An Adaptive Moving Mesh Method with Static Rezoning for Partial Differential Equations*

James M. Hyman[1], Shengtai Li[1,2], Linda R. Petzold[2]

[1] *Theoretical Division, Mail Stop, B284, Los Alamos National Laboratory*
Los Alamos, NM 87545

[2] *Department of Computer Science, University of California*
Santa Barbara, CA 93106

December 18, 2001

Abstract

Adaptive mesh methods can improve the accuracy and efficiency in the numerical solution of evolutionary systems of partial differential equations. If the mesh moves to track fronts and large gradients in the solution, then larger time steps can be taken than if it were to remain stationary. We derive explicit differential equations for moving the mesh so that the time variation of the solution at the mesh points is minimized. Moving the mesh based on this approach allows for larger time steps but does not guarantee that the solution is well resolved in space. We maintain spatial accuracy when there are new emerging layers or wave fronts by adaptively rezoning the mesh points to equidistribute an error estimate. When using a multistep integration method the past solution values are also interpolated so that the same multistep method can be used after rezoning. The resulting algorithm has very few problem-dependent numerical parameters and is appropriate for a large class of one-dimensional partial differential equations. We illustrate the performance of the algorithm by examples, and demonstrate that the proposed algorithm is efficient and accurate when compared with other adaptive mesh strategies.

*This work was partially supported by DOE contract number DE-FG03-00ER25430, NSF grant CCR-9896198, NSF/ARPA PC-239415, and NSF ACI-0086061.

Keywords: Moving mesh, static rezone, partial differential equations, h-r method, adaptive mesh method, initial value problems.

1 Introduction

We combine two adaptive mesh strategies to globally distribute the mesh points to improve both accuracy and efficiency when solving time-dependent partial differential equations (PDEs). The moving-mesh (MM) method (also called an r-adaptive method) dynamically moves the grid in time [17, 9] to minimize the numerical errors. The MM methods based on minimizing the time rate of change of the solution [10, 12, 16] define a mesh velocity in terms of the time and space derivatives of the solution. The solution is changing slowly in this MM reference frame, and larger time steps can be taken without sacrificing accuracy. However, the mesh distribution may quickly fail to adequately resolve the solution. When this happens, it is convenient to use a static rezone (SR) method [13] (also called an h-adaptive method [7]) to regain spatial accuracy. The SR method freezes the solution and defines a new mesh that reduces the spatial truncation errors, and then interpolates the solution from the old mesh to the new one. This combination of the MM and SR methods (the MMSR method) has the potential of being more accurate and efficient than either approach alone [12, 18].

When integrating the solution with a multistep time integration solver during the interpolation step in a SR method [16, 8], the efficiency of the computation can be improved by interpolating the past values of the solution to the new mesh rather than restarting the integration with a one-step method. We extend this approach (sometimes called a *warm restart* [3]) to reduce the impact of the SR on the error estimates for the ordinary differential equation (ODE) or differential algebraic equation (DAE) solver. This can often allow the solver to continue the integration with a higher order multistep method and/or a larger time step after the SR.

After reviewing the MM and SR methods that we will be using, we describe how to improve the methods and provide a detailed description of our algorithm. We then demonstrate the effectiveness of combining the MMSR method with a finite difference method (FDM).

2 The moving mesh equations

Consider the scalar PDE

$$u_t = f(u, u_x, u_{xx}), \tag{1}$$

where f is a nonlinear function of u and its spatial derivatives. For example, a scalar reaction-diffusion equation could be expressed as

$$f(u) = \mu u_{xx} + h(u). \tag{2}$$

Transforming the spatial coordinates to a moving grid system, Eq. (1) becomes

$$\dot{u} - u_x \dot{x} = f(u, u_x, u_{xx}), \tag{3}$$

where $\dot{u}$ and $\dot{x}$ are the rates of change of u and x in the moving coordinate system.

The grid velocity $\dot{x}$ can be chosen to minimize the time variation of u and x in the new coordinates,

$$\begin{aligned} \min_{\dot{x}}[|\dot{u}|^2 + \alpha|\dot{x}|^2] = & \min_{\dot{x}} \Big[\sum \dot{u}^2 + \alpha \dot{x}^2 \Big] \\ = & \min_{\dot{x}} \Big[\sum (f + u_x \dot{x})^2 + \alpha \dot{x}^2 \Big]. \end{aligned}$$

Solving this equation locally for $\dot{x}$ at each mesh point gives

$$\alpha \dot{x} + \dot{u} \cdot u_x = 0, \tag{4}$$

or equivalently

$$\dot{x} = \frac{-f(u, u_x, u_{xx}) \cdot u_x}{\alpha + u_x \cdot u_x}. \tag{5}$$

This strategy minimizes the change in u and x, but the mesh points can easily cross when solved in discrete time steps. Consider a steep moving front where a mesh point in the front moves at a velocity that is nearly equal to the speed of the front, while a mesh point at the leading edge of the front doesn't move because the solution is flat there. Unless the time steps are made sufficiently small, the mesh point moving with the front will cross the stationary point.

