How to adaptively resolve evolutionary singularities in differential equations with symmetry

C.J. Budd*and JF Williams†

*Department of Mathematical Sciences, University of Bath,
Claverton Down, Bath, BA2 7AY, U.K.
mascjb@bath.ac.uk

†Department of Mathematics, Simon Fraser University,
Burnaby, BC, V5A 1S6, Canada.
jfw@math.sfu.ca

February 18, 2009
Manuscript submitted to J. Eng. Math.

Abstract

In this paper we review the theory of self-similar blow-up in evolutionary differential equations and present a method to simulate such phenomena numerically. Our method exploits the evolving symmetries in such problems to guide the adaptivity in both time and space. The focus of this paper is on the practical implementation with examples drawn from applications.

1 Introduction

The numerical solution of many nonlinear evolutionary differential equations requires some form of adaptivity both in space and time in order to generate reliable solutions efficiently. This is due to the difficult nature of the underlying physical phenomena with evolving structures on small time and length scales possibly manifesting as shocks, singularities, localization or moving interfaces. Regular, as well as even non-regular, but fixed temporal or spatial meshes are often unable to resolve this structure due to the transient character of such behaviour. Adaptive meshes allow enhanced resolution in certain regions of the time-space domain, capturing natural time and length scales and often allowing for the computations to be extended where non-adaptive methods breakdown. There are typically three approaches to mesh adaptivity known as p, h and r [30]. While combinations of the different approaches are possible, dynamic $r-$adaptivity is the focus of this paper, which aims to describe an efficient $r-$adaptive method to simulate evolving singularities Whilst not as widely used as h or p adaptive methods, $r-$adaptivity has been used with success in many applications including computational fluid mechanics [30], phase-field models and crystal growth [23], and convective heat transfer [16]. It also has a natural application to problems with a close coupling between spatial and temporal length scales, such as in problems with symmetry, scaling invariance and self-similarity [1, 10], where the mesh points $X_i(t)$ become the *natural coordinates* for an appropriately rescaled problem and the adaptive method inherits the natural underlying dynamics of the solution.

In this paper, for the sake of exposition, we focus on partial differential equations, and systems, which are posed in one space dimension and acted on by scaling symmetry groups. In particular we show that $r-$adaptivity has a natural interpretation in the context of such equations, and that solution monitors driven by considerations of symmetry, lead to highly effective adaptive meshing procedures. We give a practical guide to both the choice of the (optimal) adaptive mesh (in both space and time) and also effective discretizations of the underlying PDE on such a (moving) mesh. This will then be illustrated by a number of example computations. The general scaling and adaptive techniques that we describe extend to higher dimension with some additional theoretical machinery and in Section 6 we present one two dimensional example. For further details on higher-dimensional adaptivity see [16, 12, 13, 17].

2 Singularities and symmetry structures in PDEs

In this section we look at some of the symmetries which occur in the solutions of PDEs which are invariant under the action of certain scaling symmetry groups. Many nonlinear evolutionary differential equations, have the property that for certain initial data they develop *blow-up* solutions $u(x,t)$ which become singular

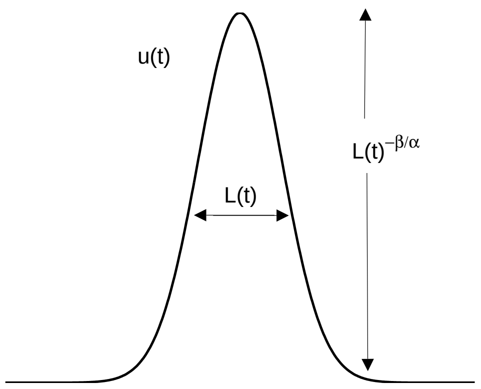

Figure 1: A typical peak, indicating the scaling relationship as the singularity develops for a self-similar solution.

at a single point x^* in a finite time T. Typically such solutions develop a peak of amplitude $U(t) \to \infty$ and width $L(t) \to 0$ as $t \to T$ as illustrated in Figure 1. Examples of PDEs with this property include: the semilinear heat equation with polynomial nonlinearity

$$u_t = u_{xx} + u^p, \qquad p > 1, \tag{1}$$

the nonlinear Schrödinger equation when posed in dimension $d \geqslant 2$,

$$iu_t = \nabla^2 u + |u|^2 u, \tag{2}$$

the long-wave unstable thin-film equation

$$h_t = -(hh_{xxx} + h^p h_x)_x, \quad p > 3/2, \tag{3}$$

and the semilinear wave equation

$$u_{tt} = \nabla^2 u + |u|^{p-1} u, \qquad p > 1. \tag{4}$$

Related to these are models which exhibit derivative blow-up such as the radially symmetric harmonic map heat flow

$$\theta_t = \theta_{rr} + \frac{1}{r}\theta_r - \frac{\sin 2\theta}{2r^2} \tag{5}$$

and an example of pinch-off in a model for thin-films

$$h_t = -(h^p h_{xxx})_x, \qquad 0 < p < 1. \tag{6}$$

Asymptotically the solution, or a derivative, often has the separated asymptotic form

$$u(x,t) = U(t)V(\xi), \quad \xi = (x - x^*)/L(t), \tag{7}$$

in the peak when $|x - x^*|$ is small. Here $U(t) \to \infty$ as $t \to T$ and V is a regular function of the scaled variable ξ. This scaled variable represents a *natural coordinate* in which to express the solution and to use for numerical calculations. An optimal r-adaptive method which correctly follows the underlying dynamics of such a PDE, should automatically distribute the computational mesh along these natural coordinates in *both* space and time. Our aim in this paper is to show how such meshes can be constructed and how the PDE can be discretized effectively on such a mesh.

A general class of partial differential equations which admit such singular solutions, of the form described in (7), are those with a scaling structure, so that the partial differential equation is invariant (at least locally) under transformations of the form

$$t \to \lambda t, \quad x \to \lambda^{\alpha} x, \quad u \to \lambda^{-\beta} u. \tag{8}$$

Each of the example equations considered above is invariant under the group rescaling (8). The exact structure of the rescaling depends on whether the equation is of semi- or quasi-linear and the exact form of the nonlinearities. For example, (1) is invariant under the group rescaling $t \mapsto \lambda t, \quad x \mapsto \lambda^{1/2} x, \quad u \mapsto \lambda^{-1/(p-1)} u$. Whereas the unstable thin-film equation (3) is invariant under $t \mapsto \lambda t, \quad x \mapsto \lambda^{(p-2)/2(2p-3)} x, \quad u \mapsto \lambda^{-1/(2p-3)} u$. These differential equations may then admit (separable) *self-similar* solutions which are themselves invariant under the action of the scaling transformation. These typically take the form

$$u(x,t) = (T-t)^{-\beta} V(x/(T-t)^{\alpha}), \tag{9}$$

and have a natural length and solution scale given by

$$L(t) = (T-t)^{\alpha}, \quad U(t) = L(t)^{-\beta/\alpha} \tag{10}$$

as illustrated in Figure 1. More generally if we introduces a *slow variable* $\tau = \log(T-t)$ (so that $\tau \to \infty$ as $t \to T$), and set

$$u(x,t) = (T-t)^{-\beta} v(y,\tau), \quad y = x/(T-t)^{\alpha} \tag{11}$$

then we may analyze the blow-up solutions by recasting the PDE in the rescaled variables [26] so that it takes the generic form

$$v_\tau = F(v, v_y, v_{yy}.) \tag{12}$$

For example in the case of (1) with a polynomial nonlinearity, this leads to a problem of the form

$$v_\tau = \nabla_y^2 v + v^p - \frac{1}{p-1} v - \frac{1}{2} y v_y. \tag{13}$$

Observe that if x is fixed and $t \to T$ then $y \to \infty$ and hence to be meaningful in the context of the underlying PDE, the rescaled equation is typically posed on the full or half line and must have certain decay conditions at infinity, often taking the form $\lim_{y \to \infty} v(y,\tau) v^q = constant$ for some $q > 0$. In contrast, using the same rescaled variables, the thin-film equation becomes

$$v_\tau = -(v v_{yyy} - v_y v^p)_y - \frac{1}{(2p-3)} v - \frac{p-3}{2(2p-3)} y v_y \tag{14}$$

In this case, the similarity profiles are defined on a *finite* interval (of possibly unknown length)

$$
\begin{aligned}
&0 = -(\varphi\varphi'' - \varphi'\varphi^p)' - \frac{1}{(2p-3)}\varphi - \frac{p-3}{2(2p-3)}y\varphi', \\
&0 = \varphi' = \varphi''' \text{ for } y = 0, \\
&0 = \varphi = \varphi' \text{ for } y = y^*.
\end{aligned}
$$

In this case, in the limit $t \to T$ we see that $x^* = (T-t)^p y^* \to 0$ and thus the blow-up profile collapses to the origin. Assuming single-point blow-up, the remainder of the solution remains $\mathcal{O}(1)$ in this limit and is thus fairly straightforward to resolve. This distinction makes one aspect of the numerical resolution of blow-up of semilinear problems more difficult than conservative quasilinear problems.

