An *h-r* moving mesh method for one-dimensional time-dependent PDEs

Benjamin Ong[1], Robert Russell[2], and Steven Ruuth[2]

[1] Institute for Cyber-Enabled Resesearch, Michigan State University, East Lansing, MI, 48824

[2] Department of Mathematics, Simon Fraser University, Burnaby, BC, V5A 1S6

Summary. We propose a new moving mesh method suitable for solving time-dependent partial differential equations (PDEs) in $\mathbb{R}^1$ which have fine scale solution structures that develop or dissipate. A key feature of the method is its ability to add or remove mesh nodes in a smooth manner and that it is consistent with r-refinement schemes. Central to our approach is an implicit representation of a Lagrangian mesh as iso-contours of a level set function. The implicitly represented mesh is evolved by updating the underlying level set function using a derived level set moving mesh partial differential equation (LMPDE). The discretized LMPDE evolves the level set function in a manner that quasi-equidistributes a specified monitor function. Beneficial attributes of the method are that this construction guarantees that the mesh does not tangle, and that connectivity is retained. Numerical examples are provided to demonstrate the effectiveness of our approach.

1 Introduction

To solve time-dependent partial differential equations (PDEs) numerically, a method of lines approach is typically implemented. The PDE is discretized in space, and the resulting system of ordinary differential equations (ODEs) is integrated in time using a suitable time-integrator.

It is often advantageous to employ an adaptive spatial mesh if fine-scale structures develop, propagate or disappear as the solution evolves. Examples of such behavior can be seen in combustion simulations [2, 17], shock formation and propagation in compressible fluid flow and shear flows [15]. Ideally, the mesh adaptation scheme clusters a higher density of grid points in a local, possibly time-dependent, region of interest.

Two well-known approaches to performing mesh adaptation are (i) h-refinement methods which coarsen or refine a mesh locally by varying the number of nodes, and (ii) r-refinement methods, or *moving mesh methods*, which redistribute a fixed number of grid nodes, concentrating a higher density of grid nodes in desired regions. Much work has been done on h-refinement methods; moving mesh methods are a more recent development [8]. Hybrid approaches may also be contemplated, and related to our work here, several h-r hybrid methods have been developed. There are several algorithms that utilize r-adaptivity to move an underlying coarse mesh, and then refine that mesh using h-adaptivity until some error criterion is reached [3, 16, 12, 7, 14]. In these methods, mesh nodes are added or removed locally, often changing the connectivity of a mesh. An alternative approach splits a domain into several regions, where h and r adaptivity are applied separately on each subdomain, e.g. [1].

Our approach also belongs to the class of h-r hybrid methods; however, it is fundamentally different in that the local density of mesh nodes is determined by the monitor function. Naturally, as local structures form and dissipate this causes the number of mesh nodes across the domain to vary. Connectivity of the mesh, however, does not vary because nodes are added or removed at the domain boundary. We further remark that underlying our method is an implicit (level set) representation of the mesh, where the location of the mesh nodes is embedded as level contours of a level set function. By construction, the level set representation circumvents all mesh crossing events; in one spatial dimension, mesh crossing would occur if two neighboring nodes, $x_k(t) < x_{k+1}(t)$, switch orderings at a later time, $t + \Delta t$, i.e., $x_k(t + \Delta t) > x_{k+1}(t + \Delta t)$. We note that the level set representation of grid nodes has been considered in earlier works [13]. Our study builds upon these by incorporating the most recent techniques for mesh evolution [8].

The remainder of the paper is divided into four main sections. In Section 2, the notion of equidistribution is reviewed, the notion of quasi-equidistribution is introduced, and de Boor's algorithm is outlined. In Section 3.2, the implicit representation of a mesh using level set functions is reviewed, and a level set evolution equation which drives a mesh towards equidistribution is derived, along with boundary conditions for r and $h - r$ adaptivity. Application of the $h - r$ algorithm is illustrated in Section 4, and concluding remarks are given in Section 5.

2 Review

This section begins with reviews of the notions of monitor functions and equidistribution. Then, quasi-equidistributed grids are defined, followed by an outline of de Boor's algorithm.

2.1 Monitor Functions

Consider PDEs of the form

$$\begin{aligned} u_t(x,t) &= f(t, u, u_x, u_{xx}, \ldots), \quad x \in [a,b], \quad t \geq 0, \\ u(x,0) &= g(x), \end{aligned} \tag{1}$$

with solution $u(x,t)$. A monitor function, $\rho(u(x,t))$, is generally chosen as some quantitative measure of the difficulty in approximating $u(x,t)$ locally. A popular choice is the scaled arc-length monitor function

$$\rho(u(x,t)) = \sqrt{1+\beta\|\partial_x u\|^2} \geq 1, \tag{2}$$

which is relatively large ($\rho \gg 1$) in regions where the solution has large variation, and is relatively small ($\rho \approx 1$) elsewhere. Another popular choice is the scaled curvature monitor function,

