma)$ for all $\gamma > 0$. Then $t_\beta \le t^* \le t_\alpha$ and if $\beta = \alpha$, we know the exact waiting-time. An example of an initial condition which satisfies this condition is $u_0(x) = \chi_{[-1,1]}(x)\sin(\pi(x+1)/2)^2$. In the case $t_\alpha = t^*$, it is furthermore known that this interface is continuously differentiable, in particular, it starts moving at zero initial speed. We call this solution the *first waiting-time solution*. Several frames of a numerical solution (using the MMPDE method) are plotted in Figure 5.

5 Numerical results

5.1 Computational meshes and computing errors in the numerical solutions

5.1.1 The computational meshes for the viscous Burgurs' equation

The simulations carried out with the MMPDE and SGWMFE methods for this problem are all initialized with the same initial mesh. The simulations

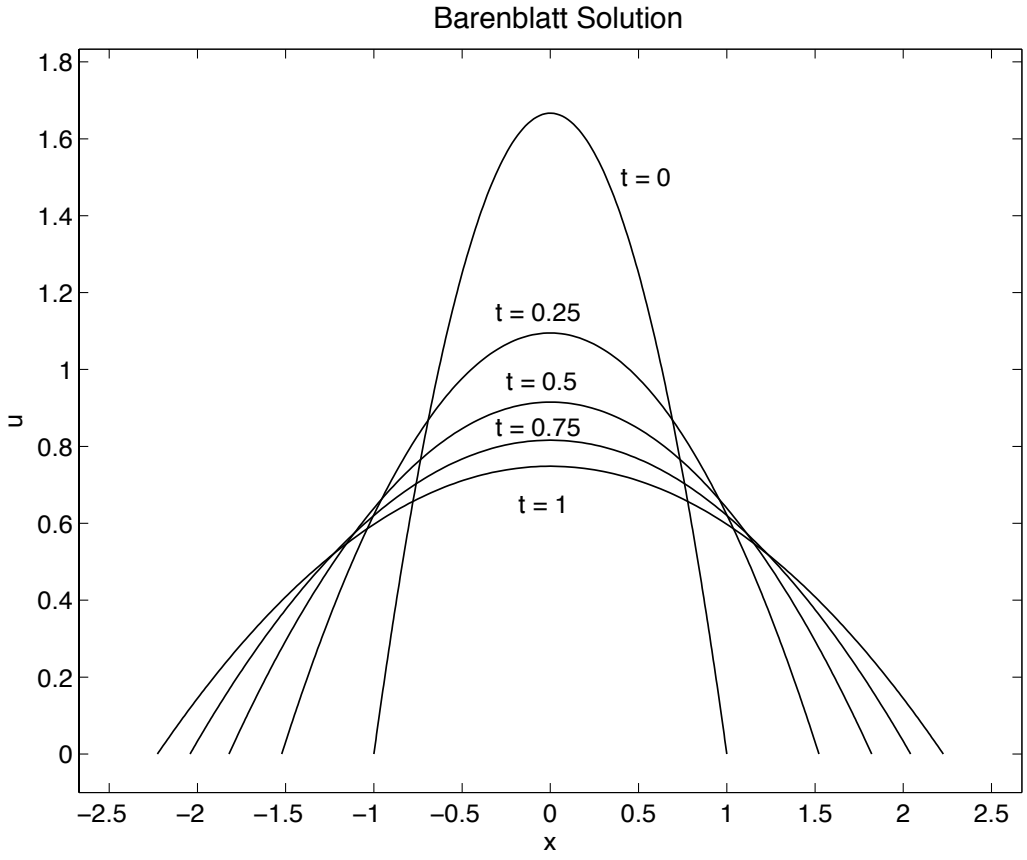

Figure 4: Analytics self-similar Barenblatt solutions of the porous medium equation with time shifted by 0.1, and $a = (0.1)^{-1/3}$.

