

Adaptivity with moving grids

Chris J. Budd

*Centre for Nonlinear Mechanics,
University of Bath, Bath BA2 7AY, UK
E-mail: mascjb@bath.ac.uk,*

Weizhang Huang

*Dept. of Mathematics,
University of Kansas, Kansas, USA
E-mail: huang@math.ku.edu*

Robert D. Russell

*Dept. of Mathematics and Computer Science, Simon Fraser University,
Burnaby V5A 1S6, Canada
E-mail: rdr@cs.sfu.ca*

In this article we describe the basic dynamics of a moving meshes as part of an r-adaptive adaptive method. We firstly examine the possible geometry of a moving mesh and strategies for discretising a partial differential equation on such a mesh. We then look in more detail (with many examples) at strategies for implementing both position based and velocity based moving meshes. We finally look at a variety of examples of the use of moving meshes in applications including scale-invariant problems, blow-up problems, meteorological problems and problems with moving fronts.

CONTENTS

1	Introduction	2
2	Moving mesh basics	10
3	Location based moving mesh methods	44
4	Velocity based moving mesh methods	74
5	Applications of moving mesh methods	81
	References	106

1. Introduction

1.1. Motivation

Time-dependent systems of partial differential equations (PDEs) often have structures that evolve significantly as the integration of the PDEs proceeds. These can be interfaces, shocks, singularities, changes of phase, high vorticity or regions of complexity. Associated with such structures are the evolution of small length (and time) scales, rapid movement of the solution features and the possibility of finite time blow-up of a component of the solution. Frequently associated are also conservation laws usually linked to underlying symmetries. Examples of these phenomena occur in many applications, such as gas and fluid dynamics, conservation laws, nonlinear optics, free boundary problems, combustion, detonation, meteorology, mathematical biology and nonlinear optics. To numerically solve such PDEs it is typical to impose some form of spatial mesh and to then discretise the solution on this mesh by using a finite element, finite volume, finite difference, or collocation method. However, this strategy may not be effective in the case of structures that involve small length scales, leading to large localised errors. In such cases it is often beneficial to use some form of non-uniform mesh, adapted to the solution, on which to perform all of the computations. The advantages of doing this can be a reduced overall error, better conditioning of the system, and better computational efficiency. Unfortunately, introducing the extra level of complexity to the system through adaptivity can also lead to additional computational cost and possible numerical instability. Mesh adaptation should thus be used with care and appropriate analysis where possible.

1.2. Adaptivity on a moving mesh

Adaptive methods for solving partial differential equations broadly speaking fit into three categories. The most extensively developed are *static regridding methods* in which a mesh is updated at each time level. The most widely used of these are **h-refinement methods** which form the basis of many commercial codes. Usually such codes start with an initially uniform mesh, and then locally coarsen or refine this by the inclusion or deletion of mesh points. The strategy for doing this is normally guided by some a-posteriori estimate of the solution error Adjerid and Flaherty (1986), Dorfi and Drury (1987), Huang and Russell (1997) and may consider problems in which the error is due to the solution geometry (such as re-entrant corners) or high derivatives. In **p-refinement methods** some finite element discretisation of the PDE is used with local polynomials of some particular order. This order is then increased or decreased in accordance with the solution error. These methods may be combined with the above and with careful a-posteriori es-

estimates to give h-p methods (Ainsworth and Oden 2000). The principle objective of the h-p methods is to obtain solutions within prescribed error bounds by such refinement procedures. There is not usually an upper bound on the number of points used in the calculation. Such methods have now been developed to a high degree of sophistication. However they are necessarily rather complex, do not necessarily take advantage of any dynamic properties of the underlying solution and the a-posteriori error estimates rely heavily on certain assumptions on the solution which may be hard to verify for strongly nonlinear problems.

The *r-refinement* (relocation refinement methods) which will form the substance of this article are a more recent development than h-p methods. 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 (Tang 2005), phase-field models and crystal growth (Mackenzie and Mekwi 2007), and convective heat transfer (Ceniceros and Hou 2001). 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 (Barenblatt 1996), (Budd and Williams 2006), where the mesh points become the *natural coordinates* for an appropriately rescaled problem. Less is known about the behaviour of *r*-adaptive methods than of *h-p* methods, and (at least in higher dimensions) they have yet to become part of established large numerical codes, although the one dimensional PDE solver MOVCOL (Russell, Williams and Xu 2007) and the celebrated continuation code AUTO (for solving two-point boundary value problems amongst others) both make use of *r*-adaptive methods. Such methods start with a uniform mesh and then *move* the mesh points, keeping the mesh topology and number of mesh points fixed as the solution evolves. Hence the alternative name of *Moving Mesh Methods* for such procedures. The mesh points are then concentrated into regions where the solution has 'interesting behaviour' usually typified by a rapid variation of either the solution or one of its (higher) derivatives. The objective of this approach is to get the smallest error possible for the number N of mesh points used, and to try to obtain error estimates which depend upon the value of N but *not* on the solution itself. For example if the solution evolves a boundary layer of width ϵ (with ϵ decreasing as time advances) then the mesh points should concentrate into this boundary layer so that the solution error is *independent of* ϵ . The moving mesh methods typically work by generating a mapping from a regular (logical or computational) domain into a physical domain in which the underlying equation is posed. The location, or the velocity, of the mesh points is then determined by solving a system of *auxiliary partial differential equations* often called the *moving mesh equations*. In all cases a vector or a scalar **monitor function** (or functions) is used to guide the position of the

Figure 1.1. A logically rectangular mesh, moved to concentrate points around a ring in an evolving singular solution of the nonlinear Schrödinger equation. Note the good radial symmetry of the adapted mesh around the ring.

mesh. The monitor function is usually designed to give an estimate of some measure of the solution error which is then *equidistributed* over each mesh cell. The monitor function is usually constructed in one of three ways. It may depend upon *a-priori solution estimates* (such as arc-length or curvature) on *a-posteriori error estimates* (such as the solution residual as used in Moving Finite Element methods (Baines 1994) or estimates of the derivative jump across element boundaries (Tang 2005)) or on some underlying *physics* related to the solution such as the potential temperature or the vorticity in a meteorological problem (Budd and Piggott 2005). In the case of *scale invariant* problems such physical estimates are often optimal. When using such methods much care has to be taken in preventing mesh tangling and ensuring mesh regularity and isotropy (where relevant). A discussion of this will form a significant part of Chapter 2. We also require that discretisations of the underlying PDE on such meshes (in either the computational or the physical domain) should retain important properties of the underlying physical solution such as conservation laws and scaling structures (Tang 2005). Provided that these conditions are carefully considered, r-adaptive methods can be used with considerable success for many time evolving systems. Examples of these include computational fluid dynamics (Yanenko, Krosho, Lisejkin, Fomin, Shapev and Shitov 1976), groundwater flow (Huang and Zhan 2004), blow-up problems (Budd, Huang and Russell 1996), (Budd and Williams 2006), (Ceniceros and Hou 2001), chemotaxis systems (Budd, Carretero-Gonzalez and Russell 2005), reaction diffusion systems (Zegeling and Kok 2004), the nonlinear Schrödinger equation (Sulem and Sulem 1999), (Ceniceros 2002), (Budd, Chen and Russell 1999), phase change problems (Mackenzie and Robertson 2002), (Mackenzie and Mekwi 2007), (Tan, Lim and Khoo 2007), shear layer calculations (Tang 2005), gas dynamics (Li and Petzold 1997), (Li, Petzold and Ren 1998), hyperbolic conservation laws with high Mach number (Li and Petzold 1997), (Tang 2005), (Stockie, MacKenzie and Russell 2000), (Tang and Tang 2003), problems with high vorticity (Ceniceros and Hou 2001), magneto-hydrodynamics (Tan 2007) and meteorological problems (Budd and Piggott 2005). More details of such applications are given in Chapter 5. In Figure 1.2 we give an example of an r-adaptive mesh which has evolved to capture the structure of a singular solution of the nonlinear Schrödinger equation which has its support concentrated around a ring.

All r-adaptive methods have intimate connections with the geometry of mapping one domain to another. They thus have intimate links with problems in differential geometry such as optimal transport (Brenier 1991), (Gangbo and McCann 1996), mean curvature flows (Huang 2007) and harmonic mappings (Dvinsky 1991). A natural application of such ideas arises in image processing, and r-adaptivity has close connections with such image processing procedures as image segmentation and image de-noising (Sapiro 2003).

There are advantages and disadvantages to each of the strategies outlined above. The h-p methods have been in use for a long time and are now well established in many commercial codes. There is a significant body of analysis supporting their use. In contrast r-adaptive methods are more recent and are less well understood. A significant criticism which has often been made of them, is that their implementation usually requires the solution of *auxiliary* partial differential equations for the mesh which must be solved in parallel to the underlying partial differential equation. This requires significant additional computational cost. Furthermore, the equations to be solved to determine a suitable mesh can often be very stiff and thus expensive to solve. Furthermore, the methods get the best estimates for a given N rather than errors necessarily lower than a specified tolerance. However, r-adaptive methods do have significant advantages in certain applications. Firstly, from a computational point of view, it is convenient to always work with the same number of mesh points and the same mesh topology. This both makes the linear algebra rather easier as the matrices considered have a constant sparsity structure and there is no need for any form of nested data structure to keep track of the node points (an issue which always complicates the use of h-refinement methods) The discretisation strategy on the mesh is also easier, especially with a finite element method, as the constancy in the mesh topology and connectivity implies that there is no possibility of hanging nodes. There are further, structural advantages to r-refinement methods. One of these is that the movement of the mesh points may well correspond to natural structures of the PDE itself. An obvious example of such are Lagrangian based methods for fluid flow problems in which mesh points move with the fluid flow. A further such example is given by the use of r-refinement methods for PDEs with natural scaling symmetries, in which the mesh points automatically follow the motion of natural similarity variables (and indeed the use of the r-refinement method becomes equivalent to the use of an appropriate coordinate transformation). A third advantage of r-refinement is that, under certain circumstances, the adaptive strategy when coupled with the PDE can be regarded as one (large) dynamical system, which may then be amenable to a combined analysis. Moving mesh methods do, of course, also have disadvantages. The limitation of having a *fixed* number of points means that it may never be possible to resolve all

of the fine structures of a PDE as it evolves (although it is surprising what can be done with often a relatively small number of mesh points). Secondly, all r-adaptive methods are, in principle, prone to *mesh tangling* in which lines connecting the mesh points can cross over during the evolution. This generally leads to severe instabilities in the system and a failure of the solution routine. Mesh tangling is often associated with *mesh racing* where some mesh points move very fast during the evolution, frequently leading to a stiff set of equations to solve. The disadvantages of having to solve an auxiliary set of partial differential equations are less severe than they might originally appear to be. Firstly, the combined system of mesh and underlying equations may be much smaller than the original system defined on a uniform mesh for the *same level of accuracy*. Indeed, in Chapter 5 we will give an example of the solution of focusing behavior in the one-dimensional Nonlinear Schrödinger (Sulem and Sulem 1999) equation (which can be written as four real first order PDEs) where a discretisation of the PDE on a set of $N = 81$ moving mesh points, is able to resolve singular structures with a length-scale of 10^{-5} and out performs discretisations on uniform meshes with 10^5 mesh points. Hence the additional 81 auxiliary equations for the mesh gives a method which outperforms one with 10^5 equations, giving a very significant cost reduction. Secondly, although the equations describing the mesh are indeed stiff in general, we don't need (in general) to solve them exactly. After all it is the underlying solution of the PDE that we are interested in and not the mesh on which it is solved. Thus a quite rough approximation to the solution of the moving mesh equations will often deliver a mesh more than adequate for the task of discretising the structures of the underlying PDE. Indeed, we will argue in Chapter 3 (and demonstrate by example in Chapter 5) that a relaxed version of the moving mesh of the moving mesh equations can be solved using a simple explicit method, will deliver similar performance to much stiffer for the mesh obtained by the more computational expensive methods. Indeed they may well be stable, more regular, and deliver a better mesh quality than solving the exact equations for the mesh. Finally, one of the main applications of h-p methods is to solve otherwise regular PDEs on irregular domains, typically with re-entrant corners, that introduce significant errors due to a lack of solution regularity at the corner. The r-adaptive methods as described in this paper are not really the right tool for this job (though see the results in (Touringy 1995)). However, a combination of h and r methods may well prove optimal in this case, where the h-method is used to mesh around the corner and the r-method to follow any evolving solution structure. Indeed, r-adaptive methods lend themselves

1.3. Computation on moving meshes

The problem of computing solutions of PDEs using a moving mesh method separates into three related problems:

- 1 Firstly, as described we need some monitor function to guide the mesh evolution, which is typically constrained either to equidistribute this function, or to relax towards an equidistributed state. In practice, whatever the choice of monitor function, some spatial (and temporal) smoothing is usually employed.
- 2 Secondly, having determined the monitor function we must determine a mesh which equidistributes this in some way. The equidistribution problem itself is a nonlinear algebraic problem and a variety of techniques have been developed to solve this problem such as a Variational Method, the Geometric Conservation law, Moving Mesh PDEs and Optimal Transport methods.
- 3 The underlying PDE is then discretised. Either on the mesh in the computational domain or in the original physical domain (in the latter case a finite element or finite volume method is usually employed (Tang 2005)). The underlying partial differential equation and the mesh equations can then be solved either simultaneously, typically by using the method of lines (Huang, Ren and Russell 1994) or alternatively (often by using a predictor-corrector method). The first method avoids the need for any interpolation from one mesh to another, but is usually associated with having to solve stiff differential equation. The alternating solution method requires interpolation from one mesh to another (which often has to be done very carefully to preserve conservation laws (Tang 2005)) and can be associated with mesh tangling phenomena. We will consider both methods in detail in this paper.

We are currently in a situation where the mesh formulation problem, mesh generation and the solution of PDES on a moving mesh are generally well understood in one spatial dimension. Reliable and efficient moving mesh methods exist (and are implemented in a number of packages) which are based on such formulations and can be used to solve time evolving PDES in one spatial dimension, with associated error estimates in certain cases. Indeed, for such problems the use of moving mesh PDEs to evolve the mesh coupled with a method of lines approach has proved to be very effective, and also amenable to analysis. In this article we will be able to give a detailed description of the theory, implementation and application of such methods. However, the problem of mesh movement, and the discretisation of PDES on such meshes, is much less understood in higher dimensions and this will form the bulk of the discussion in this paper.

1.4. A historical survey

Moving meshes and the use of adaptive strategies to minimise estimates of the solution error have a rich and diverse literature. Moving mesh methods can be classified according to the mesh movement strategy into two groups (Cao, Huang and Russell 2003), *velocity-based* methods and *location-based* ones (Tang 2005). The first group is referred to as the velocity based method since it targets directly the mesh velocity and obtains mesh point locations by integrating the velocity field. Methods in this group are more or less motivated by the Lagrange method in fluid dynamics where the mesh coordinates, defined to follow fluid particles, are obtained by integrating flow velocity. A major effort in the development of these methods has been to avoid mesh tangling, an undesired property of the Lagrange method. This type of method includes those developed in (Anderson and Rai 1983, Cao, Huang and Russell 2002, Liao and Anderson 1992, Miller and Miller 1981, Miller 1981, Petzold 1987, Yanenko, Krosho, Lisejkin, Fomin, Shapev and Shitov 1976). The method of Yanenko et al. (Yanenko, Krosho, Lisejkin, Fomin, Shapev and Shitov 1976) is of Lagrange-type. In the work of Anderson and Rai (1983), mesh movement is based on attraction and repulsion pseudo-forces between nodes motivated by a spring model in mechanics. The moving finite element method (MFE) of (Miller and Miller 1981) (Miller 1981) has aroused considerable interest. It computes the solution and the mesh simultaneously by minimizing the residual of the PDEs written in a finite element form. Penalty terms are added to avoid possible singularities in the mesh movement equations; see (Carlson and Miller 1998a, Carlson and Miller 1998b). A way of treating the singularities but without using penalty functions has been proposed by Wathen and Baines (1985). Liao and Anderson (1992) and Cai, Fleitas, Jiang and Liao (2004) use a deformation map approach. Cao, Huang, and Russell (2002) develop the GCL method based on the Geometric Conservation Law (see Chapter 4). Similar ideas have been used by Baines, Hubbard and Jimack (2005), Baines, Hubbard, Jimack and Jones (2006) for fluid flow problems.

The second group of moving mesh methods are referred to as *location based methods* because they control directly the location of mesh points. Methods in this group typically employ an adaptation functional and determines the mesh or the coordinate transformation as a minimizer of the functional. For example, the method of (Dorfi and Drury 1987) can be linked to a functional associated with equidistribution principle (Huang, Ren and Russell 1994). The moving mesh PDE (MMPDE) method developed in (Cao, Huang and Russell 1999, Huang, Ren and Russell 1994, Huang and Russell 1997, Huang and Russell 1999b) moves the mesh through the gradient flow equation of an adaptation functional which includes the energy of a harmonic mapping (Dvinsky 1991) as a special example. A combination of the MMPDE method

with local refinement is studied in (Lang, Cao, Huang and Russell 2003). Tang et al. (Li, Tang and Zhang 2002, Tang and Tang 2003) also use the energy of a harmonic mapping as their adaptation functional, but discretize the physical PDE in the rezoning approach.

So far a number of moving mesh methods and a variety of variants have been developed and successfully applied to practical problems; see the review articles (Cao, Huang, and Russell 2003, Eisman 1985, Eisman 1987, Hawken, Gottlieb and Hansen 1991, Thompson 1985, Thompson, Warsi and Martin 1982, Thompson and Weatherill 1992) and the books (Baines 1994, Carey 1997, Knupp and Steinberg 1994, Liseikin 1999, Thompson, Warsi and Martin 1985, Zegeling 1993). In particular Hawken, Gottlieb and Hansen (1991) give an extensive overview and references on moving mesh methods before 1990. In addition to the references cited above, we would also like to bring the reader's attention to the recent interesting work (Bank and Smith 1997, Beckett, Mackenzie and Robertson 2001, Budd, Huang and Russell 1996, Calhoun, Helzel and LeVeque 2008, Cenicerros and Hou 2001, Di, Li, Tang and Zhang 2005, Huang and Zhan 2004, Mackenzie and Robertson 2002, Ren and Wang 2000, Stockie, MacKenzie and Russell 2000, Tang and Tang 2003, Zegeling and Kok 2004) on moving mesh methods and their applications.

1.5. Outline of this article

The purpose of this introduction has been to give an underlying motivation for the theory and application of (adaptive) moving meshes. In Chapter 2 we will consider in detail the geometry of possible meshes (with special regard to equidistribution), and the nature of discretisations of differential equations on them. In Chapter 3 we then look in detail, and with reference to many examples, at 'location based' meshes in which the *local density* of the mesh points is controlled by a monitor function. These include *moving mesh PDE* (MMPDE) methods, optimal transport based methods and methods which implement a series of one-dimensional equidistribution strategies. This discussion will look at moving meshes in both one and higher dimensions and compare the strategies used for these two cases. In Chapter 4 we will then look at *velocity based* methods such as the Geometric Conservation Law methods (GCL) and the Moving Finite Element Methods (MFE) in which the *velocity* of the mesh points is controlled rather than their position. The concluding Chapter 5 will then look at some examples in much more detail, considering scale invariance, blow-up problems, problems with convection and moving fronts, phase change and Stefan problems, problems arising in meteorology and image processing.

