School of Mathematical
and Physical Sciences

University of Reading

Department of Mathematics and Statistics

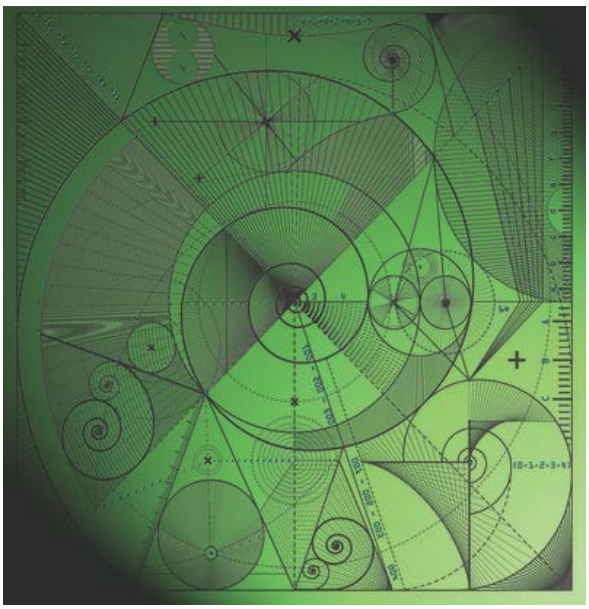
Preprint MPS-2014-14

8 May 2014

A finite difference moving mesh method based on conservation for moving boundary problems

by

T.E. Lee, **M.J. Baines** and **S. Langdon**



A finite difference moving mesh method based on conservation for moving boundary problems

T. E. Lee[a,b,1], M. J. Baines[a], S. Langdon[a]

[a]*Department of Mathematics and Statistics, University of Reading, UK*
[b]*Mathematical Institute, University of Oxford, UK*

Abstract

We propose a velocity-based moving mesh method in which we move the nodes so as to preserve local mass fractions. Consequently, the mesh evolves to be finer where the solution presents rapid changes, naturally providing higher accuracy without the need to add nodes. We use an integral approach which avoids altering the structure of the original equations when incorporating the velocity and allows the solution to be recovered algebraically. We apply our method to a range of one-dimensional moving boundary problems: the porous medium equation, Richards' equation, and the Crank-Gupta problem. We compare our results to exact solutions where possible, or to results obtained from other methods, and find that our approach can be very accurate (1% relative error) with as few as ten or twenty nodes.

Keywords: , Time dependent partial differential equations, Finite difference methods, Velocity-based moving meshes, Mass conservation
2010 MSC: , 65M06, 92-08, 92C99

1. Introduction

Time-dependent partial differential equations (PDEs) on moving domains, with known fluxes across the boundaries, occur regularly in physical and biological modelling, and must often be solved numerically. The location of the moving boundary is often critical and may require special numerical resolution. In particular, the solution may exhibit singular behaviour at the boundary that is challenging to capture numerically.

Adaptive numerical schemes modify the mesh during the course of computation in response to changes in the dependent variable (or its approximation) in order to achieve greater precision and/or greater efficiency. Generally, an adaptive mesh scheme becomes preferable to a fixed mesh scheme when areas of interest represent only a fraction of the domain being investigated. Increasing the resolution in these areas may then be computationally less expensive than refinement of the mesh over the entire grid. The most common form of mesh adaptivity is *h*-refinement adaptivity which involves repeated subdivision of the intervals of a fixed mesh. Other strategies include *p*-refinement, in which the solution is represented locally by higher order polynomials, and *r*-refinement in which the mesh points are relocated at each time step. The

*Corresponding author
Email address: tamsin.lee@maths.ox.ac.uk (+44)1865 611511 (T. E. Lee)

use of r-refinement has been stimulated by interest in geometric integration, in particular scale invariance (see, e.g., [7]). For scale invariant differential equations, independent and dependent variables are treated alike. An r-refinement method varies the solution and the mesh simultaneously, meaning that the scheme exhibits the same scale invariance as the underlying differential equation. The article by Budd, Huang and Russell [7] and the book by Huang and Russell [14] describe many theoretical and practical aspects of r-adaptivity.

In this paper a particular r-refinement adaptive scheme is described for the solution of one-dimensional time-dependent PDEs on moving domains. The approach relocates a constant number of nodes by moving the mesh points, keeping a node located at each moving boundary. We show that a mesh with as few as ten or twenty nodes can offer a relative error of less than 1% (see Tables 1–4 in §4). The work we present here preserves mass (or relative mass as appropriate), causing the mesh to naturally refine where the solution has a high gradient. This is particularly useful for solutions with blow-up, or (as demonstrated here) infinite slope. Attractive aspects of the approach are that no interpolation of the boundary is required, only the moving domain need be discretised, and the continuous movement of the mesh points allows easier inclusion of time integrators.

Under r-refinement nodes may be relocated in many ways, according to the choice of monitor functions, and the solution is often found from a modified form of the PDE. A mesh equation is often solved simultaneously with the modified PDE so as to generate the node positions in tandem with the solution, as in the Moving Mesh PDE approach [4, 13], the Moving Finite Element method of Miller [18, 19, 21], or the parabolic Monge-Ampere approach of Budd and Williams [5, 6]. By contrast, in the method described in this paper a single time-dependent equation is solved, that of the mesh, the solution being determined algebraically from a conservation principle. The approach is a finite difference version of the velocity-based moving mesh finite element scheme described by Baines, Hubbard and Jimack in [1], in which the mesh equation is based upon conserving a proportion of the total integral (mass) of the dependent variable in the domain. The method in [1] differs from methods depending on the technique of equidistribution [4, 13, 5, 6] since equidistribution is not an integral part of the strategy, but is related to the Deformation method of Liao and co-workers [16, 17] and to the Geometric Conservation Law (GCL) method of Cao, Huang and Russell [8]. The scheme described herein has been applied to a specific tumour growth problem in [15]. Here we generalise the approach to a wider class of problems, provide key implementation details, and show numerical results for three different nonlinear diffusion problems, each example demonstrating a key feature absent from the problem in [15]. Moreover, we validate our results via comparison with known exact solutions and with results from other (unrelated) approaches.

The layout of the paper is as follows. In §2 we describe the conservation approach, and its finite difference implementation. First, in §2.1, we consider mass conserving problems. Then in §2.2 these ideas are extended to non mass-conserving problems using a normalisation technique. In §3 the schemes are applied to three moving boundary problems, beginning in §3.1 with a mass-conserving problem governed by the porous medium equation (PME) (see, e.g., [24]), for which we consider a symmetrical test problem, treated with just one moving boundary. In §3.2 the method is applied to a test problem governed by Richards' equation (see [23]). This problem also conserves global mass but the test problem considered is unsymmetrical, so there are two moving boundaries. The third problem, detailed in §3.3, is known as the Crank-Gupta or diffusion-absorption problem [9], for which global mass is not conserved. We solve the Crank-Gupta problem for two sets of boundary data, one corresponding to that of the original problem (see [9]), and the other chosen so that we can easily verify our results against a known exact

solution. Numerical results for all our examples are provided in §4, and some conclusions are presented in §5.

We remark finally that our investigation is confined to initial-boundary-value problems for which the solution $u(x,t)$ is one-signed in the interior of the domain, which is sufficient for the validity of the method.

2. Conservation-based moving mesh methods

Let $u(x,t)$ be a positive solution of the generic time-dependent scalar PDE

$$\frac{\partial u(x,t)}{\partial t} = \mathcal{L}u(x,t), \qquad t > t^0, \ x \in (a(t), b(t)), \tag{1}$$

where $\mathcal{L}$ is a purely spatial differential operator. In all of our examples we have a moving boundary at $x = b(t)$ at which we impose the following boundary conditions

$$u(b(t),t) = 0, \tag{2}$$

$$u(b(t),t)\frac{\mathrm{d}b}{\mathrm{d}t} = 0. \tag{3}$$

The initial condition is

$$u(x,t^0) = u^0(x), \quad x \in (a(t^0), b(t^0)).$$

We introduce a time-dependent space coordinate $\tilde{x}(x,t)$ which coincides instantaneously with the fixed coordinate x. Consider two such coordinates, $\tilde{x}(x_1,t)$ and $\tilde{x}(x_2,t)$, in $(a(t),b(t))$, abbreviated to $\tilde{x}_1(t)$ and $\tilde{x}_2(t)$. The rate of change of the mass in the subinterval $(\tilde{x}_1(t), \tilde{x}_2(t))$ is given by Leibnitz' Integral Rule in the form

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\tilde{x}_1(t)}^{\tilde{x}_2(t)} u(s,t)\,\mathrm{d}s = \int_{\tilde{x}_1(t)}^{\tilde{x}_2(t)} \left(\frac{\partial u(s,t)}{\partial t} + \frac{\partial}{\partial s}(u(s,t)v(s,t))\right)\mathrm{d}s, \tag{4}$$

where

$$v(x,t) = \left.\frac{\mathrm{d}\tilde{x}}{\mathrm{d}t}\right|_{\tilde{x}=x} \tag{5}$$

is a local velocity. We denote the total mass by

$$\theta(t) := \int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x. \tag{6}$$