Petzold [16] adds a diffusion-like term to the MM equations that has the effect of smoothing out the difference in mesh velocities at adjacent points,

$$\alpha \dot{x} + \dot{u} \cdot u_x - \lambda(\dot{x})_{xx} = 0, \tag{6}$$

where $\lambda > 0$. With these smooth mesh velocities the mesh points are unlikely to cross on a time step and the MM method can take larger time steps. Petzold also determined that the penalty term in Eq. (6) is important even when mesh points are deleted and moved apart by the SR strategy after every time step. Also note that Equation (6) is not invariant under scaling and translation of u and x.

The MM and SR methods should be, as nearly as possible, invariant under scaling and translation $\tilde{x} \leftarrow ax + b$, $\tilde{u} \leftarrow cu + d$ to be reliable when the PDEs are not well scaled. In the MM method, we minimize a weighted l_2 norm,

$$(\dot{x}/w_x)^2 + \sum_{i=1}^{N_{PDE}} (\dot{u}_i/w_i)^2 \tag{7}$$

where N_{PDE} is the number of PDEs in the original system, and w_i are weights. Note that the weight w_x takes the place of the scaling constant α in Eq. (6). We define $w_x = \alpha(x_N - x_0)$,

where normally $\alpha = 1$, or $\alpha = 0.1$ if w_x needs to be decreased compared with w_i. Then the weighted version of Eq. (6) at x_j is given by

$$\frac{\dot{x}_j}{w_x^2} + \sum_{i=1}^{N_{PDE}} \frac{\dot{u}_{i,j} u_{xi,j}}{w_i^2} - \lambda \left(\frac{\dfrac{\dot{x}_{j+1} - \dot{x}_j}{x_{j+1} - x_j} - \dfrac{\dot{x}_j - \dot{x}_{j-1}}{x_j - x_{j-1}}}{x_{j+1} - x_{j-1}} \right) = 0. \tag{8}$$

We define the weights w_i by

$$w_i = \max(|\max\{u_i\} - \min\{u_i\}|, floor(i)), \tag{9}$$

where the maximum and minimum are taken over all the mesh points. It is easy to check that if $floor(i)$ is scaled appropriately, the MM equation is approximately invariant under scaling and translation of u and x. The choice of $floor(i)$ can be especially important for components that start out initially flat and later develop gradients. Generally $floor(i)$ can be chosen as $\max(|\max\{u_i\} - \min\{u_i\}|)$ where u_i is the estimate of solution of the steady state.

The MM system, Eqs. (3) and (8), can be discretized by a finite element, collocation, finite volume, or finite difference method. In the numerical experiments presented here, all of the spatial derivatives are discretized by three-point FDMs.

3 The static rezone algorithm

Moving the mesh according to Eq. (8) can result in many fewer time steps than with a fixed mesh. A disadvantage of MM equations based only on the time variation of the solution is that the grid points may not move to where they are needed to reduce the spatial errors and may be overly concentrated in one region and absent in another region where they are needed. To overcome this deficiency, the SRs are needed to redistribute the mesh points.

The simplest and safest approach is to perform a SR after every time step [16]. However, a considerable disadvantage of the SR is that it necessitates interpolation and interrupts the time stepping process. Frequent interpolation may damage the accuracy considerably, while interruption of the time-stepping process triggers a time-consuming restart situation for the stiff solver (in our case the BDF solver DASSL [4]). In this section, we discuss how to reduce the number of SRs and their impact on the time integration.

3.1 Identifying when to rezone

The SRs can drastically affect the efficiency of the temporal integration, so a SR should be performed only when it is needed. Thus we evaluate the mesh function every K-th time step

(K to be prescribed) or at prescribed times [8] and perform SR only if the mesh function is outside some prescribed bounds.

Because time steps will be smaller when the solution is changing rapidly, we choose to monitor the mesh function and consider a SR after a specified number (K_{sr}) of temporal integration steps rather than at fixed times. A similar strategy used by Adjerid and Flaherty [2] found that $K_{sr} = 4$ was a reasonable choice for a wide range of problems. In our experiments with the MMSR algorithm we found that K_{sr} could be much larger and still retain the same accuracy. This was especially the case when the solution had converged to a traveling wave or self-similar solution.

When a new internal layer or wave front emerges, more mesh points are needed to maintain the same accuracy in the solution. When a wave diminishes, fewer points are needed to resolve the wave and can be deleted without loss of accuracy. These two situations where a SR is needed can be identified by monitoring the change in total variation (TV) of the solution at time t_n, which is calculated by

$$TV^n = \sum_{j=1}^{N-1} ||u_{j+1}^n - u_j^n||_1 = \sum_{i=1}^{N_{PDE}} w_i \sum_{j=1}^{N-1} |u_{i,j+1}^n - u_{i,j}^n|. \tag{10}$$

Here we have assumed that all of the solution components have been scaled. We investigated estimating the total variation of the spatial derivatives u_x instead of u to capture these changes in the wave fronts but found that this approach was not effective. To signal the need for a SR in these situations, we evaluate the TV by equation (10) on every time step, and compare it with the value of the previous one (the value for the last SR or initial solution).