2. Moving mesh basics

In this Chapter we will give an overview of the main aspects of adaptive moving mesh generation, and will concentrate on the nature of the *geometry* of an adapted mesh, the equidistribution and variational approaches to defining a mesh and the relation of the mesh to solution (truncation and interpolation) errors. The *movement* of the mesh and the way that it can be *coupled to a partial differential equation* will be discussed briefly, but will mainly be the subject of Chapters 3 and 4.

As described in the Introduction, in an r -adaptive procedure a fixed number of mesh points are *moved* in response to some user designed condition. Any r -adaptive method has two main features, a description of the *optimal geometry* of the mesh (which is related both to intrinsic properties of the mesh regularity and also of the structure of the underlying solution of the PDE) and a *strategy for evolving the mesh towards this optimal geometry*. Optimal mesh geometries are typically expressed in terms of equidistribution measures (related to the solution of the underlying PDE by monitor functions) and/or of variational principles and we review these here. *Movement strategies* are generally either methods for determining the *location* of the mesh points or of the *velocity* of the mesh points. We discuss both briefly in this chapter and then in more detail in Chapters 3 and 4.

Essential to mesh adaption is the ability to control the shape, size and orientation of mesh elements, and hence to control the error of the solution of the underlying PDE. This is done in three steps, firstly an estimate of the solution error and/or mesh quality is made. Secondly the mesh is aligned and moved according to this estimate. Thirdly the solution of the underlying PDE is advanced on the new mesh. Typically this can be in response to some structure of the solution of a PDE which is evolving in the space supporting the mesh, however there are more general circumstances (such as in image processing) where we might wish to evolve a mesh in a manner independent of any PDE.

In this chapter we will study the geometry of the meshes that arise from various of the adaptive strategies, considering such aspects as local element size, skewness and orientation as well as considering both isotropic and anisotropic meshes, as well as looking at solution error estimation and control.

2.1. Mesh mapping functions

To describe an r -adaptive mesh we consider a *fixed Computational Domain* $\Omega_C \in R^n$ in which most of the computations associated with the PDE will be made. The domain Ω_C will have the usual Lebesgue measure and may have

Figure 2.1. A typical map $(X(\xi, \eta), Y(\xi, \eta))$ from a computational domain Ω_C to a physical domain Ω_P .

a non-trivial topology. We now consider there to be a *fixed* mesh τ_C on the computational domain. This can either be uniform or non-uniform depending on the nature of the underlying problem and in the simplest case will be a uniform set of logical rectangles. It can also be triangular (and usually takes this form when a finite-element or finite-volume method is used to discretise the PDE in the *physical* domain. Alternatively, if a finite difference or a spectral method is used to discretise the PDE in the *computational domain* then a regular rectangular mesh may be more appropriate. Note that we have a lot of a-priori freedom in the choice of the computational domain Ω_C and hence when Ω_C is *simply-connected* it is often convenient to consider it to be a logically rectangular domain so that

$$\Omega_C = \prod_{i=1..d} [0, 1]$$

To describe a computational mesh in the case of such simply connected domains we typically divide $\Omega_C \subset \text{Re}^d$ into N^d *uniform, regular tetrahedra or cuboids* of side proportional to $1/N$ and volume proportional to $1/N^d$, and we will initially assume that this is the case. In the r-adaptive procedure considered in this Chapter we consider the mesh points to be joined in a simple (logically rectangular or triangular) network, the topology of which (and consequently the ordering of the nodes in the network) is fixed for most (if not all) of the time during the computation. Indeed it is this constancy of ordering which makes the r-adaptive procedure very attractive for finite-element and related computations.

To derive a moving mesh, the computational domain with its associated mesh is then mapped to a *Physical Domain* $\Omega_P \in \text{Re}^n$ in which the underlying PDE is posed. We assume that there is an invertible, *adaptive mesh generating* function

$$\mathbf{F} : \Omega_C \rightarrow \Omega_P$$

describing this map, so that \mathbf{F} is *smooth* on the interior of Ω_C and continuous on Ω_C . Throughout this article we will denote variables in Ω_C by Greek letters eg. ξ and in Ω_P by Roman letters \mathbf{x} and consider the function $F(\xi, t)$ to be *time-dependent*. The action of the function \mathbf{F} on the fixed mesh τ_C generates a *moving mesh* τ_P in the physical domain. An example of such a mesh is given in Figure 2.1 in which a uniform rectangular mesh in Ω_C is mapped to a mesh τ_P . (This map was constructed by using the optimal transport method described in Chapter 3).

In the case where τ_C is a uniform rectangular mesh, the resulting mesh τ_C in

the *physical space* is then (in the representative example of a two-dimensional system) given by the points $(X_{i,j}, Y_{i,j})$ where $\mathbf{F} = (x, y)$ and

$$X_{i,j} = x \left(\frac{i}{N}, \frac{j}{N} \right), \quad Y_{i,j} = y \left(\frac{i}{N}, \frac{j}{N} \right). \quad (2.1)$$

We assume further that the boundary of $\partial\Omega_C$ of Ω_C is mapped by F to the boundary of $\partial\Omega_P$ of Ω_P . In some r -adaptive strategies (such as the multi-equidistribution and or variational strategies described in Huang and Russell (1999b), Ceniceros and Hou (2001)) the map F is augmented with a second map

$$\partial F : \partial\Omega_C \rightarrow \Omega_P$$

which explicitly describes the map from one boundary to another. This has the advantage of close control of the meshing strategy right up to the boundary, but has the disadvantage of introducing extra complexity into the system. In other algorithms, such as the optimal mapping strategy (Delzanno, Chacón, Finn, Chung and Lapenta 2008), (Budd and Williams 2006), the boundary map is obtained automatically as part of the algorithm. This is an attractive feature from the perspective of algorithmic complexity, although it does lead to a reduction of control of the boundary points.

The great merit of this approach is that it transforms the problem of finding (and describing) a mesh in Ω_P (which in the case of h -adaptive methods can require subtle data structures including hierarchical trees) to the much simpler problem of describing the function \mathbf{F} . Much of this article is devoted to deriving suitable equations for \mathbf{F} and seeking effective solution strategies for them. It goes without saying that it should not be more difficult to find \mathbf{F} than to solve the underlying PDE, and indeed that many such functions \mathbf{F} may give appropriate meshes on which to solve the PDE. The properties of the mesh τ_P then follow immediately from the structure of the map \mathbf{F} . This simple observation is a key to the success of r -adaptive methods as it allows the use of powerful mathematical tools to both describe, construct and control the mesh behaviour. These include the application of methods from differential geometry (especially the theory of optimal transport) to describe the static structure of τ_P and from the theory of dynamical systems to describe its evolution. The latter is especially appropriate when coupled to the partial differential equations which are often solved to find \mathbf{F} . The unity, and coupling that r -adaptive meshes give between solving the underlying partial differential equation and finding the mesh, is a significant advantage of using r -adaptivity over other adaptive approaches.

2.2. *Static mesh properties: skewness, regularity and smoothness.*

We consider first some immediate properties of the mesh τ_P which are related to the function F broadly speaking these divide into *local* and *global* properties. The *global* properties relate to the isotropy of the mesh, orthogonality issues and the behaviour close to boundaries. We discuss these presently.

The *local* properties relate to the size and shape of the elements of τ_P . If τ_C is divided into logical regular rectangular or triangular elements τ_C^e , then these are mapped to elements τ_P^e in τ_P . These elements will then be distorted rectangles/triangles possibly with small angles at the vertexes. Locally we can characterise each such element by the size h_e of the largest side and (in two-dimensions) the radius ρ_e of the largest inscribed circle. If a partial differential equation is discretised over τ_P using (say) a finite-element method in 2-dimensions then the error in the solution has contributions from the size and the shape of the elements, as well as from the derivatives of the solution itself. For example, if the solution u is interpolated over τ_P^e by a piecewise-linear interpolant $\Pi(u)$ then the following a-priori error estimates are standard (Johnson 1987)

$$\max_{\tau_P^e} |\Pi(u) - u| \leq 2h_e^2 \max \left| \frac{\partial^2 u}{\partial x_i \partial x_j} \right|, \quad \max_{\tau_P^e} \left| \frac{\partial}{\partial x_i} (\Pi(u) - u) \right| \leq 6 \frac{h_e^2}{\rho_e} \max \left| \frac{\partial^2 u}{\partial x_i \partial x_j} \right|. \quad (2.2)$$

An adapted mesh will usually aim to control the size and the shape of each element so that for any particular solution u the overall error is controlled. Thus, for example, if the second derivatives of u are large over τ_P^e then the error (2.2) can be controlled locally by taking h_e to be small, and ensuring that h_e/ρ_e remains bounded. We discuss these issues in detail later in this chapter. (See Cao (2005), Cao (2007), (Chen, Sun and Xu 2007) for a very complete analysis of this problem.)

Both the size and the shape of the mesh elements can be described in terms of the local properties of F , and in particular its Jacobian J given by

$$J = \frac{\partial F}{\partial \xi} \quad (2.3)$$

The following is immediate

Condition 1 *For the map to be locally well-posed we require that J should be both bounded and invertible at all points in Ω_C .*

Mesh scaling The local scaling factor $1/\rho$ of the transformation (also called the *adaptation factor*) is given by

$$\Lambda \equiv 1/\rho = \det(J) \equiv |J|.$$

Assuming that J has a full set of eigenvalues $\lambda_1, \dots, \lambda_d$ the local stretching given by the product

$$\Lambda = |\lambda_1| |\lambda_2| \dots |\lambda_d| \quad (2.4)$$

The adaption factor controls the (possibly higher dimensional) *area* $|\tau_P^e|$ of the element τ_P^e so that

$$\rho |\tau_C^e| = |\tau_P^e| \quad (2.5)$$

The area $|\tau_p^e|$ implicitly enters into the expression (2.2). Indeed, in so-called *shape-regular* two-dimensional meshes there exist constants α and β so that for all elements $\tau_p^e \in \tau_p$ we have

$$\alpha |\tau_p^e| \leq h_e^2 \leq \beta |\tau_p^e|.$$

Accordingly many moving mesh methods (such as those based on equidistribution or variational methods) aim to control the adaption factor. Scale-invariant methods relate the adaption factor to local length scales of the underlying PDE. It is easily possible for the adaption factor to vary over *many orders of magnitude*, particularly when the adaptive method is being used to compute singular structures in the underlying PDE in which the solution u and/or its derivatives vary over similar orders of magnitude.

Mesh skewness

In the case of one-dimensional meshes, control of the adaption factor for each element completely describes the mesh. In higher dimensions many more mesh properties are important, such as the local rotation or the skewness of the mesh. A special class of *irrotational* meshes control the local element rotation by requiring that J is symmetric so that

$$J^T = J,$$

or equivalently that

$$\nabla_\xi \times \mathbf{F} = \mathbf{0}.$$

This is by no means true of all such mappings, but it can be shown (Delzanno et al. 2008), (Brenier 1991) that meshes in an averaged sense closest to uniform meshes have this property.

The shape of the element τ_P^e , in particular the existence of any small angles, is also important in the error estimate (2.2). A measure of this is the local *mesh skewness*. A measure for the *local skewness* s of the mesh is then given by

$$s = \frac{\max |\lambda_i|}{\min |\lambda_j|} \quad (2.6)$$

Other measures of mesh skewness are also referred to as shape or quality

measures. Liu and Joe (1994a) investigate several shape measures for tetrahedra and show that they are equivalent to each other. Denote the four vertexes of a tetrahedron τ_P^e by a_0, \dots, a_3 and define the so-called edge matrix as $E = [a_1 - a_0, a_2 - a_0, a_3 - a_0]$. Let \hat{e} be a regular tetrahedron having the same volume as τ_P^e . Denote the corresponding vertexes and the edge matrix of \hat{e} by $\hat{a}_0, \dots, \hat{a}_3$ and \hat{E} , respectively. Then one of the shape measures for τ_P^e is defined as

$$\eta(\tau_P^e) = \frac{3 \left[\det((E\hat{E}^{-1})^T(E\hat{E}^{-1})) \right]^{\frac{1}{3}}}{\text{trace}((E\hat{E}^{-1})^T(E\hat{E}^{-1}))}.$$

Notice that the $\eta(\tau_P^e)$ ranges from 0 to 1, with $\eta(\tau_P^e) = 1$ for a regular tetrahedron and $\eta(\tau_P^e) = 0$ for a flat tetrahedron. A geometric quality measure is introduced by Huang (2005a) for measuring the shape of a simplicial element in any dimension. Let τ_P^e be a simplicial element in d dimensions and \hat{K} be a d -simplex with unit edge length. There exists a unique invertible affine mapping

$$F_e : \hat{K} \rightarrow \tau_P^e, \quad \tau_P^e = F_e(\hat{K}).$$

Denote the Jacobian matrix of F_e by F'_e . Then the geometric measure is defined as

$$Q_{geo}(\tau_P^e) = \frac{\text{trace}((F'_e)^T F'_e)}{d \left[\det((F'_e)^T F'_e) \right]^{\frac{1}{d}}}.$$

Notice that $Q_{geo}(\tau_P^e)$ ranges from 1 to ∞ , with $Q_{geo}(\tau_P^e) = 1$ for a regular d -simplex and $Q_{geo}(\tau_P^e) = \infty$ for a flat d -simplex. Interestingly, for tetrahedra these two shape measures have the relation $Q_{geo}(\tau_P^e) = 1/\eta(\tau_P^e)$. To see this, we first notice that \hat{K} and \hat{e} are similar. Thus, the mapping $G_e : \hat{e} \rightarrow \tau_P^e$ is related to F_e by

$$G_e = cF_e, \quad G'_e = cF'_e$$

for some positive constant c . Then the edge matrices E and \hat{E} are related by

$$E = G'_e \hat{E} = cF'_e \hat{E}.$$

Using this relation we can rewrite $\eta(\tau_P^e)$ as

$$\eta(\tau_P^e) = \frac{3 \left[\det((F'_e)^T F'_e) \right]^{\frac{1}{3}}}{\text{trace}((F'_e)^T F'_e)} = \frac{1}{Q_{geo}(\tau_P^e)}.$$

It is shown by Huang (2005a) that measures s defined in (2.6) and Q_{geo} are mathematically equivalent.

Some other quality measures can be found in Liseikin (1999) Chapter 3, Knupp (2001) and Shewchuk (2002). Again, a good adaptive method aims

to control some or all of these measures of skewness, either explicitly or implicitly throughout the calculation and we discuss this later in this chapter. In the case of scale-invariant meshes we shall show that whilst the adaption factor changes a great deal *the skewness hardly varies*. More generally, it should be noted that whereas the adaption factor often changes a great deal in a mesh the skewness generally does not. To control terms in the error expression (2.2) arising from large solution gradients it is generally more important to vary the adaption factor. If this results in a locally larger value of the skewness then this can usually be tolerated.

Mesh smoothness and regularity

The smoothness or regularity of a mesh is a measure of how much the mesh elements vary over the mesh. This can be important since the accuracy and error in the numerical solution of partial differential equations generally depend upon the type of discretization, the quality of the mesh, the treatment of boundary conditions, etc. A uniform mesh has the highest degree of regularity, which can lead to particularly low error estimates on such meshes. It is sometimes claimed that only uniform meshes have such low estimates, but in fact, as we shall see, they share this with sufficiently regular meshes. Although there is generally no simple relationship between the smoothness of the mesh and the error (cf. Veldman and Rinzema (1992)), for most problems and most discretization methods, abrupt variations in the mesh will cause a deterioration in the convergence rate and an increase in the error Thompson, Warsi and Mastin (1985), or indeed in the accuracy of the approximation of a function over the mesh. Moreover, most discrete approximations of spatial differential operators have much larger condition numbers on an abruptly varying mesh than they do on a gradually varying one, and these ill-conditioned approximations may result in stiffness in the time integration for time-dependent problems.

The smoothness of a mesh can be expressed in terms of the regularity of the underlying mesh function \mathbf{F} .

Definition *A mesh τ_P has degree of regularity r if*

$$\mathbf{F} \in C^r(\Omega_C).$$

The regularity of \mathbf{F} can often be achieved by determining \mathbf{F} as a solution of a PDE system or a minimizer of a functional as in variational mesh generation methods. In many cases it is possible to have strong control over the derivatives of \mathbf{F} allowing guaranteed regularity of the mesh τ_P . It should be noted that an obvious way to determine a mesh is to prescribe the Jacobian function J exactly (Brackbill and Saltzman 1982). However, it it

in general very hard to do this, and instead some property of J (such as its determinant) is prescribed and this is used to then determine the mesh.

A mesh can also be made smoother by some direct methods. For example, (weighted) Laplace smoothing is often used in h-p-adaption (cf. (Carey 1997)). In this strategy, the coordinates of an interior mesh point are adjusted so that they become the (weighted) average of the coordinates of its neighboring points. Typically this is carried out in a Jacobian or Gauss-Seidel fashion. When this is the case, Laplace smoothing can be viewed as the application of the Jacobian or Gauss-Seidel iteration to the solution of a discretization of the partial differential equation

$$-\Delta_{\xi}(\hat{x}, \hat{y}) = (x, y), \quad (2.7)$$

where Δ_{ξ} is the Laplacian operator applied in the computational domain. In r-adaptive methods based on equidistribution, on the other hand, a smoother mesh is often obtained indirectly by smoothing the monitor function M used for controlling mesh adaptation and movement. We will describe this strategy presently.

When calculating solutions to a (partial) differential equation on a non-uniform mesh it is essential that there is a strong control on the mesh variation. For one-dimensional meshes for which we have a mesh function $x(\xi)$, mesh points $X_i = x(i\Delta\xi)$ and local mesh spacing given by $\Delta_i = X_{i+1} - X_i$ then the *grid size ratio* or *local stretching factor* r is given by

$$r = \frac{\Delta_i}{\Delta_{i-1}}. \quad (2.8)$$

In a uniform mesh we have that $r = 1$. For many calculations on a non-uniform mesh we require instead that

$$r = 1 + \mathcal{O}(\Delta_i). \quad (2.9)$$

Such grids are termed *quasi-uniform* Li, Petzold and Ren (1998), Zegeling (2007), Kautsky and Nichols (1980), Kautsky and Nichols (1982), and normally lead to truncation (and approximation) errors of the same order as uniform meshes, (Veldman and Rinzema 1992). We note that

$$r = 1 + \frac{\Delta_i - \Delta_{i-1}}{\Delta_i} = 1 + \frac{X_{i+1} - 2X_i + X_{i-1}}{X_i - X_{i-1}}.$$

Consequently, as $\Delta_i \approx \Delta\xi^2 x_{\xi}$ etc. we have

$$r = 1 + \Delta_i \frac{x_{\xi\xi}}{x_{\xi}^2}.$$

Thus the mesh is quasi-uniform provided that

$$\Lambda \equiv \frac{x_{\xi\xi}}{x_{\xi}^2} = \mathcal{O}(1). \quad (2.10)$$

The condition (2.10) plays an important role in our subsequent analysis of the errors of computations on both static and moving non-uniform meshes. The ratio between lengths of adjacent elements is also used in Dorfi and Drury (1987) and studied by Verwer,Blom,Furzeland and Zegeling (1989).