$$\rho(u(x,t)) = \sqrt{1+\beta\|\partial_{xx} u\|^2} \geq 1. \tag{3}$$

2.2 Equidistribution

We consider the general case where the monitor function has an explicit space and time dependence, $\rho(x,t)$. Traditional moving mesh methods seek to evenly distribute monitor functions using a fixed number of nodes [4]. Consider the set of $(N+1)$ grid nodes

$$x_0 = a < x_1(t) < \cdots < x_{N-1}(t) < x_N = b.$$

Definition 1. *A set of $(N+1)$ mesh nodes, $\{x_j(t)\}_{j=0}^{N}$,* **equidistributes** *a monitor function, $\rho(x,t)$, on the interval $[a,b]$ if the integral of the monitor function is equally distributed in each sub-interval, $[x_{j-1}(t), x_j(t)]$, i.e.,*

$$\int_{x_{j-1}(t)}^{x_j(t)} \rho(x,t)\,dx = \frac{1}{N}\int_a^b \rho(x,t)\,dx = \alpha(t), \quad \forall j = 1,2,\ldots,N, \tag{4}$$

where $\alpha(t)$, is independent of space.

2.3 Quasi-equidistribution

Suppose it is desirable that $\alpha(t)$, the integral of the monitor function in each subinterval, is constant in time, i.e., $\alpha(t) = \alpha_0$. This might arise for example, if ρ is some error estimate, and a control of the total error in each interval of the domain is sought. If the number of intervals is fixed, then it is not possible in general to find a set of mesh nodes which can keep $\alpha(t) = \alpha_0$. By allowing for a variable number of intervals, it is possible to keep $\alpha(t) = \alpha_0$ everywhere except in the final interval, where

$$\int_{x_{N-1}(t)}^{x_N(t)} \rho(x,t)\,dx < \alpha(t)$$

This leads us to the notion of equidistribution with a variable number of intervals in the domain:

Definition 2. *Suppose that a set of* (N_0+1) *mesh nodes,* $\{x_j(0)\}_{j=0}^{N_0}$*, equidistributes a monitor function,* $\rho(x,0)$ *at some initial time* $t=0$*, i.e.,*

$$\int_{x_{j-1}(0)}^{x_j(0)} \rho(x,0)\,dx = \frac{1}{N_0}\int_a^b \rho(x,0)\,dx = \alpha_0, \quad \forall j = 1,2,\ldots,N_0.$$

Then at time t*,*

$$N(t) = \lceil N_0 \frac{\int_a^b \rho(x,t)\,dx}{\int_a^b \rho(x,0)} \rceil$$

intervals **quasi-equidistributes** *a monitor function,* $\rho(u(x,t))$*, on the interval* $[a,b]$ *if* $x_0(t)=a$*,* $x_{N(t)}(t)=b$ *if*

$$\int_{x_{j-1}(t)}^{x_j(t)} \rho(x,t)\,dx = \alpha_0 \quad \forall j = 1,2,\ldots,N(t)-1,$$

and

$$\int_{x_{N(t)-1}(t)}^{x_{N(t)}} \rho(x,t)\,dx \le \alpha_0.$$

2.4 de Boor's algorithm

A method to generate an equidistributed mesh which satisfies Definition 1 at a discrete time level $t = t^n$ is de Boor's algorithm [5]. We shall use de Boor's algorithm, Algorithm 1, to initialize an equidistributed mesh later in our *h*-*r*-algorithm (Section 4).

To equidistribute a time-dependent monitor function, $\rho(x,t)$, and consequently evolve grids, one can implement de Boor's algorithm at each discrete time level. One such implementation is described in Algorithm 2. While de Boor's algorithm is generally successful at generating equidistributed grid points $\{x_j(t^n)\}$, it is too expensive to apply at each time step, and there is no control over how smoothly the mesh changes from time level t^n to time level t^{n+1}. Furthermore, de Boor's algorithm does not extend to $\mathbb{R}^n$. This motivates our development of moving mesh equations which smoothly evolve implicit representations of grids towards equidistribution.

3 Hybrid $h-r$ algorithm

To allow for the smooth addition or deletion of mesh nodes, we will utilize an *implicit* representation of an equidistributed mesh. In this section, the implicit representation of a mesh is introduced and then evolution equations, also referred to as Level set Moving mesh PDEs (LMPDEs) are derived. Boundary conditions which lead to r and $h-r$ adaptivity are discussed.

```
  Input: endpoints $\{a, b\}$; number of intervals $N$; monitor function $\rho(x) \geq 1$;
         initial mesh, $[x_0, x_1, \ldots, x_N]$
  Output: Equidistributed mesh, $[x_0, x_1, \ldots, x_N]$ (equidistributes $\rho(x)$)
1 Compute integral of the monitor function in each subinterval
      $S_j = (x_j - x_{j-1}) \left( \frac{\rho(x_{j-1}) + \rho(x_j)}{2} \right), \quad j = 1, \ldots, N,$
2 while $(\max_{1\leq j\leq N} S_j - \min_{1\leq j\leq N} S_j) > TOL$ do
3     Construct the piecewise linear function
          $I(x) = (x - x_{j-1}) \left( \frac{\rho(x_{j-1}) + \rho(x_j)}{2} \right) + \sum_{r=1}^{j-1} S_r$ for $x \in (x_{j-1}, x_j)$
4     Find $\{y_j\}$ such that $I(y_j) = \frac{j}{N} I(b), \quad j = 0, \ldots, N$
5     Set $\{x_j\} \leftarrow \{y_j\}, \quad j = 0, \ldots, N.$
6     Compute integral of the monitor function in each subinterval
          $S_j = (x_j - x_{j-1}) \left( \frac{\rho(x_{j-1}) + \rho(x_j)}{2} \right), \quad j = 1, \ldots, N,$
7 end
```