The relative change of the total variation

$$RTV = 2.0|(TV^n - TV^{n+1})|/(TV^n + TV^{n+1}),$$

measures the change in the TV of the solution. The mesh function is evaluated and a static rezone may be considered only if $RTV \geq 20\%$ (0.2) or $RTV \leq 50\%$ (0.5). These two choices were determined experimentally to give the best performance over a wide range of problems.

Another case when a SR is needed is when the solution has converged to a traveling wave or self-similar solution. In this case, the TV does not change much and the MM Eq. (8) slowly causes the points to drift from their optimal positions. A SR is needed to pull the mesh points back to improve the spatial accuracy and resolution in those areas. We solve this problem by applying a SR after a fixed number (K_{sr}) of temporal integration steps.

The size of K_{sr} depends on the average mesh velocity $\overline{\dot{x}}$ over all of the mesh points and the maximum allowable spacing $\Delta x_{\max}$ of the grid. For a general problem, K_{sr} can be estimated by

$$K_{sr} \approx \Delta x_{\max}/|(\overline{\dot{x}}\Delta t)|, \tag{11}$$

where Δt is the current time step. We also have an option to use a constant K_{sr}, which is prescribed by the user, in our software. Generally, for problems which do not have emerging layers or waves fronts, K_{sr} can be very large (e.g., $K_{sr} = 100$); for problems where emerging layers or wave fronts are generated constantly, K_{sr} should be small (e.g., $K_{sr} = 5$).

To avoid unnecessary SRs for solutions that start smooth and later develop steep gradients, we do a SR only after the TV exceeds a threshold value. In all of our test examples, we set the threshold for TV to be 0.5 after scaling.

When discontinuities (weak solutions) occur in the solution of PDEs, some FDMs (for example, central difference) are unable to resolve them, and introduce small spurious oscillations near the discontinuity which increase the TV and trigger a SR. The refined grid will quickly eliminate the oscillations, and the refined grid near the discontinuity will be coarsened. On the coarser grid, the oscillations will reappear and start the cycle again.

A simple but effective approach to smooth a discontinuity is to replace the solution by the averaged value

$$u_j \leftarrow \frac{1}{2}\left(u_j + \frac{(x_j - x_{j-1})u_{j+1} + (x_{j+1} - x_j)u_{j-1}}{x_{j+1} - x_{j-1}}\right). \tag{12}$$

The smoothing is performed only before the mesh function is computed. A much better approach for hyperbolic PDEs is to use a high-resolution upwind FDM [15], or to explicitly include an artificial dissipation term in the equations.

3.2 Equidistributing the mesh function

After a SR has been called for, the next step is to redistribute the grid points based on the information about the current solution. Almost any static rezone method can be used in this step. We used a global equidistribution method. That is, we compute a mesh function for the current grid and then equidistribute it to find the new locations of the grid points.

How to choose the mesh function for a SR is not as critical as for the moving mesh [14]. In all of the examples in Section 4, we chose the monitor function based on the first and second derivatives of the solution

$$M(u, x) = \sqrt{\alpha_1 + u_x^2 + \alpha_2 N(\Delta u_x)^2}, \tag{13}$$

where α_1, α_2 are user-defined parameters, and N is the number of grid cells. Then the mesh function in each cell and the new number of grid points are defined as

$$m(u, x) = \Delta x M(u, x), \qquad N_{new} = \sum m(u, x)/TOL. \tag{14}$$

The parameter TOL is a user-defined tolerance.

The parameters α_1 and α_2 in (13) are easy to choose for a general problem. For example, we choose $\alpha_1 = 0.1$, $\alpha_2 = 0$ for a hyperbolic problem with shocks, and we choose $\alpha_1 = \alpha_2 =$

0.1 for a reaction-diffusion problem. The parameter TOL in (14) determines the number of nodes needed between two adjacent SRs, and is difficult to choose for general problems. We can avoid using TOL if the initial number of nodes is specified by the user. We increase or decrease the number of nodes based on the relative change in the total variation. That is, the number of nodes for a new SR grid is computed by

$$N_{new} = N_{old} \frac{TV_{new}}{TV_{old}},$$

where N_{old} is the number of nodes before the SR. Because FDMs are more accurate when the grid spacing is smoothly varying, we apply a global smoothing step [15] as part of the SR algorithm.

To reduce the number of times the solution is interpolated, the number of mesh points is changed only if the sum of $m(u, x)$ increases by at least ten percent or decreases by at least twenty percent. The local optimal mesh spacing is obtained by equidistributing $m(u, x)$ by inverse interpolation [13]. Next, the solution is interpolated onto the new mesh.

3.3 Restarting after a static rezone

After a new grid is generated and the solution has been interpolated from the old mesh to the new one, the simplest approach would be to restart the time integrator as though solving a new problem. This is called a *full restart* by Berzins et al.[3] and is appropriate for single-step time integration methods such as Runge-Kutta methods. For multistep methods, a full restart would cause the ODE/DAE solver to choose the lowest order single-step method and to reduce the time step size to satisfy the error tolerance of the lowest order method.