The concept of quasi-uniformity has natural extensions to higher dimensions augmented with small angle conditions. For example, in two-dimensions, if we have a triangulation τ_P then this is *shape regular*, ensuring control over small angles, if for each element $\tau_e \in \tau_P$ with area $|\tau_e|$, longest side of length h_e and interior circle of diameter ρ_e we have a constant σ_1 so that

$$\max_{\tau_e} \frac{h_e}{\rho_e} \leq \sigma_1. \quad (2.11)$$

Such a shape regular mesh is then *quasi-uniform* if there is a second constant σ_2 for which

$$\frac{\max_{\tau_e \in \tau_P} |\tau_e|}{\min_{\tau_e \in \tau_P} |\tau_e|} \leq \sigma_2. \quad (2.12)$$

As in the one-dimensional case, quasi-uniform meshes have similar error estimates to uniform ones (Johnson 1987) . However, it is often much harder to achieve this for time dependent problems.

2.3. Mesh calculation, mesh tangling and mesh racing

The function \mathbf{F} must be determined as part of the process of calculating τ_P . This map can be calculated either explicitly or implicitly. In the explicit method, an equation is derived for \mathbf{F} which is expressed in terms of the *position* of the mesh points. This (usually large and nonlinear) system is then solved to find \mathbf{F} and hence to determine the location of the mesh. This procedure lies at the heart of a number of equidistribution *position based* methods for calculating the mesh such as the moving mesh partial differential equation, optimal transport and variational methods. Typically such methods *cluster* the mesh points where high precision is required, and the location of points of *density* of the mesh points moves as the solution evolves (in a similar manner to a longitudinal wave passing down the length of a spring, whilst the coils of the spring do not move very far from their equilibrium positions).

In an alternative procedure, the *velocity* \mathbf{v} of the mesh points in τ_P is determined. This is given by

$$\mathbf{v} = \mathbf{F}_t. \quad (2.13)$$

Figure 2.2. An example of a mesh which has tangled whilst attempting to resolve a front.

The mesh point positions are updated using this velocity. This approach is very closely linked with particle and Lagrangian methods and includes methods such as GCL, MFE and the Harmonic map method. (Using the analogy above, such methods are like moving the whole spring).

In general, position based methods tend to produce smoother meshes, and are much less prone to the problem of *mesh tangling* than velocity based methods. Mesh tangling occurs when the lines connecting adjacent mesh points intersect. An example of this is given in Fig. 2.3 in which we see an attempted calculation of a solution front for Burgers' equation which has led to a tangled mesh through the use of an inappropriately large time step in evolving the mesh.

Mesh tangling can occur either locally or globally and can often arise in Lagrangian type methods computing solutions with high vorticity. It is associated with a local loss of invertibility of the map \mathbf{F} or, equivalently, at a point for which $\Lambda = \det(J) = 0$. If Λ is controlled throughout the evolution of the mesh, then mesh tangling can be avoided. Position based methods usually try to do this, see the calculations using the optimal transport and MMPDE methods, hence their robustness to mesh tangling. Of course, control of Λ for all time is impossible as any equations describing the time evolution of \mathbf{F} will inevitably be discretised in time. If this discretisation is too coarse then mesh tangling may result.

Mesh racing is related to mesh tangling and occurs if \mathbf{v} is too large relative to the evolutionary behaviour of the underlying system being solved (so that the mesh evolves more rapidly than the solution of the underlying PDE). Mesh racing can occur for a variety of reasons, such as an inappropriate choice of adaptivity strategy, gross mesh distortion or problems when the moving mesh interacts with a fixed boundary. In a purely *Lagrangian* setting a moving mesh used to calculate (for example) a fluid flow might seek to have \mathbf{v} equal to the local velocity of the fluid particles. In practice, as we will demonstrate, such a procedure can lead to mesh tangling in the presence of flows with high vorticity and mesh racing when the fluid particles leave the boundary and the mesh labeling has to be reassigned. In practice (and in a manner to be made precise presently) it is often optimal for the mesh points to move in a similar manner to the particles, but not to follow their motions too precisely.

The connectivity of a mesh reflects how adjacent nodes are connected together. In an h-adaptive mesh connectivity can be a significant issue and changes every time a local mesh refinement step is implemented. This causes

additional overheads in setting up the equations of any discretisation on this mesh as the connectivity matrix needs to be constantly updated. In contrast, in an r-adaptive method, the connectivity of the moving mesh is usually determined by the connectivity of the underlying computational mesh, which usually does not change during the calculation, and we can presume to be relatively simple. A significant advantage of this approach is that various mesh smoothing methods can take advantage of this constant connectivity and can (for example) exploit fast spectral methods which take advantage of the constant mesh connectivity in the computational domain. As an example, if the functions $(x(\xi, \eta), y(\xi, \eta))$ determine a particular mesh, then it is possible to construct a smoother mesh from this. One example is given by

$$(\hat{x}, \hat{y}) = (I - \gamma \Delta_\xi)^{-1} (x, y) \quad (2.14)$$

where Δ_ξ is the Laplacian operator applied in the *computational domain* and γ is chosen appropriately. An important reason for constructing a smoother mesh is to avoid significant variation in mesh size between adjacent elements. We presently consider the effect of this on the solution error.

This procedure was introduced by Huang and Russell (1997) and developed by many authors such as Cenicerros and Hou (2001). The Laplacian operator can be inverted very rapidly on a simply connected uniform rectangular mesh by using a fast spectral method, for example the fast cosine transform. This smoothing procedure also damps out the creation of certain checker-board modes that can lead to a deterioration of the mesh quality.

2.4. Mesh Topology

The discussion so far has been restricted to the use of moving meshes mapping one convex region, indeed logically rectangular regions to logically rectangular regions. There is no real problem mapping a logically rectangular region Ω_C to another convex region. For example, the article Calhoun, Helzel and LeVeque (2008) describes in detail how a logically rectangular mesh can be mapped to both circular and spherical domains. This is especially useful for calculations in meteorology involving whole earth models. However, moving mesh methods are not ideal for mappings to and from non-convex regions due to the inherent singularities associated with re-entrant corners. See Cenicerros and Hou (2001) for a brief discussion of this point. The issue of refining a mesh close to such a corner where the geometry of the solution and the associated singularity is (of course) known a-priori has been extensively covered in the literature of h-adaptive methods, see for example (Ainsworth and Oden 2000), Johnson (1987) and many other texts. This approach can be very naturally coupled with a moving mesh approach by using an h-adaptive method to construct a mesh in the *computational*

domain which is refined close to the re-entrant corner. This mesh can then be mapped, in a similar manner to that described earlier, to a moving mesh in the *physical domain*. We will not pursue this further here as this article is largely concerned with the construction of meshes adapted to the evolving structures of time-dependent PDEs.

2.5. Equidistribution and monitor functions

Having considered the general aspects of the mesh geometry and the mesh function \mathbf{F} we now consider the issues associated with calculating appropriate functions \mathbf{F} to give meshes with certain properties. There are a number of general approaches to do this, and we consider two closely related methods; equidistribution based and variational based. Both methods generate meshes determined by suitable *monitor functions* which typically are determined both by properties of the solution of the underlying partial differential equation and by other considerations of the mesh regularity.

Equidistribution

At the heart of many r-adaptive methods is the concept of *equidistribution*, introduced as a computational device by de Boor (1973). Equidistribution is a widely used means of prescribing the *optimum geometry* of the mesh, but many different strategies have been devised to move the mesh towards this optimum state leading to a variety of moving mesh methods. In a certain sense, all meshes equidistribute some function, and to motivate equidistribution we consider the fundamental Radon-Nikodym theorem from measure theory. To do this we consider an invertible mesh mapping function F which maps an arbitrary set A in Ω_P to an image set $B = F(A)$ in Ω_P . We can induce a measure $\nu(B)$ on Ω_P by $\nu(B) = |A|$ where $|A|$ is the usual Lebesgue measure on Ω_C . We then have

Theorem 1. Radon-Nikodym If B is an arbitrary set in $\Omega_P \subset R^n$ and $\nu(B)$ is a well defined Borel measure, then there is a unique function $M(\mathbf{x}) > 0$ so that

$$\nu(B) = \int_B M(\mathbf{x}) d^n \mathbf{x},$$

where the integration is with respect to the usual Lebesgue measure on Ω_P .

Proof. See Shilov and Gurevich (1978) □

The Radon-Nikodym theorem shows that for any map \mathbf{F} we can find a unique function $M(\mathbf{x})$ so that for any set $A \subset \Omega_C$ we have

$$\int_A d^n \mathbf{x} = \int_{F(A)} M(x) d^n \mathbf{x}. \quad (2.15)$$

Note however, that (other than the special case of one-dimension) the same function M may be associated with many different maps.

The function $M(\mathbf{x}) > 0$ is a function of \mathbf{x} and t , but is more usually defined in terms of the solution $u(\mathbf{x}, t)$ of the underlying PDE so that we might have

$$M(\mathbf{x}, t) \equiv M(\mathbf{x}, u(\mathbf{x}, t), \nabla u(\mathbf{x}, t), \dots, t).$$

In this context M is usually called a **scalar monitor function** and is chosen to be large when the mesh points need to be clustered, for example if the solution of the underlying problem has a high gradient. In this case the Lebesgue measure of the set B may be small even if the measure $\nu(B)$ is not. This may occur, for example, in the neighbourhood of a solution singularity or a sharp front. An obvious example of a monitor function is some estimate of the truncation error in the calculation of the solution of the underlying PDE, and this was the original motivation of the equidistribution approach of de Boor (1973). Loosely speaking, equidistributing the error in calculating the solution of a PDE over all mesh elements is a necessary condition for finding a global minimum of that error Johnson (1987)

Assume now that we know the scalar function M and consider how we might determine an appropriate map \mathbf{F} . To do this we introduce an arbitrary non-empty set $\mathbf{A} \subset \Omega_C$ in the computational domain, with a corresponding image set $\mathbf{F}(\mathbf{A}, t) \subset \Omega_P$. The map \mathbf{F} *equidistributes the respective scalar monitor function* M if the Stieltjes measure of \mathbf{A} and $\mathbf{F}(\mathbf{A}, t)$ normalised over the measure of their respective domains are the same. This implies that

$$\frac{\int_{\mathbf{A}} d^n \xi}{\int_{\Omega_C} d^n \xi} = \frac{\int_{\mathbf{F}(\mathbf{A}, t)} M(\mathbf{x}, t) d^n \mathbf{x}}{\int_{\Omega_P} M(\mathbf{x}, t) d^n \mathbf{x}}. \quad (2.16)$$

It follows from a change of variables that

$$\frac{\int_{\mathbf{A}} d^n \xi}{\int_{\Omega_C} d^n \xi} = \frac{\int_{\mathbf{A}} M(\mathbf{x}(\xi, t), t) |J(\xi, t)| d^n \xi}{\int_{\Omega_P} M(\mathbf{x}(\xi, t), t) d^n \mathbf{x}}. \quad (2.17)$$

As the set \mathbf{A} is arbitrary, the map $\mathbf{F}(\xi, t)$ must (for all (ξ, t)) obey the identity

$$M(\mathbf{x}(\xi, t), t) |J(\xi, t)| = \theta(t) \quad \text{where} \quad \theta(t) = \frac{\int_{\Omega_P} M(\mathbf{X}(\xi, t), t) d^n \mathbf{x}}{\int_{\Omega_C} d^n \xi}. \quad (2.18)$$

We shall refer to (2.18) as the *equidistribution equation*. This equation must always be satisfied by the map $\mathbf{F}(\xi, t)$. It is the central equation of much of mesh generation, and we shall show presently that it is strongly connected to a variational representation of the mesh transformation.

Choice of a scalar monitor function

The choice of a scalar monitor function M appropriate to the accurate solution of a PDE is difficult, problem dependent and the subject of much research. We do not consider this in detail here but give a brief review of various choices used for certain problem classes. The function M can be determined by a-priori considerations of the geometry or of the physics of the solution. An example is the generalised solution arc-length given by

$$M = \sqrt{1 + c^2 |\nabla_x u(x)|^2} \quad \text{or alternatively} \quad M = \sqrt{1 + c^2 |\nabla_\xi u(x(\xi))|^2}. \quad (2.19)$$

The first of these is often used to construct meshes which can follow moving fronts with locally high gradients (Winslow 1967), (Huang 2007). A careful analysis of the application of arc-length based monitor functions to the resolution of the solution of singularly perturbed PDES is given in (Kopteva and Stynes 2001). In (Ceniceros and Hou 2001) the second monitor function (with u being the temperature) was used successfully to resolve small scale singular structures in Boussinesq convection. It is also common to use monitor functions based on the (potential) vorticity, or curvature, of the solution (Beckett and Mackenzie 2000) and these have been used in computations of weather front formation (Budd and Piggott 2005), (Budd, Piggott and Williams 2008), (Walsh, Budd and Williams 2009). In certain problems, moving fronts are associated with changes in the physics of the solution. An example is problems with phase changes, where the phase front occurs at those points $(x_m)_i$ at a temperature $T = T_m$. In such cases it is possible to construct meshes which resolve behavior close to the phase boundary by using the monitor function $M = a/\sqrt{b}|x - x_m| + c$ where $|x - x_m| = \min |x - (x_m)_i|$ (Mackenzie and Mekwi 2007). Alternatively, M can be linked to estimates of the solution error. A significant calculation in which M was determined in terms of *a-priori* error estimates (typically proportional to the higher derivatives of the solution or estimates of these) was given in Dorfi and Drury (1987) and is discussed in more detail presently. More recently, monitor functions determined by *a-posteriori* error estimates have been constructed. An example of such, in the context of a piecewise linear finite element approximation u_h to a function u , is $M = \sqrt{1 + \alpha \zeta^2}$, where

$$|u - u_h|_{1,\Omega_P}^2 \sim \zeta^2(u_h) \equiv \sum_{l: \text{interior edge}} \int_l [\nabla u_h \cdot n_l]_l^2 dl \quad (2.20)$$

and $[\cdot]_l$ is the jump in the computed solution along the element edges. This monitor function is used by Tang (2005) to adaptively compute solutions to the Navier-Stokes equations with thin shear layers and/or high Mach numbers. Similarly, in a series of papers studying both isotropic and anisotropic meshes, (Huang 2001),(Huang 2001), (Huang 2005a), (Huang

2005b), (Huang 2007), Huang explicitly considers monitor functions which are designed to control the regularity, alignment and quality of the mesh. These include monitor functions which are based as estimates of the interpolation error of the computed solution and we consider them presently. Other measures of mesh quality can be incorporated into the monitor function including maximum and minimum angle conditions (Zlamal 1968), (Babuska and Rheinboldt 1979), conditions on aspect ratio and quantities that combine both shape and solution behaviour (Berzins 1998). Finally, it is sometimes possible in the case of PDES with *strong scaling structures* (such as problems related to combustion and gas dynamics) to find suitable monitor functions which give meshes that reflect the natural scales of the problem (Budd and Williams 2006). We give an example of such in Chapter 5, looking at a PDE which has solutions which blow-up in a finite time. In this case we need a fine mesh when the solution is large, and take $M(u) = \sqrt{a^2 + b^2 u^{2p}}$, $p > 0$.

We note at this stage that most choices of monitor function need a degree of smoothing and regularisation and smoothing to perform effectively, and we will consider this presently.

Moving the mesh to an equidistributed state

The equidistribution equation must be solved to find a mesh which equidistributes the scalar monitor function M and this equation must be augmented with additional conditions to obtain a unique map \mathbf{F} . Indeed, the equidistribution principle has different consequences in one-dimension from higher dimensions. In one-dimension it (together with boundary conditions) uniquely defines the map $\mathbf{F}(\xi, t)$. In this case the strategies for moving the mesh are all similar (or indeed trivially equivalent) and rely on either exactly solving the equidistribution equation (2.18), or relaxing towards a solution of some differentiated form of (2.18). Examples of the relaxation methods are the variety of *Moving Mesh PDE MMPDE* methods such as **MMPDE6** given by

$$-\epsilon x_{\xi\xi} = (Mx_{\xi})_{\xi}. \quad (2.21)$$

Here $0 < \epsilon \ll 1$ is a *relaxation time* over which the mesh evolves to the equidistributed state. We consider the derivation and properties of MMPDES (and their higher-dimensional extensions) in detail in Chapter 3.

However, uniqueness of the solution of the equidistribution equation is lost in higher dimensions. Informally this is because there is a unique interval (up to translation) of prescribed length in one-dimension, but there are an uncountable number of sets of prescribed area in higher dimensions. Thus the equidistribution principle needs to be augmented with some additional conditions if it is to be applied in dimension $d > 1$. The determination of

these additional conditions is neither straightforward nor unique and leads to a variety of different methods for mesh generation for which control of mesh skewness and other geometrical properties must also be considered. Two examples of the additional conditions might be to impose an irrationality condition in the computational domain so that $\nabla_{\xi} \times \mathbf{F} = 0$ (Budd and Williams 2006) (which is the basis of the optimal transport methods) or to require that the mesh velocity is irrotational in the physical domain so that $\nabla_x \times \mathbf{v} = 0$ (Cao, Huang, and Russell 2002) (which is the basis of the GCL methods). Alternatively, a vector valued monitor function can be used and we consider this in the next section.

The augmented equations, can then be solved in a number of ways to find the mesh. By directly solving the nonlinear system, which can be expensive. By differentiating the condition and solving the resulting differential equations which leads to the GCL method. By relaxing towards a solution of the system, which leads to the MMPDE methods in one and higher dimensions. Or to have a global variational principle associated with the error and to find the gradient flow equations associated with it. We consider each of these in turn presently.

Matrix-valued monitor functions

The monitor function defined above is a scalar measure and is effective in the specification and generation of certain isotropic meshes. However much greater freedom in mesh calculation may be required when calculating anisotropic meshes and in this case a **matrix-valued monitor function** \mathbf{M} can be used. In this case the meshes are defined via the metric determined by an $n \times n$ - matrix valued monitor function that specifies the shape, size and orientation of the elements throughout the physical domain Ω_P . Huang (2007) defines a matrix-monitor function $\mathbf{M}(\mathbf{x})$ via the identity

$$J^{-T} J^{-1} = \left(\frac{\int_{\Omega_P} \sqrt{\det(\mathbf{M})} dy}{|\Omega_C|} \right)^{-\frac{2}{n}} \mathbf{M}(\mathbf{x}) \quad (2.22)$$

which is closely linked to the equidistribution principle for the scalar function. Indeed the mesh satisfies an *equidistribution equation*

$$|J| \sqrt{\det(\mathbf{M})} = \frac{\int_{\Omega_P} \sqrt{\det(\mathbf{M})} dy}{|\Omega_C|}. \quad (2.23)$$

It also follows an *alignment condition*

$$\frac{1}{n} \text{trace}(J^{-1} \mathbf{M}^{-1} J^{-T}) = \det(J^{-1} \mathbf{M}^{-1} J^{-T})^{\frac{1}{n}}. \quad (2.24)$$

A matrix monitor function \mathbf{M} , together with a proper boundary correspondence, then specifies a mesh via the conditions (2.23) and (2.24). Presently

we relate these conditions of alignment and equidistribution to mesh quality and interpolation error and show how to construct monitor functions which give explicit control over mesh quality.