2.1. A method based on preservation of partial masses

We begin by describing a solution method for problems that conserve the total integral (global mass) of the solution, i.e. for which $\theta(t)$ remains constant for all $t \geq t^0$. Since $\tilde{x}_1(t)$ and $\tilde{x}_2(t)$ are arbitrary, equation (4) demonstrates the equivalence of the Lagrangian conservation law,

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{\tilde{x}_1(t)}^{\tilde{x}_2(t)} u(s,t)\,\mathrm{d}s = 0, \tag{7}$$

and the Eulerian conservation law,

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial}{\partial x}(u(x,t)v(x,t)) = 0. \tag{8}$$

From (8) and the PDE (1) we have

$$\mathcal{L}u(x,t) + \frac{\partial}{\partial x}(u(x,t)v(x,t)) = 0, \tag{9}$$

which, given $u(x,t)$, may be regarded as an equation for the velocity $v(x,t)$. For a unique solution of (9), $u(x,t)v(x,t)$ must be imposed at one point which may be thought of as an 'anchor' point. Integrating (9) from $a(t)$ to x,

$$\int_{a(t)}^{x} \mathcal{L}u(s,t)\,\mathrm{d}s + u(x,t)v(x,t) = u(a(t),t)v(a(t),t),$$

where $u(x,t)v(x,t)$ is imposed at the anchor point $x = a(t)$. The velocity $v(x,t)$ is then given by

$$v(x,t) = \frac{u(a(t),t)\,v(a(t),t) - \int_{a(t)}^{x} \mathcal{L}u(s,t)\,\mathrm{d}s}{u(x,t)}, \tag{10}$$

at all interior points, since $u(x,t) > 0$ in the interior of the domain.

Our numerical method is based on the idea that points $\tilde{x}(x,t)$ of the domain can be moved with this velocity in a Lagrangian manner using

$$\tilde{x}(x,t+\Delta t) = \tilde{x}(x,t) + \Delta t\; v(x,t) + O(\Delta t)^2. \tag{11}$$

To recover the solution $u(\tilde{x}(t),t)$, given $\tilde{x}_1(t)$ and $\tilde{x}_2(t)$, we use the conservation law (7) in the integrated form

$$\int_{\tilde{x}_1(t)}^{\tilde{x}_2(t)} u(s,t)\,\mathrm{d}s = c(\tilde{x}_1(t), \tilde{x}_2(t)), \tag{12}$$

where $a(t) < \tilde{x}_1(t) < \tilde{x}_2(t) < b(t)$, and

$$c(\tilde{x}_1(t), \tilde{x}_2(t)) = c(\tilde{x}_1(t^0), \tilde{x}_2(t^0)) = \int_{\tilde{x}_1(t^0)}^{\tilde{x}_2(t^0)} u^0(s)\,\mathrm{d}s.$$

A one point quadrature approximation to (12) leads to

$$u(\tilde{x},t) = \frac{c(\tilde{x}_1(t^0), \tilde{x}_2(t^0))}{\tilde{x}_2(t) - \tilde{x}_1(t)} + O\left(\Delta\tilde{x}\right), \tag{13}$$

where $\Delta\tilde{x} = \tilde{x}_2(t) - \tilde{x}_1(t)$, for all $\tilde{x} \in (\tilde{x}_1, \tilde{x}_2)$. Boundary conditions may be imposed on $u(\tilde{x},t)$ at this stage. Examples are described in §3 below.

We now define our notation. Given a time step $\Delta t > 0$ and a fixed number $N+1$ of spatial nodes, choose discrete times $t^m = m\Delta t$, $m = 0, 1, \ldots$, and discretise the interval at each discrete time t^m using the nodal points $X_j^m = \tilde{x}_j(t^m)$, $j = 0, 1, \ldots, N$, for which $a(t^m) = X_0^m < X_1^m < \ldots < X_N^m = b(t^m)$. Also define approximations $U_j^m \approx u(\tilde{x}_j, t^m)$ and $V_j^m \approx v(\tilde{x}_j, t^m)$.

Our finite difference moving mesh algorithm for mass-conserving problems is then as follows. Choose initial node positions X_j^0, $j = 0, 1, \ldots, N$, with corresponding approximate solution values $U_j^0 > 0$, and use them to determine the approximate masses

$$C_j = \left(X_{j+1}^0 - X_{j-1}^0\right) U_j^0, \qquad j = 1, \ldots, N-1.$$

Then at time t^m for $m = 1, 2, \ldots$, given X_j^m and U_j^m we compute X_j^{m+1} and U_j^{m+1} as follows:

1. Evaluate the interior velocities ($cf.$ (10))

$$V_j^m = \frac{U_0^m V_0^m - \int_{X_0^m}^{X_j^m} \mathcal{L}u(s, t^m)\, \mathrm{d}s}{U_j^m}, \qquad j = 1, ..., N-1,$$

where the integral is discretised, for example, by a trapezium rule. At the boundaries extrapolate the velocity from interior values.

2. Evolve the nodal positions X_j^m, $j = 1, \ldots, N-1$, in time from t^m to t^{m+1} by the explicit Euler timestepping scheme ($cf.$ (11))

$$X_j^{m+1} = X_j^m + \Delta t\, V_j^m. \tag{14}$$

3. Recover the solution U_j^{m+1} at interior points as ($cf.$ (13))

$$U_j^{m+1} = \frac{C_j}{X_{j+1}^{m+1} - X_{j-1}^{m+1}}, \qquad j = 1, ..., N-1, \tag{15}$$

with $U_N^{m+1} = 0$ from (2) and U_0^{m+1} being updated either from given boundary conditions or by extrapolation, depending on the nature of the problem (see §3).

2.2. A method based on preservation of relative partial masses

For more general problems that do not conserve mass, $\theta(t)$ (defined by (6)) varies with time. Hence (7) and (8) no longer hold. We may however make use of Leibnitz' Integral Rule applied to the *normalised* function $u(x,t)/\theta(t)$, giving

$$\frac{\mathrm{d}}{\mathrm{d}t}\left\{\frac{1}{\theta(t)}\int_{\tilde{x}_1(t)}^{\tilde{x}_2(t)} u(s,t)\, \mathrm{d}s\right\} = \frac{1}{\theta(t)}\int_{\tilde{x}_1(t)}^{\tilde{x}_2(t)} \left(\frac{\partial u(s,t)}{\partial t} + \frac{\partial}{\partial s}(u(s,t)v(s,t)) - \frac{\dot{\theta}(t)}{\theta(t)}u(s,t)\right) \mathrm{d}s, \tag{16}$$

for all $a(t) < \tilde{x}_1(t) < \tilde{x}_2(t) < b(t)$, where $v(x,t)$ is the local velocity (5) and $\dot{\theta}(t) = \mathrm{d}\theta/\mathrm{d}t$. Since $\tilde{x}_1(t)$ and $\tilde{x}_2(t)$ are arbitrary, equation (16) shows that the Lagrangian conservation equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left\{\frac{1}{\theta(t)}\int_{\tilde{x}_1(t)}^{\tilde{x}_2(t)} u(s,t)\, \mathrm{d}s\right\} = 0, \tag{17}$$

is equivalent to the generalised Eulerian conservation equation ,

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial}{\partial x}(u(x,t)v(x,t)) = \frac{\dot{\theta}(t)}{\theta(t)}u(x,t). \tag{18}$$

We derive the velocity from this generalised form in the same manner that we used in (8). That is, from (18) and the PDE (1) we derive

$$\mathcal{L}u(x,t) + \frac{\partial(u(x,t)v(x,t))}{\partial x} = \frac{\dot{\theta}(t)}{\theta(t)}u(x,t), \tag{19}$$

which, given $u(x,t)$, can be regarded as an equation for $v(x,t)$ in terms of $\theta(t)$ and $\dot{\theta}(t)$. As before, for a unique solution $u(x,t)v(x,t)$ must be imposed at the anchor point $x = a(t)$, so that the integral of (19) from $a(t)$ to x gives

$$u(x,t)v(x,t) = u(a(t),t)\, v(a(t),t) - \int_{a(t)}^{x} \mathcal{L}u(s,t)\, \mathrm{d}s + \frac{\dot{\theta}(t)}{\theta(t)} \int_{a(t)}^{x} u(s,t)\, \mathrm{d}s.$$

Hence the velocity is given by

$$v(x,t) = \frac{u(a(t),t)\, v(a(t),t) - \int_{a(t)}^{x} \mathcal{L}u(s,t)\, \mathrm{d}s + \frac{\dot{\theta}(t)}{\theta(t)} \int_{a(t)}^{x} u(s,t)\, \mathrm{d}s}{u(x,t)} \tag{20}$$

at all interior points, since $u(x,t) > 0$ in the interior of the domain.

To evaluate $\dot{\theta}$ we integrate (19) from $a(t)$ to $b(t)$, assuming that $u(x,t)$ and $v(x,t)$ are continuous up to the boundary, yielding

$$\int_{a(t)}^{b(t)} \mathcal{L}u(s,t)\, \mathrm{d}s + \Big[u(x,t)v(x,t)\Big]_{a(t)}^{b(t)} = \dot{\theta}(t), \tag{21}$$

which determines $\dot{\theta}$ explicitly (using (3)).