Algorithm 1: de Boor's algorithm computes an equidistributed mesh for a prescribed monitor function $\rho(x) \geq 1$. Note that ρ is independent of time for this algorithm.

```
  Input: endpoints $\{a, b\}$; number of intervals $N$; monitor function $\rho(x, t)$;
         initial time $t_0$; final time $T$; time step $\Delta t$.
1 set $n = 1$; set $t^n = t_0$
2 set $\mathbf{x}(t^0)$ to be equispaced in the desired domain.
3 while $t^n < T$ do
4     $\mathbf{x}(t^n) = deBoor(a, b, N, \rho(\cdot, t^n), \mathbf{x}(t^{n-1}))$
5     set $t \leftarrow t + \Delta t$; set $n \leftarrow n + 1$.
6 end
```

Algorithm 2: One can generate a simple moving mesh by using de Boor's algorithm to regenerate an equidistributed mesh for each discrete time level.

3.1 Implicit mesh representation

In an explicit representation of a mesh, the mesh points $\{x_j(t)\}$ are known and evolved directly. In an *implicit* representation of a mesh, the mesh points are the iso-contours of an evolving level set function.

Suppose a monitor function $\rho(x, 0) \geq 1$ is given. Consider the level set function

$$\psi(x, 0) = \int_a^x \rho(\tilde{x}, 0)\, d\tilde{x}, \tag{5}$$

and the contour levels

$$c_j = \frac{j}{N} \int_a^b \rho(x, 0)\, dx, \quad j = 0, \ldots, N. \tag{6}$$

By construction, the iso-contours of the level set function $\psi(x, 0)$, i.e.

$$\{x_j(0)\} = \{x_j(0) \mid \psi(x_j(0), 0) = c_j\}, \quad j = 0, \ldots, N,$$

give the equidistributed mesh.

For example, suppose

$$u(x, 0) = a(\exp(-(b(x-c))^2) + \tanh(d(x-e))) \tag{7}$$

with $a = 20, b = 40, c = 0.18, d = 40, e = 0.5$. Then the scaled arc-length monitor function (2) with $\beta = 1/1000$ is plotted in Figure 1(a). The level set function, approximated using a trapezoid quadrature approximation to equation (5), is plotted along with its iso-contours (dashed blue lines) and the corresponding adaptive grid (red crosses) in Figure 1(b). Figure 1(c) compares the piecewise linear reconstructions of the function u given by equation (7) using the a uniform grid and the equidistributed grid specified by the level set function.

3.2 Evolution equation

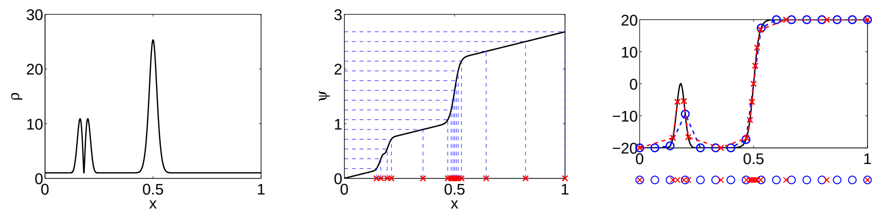
For an evolving mesh, the equidistribution principle is satisfied by finding iso-contours of the evolving level set function

$$\psi(x, t) = \int_a^x \rho(\tilde{x}, t)\, d\tilde{x}. \tag{8}$$

Taking a derivative with respect to x gives $\psi_x = \rho(x, t)$, or equivalently,

$$\frac{\psi_x(x, t)}{\rho(x, t)} = 1. \tag{9}$$

Taking a second derivative with respect to x gives



(a) Monitor Function (b) Level Set Function (c) Adaptive Grid

Fig. 1: (a) shows the monitor function of a prescribed function u. (b) shows the level set function and the iso-contours as defined by equations (5) and (6). The plot in (c) shows the prescribed function u and the linear interpolant formed when using a uniform mesh (blue) and an equidistributed mesh (red)

$$\frac{\partial}{\partial x}\left(\frac{1}{\rho(x,t)}\frac{\partial}{\partial x}\psi(x,t)\right)=0. \tag{10}$$

To approximate the solution of these equations, the corresponding descent equations

$$\dot{\psi}=-\frac{1}{\tau}\frac{\partial}{\partial x}\left(\frac{1}{\rho(x,t)}\frac{\partial}{\partial x}\psi(x,t)\right), \tag{LMPDE5}$$

may be evolved, where $\dot{\psi}=\psi_t$ and τ is a relaxation parameter. Notably, (LMPDE5) strongly resembles MMPDE5 in [9].