A similarity solution of the underlying PDE then corresponds exactly to a non-constant steady solution of the rescaled equation (12) satisfying the ODE

$$
F(v, v_y, v_{yy}) = 0 \tag{15}
$$

together with the boundedness conditions at infinity (so that they are homoclinic or heteroclinic solutions of such equations) or other appropriate boundary conditions. Such solutions exist in the case of the nonlinear Schrödinger equation in *three*-dimensions and many other examples of such problems are found in [1]. However in many problems the only solutions of (15) satisfying the growth conditions at infinity are zero or constant, and the self-similar profile does not adequately describe the asymptotic form of the singularity. This class of problems includes the semilinear heat equation in one dimension and chemotaxis systems and the nonlinear Schrödinger equation in two dimensions. To study the asymptotic form of the solutions of (11) we must then consider the *center-manifold* of the system in the neighbourhood of this zero (or constant) solution. This typically leads to *approximately self-similar solutions* for which the solution scale and/or the length scale have additional logarithmic terms so that (for example)

$$
L(t) = (T-t)^{\alpha} |\log(T-t)|^{\kappa}. \tag{16}
$$

Both of these cases will be considered in this paper, and we show that the scale invariant $r-$adaptive method will correctly distribute the mesh in both cases. Our basic goal is to show that $r-$ adaptivity based on symmetry mimics many of the features of studying the problem in the similarity co-ordinates. This is a natural transformation to make and is done routinely in the analysis of such problems, but, recall, that *the exact transformation is not known a priori.* The use of scaling allows the use of the *emerging* symmetry structure to guide the adaptive procedure without building the explicit transformation in to the numerical scheme. This allows for methods applicable to a broad class of problems and greater confidence in the observed dynamics.

3 Introduction to $r-$adaptivity for blow-up problems

3.1 Overview

When semi-discretizing a PDE of the form (for example) of (1) it is usual impose a fixed *spatial mesh* X_i with discrete solution values $U_i(t)$ satisfying $U_i(t) \approx u(X_i, t)$. When discretizing the PDE we then construct a set of ordinary differential equations for the unknowns $U_i(t)$ in which the spatial derivatives are replaced by terms involving U_i, typically constructed by finite difference, finite element, finite volume or collocation methods. For example, a simple finite difference approximation to (1) can take the form

$$\dot{U}_i = \frac{\frac{U_{i+1}-U_i}{X_{i+1}-X_i} - \frac{U_i-U_{i-1}}{X_i-X_{i-1}}}{\frac{X_{i+1}-X_{i-1}}{2}} + U_i^p. \tag{17}$$

The resulting ODES can then be solved by using appropriate ODE solution software such as the Matlab routine `ode15s`.

Two immediate criticisms can be made of the discretization (17) on a fixed mesh. From a numerical point of view, the discretization of any spatial derivatives will cease to be accurate (or indeed meaningful) if the spatial length-scale $L(t)$ is *smaller* than the smallest value of $H_i \equiv X_{i+1} - X_i$. This is an acute problem in any calculation of blow-up in which we may easily have $L(t) = 10^{-15}$. In such cases a fixed mesh would have to take $H_i < 10^{-15}$ and hence of the order of 10^{15} points would be needed in any calculation using a *uniform spatial mesh.* Of course this problem can be overcome if extra mesh points are added close to the singularity as the evolution of the solution proceeds. When done in the context of suitable a-posteriori estimates of the solution error, this leads to the h-adaptive method for such problems. Secondly, and more fundamentally, the use of such a fixed mesh destroys the scaling structure of the PDE and the discrete system (17) does not admit any scaling symmetries of the form (8) unless $\alpha = 0$.

A solution to *both* of these issues is to allow the mesh points $X_i \equiv X_i(t)$ to move with time as the solution evolves. By doing this we may, firstly, concentrate the mesh points into the region of the singularity so that the spacing of the points there can be very small, even if the spacing between other points is much larger. Secondly, we can create an extended system comprising both the solution and the mesh points which can then admit the scaling structure of the underlying PDE. In a moving mesh method we must prescribe equations which determine the motion of the mesh points. Typically these will take the form of ODES such as $\dot{X} = G(X_i, U_i)$, and we will explain how these can be derived in the next sub-section. The PDE is then discretized on this mesh using a finite difference, finite element or collocation method. Such a discretization will lead to a system of ODES of the form

$$A(\mathbf{X}, \mathbf{U})\dot{U}_i + B(\mathbf{X}, \mathbf{U})\dot{X} = F(\mathbf{X}, \mathbf{U}), \tag{18}$$

where $\mathbf{X} = (X_1, X_2, \ldots), \mathbf{U} = (U_1, U_2, \ldots)$. As an example, a discretization of (1) taking the same form as (17) but this time posed on a moving mesh takes the form

$$\dot{U}_i - \frac{U_{i+1}-U_{i-1}}{X_{i+1}-X_{i-1}}\dot{X}_i = \frac{\frac{U_{i+1}-U_i}{X_{i+1}-X_i} - \frac{U_i-U_{i-1}}{X_i-X_{i-1}}}{\frac{X_{i+1}-X_{i-1}}{2}} + U_i^p. \tag{19}$$

Here the additional term on the left hand side of this expression is an additional advective term which occurs because U_i is an approximation to $u(X_i(t),t)$ and we note that $du(X_i(t),t)/dt = u_t(X_i,t) + u_x\dot{X}_i$ so that $u_t = \frac{du}{dt} - u_x x_t$.

It is immediate that if the PDE (1) is invariant under the scaling transformation

$$t \to \lambda t, \quad x \to \lambda^\alpha x, \quad u \to \lambda^\beta u \tag{20}$$

then the discrete equation (19) has exactly the invariance $t \to \lambda t, \quad X_i \to \lambda^\alpha X_i, \quad U_i \to \lambda^\beta U_i$. This follows from the almost obvious observation that

the actions of linear scaling and discretization commute.

For example under the above scalings, the differential operator u_x scales as $\lambda^{\beta-\alpha}$ as does its finite difference approximation $(U_{i+1}-U_{i-1})/(X_{i+1}-X_{i-1})$. For this reason, if the underlying PDE has the scaling invariance (20) then so will the discretized system (18). As a direct consequence, there is a constant γ for which the various terms in (18) have the following scalings

$$\begin{aligned} F(\lambda^\alpha X_i, \lambda^\beta U_i) &= \lambda^\gamma F(X_i, U_i), \\ \lambda^{\beta-1} A(\lambda^\alpha \mathbf{X}, \lambda^\beta \mathbf{U}) &= \lambda^\gamma A(\mathbf{X}, \mathbf{U}), \\ \lambda^{\alpha-1} B(\lambda^\alpha \mathbf{X}, \lambda^\beta \mathbf{U}) &= \lambda^\gamma B(\mathbf{X}, \mathbf{U}). \end{aligned} \tag{21}$$

In the case of a fixed finite difference discretization, the operator A is constant, so that $\gamma = \beta - 1$

When evolving a PDE on a moving mesh it is usual to solve the two equations (18) and the mesh equation *simultaneously* so that they become one larger system. We require this to have the same symmetries as the underlying PDE. This then implies that the function G describing the mesh velocities must satisfy the functional equation

$$G(\lambda^\alpha X_i, \lambda^\beta U_i) = \lambda^{\alpha-1} G(X_i, U_i). \tag{22}$$

We will give examples of such functions in the next sub-section. We then observe that the symmetries of the PDE then lead to symmetries of the (extended set) of ODEs discretizing it on the moving mesh. (In Section 4 we will show that such ODEs can be discretized in a manner which respects this scaling structure by using the Sundman transformation.)