The points $\tilde{x}(x,t)$ of the domain are now moved with the velocity (20) in a Lagrangian manner, again using (11), and we can also update θ using

$$\theta(t + \Delta t) = \theta(t) + \Delta t\, \dot{\theta}(t) + O(\Delta t)^2.$$

To recover the solution $u(\tilde{x}(t),t)$ we choose $\tilde{x}_1$, $\tilde{x}_2$, such that (17) holds, in which case

$$\frac{1}{\theta(t)} \int_{\tilde{x}_1(t)}^{\tilde{x}_2(t)} u(s,t)\, \mathrm{d}s = c(\tilde{x}_1(t), \tilde{x}_2(t)), \tag{22}$$

for $a(t) < \tilde{x}_1(t) < \tilde{x}_2(t) < b(t)$, where

$$c(\tilde{x}_1(t), \tilde{x}_2(t)) = c(\tilde{x}_1(t^0), \tilde{x}_2(t^0)) = \frac{1}{\theta(t^0)} \int_{\tilde{x}_1(t^0)}^{\tilde{x}_1(t^0)} u^0(s)\, \mathrm{d}s,$$

and thus

$$u(\tilde{x},t) = \theta(t) \frac{c(\tilde{x}_1(t), \tilde{x}_2(t))}{\tilde{x}_2(t) - \tilde{x}_1(t)} + O(\Delta\tilde{x}) \tag{23}$$

for all $\tilde{x} \in (\tilde{x}_1, \tilde{x}_2)$, as in (13). Again, the boundary conditions may be imposed on $u(\tilde{x},t)$ at this stage.

The discretisations given in §2.1 are augmented by the additional approximations $\Theta^m \approx \theta(t^m)$ and $\dot{\Theta}^m \approx \dot{\theta}(t^m)$, and then our finite difference moving mesh algorithm for non mass-conserving

problems is as follows. Choose initial node positions X_j^0 with corresponding approximate solution values $U_j^0 > 0$, $j = 1, \ldots, N-1$, and use them to calculate the approximate relative masses

$$C_j = \frac{1}{\Theta^0}\left(X_{j+1}^0 - X_{j-1}^0\right) U_j^0,$$

where Θ^0 is given by ($cf.$ (6))

$$\Theta^0 = \frac{1}{2}\sum_j \left(X_{j+1}^0 - X_j^0\right)\left(U_j^0 + U_{j+1}^0\right),$$

using a trapezium rule. Then at time t^m for $m = 1, 2, \ldots$, given Θ^m, X_j^m and U_j^m we compute Θ^{m+1}, X_j^{m+1} and U_j^{m+1} as follows:

1. Evaluate the rate of change $\dot{\Theta}^m$ of the approximate total mass Θ^m in the form ($cf.$ (21))

$$\dot{\Theta}^m = \int_{X_0^m}^{X_N^m} \mathcal{L}u(s, t^m)\, \mathrm{d}s + U_N^m V_N^m - U_0^m V_0^m,$$

   where the integral is discretised using a trapezium rule;

2. Evaluate the discrete velocity at interior points as ($cf.$ (20)),

$$V_j^m = \frac{U_0^m V_0^m - \int_{X_0^m}^{X_j^m} \mathcal{L}u(s, t^m)\, \mathrm{d}s, + C_j \dot{\Theta}^m}{U_j^m}, \qquad j = 1, \ldots, N-1,$$

   where the integral is discretised using, for example, a trapezium rule. At the boundaries extrapolate the velocity from interior values.

3. Evolve both the nodal positions X_j^m, $j = 1, \ldots, N-1$, and the total mass Θ^m from t^m to time t^{m+1} by the explicit Euler time-stepping scheme (14) and $\Theta^{m+1} = \Theta^m + \Delta t \dot{\Theta}^m$.

4. Recover the solution U_j^m at interior points as ($cf.$ (23))

$$U_j^m = \Theta^m \frac{C_j}{X_{j+1}^m - X_{j-1}^m}, \qquad j = 1, \ldots, N-1,$$

   and at $j = 0$, $j = N$ as in Step 3 of the algorithm of §2.1.

3. Examples

In this section we apply the methods outlined in §2 to some specific moving boundary problems in one-dimension.

3.1. The Porous Medium Equation

The PME is the simplest nonlinear diffusion problem which arises in a physically natural way, describing processes involving fluid flow, heat transfer or diffusion. It also occurs in mathematical biology and other fields [24]. We assume the initial data is symmetrical about its centre of mass, taken to be the origin, in which case the PME takes the form

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u^n \frac{\partial u}{\partial x}\right), \qquad t > t^0,\ x \in (-b(t), b(t)),$$

with $u(-b(t),t) = u(b(t),t) = 0$ and $u(\pm b(t),t)\mathrm{d}b/\mathrm{d}t = 0$. For this problem the total mass (6) is conserved and the centre of mass is fixed in time [24], from which it follows that the solution retains the symmetry of the initial data for all time. We therefore model only half of the region, i.e. $x(t) \in [0, b(t)]$, with $a(t) = 0$ as the anchor point for all t. For the half problem we have

$$\frac{\partial u}{\partial x} = 0 \qquad \text{at} \quad x = 0, \tag{24}$$

by symmetry. From (10) the velocity is given by

$$v(x,t) = -\frac{1}{u(x,t)} \int_0^x \frac{\partial}{\partial s}\left(u(s,t)^n \frac{\partial u}{\partial s}\right) \mathrm{d}s \;=\; -u^{n-1}\frac{\partial u}{\partial x} \;=\; -\frac{1}{n}\frac{\partial (u^n)}{\partial x}, \quad t > t^0,\ x \in [0, b(t)). \tag{25}$$

Given X_j^m and U_j^m, $j = 0, 1, \ldots, N$, $m = 0, 1, 2, \ldots$, the finite difference algorithm of § 2.1 is used to calculate the velocity V_j^m at each node j, $j = 0, 1, \ldots, N$, then the new nodal positions X_j^{m+1}, and finally the approximate solution U_j^{m+1}. A standard discretisation of the velocity (25) at interior nodes is

$$V_j^m = -\frac{1}{n}\left(\frac{(U_{j+1}^m)^n - (U_{j-1}^m)^n}{X_{j+1}^m - X_{j-1}^m}\right), \qquad j = 1, 2, ..., N-1,$$

which, although of second order on a uniform mesh, is only a first order discretisation on a non-uniform mesh. An approximation which is second order on a non-uniform mesh (i.e. exact for quadratics) uses all three values U_{j-1}^m, U_j^m and U_{j+1}^m, and is

$$V_j^m = -\frac{1}{n}\left(\frac{\frac{1}{\Delta_+ X_j^m}\left(\frac{\Delta_+ (Uj^m)^n}{\Delta_+ X_j^m}\right) + \frac{1}{\Delta_- X_j^m}\left(\frac{\Delta_- (Uj^m)^n}{\Delta_- X_j^m}\right)}{\frac{1}{\Delta_+ X_j^m} + \frac{1}{\Delta_- X_j^m}}\right), \qquad j = 1, 2, ..., N-1, \tag{26}$$

where

$$\Delta_+(\cdot)_j = (\cdot)_{j+1} - (\cdot)_j \quad \text{and} \quad \Delta_-(\cdot)_j = (\cdot)_j - (\cdot)_{j-1}$$

(see [21]). We note that equation (26) is an inversely weighted sum, or linear interpolation, of the gradients $\Delta_\pm (U_j^m)^n / \Delta_\pm X_j^m$. The velocity at $x = 0$ is zero and at the moving boundary $x = X_N^m$ the velocity V_N^m is extrapolated by a polynomial approximation using three adjacent points. The new mesh is obtained at time $t^{m+1} = t^m + \Delta t$ by the explicit Euler time-stepping scheme (14).

The updated approximate solution U_j^{m+1} is given by (15), $j = 1, \ldots, N-1$. At $j = 0$ the approximate solution U_0^{m+1} is calculated using (26) with $X_{-1} = -X_1$, approximating the boundary condition (24). At the outer boundary, $U_N^{m+1} = 0$ from (2). Results are presented in §4.

3.2. Richards' Equation

Richards' equation is a nonlinear PDE which models the movement of moisture in an unsaturated porous medium [23]. In the present paper we model a particular form of Richards' equation, where the solution describes liquid flowing downwards through an unsaturated porous medium, making it applicable to the tracking of a contaminated liquid. The equation is of the form

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(u^{n-2}\frac{\partial u}{\partial x}\right) + \frac{\partial u^n}{\partial x}, \qquad t > t^0,\ x \in (a(t), b(t)), \tag{27}$$

for some integer $n > 2$, with $u(a(t), t) = u(b(t), t) = 0$ and $u(\partial a/\partial t) = u(\partial b/\partial t) = 0$ at the boundaries. The total mass is again conserved in time [23]. The velocity is given by (10) with $\mathcal{L}u$ defined as the right-hand side of (27),

$$v(x, t) = -u^{n-3}\frac{\partial u}{\partial x} - u^{n-1} = -\frac{1}{n-2}\frac{\partial (u^{n-2})}{\partial x} - u^{n-1}. \tag{28}$$

In a similar way to (26) we discretise (28) as

$$V_j^m = -\frac{1}{n-2}\left(\frac{\frac{1}{\Delta_+ X_j^m}\left(\frac{\Delta_+ (U j^m)^{n-2}}{\Delta_+ X_j^m}\right) + \frac{1}{\Delta_- X_j^m}\left(\frac{\Delta_- (U j^m)^{n-2}}{\Delta_- X_j^m}\right)}{\frac{1}{\Delta_+ X_j^m} + \frac{1}{\Delta_- X_j^m}}\right) - (U_j^m)^{n-1}, \qquad j = 1, \ldots, N-1.$$

Again, the outer boundary velocities V_0^m, V_N^m are extrapolated from interior points, using three adjacent nodes. The new mesh X_j^{m+1} is obtained from V_j^m by an explicit Euler time-stepping scheme, as in (14). The updated approximate solution U_j^{m+1}, $j = 1, \ldots, N-1$, is given by (15), and at the boundaries $U_0^{m+1} = U_n^{m+1} = 0$. Results for this example are shown in §4.