Alternatively, we can perturb equation (10) in a similar fashion to [9] to derive other evolution equations for the level set function. If we require that the iso contours of the level set function give an equidistributed grid at some later time $t+\tau$, where $0<\tau\ll 1$, then the following equation must hold:

$$\frac{\partial}{\partial x}\left(\frac{\psi_x(x,t+\tau)}{\rho(x,t+\tau)}\right)=0. \tag{11}$$

Using the expansions

$$\begin{aligned}\frac{\partial}{\partial x}\psi(x,t+\tau)&=\frac{\partial}{\partial x}\psi(x,t)+\tau\frac{\partial}{\partial x}\dot{\psi}(x,t)+\mathcal{O}(\tau^2),\\ \frac{1}{\rho(x,t+\tau)}&=\frac{1}{\rho(x,t)}-\tau\frac{\dot{\rho}(x,t)}{\rho(x,t)^2}+\mathcal{O}(\tau^2)\end{aligned}$$

in equation (11), we obtain

$$\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial\psi}{\partial x}\right)+\tau\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial\dot{\psi}}{\partial x}\right)-\tau\frac{\partial}{\partial x}\left(\frac{\dot{\rho}}{\rho^2}\frac{\partial\psi}{\partial x}\right)=0, \tag{LMPDE2}$$

where higher order terms have been dropped. Equation (LMPDE2) evolves the level set function in such a way that the iso-contours give an equidistributed mesh, even when ρ is independent of t. Noting that the term involving $\dot{\rho}$ is less important and can be dropped [9], giving the equation

$$\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial \dot{\psi}}{\partial x}\right) = -\frac{1}{\tau}\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial \psi}{\partial x}\right), \qquad \text{(LMPDE4)}$$

which closely resembles the MMPDE4 equation in [9]. Other evolution equations corresponding to the MMPDEs can be generated in a similar fashion.

3.3 Building in $h - r$ adaptivity through the boundary conditions

The LMPDEs evolve the level set function in such a way that the iso-contours give a mesh which equidistributes a monitor function; however, boundary conditions must still be specified. The boundary conditions will be used to control how many iso-contours (or equivalently, how many grid nodes) are used to equidistribute the monitor function.

In r-adaptivity, a fixed number of mesh nodes is used to equidistribute a monitor function. One method to enforce this condition is to find mesh trajectories which equidistribute a scaled monitor function,

$$\tilde{\rho}(x,t) = \frac{\rho(x,t)}{\int_a^b \rho(\tilde{x},t)\, d\tilde{x}}.$$

Alternatively, in terms of boundary conditions for the level set function, r-adaptivity corresponds to $\dot{\psi}(a,t) = \dot{\psi}(b,t) = 0$, or equivalently, the Dirichlet conditions

$$\psi(a,t) = \psi(a,0), \quad \psi(b,t) = \psi(b,0). \qquad (12)$$

For $h - r$ adaptivity, boundary conditions that allow for the addition or removal of grid nodes are desired. From (8), $\psi(x,t) = \int_a^x \rho(\tilde{x},t)\, d\tilde{x}$ which suggests using $\psi(a,t) = 0$ and $\psi(b,t) = \int_a^b \rho(\tilde{x},t)\, d\tilde{x}$. These Dirichlet boundary conditions can cause difficulty numerically due to the sensitivity of approximating $\int_a^b \rho(x,t)\, dx$. Instead, we enforce equation (9) at a boundaries, i.e., $\psi_x(a,t) = \rho(a,t)$ or $\psi_x(b,t) = \rho(b,t)$. A well-posed problem for equation (10) is subsequently formed by fixing the location of one grid point. Without loss of generality, we specify that there is a grid node at $x = a$ by choosing the boundary conditions

$$\psi(a,t) = 0, \quad \psi_x(b,t) = \rho(b,t). \qquad (13)$$

4 Numerical Examples

4.1 $\rho(x,t)$ is specified

To illustrate that the evolution equations correctly evolve the implicit representation of the mesh, we first consider the example from [9], where $\rho = \sqrt{1 + u_x^2}$ is a prescribed, dynamic arclength monitor function of

$$u(x,t) = \frac{1}{2}\left[1 - \tanh\left(c(t)(x - t - 0.4)\right)\right], \quad t \in [0, 0.45] \tag{14}$$
$$c(t) = 1 + \frac{10^3 - 1}{2}[1 + \tanh\left(100(t - 0.2)\right)].$$

The function $u(x,t)$ mimics a smooth wave that suddenly develops a steep gradient at about $t = 0.2$ before propagating towards $x = 1$. Figure 2 shows a plot of the solution, u. The arclength monitor function (2) with $\beta = 1$ is used for this example. This is a good test problem to ensure that our algorithm adds grid nodes smoothly as the layer develops, and then properly moves the mesh as the layer travels to the right.