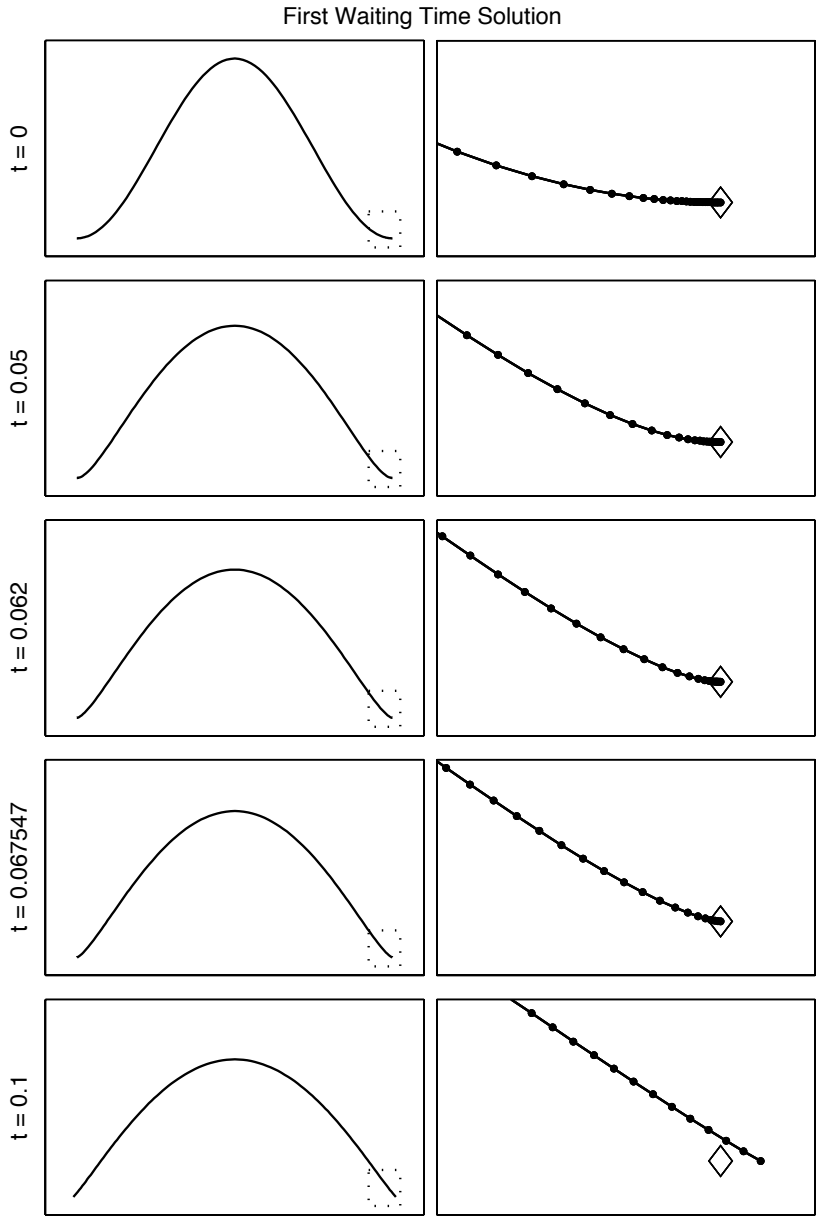

Figure 5: Frames at different times of an MMPDE numerical solution using the arc length monitor function. The solutions are of the first waiting-time solution of the porous medium equation. The figures to the right show the zoomed in plot, of the solution inside the dotted line boxes, highlighted in the corresponding figures to the left. The initial solution is zero velocity, and so the diamond in the zoomed figures is used as a reference position which highlights when the boundary is still not moving, and then when it has moved by the time t = 0.1.

are carried out for each method for a varying number of grid sizes with 17, 33, 65, 129 and 257 grid points. In addition to these simulations, a further simulation of this problem is carried out with the MMPDE method on a mesh with 513 grid points as a reference solution for computing the grid convergence error, since an analytic solution to this problem is not available.

This benchmark problem was chosen as an example which does not require a well-adapted initial mesh. The initial meshes for this benchmark problem is one in which the nodes are equally spaced over the physical domain $x = [0, 1]$. As a sample of how the solutions look for this problem see Figure 2 for solutions using an MMPDE method and see Figure 3 for solutions using the SGWMFE method.

5.1.2 The computational meshes for the porous medium equation

For both porous medium equation benchmark problems, the simulations are carried out for each method for a varying number of grid sizes with 17, 33, 65, 129 and 257 grid points. In the case of the waiting-time solution benchmark problem, a further simulation is carried out with the MMPDE method on a mesh with 513 grid points as a reference solution for computing the grid convergence error, since an analytic solution to that problem is not available.

For computing solutions to the Barenblatt self-similar initial conditions, we use an initial physical domain $x = [-0.5, 0.5]$, and initial condition $u_0(x) = 1 - 4x^2$. For the SGWMFE method applied to this problem we use an initial mesh in which the nodes are equally spaced over the initial physical domain. For the MMPDE method, the method itself chooses it's initial node locations based on equidistributing the nodes over the initial physical domain.