A significant advantage of doing this is that the combined system describing the solution and the mesh may then admit *discrete self-similar solutions* . It follows immediately from (21) and (22) that the combined ODE system for the solution and the mesh admits a *discrete self-similar solution* of the form

$$U_i(t) = t^{\beta} V_i, \quad X_i(t) = t^{\alpha} Y_i \tag{23}$$

where Y_i and V_i are constant in time and satisfy the discrete difference equations

$$\beta A(\mathbf{Y}, \mathbf{V}) V_i + \alpha B(\mathbf{Y}, \mathbf{V}) Y_i = F(Y_i, V_i), \quad \alpha Y_i = G(\mathbf{Y}, \mathbf{V}). \tag{24}$$

(Note that we can substitute $T-t$ for t in (23) if needed.) The equation (24) is a consistent discretization of the ODE satisfied by the true self-similar solution on the mesh Y_i. Thus the true shape of the self-similar solution will be captured by this method, even if the self-similar solution corresponds to a singular physical solution. More generally, following the discussion of the previous section, we can introduce a new slow variable $\tau = \log(t)$ (or $\tau = \log(T-t)$) and consider a solution in the form

$$U_i(t) = t^{\beta} V_i(\tau), \quad X_i(t) = t^{\alpha} Y_i \tag{25}$$

Substituting this then satisfies the ODE system

$$A(\mathbf{Y}, \mathbf{V})(\beta V_i + V_{i,\tau}) + \alpha B(\mathbf{Y}, \mathbf{V}) Y_i = F(Y_i, V_i), \quad \alpha Y_i = G(\mathbf{Y}, \mathbf{V}). \tag{26}$$

This is again a consistent discretization of the ODE system satisfied by the rescaled solution in the previous section and will admit an approximately self-similar solution in the same manner. The above discussion shows that if a singularity is described by a true or an approximate self-similar solution then an invariant $r-$adaptive method should be able to compute it. We now consider how to construct such a method.

3.2 Moving mesh PDES and equidistribution

Moving mesh PDES, MMPDEs based on equidistribution, give a coherent mechanism for constructing moving meshes with the symmetry properties required in the previous sub-section. To construct such methods we introduce a *computational coordinate* ξ in a suitable *computational* space Ω_C (typically the interval $[0,1]$) and consider the mesh points X_i to be the image of a uniform mesh in Ω_C under the map $x(\xi, t)$ from Ω_C into the *physical space* Ω_P (where the underlying PDE is posed) so that $X_i(t) = x(i\Delta\xi, t)$. Typically this is done so as to make the equation 'smoother' or better resolved in the computational space. In one dimension this procedure is well understood, being based on the *equidistribution* principle [5]. In it's integral form one poses a *monitor function* $M(x)$ and determines a bijective mapping $x : \Omega_c \to \Omega_p$ such that

$$\int_0^{x(\xi)} M(s) ds = \xi \int_0^a M(s) ds, \qquad \xi \in \Omega_c = [0,1]. \tag{27}$$

Any standard discrete approximation to (27) will then have the property that the integral of the monitor function is the same over each computational interval. This has the desirable consequence that if the monitor function is taken to be the local error then this method produces the *optimal* mesh with the total error being minimized. Another, more useful geometric interpretation comes from differentiating of (27) with respect to ξ,

$$M(x)x_\xi = \int_0^a M(x)dx, \quad \Rightarrow \quad (M(x)x_\xi)_\xi = 0. \tag{28}$$

to the physical variables.

3.3 Scale-invariant MMPDEs

A continuous in time adaptive strategy using equidistribution is to solve the physical PDE and the mesh equation simultaneously. This determines a coupled system of differential algebraic equations for the solution and the grid. In practice, this system is difficult to start and only neutrally stable in time. An extension of this is to use moving mesh PDEs (MMPDEs) whose steady-state solutions in time are identically (28). This parabolic regularization of (28) leads to smoother mesh trajectories in time and reduces overall stiffness in the DAE. Many different MMPDEs have been proposed and used and the important properties such as stability and prevention of node crossing are well understood. A widely used MMPDE is given by

$$-\varepsilon x_{t\xi\xi} = (M(x)x_\xi)_\xi, \qquad \varepsilon \ll 1. \tag{29}$$

which is known as MMPDE6 as described in [18]. The system (29) can then be discretized in terms of the computational coordinate to give a series of differential equations for X_i. For example a simple centered discretization gives

$$-\varepsilon \frac{\dot{x}_{i+1} - 2\dot{x}_i + \dot{x}_{i-1}}{\Delta\xi^2} = \frac{M_{i+1/2}(x_{i+1} - x_i) - M_{i-1/2}(x_i - x_{i-1}}{\Delta\xi^2}), \tag{30}$$

$$\text{where} \qquad M_{i+1/2} = \frac{M(x_{i+1}) + M(x_i)}{2}.$$

In this formulation the small parameter ε determines the relaxation timescale for convergence to the stationary profile (28) and the Laplacian acting on $\dot{x}$ helps control the smoothness of the right hand side (and thus reduce stiffness). The idea of scale invariant mesh adaptation is to choose a monitor function $M(u(x,t)) > 0$ so that (22) is satisfied. In terms of time integration this means that the grid and solution should evolve on the same timescale. This means that the mesh should never be seen to "freeze" but should continue to adapt as the the singularity is approached but should not adapt so rapidly that the system describing the mesh becomes overly stiff. For this we require that the monitor function satisfy the functional equation

$$M(\lambda^\alpha x, \lambda^\beta u, \lambda^{\beta-\alpha} u_x) = \lambda^{-1} M(x, u, u_x). \tag{31}$$

As an example, in ((1)) we have $\beta = -1/(p-1)$. If we consider $M(x,u,u_x) \equiv M(u)$ then from (31) we have simply that $M = u^{p-1}$. This approach to adaptivity has been successfully used to compute blow-up profiles in many cases, see [7],[11]. Unfortunately, it has the undesirable effect of absorbing all available grid nodes into the blow-up region as $t \to T$ and the computations often halt due to lack of resolution in the $\mathcal{O}(1)$ tail region, not the blow-up core! In this paper we shall use a small but vital alteration in order to better resolve all aspects of the solution

$$M_c \equiv M + \alpha \int_0^a M(x)dx. \tag{32}$$

The standard case with $\alpha = 0$ has been previously used in the references above for the computation of blow-up and with $\alpha = 1$ was developed for the computation of singularly perturbed boundary value problems in [22]. In the latter case approximately half of the mesh points are in the peak and half in the tail of the solution. In the case of blow-up problems M_c is asymptotically close to M in the peak region so that the symmetry properties of the solution are not affected there and the mesh will still reproduce self-similar types of behaviour.

By choosing M to satisfy (31) we do not directly use the local truncation error of our spatial discretization method. This approach would generate optimal grids in the sense of the smallest constant in an asymptotic error estimate. Unfortunately, this approach would require a very accurate representation of a high derivative of a singular solution which is not always available on non-uniform grids. Instead we use the scaling properties of the solution to let a well resolved aspect of the solution indicate where the grid should be refined. It should be noted however that this approach turns even semilinear PDEs into fully nonlinear systems and so its effectiveness must be in some way justified.

3.4 Temporal behaviour of scale invariant MMPDEs

When all quantities are order one, (29) is simply a parabolic regularization to (28). Because the parameter $\varepsilon \ll 1$ we would anticipate that the timescale of the mesh dynamics to be faster than the PDE on that mesh and hence that the mesh to be quasi-equidistributed at all times. However, this conclusion is not immediately clear in the case of blow-up where the quantities involved are not always order one and we conclude this section with a brief analysis of this case for the problem (1).

To do this we consider the effects of taking a more general monitor function which is an arbitrary power of u given by $M = |u|^q$. We now substitute the solution expressed in the scaled variables

$$(T-t)^{1/(p-1)}u(x,t) = v(y,\tau), \qquad x(\xi,t) = (T-t)^{1/2}y(\xi,\tau) = e^{\tau/2}y(\xi,\tau) \qquad \tau = -\log(T-t)$$

into (29). After some manipulation this gives

$$-\varepsilon\partial_{\xi\xi}\left(y_\tau - \frac{y}{2}\right) = (T-t)^{1-q/(p-1)}\left(v^q y_\xi\right)_\xi. \tag{33}$$

We consider three cases.