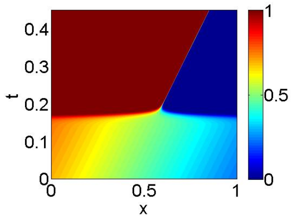


Fig. 2: Prescribed function, u from equation (14).

The idealized mesh trajectories are computed using Algorithm 2, where de Boor's algorithm is used to generate an equidistributed mesh for each discrete time level. In Figure 3(a), the number of intervals is kept constant at 16, giving the idealized r-adaptivity trajectories. In Figure 3(b), the same algorithm is used with the number of intervals varied, with

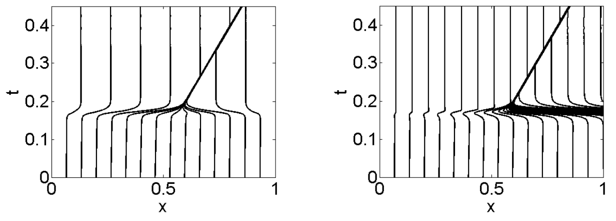
$$N(t) = \lceil 16 \frac{\int_a^b \rho(x,t)\, dx}{\int_a^b \rho(x,0)\, dx} \rceil$$

giving the idealized $h - r$ adaptivity.

The evolution equation (LMPDE4) is discretized using centered finite differences, viz.,

$$\begin{aligned} &\left[\left(\frac{1}{\rho_{i+1} + \rho_i}\right)\left(\frac{\dot{\psi}_{i+1} - \dot{\psi}_i}{x_{i+1} - x_i}\right) - \left(\frac{1}{\rho_i + \rho_{i-1}}\right)\left(\frac{\dot{\psi}_i - \dot{\psi}_{i-1}}{x_i - x_{i-1}}\right)\right] \\ &= -\frac{1}{\tau}\left[\left(\frac{1}{\rho_{i+1} + \rho_i}\right)\left(\frac{\psi_{i+1} - \psi_i}{x_{i+1} - x_i}\right) - \left(\frac{1}{\rho_i + \rho_{i-1}}\right)\left(\frac{\psi_i - \psi_{i-1}}{x_i - x_{i-1}}\right)\right], \end{aligned} \tag{15}$$

and the resulting system of ODEs, coupled with boundary conditions (12) or (13), is solved using a backward Euler integrator. For illustration purposes, the level set function is evolved on a computational uniform mesh with $\Delta x =$



(a) Idealized r adaptivity (b) Idealized $h{-}r$ adaptivity

Fig. 3: Idealized r adaptivity is shown in the left plot, where the total number of grid nodes is held constant at 16. In the right plot, we show the idealized trajectories which quasi-equidistribute the arclength monitor function. At time $t = 0.45$, 28 nodes are required to quasi-equidistribute ρ.

10^{-3}. The relaxation parameter is set to $\tau = 10^{-4}$. The level set function is initialized using a trapezoid quadrature approximation to equation (8), i.e.,

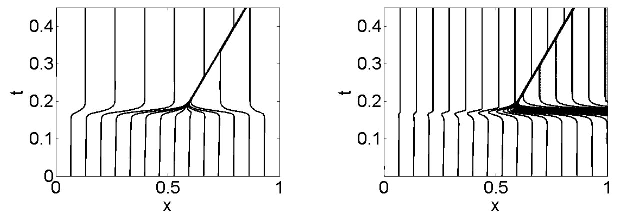
$$\psi_i^0 = \psi(x_i, 0) = \sum_{j=1}^{i} \left(\rho(x_{j-1}, 0) + \rho(x_j, 0)\right) \frac{\Delta x}{2}.$$

A total of 10^4 time steps are used. In Figure 4(a), the evolution equation is solved with the boundary condition in equation (12), resulting in r adaptivity. In Figure 4(b), the evolution equation is solved with the boundary condition in equation (13), resulting in $h{-}r$ adaptivity. In both cases, the mesh trajectories, i.e., the iso contours of the evolving level set function, agree qualitatively with the ideal mesh trajectories shown in Figure 3. Note that in Figure 4(b), nodes are added smoothly through the right boundary to resolve the steepening layer.

Using a fine uniform computational mesh for evolving the level set function is generally impractical. By extracting the mesh location from the iso-contours, an underlying moving mesh can be used to evolve the level set function. More precisely, suppose the level set function has been computed at time level t^n using a computational mesh x^n. Then,

1. Solve equation (15) for the level set function at time t^{n+1} using the mesh x^n.
2. Using first order interpolation, find the location of the quasi-equidistributed mesh at time t^{n+1}.
3. Denote this quasi-equidistributed mesh as the new computational mesh at time t^{n+1}, and re-initialize the level set function to the new computational mesh

Figure 5 shows mesh trajectories obtained by evolving the level set function using this moving-mesh, reinitialization approach. The boundary conditions (12) are used to recovery r-adaptivity. In Figure 5(a), 10^4 time steps were used.