2.6. Variational principles and Harmonic maps.

Overview

An alternative strategy for determining a mesh, also based on an appropriate monitor function, is the variational method. In such a method the stationary points determine the optimal mesh, and the associated gradient flow equations towards the stationary points determine a suitable mesh motion strategy.

Suppose that $I(\xi)$ is a certain functional and that the mesh generation strategy is equivalent to minimising I over a certain function space. Finding the Euler-Lagrange equations then leads to a gradient flow equation to evolve the mesh towards the equilibrium state (a stationary point of I) which in two-dimensions is given by

$$\frac{\partial \xi}{\partial t} = -\frac{\delta I}{\delta \xi}, \quad \frac{\partial \eta}{\partial t} = -\frac{\delta I}{\delta \eta}. \quad (2.25)$$

This can then lead directly to a *moving mesh partial differential equation* or **MMPDE** (the first of many we will see in this article) to move the mesh by introducing some additional local control on the mesh movement in the form

$$\frac{\partial \xi}{\partial t} = -\frac{P \delta I}{\tau \delta \xi}, \quad \frac{\partial \eta}{\partial t} = -\frac{P \delta I}{\tau \delta \eta}, \quad (2.26)$$

where P is a positive differential operator and $\tau > 0$ is a parameter for adjusting the time scale of mesh movement.

Variational methods in one-dimension and links to equidistribution

In one dimension equidistributing the scalar monitor function M is exactly equivalent to minimising the functional

$$I(\xi) = \frac{1}{2} \int_0^1 \frac{1}{M} \left(\frac{\partial \xi}{\partial x} \right)^2 dx \quad (2.27)$$

If the function P in (2.26) is taken to be

$$P = \left(\frac{M}{\xi_x} \right)^2$$

then we obtain the MMPDE

$$\frac{\partial x}{\partial t} = \frac{1}{\tau} \frac{\partial}{\partial \xi} \left(M \frac{\partial x}{\partial \xi} \right). \quad (2.28)$$

This equation can be used to evolve the one-dimensional mesh towards an equidistributed state. It also has a natural generalisation to the PMA equation derived from the *optimally transported meshes* we will consider in the next chapter.

Variational methods in higher dimensions and Harmonic Maps

Motivated by (2.27) we can consider a generalisation to two-dimensions which is essentially a form of *equidistribution in each coordinate direction* (Huang and Russell 1999a), (Huang and Russell 1999b). This is given by

$$I(\xi, \eta) = \frac{1}{2} \int_{\Omega_P} [\nabla \xi^T \mathbf{M}^{-1} \nabla \xi + \nabla \eta^T \mathbf{M}^{-1} \nabla \eta] dx dy, \quad (2.29)$$

where \mathbf{M} is now a symmetric positive definite matrix valued monitor function which is a generalisation of the original scalar monitor function. The Euler-Lagrange equations which define the coordinate transformation at the steady state are then given by

$$\nabla \cdot (\mathbf{M}^{-1} \nabla \xi) = 0, \quad \nabla \cdot (\mathbf{M}^{-1} \nabla \eta) = 0. \quad (2.30)$$

where *all derivatives are expressed in terms of the physical variables* so that $\nabla = (\partial_x, \partial_y)$. A moving mesh can then be obtained via the gradient flow equations (2.25). When these are inverted so that they can be expressed in terms of the physical variables they take the form

$$\begin{aligned} \frac{\partial}{\partial t} \begin{pmatrix} x \\ y \end{pmatrix} = & - \frac{1}{\tau |J| \sqrt{\det(\mathbf{M})}} \begin{pmatrix} x_\xi \\ y_\xi \end{pmatrix} \left\{ \frac{\partial}{\partial \xi} \left[\frac{1}{|J| \det(\mathbf{M})} \begin{pmatrix} x_\eta \\ y_\eta \end{pmatrix}^T \mathbf{M} \begin{pmatrix} x_\eta \\ y_\eta \end{pmatrix} \right] \right. \\ & \left. - \frac{\partial}{\partial \eta} \left[\frac{1}{|J| \det(\mathbf{M})} \begin{pmatrix} x_\eta \\ y_\eta \end{pmatrix}^T \mathbf{M} \begin{pmatrix} x_\xi \\ y_\xi \end{pmatrix} \right] \right\} \\ & - \frac{1}{\tau |J| \sqrt{\det(\mathbf{M})}} \begin{pmatrix} x_\eta \\ y_\eta \end{pmatrix} \left\{ - \frac{\partial}{\partial \xi} \left[\frac{1}{|J| \det(\mathbf{M})} \begin{pmatrix} x_\xi \\ y_\xi \end{pmatrix}^T \mathbf{M} \begin{pmatrix} x_\eta \\ y_\eta \end{pmatrix} \right] \right. \\ & \left. + \frac{\partial}{\partial \eta} \left[\frac{1}{|J| \det(\mathbf{M})} \begin{pmatrix} x_\xi \\ y_\xi \end{pmatrix}^T \mathbf{M} \begin{pmatrix} x_\xi \\ y_\xi \end{pmatrix} \right] \right\}. \end{aligned} \quad (2.31)$$

where $0 < \tau \ll 1$ is a relaxation parameter. A special case of this system is given by

$$\mathbf{M} = wI, \quad (2.32)$$

where w is known as the (scalar) weight function. This is closely equivalent to one-dimensional equidistribution and gives the equations

$$\nabla \cdot \left(\frac{1}{w} \nabla \xi \right) = 0, \quad \nabla \cdot \left(\frac{1}{w} \nabla \eta \right) = 0. \quad (2.33)$$

Finding a mesh which satisfies this is called *Winslow's variable diffusion method* Winslow (1967), Winslow (1981). (We give examples of these in Chapter 3.) In this sense, (2.29) can be considered as a generation of Winslow's method, viz., a *generalized variable diffusion method*.

Harmonic maps

Another method closely related to (2.29) is the method based on *harmonic maps* Dvinsky (1991) It defines the coordinate transformation used for mesh adaptation as a harmonic map minimizing the functional

$$I(\xi, \eta) = \frac{1}{2} \int_{\Omega_P} \sqrt{\det(\mathbf{M})} [\nabla \xi^T \mathbf{M}^{-1} \nabla \xi + \nabla \eta^T \mathbf{M}^{-1} \nabla \eta] \, dxdy, \quad (2.34)$$

where, once again, \mathbf{M} is a matrix-valued monitor function. We note that in this case the matrix-valued function \mathbf{M} cannot be chosen to be a scalar monitor function (cf. Winslow (1967)) as this would lead to no mesh adaptivity in two dimensions. Brackbill and Saltzman (1982) generalise Winslow's idea and defines the needed coordinate transformation by minimizing a combination of three functionals characterizing adaptivity, smoothness, and orthogonality, respectively. Its final functional takes the form

$$\begin{aligned} I(\xi, \eta) = & \theta_a \int_{\Omega_P} w|J| \, dxdy + \theta_s \int_{\Omega_P} (\nabla \xi^T \nabla \xi + \nabla \eta^T \nabla \eta) \, dxdy \\ & + \theta_o \int_{\Omega_P} (\nabla \xi^T \nabla \eta)^2 \, dxdy, \end{aligned} \quad (2.35)$$

where w is the (scalar) weight function and θ_a , θ_s , and θ_o are positive parameters. Notice that the three integrals on the right hand side have different dimensions. As a consequence, the choice of the parameters may depend on specific applications. Directional control is further considered by Brackbill (1993). Variational methods have also been developed based on mechanical models, see Jacquotte (1988), Jacquotte and Coussement (1992) and de Almeida (1999).

A disadvantage of all of the above mentioned methods (e.g. (2.31)) is that the computations have to be done on a highly nonlinear system. In (Ceniceros and Hou 2001) a variational principle using a scalar monitor function is also considered, but this time in the *computational domain*. After certain further simplifications this leads (in two-dimensions) to the equations

$$\nabla_\xi \cdot (M \nabla_\xi x) = 0, \quad \nabla_\xi \cdot (M \nabla_\xi y) = 0. \quad (2.36)$$

Now *all derivatives are expressed in terms of the computational variables* so that $\nabla_\xi = (\partial_\xi, \partial_\eta)$, and the monitor function is considered as a function of the computational coordinates, i.e., $M = M(\xi, \eta)$. A relaxation method is proposed by (Ceniceros and Hou 2001) to solve (2.36) which leads to a set of moving mesh PDEs of the form

$$x_\tau = \nabla_\xi \cdot (M \nabla_\xi x), \quad y_\tau = \nabla_\xi \cdot (M \nabla_\xi y). \quad (2.37)$$

which is significantly simpler than (2.31). The above equation, when discretised, is rather stiff and can also benefit from a degree of mesh smoothing. To do this Cenicerros and Hou (2001) proposed that the following more regular system should be discretised

$$(1 - \gamma \partial_{\xi\xi}^2) x_\tau = \nabla_\xi \cdot (M \nabla_\xi x), \quad (1 - \gamma \partial_{\xi\xi}^2) y_\tau = \nabla_\xi \cdot (M \nabla_\xi y). \quad (2.38)$$

where $\gamma > 0$ is related to M (typically if a time-step of Δt is used then $\gamma = \Delta t \max(M)$). See (Tang 2005), (Zegeling 2007), (Ceniceros 2002) and the examples in Chapter 5 for further implementations of this method. It is very easy to use this method both to move a mesh and to couple it to a PDE. We will give examples of this and further calculations of the variational method in Chapter 3. Dvinsky (1991) also discusses the advantages and disadvantages of formulating the harmonic map method in the physical domain and in the computational domain. However, numerical results show that the method formulated in the computational domain produces cross-over meshes for a non-convex physical domain whereas the method formulated in the physical domain leads to non-singular meshes.

2.7. Mesh quality, isotropy and alignment

The methods discussed in the previous subsection are most based on physical and/or geometric considerations. Although they have been applied with a degree of success to numerical solution of a variety of PDEs, it is unclear how mesh concentration is controlled precisely through the monitor function for these methods. This is important because a clear understanding of the effect of the monitor function on mesh concentration will lead to a better choice of the monitor function as well as a better design of the mesh adaptation method itself. Moreover, neither of the methods nor their choice of the monitor function is directly connected to any sort of error analysis. (A qualitative analysis of the effect of the monitor function on mesh concentration is given by Cao, Huang and Russell (1999) for the functional (2.29).)

A variational method based on appropriate functionals which addresses these issues has been developed based on the equidistribution and alignment conditions (2.23) and (2.24) in Huang (2001), Huang and Sun (2003), Huang

(2007). Recall that for a given matrix-valued monitor function $\mathbf{M}(\mathbf{x})$, the condition (2.23) specifies the size of elements while (2.24) determines the shape and orientation of elements. The main idea of the variational method in this context is to then generate a coordinate transformation that closely satisfies these two conditions.

A functional for mesh alignment.

First consider the alignment condition (2.24). Let the eigenvalues of the matrix $J^{-1}\mathbf{M}^{-1}J^{-T}$ be $\lambda_1, \dots, \lambda_n$. By the arithmetic-mean geometric-mean inequality the desired coordinate transformation can be obtained by minimising the difference between the two sides of the inequality

$$\left(\prod_i \lambda_i \right)^{\frac{1}{n}} \leq \frac{1}{n} \sum_i \lambda_i.$$

Notice that

$$\begin{aligned} \sum_i \lambda_i &= \text{trace}(J^{-1}\mathbf{M}^{-1}J^{-T}) = \sum_i (\nabla \xi_i)^T \mathbf{M}^{-1} \nabla \xi_i, \\ \prod_i \lambda_i &= \det(J^{-1}\mathbf{M}^{-1}J^{-T}) = \frac{1}{(|J| \sqrt{\det(\mathbf{M})})^2}. \end{aligned}$$

Then we have

$$\left(\frac{1}{(|J| \sqrt{\det(\mathbf{M})})^2} \right)^{\frac{1}{n}} \leq \frac{1}{n} \sum_i (\nabla \xi_i)^T \mathbf{M}^{-1} \nabla \xi_i$$

or equivalently

$$\frac{n^{\frac{n}{2}}}{|J|} \leq \sqrt{\det(\mathbf{M})} \left(\sum_i (\nabla \xi_i)^T \mathbf{M}^{-1} \nabla \xi_i \right)^{\frac{n}{2}}. \quad (2.39)$$

Integrating the above inequality over the physical domain yields

$$n^{\frac{n}{2}} \int_{\Omega_C} d\xi \leq \int_{\Omega_P} \sqrt{\det(\mathbf{M})} \left(\sum_i (\nabla \xi_i)^T \mathbf{M}^{-1} \nabla \xi_i \right)^{\frac{n}{2}} dx.$$

Hence, the adaptation functional associated with mesh alignment for the inverse coordinate transformation $\xi = \xi(\mathbf{x})$ can be defined as

$$I_{ali}(\xi) = \frac{1}{2} \int_{\Omega} \sqrt{\det(\mathbf{M})} \left(\sum_i (\nabla \xi_i)^T \mathbf{M}^{-1} \nabla \xi_i \right)^{\frac{n}{2}} dx. \quad (2.40)$$

We remark that the functional (2.40) can also be derived from the concept of conformal norm in the context of differential geometry Huang (2001).

Moreover, in two dimensions ($n = 2$), (2.40) gives the energy of a harmonic mapping (Dvinsky 1991). In this sense, the harmonic map method can be understood as a functional associated with alignment. Similarly, we can take squares on both sides of (2.39) and integrate the resulting inequality over Ω_P . We get

$$n^n \int_{\Omega_P} \frac{\sqrt{\det(\mathbf{M})}}{(|J|\sqrt{\det(\mathbf{M})})^2} dx \leq \int_{\Omega_P} \sqrt{\det(\mathbf{M})} \left(\sum_i (\nabla \xi_i)^T \mathbf{M}^{-1} \nabla \xi_i \right)^2 dx.$$

The resulting functional for alignment then takes the form

$$\tilde{I}_{ali}(\xi) = \int_{\Omega_P} \sqrt{\det(\mathbf{M})} \left(\sum_i (\nabla \xi_i)^T \mathbf{M}^{-1} \nabla \xi_i \right)^2 dx - n^n \int_{\Omega_P} \frac{\sqrt{\det(\mathbf{M})}}{(|J|\sqrt{\det(\mathbf{M})})^2} dx. \quad (2.41)$$

A functional for equidistribution.

We now consider the equidistribution condition (2.23). From Hölder's inequality we have

$$\left(\int_{\Omega_P} \frac{\sqrt{\det(\mathbf{M})}}{|J|\sqrt{\det(\mathbf{M})}} dx \right)^2 = \left(\int_{\Omega_C} d\xi \right)^2 \leq \int_{\Omega_P} \frac{\sqrt{\det(\mathbf{M})}}{(|J|\sqrt{\det(\mathbf{M})})^2} dx,$$

which leads to the functional for equidistribution given by

$$I_{eq}(\xi) = \int_{\Omega_P} \frac{\sqrt{\det(\mathbf{M})}}{(|J|\sqrt{\det(\mathbf{M})})^2} dx. \quad (2.42)$$

An adaption functional based on equidistribution and alignment.

We note that neither of the adaption functionals defined in the previous sections can alone lead to a robust mesh adaptation method because either of them represents only one of the mesh control conditions (2.24) and (2.23). It is necessary and natural to combine them together. A way to achieve this goal is to take an average of the functionals (2.41) and (2.42), i.e.,

$$\begin{aligned} I(\xi) &= \theta \int_{\Omega_P} \sqrt{\det(M)} \left(\sum_i (\nabla \xi_i)^T M^{-1} \nabla \xi_i \right)^n dx \\ &+ (1 - 2\theta) n^n \int_{\Omega_P} \frac{\sqrt{\det(M)}}{(|J|\sqrt{\det(M)})^2} dx, \end{aligned} \quad (2.43)$$

where $\theta \in [0, 1]$ is a parameter. Notice that the two terms in the functional have the same dimension. The balance between them is controlled by a dimensionless parameter θ . When $\theta = 1/2$, only the first term remains. Regarding well posedness, it is noted that the first term of the functional

is convex, and the existence, uniqueness, and the maximal principle for its minimizer are guaranteed; e.g., see (Reshetnyak 1989). It is unclear if this result can apply to the whole functional.

Mesh quality measures: alignment and equidistribution.

Mesh quality measures can also be developed based on the alignment and equidistribution conditions (2.24) and (2.23). Indeed, for a given matrix-valued monitor function $\mathbf{M} = \mathbf{M}(\mathbf{x})$ and a coordinate transformation $\mathbf{x} = \mathbf{x}(\xi)$ (or its inverse), we can use

$$Q_{ali} = \left[\frac{\text{trace}(J^T \mathbf{M} J)}{n \det(J^T \mathbf{M} J)^{\frac{1}{n}}} \right]^{\frac{n}{2(n-1)}}, \quad (2.44)$$

$$Q_{eq} = \frac{|J| \sqrt{\det(\mathbf{M})} |\Omega_C|}{\int_{\Omega_P} \sqrt{\det(\mathbf{M})} dy} \quad (2.45)$$

to measure how closely the coordinate transformation (i.e. mesh) satisfies the alignment and equidistribution conditions (2.24) and (2.23), respectively. We note that Q_{ali} is equivalent to

$$\hat{Q}_{ali} = \left[\frac{\text{trace}(J^{-1} \mathbf{M}^{-1} J^{-T})}{n \det(J^{-1} \mathbf{M}^{-1} J^{-T})^{\frac{1}{n}}} \right]^{\frac{n}{2(n-1)}}. \quad (2.46)$$

The quantity Q_{ali} ranges from 1 to ∞ , with $Q_{ali} \equiv 1$ for the identity mapping, while Q_{eq} takes values in $(0, \infty)$, with $\max_x Q_{eq} = 1$ implying a uniform mesh. Interestingly, Q_{ali} reduces to an equivalence of Q_{geo} when $M = Id$. In this sense, Q_{ali} can be viewed as a geometric quality measure in the metric specified by \mathbf{M} .