3.3. The Crank-Gupta problem

The Crank-Gupta problem was derived to model the diffusion of oxygen through an absorbing tissue [9], but also applies within the Black-Scholes framework of financial modelling due to the valuation of an American option being a similar free boundary problem [11].

The differential equation is

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} - 1, \qquad 0 < x < b(t), \tag{29}$$

with boundary conditions

$$\frac{\partial u}{\partial x} = 0 \quad \text{at} \quad x = 0, \text{ for } t > 0, \tag{30}$$

$$u = 0, \quad \frac{\partial u}{\partial x} = 0 \quad \text{at} \quad x = b(t), \text{ for } t > 0. \tag{31}$$

For this problem the total mass $\theta(t)$ decreases with time due to the negative source term in (29).

The initial condition at $t^0 = 0$ is taken as

$$u(x, 0) = \frac{1}{2}(1 - x)^2$$

for $x \in [0,1]$, as in [9], giving initial total mass $\theta(0) = 1/6$. Similarly, we can determine the normalised partial integrals $c(x)$ from (22) as

$$c(x) = \frac{1}{\theta(0)} \int_0^x \frac{1}{2}(1-s)^2 \, \mathrm{d}s = x^3 - 3x^2 + 3x. \tag{32}$$

The rate of change $\dot{\theta}$ of the total mass θ is given by substituting the PDE (29) and the boundary conditions (30)–(31) into (21), yielding

$$\dot{\theta}(t) = \int_0^{b(t)} \left(\frac{\partial^2 u}{\partial x^2} - 1 \right) \mathrm{d}x = \left[\frac{\partial u}{\partial x} - x \right]_0^{b(t)} = -b(t). \tag{33}$$

The velocity $v(x,t)$ is obtained by substituting the PDE (29) and the boundary conditions (30)–(31) into (20) and evaluating the integral, giving for $x \in (0, b(t))$

$$v(x,t) \quad = \quad \frac{1}{u(x,t)} \left(\dot{\theta}(t) c(x) - \int_0^x \left\{ \frac{\partial^2 u}{\partial s^2} - 1 \right\} \, \mathrm{d}s \right) = \frac{1}{u(x,t)} \left(-c(x)\, b(t) - \frac{\partial u}{\partial x} + x \right) \tag{34}$$

(substituting for $\dot{\theta}(t)$ from (33) and using the boundary condition at $x = 0$).

We use the algorithm of §2.2. The discrete form C_j of $c(x)$ at interior points is ($cf.$ (32))

$$C_j = X_j^3 - 3X_j^2 + 3X_j, \qquad j = 1, \ldots, N,$$

while the discrete form $\Theta(t^m)$ of $\theta(t^m)$ is ($cf.$ (33))

$$\dot{\Theta} = -X_N^m.$$

Also the discrete form V_j^m of the velocity $v(x,t)$ at interior points is ($cf.$ (34))

$$V_j^m = \frac{1}{U_j^m} \left\{ -C_j X_N^m - \left(\frac{\frac{1}{\Delta^+ X_j^m} \left(\frac{\Delta^+ U_j^m}{\Delta^+ X_j^m} \right) + \frac{1}{\Delta^- X_j^m} \left(\frac{\Delta^- U_j^m}{\Delta^- X_j^m} \right)}{\frac{1}{\Delta^+ X j^m} + \frac{1}{\Delta^- X_j^m}} \right) + X_j(t) \right\}, \qquad j = 1, \ldots, N.$$

At the outer boundary we would normally extrapolate the boundary velocity V_N^m from velocities at internal points and update the position of the outer node along with the internal nodes. However in this case extrapolation can produce a positive boundary velocity whereas the boundary velocity should be negative [9]. An alternative is to exploit the asymptotic behaviour of the solution at the outer boundary with

$$u(x,t) \sim \frac{1}{2}(x - b(t))^2 \quad \text{as } x \to b(t),$$

following from (29) and (31). Therefore, in the discrete case we make the approximation

$$U_{N-1}^{m+1} \approx \frac{1}{2}(X_{N-1}^{m+1} - X_N^{m+1})^2,$$

which leads to

$$X_N^{m+1} = X_{N-1}^{m+1} + \sqrt{2U_{N-1}^{m+1}} \tag{35}$$

(taking the positive square root).

The new node positions X_j^{m+1}, $j = 0, \ldots, N$ at time t^{m+1} as well as the new total mass Θ^{m+1} are obtained by the explicit Euler time-stepping scheme.

3.4. The Crank-Gupta problem with a modified boundary conditions

There is no known analytical solution for the Crank-Gupta problem although approximate solutions have been given in [10]. Hence, in order to compare our results to an exact solution we have modelled the Crank-Gupta PDE with a modified boundary condition for which an exact solution is known, which can then be used for comparison [1]. The one-dimensional Crank-Gupta problem with a modified boundary condition

$$\frac{\partial u}{\partial x} = e^{t-1} - 1 \qquad \text{at} \quad x = 0, \qquad t > t^0, \tag{36}$$

replacing (30), and initial conditions

$$u(x,0) = e^{x-1} - x, \qquad t^0 = 0, \ x \in [0,1], \tag{37}$$

has solution

$$u(x,t) = \begin{cases} e^{x+t-1} - x - t & x \le 1-t, \\ 0 & x > 1-t \end{cases} \tag{38}$$

(see, e.g., [1]). By applying the conservation based moving mesh method to this modified problem we can investigate the accuracy of the scheme for a non mass-conserving problem. The normalised partial integrals $c(x)$ (see (22)) are

$$c(x) = \frac{1}{\theta(0)} \int_0^x \left(e^{s-1} - s \right) \mathrm{d}s = \frac{e^{-1}(e^x - 1) - \frac{x}{2}}{\frac{1}{2} - e^{-1}}, \tag{39}$$

where $\theta(0) = 1/2 - e^{-1}$ from (6) and (37). The rate of change $\dot{\theta}$ of the approximate total mass θ (21), and the velocity of the interior nodes (20), are

$$\dot{\theta}(t) = 1 - e^{t-1} - b(t). \tag{40}$$

$$v(x,t) = \frac{1}{u(x,t)} \left(\dot{\theta}(t) c(x) - \frac{\partial u}{\partial x} + x - 1 + e^{t-1} \right), \tag{41}$$

from (29), (31) and (36). Equations (40)–(41) are equivalent to (33)–(34), but with an additional $\pm(1 - e^{t-1})$ term from the modified boundary condition. We again apply the algorithm of §2.2 using discrete forms of (39)–(41). At the fixed boundary, $V_0^m = 0$. At the moving boundary, equation (35) is again employed since the moving boundary conditions are the same as for the original problem. The new node positions X_j^{m+1}, as well as the new total mass Θ^{m+1}, are obtained from V_j^m by the explicit Euler time-stepping scheme. The solution is recovered in the same manner as for the original Crank-Gupta problem in §3.3.

4. Numerical results

In this section we present results from applying the moving mesh method to the four problems described above: the PME, Richards' equation, the original Crank-Gupta problem, and the Crank-Gupta problem with modified boundary conditions. In each case the initial mesh is equally spaced. For each problem we examine the convergence of the finite difference moving mesh method as the number of nodes N increases and as Δt decreases. We solve for $t \in [t^0, T]$ and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 1, 2, \ldots$. In order to compare results for different

values of $\hat{N}$, we denote the points of the mesh for a particular value of $\hat{N}$ by $x_{j,\hat{N}}(t)$, $j = 0, \ldots, N$. We then compute both $x_{2^{\hat{N}-1}i,\hat{N}}(t)$ and $u_{2^{\hat{N}-1}i,\hat{N}}(t) \approx u(x_{2^{\hat{N}-1}i,\hat{N}}(t), t)$ for each $i = 0, \ldots, 10$; this new notation allows comparison of $x_{j,\hat{N}}(t)$ and $u_{j,\hat{N}}(t)$ at eleven different points, determined by $j = 2^{\hat{N}-1}i$, $i = 0, \ldots, 10$, for various N. Where possible we compare the numerical outcomes with the exact solution and boundary position. When such a solution is not known, we compare with numerical results determined using other methods. In each case we denote our reference solution by $\overline{u}(x, t)$, and our reference boundary position by $\overline{x}(t)$.