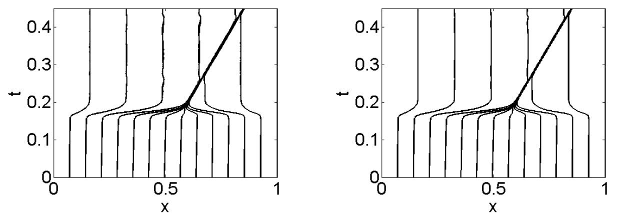


(a) Computed r adaptivity using a computational static uniform mesh

(b) Computed $h-r$ adaptivity using a computational static uniform mesh

Fig. 4: Mesh trajectories are obtained by evolving the level set function on a computational static uniform mesh. Observe that when the boundary condition given in equation (12) is used to solve the evolution equation, we recover r adaptivity. When the boundary condition in equation (13) is used, we get $h-r$ adaptivity as desired.

Although the trajectories look qualitatively reasonable, there are observable oscillations. If the number of time steps is increased to 10^5, the oscillations are reduced, as shown in Figure 5(b) .

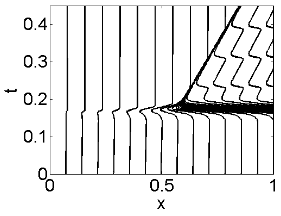


(a) Computed r adaptivity using a computational moving mesh, 10^4 time steps

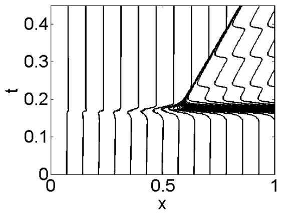
(b) Computed r adaptivity using a computational moving mesh, 10^5 time steps

Fig. 5: Mesh trajectories are obtained by evolving the level set function on a computational moving mesh consisting of 16 nodes. The mesh oscillations observed in (a) are reduced in (b).

Figure 6 shows mesh trajectories obtained by evolving the level set function using the moving-mesh, reinitialization approach. The boundary conditions (13) are used to recovery $h-r$ adaptivity. After the layer is formed and begins to propagate, mesh nodes are inserted to keep the same number of nodes in the layer, as nodes are used to resolve the solution to the left of the layer. The size of the oscillations decreases, however, as the number of initial mesh nodes is increased, as shown in Figure 7.

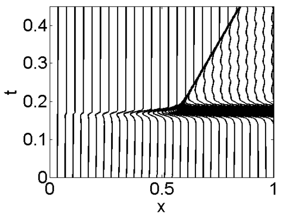


(a) Computed hr adaptivity using a computational moving $h-r$ mesh, 10^4 time steps

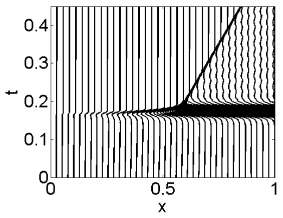


(b) Computed $h-r$ adaptivity using a computational moving $h-r$ mesh, 10^5 time steps

Fig. 6: Mesh trajectories obtained by evolving the level set function on a computational moving $h-r$ mesh. The mesh initially consists of 16 nodes, though more nodes are added as the layer appears. After a node is inserted, there is a brief relaxation time.



(a) 30 initial grid nodes



(b) 40 initial grid nodes

Fig. 7: Mesh trajectories obtained by evolving the level set function on a computational moving $h-r$ mesh. The number of initial mesh nodes are varied.

4.2 Solving a PDE using this hybrid h-r moving mesh method

Two possible methods for coupling PDEs to mesh dynamics are a quasi-Lagrange approach and a rezoning approach [11]. Since mesh nodes are being added and removed in our formulation, we will utilize the rezoning approach in this paper. In principle, a quasi-Lagrangian approach, augmented with interpolation when necessary, can also be used with our formulation.

In a rezoning approach, one alternates between updating the mesh and updating the solution to the PDE. More precisely, suppose the solution to the physical PDE has been computed at time level t^n using the mesh $x_i^n = x_i(t^n), i = 1, \dots, N(t^n)$; denote this as $u_i^n = u(x_i^n, t^n)$. Then, one computes the solution to a PDE, u_i^{n+1}, at time level t^{n+1} using the mesh x_i^{n+1} by:

1. Solving for the new mesh at time level t^{n+1}, $x_i^{n+1} = x_i(t^n), i = 1, \dots, N(t^{n+1})$
2. Interpolating the physical solution from the old mesh to the new one

3. Discretizing the PDE on the new mesh, holding the mesh fixed for the current time step.

Since the mesh x_i^{n+1} is generated using the mesh and physical solution (x^n, u^n), the mesh consequently only adapts to the current solution u^n and lags in time. Provided the time step is reasonably small or the solution does not change abruptly in time, the moving mesh should resolve the solution adequately.