1) If $q < p - 1$, then as $t \to T^-$ the RHS of (33) is asymptotic to zero, and we have

$$-\partial_{\xi\xi}(y_\tau - y/2) = 0.$$

When coupled to the scaled boundary conditions $y(0, \tau) = 0$, $y(1, \tau) = e^{\tau/2}$ this equation has the solution

$$y(\xi, \tau) = e^{\tau/2} w(\xi)$$

where $w(\xi)$ is arbitrary. This implies that as $t \to T$ then $x(\xi, t)$ convergences to $w(\xi)$. In other words, the spatial mesh freezes and there is no possibility of resolving the singularity.

2) If $q > p - 1$, then as $t \to T^-$ the RHS of equation (33) rapidly tends to infinity. The differential equation thus becomes stiffer and stiffer, and even in the computational co-ordinates the mesh dominates the time-stepping, with any computations generally breaking down early [6].

3) If $q = p - 1$, then in the similarity co-ordinates both the MMPDE and rescaled PDE are order one and we have

$$\varepsilon - \partial_{\xi\xi}\left(y_\tau - \frac{y}{2}\right) = \left(v^q y_\xi\right)_\xi, \quad \text{so that} \quad \varepsilon\left(y_\tau - \frac{y}{2}\right) = G\left(v^q y_\xi\right)_\xi, \tag{34}$$

where the Green's function $G = (-\partial_{\xi\xi})^{-1}$ is a positive compact operator which acts as a strongly stabilizing function. We observe that as $v^q > 0$ is a regular function of ξ then this equation is essentially a rescaling of the heat equation and it has stable solutions provided that ε is sufficiently small (see [12]). Hence the grid will stabilize quickly to the equidistributed case *in the similarity variables* satisfying (a suitable discretization of) the equation

$$-\varepsilon\frac{y}{2} = G\left(v^q y_\xi\right)_\xi, \tag{35}$$

This is the scale-invariant regime that we will attempt to always work in. Of course in practice we will use $M_c = M + \alpha$, but as $\alpha \ll M(x)$ in the peak this calculation will not be affected there.

4 Temporal adaptivity

4.1 Scaling in ordinary differential equations and the Sundman transform

The previous section has shown how we may use a semi-discrete spatially adaptive method to approximate the solution of a PDE by the solution of a set of ODEs. When considering blow-up problems these equations will have singularities in time, with large changes in behaviour as $t \to T$. This can lead to very stiff equations and they are very hard to solve using a standard numerical method. Furthermore, if a constant step-size is used in the time integration then not only will the singularity be missed (in the case of explicit methods) or not computed at all (in the case of implicit methods). Furthermore, as in the PDE case, the use of a constant size time-step destroys the scaling structure of the ODES. Both of these problems can be overcome by using a suitable adaptive method and we now describe one based on rescaling.

To reduce the amount of stiffness and allow for the possibility of scalng we introduce a *computational time coordinate* τ so that computations in this coordinate become more regular. A natural way to do this is via the Sundman transform [9] in which we set $\dfrac{dt}{d\tau} = g(u)$ and express all ordinary differential equations in terms of τ. Thus the complete system used to calculate the solution of a PDE using both a moving mesh and a Sundman transformation is given by

$$A(\mathbf{X},\mathbf{U})U_{i,\tau} + B(\mathbf{X},\mathbf{U})X_{i,\tau} = g(\mathbf{U})F(\mathbf{X},\mathbf{U}), \quad X_{i,\tau} = g(\mathbf{U})G(\mathbf{X},\mathbf{U}), \quad \frac{dt}{d\tau} = g(\mathbf{U}), \tag{36}$$

where $\mathbf{X} = (X_1, X_2, \ldots), \mathbf{U} = (U_1, U_2, \ldots)$. To generalize this discussion we consider an extended vector $\mathbf{W} = (\mathbf{U}, \mathbf{X})$ so that the combined solution and mesh can be considered to solve the one system of ODEs given by

$$\frac{d\mathbf{W}}{dt} = \mathbf{f}(\mathbf{W}), \qquad \mathbf{f} : \mathbf{R}^n \mapsto \mathbf{R}^n. \tag{37}$$

Under the Sundman transform (37) becomes

$$\frac{du}{d\tau} = g(\mathbf{W})\mathbf{f}(\mathbf{W}) \qquad (g : \mathbf{R}^n \mapsto \mathbf{R}). \tag{38}$$

This system can then be solved using a standard numerical method such as BDF3 with a fixed step size $\Delta\tau$. In [9], [4] an analysis is made of the choice of g when solving ordinary differential equations with a similar scaling structure to that in (8). It is shown that if g is a function of u only and if g satisfies the scaling law

$$g(\lambda^\beta u) = \lambda g(u) \tag{39}$$

then the numerical method inherits discrete self-similar solutions which uniformly approximate the true self-similar solution. Such methods are thus very well suited to calculating singular solutions of the form (7). For example, suppose that we wish to solve (1), then motivated by the scaling law (39) we choose

$$g(u) = \frac{1}{\|u(\cdot,t)\|_\infty^{p-1}}$$

which leads to the differential equation

$$\frac{dt}{d\tau} = \frac{T-t}{v(0)} \quad \text{so that} \quad \tau = -v(0)\log(T-t).$$

We can see that the computational coordinate automatically identifies the slow-time-scale of the similarity solution even though the blowup time T is unknown. Observe that if $\Delta\tau$ is fixed then the corresponding temporal step size Δt is given, to leading order, by

$$\Delta t = (T-t)v(0)\Delta\tau$$

so that the temporal step size chosen is proportional (as it should be) to the natural time-scale $(T-t)$ of the underlying solution.

In practice this method is very robust as it provides adaptivity without the need to approximate high derivatives. It is also efficient in that, in theory, no overhead is required to adaptively choose the stepsizes. However, starting moving mesh methods is non-trivial and standard adaptive approaches may be required at this stage.

4.2 Other forms of step-size control

Most modern software packages for solving ODEs have inbuilt adaptive stepsize control. This adaptation is performed by a local approximation of a higher derivative, often (in the use of the Milne device) by comparing an intermediate calculation of the solution by a high order method with one by a low order method. However, this approach can potentially be problematic when computing blow-up type solutions. To see this, in the context of using a linear multi-step method, we suppose that we have a such time-integration method of order $q-1$, for which the absolute local truncation error at the $n-$th time-step has the form

$$e_n \approx C(\Delta t_n)^q \frac{d^q u}{dt^q}.$$

Then any adaptive time-stepping strategy will try to keep this quantity small and approximately constant hence equidistributing the error (or more possibly constrained to lie within lower an upper bounds) throughout the integration. In the case of a blow-up similarity solution the natural time-scale is $(T-t)$ and as we have seen above it is desirable that the computational time-step should scale accordingly and hence scale like $T-t$. However, in the case of the simple ODE $du/dt = u^p$ we have a blow-up solution of the form $u = K/(T-t)^{1/(p-1)}$. A simple calculation then shows that if the above error estimate is used to determine Δt_n then we have

$$\Delta t_n \approx \left(\frac{C}{\frac{d^q u}{dt^q}} \right)^{\frac{1}{q}}$$

from which we have the estimate

$$\Delta t_n \approx (T-t)^{\frac{p-1}{q}+1}$$

This calculation shows that the estimated time-step is much smaller than $(T-t)$ (as $p>1$) and hence the timestep restriction becomes overly severe very quickly. Moreover, as the solution is blowing up any numerical estimate of the derivative $\dfrac{d^q u}{dt^q}$ is likely to become increasingly less reliable leading to an instability in the choice of Δt_n. We see evidence for this in the next subsection.

4.3 ODE Examples

We now consider two simple examples in order to highlight the advantages of the rescaling approach to adaptivity based on the Sundman transform over the use of standard error control.