For both the SGWMFE and the MMPDE methods applied to the waiting-time solution benchmark problem for the porous medium equation, we use an initial mesh which is equidistributed using the MMPDE initial mesh equidistribution over the physical domain $x = [-1, 1]$.

5.1.3 Calculating the errors in the moving mesh methods

For each of the benchmark problems we calculate an error in each method either as compared to an analytic solution or as compared to a numerical solution on a much finer mesh . For Burgers' equation since no analytic solution is available, we compare our numerical solutions to the numerical

solutions produced by the MMPDE method, on a finer mesh with 513 grid points, as a reference solution. For the porous medium equation in the case of the Barenblatt problem, we directly compute the errors in our numerical solutions compared to the Barenblatt self-similar analytic solution. In the case of the waiting-time solution benchmark problem, we compare our numerical solutions to the numerical solutions produced by the MMPDE method, on a finer mesh with 513 grid points, as a reference solution.

5.1.4 Labels used in the error plots in the figures

The notation used in the solution figures is as follows: 'sgwmfe' denotes computations with the string gradient weighted moving finite element method (see Section 2).

Solutions computed by the MMPDE method (see Section 3) are denoted by the name of the monitor function used: 'arclen' for the arc length monitor function, 'jmp' for the second derivative-based monitor and 'pmmon' for the monitor which was specifically constructed for the porous medium equation. In all cases with these three labels, we enforce the boundary condition where the interface is restricted to move on away from the mass as in equation (23).

For the benchmark problem of solving the waiting-time solutions, we only use the 'pmmon' monitor function. For the results in Section 5.4, the notation for MMPDE computations depends on whether the interface is completely free ('mmpde') or where the interface is restricted to move on away from the mass ('mmpde, eb').

5.2 Benchmark 1: Viscous Burgers' equation with $\nu = 10^{-3}$

In our first benchmark problem, we solve the viscous Burgers' equation, described in Section 4.1. Frames of the solutions using the MMPDE method are shown in Figure 2. Solutions using the SGWMFE method are shown in Figure 3, with accompanying plots of the grid point positions as a function of time. We see from Figure 6 that the MMPDE method and the SGWMFE method are comparable in the order of accuracy of the grid convergence error, relative to the number of nodes used. Both methods show a nearly second order grid convergence error rate, where the infinity norm is taken over all integration time steps. In Figure 7 we see the grid convergence error plots when taking the infinity norm over all the nodes but only sampled at the

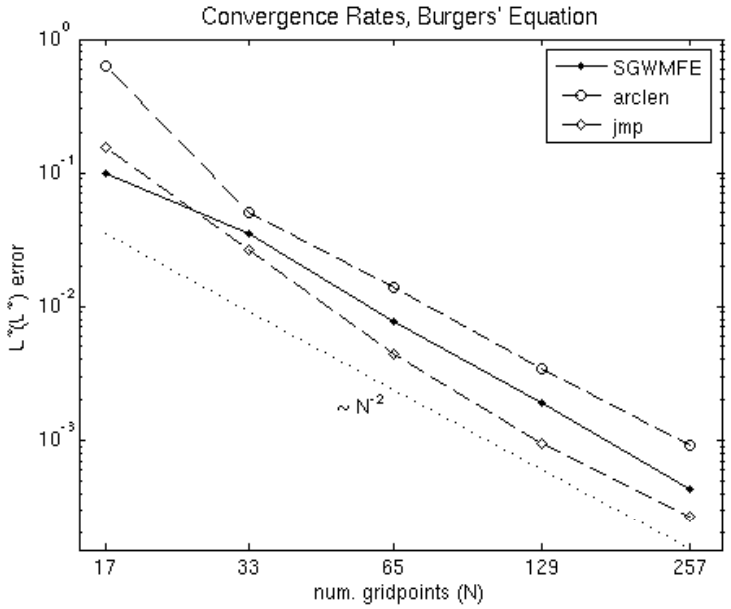

Figure 6: Grid convergence error plots in the $L^\infty(L^\infty)$ error norm of SGWMFE and MMPDE (with the arc length (arclen) and jump (jmp) monitor functions) numerical solutions to the viscous Burgers' equation with $\nu = 10^{-3}$. To compute these errors in the absence of an analytic solution, the solutions are compared to an MMPDE solution on a fine grid using 513 grid points. The infinity norms are evaluated over all nodes, and over all the time steps taken.

particular time: $t = 0.2$. In this figure we see that similar slopes for the convergence rates for both methods are observed (nearly second order), though it is clear that the 'jump' monitor function produces more accurate results than the arc length monitor function solutions. For this benchmark problem one can see that the MMPDE method with the jump monitor function produces more accurate results overall, than the arc length monitor function and the SGWMFE method, despite the SGWMFE method shows a small advantage when fewer mesh nodes are used.