In a warm restart (or flying restart [3]) the history array used by the ODE/DAE solver is also interpolated to the new mesh, and the integration is continued with the same step size and order as would have been used had the SR not taken place. Because the number of equations may have changed in the SR and the Jacobian matrix is difficult to interpolate accurately, we always reevaluate the Jacobian matrix in a warm restart. We typically use cubic Hermite interpolation in our SR and occasionally apply a monotonicity limiter [11].

When the interpolation is not accurate, the implicit ODE solver might still choose a first order method after some error test failures because of the large residuals. The mesh velocities are difficult to interpolate accurately if they are not smoothly varying in space. When the mesh velocities are smooth functions of x, the integration will continue after a warm restart with almost the same time step and order as though the mesh had not been changed. The regularization in Eq. (8) smoothes the mesh velocities.

For problems with very steep wave fronts, the interpolation errors in a warm restart may not be sufficiently small and may cause the ODE solver to reduce the stepsize and/or order. In our experience, even if the ODE/DAE solver eventually restarts with the first-order method, the time stepsize is much larger than that for a full restart. Only when the mesh

velocities are very rough, or when there were insufficient mesh points to adequately resolve the solution, did we find that it was more effective to do a full restart.

4 Numerical experiments

Many of the MM methods that have been proposed in the literature [9, 13, 15, 17] are based on equidistributing a measurement (e.g., arclength or curvature) of the solution. These equidistributing MM (ED-MM) methods have a fixed number of grid points and can accurately solve many problems without a static rezone procedure, including problems in which new wave fronts or sharp gradients develop. In this section we will compare the MMSR method with the best performance of the Dorfi and Drury (DD) [6] and moving mesh PDE (MMPDE6) [9] methods. In these ED-MM methods, when the shape of the solution changes quickly during a simulation, mesh points may have to move a large distance to equidistribute the spatial error. When this happens, the MMSR method is more efficient because grid points are removed from over-resolved regions and new mesh points are added to unresolved regions by the SR.

The diffusion coefficient λ in Eq. (8) (which determines the smoothness of the mesh velocity and also affects the minimum spacing during the mesh moving) and the time-step interval K_{sr} between two static rezones are two of the most important parameters in determining the effectiveness of the MMSR method.

For problems that require small spacing to resolve shock waves, we chose $\lambda = 0.01$. In our examples, this choice allows the grid spacing to shrink down to 10^{-5}. For problems with smooth solutions, we chose $\lambda = 0.2$ and observed that the grid movement was smoother and that the warm restart technique worked better than $\lambda = 0.01$.

K_{sr} can be estimated by Eq. (11) if the solution structure changes rapidly. However, for a solution that converges to a steady-state, K_{sr} can be much larger than (11). We selected $K_{sr} = 100$ (unless it is stated otherwise) in our software. The warm restart solutions were interpolated with either cubic Hermite interpolation (the MMSR(WR-M) method) or monotone cubic Hermite interpolation (the MMSR(WR-C) method). There were no convergence test failures of the DAE solver in any of these examples, unless noted.

The resulting ODE systems are solved using the double precision version of DASSL [4], where an approximate Jacobian is computed by DASSL internally using finite differences. The relative and absolute local time stepping error tolerances (in a root-mean-square-norm) are chosen based on the requirements of the problems.

The acronyms used to describe the computational statistics and numerical methods are listed in Table 1.

MMM	Moving-mesh method
NSR	Number of SRs
NSTP	Number of time steps used
NRES	Number of residual evaluations (excluding Jacobian evaluations)
NJAC	Number of Jacobian evaluations
NCTF	Number of convergence test failures
NETF	Number of error test failures
NRJ	Number of residual evaluations (including Jacobian evaluations)
CPU	Total CPU time taken to solve the problem
K_{sr}	Maximum number of time steps between SRs
E_{L2}	L_2 difference between the numerical and exact solution
E_{max}	L_∞ difference between the numerical and exact solution
DD(N)	Dorfi and Drury method [6] with N grid points
MMPDE6(N)	Moving mesh PDE6 of Huang et al. [9]with N grid points
MMSR(WR-C)	MMSR with warm restart and cubic Hermite interpolation
MMSR(WR-M)	MMSR with warm restart and monotone cubic Hermite interpolation
MMSR(FR)	MMSR method with full restart

Table 1: Acronyms used in the tables describing the performance of the numerical methods.

4.1 Scalar combustion model

This example illustrates that for MM based on time variation, SR is crucial during the formation of steep fronts in reaction-diffusion equations. It is demonstrated that reducing the number of SRs can greatly improve the efficiency of the MMSR method, and that a warm restart is more effective than a full restart when the mesh moves smoothly.