Recalling that we have used explicit Euler time-stepping, in order to balance the spatial and temporal errors, we take $\Delta t = O\left(1/N^2\right)$, anticipating that the pointwise errors $|\overline{x}(t) - x_{N,\hat{N}}(t)|$ and $|\overline{u}(x_{2^{\hat{N}-1}i,\hat{N}}(t), t) - u_{2^{\hat{N}-1}i,\hat{N}}(t)|$ will decrease as $\hat{N}$ increases, for each $i = 0, \ldots, 10$.

As a measure of the errors, we calculate the ℓ_2 norm of the error in our solution, and the relative error of our boundary position, as defined by

$$E^u_N := \sqrt{\frac{\sum_{i=0}^{10} |\overline{u}(x_{2^{\hat{N}-1}i,\hat{N}}(T), T) - u_{2^{\hat{N}-1}i,\hat{N}}(T)|^2}{\sum_{i=0}^{10} |\overline{u}(x_{2^{\hat{N}-1}i,\hat{N}}(T), T)|^2}}, \qquad E^x_N := \frac{|\overline{x}(T) - x_{N,\hat{N}}(T)|}{|\overline{x}(T)|},$$

for $\hat{N} = 1, 2, 3, 4, \ldots$ (i.e. $N = 10, 20, 40, 80, \ldots$). We investigate the hypothesis that

$$E^u_N \sim \frac{1}{N^p} \qquad \text{and} \qquad E^x_N \sim \frac{1}{N^q}, \tag{42}$$

for large N, where p and q are the estimated orders of convergence. If (42) holds then we expect that p_{2N} and q_{2N} defined by

$$p_{2N} = -\log_2\left(\frac{E^u_{2N}}{E^u_N}\right), \qquad q_{2N} = -\log_2\left(\frac{E^x_{2N}}{E^x_N}\right), \tag{43}$$

will approach the constant values p and q as N increases. Since each step of our scheme is second order in space and first order in time, and recalling that $\Delta t = O\left(1/N^2\right)$, we might expect to see $p, q \approx 2$.

4.1. Porous Medium Equation

We solve for $t \in [1, 5]$ and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 1, \ldots, 6$. We use the self-similar initial conditions for $n = 1, 2, 3$,

$$n = 1: \qquad u(x, 1) = 1 - \frac{x^2}{6}, \qquad b(1) = \sqrt{6}, \tag{44}$$

$$n = 2: \qquad u(x, 1) = \left(1 - \frac{x^2}{4}\right)^{\frac{1}{2}}, \qquad b(1) = 2, \tag{45}$$

$$n = 3: \qquad u(x, 1) = \left(1 - \frac{3x^2}{10}\right)^{\frac{1}{3}}, \qquad b(1) = \sqrt{\frac{10}{3}}, \tag{46}$$

see [3, 22]. The exact solution at the calculated mesh points is

$$\overline{u}(x, t) = \frac{1}{t^{1/(2+n)}}\left(1 - \frac{x^2}{b(t)^2}\right)^{1/n}, \tag{47}$$

and the exact boundary position, is

$$\bar{x}(t) = b(t) = t^{1/(n+2)}\sqrt{\frac{2(n+2)}{n}}.$$

As stated above, to balance the spatial and temporal errors we use $\Delta t = O\left(1/N^2\right)$, precisely $\Delta t = 0.4\left(4^{-\hat{N}}\right)$. Convergence results for $n = 1$ are shown in Table 1. We see that E_N^u and E_N^x decrease as N increases. This suggests that as the number of nodes increases our approximations to both the solution and the boundary position are converging. The p and q values presented strongly indicate second-order convergence of both the numerical solution and numerical boundary position.

N	E_N^u	p_N	E_N^x	q_N
10	7.715×10^{-3}	-	1.451×10^{-3}	-
20	1.941×10^{-3}	2.0	3.066×10^{-4}	2.2
40	4.976×10^{-4}	2.0	7.138×10^{-5}	2.1
80	1.259×10^{-4}	2.0	1.730×10^{-5}	2.0
160	3.166×10^{-5}	2.0	4.262×10^{-6}	2.0
320	7.937×10^{-6}	2.0	1.058×10^{-6}	2.0

Table 1: Relative errors E_N^u and E_N^x, for the porous medium equation with $n = 1$.

The results from the self-similar solutions for $n = 1, 2, 3$ and $N = 20$ are given in Figures 1–3. In each case we see that with only twenty nodes in our mesh, the boundary position (Figures 1(b)–3(b)) is computed very accurately (better than 1% relative error at $t = 5$ in each case). From (47) we note that the exact solution for $n = 2, 3$ has a steep gradient at the boundaries, as can be seen in Figures 2(a) and 3(a). Figures 1(c)–3(c) show exactly how the mesh moves. We observe a smooth even spread of the nodes, without mesh tangling, in all three cases.

4.2. Richards' Equation

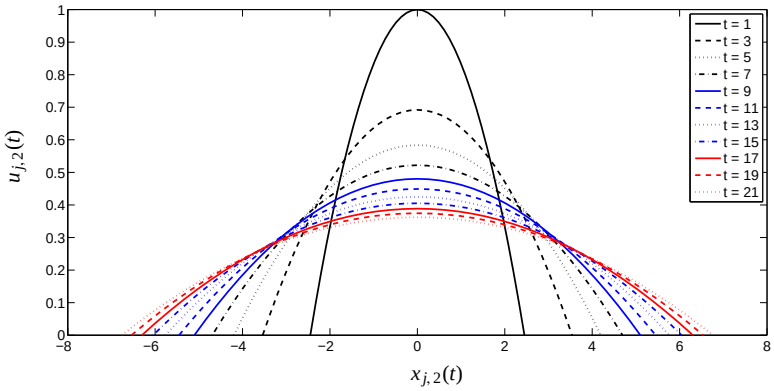
In this section we present results from applying the moving mesh method to Richards' equation, as described in §3.2. To test that the numerical solution from the moving mesh method converges we compare the solution with that from a very fine fixed mesh. All numerical results presented here are for $n = 3$. In the absence of an exact reference solution we do not compare the position of the boundary.

We solve for $t \in [0, 0.5]$ and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 1, \ldots, 4$. We compare the numerical solutions with a numerical solution calculated by solving Richards' equation on the fixed mesh $\bar{x}_{\bar{j}} \in [-4, 4]$, $\bar{j} = 0, 1, \ldots, 10000$, which is given by

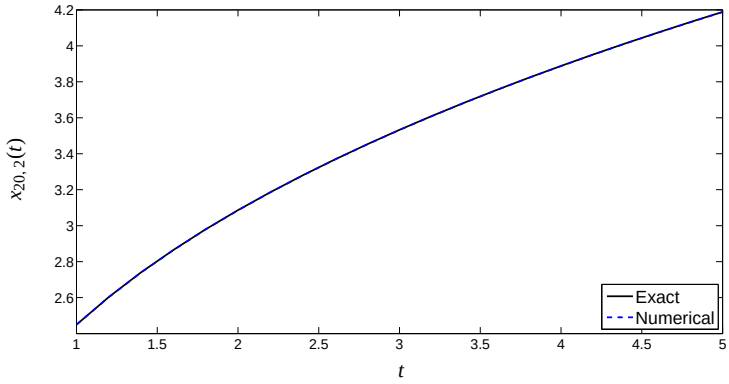
$$\begin{aligned}\frac{\bar{u}_{\bar{j}+\frac{1}{2}}(t+\Delta t) - \bar{u}_{\bar{j}+\frac{1}{2}}(t)}{\Delta t} &= (\bar{u}^{n-2})_{\bar{j}+\frac{1}{2}}(t)\frac{\bar{u}_{\bar{j}+1}(t) - \bar{u}_{\bar{j}}(t)}{h} + (\bar{u}^{n})_{\bar{j}+\frac{1}{2}}(t) \\ &\quad -(\bar{u}^{n-2})_{\bar{j}-\frac{1}{2}}(t)\frac{\bar{u}_{\bar{j}}(t) - \bar{u}_{\bar{j}-1}(t)}{h} - (\bar{u}^{n})_{\bar{j}-\frac{1}{2}}(t),\end{aligned}$$

where $h = 8 \times 10^{-4}$ is the uniform spacing between two mesh points,

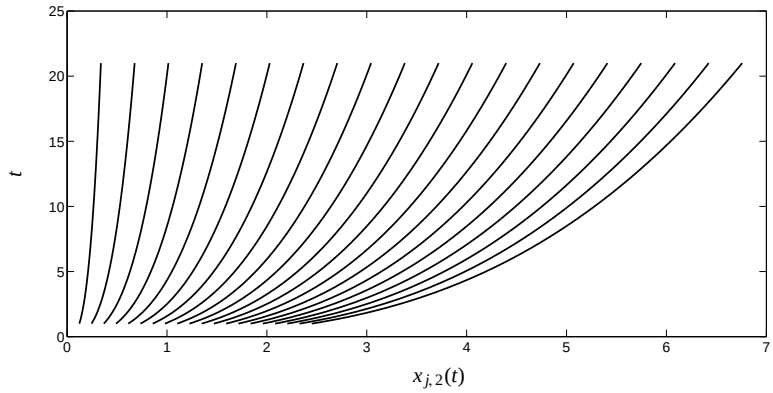
$$(\bar{u}^n)_{\bar{j}+\frac{1}{2}}(t) \approx \frac{1}{2}\left((\bar{u}^n)_{\bar{j}+1}(t) + (\bar{u}^n)_{\bar{j}}(t)\right) \quad \text{and} \quad (\bar{u}^n)_{\bar{j}-\frac{1}{2}}(t) \approx \frac{1}{2}\left((\bar{u}^n)_{\bar{j}}(t) + (\bar{u}^n)_{\bar{j}-1}(t)\right).$$



(a) The approximate solution.