5.3 Benchmark 2: Barenblatt solutions of the porous medium equation

Figure 8 displays error graphs in the numerical solutions of SGWMFE and MMPDE as compared to the Barenblatt analytic solutions over a short time integration domain with a final computational time $T = 0.1$. Figure 9 displays error graphs in the numerical solutions of SGWMFE and MMPDE as

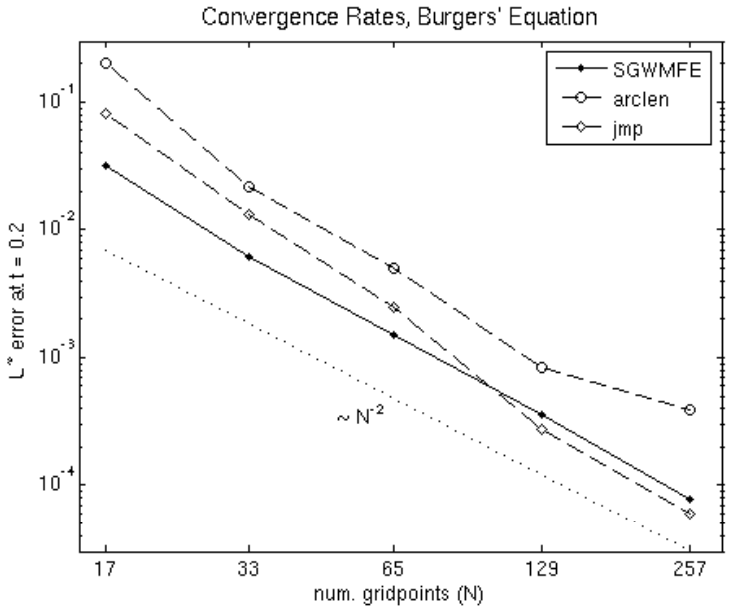

Figure 7: Grid convergence error plots in the L^∞ error norm at a fixed time $t = 0.2$, of SGWMFE and MMPDE (with the arc length (arclen) and jump (jmp) monitor functions) numerical solutions to the viscous Burgers' equation with $\nu = 10^{-3}$. To compute these errors in the absence of an analytic solution, the solutions are compared to an MMPDE solution on a fine grid using 513 grid points. The infinity norms are evaluated over all nodes.

compared to the Barenblatt analytic solutions over a long time integration domain with a final computational time $T = 10$. The errors are plotted relative to the number of nodes used. The solutions of both of the monitor functions implemented with the MMPDE method, as well as the solutions using the SGWMFE method, behave as expected (nearly second order rate of convergence) for the smooth Barenblatt similarity solution. Comparable results (for the long time integration domain) were also obtained in [6] for their moving mesh method. For the errors sampled at a short time integration domain, the MMPDE method with the jump monitor function, is the most accurate overall. However, over a longer time integration domain the SGWMFE method is more accurate, particularly when fewer mesh nodes are used. There is a crossover in the error plots when the number of nodes used is 129, when the MMPDE method, with any of the three monitor functions tested, are all more accurate than the SGWMFE method. Between the MMPDE monitor functions, the pmmon monitor function constructed for the porous medium equation is the most accurate over longer time integration domains.

5.4 Benchmark 3: A waiting-time solution of the porous medium equation

In this benchmark, we examine whether the two numerical schemes are capable of reproducing the waiting-time solution described in Section 4.2. Here, we are interested in the correct behaviour of the interface.

Due to the lack of an analytical solution, we compare with a numerical solution, using the MMPDE approach with 513 grid points. We use the short labels `sgwmfe`, `mmpde-eb` and `mmpde` to respectively denote the numerical solutions using the string gradient weighted moving finite element method, the MMPDE method with enforced positive interface speed and the pure MMPDE method. Figure 10 shows plots of the interfaces for the waiting-time solution and different mesh sizes. The grid convergence error rates of the approximations of the interfaces are plotted in Figure 11.