Adjerid and Flaherty [1] evaluated their adaptive MM method on the single-step, reaction diffusion PDE,

$$\begin{aligned} &u_t = u_{xx} + D(2-u)\exp(-d/u), \quad 0 < x < 1, \quad 0 < t, \\ &u_x(0,t) = 0, \quad u(1,t) = 1, \quad u(x,0) = 1, \end{aligned} \tag{15}$$

where $D = 5e^d/d$. The solution is the temperature of a reactant that gradually increases from unity until a "hot spot" forms at $x = 0$ causing the temperature to rapidly increase to 2. A front forms and quickly propagates towards $x = 1$ with speed proportional to d.

In Fig. 4.1-a, the solution to Eq. 16 with $d = 20$ is shown developing a hot spot (ignition) at $t = 0.26$ and moving across the domain. We chose the relative and absolute error tolerances for DASSL to be 10^{-6}, approximated the spatial derivatives with a central three-point FDM, used the parameters $K_{sr} = 100$, $\lambda = 0.2$ (as in [16, 8]), and did not smooth the solution. There are three SRs, all between times 0.26 and 0.27 in the wave-forming phase. The number of grid points following each SR is 21, 26, and 32. The reference solutions are

computed with DASSL using 600 equally spaced non-moving grid points.

MMM	NSTP	NRES	NJAC	NETF	NRJ	CPU	E_{L2}	$E_{\max}$
MMSR(FR)	305	505	74	16	1024	0.932	0.0029	0.0150
MMSR(WR-C)	265	470	49	24	813	0.693	0.0012	0.0071
MMSR(WR-M)	290	509	49	25	852	0.741	0.0023	0.0133
DD(30)	452	876	72	56	1668	2.210	0.0013	0.0064
MMPDE6(30)	448	813	60	53	1953	2.452	0.0025	0.0118

Table 2: Results of the scalar combustion model with $d = 20$. The warm restart method gives a slight improvement in efficiency in this relatively smooth problem.

In Table 2 we compare the MMSR method (see Fig. 4.1-a) with the performance of the DD method [6] and MMPDE6 of Huang et al. [9] with 30 nodes. Note that the warm restart is more effective than the full restart, although only three SRs are performed. After the warm restarts, DASSL retained the same order and almost the same step size.

The wave front develops much faster when $d = 30$ (see Table 3 and Fig. 4.1-b). We decreased the error tolerances to 10^{-7}. We used $K_{sr} = 100, \lambda = 0.1$. We also tested with $K_{sr} = 50$ (see MMSR(WR-M)* in Table 3). The computation is much slower than when $K_{sr} = 100$. Since evaluating Eq. (8) for the mesh does not require additional mesh function evaluations, the average cost of one residual evaluation for the MMSR method is much less than for an ED-MM method. Thus, even when the number of residual evaluations (NRJ in the table) is larger for the MMSR method than for an ED-MM method, the CPU cost is less (see Table 3). The warm restart did not work as well as when $d = 20$; DASSL always warm restarted with a first order method after a SR. Comparing with the case of $d = 20$, we find that the mesh velocity for $d = 30$ is not a smooth function in space. This may be the reason why the integration fails to continue using a higher-order method after a warm restart. The apparent need to adjust the time steps after a SR causes a large number of error test failures (NETF in Table 3).

The static rezones are especially important to redistribute mesh points to where they are most needed during the formation of steep wave fronts, even when the number of grid points is fixed. We also verified that if the time variation MM method is used without the SR, the spatial resolution quickly degrades and the solution becomes very poorly resolved.

4.2 Burgers' equation.

The advantage of combining the MM with a SR method is particularly clear when solving equations where a shock wave forms from smooth initial data. The mesh points track the characteristics of the solution during the shock formation to improve the accuracy and efficiency of the method. This example demonstrates that a SR can quickly redistribute (e.g.,

MMM	NSR	NSTP	NRES	NJAC	NETF	NRJ	CPU	E_{L2}	E_{max}
MMSR(FR)	13	1195	1891	235	35	3536	3.12	0.0103	0.143
MMSR(WR-M)	13	1139	1807	202	88	3221	3.04	0.0089	0.123
MMSR(WR-M)*	43	2103	3309	476	204	6641	5.92	0.0101	0.140
MMSR(WR-C)	12	1104	1762	191	81	3099	2.84	0.0096	0.133
DD(41)	N/A	700	1244	67	28	1981	4.68	0.0092	0.128
MMPDE6(41)	N/A	890	1514	80	16	3034	5.76	0.0098	0.136

Table 3: Results of the scalar combustion model with $d = 30$. The warm restart method offered little improvement over the full restart method for this problem with a relatively steep solution. Although MMSR methods may take more time steps (NSTP) than the ED-MM methods, the CPU cost is often less because the cost for each step is much less for MMSR methods. The average number of nodes for the MMSR method is 45, compared to 41 for the ED-MM methods. Decreasing K_{sr} can increase the number of SRs, which can dramatically affect the computational efficiency. *This is for K_{sr}=50 (K_{sr}=100 for the others).