2.8. Error control and associated monitor functions

The measures for mesh quality and geometry described in the previous subsection have largely been constructed in the absence of a clear application. For the majority of this article we are considering the effectiveness of a mesh for computing the solution of a partial differential equation. In this case we are expecting to impose some form of discretisation of the system on the mesh. From this discretisation we hope to solve a, typically rapidly evolving, partial differential equation. The mesh so constructed should attempt to minimise error (such as the truncation or the interpolation error) in some way. In this section we consider three forms of error, namely static truncation error, static interpolation error and dynamic errors, and in the first two cases look at monitor functions which lead to reduced errors.. The difficulty in implementing such a procedures is, of course, that not only is it difficult to

measure (or indeed to precisely define) the error during the calculation and adapting the mesh accordingly, but also some of the 'best' adaptive meshes for solving time-dependent problems lead to very stiff differential equations, thus significantly increasing the computational cost of the process and in some cases making the whole calculation significantly unstable. However, it is intuitively reasonable that for solutions with small length scales over part of the domain, and larger length scales elsewhere, then we might expect to gain significant efficiency by using a smaller mesh in the region of high variation. Exactly this observation motivated the important early work of Dorfi and Drury (1987).

Static truncation error and 'optimal' meshes.

The most natural reason for using an adaptive mesh in the context of solving a PDE is to control the truncation error in any discretisation. As examples of two *steady-state* problems for which adaptivity may be required we may wish to solve the Poisson equation

$$-\Delta u = f(x, y, z, \dots) \quad (2.47)$$

for a potentially singular right-hand side f . Alternatively we may seek to solve the singular diffusion equation

$$-\epsilon u'' - c(x)u' = f(x), \quad \epsilon \ll 1 \quad (2.48)$$

An ideal mesh with N points, used to compute the function u is one which leads to low errors, ideally in the case of (2.48) errors which are ϵ -independent and depend only upon N , and which also keeps computational costs low. Essentially we can consider two types of non-uniform mesh for the computation. One type is an *optimal, fitted or a-priori mesh* which is prescribed in advance of the calculation and gives best possible errors for that computation in some appropriate norm. Important examples of this class are the *Shishkin and Bakhvalov meshes* for the singularly perturbed problems (2.48) and optimal meshes for Poisson type problems. In (Babuska and Rheinboldt 1979) a general analysis of such meshes is made in the context of finite element calculations in one-dimension. An adaptive mesh on the other hand attempts to approximate an optimal mesh through equidistributing a suitable monitor function determined during the computation.

It is important to note at this stage that the error in discretising a differential equation (or indeed in interpolating the solution to that equation or a function in general) is a combination of the error that would occur on a uniform mesh together with further errors that arise from the non-uniformity of the mesh (variation in the size of the elements) and (in more than one-dimension) the mesh skewness. The latter errors have to be treated with great care as they can easily dominate the truncation error on the uniform

mesh and make an adapted mesh worse than useless in solving the underlying problem. However, in contrast, a common error in many numerical analysis texts, is to either assume that these errors add together to give a larger error, or that, for example, the error due to the non-uniformity of the mesh is always at a lower order than the error on the uniform mesh and thus dominates the overall calculation. In fact, provided that the mesh function is *suitably smooth*, for example if (in one-dimension) the mesh is quasi-uniform and obeys the condition (2.9) then the three errors can be *at the same order* when expressed in terms of $1/N^p$. In an 'optimal' mesh it may be possible for the errors to cancel each other out to leading order. However, such meshes are usually very hard to construct and require a lot of a-priori information about the solution. Adaptive meshes generally work by bounding the leading order error (regardless of the behaviour of the underlying solution).

We start by looking at both optimal and adaptive non-uniform meshes on which we can pose finite difference discretisation of the Poisson equation

$$-\frac{d^2u}{dx^2} = f(x). \quad (2.49)$$

If the mesh is a function $x(\xi)$ of the computational variable then in the computational domain we have

$$-\frac{1}{J} \left(\frac{u_\xi}{J} \right)_\xi = f(x(\xi)), \quad J = x_\xi. \quad (2.50)$$

The equation (2.50) can then be discretised in the computational domain for which we use the approximation

$$U_j \approx u(X_j) \quad X_j = x(j\Delta_\xi).$$

A natural centred difference approximation to (2.50) then takes the form

$$-\frac{2}{(\Delta\xi)^2} \frac{\left(\frac{U_{j+1}-U_j}{X_{j+1}-X_j} - \frac{U_j-U_{j-1}}{X_j-X_{j-1}} \right)}{X_{j+1}-X_{j-1}} \quad (2.51)$$

with

$$\Delta_i = X_{i+1} - X_i.$$

We now assume that the mesh function $x(\xi)$ has regularity C^2 and exactly equidistributes a scalar monitor function M

Lemma 1. (i) The local truncation error T of the above discretisation is given in the original variables by

$$T = \frac{\Delta_i^2}{3} \left[\frac{x_{\xi\xi}}{x_\xi^2} u_{xxx} + \frac{u_{xxxx}}{4} \right] + \mathcal{O}(\Delta_i^3) \quad (2.52)$$

and in the computational variables by

$$T = \frac{\Delta\xi^2 x_\xi^2}{3} \left[-\frac{M_\xi u_{xxx}}{Mx_\xi} + \frac{u_{xxxx}}{4} \right] + \mathcal{O}(\Delta\xi^3) \quad (2.53)$$

(ii) The truncation error is of second order if the mesh is quasi-uniform so that condition (2.10) is satisfied.

(iii) The truncation error is *zero to leading order* if the mesh equidistributes the monitor function

$$M_{opt} = (u_{xxx})^{1/4} = (-f_x)^{1/4}. \quad (2.54)$$

Proof. Letting $\Delta_j = X_{j+1} - X_j$. A simple Taylor expansion gives

$$U_{j+1} = U_j + \Delta_j u' + \frac{\Delta_j^2}{2} u'' + \frac{\Delta_j^3}{6} u''' + \frac{\Delta_j^4}{24} u'''' + \mathcal{O}(5),$$

$$U_{j-1} = U_j - \Delta_{j-1} u' + \frac{\Delta_{j-1}^2}{2} u'' - \frac{\Delta_{j-1}^3}{6} u''' + \frac{\Delta_{j-1}^4}{24} u'''' + \mathcal{O}(5),$$

where all derivatives of u are expressed in terms of x . Hence, expanding the LHS of (2.51) we obtain (after some manipulation) that the truncation error is given by

$$T = \frac{1}{3} (\Delta_j - \Delta_{j-1}) u''' + \frac{1}{12} \frac{\Delta_j^3 + \Delta_{j-1}^3}{\Delta_j + \Delta_{j-1}} u'''' + \mathcal{O}(3).$$

This error has two components. The second is the usual component (of order Δ_j^2) which is seen on a uniform mesh. The first, is an additional error due to the variation in the size of the mesh. In many texts this is considered to be large (as it is apparently of higher order) however, if Δ_j varies smoothly over the domain then it is actually of the same order as the second error. Since, to leading order

$$\Delta_j = \Delta\xi x_\xi$$

we have, to leading order

$$T = (\Delta\xi)^2 \left(\frac{1}{3} x_{\xi\xi} u''' + \frac{1}{12} x_\xi^2 u'''' \right) + \mathcal{O}(\delta\xi^3).$$

Setting (to leading order) $\Delta_j = \Delta\xi x_\xi$ gives (2.52). Now, from the equidistribution equation we have

$$x_\xi = \frac{\theta}{M}.$$

Hence $M_\xi X_\xi + Mx_{\xi\xi} = 0$. Substituting for M in the above gives (2.53).

Result (ii) follows immediately from the expression (2.52)

The optimal form of M in (iii) arises from setting the leading order term to zero and integrating. Note that this latter calculation can break down if u_{xxx} vanishes at some point. \square

An almost identical calculation to the above leads to the following result.

Lemma 2. (i) If we consider using the standard central difference approximation to u_x given by

$$u_x = \frac{U_{i+1} - U_{i-1}}{X_{i+1} - X_{i-1}}$$

then the truncation error is given in the physical coordinates by

$$T = \frac{\Delta_i^2}{2} \left[\frac{x_{\xi\xi}}{x_\xi^2} u_{xx} + \frac{1}{3} u_{xxx} \right] + \mathcal{O}(\Delta_i^3) \quad (2.55)$$

or in the computational coordinates by

$$T = \frac{\Delta\xi^2 x_\xi^2}{2} \left[-\frac{M_\xi u_{xx}}{M x_\xi} + \frac{1}{3} u_{xxx} \right] + \mathcal{O}(\Delta\xi^3). \quad (2.56)$$

(ii) This error is of second order on a quasi-uniform mesh, and is zero to leading order on an 'optimal mesh' given when

$$M = (u_{xx})^{1/3}. \quad (2.57)$$

These calculations, both of the errors in approximating u_x and u_{xx} and of the possible optimal meshes, are revealing in a number of ways. Firstly, they show that in all such calculations there is a subtle interplay between the mesh variability and the mesh size. This is even more marked in the case of singular perturbation problems. By choosing M very carefully we can exploit this to give very high accuracy and an *optimal mesh*. In general this is not usually possible. Indeed this calculation requires an accurate knowledge of the third derivative of the function u .

Secondly, we can also see from (2.53) the effect of choosing other types of monitor function as part of an adaptive calculation. The optimal mesh eliminates the truncation error to leading order. However, the truncation error is actually an estimate for the second derivative (with respect to x) of the solution error between the calculated solution U_j and $u(X_j)$. To obtain a true estimate, this expression needs to be integrated. A useful expression for the error for both (2.49) and (2.48) (see (Andreev and Kopteva 1998)) is then given by

Lemma 3.

$$\|U_j - u(X_j)\|_\infty \leq C \max \left[\Delta_i^2 \left| \max_{[X_i, X_{i+1}]} |u''| + \Delta_i^2 \right. \right] \quad (2.58)$$

Given suitable a-priori estimates, this error can be bounded by using a monitor function M which controls this via the expression

$$\Delta_i^2 \left(\left| \max_{[X_i, X_{i+1}]} |u''| + 1 \right. \right) = \Delta \xi^2 x_\xi^2 \left(\left| \max_{[X_i, X_{i+1}]} |u''| + 1 \right. \right) = \Delta \xi^2 \left(\left| \max_{[X_i, X_{i+1}]} |u''| + 1 \right. \right) \theta^2 / M^2.$$

This motivates the choice of the curvature dependent monitor function given by

$$M = \sqrt{1 + |u''|^2}.$$

Such a function has been used by (Mackenzie and Robertson 2002), see also (Kopteva 2007), (Kopteva and Stynes 2001). In this case the error *becomes a function of $\Delta \xi$ only* and does not depend upon the solution. Hence it has the great advantage of yielding a mesh for which large variations in u'' (for example at boundary layers) do not affect accuracy. This is good enough for most calculations. Observe however, the difference between using the curvature based monitor function to bound the error, and the optimal monitor function for the Poisson equation which eliminates this error to leading order.

Similar issues arise in the case of the singularly perturbed problems (2.48). For example, it is possible to get very sharp estimates on the solution in certain cases (such as when $c(x) = 1$). In the latter case (Andreev and Kopteva 1998) we have

Lemma 4.

$$\|U_j - u(X_j)\|_\infty \leq C \left[\left\| \min\{\Delta_i^2/\epsilon^2, 1\} e^{-x_{i-1}/\epsilon} \right\|_\infty + \max \Delta_i^2 \right]. \quad (2.59)$$

This error can be completely controlled to be proportional only to $\Delta \xi^2$ using a Bakhvalov mesh. For such problems it can also be shown (Kopteva 2007) that if the monitor function is chosen to be discrete arc-length of the form

$$M = \sqrt{1 + u_x^2},$$

then, provided that the solution has converged closely to an equidistributed one, then the computed solution is first order accurate with errors $\mathcal{O}(\Delta \xi)$ independent of the value of ϵ .

Static interpolation error

In higher dimensions it is very hard to obtain reliable estimates for the truncation error when solving a general PDE. A somewhat easier, but still very important, question to address is whether a mesh is suitable to approximate

the solution of the PDE, in particular to interpolate the solution. We now consider this question.

Suppose that the solution of the differential equation (or indeed any appropriate function defined in the physical domain) is given by $u(x, y, \dots)$. For the case of a problem in two dimensions we can define the point values of u on the non-uniform mesh by

$$U_{i,j} = u(X_{i,j}, Y_{i,j}).$$

A natural measure of error is the *interpolation error* obtained by approximating u on the mesh with suitable functions using the above point values. Significant progress in finding meshes with good properties in reducing the interpolation error of a solution has been made in this direction in the past decade. Formulae giving the optimal monitor function to minimise this error over a suitable mesh have been developed based on interpolation error estimates by Huang and Sun (2003) in the H^m norm, (Chen, Sun and Xu 2007) in the L^q norm, (Huang 2005a), (Huang 2005b) in the $W^{m,q}$ norm, and in Cao (2005), Cao (2007), Cao (2007), (Cao Math Comput 2008) for higher order interpolation in two-dimensions. Formulae have also been developed based on an a-posteriori error estimate for one-dimension (He and Huang, 2008), a hierarchical basis a-posteriori error estimate (Kamenski, Huang, and Lang 2008), and semi a-posteriori error estimates for variational problems (Huang and Li 2008). Formulae for the monitor functions in these cases can be obtained as follows Huang and Sun (2003), Huang (2005a). A so-called anisotropic error bound, taking into consideration the directional effect of the error or solution derivatives, is first developed. This error bound can be regarded as a function of the monitor function M when only meshes satisfying the alignment and equidistribution conditions (2.24) and (2.23) are concerned. Then the optimal monitor function is obtained by minimizing the bound among all possible matrix valued functions \mathbf{M} . Consider a simple situation where a function $u \in H^2(\Omega_P)$ is interpolated by piecewise-linear polynomials on a simplicial mesh (of N elements) and the error is measured in L^2 norm. Then an anisotropic asymptotic bound (as $N \rightarrow \infty$) can be obtained from the interpolation theory in Sobolev spaces as Huang and Sun (2003)

$$\|e_h\|_{L^2(\Omega_P)}^2 \leq C\alpha^2 N^{-\frac{4}{n}} \int_{\Omega_P} (\text{trace}(J^T[I + \alpha^{-1}|H(u)||J]))^2 dx + h.o.t. \quad (2.60)$$

where $H(u)$ denotes the Hessian of function u , $|H(u)| = \sqrt{H(u)^T H(u)}$, and $\alpha > 0$ is an arbitrary number which serves as a regularization parameter and its value will be determined later. Note that a rigorous bound can be obtained. But in the situation the derivation has to go with a discrete form; e.g. see (Huang 2007). Since the procedure is the same for both, for

simplicity we use here the non-rigorous continuous form. Noticing that a mesh satisfying (2.24) and (2.23) (and a proper boundary correspondence) is a function of \mathbf{M} , we can regard the integral on the right hand side of (2.60) as a function of \mathbf{M} , viz.,

$$B(\mathbf{M}) = \int_{\Omega_P} (\text{trace}(J^T[I + \alpha^{-1}|H(u)||J]))^2 dx, \quad (2.61)$$

In the following analysis, we consider only meshes satisfying (2.24) and (2.23) and derive the optimal monitor function by minimizing $B(\mathbf{M})$ amongst all possible matrix-valued functions \mathbf{M} . First we notice that (2.24) is mathematically equivalent to

$$\frac{1}{n} \text{trace}(J^T M J) = \det(J^T \mathbf{M} J)^{\frac{1}{n}}. \quad (2.62)$$

A direct comparison of (2.62) suggests that \mathbf{M} be chosen in the form

$$\mathbf{M} = \theta(x)[I + \alpha^{-1}|H(u)|], \quad (2.63)$$

where $\theta = \theta(x)$ is a scalar function. For matrix-valued monitor functions in this form, (2.62) reduces to

$$\frac{1}{n} \text{trace}(J^T [I + \alpha^{-1}|H(u)||J]) = \det(J^T [I + \alpha^{-1}|H(u)||J])^{\frac{1}{n}}.$$

Inserting this into (2.61) and using Hölder's inequality we have

$$\begin{aligned} B(\mathbf{M}) &= n^2 \int_{\Omega_P} |J|^{\frac{4}{n}} \det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n}} dx \\ &= n^2 \int_{\Omega_C} \left[|J| \det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n+4}} \right]^{\frac{n+4}{n}} d\xi \\ &\geq n^2 |\Omega_C|^{-\frac{4}{n}} \left[\int_{\Omega_C} |J| \det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n+4}} d\xi \right]^{\frac{n+4}{n}} \end{aligned} \quad (2.64)$$

$$= n^2 |\Omega_C|^{-\frac{4}{n}} \left[\int_{\Omega_P} \det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n+4}} dx \right]^{\frac{n+4}{n}}. \quad (2.65)$$

We note that the equality in (2.64) holds when the mesh satisfies

$$|J| \det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n+4}} = \frac{1}{|\Omega_C|} \int_{\Omega_P} \det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n+4}} dy.$$

Comparing this with the equidistribution condition (2.23) we have

$$\sqrt{\det(\mathbf{M})} = \det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n+4}}.$$

From this and (2.63) the optimal matrix-valued monitor function to minimise the interpolation error is given by

$$\mathbf{M} = \det(I + \alpha^{-1}|H(u)|)^{-\frac{1}{n+4}} [I + \alpha^{-1}|H(u)|]. \quad (2.66)$$

Inserting (2.65) into (2.60), the interpolation error bound for a mesh satisfying (2.24) and (2.23) with optimal \mathbf{M} given in (2.66) is then

$$\|e_h\|_{L^2(\Omega_P)}^2 \leq C\alpha^2 N^{-\frac{4}{n}} \left[\int_{\Omega_P} \det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n+4}} dx \right]^{\frac{n+4}{n}} + h.o.t. \quad (2.67)$$

We now discuss how to choose α . We first notice that conditions (2.65) into (2.60) are invariant for scaling transformation of \mathbf{M} of the form $\mathbf{M} \rightarrow c\mathbf{M}$ for any positive constant c . Thus, if $|H(u)|$ is strictly positive definite on Ω_P , we can take $\alpha \rightarrow 0$ in (2.66) and (2.67). This gives

$$\mathbf{M} = \det(|H(u)|)^{-\frac{1}{n+4}} |H(u)|, \quad (2.68)$$

$$\|e_h\|_{L^2(\Omega_P)}^2 \leq CN^{-\frac{4}{n}} \left[\int_{\Omega_P} \det(|H(u)|)^{\frac{2}{n+4}} dx \right]^{\frac{n+4}{n}} + h.o.t. \quad (2.69)$$

When $|H(u)|$ vanishes locally, the monitor function cannot be defined as in (2.66) since the right hand side is not positive definite. In this case, a positive α should be used. Huang (2001) suggests that α be defined implicitly through

$$\int_{\Omega_P} \det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n+4}} dx = 2|\Omega_P|. \quad (2.70)$$

It is easy to show that (2.70) has a unique solution for α . A simple iteration method such as the bisection method can be used for solving this equation. Moreover, when α is defined in this way, \mathbf{M} is invariant for scaling transformation of $|H(u)|$. Furthermore, it is shown in Huang (2001) that about fifty percent of the mesh points are then concentrated in regions where $\det(I + \alpha^{-1}|H(u)|)^{\frac{2}{n+4}}$ is large. Finally, the error bound reads as

$$\|e_h\|_{L^2(\Omega_P)}^2 \leq C\alpha^2 N^{-\frac{4}{n}}. \quad (2.71)$$

From (2.70) it is not difficult to show that, for $n \leq 4$, α is bounded as

$$\left[\frac{1}{2|\Omega_P|} \int_{\Omega_P} \det(|H(u)|)^{\frac{2}{n+4}} dx \right]^{\frac{n+4}{2n}} \leq \alpha \leq \left[\frac{1}{n^{\frac{2n}{n+4}} |\Omega_P|} \int_{\Omega_P} (\text{trace}(|H(u)|))^{\frac{2n}{n+4}} dx \right]^{\frac{n+4}{2n}}. \quad (2.72)$$

Dynamic error

Usually when we apply a moving mesh method we are interested in solving a time evolving PDE. This leads to additional dynamic errors (Li and Petzold 1997),(Li,Petzold and Ren 1998) such as oscillations around rapidly evolving fronts or miscalculations of the front speed. These depend significantly on

the way that the mesh is updated and coupled to the PDE. We consider these in more detail in the next Chapter when we look at how the moving mesh equations are coupled to the underlying PDE.