The results in Figures 10 show a number of things. One can observe that clearly SGWMFE does not keep its boundary node on the point $x = 1$ as does the enforced boundary implemented in the MMPDE method, this is because the boundary nodes in the SGWMFE method were not enforced to only move in one direction. This however does not imply a poor solution at

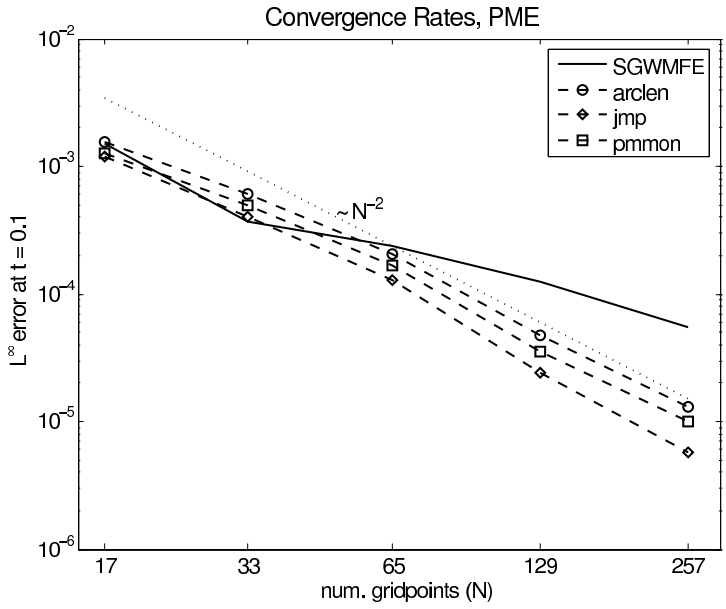

Figure 8: Convergence rates in the L^∞ error norm at a fixed time $t = 0.1$, of SGWMFE and MMPDE (with the arc length (arclen), jump (jmp) and the modified jump (pmmon, from Section 3.5) monitor functions) numerical solutions to the porous medium equation. To compute these errors the analytic Barenblatt self-similar solutions are used. The infinity norms are evaluated over all nodes.

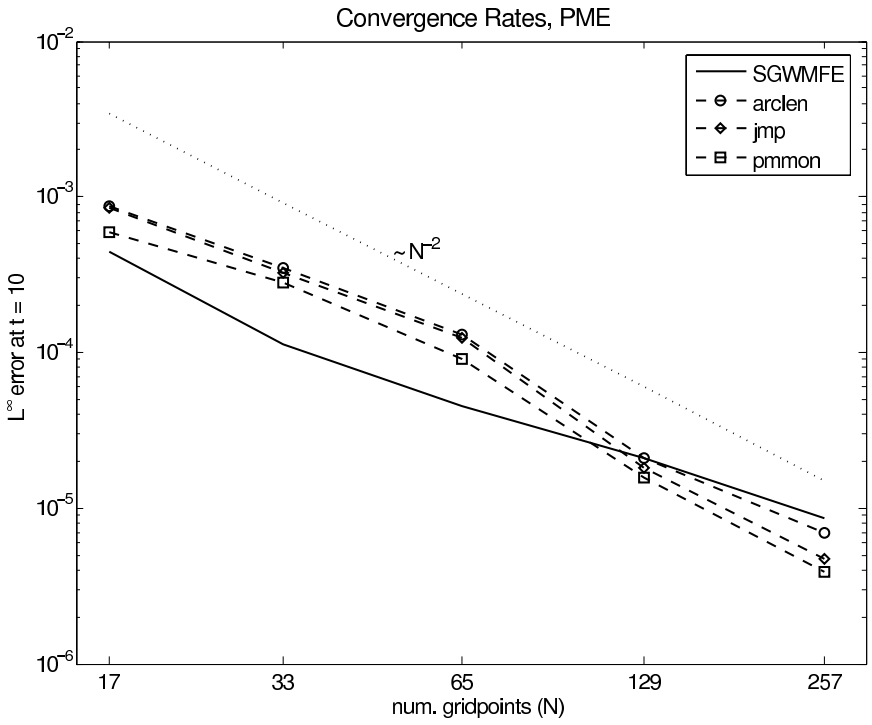

Figure 9: Convergence rates in the L^∞ error norm at the fixed time $t = 10$, of SGWMFE and MMPDE (with the arc length (arclen), jump (jmp) and the modified jump (pmmon, from Section 3.5) monitor functions) numerical solutions to the porous medium equation. To compute these errors the analytic Barenblatt self-similar solutions are used. The infinity norms are evaluated over all nodes.

the boundary since the solution is zero beyond the location of the boundary point until the boundary begins to move. If you look very closely, you will see that MMPDE without the enforced boundary also does this to a much smaller extent, see the plot with 33 nodes in Figure 10.