We demonstrate that our hybrid h-r moving mesh method works with the rezoning approach by solving viscous Burgers' equation [6]:

$$\begin{aligned} &u_t + uu_x = \epsilon u_{xx}, \quad x \in [0,1], \quad t \in [0,2], \\ &u(0) = u(1) = 0, \\ &u(x,0) = \sin(2\pi x) + \frac{1}{2}\sin(\pi x). \end{aligned} \tag{16}$$

Initial and boundary conditions are chosen such that the solution develops a steep layer of width ϵ, which subsequently moves towards $x = 1$. Because of the boundary values, the wave amplitude diminishes over time. It is desirable for our numerical method to add mesh nodes as the steep gradient develops, and to remove mesh nodes as the wave amplitude diminishes.

A semi-discretization of Burgers' equation (16) using centered finite differences in space is given by

$$\begin{aligned} \frac{du_i}{dt} &= -\frac{1}{2}\left(\frac{u_{i+1}^2 - u_{i-1}^2}{x_{i+1} - x_{i-1}}\right) + \epsilon u_{i-1}\left(\frac{2}{x_{i-1} - x_i}\right)\left(\frac{1}{x_{i-1} - x_{i+1}}\right) + \dots \\ \epsilon u_i &\left(\frac{2}{x_i - x_{i-1}}\right)\left(\frac{1}{x_i - x_{i+1}}\right) + \epsilon u_{i+1}\left(\frac{2}{x_{i+1} - x_{i-1}}\right)\left(\frac{1}{x_{i+1} - x_i}\right). \end{aligned} \tag{17}$$

Solution on a fine uniform mesh

When ϵ is small, e.g. $\epsilon = 10^{-4}$, a moving mesh or a fine uniform static mesh is needed to resolve the steep front. Even with a fine uniform static mesh of 2001 points, oscillations are still visible in the solution. The `ODE15i` integrator in MATLAB is used to compute the solution shown in Figure 8.

Solution using a moving mesh with rezoning.

Using the rezoning approach, the mesh and physical solution are solved alternately. The arclength monitor function (2) with $\beta = 10^{-2}$ is used. Additionally, as noted in [10], spatial mesh smoothing is generally necessary to keep the mesh from varying too rapidly. This is accomplished by smoothing the monitor function using

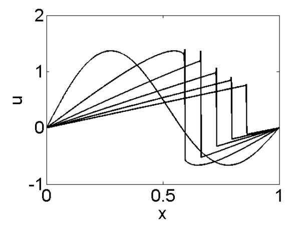
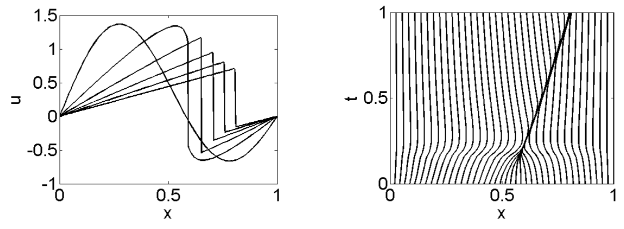


Fig. 8: Solution to Burgers' equation at $t = 0, 0.2, 0.4, 0.6, 0.8, 1.0$, obtained using a static uniform mesh of 2001 mesh points, with $\epsilon = 10^{-4}$.

$$\rho_j = \frac{\displaystyle\sum_{k=-p}^{p} \gamma^{|k|} \rho_{j+k}}{\displaystyle\sum_{k=-p}^{p} \gamma^{|k|}}, \quad j = 1, \ldots, N-1$$

with $\gamma = \frac{2}{3}, p = 1$.

Figure 9 shows the updated mesh is computed by (i) evolving the level set function using the discretized LMPDE4 (15) coupled with boundary conditions (12), then (ii) finding the iso-contours using linear interpolation. The physical solution is interpolated from the old mesh to the new one using linear interpolation. Then, the physical solution is updated to the new time level using an IMEX discretization of equation (17). Finally, the level set function is re-initialized, and the process repeated for the next time level. A total of 81 mesh nodes are used in the computation of Figure 9, although only 41 mesh nodes are shown in Figure 9(b) for visualization purposes. Our LMPDE formulation clusters mesh nodes about the developing and propagating layer, and correctly finds the solution to Burgers' equation.

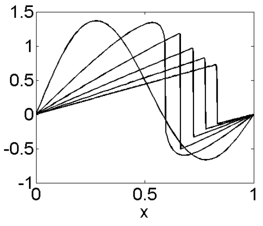


(a) Solution to Burgers' equation

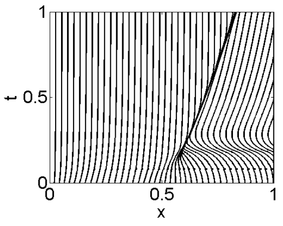
(b) r adaptive mesh

Fig. 9: Solution to Burgers' equation using the rezoning approach. An r adaptive mesh is obtained by using the boundary condition (12) together with the discretized LMPDE (15).