2.9. Monitor function smoothing and regularisation

Having considered the mesh quality, we now return to further considerations of the monitor function and of mesh smoothness. Recall that for one-dimensional problems it is essential that the mesh should be quasi-uniform in order to have a low truncation error. Smoothing a mesh either directly, or indirectly through smoothing/averaging the monitor function aims to achieve this.

The Dorfi and Drury Method

A direct approach to smoothing a one-dimensional mesh derived from an equidistribution principle) is proposed in Dorfi and Drury (1987) and is often called the Dorfi and Drury method Li and Petzold (1997). In this method, if

$$n_i = (\Delta X_i)^{-1} \equiv (X_{i+1} - X_i)^{-1}$$

then a smoother mesh is given by computing \hat{n}_i where

$$\hat{n}_i = n_i - \gamma(n_{i+1} - 2n_i + n_{i-1}) \quad (2.73)$$

for a suitable constant γ . A variant of this, considered in Li and Petzold (1997) is given by updating the mesh differences by

$$\Delta \hat{X}_i = \sum_{j=i-p}^{i+p} \theta^{|i-j|} \Delta X_j$$

for a suitable constant $0 < \theta < 1$. There are many other strategies for direct mesh smoothing. For example it is possible to use a-posteriori estimates of the solution on the mesh to do this Bank and Smith (1997).

Monitor function smoothing

Alternatively we can generate a smoother mesh by averaging the monitor function prior to the mesh calculation. Suppose that point values of (a scalar or matrix-valued) monitor function $M_{i,j}$ are given. A Jacobian-type strategy, also referred to as averaging or low-pass filtering in literature, is commonly used, e.g. see Dorfi and Drury (1987), Verwer,Blom,Furzeland and Zegeling (1989), and Huang and Sloan (1994). When a rectangular computational mesh is used, this smoothing can be conveniently expressed as

$$\hat{M}_{i,j} = \frac{\sum_{k=-1}^1 \sum_{l=-1}^1 M_{i+k,j+l} \gamma^{|k|+|l|}}{\sum_{k=-1}^1 \sum_{l=-1}^1 \gamma^{|k|+|l|}},$$

where $\gamma \in (0, 1)$ is a parameter. This type of local smoothing of the monitor function can be viewed as an approximation of Laplace operator based smoothing:

$$(I - \lambda^{-2} \Delta_\xi) \hat{M} = M,$$

where λ is a parameter. In one dimension (Huang and Russell 1997) show that, when λ is chosen as a value of order $O(N)$, where N is the number of sub-intervals, a mesh equidistributing \hat{M} is *locally quasi-uniform*, indeed there exists a reasonably small constant $\nu \geq 1$ such that

$$\frac{1}{\nu} \leq \frac{X_{j+1} - X_j}{X_j - X_{j-1}} \leq \nu \quad \forall j$$

Another interesting strategy is to use a reference Jacobian matrix Knupp (1996) and Knupp, Margolin and Shashkov (2002) where a new mesh is generated to have a close Jacobian matrix to the reference one that is typically obtained from a reference, often non-smooth mesh.

50:50 meshes and the Mackenzie regularisation

A recurring problem with r -adaptive meshes which equidistribute a poorly chosen monitor function is that they can concentrate points in areas of particular identified interest where high resolution is needed, but leave other regions sparse of points. This can lead not only to low solution resolution in such areas, but also to a severe lack of mesh regularity and consequent large errors caused by a too rapid mesh variation. An example of such would be the calculation of the solution of a system with is blowing up in finite time with a large peak, in which all of the mesh points are concentrated in the peak alone. Such problems can be significantly reduced if the mesh is designed so that roughly half of the points are concentrated in the areas where high resolution is required, and half where it is not. Such meshes are called 50:50 meshes (see (Budd, Carretero-Gonzalez and Russell 2005)) and a regularisation of M to insure that such meshes arise in practice has been proposed by (Beckett and Mackenzie 2000). To show how such problems arise in a calculation in n -dimensions, suppose that we have a scalar monitor function M for which $\int_{\Omega_P} M d^n x = \theta$. Consider now the situation in which there are two subsets A and B of Ω_P , with $\Omega_P = A \cup B$ so that the monitor function is designed to concentrate points in a small region A (so that A may be the support of a singularity or of a front). The preimage $A' = F^{-1}(A) \subset \Omega_C$ represents those points in the computational domain which are mapped to A , with a similar set B' . Suppose now that $|A'|$ and $|B'|$ are the areas of these sets in Ω_C , with respective areas $|A|$ and $|B|$ in Ω_P . These areas measure the proportion of mesh points allocated to the corresponding sets A and B . Note that in most applications of an adaptive method, where mesh

points have to be concentrated into a small region we would expect that

$$|A'| = \mathcal{O}(1), \quad |A| = \mathfrak{o}(1), \quad |B'| = \mathcal{O}(1), \quad |B| \approx |\Omega_P|. \quad (2.74)$$

Problems arise with mesh regularity and solution resolution away from the set A if $|B'| \ll |A'|$. It follows immediately from the equidistribution principle that if $\theta = \int_{\Omega_P \equiv A \cup B} M d^d x$ then

$$\Lambda = \frac{|A'|}{|B'|} = \frac{\int_A M d^n x}{\int_B M d^n x} = \frac{\theta}{\int_B M d^n x} - 1.$$

Furthermore

$$|A'| = \frac{\int_A M d^n x}{\theta}.$$

It follows from the conditions on A' and A in (2.74) that over the set A we have $M \gg \theta$. However, if the monitor function is so constructed such that over the set B we have $M \ll \theta$ and hence $\int_B M d^d x \ll \theta$ (so that the integral of M is concentrated in A) then Λ will be very large and the mesh will lose regularity. Indeed we will have $|A'| \approx 1$. Exactly such problems arose in some of the blow-up calculations reported in (Budd, Huang and Russell 1996). In such cases we must replace M by the regularised function introduced by (Beckett and Mackenzie 2000) and given by

$$\hat{M} = M + \frac{\theta}{|\Omega_P|}. \quad (2.75)$$

Observe that over the set A we have $\hat{M} \approx M$ and over B we have $\hat{M} \approx \theta/|\Omega_P|$ and trivially

$$\int_{\Omega_P} \hat{M} d^n x = 2\theta.$$

Consequently when we make use of the regularised monitor function \hat{M} to define the mesh we have

$$|A'| = \frac{\int_A \hat{M} d^n x}{2\theta} \approx \frac{\int_A M d^n x}{2\theta} \approx \frac{1}{2},$$

and

$$\Lambda \approx \frac{2|\Omega_P|}{|B|} - 1 \approx 1.$$

This gives the desired 50:50 quality to the mesh.

3. Location based moving mesh methods

In this Chapter we will look in greater detail at the various methods described in the previous chapter under the general heading of *location based methods*. These are those methods which determine the *location* (or more precisely the density) of the mesh points, typically through solving some form of nonlinear differential equation through some form of gradient flow method. The latter can be hard to solve. However, the advantage of these methods is that they tend to give meshes with good global properties, avoiding excessive skewness. We will consider in detail various of the methods outlined in the previous chapter, such as MMPDE based methods, variational methods and optimal transport methods.

3.1. MMPDE Methods in one-dimension

Moving mesh partial differential equation **MMPDE** based methods are now universally used as a means of r-adaptivity in one dimension, and have been incorporated into codes such as MOVCOL and AUTO. There are many different MMPDES, which together encapsulate most of the methods used to derive adaptive meshes in one-dimension.

We consider a one-dimensional map $x(\xi, t)$ from $[0, 1]$ to $[a, b]$ with associated mesh points $X_i = x(i\Delta\xi, t)$ which equidistributes the monitor function M . This map satisfies the equidistribution equation

$$Mx_\xi = \theta, \quad x(0, t) = a, \quad x(1, t) = b, \quad \theta = \int_a^b M dx. \quad (3.1)$$

Lemma 5. The equidistribution equation (3.1) has a unique monotone increasing solution $x(\xi, t)$ for all $M > 0$.

Proof. Integrating (3.1) with respect to ξ and changing variables gives

$$\int_a^x M dx' = \theta\xi \quad \text{or} \quad \int_{X_{i-1}}^{X_i} M dx = \frac{1}{N} \int_a^b M dx. \quad (3.2)$$

Now, as $M > 0$ the LHS of this expression is a *monotone increasing* function of x . It immediately follows that x is a unique monotone increasing function of ξ . Observe further that this function is as smooth as the function M . \square

This simple observation makes equidistribution relatively easy in one-dimension. The most direct way to enforce equidistribution is to solve (3.1) directly. However this has the disadvantage that it requires the calculation of the integral θ . This can be avoided by a further differentiation with respect to ξ , and thus solving the moving mesh equation (together with boundary

conditions) given by

$$(Mx_\xi)_\xi = 0, \quad x(0, t) = a, \quad x(1, t) = b. \quad (3.3)$$

To determine an equidistributed mesh the equation (3.3) can be discretised over the computational domain and then solved. This discretisation does not have to be done to high accuracy in order to obtain a regular mesh. A typical such discretisation takes the form

$$E_i \equiv \frac{2}{\Delta\xi^2} (M_{i+1/2}(X_{i+1} - X_i) - M_{i-1/2}(X_i - X_{i-1})) = 0,$$

$$M_{i+1/2} = \frac{1}{2}(M_i + M_{i+1}). \quad (3.4)$$

However, the solution of the system (3.11) requires solving a system of non-linear equations which is usually difficult and requires the use of some form of iterative procedure. See Pryce (1989), Xu, Huang, Russell and Williams (2008), He and Huang (2008), Kopteva and Stynes (2001), Kopteva (2007) for a discussion of such methods, and conditions for them to converge to a solution.

This problem can be avoided, by instead introducing a natural time evolution into the mesh equations. Differentiating the equidistribution equation (3.3) with respect to time gives the (so called) MMPDE0 (Huang, Ren and Russell 1994)

$$\frac{d}{dt} \left((Mx_\xi)_\xi \right) = 0 \quad (3.5)$$

Instead we may also differentiate (3.1) with respect to time (Adjerid and Flaherty 1986). This leads directly to the GCL method described in the next chapter. The resulting equation then takes the form

$$\frac{\partial}{\partial\xi} (Mx_t) + M_t x_\xi = \theta_t \quad (3.6)$$

This equation can then also be differentiated with respect to ξ to eliminate the θ contribution, giving MMPDE1 (Huang, Ren and Russell 1994), where it is assumed that we can find M_t although in practice this may not be easy.

Starting from any mesh (uniform or otherwise) we can evolve the mesh by solving MMPDE0 (3.5) or MMPDE1. Unfortunately, a uniform mesh does not necessarily satisfy the equidistribution equation (3.1). Furthermore, even if a mesh does exactly satisfy it at some time, solving a discretised form of (3.5) inevitably leads to meshes that drift away from an equidistributed

state. Both of these can lead to problems with mesh crossing (the one-dimensional version of mesh tangling) which occurs when $x_\xi = 0$.

As an example of this, which also demonstrates the general applicability of the method, we consider solving (3.5) starting from an initially *uniform mesh* on $[0, 1]$ for which $x_\xi = 1$. In this calculation we will assume that we have a time evolving monitor function $M(x, t)$ with $M(x, 0) \equiv M^0(x)$. It follows from integrating (3.5) with respect to t and applying the initial conditions, that for all time we have

$$(Mx_\xi)_\xi = M_\xi^0.$$

Hence, integrating again we have

$$Mx_\xi = M^0 + B(t),$$

for some function $B(t)$. This can be determined by integrating this expression with respect to ξ to give

$$Mx_\xi = M^0 + \theta(t) - \theta(0).$$

As $M^0 > 0$ it follows that if θ is increasing in time then $x_\xi > 0$. However, if θ decreases with t then it is quite possible for x_t to vanish (initially at the point where M^0 takes its minimum value) and for mesh crossing (tangling) to result.

Such problems can be avoided (both in one and in higher dimensions) by introducing a relaxation time into the solution of (3.3). The philosophy behind doing this is that the equation (3.3) needs not be solved exactly, to obtain a mesh which is perfectly reasonable for any computation. What is more important is that the mesh evolves *at least as fast as any significant features of the solution*. Exactly the same philosophy applies to moving meshes in any number of dimensions. Thus it is possible to consider meshes which relax towards an equidistributed mesh, provided *the relaxation time is smaller than the natural time-scale of the solution*. Ideally the relaxation time should be of a similar order to that of the solution evolution. This prevents the mesh equations becoming unnecessarily stiff. Various different forms of mesh relaxation are possible.

The most obvious way of relaxing towards an equidistributed state was proposed by Anderson and Rai (1983) who computed the mesh through a relaxation equation, based on considering pseudo-forces between the mesh points, given by

$$\epsilon \dot{x} = (Mx_\xi)_\xi \tag{3.7}$$

where $\epsilon > 0$ is presumed to be small. Alternatively, we can consider the original equidistribution equation in integral form. If a mesh is not exactly

equidistributed then we can determine the residual

$$R = \int_a^x M dx - \xi \int_a^b M dx.$$

If we then set $\epsilon \dot{x} = -R$ and differentiate this expression twice with respect to ξ we obtain

$$\epsilon (\dot{x})_{\xi\xi} = -(Mx_\xi)_\xi. \quad (3.8)$$

This equation was originally derived in Adjerid and Flaherty (1986). The equations (3.7) and (3.8) are known respectively (Huang, Ren and Russell 1994) as MMPDE5 and MMPDE6. We can combine them to give the (smoothed) moving mesh equation considered in Huang and Russell (1997) (see also Cenicerros and Hou (2001) and the discussion in Chapter 2) which takes the form

$$\epsilon \left(1 - \gamma \frac{\partial^2}{\partial \xi^2} \right) \dot{x} = (Mx_\xi)_\xi. \quad (3.9)$$

Here γ can be chosen to give some control over the smoothness of the mesh. The equation (3.9) (and its various discretisations) is very dissipative, and leads to extremely stable meshes under most discretisations. The equation (3.9) also has natural extensions to higher dimensions both in the context of the methods described in Huang and Russell (1997) and Cenicerros and Hou (2001) and also in the Optimal Transport methods we consider later in this chapter. Other smoothed versions of the MMPDEs have also been considered by Huang and Russell (1997) One of them (i.e. equation (25) in the reference) is given by

$$\epsilon \left(1 - \gamma \frac{\partial^2}{\partial \xi^2} \right) \frac{\partial}{\partial \xi} \left(- \left(\frac{\partial x}{\partial \xi} \right)^{-2} \frac{\partial \dot{x}}{\partial \xi} \right) = - \left(1 - \gamma \frac{\partial^2}{\partial \xi^2} \right) \frac{\partial}{\partial \xi} \left(\frac{\partial x}{\partial \xi} \right). \quad (3.10)$$

Huang and Russell (1997) have proved that the solutions (i.e. the coordinate transformation and the mesh) to the continuous equation (3.10) using a central finite difference discretization have the properties both of local quasi-uniformity and no node-crossing. (Theorems 3.1 and 4.1 in the reference.)

NOTE Many other MMPDES have been derived such as MMPDE2 given by

$$(M\dot{x})_{\xi\xi} = -(M_t x_\xi)_\xi - \frac{1}{\epsilon} (Mx_\xi)_\xi.$$

however, we will focus our discussion on the more widely used (3.9).

The moving mesh equation evolves a mesh towards an equidistributed state satisfying (3.3). When implementing the MMPDE method, the equation (3.9) is typically discretised over the computational space leading to a set of ordinary differential equations for the location of the mesh points X_i .

These can then be solved using standard stiff ODE software eg. by using an SDIRK method. A simple such semi-discretisation of (3.9) is given by

$$\epsilon \left(\dot{X}_i - \gamma \frac{\dot{X}_{i+1} - 2\dot{X}_i + \dot{X}_{i-1}}{(\Delta\xi)^2} \right) = E_i(t) \quad (3.11)$$

where the equidistribution measure $E_i(t)$ is as given in (3.11). This leads (on inversion of the simple tri-diagonal system on the right-hand side of this equation) to a simple set of ODES for the location of the mesh points. Alternatively, a simple full discretisation of (3.9) for a mesh X_i^n evaluated at the time $t_n = n\Delta t$ is proposed in Cenicerros and Hou (2001) and takes the form

$$\begin{aligned} \epsilon \left(X_i^{n+1} - \gamma \frac{X_{i+1}^{n+1} - 2X_i^{n+1} + X_{i-1}^{n+1}}{(\Delta\xi)^2} \right) = \\ \epsilon \left(X_i^n - \gamma \frac{X_{i+1}^n - 2X_i^n + X_{i-1}^n}{(\Delta\xi)^2} \right) + \Delta t E_i^n \end{aligned} \quad (3.12)$$

These equations for the mesh can then be solved together with a suitable discretisation of the underlying PDE, either simultaneously or alternately. We will give more details of this procedure later in this chapter in the context of moving meshes in higher dimensions, but we note at this stage, that the simultaneous solution method is both possible and effective in such one-dimensional problems.

The method for evolving the mesh is typically implemented in two stages.

- 1 Starting from an *initially uniform mesh* in the physical space we evolve this to equidistribute the monitor function *at the initial time* over Ω_P . To do this we set $M_0(x) \equiv M(x, 0)$, with $x_\xi = a + (b - a)\xi$, and solve (3.36) with M fixed to equal the function M_0 , with $\epsilon = 1$ for $0 < t < T$, where T is a fixed time. In this first calculation t is an *artificial time* during which the uniform mesh evolves toward an equidistributed mesh for which the right hand side of (??) is zero. It follows from the earlier results that provided $M_0 > 0$ such a mesh exists and we show presently that it is stable. In this initial calculation the right-hand side of (3.9) is initially relatively large. and taking $\epsilon = 1$ prevents the numerical calculation of the solution of the ODES for the mesh point locations from being unnecessarily stiff. The value of T is chosen large enough to allow the mesh to relax toward the equidistributed state.
- 2 We then solve (3.9) with the true *time dependent* monitor function $M(x, t)$ with t now *actual time*. For this calculation we typically set $\epsilon = 0.01$. We show presently that the resulting mesh is then ϵ -close to a mesh which exactly equidistributes $M(x, t)$, provided that M does not

change too rapidly with time). As this procedure starts from a mesh which exactly equidistributes $M(\mathbf{X}, 0)$ the right-hand side of (3.9) is always close to zero and the resulting differential equations are not especially stiff.