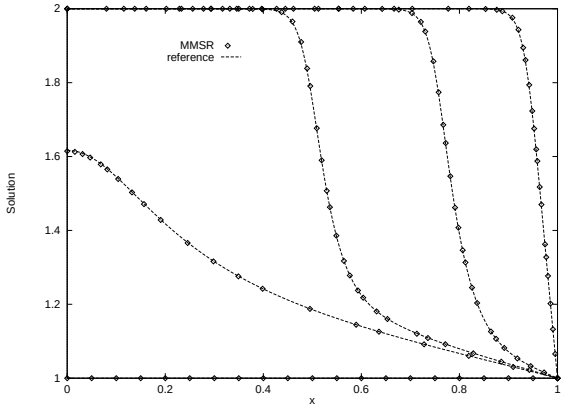

Figure 4.1-a: The solution of the combustion model Eq. (16) with $d = 20$ is shown at times $t =$ 0.0, 0.26, 0.27, 0.28, and 0.29. The MMSR methods prove to be an effective approach for solving the scalar combustion model.

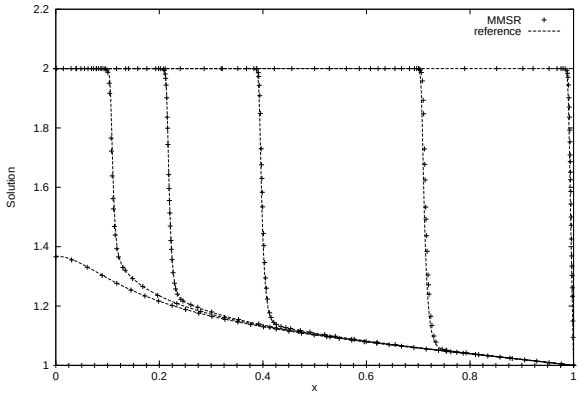

Figure 4.1-b: The solution of the combustion model with d=30 and K_{sr}=100 is shown at t= 0.0, 0.24, 0.2405, 0.2410, 0.242, 0.244, and 0.246. The mesh moving is not as smooth as for d=20. The MMM based on the time variation is inaccurate without the SRs.

add and/or delete) the mesh points to resolve the solution, and the integration is more efficient with the equidistribution done by a SR instead of an ED-MM. It also demonstrates that the K_{sr} should be kept small if new emerging layers or fronts are generated constantly and rapidly.

We consider Burgers' equation

$$u_t = -(u^2/2)_x + 0.0001u_{xx}, \quad x \in (0, 3) \tag{16}$$

with the initial condition

$$u(x,0) = \begin{cases} 1+\cos(2\pi x), & x \le 1 \\ 2, & x > 1. \end{cases}$$

The inflow left boundary $u(0,t) = 1+\cos(2\pi t)$, and the outflow right boundary are defined by first-order extrapolation. The initial solution (Figs. 4.2-a and 4.2-b) steepens and moves to the right until a viscous shock forms and propagates to the right boundary and disappears there. New waves keep on coming from the left boundary and are rapidly transformed into shocks and propagated to the right. We chose the relative and absolute error tolerance for DASSL to be 10^{-3}.

This example is a challenging problem for an ED-MM method, because when a shock disappears suddenly on the right boundary or a new shock forms rapidly near the left boundary, the equidistribution will be destroyed quickly unless the grid points are rapidly redistributed to maintain an ED mesh. The rapid redistribution of the mesh results in very small time steps and many convergence test failures in DASSL.

Note that the redistribution for the ED-MM method is done inside the integration by DASSL. However, for the MMSR method, this equidistribution is done occasionally, outside the integration. Using $K_{sr} = 5$, as estimated by (11), resulted in 73 SRs. Even with so many SRs, Table 4 shows that the MMSR with the warm-restart was the most efficient approach. The full restart (see Table 4) does not work well for this example.

MMM	NSTP	NRES	NJAC	NRE	NETF	NCTF	CPU	E_{L2}
MMSR(FR)	2709	6282	2533	24013	0	0	35.68	0.3363
MMSR(WR-M)	369	774	186	2076	27	13	3.23	0.0918
MMSR(WR-C)	531	970	232	2594	28	13	4.68	0.1035
DD(121)	1545	3644	781	12235	82	110	44.72	0.1397
DD(161)	1155	2673	593	9196	54	83	44.44	0.0929

Table 4: The MMSR method with warm restart is more efficient than the tuned DD methods for the solution of Burgers' equation. The MMPDE6 failed to solve this problem because of too many mesh crossings.

The solution in Fig. 4.2-b was obtained with a first-order centered FDM combined with local smoothing as in Eq.(12). Without the local smoothing operator of Eq. (12), spurious oscillations quickly appear when shocks form and increase the number of SRs. In this simulation, the parameter $\lambda = 0.01$ limited the minimum grid spacing to 0.00001. When $\lambda = 0.2$ we observed the minimum spacing to be 0.0012 and the shock wave was poorly resolved.