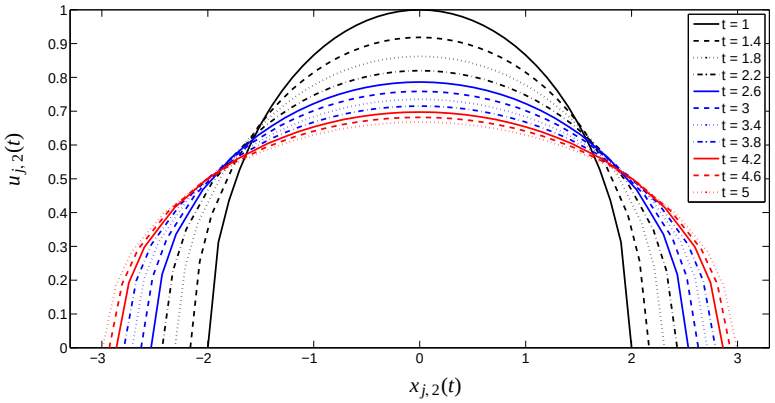


(b) The boundary position (relative error at $t = 5$ is 0.0015).

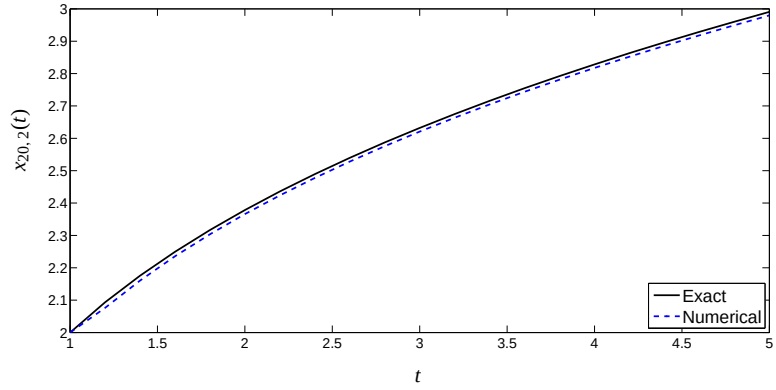


(c) The mesh trajectory.

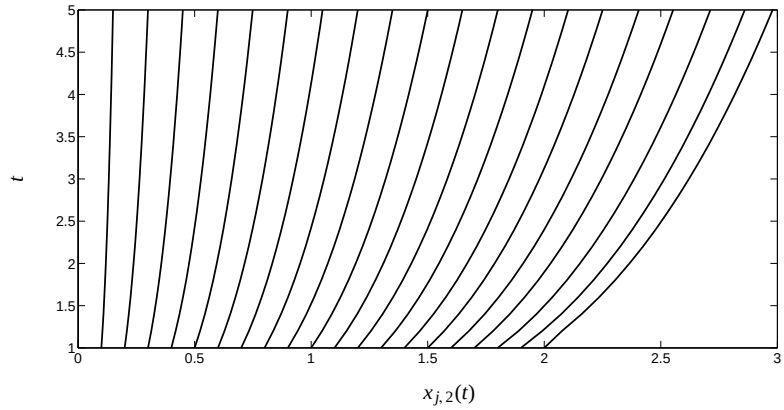
Figure 1: The PME with self-similar initial conditions for $n = 1$ (44), $N = 20$ ($\hat{N} = 2$), $\Delta t = 0.04$.



(a) The approximate solution.

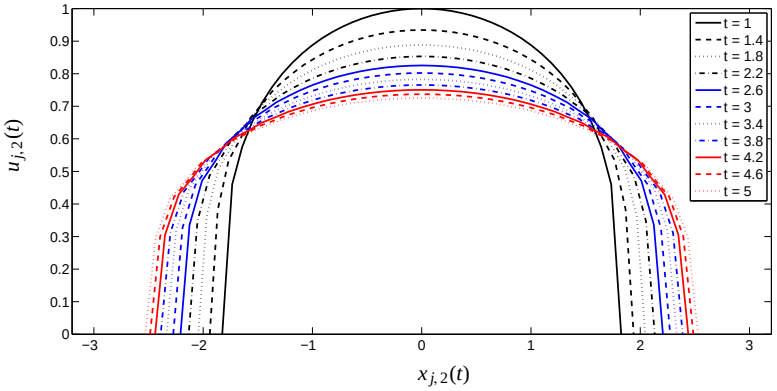


(b) The boundary position (relative error at $t = 5$ is 0.0037).

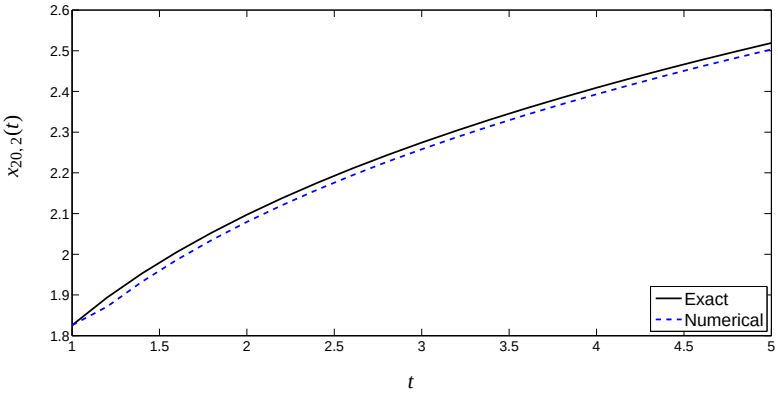


(c) The mesh trajectory.

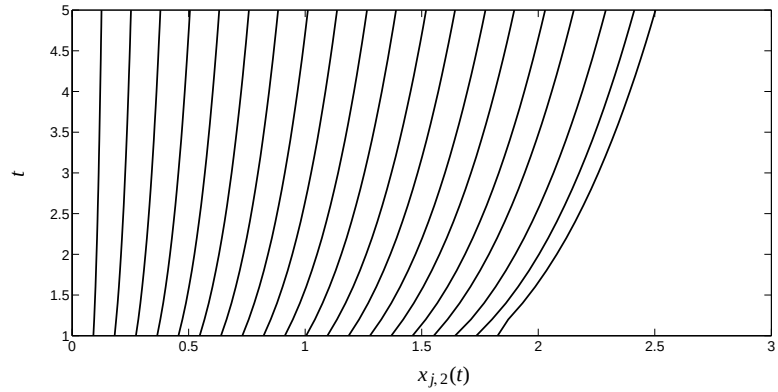
Figure 2: The PME with self-similar initial conditions for $n = 2$ (45), $N = 20$ ($\hat{N} = 2$), $\Delta t = 0.04$.



(a) The approximate solution.



(b) The boundary position (relative error at $t = 5$ is 0.0064).



(c) The mesh trajectory.

Figure 3: The PME with self-similar initial conditions for $n = 3$ (46), $N = 20$ ($\hat{N} = 2$), $\Delta t = 0.04$.

We take the initial conditions to be

$$u(x,0) = 1 - x^2, \qquad x \in [-1,1].$$

To balance the spatial and temporal errors we use $\Delta t = O(1/N^2)$, precisely $\Delta t = 0.4\left(4^{-\hat{N}}\right)$ (as with the PME). To compare solutions, we find the closest match to $x_{2^{\hat{N}-1}i,\hat{N}}(t)$, $i = 0,\ldots,10$, from the fixed mesh $\overline{x}_j$ (where the points match to four decimal places), then the corresponding solution on the fixed mesh $\overline{u}_j(t)$ is compared to $u_{2^{\hat{N}-1}i,\hat{N}}(t)$.

Computed values of E_N^u for $\hat{N} = 1,\ldots,4$ (i.e. $N = 10, 20, 40, 80$) are shown in Table 2. The relative error is less than 5% for $N = 10$ and less than 1% for $N = 20$. The values of p_N suggest second-order convergence.

N	E_N^u	p_N
10	4.47×10^{-2}	-
20	7.97×10^{-3}	2.5
40	1.90×10^{-3}	2.1
80	4.75×10^{-4}	2.0

Table 2: Relative errors E_N^u for Richards' equation, $n = 3$.

The numerical solution as computed with $N = 40$ is plotted in Figure 4. We see from Figure 4(b) that the mesh moves smoothly and does not tangle.

4.3. The Original Crank-Gupta problem

In this section we present results from applying the moving mesh method to the Crank-Gupta problem as described in §3.3. The boundary position was calculated using (35). Figure 5(a) shows the numerical solution at various times for $t \in [0, 0.19]$. We note that the solution is behaving as expected; the outer boundary is moving in, whilst the inner boundary is levelling out to satisfy the boundary condition.