Observe that this algorithm has the convenience of *starting from a uniform mesh*. This is a significant advantage over method based on (3.5) or related methods such as GCL method described in Chapter 4.

We will discuss later in this chapter the exact mechanism by which this algorithm for moving the mesh is coupled to the solution method for the underlying PDE.

This procedure have been criticised (for example see (Tang 2005)) for being imprecise about the way that ϵ is defined and the possibility of having to solve a very stiff system of equations. However it can be given a very precise meaning. In the second stage of this calculation we are trying to find a mesh which is close to an equidistributed mesh. The natural timescale τ over which this mesh evolves is given simply by

$$\tau \approx \frac{\epsilon}{M}. \quad (3.13)$$

The key factor governing the choices of both ϵ and M is to then *ensure that τ is smaller, but of the same order as the natural evolutionary time-scale of the underlying PDE*. In Chapter 5 we will show that in the context of PDEs with a strong scaling structure, this allows a natural choice to be made for both ϵ and M .

We now substantiate some of the claims made above, as well as stating another important property of the solutions of the moving mesh PDE (3.9).

- Theorem 2.** (i) If $M_t = 0$ then the equidistributed mesh is a solution of (3.9) and is *linearly stable*.
(ii) If $M_t = \mathcal{O}(1)$ then an initially ϵ -close to equidistributed solution of (3.9) remains ϵ -close for all subsequent times.
(iii) At all times the solution of (3.9) satisfies $x_\xi > 0$ so that mesh crossing (tangling) does not occur.

NOTE This all applies for exact solutions of (3.9). If a too coarse discretisation is used to approximate these solutions then (iii) above may be violated (?).

Proof. (i) Let \hat{x} be an equidistributed mesh satisfying $(M(\hat{x})\hat{x}_\xi)_\xi = 0$, with $\hat{x}_t = M_t = 0$. It follows immediately that $\epsilon(\dot{\hat{x}} - \gamma\dot{\hat{x}}_{\xi\xi}) = (M\hat{x}_{xi})_\xi$ so that \hat{x} satisfies the equidistribution equation. Now set $x = \hat{x} + R(\xi, t)$ with $R \ll 1$ and $R(a) = R(b) = 0$. To leading order R satisfies the equation

$$\epsilon(\dot{R} - \gamma R_{\xi\xi}) = (MR_\xi)_\xi + (M_x x_\xi R)_\xi = (MR_\xi + M_\xi R)_\xi = (MR)_{\xi\xi}.$$

Therefore

$$\epsilon \dot{R} = (1 - \gamma \partial_\xi^2)^{-1} (MR)_{\xi\xi} \equiv G(MR)_{\xi\xi}. \quad (3.14)$$

Here G is a *positive compact operator* and, as $M > 0$ $ER \equiv (MR)_{\xi\xi}$ is a uniformly elliptic operator with a negative real spectrum. It follows that R must decay to zero. Hence the equidistributed solution is locally stable.

(ii) To prove this result, consider a slowly varying monitor function $M(x, t)$ and an exact solution \hat{x} of the equidistribution equation $(M\hat{x}_\xi)_\xi = 0$. If $x = \hat{x} + \epsilon R$ is a solution of (3.9) then, extending the calculation in (3.14) we see that R satisfies the equation

$$\epsilon(\dot{x} - \gamma \dot{x}_{\xi\xi}) + \mathcal{O}(\epsilon^2) = \epsilon(MR)_{\xi\xi} + \mathcal{O}(\epsilon^2).$$

Hence, we have

$$(MR)_{\xi\xi} = \dot{\hat{x}} - \gamma \dot{\hat{x}}_{\xi\xi} + \mathcal{O}(\epsilon).$$

But as $(M\hat{x}_\xi)_\xi = 0$ we have

$$(M\dot{\hat{x}})_\xi = -(M_t \hat{x}_\xi)_\xi.$$

As the operator $E\phi \equiv (M\phi)_{\xi\xi}$ is uniformly elliptic, it follows that provided \dot{M} is of order one, then $\dot{\hat{x}}$ and hence R and its derivatives are also of order one. Consequently, the solution x of (3.9) stays ϵ -close to the solution of the equidistribution equation.

(iii) To show this we need to show that X_ξ cannot vanish. This result is a consequence of the maximum principle. In the case when $\gamma = 0$ we have, on differentiating (3.9) that

$$\dot{X}_\xi = M_{\xi\xi} X_\xi + 2M_\xi X_{\xi\xi} + M X_{\xi\xi\xi}.$$

Suppose that X_ξ is initially positive everywhere, and as x evolves it vanishes for a first time at (wolog) the point $\xi = 0$. Then locally close to this point we have $X_\xi = a\xi^2 + \mathcal{O}(\xi^3)$ for some $a > 0$. Hence

$$\dot{X}_\xi = a\xi^2 M_{\xi\xi} + 2a\xi M_\xi + aM + \mathcal{O}(\xi).$$

Thus $\dot{X}_\xi > 0$ at this point and time. Hence X_ξ must remain positive. The more general result follows from the positivity of the compact operator G . \square

Coupling a one-dimensional MMPDE method to a partial differential equation.

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. The mesh equations

and the PDE equations can then be solved together or alternately. We will discuss this in more detail presently in the context of moving mesh methods in higher dimensions, but it is appropriate to make some preliminary remarks here.

A. Finite difference methods

To motivate the discussion of appropriate discretisations, we assume that the underlying PDE system takes the form

$$\mathbf{u}_t = f(t, x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}). \quad (3.15)$$

If $x(\xi, t)$ is itself a time dependent function of a computational variable ξ then (3.15) can be cast into the Lagrangian form in the moving coordinate system given by

$$\frac{d\mathbf{u}}{dt} = f(t, x, \mathbf{u}, \mathbf{u}_x, \mathbf{u}_{xx}) + \mathbf{u}_x x_t. \quad (3.16)$$

The MMPDE governing the mesh motion gives a direct value for x_t . A method effective for solving (3.16) (in one-dimensional problems) is to use a semi-discretisation. In this approach we discretise the differential equation (3.16) in the computational coordinates together with a similar discretisation of the MMPDE (3.9). In such a semi-discretisation we set

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

As a simple example of the use of a finite difference method we can then take

$$u_x \approx \frac{U_{i+1} - U_{i-1}}{X_{i+1} - X_{i-1}} \quad \text{and} \quad u_{xx} \approx \frac{\frac{U_{i+1} - U_i}{X_{i+1} - X_i} - \frac{U_i - U_{i-1}}{X_i - X_{i-1}}}{\frac{X_{i+1} - X_{i-1}}{2}}. \quad (3.17)$$

These discretisations can then be substituted into (3.16) and the resulting set of ODES for X_i and U_i solved along with one of the discretisations of (3.9). We discuss presently and in more detail the various alternating and simultaneous approaches for discretising in time and then solving the resulting combined system.

On a static mesh, the truncation errors in calculating these finite difference approximations were given in the expressions (2.52) and (2.55). Provided that the stretching condition (2.10) is satisfied then these errors are of second order.

We note, however, that additional errors may arise from the additional convective terms arising from the mesh movement, in particular the term

$$\mathbf{u}_x x_t \quad (3.18)$$

arising in (3.16). This additional term leads 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 (3.15) (such as symmetries, Hamiltonian structure and/or conservation laws) may not be inherited by the Lagrangian form (3.16). A practical difficulty, observed by Li, Petzold and Ren (1998) arises from certain discretisations of (3.18). In particular it was shown in this paper that it is possible that these can lead to instabilities and degrade the accuracy of the calculation. For example, if a centred finite difference approximation is used to discretise u_x then from the expression (2.55) we have an additional truncation error given to leading order by (Li, Petzold and Ren 1998)

$$\dot{x} \frac{\Delta \xi^2}{2} \left[\frac{x_{\xi\xi}}{x_\xi^2} u_{xx} + \frac{1}{3} u_{xxx} \right]. \quad (3.19)$$

It was observed in (Li, Petzold and Ren 1998) that as $x_{\xi\xi}$ can 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 destabilising, and also potentially quite large. It was considered in Li, Petzold and Ren (1998) that this contributed to some large errors and instabilities arising in their calculation of the front solutions to Burgers' equation. Such problems were also observed in calculations of the nonlinear Schrödinger equation reported in Cenicerros (2002). Various strategies can be used to overcome such problems. These include increasing the mesh density in the unstable region, using a higher order upwind strategy such as an ENO or Roe scheme (Li and Petzold 1997), (Li, Petzold and Ren 1998) or a higher order (fourth order) centred difference scheme (Cenicerros 2002). (Alternatively a static rezoning method can be used as discussed in the next sections). An alternative strategy in one-dimension is to use a collocation, which deals with errors on non-uniform meshes very effectively and we now describe this.

B. Collocation methods

Collocation gives a powerful method of discretising the underlying partial differential equation in the physical domain which has the significant advantage over finite difference and finite element methods that by using collocation we are able to avoid the problem of approximating high order derivatives over a widely non-uniform mesh (Saucez, Vande Vouwer and Zegeling 2005). A very effective collocation discretisation procedure coupled to various possible MMPDEs is adopted in the moving mesh collocation code MOVCOL described in Huang and Russell (1996) (with extensions to higher order systems given in the code MOVCOL Russell, Williams and Xu (2007)) and this package has been used in many tests of adaptive methods in one-dimension, see for example Huang and Russell (1996), citeBCR:99. The collocation methods work by using a piece-wise Hermite polynomial approximation to

the solution $u(x, t)$ of the Lagrangian form of the underlying partial differential equation on a series of intervals $x \in [X_{i-1}(t), X_i(t)]$, with collocation points chosen to be the Gauss points within the intervals. The interval points then become the mesh points moved by solving the MMPDE. In one-dimensions the compactness of the system allows for the possibility of a direct coupling of the Lagrangian system (3.16) with the MMPDE (3.9). These can be solved simultaneously, with no need for mesh interpolation. The underlying PDE (3.16) is then discretised by using the (high-order) collocation procedure with generally a lower order method (such as a finite difference method) used to discretise the MMPDE. The resulting ODEs are somewhat stiff and are typically solved using an appropriate stiff solver such as an SDIRK (Singly Diagonally Implicit Runge-Kutta) or a BDF (Backward Differentiation Formula) method. In MOVCOL they are solved using a BDF method in the code `dassl` (Petzold 1982).

3.2. MMPDE and Variational Methods in n -dimensions

In the variational and MMPDE approaches of mesh adaptation in n -dimensions, adaptive meshes are also generated as images of a computational mesh under a coordinate transformation from the computational domain to the physical domain. Such a coordinate transformation is determined by a adaptation functional which is commonly designed to measure the difficulty in the numerical approximation of the physical solution. The functional often involves mesh properties and employs a monitor function to control mesh quality and mesh concentration.

The key to the development of variational and MMPDE methods is to formulate the adaptation functional. Upon the well-posedness consideration, people usually do not directly use standard error estimates since they lead to non-convex functionals in two and higher dimensions. Instead, most of the existing methods have been developed based on physical, geometric, mesh quality control, and/or other considerations.

Denote the physical and computational coordinates by $x = (x_1, \dots, x_d)^T$ and $\xi = (\xi_1, \dots, \xi_d)^T$, respectively. The functional can be formulated in terms of either the coordinate transformation $x = F(\xi, t)$ or its inverse transformation $\xi = F^{-1}(x, t)$. The latter has been used more commonly than the former because it less likely produces mesh cross-over for non-convex domains (e.g. see (Dvinsky 1991)). In the latter case, the adaptation functional takes a general form as

$$I(\xi) = \int_{\Omega_P} G(\mathbf{M}, \xi, \nabla \xi) dx, \quad (3.20)$$

where G is a continuous function of three arguments and \mathbf{M} is the monitor function, and ∇ is the gradient operator with respect to the physical coordinate x .

Once a functional have been defined, an MMPDE can be obtained as the gradient flow equation of the functional (Huang and Russell 1999a and 1999b) (cf. (2.26)), i.e.,

$$\frac{\partial \xi_i}{\partial t} = -\frac{P}{\tau} \frac{\partial I}{\partial \xi_i}, \quad i = 1, \dots, d \quad (3.21)$$

where P is a positive differential operator and $\tau > 0$ is a parameter for adjusting the time scale of mesh movement. For the general form (3.20), this becomes

$$\frac{\partial \xi_i}{\partial t} = \frac{P}{\tau} \left(\nabla \cdot \frac{\partial G}{\partial (\nabla \xi_i)} - \frac{\partial G}{\partial \xi_i} \right), \quad i = 1, \dots, d. \quad (3.22)$$

Since $\xi = \xi(x, t)$ does not explicitly define the location of mesh points, a mesh equation for $x = x(\xi, t)$ is commonly used in actual computation. Such an equation can be obtained by interchanging the dependent and independent variables in (3.21) or (3.22); e. g., see (2.31) and (Huang JCP 2001, Practical aspects ...).

Most of variational methods can be straightforwardly extended from two dimensions to d -dimensions. For example, Winslow's variable diffusion method (Winslow, 1981, Tech Report) read as

$$I(\xi) = \frac{1}{2} \int_{\Omega_P} \frac{1}{w} \sum_i (\nabla \xi_i)^T \nabla \xi_i dx, \quad (3.23)$$

where w is the weight function prescribed by the user. A generalized version of the this method (Huang and Russell 1999a and 199b) (also see (2.29))

$$I(\xi) = \frac{1}{2} \int_{\Omega_P} \sum_i (\nabla \xi_i)^T \mathbf{M}^{-1} \nabla \xi_i dx, \quad (3.24)$$

where \mathbf{M} is the monitor function in d -dimensions. Obviously, (3.24) reduces to (3.23) when $\mathbf{M} = wI$. The harmonic map method (Dvinsky JCP 1991) (cf. (2.34)) reads as

$$I(\xi) = \frac{1}{2} \int_{\Omega_P} \sqrt{\det(\mathbf{M})} \sum_i (\nabla \xi_i)^T \mathbf{M}^{-1} \nabla \xi_i dx, \quad (3.25)$$

while the method of Brackbill and Sattmann (JCP 1982) (cf. (2.35)) takes the form

$$I(\xi) = \theta_a \int_{\Omega_P} w |J| dx + \theta_s \int_{\Omega_P} \sum_i (\nabla \xi_i)^T \nabla \xi_i dx + \theta_o \int_{\Omega_P} \sum_{i \neq j} ((\nabla \xi_i)^T \nabla \xi_j)^2 dx. \quad (3.26)$$

Following Winslow (JCP 1967), Thompson et al. (1985 book) use a system of elliptic differential equations for generating body-fitted, adaptive meshes.

They propose to use the Poisson equations

$$\nabla^2 \xi_i = P_i(x)$$

to control the mesh concentration and direction, where P_i , $1 \leq i \leq d$, are control functions. The system can be interpreted as the Euler-Lagrange equation of the quadratic functional

$$I(\xi) = \int_{\Omega_P} \sum_i (|\nabla \xi_i|^2 - P_i \xi_i) dx. \quad (3.27)$$

Knupp and his coworkers (Knupp, JCP 1995; Kunn, 1996, SISC; Knupp and Robidoux, SISC, 2000; Knupp and Margolin and Shashkov, JCP, 2002) determine the coordinate transformation such that its Jacobian matrix is as close as possible to a reference Jacobian matrix in the least squares sense. One of the functionals they use is

$$I(\xi) = \int_{\Omega_P} \left\| \frac{\partial \xi}{\partial x} - K \right\|_F^2 ds, \quad (3.28)$$

where $\| \cdot \|_F$ is the Frobenius norm of matrix and $K = K(x)$ is the user prescribed, reference Jacobian matrix. A detailed discussion on how to choose the matrix K is given in (Knupp 1996); see also (Knupp and Robidoux, SISC, 2000) for a broader discussion on algebraic properties of the Jacobian matrix.

The method of (Huang JCP 2001) given in (2.43) is based on mesh quality control of equidistribution alignment, and orientation. The choice of the monitor function \mathbf{M} in this method based on interpolation error estimates has been extensively studied in (Huang and Sun, JCP 2003 and Huang, SISC 2005). The idea of mesh quality control has also been used by Branets and Carey (IMR12, 2003) in developing their grid smoothing variational.

Example 3.2-1. This example is to generate adaptive meshes for a given weight function

$$w(x, y) = 1 + 10 \exp(-50(y - \frac{1}{2} - \frac{1}{4} \sin(2\pi x))^2), \quad \text{in } \Omega \equiv (0, 1) \times (0, 1).$$

The monitor function is chosen as $\mathbf{M} = wI$. Adaptive meshes are shown in Fig. 3.1.

Figure 3.1. Example 3.2-1. Adaptive moving meshes obtained by the harmonic mapping method, Winslow's method, and the method based on equidistribution and alignment control (2.43) ($\theta = 0.1$).

Example 3.2-2. This example is to generate adaptive moving meshes for a given weight function

$$w(x, y, t) = 1 + 10 \exp(-50 | (x - \frac{1}{2} - \frac{1}{4} \cos(2\pi t))^2 + (y - \frac{1}{2} - \frac{1}{4} \sin(2\pi t))^2 - (\frac{1}{10})^2 |).$$

The monitor function is chosen as $\mathbf{M} = wI$. Adaptive meshes obtained using MMPDEs based on three different functionals are shown in Fig. 3.2, 3.3, and 3.4.

Figure 3.2. Example 3.2-2. Adaptive moving meshes obtained by MMPDE based on the harmonic mapping method.

Figure 3.3. Example 3.2-2. Adaptive moving meshes obtained by MMPDE based on Winslow's method.

Figure 3.4. Example 3.2-2. Adaptive moving meshes obtained by MMPDE based on the method with equidistribution and alignment control (2.43) ($\theta = 0.1$).

Example 3.2-3 This example is to generate an adaptive mesh for for a given analytical solution

$$\begin{aligned} u(x, y) = & \tanh(30(x^2 + y^2 - \frac{1}{8})) + \tanh(30((x - 0.5)^2 + (x - 0.5)^2 - \frac{1}{8})) \\ & + \tanh(30((x - 0.5)^2 + (x + 0.5)^2 - \frac{1}{8})) + \tanh(30((x + 0.5)^2 + (x - 0.5)^2 - \frac{1}{8})) \\ & + \tanh(30((x + 0.5)^2 + (x + 0.5)^2 - \frac{1}{8})) \end{aligned}$$

defined in $[-2, 2] \times [-2, 2]$. An adaptive mesh is expected to concentrate around five circles. Results are shown in Fig. 3.5.