From Figure 10 one can see that SGWMFE predicts the critical waiting-time later than does the MMPDE method and the analytic solution. This problem could possibly be removed, as with the MMPDE method, by enforcing the boundary node to only move in one direction. For the results shown in this paper this was not applied, so regardless of the initial positions you can see that, as time progresses, the solutions from SGWMFE are more accurate than the MMPDE method once the boundary begins to move (beyond the waiting-time). It is clear that for this start up problem an enforced velocity boundary condition on the MMPDE method is much better at predicting the waiting-time solution, given that one uses a sufficient number of nodes.

If we now focus our attention on Figure 11 where we plot the convergence of a) the maximum error of the interface position over the integration steps for $t \in [0, 0.5]$, b) the error of the interface position at a particular time $t = 0.5$, and c) the mass error at a particular time $t = 0.5$. We see here that the SGWMFE method is more accurate than the MMPDE method within the waiting-time interval, even though the actual waiting-time is predicted more accurately with the MMPDE method. The convergence rate of the maximum error of the interface position for the MMPDE methods is of order 1 and for the SGWMFE is of order 0, and similarly for the error in the interface position at the particular time $t = 0.5$. Finally for the grid convergence error in the mass, calculated a the particular time $t = 0.5$, the SGWMFE converges at a second order rate and the MMPDE method converges at a first order rate.

It should be noted that for this start up problem the time steps were chosen initially for both methods. Generally the time steps for SGWMFE are chosen dynamically, though for the MMPDE method implemented here the time steps are chosen initially since the interest here was on the waiting-time solutions, and for comparison purposes the same time steps had to be imposed for the SGWMFE as well. Because of this, smaller time steps were chosen and the solutions from the SGWMFE method are sometimes more accurate than would have been required by the error tolerance set for the SGWMFE method.

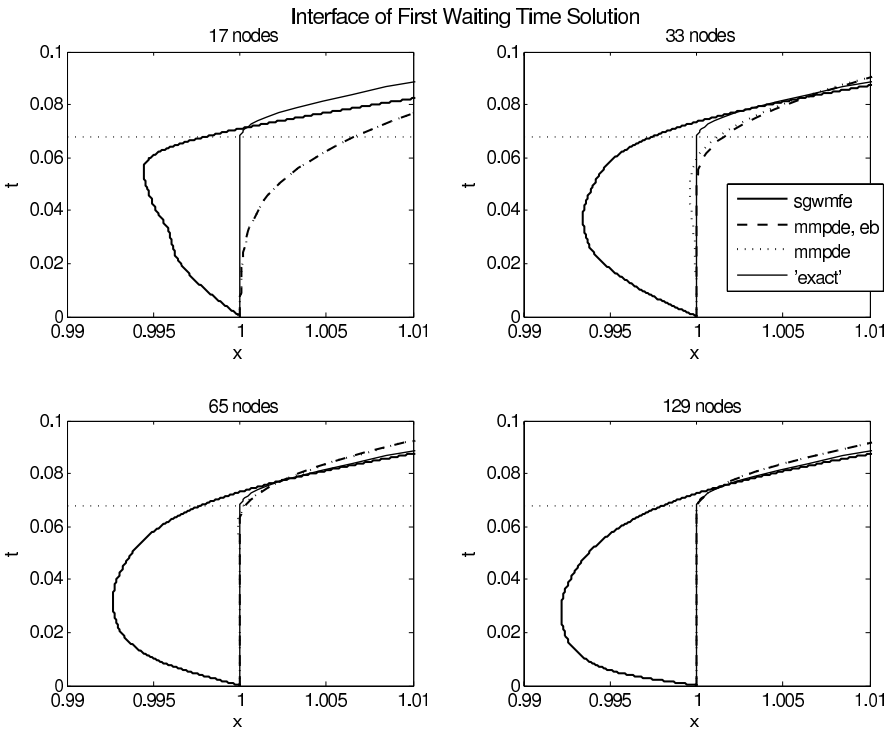

Figure 10: Right boundary interface of the waiting-time solution of the porous medium equation, using the numerical schemes MMPDE and SG-WMFE. The MMPDE method in both cases uses the 'pmon' monitor function discussed in Section 3.5. The single case labelled 'mmpde,eb' enforces the boundary condition from equation (23).