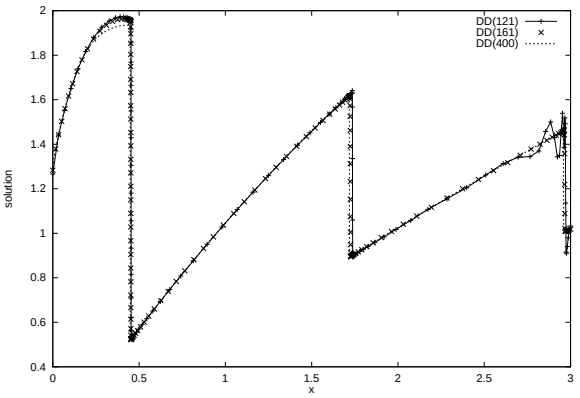

Figure 4.2-a: The results of Burgers' equation for Dorfi and Drury method ($t = 3.2$). The oscillation for DD(121) is because the equidistribution in the DD method requires additional points. Oscillations disappear when the number of nodes is increased to 161.

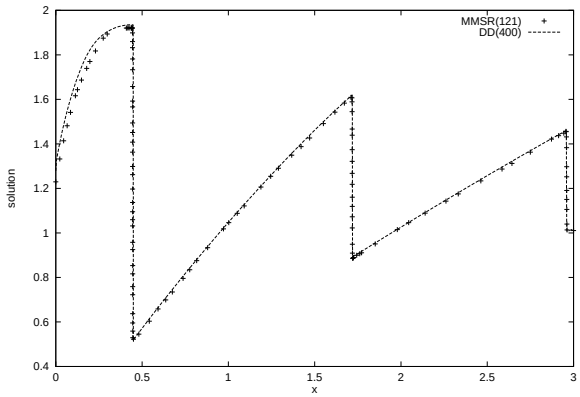

Figure 4.2-b: The results of Burgers' equation for MMSR method ($t = 3.2$). The average number of nodes is 123. Because the solution structure is varying rapidly (new shocks emerge and old shocks disappear constantly), we chose $K_{sr} = 5$.

4.3 Near steady-state convection problem

For some convection problems, the MMM will move the mesh points with the flow instead of where they are most needed to resolve the solution. In those cases, a SR is required occasionally to redistribute the mesh points.

Carlson and Miller [5] solved the convection equation

$$u_t = -u_x + b(x), \quad -4 \le x \le 4,$$

with the Dirichlet boundary condition $u(-4,t) = u(4,t) = 0$, to test the gradient- weighted moving finite element (GWMFE) method. The forcing function $b(x) = 0.1x\exp(-x^8)$ was chosen so that the steady state solution has sharp corners at $x = \pm 1$. This seemingly simple linear PDE has been used to illustrate the weaknesses of some adaptive MM methods [5]. The solution cannot be adequately resolved by a uniform grid of 80 nodes or by an ED-MM (DD or MMPDE with an arclength monitor) method with 80 nodes. The GWMFE [5] also fails to solve this problem, even after including a small diffusion term ϵu_{xx}.

We also added a small diffusion term, $0.0001u_{xx}$ to Eq. (4.3) and because the solution is so small, we set $floor = 0.05 \approx \max|u|$ in Eq.(9) and the absolute and relative error tolerances to 10^{-4}. The solution and mesh velocities are smooth (see 4.3) and we chose $\lambda = 0.2$, $K_{sr} = 100$. The average number of nodes was 72.

There were 14 SRs, 13 of which occurred during the wave formation and convergence to steady state. Near steady state, the nodes slowly migrate to the right and concentrate in the right side of the domain. The later SRs move the points back so that they are better

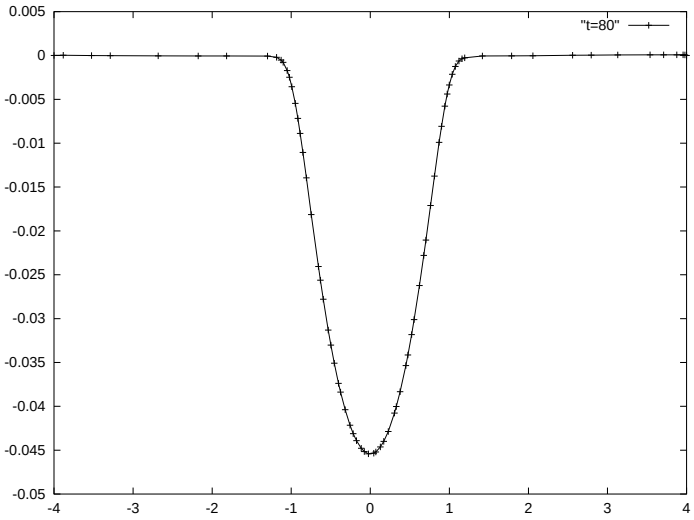

Figure 4.3: The MMSR method concentrates the mesh points in the regions of high curvature in the solution of the linear hyperbolic PDE (4.3) at time $t = 80$.

distributed. This behavior has also been observed for the scaled gradient weighted moving finite element method (SGWMFE) method [5].

Because the mesh velocity varies smoothly for this problem, it is more efficient to use warm restarts than full restarts (see Table 5).

MMM	NSR	NJAC	NRES	NSTP	NETF
MMSR(FR)	14	147	920	496	8
MMSR(WR-C)	14	76	727	396	12

Table 5: The warm restart was an effective approach in solving the linear hyperbolic PDE (4.3)to steady state.