Example 1 A simple problem which exhibits finite time blow-up is given by

$$\frac{du}{dt} = u^2, \quad u(0) = 1. \tag{40}$$

This problem has the exact solution

$$u(t) = \frac{1}{1-t} \qquad t \in [0, 1).$$

The ODE is invariant under the transformation

$$t \mapsto \lambda t, \qquad u \mapsto \frac{1}{\lambda} u$$

and hence a scale-invariant method is given by taking

$$\frac{dt}{d\tau} = g(u) = \frac{1}{u}$$

leading to the system

$$\frac{du}{d\tau} = u \qquad \text{and} \qquad \frac{dt}{d\tau} = \frac{1}{u}. \tag{41}$$

with $u(0) = 1,\ t(0) = 0$. One striking feature of this transformation is that it has *linearized* the ODE for the unknown variable u. The trade-off for this is a nonlinear transformation of the time variable. In the case of ODEs this may not seem all that desirable, however, for singular PDEs the actual blow-up time may not be of primary interest, instead it is the blow-up rate that we are concerned with and the spatial structure of the solution near blow-up. Under the Sundman transformation all three can be resolved, though perhaps some post-processing will be required to find T more accurately. We now compare the results of solving (40) and (41) using the Matlab ODE solvers `ode15s` and `ode45`. `ode15s` is an implicit code based on backwards differentiation formulae while `ode45` is an explicit Runge-Kutta code. Both methods use adaptive time-stepping based on local error estimation. For all tests will set both the absolute and relative error tolerance to 10^{-8}. The results of this test are presented in Table 1. From Table 1 we see that using the Sundman transformation leads to a more efficient solution but also one that can be computed much further in to the singularity. In Figure 2 we see that after an initial start-up period the ODE solver applied to the rescaled problem settles on using a *uniform* computational step size. Thus, all the adaptivity is taken care of by the Sundman transformation. However, the numerical simulation of the problem in standard form is not so straightforward. Firstly we see significant variation in the stepsizes suggesting that the local error estimator is imprecise (a problem surely to be exacerbated when solving large systems) and secondly we recognize that even if this was not a problem, the limiting

	Sundman		Standard	
$u(T)$	Implicit	Explicit	Implicit	Explicit
10^{10}	445	1101	1149	1679
6.3×10^{12}	577	1383	1477	2147
1.8×10^{13}	597	1429	FAIL	2234
10^{100}	4330	10774	FAIL	FAIL

Table 1: **Example 1** Comparison of calculations on the rescaled and original ODEs. The values of $u(T) = 6.3 \times 10^{12}$ and 1.8×10^{13} correspond to the points at which the Implicit and Explicit solvers failed for the ODE in standard form. The solvers for the transformed system could carry on until $u(T)$ approached the largest representable number in Matlab: 1.7977×10^{308}.

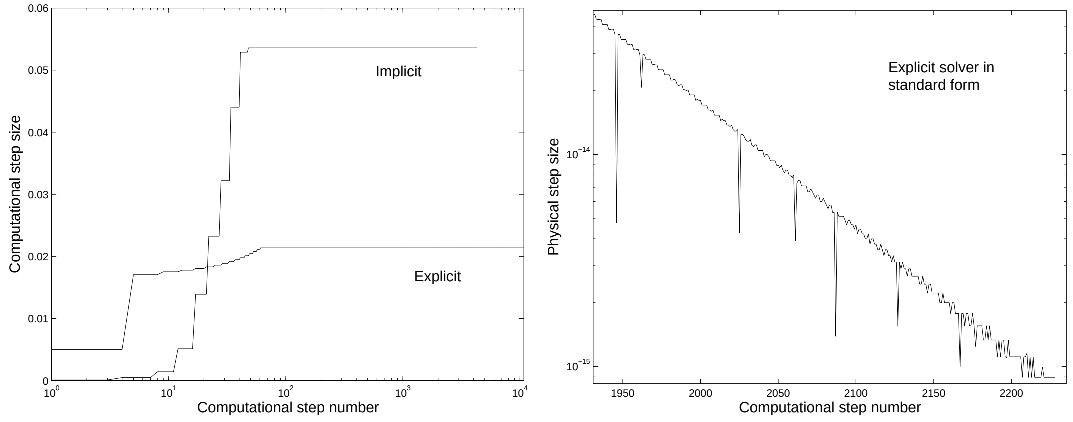

Figure 2: **Example 1** (Left) Convergence to uniform computational steps when using the Sundman transformation. (Right) Rapidly varying time steps when solving in standard form.

factor is the minimum time-step that the solver can recognize. That is, we will always be limited when

$$\frac{\Delta t_n}{\sum_{i=1}^{n} \Delta t_i} \sim \epsilon$$

where ϵ denotes the machine precision. This will not be a problem when using the Sundman transformation.

Example 2 As a second example we will consider the closely related PDE example (1) with $f(u) = u^2$. We have performed two computations, one with the Sundman transformation in time and one without. Details of the spatial adaptation strategy will be postponed until the following Section, however the ODE system resulting from this has, as described in the previous section, exactly the same scaling properties as the ODE in Example 1 above and hence we again take $g(u) = 1/\|u\|_\infty$. Results of these computations are presented in Figure 3. Here we again see that through the use of the Sundman transformation we are able to compute further in to blow-up and that the timestepping is much smoother. Recall that this is using the scaling of the problem to perform the primary adaptation and step-size selection, not the local information about the derivative approximations.

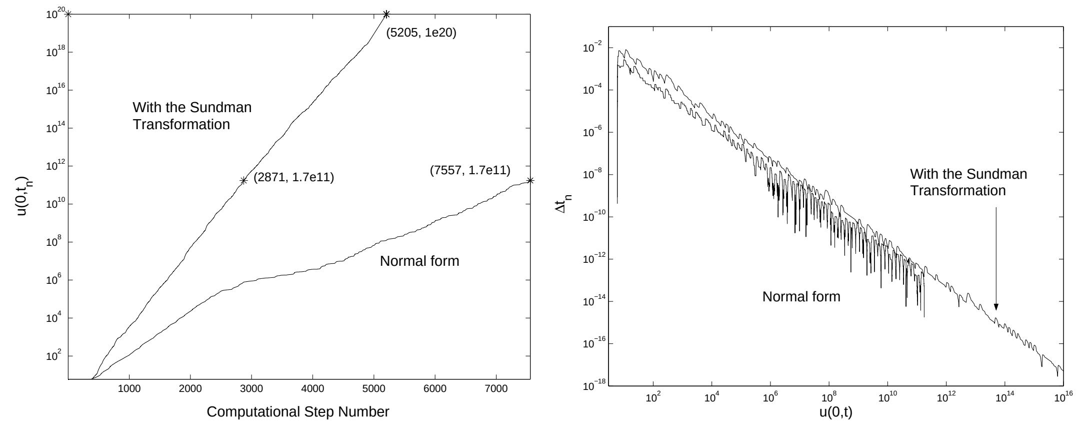

Figure 3: **Example 2** (Left) In this figure we see the growth of the solution size as a function of step number. We see that the Sundman system integrates further faster. (Right) In this figure we see the physical time steps used by each approach. With Sundman these are given by the solution method from solving $\frac{dt}{d\tau} = g(u)$, in the standard approach these steps are chosen directly by the solver. Again we see that the Sundman approach uses much larger steps. When dealing with the full system the solver gives up much earlier.

5 Discretization methods

The simplest form of moving mesh method is to use moving finite differences. While certainly not the best method for all cases, it is the easiest to start with and use to get a good understanding of how the method works for different problems. To this end, we have included an example code at `http://www.math.sfu.ca/~jfwillia/ResearchCodes`. For examples using moving collocation see [6, 25].

In one-dimensions MMPDE methods can be very effectively coupled to an underlying PDE system by using a variety of different methods including finite difference, finite element and collocation methods. We describe two such methods here.

A. Finite difference methods To motivate the discussion of appropriate discretizations, we assume that the underlying PDE system takes the form

$$\mathbf{u}_t = f(t, x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}). \tag{42}$$

If $x(\xi, t)$ is itself a time dependent function of a computational variable ξ then (42) can be cast into the Lagrangian form in the moving coordinate system given by

$$\frac{d\mathbf{u}}{dt} = \mathbf{f}(t, x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}) + \mathbf{u}_x x_t. \tag{43}$$

This introduces a nonlinear coupling between the solution components and the time derivative of the grid. Due to this, we will need to solve a Differential Algebraic Equation for the solution to our system.