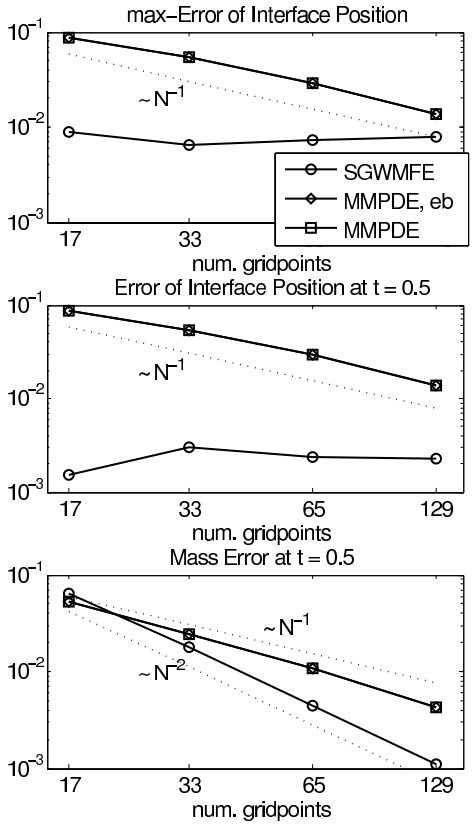

Figure 11: The top figure shows the grid convergence error rates for the maximum error for $0 \leq t \leq 1/2$ of the right interface boundary approximations of the waiting-time solution. The middle figure shows the error at the time $t = 1/2$ of the right interface boundary approximations of the waiting-time solution. The bottom figure shows the error in the mass integral of the solution between the two boundaries at the time $t = 1/2$.

6 Summary

From the numerical results shown in this paper we conclude that for the viscous Burgers' equation the SGWMFE/GWMFE and the MMPDE methods produce comparable results. Depending on the monitor function used for the MMPDE to solve the viscous Burgers' equation, the MMPDE can produce results that are slightly better or worse than those of SGWMFE/GWMFE. For both porous medium equation examples studied in this paper it is observed that the longer time results obtained using SGWMFE/GWMFE are more accurate than those produced using the MMPDE method. However for the startup problem of obtaining the critical time $t*$ (the waiting-time) the MMPDE method predicts this more accurately.

It should be noted that a lot of work on MMPDE moving mesh methods has been carried out over the last decade, for example see the book by [25] for an extensive review of adaptive moving mesh methods. As a follow up paper to this article we are interested in further studies comparing SGWMFE/GWMFE methods with MMPDE type methods using different monitor functions, which could prove to be even more accurate. For example, it would be worthwhile considering monitor functions such as those found in [26] where the authors use monitor functions which are 'defined based on asymptotic estimates of interpolation error obtained using the interpolation theory of finite element methods'.

A clear advantage of the SGWMFE/GWMFE is that it can solve the moving boundary and moving shock problems blindly, that is with comparable accuracy to the MMPDE method but without the need of any specially constructed function for a particular problem. From the studies carried out in this paper however, we conclude that the MMPDE method has the strong advantage of being a method based on a monitor function which can be constructed for a particular class of problems and geometry. The better constructed the monitor function is, the more accurate results are obtained.

Acknowledgements I would like to thank Christoph Ortner for his efforts and the use of his codes to obtain the MMPDE simulations, as well as for parts of his MSc thesis which contributed to the MMPDE section of this paper. I would also like to thank Prof. Keith Miller for inspiring me to study MFE methods, and also for providing me with many of his invaluable Fortran codes.

References

[1] Carlson N.N., Miller K. Design and application of a gradient-weighted moving finite element code I: in one dimension. *SIAM J. Sci. Comput.* 1998; **19**(3):728–765.

[2] Carlson N.N., Miller K. Design and application of a gradient-weighted moving finite element code II: in two dimensions. *SIAM J. Sci. Comput.* 1998; **19**(3):766–798.

[3] Miller K. A geometrical-mechanical interpretation of gradient-weighted moving finite elements. *SIAM J. Numer. Anal.* 1997; **34**(1):67–90.

[4] Wacher A., Sobey I., Miller K. String gradient weighted moving finite elements for systems of partial differential equations. *Numerical Analysis Group report* (03/15), Computing Laboratory, Oxford, 2003.

[5] Wacher A., Sobey I. String Gradient Weighted Moving Finite Elements in Multiple Dimensions with Applications in Two Dimensions. *SIAM J. Sci. Comp.* 2007; **29** (2): 459-480.