5 Conclusion

We combine a MM method where the mesh velocities are defined to minimize the time variation of the solution with a static rezone method to distribute the grid points to reduce the spatial errors. This MM approach allows the time integration method to use large time steps without sacrificing temporal accuracy. Because the distribution of the mesh points chosen to minimize time accuracy may not resolve the solution well in space, a static rezoning of the mesh is needed to equidistribute the mesh and to reduce the spatial truncation error.

The numerical experiments verify the efficiency and robustness of the combined moving mesh-static rezone method for solving one-dimensional, time-dependent PDEs with moving wave fronts or shock waves. The parameters for the method are relatively easy to choose. The bandwidth of the linear equations for the MMSR method is less than that for the ED-MM approach.

Because the mesh function is evaluated only in the SR, which happens a few times during the integration, the average cost of one time step for the simple MMSR method is less than the ED-MM method where the mesh function is calculated at every residual evaluation. The overhead of restarting the time integration after a SR can be much reduced by the warm-restart techniques. Also, new emerging layers or wave fronts are more easily captured and resolved by the adaptive SR procedure than by an ED-MM method with a fixed number of grid points. Because the equidistribution for the MMSR method is performed independent of the time integration, it is more efficient for problems when the solution structures vary (e.g. new fronts emerge or old fronts disappear) constantly and rapidly.

Our MMSR method can be more efficient than other MM approaches for problems with new emerging layers or wave fronts, and for problems where the solution structures change frequently, and in situations where fast computation of steady-state solution is required.

References

[1] S. Adjerid and J. E. Flaherty, A moving finite element method with error estimation and refinement for one–dimensional time dependent partial differential equations, *SIAM J. Num. Anal.* 23 (1986), 778–795.

[2] S. Adjerid and J. E. Flaherty, A moving–mesh finite element method with local refinement for parabolic partial differential equations, *Comput. Meth. in Appl. Mech. Eng.* 55 (1986), 3–26.

[3] M. Berzins, P. J. Capon and P. K. Jimack, On spatial adaptivity and interpolation when using the method of lines, *Appl. Numer. Math.*, 26 (1998), 117–133.

[4] K.E. Brenan, S.L. Campbell and L.R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, Second Edition, SIAM, 1995.

[5] N. N. Carlson and K. Miller, Design and application of a gradient weighted moving finite element code, part I, in 1-D. *SIAM J. Sci. Comp.*, 19 (1998), 728–765.

[6] E. A. Dorfi and L. O'c. Drury, Simple adaptive grids for 1–D initial value problems, *J. Comput. Phys.* 69 (1987), 175–195.

[7] J. E. Flaherty and P. K. Moore, Integrated space-time adaptive *hp*–refinement methods for parabolic systems, *Appl. Numer. Math.* 16 (1995), 317–341.

[8] R. M. Furzeland, J. G. Verwer and P. A. Zegeling, A numerical study of three moving grid methods for one–dimensional partial differential equations which are based on the method of lines, *J. Comput. Phys.* 89 (1990), 349–388.

[9] W. Z. Huang, Y. Ren, and R. D. Russell, Moving mesh methods based on moving mesh partial differential equations, *J. Comput. Phys.* 113 (1994), 279–290.

[10] J. M. Hyman, Adaptive moving mesh methods for partial differential equations, in *Advances in Reactor Computations*, American Nuclear Society Press, La Grange Park, IL, 1983, 24–43.

[11] J. M. Hyman, Accurate monotonicity preserving cubic interpolation, *SIAM J. Sci. Stat. Comp.*, 4 (1983), 645-654

[12] J. M. Hyman and B. Larrouturou, Dynamic rezone methods for partial differential equations in one space dimension, *Appl. Numer. Math.* 5 (1989), 435–450.

[13] J. M. Hyman and M. J. Naughton, Static rezone method for tensor product grids, *Lectures in Applied Mathematics*, Vol. 22, 321–343, American Mathematical Society (1985).

[14] S. Li, Adaptive Mesh Methods and Software for Time Dependent Partial Differential Equations, Ph.D. Thesis (1998), Department of Computer Science, University of Minnesota.

[15] S. Li and L. Petzold, Moving mesh method with upwinding schemes for time–dependent PDEs, *J. Comput. Phys.* 131 (1997), 368–377.

[16] L. R. Petzold, Observations on an adaptive moving grid method for one-dimensional systems of partial differential equations, *Appl. Numer. Math.* 3 (1987), 347–360.

[17] J. G. Verwer, J. G. Blom, R. M. Furzeland, and P. A. Zegeling, A moving–grid method for one–dimensional PDEs based on the method of lines, in Adaptive Methods for Partial Differential Equations, J. E. Flaherty, P.J. Paslow, M. S. Shephard, and J. D. Vasilakis, eds., Society for Industrial and Applied Mathematics, Philadelphia, PA, 1989.

[18] A. V. Wouwer, P. Saucez, and W. Schiesser, Adaptive Method of Lines, CRC Press, 2001.