An effective method for solving (43) (in one-dimensional problems) is to use the semi-discretization approach described in the previous section. In this approach we discretize the differential equation (43) in the computational coordinates together with a similar discretization of the MMPDE such as (29). It is common when doing this calculation to introduce some additional smoothing by averaging the monitor function over several adjacent mesh points [19]. In such a semi-discretization we set, as before,

$$X_i(t) \approx x(i\Delta\xi, t) \quad \text{and} \quad U_i(t) \approx u(X_i(t), t).$$

These discretizations can then be substituted into (43) and the resulting set of ODES for X_i and U_i solved along with the MMPDE. In the case of a singular problem this will usually also require a further scaling using the Sundman transformation to give a suitably smooth system of ODEs.

$$\begin{aligned} \frac{dt}{d\tau} &= g(u) \\ u_\tau - x_\tau u_x &= g(u) f(u, u_x, u_{xx}) \\ -x_{\tau\xi\xi} &= \frac{g(u)}{\varepsilon} \left(M(u) x_\xi\right)_\xi \end{aligned} \tag{44}$$

The entire problem is solved using the method of lines with standard differences in space

$$\begin{aligned} u_x(X_i, \tau) &= \frac{U_{i+1}(\tau) - U_{i-1}(\tau)}{X_{i+i}(\tau) - X_{i-1}(\tau)} \\ u_{xx}(X_i, \tau) &= \left(\frac{U_{i+1}(\tau) - U_i(\tau)}{X_{i+i}(\tau) - X_i(\tau)} - \frac{U_i(\tau) - U_{i-1}(\tau)}{X_i(\tau) - X_{i-1}(\tau)}\right) \frac{2}{X_{i+1}(\tau) - X_{i-1}(\tau)} \\ x_{\xi\xi}(\xi_i, \tau) &= \frac{X_{i+1} - 2X_i + X_{i-1}}{\Delta\xi^2} \\ \left(M(u) x(\xi_i, \tau)_\xi\right)_\xi &= \left(\frac{M_{i+1} + M_i}{2} \frac{x_{i+1} - x_i}{\Delta\xi} - \frac{M_i + M_{i-1}}{2} \frac{x_i - x_{i-1}}{\Delta\xi}\right) \frac{1}{\Delta\xi} \end{aligned} \tag{45}$$

This spatial disretization leads to a Differential Algebraic Equation of the form

$$h(t, y, y') = 0 = A(t, y) \frac{dy}{dt} - h_1(t, y).$$

With the vector $y \in \mathbf{R}^{2N+1}$ defined as

$$y = (t(\tau), U_1(\tau), U_2(\tau), \ldots, U_N(\tau), X_1(\tau), X_2(\tau), \ldots, X_N(\tau))$$

the matrix $A \in \mathbf{R}^{2N+1, 2N+1}$ has the block form

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\partial_\xi^2 & 0 \\ 0 & -u_x & I \end{bmatrix}$$

where

$$(-\partial_\xi^2)_{ij} = \frac{1}{\Delta\xi^1} \begin{cases} 1 & \text{if } i = j = 1 \text{ or } N \text{ or } |i - j| = 1 \\ -2 & \text{if } i = j \text{ for } 2 \leqslant i \leqslant N - 1 \\ 0 & \text{else.} \end{cases}$$

$$(u_x)_{ij} = \begin{cases} \dfrac{U_{i+1} - U_{i-1}}{X_{i+1} - X_{i-1}} & \text{if } i = j \text{ and } 2 \leqslant i \leqslant (N - 1) \\ 0 & \text{else.} \end{cases}$$

This allows one to easily compute the Jacobian $\partial h/\partial y' = A$ to assist the DAE solver. The function $h_1(t, y)$ is determined by (44) using the difference approximations above. The sample code solves this problem in Matlab using the NDF code `ode15i.m`.

These two systems can then be solved either alternately or simultaneously. The former is preferred for problems in higher dimensions and the latter for one-dimension. On a static mesh, the truncation errors in calculating these finite difference approximations are of second order provided that M does not vary too rapidly. We note, however, that additional errors may arise from the additional convective terms arising from the mesh movement $u_x x_t$. This additional term may lead to both theoretical and practical difficulties in applying the moving mesh methods. From a theoretical perspective it is very possible that certain desirable properties of the equation (42) (such as Hamiltonian structure and/or conservation laws) may not be inherited by the Lagrangian form (43). Note however that this term scales as u_t when the mesh equation is chosen correctly and thus scaling symmetries are preserved.

A practical difficulty, observed by [21] arises from certain discretizations of this term which can lead to instabilities and degrade the accuracy of the calculation. For example, if a centered finite difference approximation is used to discretize u_x then we have an additional truncation error given in [21] to leading order by

$$\dot{x}\frac{\Delta_i^2}{2}\left(\frac{x_{\xi\xi}}{x_\xi^2}u_{xx} + \frac{1}{3}u_{xxx}\right). \tag{46}$$

It was observed in [21] that as $x_{\xi\xi}$ can, in general, be negative and $\dot{x}$ large, then the term $x_{\xi\xi}u_{xx}/x_{\xi^2}$ can be anti-diffusive (even dominating the diffusive terms in the underlying PDE), and hence destabilizing. This problem is typically not encountered in the problems discussed in this paper as the underlying symmetries driving the mesh adaptivity mean that this term scales as the other truncation errors in the equation and remains bounded throughout the computation [14].

B. Collocation methods Spline collocation gives a powerful method of discretizing the underlying partial differential equation in the physical domain which has significant advantages over finite difference and finite element methods. In particular it affords a continuous representation of the solution and its derivatives, provides a higher order of convergence, easily handles boundary conditions and gives errors independent of local mesh grading so that by using collocation we are able to avoid the problem of approximating high order derivatives over a widely non-uniform mesh [27]. It also discretizes the PDE in the physical domain Ω_P and avoids the problems with the additional advective terms for the mesh movement described in the previous section. A very effective spline collocation discretization procedure coupled to various possible MMPDEs is adopted in the moving mesh collocation code MOVCOL described in [19] (with extensions to higher order systems using higher degree Hermite polynomials, given in the code MOVCOL [25]) and this package has been used in many tests of adaptive methods in one-dimension, see for example [19] and [7].

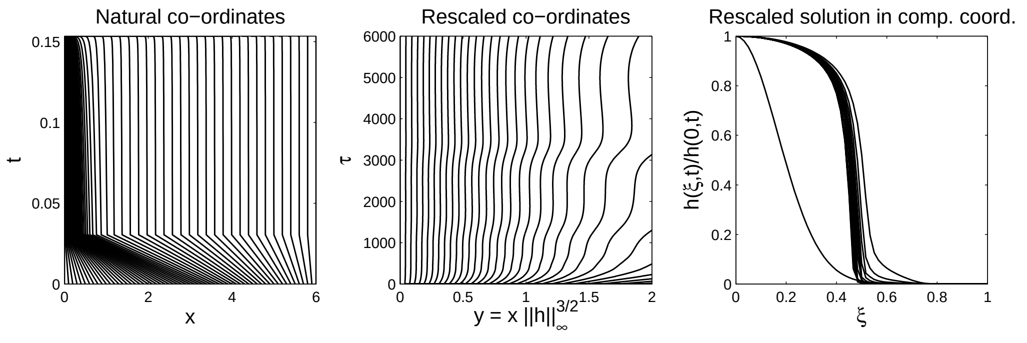

Figure 4: Some observations from integrating the thin film equation. (Left) We see that the mesh clusters in the region of the singularity but is not starved in the outer region. (Middle) The peak is resolved on a quasi-static grid in the rescaled variables. (Right) The peak region merely amplifies in the computational coordinate.

6 PDE Examples in Practice

In this Section we present detailed examples many of the problems described in the Introduction. In all cases we use a moving finite difference scheme for the PDE and (29). We set $\varepsilon = 1e-5$ and took $N = 61, 61, 121$ and 241 in Examples 1-4 respectively. The resulting system of ODEs was integrated using `ode15i` with relative and absolute tolerances set at $1e-5$. All calculations were performed using Matlab on a laptop computer and run in a few minutes.

6.1 Example 1 - Blow-up in thin films

We begin with a challenging example by considering the unstable thin film equation (3) with $p = 4$. This problem is described in [3] and [31] with u representing the thickness of a liquid film hanging from a surface. For initial data of sufficient mass it develops a singularity in finite time of the form $\|h\|_\infty = h(0,t) \sim (T-t)^{-1/7}$. The solution approaches a blow-up profile which is compactly supported in the similarity variable. This problem is invariant under scaling if $t \to \lambda t, \quad x \to \lambda^{3/14} x, \quad h \to \lambda^{-1/7} h$ and a scale invariant system then results from taking

$$M(h) = |h|^7, \qquad \text{and} \qquad g(h) = \frac{1}{||M(h)||_\infty}.$$

While this problem does not have the exact structure of the parabolic heat equation, considered thus far, we know [31] that the boundary of the blow-up region is given by the point $x^* \sim \mathcal{O}(L)$ (corresponding to the computational coordinate ξ^* where $L = \mathcal{O}((T-t)^{3/14})$). In [25] this problem was solved by using a collocation method. Here however, we present an example computed using a moving finite difference code. Typical results from a calculation with $||h||_\infty = 5e11$ are presented in Figure 4.