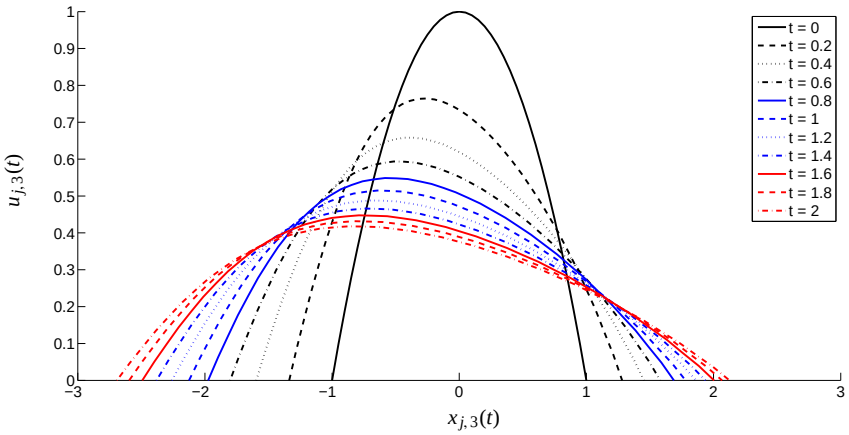
There is no known analytical solution to the Crank-Gupta problem but, as a comparison, we may use the results of Dahmardah and Mayers [10] who derived a Fourier Series solution (also see [20]). By comparing their results with earlier work in [12] they concluded that their method is very accurate. To check whether our method converges as N increases and Δt decreases, we compare $u_{0,\hat{N}}(0.1)$ and $x_{N,\hat{N}}(0.1)$ to the results given in [10] for $t = 0.1$, which are

$$\begin{aligned} \overline{u}(0,0.1) &= 0.143177, \\ \overline{x}(0.1) &= 0.935018. \end{aligned}$$

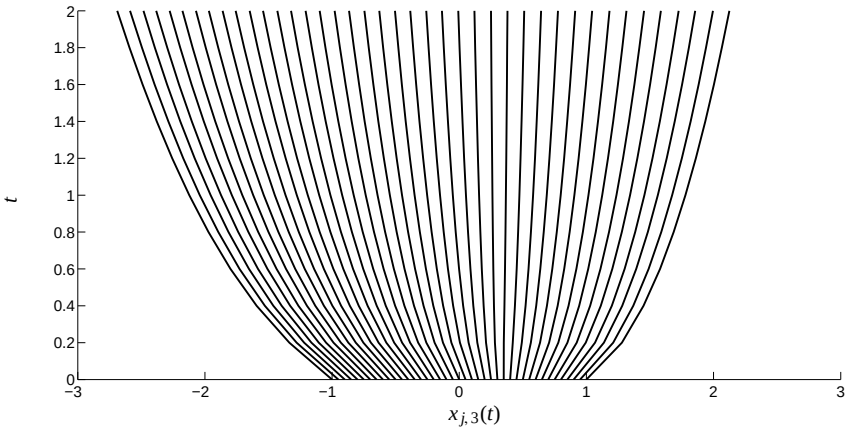
We solve for $t \in [0, 0.1]$ and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 1,\ldots,6$. To balance the spatial and temporal errors we use $\Delta t = O\left(1/N^2\right) = 1/[1600(4^{\hat{N}})]$. As a measure of the relative pointwise errors, we calculate

$$\hat{E}_N^u = \frac{|\overline{u}(0,0.1) - u_{0,\hat{N}}(0.1)|}{|\overline{u}(0,0.1)|}, \qquad \hat{E}_N^x = \frac{|\overline{x}(0.1) - x_{N,\hat{N}}(0.1)|}{|\overline{x}(0.1)|},$$

for $\hat{N} = 1,\ldots,6$ (i.e. $N = 10, 20, 40, 80, 160, 320$). We investigate the same hypothesis (42) as in the two previous sections (though note that our measure of error is slightly different here). We again compute p_{2N} and q_{2N} via (43), but with E_N^u and E_N^x replaced by $\hat{E}_N^u$ and $\hat{E}_N^x$, respectively.



(a) The approximate solution.

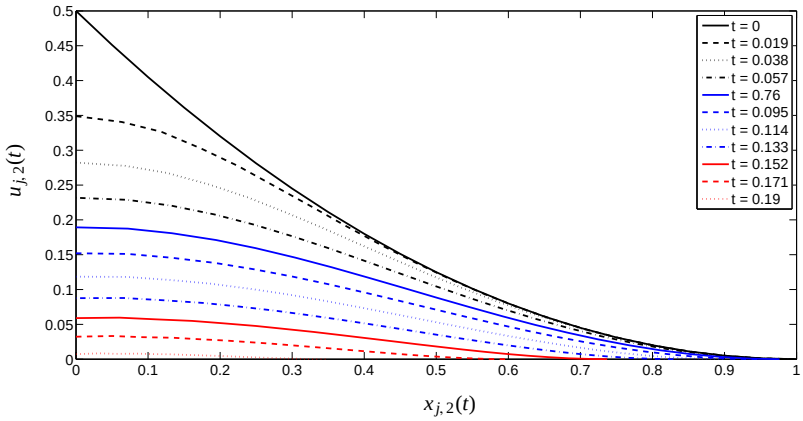


(b) The mesh trajectory.

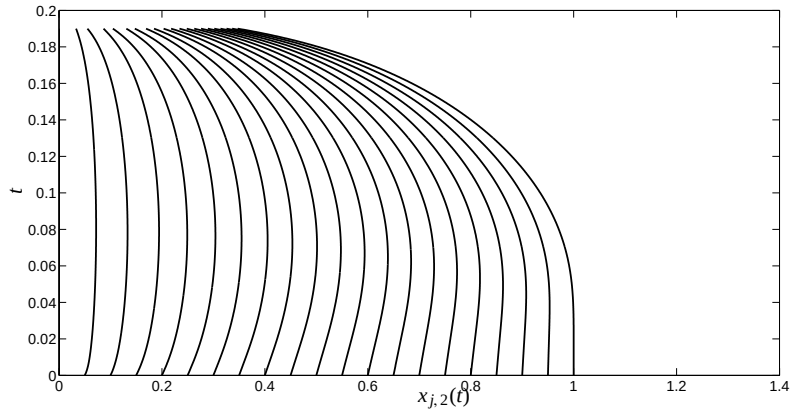
Figure 4: Richards' equation with $n = 3$, $N = 40$ ($\hat{N} = 3$), $\Delta t = 0.01$.

N	$u_{0,\hat{N}}(0.1)$	$\hat{E}_N^u$	p_N	$x_{N,\hat{N}}(0.1)$	$\hat{E}_N^x$	q_N
10	0.142791	2.696×10^{-3}	-	0.935761	7.946×10^{-4}	-
20	0.142721	3.185×10^{-3}	-0.2	0.935385	3.925×10^{-4}	1.0
40	0.143040	9.569×10^{-4}	1.7	0.935120	1.091×10^{-4}	1.8
80	0.143141	2.514×10^{-4}	1.9	0.935043	2.674×10^{-5}	2.0
160	0.143168	6.286×10^{-5}	2.0	0.935024	6.417×10^{-6}	2.0
320	0.143175	1.397×10^{-5}	2.2	0.935019	1.069×10^{-6}	2.6

Table 3: Relative errors $\hat{E}_N^u$ and $\hat{E}_N^x$, for the original Crank-Gupta problem.



(a) The approximate solution.



(b) The mesh trajectory.

Figure 5: The Crank-Gupta problem solved using relative partial mass conservation, $N = 20$ ($\hat{N} = 2$), $\Delta t = 2 \times 10^{-4}$.

There are some irregularities in Table 3, as we might expect since we are comparing relative pointwise errors. Nonetheless, it would be reasonable to suggest that the non mass-conserving moving mesh method with explicit Euler time-stepping has second-order convergence. The movement of the nodes for $N = 20$, $t \in [0, 0.19]$, is shown in Figure 5(b). The nodes are moving smoothly and not tangling, with the ratio between the nodes remaining roughly constant. We observe that despite the boundary moving in, the nodes still cluster towards the boundary, where higher resolution allows greater accuracy to track the boundary movement.

4.4. The Crank-Gupta problem with modified boundary conditions

As mentioned before, we were unable to compare the original Crank-Gupta problem to an analytical solution. However, by imposing an alternative boundary condition (36) we can examine convergence as N increases and Δt decreases over the whole region. We solve for $t \in [0, 0.1]$

and compute results for $N = 10 \times 2^{\hat{N}-1}$, $\hat{N} = 1, \ldots, 6$. We compare the numerical outcomes with the exact solution (38), at $t = 0.1$,

$$\overline{u}(x_{j,\hat{N}}(0.1), 0.1) \quad = \quad e^{x_{j,\hat{N}}(0.1)-0.9} - x_{j,\hat{N}}(0.1) - 0.1.$$

To balance the spatial and temporal errors we use $\Delta t = O\left(1/N^2\right) = 0.02\left(4^{-\hat{N}}\right)$.

Numerical results are shown in Table 4. We see that E_N^u decreases as N increases, and the

N	E_N^u	p_N
10	7.581×10^{-3}	-
20	2.502×10^{-3}	1.6
40	6.796×10^{-4}	1.9
80	1.825×10^{-4}	1.9
160	4.879×10^{-5}	1.9
320	1.235×10^{-5}	2.0

Table 4: Relative errors E_N^u for the Crank-Gupta problem with modified boundary conditions.

values of p_N suggest second-order convergence.