[6] Baines M.J., Hubbard M.E., Jimack P.K. A moving finite element method using monitor functions. *School of Computing report* (2003.04), University of Leeds, 2003.

[7] Jimack P.K., Wathen A.J. Temporal derivatives in the finite-element method on continuously deforming grids. *SIAM J. Numer. Anal.* 1991; **28**(4): 990–1003.

[8] Miller K., Miller R.N. Moving finite elements I. *SIAM J. Numer. Anal.* 1981; **18**(6): 1019–1032.

[9] Miller K. Moving finite elements II. *SIAM J. Numer. Anal.* 1981; **18**(6): 1033–1057.

[10] Wacher A. *String Gradient Weighted Moving Finite Elements for Systems of Partial Differential Equations, DPhil thesis.* Oxford University Computing Laboratory, Oxford, 2004.

[11] Jeffreys H., Jeffreys B.S. *Methods Of Mathematical Physics.* Cambridge University Press, Cambridge, 1946.

[12] Baines M.J. *Moving Finite Elements.* Oxford University Press Inc., New York, 1994.

[13] Wathen A.J., Baines M.J. On the structure of the moving finite-element equations. *IMA J. Numer. Anal* 1985; **5**: 161–182.

[14] Huang W., Ren Y., Russell R.D. Moving mesh partial differential equations (MMPDES) based on the equidistribution principle. *SIAM J. Numer. Anal.* 1994; **31**(3): 709–730.

[15] Huang W., Russell R.D. Analysis of moving mesh partial differential equations with spatial smoothing. *SIAM J. Numer. Anal.* 1997; **34**(3): 1106–1126.

[16] Huang W., Russell R.D. Moving mesh strategy based on a gradient flow equation for two-dimensional problems. *SIAM J. Sci. Comp.* 1999; **20**(3): 998–1015.

[17] Ortner C. *Moving Mesh Partial Differential Equations, MSc thesis.* Oxford University Computing Laboratory, Oxford, 2003.

[18] Blom J.G., Verwer J.G. On the use of the arclength and curvature monitor in a moving grid method which is based on the method of lines. *Tech. Report NM-N8902*, CWI, Amsterdam, 1989.

[19] Cao W., Huang W., Russell R.D. An r-adaptive finite element method based upon moving mesh PDEs. *J. Comp. Phys*, 1999; **149**: 221–244.

[20] Beckett G., Mackenzie J.A., Ramage A., Sloan D.M. On the numerical solution of one-dimensional PDEs using adaptive methods based on equidistribution. *J. Comp. Phys* 2001; **167** (2): 372–392.

[21] Beckett G., Mackenzie J.A., Ramage A., Sloan D.M. Computational solution of two-dimensional unsteady PDEs using moving mesh methods. *J. Comp. Phys*, 2002; **182** (2): 478–495.

[22] Hairer E., Wanner G. *Solving Ordinary Differential Equations II – Stiff And Algebraic Problems.* Springer Series in Computational Mathematics, Springer-Verlag Berlin Heidelberg, 1991.

[23] Lacey A.A, Ockendon J.R., Tayler A.B. "Waiting-time" solutions of a nonlinear diffusion equation. *SIAM J. Appl. Math.*, 1982; **42**(6): 1252–1264.

[24] Lacey A.A. Initial motion of the free boundary for a non-linear diffusion equation. *IMA J. Appl. Math.* 1983; **31**(2): 113–119.

[25] Huang W., Russell R.D. *Adaptive Moving Mesh Methods* Springer series Applied Mathematical Sciences **174**, 2011.

[26] Huang W., Sun W. Variational mesh adaptation II: error estimates and monitor functions. *J. Comp. Phys.* 2003; **184**(2): 619–648.

[27] White A.B. On Selection Of Equidistributing Meshes For Two-Point Boundary-Value Problems. *SIAM J. Numer. Anal.* 1979; **16**(3): 472–502.

[28] White A.B. On The Solution Of Initial/Boundary-Value Problems In One Space Dimension. *SIAM J. Numer. Anal.* 1982; **19**(4): 683–697.

[29] Dorfi E.A., Drury L.O'C. Simple Adaptive Grids For 1-D Initial Value Problems. *J. Comput. Phys.* 1987; **69**(1): 175–195.

[30] de Boor C. *A Practical Guide To Splines*. Applied Mathematical Sciences 27, Springer-Verlag New York, 1978.