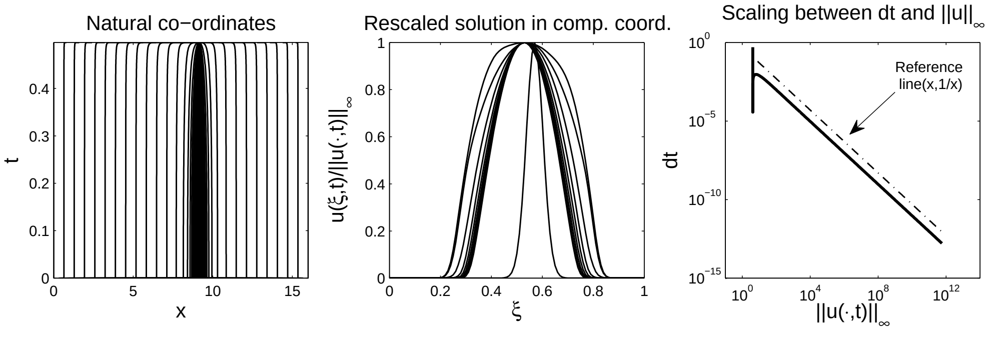

Figure 5: (Left) The grid focusses where the singularity forms but does not starve the external region. (Middle) In the computational coordinates the solution remains static up to rescaling in amplitude. (Right) The Sundman transformation correctly determines the correct time-stepping in in the physical variables – here we see it $\Delta t \sim ||u||_\infty^{-1}$ as expected.

6.2 Example 2 - The semilinear wave equation

Equation (4) is invariant under $t \to \lambda t,\ x \to \lambda^x,\ u \to \lambda^{-2/(p-1)}u$. For this example we set $p = 3$ and used

$$M(u) = |u|^2, \qquad \text{and} \qquad g(u) = \frac{1}{||M(u)||_\infty}$$

to integrate until $||u||_\infty = 1e25$. Moving mesh methods can be difficult for hyperbolic problems as there is the possibility of grid motion leading to spurious oscillations. However, this is not a problem here as the dynamics occur on a very short period of physical time. In Figure (5) we see that during an initial transient period the grid moves to follow the dynamics before focussing at the location of the singularity.

6.3 Example 3 - Bubbling in harmonic maps

The harmonic map problem (5) is well studied in the geometry and applied mathematics literatures. It has been proven that there must be a finite-time singularity for a certain class of initial data in the radially symmetric setting [28]. Unlike our previous examples, the solution in this example remains bounded for all time but a derivative singularity develops at the origin whose limiting behaviour is given by

$$\theta_r(r,t) \to \frac{2R(t)}{R(t)^2 + r^2} \text{ as } t \to T \text{ and } R(T) = 0.$$

Where the rate function $R(t)$ is initially unknown. Because blow-up in this situation corresponds to a local rescaling of space, we take $M = |\theta_r| + \sqrt{\theta_{rr}}$ to capture this and the transition out of this region. The sharp transition can be seen in Figure (6). For this example we used $g(\theta) = r^*$ where $\theta(r^*, \tau) = .01$ to capture the evolving dynamics.

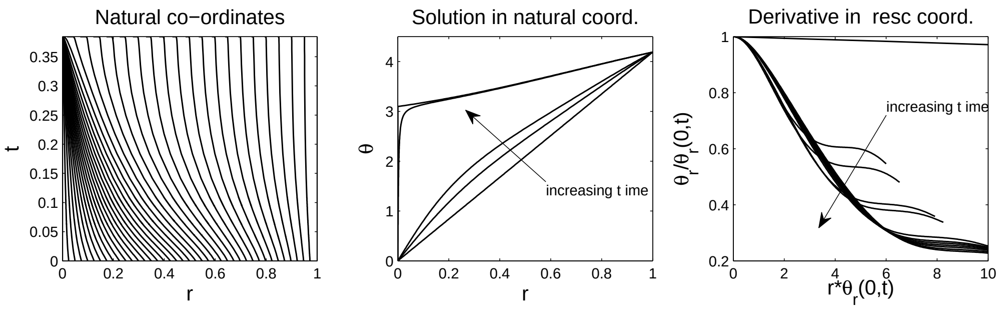

Figure 6: (Left) The grid adapts to resolve both the inner and outer solutions. (Middle) A jump discontinuity evolves at the origin. (Right) In the rescaled coordinates the derivative converges to a fixed profile.

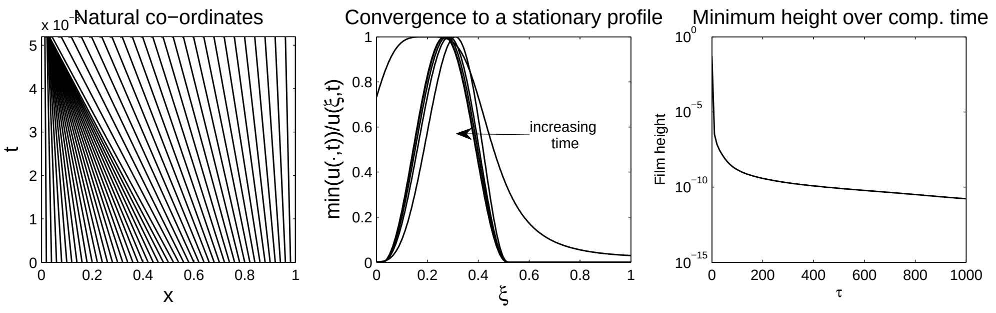

Figure 7: (Left) We capture dynamics of the moving singularity on the adaptive grid. (Middle) The profile $g = \dfrac{1}{h}$ converges to a similarity solution which is simply rescaled in amplitude in the computational coordinates. Note also the parabolic profile of the maximum of g as predicted by asymptotics. (Right) We see that the Sundman transform has identified a time-scale on which the finite-time pinch-off behaviour is resolved as $\tau \to \infty$.

6.4 Example 4 - Pinch-off in thin-films

Our methods are also not restricted to semi-linear problems. For example, the thin-film equation (6) is known to exhibit finite-time pinch-off. Asymptotic analysis leads one to conclude that this involves a jump discontinuity in the third-derivative [2]. Based on asymptotics [2], we expect a solution invariant under $t \mapsto \lambda t,\ x \mapsto \lambda^{1/2} x,\ h \mapsto \lambda h$. This motivates $M = 1/h$ and $g = 1/||M(h)||_\infty$ Again, the location of the singularity is not known in advance and, in fact, may move. In Figure (7) we see that the profile $1/h(x,t)$ converges to a similarity profile.

7 Conclusions

In this paper we have shown that carefully constructed moving mesh methods with time and space adaptivity based on symmetry conditions inherent in the solutions themselves provide an efficient and reliable way to simulate self-similar singularity formation. This enables us to capture dynamics where the behaviour is both exactly or only asymptotically self-similar. These methods are simple to program and extend the utility of even naive finite difference discretization methods.

Similar methods have been successfully implemented in two dimensions in [12, 13].

References

[1] G.I. Barenblatt (1996), Scaling, self-similarity and intermediate asymptotics, (1996) *Cambridge*

[2] A.L. Bertozzi, *The mathematics of moving contact lines in thin liquid films*, Notices Amer. Math. Soc. 45 (1998), no. 6, 689–697.

[3] A.L. Bertozzi and M.V. Pugh, *Long wave instabilities and saturation in thin film equations*, Comm. Pure. Appl. Math., **51**, (1998), 625–661.

[4] S. Blanes and C.J. Budd, *Adaptive geometric integrators for Hamiltonian problems with approximate scale invariance*, SIAM J. Sci. Comput. 26 (2005), no. 4, 1089–1113.

[5] C. de Boor, *Good approximations by splines with variable knots II.* Springer Lecture note series, **363**, (1973), Berlin.

[6] C.J. Budd, R. Carretero-Gonzalez, and R.D. Russell, *Precise computations of chemotactic collapse using moving mesh methods*, J. Comput. Phys., **202**, (2005), 462–487.