Figures 6(a)–6(b) show the results from imposing the modified boundary condition, as computed with $N = 20$. The solution to the original problem is very small for $t = 0.19$, see Figure 5(a), whereas the modified problem decays more slowly. This is partly because the outer boundary moves in at a slower rate for the modified problem, which can be seen by comparing the movement of the last node in Figures 5(b) and 6(b) (where we observe that the boundary moves in linearly). Lastly, from Figure 6(b) we note that the nodes move in a fairly uniform manner, without tangling.

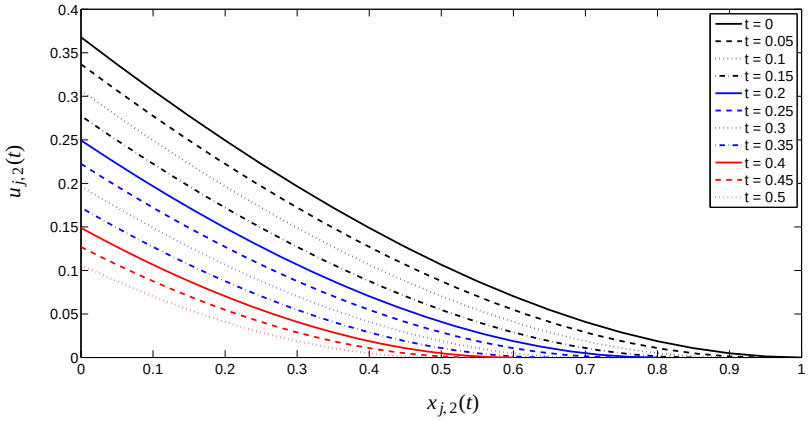
5. Conclusions

Work on moving meshes has evolved considerably over recent years, becoming a versatile tool to accurately simulate a wide range of problems. The key advantage of a moving mesh is its ability to adjust its distribution to focus on areas of interest, such as a moving boundary or blow-up. In this paper we have discussed one such method, a finite difference moving mesh method which is well-adapted to solving one-dimensional nonlinear initial boundary value problems. The velocity was determined by keeping the relative partial integrals of the solution,

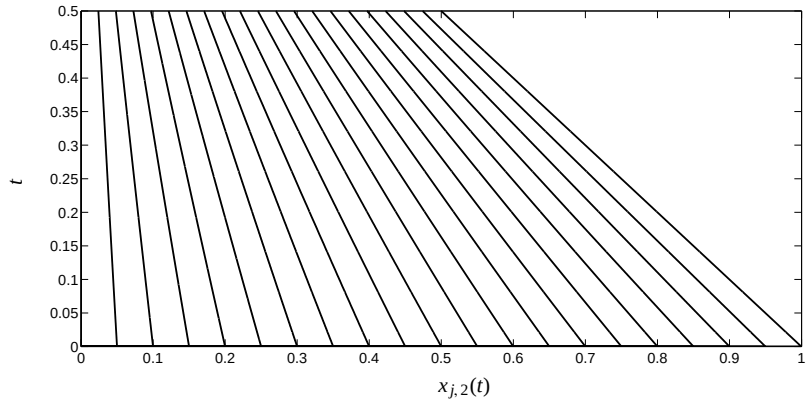
$$\frac{\int_{a(t)}^{\tilde{x}_j(t)} u(x,t)\,\mathrm{d}x}{\int_{a(t)}^{b(t)} u(x,t)\,\mathrm{d}x},$$

constant. This strategy is related to the GCL method and is similar to that used by Baines, Hubbard and Jimack for their moving mesh finite element algorithm [1].

We applied these methods to a number of moving boundary problems to investigate the effectiveness of this moving mesh approach. The problems we solved numerically increased in complexity, initially problems which conserve mass: the PME and Richards' equation (both of which are fluid flow problems). Then we looked at a problem with a variable total mass: the



(a) The approximate solution.



(b) The mesh trajectory.

Figure 6: The Crank-Gupta problem with modified boundary conditions, $N = 20$ ($\hat{N} = 2$), $\Delta t = 2 \times 10^{-4}$.

Crank-Gupta problem, which models oxygen-diffusion through tissue. We examined the accuracy in all cases and found that the numerical solution converged with roughly second-order accuracy. Furthermore, for the Crank-Gupta problem, we found that preservation of mass fractions can lead to higher resolution at the boundary, which is desirable.

Throughout this paper we have used an explicit Euler time-stepping scheme. Other explicit time-stepping schemes we experimented with are the higher order methods built into Matlab (ODE23, ODE45, ODE15s); see [15] for details. There was little difference in the results from all the Matlab solvers, indicating that none of the problems lead to a stiff system of ODEs for the $\tilde{x}_j(t)$. We found that all the time-stepping schemes produced accurate and stable results, with no mesh tangling, provided that sufficiently small time-steps were taken. It has been shown in [2] that the PME can also be solved by this moving mesh method with a semi-implicit time-stepping scheme using larger time steps.

We conclude that this moving mesh approach with an explicit time-stepping scheme is accurate for a range of problems. In particular, only twenty nodes (and in most cases only ten nodes) were sufficient to achieve better than 1% accuracy for every example presented here.

References

[1] Baines, M.J., Hubbard, M.E. and Jimack, P.K. (2005) A moving mesh finite element algorithm for the adaptive solution of time-dependent partial differential equations with moving boundaries. *Appl. Numer. Math.* **54** 450–469.

[2] Baines, M.J. and Lee, T.E. (2014) A large time-step implicit moving mesh scheme for moving boundary problems. *Numer. Methods Partial Differential Eq.* **30** 321-338.

[3] Barenblatt, G.I. (1952) On some unsteady motions of fluids and gases in a porous medium. *Prikladnaya Matematika i Mekhanika. (Translated in J. Appl. Math. Mech.)* **6** 67–78.

[4] Becket, G., Mackenzie, J.A. and Roberston, M.L. (2001) A moving mesh finite element method for the solution of two-dimensional Stefan problems. *J. Comput. Phys.* **168** 500-518.

[5] Budd, C.J. and Williams, J.F. (2006) Parabolic Monge-Ampere methods for blow-up problems in several spatial dimensions. *J. Phys. A* **39** 5425–5444.

[6] Budd, C.J. and Williams, J.F. (2008) Moving mesh generation using the Parabolic Monge-Ampere equation. *SIAM J. Sci. Comput.*

[7] Budd, C., Huang, W., and Russell, R.D. (2009) Adaptivity with moving grids. *Acta Numer.* 111–241.

[8] Cao, W., Huang, W. and Russell, R.D. (2002) A moving-mesh method based on the geometric conservation law. *SIAM J. Sci. Comput.* **24** 118-142.

[9] Crank. J. and Gupta. R.S. (1972) A moving boundary problem arising from the diffusion of oxygen in absorbing tissue. *J. Inst. Maths. Applics.* **10** 19–33.

[10] Dahmardah, H.O. and Mayers, D.F. (1983) A Fourier-Series solution of the Crank-Gupta equation. *IMA J. Numer. Anal.* **3** 81–85.

[11] Dewynne, J.N., Howison, S.D., Rupf, I., Wilmott, P. (1993). Some mathematical results in the pricing of American options. *Eur. J. of Appl. Math*, **4**, 381–398.

[12] Hansen, E. and Hougaard, P. (1974) On a moving boundary problem with biomechanics. *H. Inst. Maths. Applics.* **13** 385–398.

[13] Huang, W., Ren, Y. and Russell, R.D. (1994) Moving mesh partial differential equations (MMPDEs) based on the equidistribution priciple. *SIAM J. Sci. Comput.* **31** 709-730.

[14] Huang, W. and Russell, R.D. (2011) Adaptive Moving Mesh Methods. *Springer, New York.*

[15] Lee, T.E., Baines, M.J., Langdon, S. and Tindall, M.J. (2013) A moving mesh approach for modelling avascular tumour growth. *Appl. Numer. Math.* **72**, 99–114.

[16] Liao, G. and Anderson, D. (1992) A new approach to grid generation. *Appl. Anal.* **44** 285-298.

[17] Liao, G. and Xue, J. (2006) Moving meshes by the deformation method. *J. Comput. Appl. Math.* **195** 83-92.

[18] Miller, K. and Miller, R.N. (1981) Moving finite elements. I. *SIAM J. Numer. Anal.* **18** 1019-1032.

[19] Miller, K. (1981) Moving finite elements. II. *SIAM J. Numer. Anal.* **18** 1033-1057.

[20] Moody, R.O., & Baines, M.J. (1992). A constrained moving finite element solution of the one-dimensional oxygen diffusion with absorption problem. *J. Comput. Phys.*, **103(2)**, 442–449.

[21] Parker, J. (2010) An invariant approach to moving-mesh methods for PDEs. *MSc Dissertation, Brasenose College, University of Oxford, UK.*

[22] Pattle, R.E. (1959) Diffusion from an instantaneous point source with a concentration-dependent coefficient. *Q. J. Mech. Appl. Math.* **12** 407–409.

[23] Richards, L.A. (1931) Capillary conduction of liquids through porous mediums. *Physics 1* **5** 318-333.

[24] Vazquez, J.L. (2007) The porous medium equation: Mathematical theory. *Oxford University Press.*