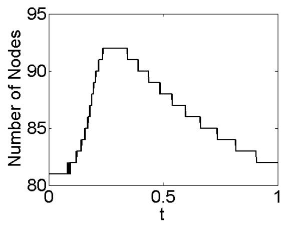
In Figure 10, the $h - r$ formulation, where the discretized LMPDE4 (15) is solved with boundary conditions (13), also clusters mesh nodes about the developing and propagating layer, and correctly finds the solution to Burgers' equation, using the rezoning approach. The simulation starts with 81 mesh nodes. The number of nodes varies during the simulation, as depicted in Figure 10(c).



(a) Solution to Burgers' equation



(b) $h - r$ adaptive mesh



(c) Number of mesh nodes

Fig. 10: Solution to Burgers' equation using the rezoning approach. An $h{-}r$ adaptive mesh is obtained by using the boundary condition (13) together with the discretized LMPDE (15).

5 Conclusions

In this paper, we have constructed, discussed and implemented a new moving mesh h-r hybrid method for solving time dependent partial differential equations. The novelty of the framework is that it enables the addition or removal of mesh nodes in a smooth manner, consistent with r-refinement schemes. The underlying mechanism is an implicit representation and evolution of a mesh using level set functions. This scheme is capable of solving problems which are otherwise difficult to solve using a static uniform mesh, or a moving mesh with a fixed number of nodes. Ongoing research considers the extension of the hybrid hr refinement to $\mathbb{R}^2$, as well as the application of this method to problems in plasma physics.

References

1. H. Askes and A. Rodríguez-Ferran. A combined hr-adaptive scheme based on domain subdivision. formulation and linear examples. *International Journal for Numerical Methods in Engineering*, 51(3):253–273, 2001.
2. Weiming Cao, Weizhang Huang, and Robert D. Russell. A moving mesh method in multiblock domains with application to a combustion problem. *Numer. Methods Partial Differential Equations*, 15(4):449–467, 1999.
3. P. J. Capon and P. K. Jimack. On the adaptive finite element solution of partial differential equations using h-r-refinement. Technical report, University of Leeds, 1996.
4. Carl de Boor. Good approximation by splines with variable knots. In *Spline functions and approximation theory (Proc. Sympos., Univ. Alberta, Edmonton, Alta., 1972)*, pages 57–72. Internat. Ser. Numer. Math., Vol. 21. Birkhäuser, Basel, 1973.
5. Carl de Boor. Good approximation by splines with variable knots. II. In *Conference on the Numerical Solution of Differential Equations (Univ. Dundee, Dundee, 1973)*, pages 12–20. Lecture Notes in Math., Vol. 363. Springer, Berlin, 1974.
6. RJ Gelinas, SK Doss, and K. Miller. The moving finite element method: applications to general partial differential equations with multiple large gradients. *Journal of Computational Physics*, 40(1):202–249, 1981.
7. Weizhang Huang, Lennard Kamenski, and Jens Lang. A new anisotropic mesh adaptation method based upon hierarchical a posteriori error estimates. *J. Comput. Phys.*, 229(6):2179–2198, 2010.
8. Weizhang Huang, Yuhe Ren, and Robert D. Russell. Moving mesh methods based on moving mesh partial differential equations. *J. Comput. Phys.*, 113(2):279–290, 1994.
9. Weizhang Huang, Yuhe Ren, and Robert D. Russell. Moving mesh partial differential equations (MMPDES) based on the equidistribution principle. *SIAM J. Numer. Anal.*, 31(3):709–730, 1994.
10. Weizhang Huang and Robert D. Russell. Analysis of moving mesh partial differential equations with spatial smoothing. *SIAM J. Numer. Anal.*, 34(3):1106–1126, 1997.
11. Weizhang Huang and Robert D. Russell. *Adaptive Moving Mesh Methods*, volume 174 of *Applied Mathematical Sciences*. Springer, New York, 2011.
12. Jens Lang, Weiming Cao, Weizhang Huang, and Robert D. Russell. A two-dimensional moving finite element method with local refinement based on a posteriori error estimates. *Appl. Numer. Math.*, 46(1):75–94, 2003.
13. Guojun Liao, Feng Liu, Gary C. de la Pena, Danping Peng, and Stanley Osher. Level-set-based deformation methods for adaptive grids. *J. Comput. Phys.*, 159(1):103–122, 2000.
14. M.D. Piggott, C.C. Pain, G.J. Gorman, P.W. Power, and A.J.H. Goddard. h, r, and hr adaptivity with applications in numerical ocean modelling. *Ocean Modelling*, 10(12):95 – 113, 2005.
15. Joe F. Thompson, Bharat K. Soni, and Nigel P. Weatherill, editors. *Handbook of Grid Generation*. CRC Press, Boca Raton, FL, 1999.
16. Han Wang and Huazhong Tang. An efficient adaptive mesh redistribution method for a non-linear Dirac equation. *J. Comput. Phys.*, 222(1):176–193, 2007.

17. Li Yuan and Tao Tang. Resolving the shock-induced combustion by an adaptive mesh redistribution method. *J. Comput. Phys.*, 224:587–600, 2007.