[7] C.J. Budd, S-N. Chen and R.D. Russell, *New self-similar solutions of the nonlinear Schrödinger equation, with moving mesh computations*, J. Comput. Phys., **152**, (1999), 756–789.

[8] C. J. Budd, W. Z. Huang, and R. D. Russell, *Moving mesh methods for problems with blow-up*, SIAM J. Sci. Comput., **17**,(1996), 305–327.

[9] C. J. Budd, B. Leimkuhler, and M. D. Piggott, *Scaling invariance and adaptivity*, Appl. Numer. Math., **39**, (2001), 261–288.

[10] C. J. Budd and M. D. Piggott, *Geometric integration and its applications*, In handbook of Numerical Analysis ed. F. Cucker, (2005).

[11] C. J. Budd, V. Rottshafer and J. F. Williams, *Multibump, blow-up, self-similar solutions of the complex Ginzburg-Landau equation*, SIAM J. Appl. Dyn. Syst. 4 (2005), no. 3, 649–678.

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

[13] C.J. Budd and J. F. Williams, *Moving mesh generation using the Parabolic Monge-Ampere equation*, SIAM J. Sci. Comp. (Accepted, 2009)

[14] C. J.Budd and J. F. Williams, *Optimal grids and uniform error estimates for PDEs with singularities.*, (2009, in preparation.)

[15] H. D. Ceniceros, *A Semi-implicit moving mesh method for the focusing nonlinear Schrödinger equation*, Communications on pure and applied analysis, **4**, (2002) 1–14.

[16] H. D. Ceniceros and T. Y. Hou, *An efficient dynamically adaptive mesh for potentially singular solutions*, J. Comput. Phys., **172**, (2001), 609–639.

[17] G. Delzanno, L. Chacón, J. Finn, Y. Chung and G. Lapenta, *An optimal robust equidistribution method for two-dimensional grid adaptation* based on Monge-Kantorovich optimization', J. Comput. Physics **227**(23), (2008), 9841 – 9864.

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

[19] W. Huang and R. D. Russell, *A moving collocation method for solving time dependent partial differential equations*, Appl. Numer. Math, **20**, (1996), 101–116.

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

[21] S.T. Li, L.R. Petzold and Y. Ren, *Stability of moving mesh systems of partial differential equations*, SIAM J. Sci. Comput., **20**, (1998), 719–738.

[22] G. Beckett and J.A. MacKenze, *On a uniformly accurate finite difference approximation of a singularly perturbed reaction-diffusion problem using grid equidistribution*, J. Comput. Appl. Math. 131 (2001), no. 1-2, 381–405.

[23] J. A. Mackenzie and W. R. Mekwi, *On the use of moving mesh methods to solve PDEs*, in Adaptive Computations: Theory and Algorithms, eds. T.Tang and J.Xu, Science Press, (2007), Bejing.

[24] L.R. Petzold, *A description of DASSL: A differential/algebraic system solver*, Tech report SAND82-8637, Sandia National Labs, Livermore, CA, (1982).

[25] R. D. Russell, J.F. Williams, and X. Xu, *MOVCOL4: A Moving Mesh Code for Fourth-Order Time-Dependent Partial Differential Equations*, SIAM J. Sci. Comput., **29**, (2007), 197-220

[26] A.A. Samarskii, V.A Galaktionov, S.P. Kurdyumov and A.P. Mikhailov. Blow-up in quasilinear parabolic equations. Translated from the 1987 Russian original by Michael Grinfeld and revised by the authors. de Gruyter Expositions in Mathematics, 19. *Walter de Gruyter & Co.*, Berlin, 1995

[27] P. Saucez, A. Vande Vouwer and P.A. Zegeling, *Adaptive method of lines solutions for the extended fifth order Korteveg-De Vries equation*, J. Comput. Math.,**183**, (2005), 343–357.

[28] M. Struwe, *On the evolution of harmonic mappings of Riemannian surfaces*, Comment. Math. Helv. 60 (1985), no. 4, 558–581.

[29] C. Sulem and P. L. Sulem, *The nonlinear Schrodinger equation: self focusing and wave collapse*, (1999), Springer-Verlag.

[30] T. Tang (2005), *Moving mesh methods for computational fluid dynamics*, Contemporary mathematics, **383**, (2005) 141–173.

[31] T.P. Witelski, A.J. Bernoff and A.L. Bertozzi, *Blowup and dissipation in a critical-case thin film equation*, European J. Appl. Math., **15**, (2004), 223–256.

Appendix Example Matlab driver code for the semilinear heat equation

```
function [tau,t,x,u] = SLH(N, tf)

% define the comp co-ordinate size
h = 1/(N+1);
% define the initial mesh
x = linspace(0,8,N)';
u = exp(-2*x.^2);
% Set up the initial vector for ode15i
y0 = [0;x; u];
yp0 = zeros(size(y0));



% Find consistent initial conditions.
opts = odeset('RelTol',1e-3,'AbsTol',1e-2,'Jacobian',@Jac);

[y0,yp0,resnrm] = decic(@uSLH,0,y0,[],yp0,[],opts,N);
% Lets go!
```

```
opts = odeset('RelTol',1e-4,'AbsTol',1e-4,'Jacobian',@Jac,'Events',@defout1,'Stats','on');
Tvec = linspace(0,tf,200);

[t,y] = ode15i(@uSLH,Tvec,y0,yp0,opts,N);



tau = t;
t = y(:,1);
x = y(:,2:N+1);
u = y(:,N+2:end);
%-------------------------------------------------------------------
function g = f(tau,y,N)

t = y(1);
x = y(2:N+1);
u = y(N+2:end);
% Set up the output vector
g = zeros(1 + 2*N,1);
g(1) = 1;
% Evaluate the PDE Right hand side
uxx = zeros(N,1);
i = 2:N-1;i=i.';
uxx(i) = 2*((u(i+1)-u(i))./(x(i+1)-x(i)) - (u(i)-u(i-1))./(x(i)-x(i-1)))./(x(i+1)-x(i-1));
uxx(1) = 2*(u(2)-u(1))/(x(2)-x(1))^2;
g(N+2) = uxx(1)+u(1)^2;
g(N+3:end-1) = uxx(i) + u(i).^2;
g(end) = 0;
% Evaluate monitor function.
M = abs(u);
Mc = diff(x')*(M(1:end-1)+M(2:end))/2/x(end);
M = M + Mc;
% Smooth the monitor function
M(2:end-1) = (M(1:end-2)+2*M(2:end-1)+M(3:end))/4;
M(1) = .5*(M(1)+M(2));
M(N) = .5*(M(N)+M(N-1));
% The mesh equation
tau = 1e-4;
```

```
g(2) = 0;
g(N+1) = 0;
g(i+1) = -((M(i+1)+M(i)).*(x(i+1)-x(i))-(M(i)+M(i-1)).*(x(i)-x(i-1)))/tau;
%Sundman
g = g/max(M);
%-------------------------------------------------------------------
function res = uSLH(tau,y,yp,N)
res = mass(tau,y,N)*yp - f(tau,y,N);
%--------------------------------------------------------------------
function out = mass(tau,y,N)

t = y(1);
x = y(2:N+2);
u = y(N+2:end);
% Set up the mass matrix for the DAE: M(y) y' = f(t,y)
M1 = speye(N);
M2 = sparse(N,N);
M2(1,1) = 0; %Use the zero Neumann condition at x = 0;
for i = 2:N-1
M2(i,i) = - (u(i+1) - u(i-1))/(x(i+1) - x(i-1));
end
M2(N,N) = - (u(N) - u(N-1))/(x(N) - x(N-1));
% MMPDE6
M3 = sparse(N,N);
e = ones(N,1);
M4 = spdiags([e -2*e e],-1:1,N,N);
M4(1,1) = 1;
M4(1,2) = 0;
M4(end,end) = 1;
M4(end,end-1) = 0;
out = [M4 M3
M2 M1];
out = [zeros(2*N,1) out];
out = [[1 zeros(1,2*N)]; out];
%---------------------------------------------------------
function [dfdy,dfdyp] = Jac(tau,y,yp,N)
dfdy = [];
```

```
dfdyp = mass(tau,y,N);
%-----------------------------------------------------------
% Events location function
function [v, ist, dir] = defout1(varargin)


y = varargin{2};
ist = 1;
dir = 0;
v = 1e25 - max(abs(y));
```