Example 3.2-4 This example is to generate 3D adaptive meshes for

$$u(x, y, z) = \tanh(100((x - 0.5)^2 + (y - 0.5)^2 + (z - 0.5)^2) - 0.0625)$$

defined in the unit cube. An adaptive mesh for this function is expected to

Figure 3.5. Example 3.2-3. Adaptive meshes of size 81×81 are obtained using the variational method (2.43) ($\theta = 0.1$) for different monitor functions based on isotropic and anisotropic interpolation error estimates.

concentrate near the sphere centered at $(0, 0, 0)$ with radius 0.25. Results are shown in Fig. 3.6.

Figure 3.6. Example 3.2-4. An Adaptive mesh of size $65 \times 65 \times 65$ obtained using the variational method (2.43) ($\theta = 0.1$) for a monitor function based on interpolation error. (a): Cutaway plot of the mesh. (b): Plane projection of slice at $K_z = 32$ of the mesh in (a).

3.3. Optimal transport methods

Derivation of the optimal transport equations

Optimal transport methods are a very natural generalisation of MMPDE methods in one-dimension, that retain much of the simplicity of the one-dimensional approach (such as always solving scalar equations and automatic calculation of the mesh on a boundary) whilst being general enough to deliver meshes of provable mesh quality (with many of the proofs following directly from the one-dimensional case). They have the disadvantage of being less flexible than some of the moving mesh methods described above. However, in practice they can give very regular meshes for a wide range of possible monitor functions. The key idea behind an optimal mesh is that it should be one which is closest to a uniform mesh in a suitable norm, consistent with satisfying the equidistribution principle. The simplest such norm is the least squares norm given by

$$I = \int_{\Omega_C} |\mathbf{F}(\xi, t) - \xi|^2 d\xi. \quad (3.29)$$

Minimising I subject to the equidistribution principle is in fact the celebrated Monge-Kantorovich problem from differential geometry. This principle is often called optimum transport as it leads to a transformation the creation of which takes a minimum amount of work as a deviation from the identity. Intuitively, this is likely to deliver a regular mesh, as this mesh will be as close (in an averaged sense) to the most regular possible mesh ie. a completely uniform one. remarkably this minimisation problem has a unique solution with a very elegant expression for the transformation.

Theorem 3. There *exists* a *unique* optimal mapping $\mathbf{F}(\xi, t)$ satisfying the equidistribution equation. This map has the same regularity as M . Furthermore $\mathbf{F}(\xi, t)$ is the unique mapping from this class which can be written

as the gradient (with respect to ξ) of a *convex* (mesh) potential $P(\xi, t)$, so that

$$\mathbf{F}(\xi, t) = \nabla_{\xi} P(\xi, t), \quad \Delta_{\xi} P(\xi, t) > 0. \quad (3.30)$$

Proof. See ? for an abstract proof and Delzanno et al. (2008) for a proof in the context of adaptive mesh generation. \square

The following is then immediate

Lemma 6. The map \mathbf{F} is *irrotational* so that $\nabla_{\xi} \times \mathbf{F} = \mathbf{0}$, and the Jacobian of \mathbf{F} is *symmetric*.

Significantly, the transformation above is an example of a *Legendre Transformation* (?). Such transformations include translations and linear maps by positive definite symmetric matrices. We now show how to calculate such a transformation.

Properties of optimally transported meshes

It is immediate that if $\mathbf{X} = \nabla_{\xi} P$ then

$$\frac{\partial \mathbf{X}}{\partial \xi} = H(P)$$

where $H(P)$ is the *Hessian* of P . Additionally, if the measure $M \in W^2(\Omega_P)$ is strictly positive on its supports (assumed to be convex) then the potential $P \in W_{loc}^2(\Omega_C)$ and satisfies, in the classical sense, the Monge-Ampère equation

$$M(\nabla_{\xi} P, t) H(P) = \theta(t). \quad (3.31)$$

Here $H(P)$ denotes the determinant of the Hessian matrix of P . This is a famous equation in differential geometry. Solving it defines the map \mathbf{F} *uniquely*. In order to solve it we must specify boundary conditions for the solution. In general applications of mesh generation we consider bounded domains mapping to bounded domains. We can then prescribe a boundary condition where equation (3.31) is supplemented with the condition that the boundary must map to the boundary. Suppose that the points on the boundary of Ω_C satisfy the implicit equation $G_C(\xi) = 0$ and those on the boundary of Ω_P satisfy the implicit equation $G_P(\mathbf{X}) = 0$ we then have the following (nonlinear) Neumann boundary condition for (3.31)

$$G_P(\nabla_{\xi} P, t) = 0 \quad \text{if} \quad G_C(\xi, t) = 0. \quad (3.32)$$

The existence, uniqueness and regularity of the solutions of (3.31,3.32) has been well studied (Brenier 1991), (Cafferelli 1992), (Cafferelli 1996). From the point of view of grid generation this puts us into the nice situation of

being able to infer properties of the mesh from those of a function P with known regularity. We have the following very important result

Theorem 4. If both Ω_C and Ω_P are *smooth, convex* domains then (3.31,3.32) has a unique solution (up to an additive constant) which is as regular as the monitor function M . This in turn leads to a unique regular mesh.

Proof. This follows immediately from the results of Brenier (Brenier 1991) and Caffarelli (Cafferelli 1992),(Cafferelli 1996). \square

If Ω_C and/or Ω_P are not smooth then there is a possible loss of mesh regularity. In the case of solutions which are logical rectangles, this is not severe and applies only at the corners, where it can be shown (Cullen 1989) that $P \in C^3$ and $F \in C^2$. This is sufficient regularity for most applications.

Boundary behavior

An immediate consequence of Theorem 4.3 is that the solution of the Monge-Ampère equation not only defines a map from Ω_C to Ω_P but also from the boundaries of the respective domains. This is a desirable property from the perspective of mesh generation as we do not have to consider separate equations for the mesh on the boundary. This is in contrast to the variational method in (Huang and Russell 1999a) described above which require separate equations to describe the mesh on the boundaries. As an example, we can consider meshes on logical cubes, taking

$$\Omega_C = [0, 1]^d = \Omega_P. \quad (3.33)$$

The boundary conditions above then reduce to

$$P_\xi = 0, 1 \quad \text{if} \quad \xi = 0, 1, \quad P_\eta = 0, 1 \quad \text{if} \quad \eta = 0, 1. \quad (3.34)$$

The boundary condition, (3.34) applied in two-dimensions implies the orthogonality of the grid lines at the boundaries of the square. To see this, consider the bottom boundary of the square in the physical domain, for which $Y = 0$. A grid line Γ which intersects this side is given by

$$\Gamma = \{(X, Y) : 0 \leq \eta \leq 1, \quad \xi \text{ fixed}\}.$$

The tangent to this line in the physical domain is given by $\tau = (X_\eta, Y_\eta)^T$, and, trivially, the tangent to the bottom boundary of the square by the vector $\mathbf{t} = (1, 0)^T$. However, on the bottom boundary, the function P satisfies the identity $P_\eta(\xi, 0) = 0$ and by definition $X = P_\xi$. It follows immediately that when $\eta = 0$,

$$X_\eta(\xi, 0) = P_{\eta\xi}(\xi, 0) = 0,$$

so that on the lower boundary $\tau \cdot \mathbf{t} = 0$. Thus the mesh is orthogonal to the lower boundary. Similar results apply to the other sides as well. The condition of mesh orthogonality at the boundary of the square is often desirable in certain circumstances (Thompson, Warsi and Mastin 1985). However it may cause problems when resolving features, such as fronts, which intersect boundaries at an angle. In such cases it may be useful to refine the mesh close to the boundary, either by introducing a finer computational mesh there, or by locally increasing the value of the monitor function M at mesh points adjacent to the boundary.

Mesh symmetries

Some very desirable properties of the mesh follow immediately from the properties of the Monge-Ampère equation. It is trivial to see that the equation is invariant under translations in (ξ, η) . It is also easy to see that (in a similar manner to the Laplacian operator) the Monge-Ampère equation is also invariant under any *orthogonal* map such as a rotation or a reflexion. This is because the Monge-Ampère equation is the determinant of the Hessian which transforms covariantly under such maps. This simple observation implies that (away from boundaries) the meshes generated by solving the Monge-Ampère equation should have no difficulty aligning themselves to structures, such as shocks, which may occur anywhere in a domain and at any orientation. We will see evidence for this in the examples at the end of this paper. (In this sense Monge-Ampère generated meshes should perform better than meshes constrained by the underlying coordinate axes.) It is also immediate that Monge-Ampère equation is also invariant under *scaling transformations* of the form $\xi \rightarrow \lambda\xi$, $\eta \rightarrow \mu\eta$, $P \rightarrow \nu P$ provided that M is chosen carefully.

Mesh skewness

The regularity of the mesh generated by this approach gives a useful feature in seeing control in the variation of the element size across the domain. However this does not immediately indicate how appropriate the mesh will be for calculations, such as in the finite element method, where we might seek to avoid long, thin regions. However, it is possible to make some estimates for the resulting skewness of the mesh in terms of the properties of the function P . Consider a two-dimensional problem for which the map from Ω_C to Ω_P has the Jacobian J . A measure for the skewness s of the mesh in Ω_P is given by

$$s = \frac{\lambda_1}{\lambda_2} + \frac{\lambda_2}{\lambda_1} s = \frac{(\lambda_1 + \lambda_2)^2}{\lambda_1 \lambda_2} - 2 = \frac{\text{trace}(J)}{\det(J)} - 2.$$

where λ_1 and λ_2 are the (real and positive) eigenvalues of J . As J is the Hessian of P it follows immediately that

$$s = \frac{\Delta(P)^2}{H(P)} - 2. \quad (3.35)$$

The value of s could, in principle, be controlled directly by including (3.35) directly into the monitor function. This is similar to the approach used by Huang (Huang 2005a),(Huang 2005b) in which he took a monitor function which controlled the mesh alignment. Alternatively, the skewness can be estimated in certain cases. One example of this arises in the scale-invariant meshes for the local singularities blow-up problems studied in (Budd and Williams 2006), in which a sequence of meshes are calculated which close to the singularity take the form $P(\xi, t) = \Lambda(t)\hat{P}(\xi)$ for an appropriate scaling function $\Lambda(t)$. It is immediate that

$$\frac{\Delta(P)^2}{H(P)} - 2 = \frac{\Delta(\hat{P})^2}{H(\hat{P})} - 2$$

so that the skewness of the rescaled mesh is the same as the original. Hence, if an initially uniform mesh is used then the mesh close to the singularity will retain local uniformity. In general the meshes generated by the Monge-Ampère equation have good regularity properties and are effective in interpolating functions. An analysis of the regularity of these meshes is given in (Delzanno et al. 2008)

noindent INTERPOLATION AND SKEWNESS HERE .. USING FINN TO GUIDE THE PRESENTATION.

Solution of the Monge-Ampère equation

The equation (3.31) can be solved either directly (Delzanno et al. 2008), (?) or by a relaxation method.

The direct method

The Monge-Ampère equation (Evans 1999, Gutiérrez 2001, ?) belongs to the class of fully nonlinear second-order equations and has two sources of nonlinearity. Firstly, the Hessian $H(P)$ is nonlinear in the second derivatives (except in the one-dimensional case). Secondly, the monitor function in general depends nonlinearly on the first derivatives of P (either directly or through the solution of the original PDE). For any suitably smooth positive monitor function the equation has a unique solution, which is a convex function. Linearisation of the equation shows that it is elliptic in the space of convex functions. The Monge-Ampère equation arises from prescribing the product of the eigenvalues (the determinant) of the Jacobian matrix of a gradient mapping. If we prescribe the sum of the eigenvalues (the trace) then we obtain a standard Poisson equation. For the Poisson equation,

multigrid methods can find the solution of a discretisation on a grid with $O(N)$ unknowns using $O(N)$ operations. Methods for solving the Monge-Ampère aim to obtain the same computational complexity.

In (Oliker and Prussner 1988) a specially designed discretisation and iterative method are proposed which explicitly preserve the convexity of the iterates. In (Benamou and Brenier 2000) the Monge-Ampère equation is transformed into a time-dependent fluid mechanics problem which is solved using an iterative method based on an augmented Lagrangian approach. In (Dean and Glowinski 2003) and (Dean and Glowinski 2004) finite element discretisations are proposed based on an augmented Lagrangian approach and a least squares formulation respectively. In (Feng and Neilan 2008) the nonlinear second-order equation is considered as the limiting equation of a singularly perturbed fourth-order quasilinear equation. In (Chartrand, Vixie, Wohlberg and Bollt 2007) it is shown that the Monge-Ampère equation can be reformulated as an unconstrained optimisation problem, which can be solved by a gradient descent method. A special property of the mappings generated by the Monge-Ampère is that they are irrotational, i.e., the curl is zero. In (Haker and Tannenbaum 2003) a gradient descent method is proposed that uses a Poisson solve in each step to “remove the curl”. The nonlinear multigrid method developed in (Fulton 1989) for the semi-geostrophic equation should be easily adapted to our problem. A Newton-Krylov-multigrid method is proposed in (Delzanno et al. 2008).

The above methods all try to solve the fully nonlinear Monge-Ampère equation directly. It was a remarkable achievement of Kantorovich to show that the problem can be relaxed to a linear one by not considering a transport map $\xi \rightarrow x = F(\xi)$, but a transport plan $G(\xi, x)$ indicating the amount of material to be transported from ξ to x (Rachev and Rüschendorf 1998, Evans 1999, ?). Robust methods exist for solving the corresponding linear programming problem, but to the best of our knowledge these methods typically require $O(N^2)$ operations (Kaijser 1998, Balinski 1986), which is unacceptable except for small problems.

The Parabolic Monge-Ampère method (PMA)

An alternative approach motivated by the discussion of the MMPDEs given earlier, is to introduce a parabolic regularization to (3.31) so that the gradient of solutions of this evolve toward the gradient of the solutions of (3.31) over a (relatively) short timescale. This method also couples naturally to the solution of a time dependent PDE. Accordingly we consider using *relaxation* to generate an approximate solution of (3.31) which evolves together with the solution of the underlying PDE. Accordingly we consider a time evolving function $Q(\xi, t)$ with associated mesh $\mathbf{X}(\xi, t) = \nabla_{\xi} Q(\xi, t)$ with the property that this mesh should be close to that determined by the solution of the Monge-Ampère equation. To do this we consider a relaxed form of

(3.31) taking the form of a Parabolic Monge-Ampère equation (PMA) of the form

$$\varepsilon(I - \gamma\Delta)Q_t = (H(Q)M(\nabla Q))^{1/d}. \quad (3.36)$$

To find a moving mesh, we start with an initially uniform mesh for which

$$Q(\xi, 0) = \frac{1}{2}|\xi|^2.$$

The function Q then evolves according to (3.36). In (3.36) the scaling power $1/d$ is necessary for global existence of the solution. This is because if Q is scaled by a factor $L(t)$ then the Hessian term $H(Q)$ scales as $L(t)^d$. If M is constant, then equation (3.36) without the power law scaling admits a variables separable solution for which $L_t = CL^d$. If $d > 1$ then this equation has solutions which blow-up in a finite time. The rescaling prevents this possibility. The operator on the left of this system is a smoothing operator, similar to the operator used by Cenicerros and Hou in (Cenicerros and Hou 2001), which reduces the stiffness of this system when it is discretised. Observe that the term $\theta(t)$ has not been included in (3.36). Indeed this term arises naturally as a constant of integration.

The PMA equation (3.36) has many properties in common with the moving mesh equation (3.9). In particular if M is independent of time then the solution of (3.31) corresponds to a stable solution of (3.36). Indeed we have the following results

Lemma 7. (a) Suppose that $M_t = 0$ and the Monge-Ampère equation (3.31) admits a (steady) convex solution $P(\xi)$ with associated map $\mathbf{X}(\xi)$ for which $H(P) > 0$, so that P satisfies the equation

$$M(\nabla P)H(P) = \theta$$

then

(i) the PMA equation (3.36) admits a time dependent solution

$$Q(\xi, t) = \frac{\theta^{1/d}t}{\varepsilon} + P(\xi) \quad (3.37)$$

for which $\nabla Q = \nabla P = \mathbf{X}(\xi)$.

(ii) The resulting mesh is locally stable.

(b) If M is slowly varying, then the solution of (3.36) remains ε close to a solution of (3.31) for all time.

Proof. This is given in Budd and Williams (2008) and is very similar to the corresponding proof given for the stability of (3.9) \square

It is also possible to show Budd and Williams (2008) that throughout the evolution of the mesh, if $H(Q)$ and ΔQ are initially positive then they stay positive for all time. This application of the maximum principle guarantees

that the map generating the mesh is locally invertible for all time, and hence no mesh tangling can occur. This is a very useful feature of such r-adaptive methods and we will see numerical examples of this in the following calculations.

Discretising the Parabolic Monge-Ampère Equation (3.36).

To integrate (3.36) in time together with the Neumann boundary conditions (3.34) we can use a semi-discrete method, taking a finite difference/spectral method in the computational space to lead to a set of ordinary differential equations (in time), and then solving these equations using a *fully explicit* time-stepping method such as `ode45` or any other appropriate ODE solver. To discretise (3.36) in space we can impose a uniform grid of mesh size $\Delta\xi$ on the computational space and assume that Q and Q_t take point values $Q_{i,j}(t)$, $i, j = 0 \dots N$ etc. on this grid, with indices 0 and N corresponding to mesh points on the boundary. To discretise the Hessian operator $H(Q)$ we can use a nine point stencil, and interior to the domain Ω_C , use central differencing to evaluate all second derivatives in $H(Q)$. For the boundary points, the central differences are replaced by Taylor series expansions, using the respective conditions $Q_\xi = 0$ or $Q_\xi = 1$ etc. The gradient ∇Q is calculated similarly, so that the right hand side of (3.36) can be determined. To determine the values of Q_t at the mesh points we then invert the operator $(I - \gamma\Delta_\xi)$. As the system is posed on a uniform (rectangular) mesh in the computational domain, this inversion was done very rapidly by using a fast spectral solver based upon the discrete cosine transform (invoking the Neumann boundary conditions for Q_t). Applying the various procedures above to generate a moving mesh for which the function $M(\mathbf{X}, t) = M(\nabla Q, t)$ is known explicitly leads to a system system of ordinary differential equations for the evolution of the mesh potential Q and hence of the mesh itself. We solve these using an explicit method (typically the MATLAB routine `ode23` or `ode45`), as the strongly nonlinear nature of the equations makes implicit time-stepping too expensive in general. We apply this in two stages.

- 1 Starting from an initially uniform mesh in the physical space, we adapt this to equidistribute the monitor function *at the initial time* over Ω_P . To do this we set $M_0(\mathbf{X}) \equiv M(\mathbf{X}, 0)$, $Q(\xi, 0) = (\xi^2/2 + \eta^2/2)$ and solve (3.36) with M fixed to equal the function M_0 , with $\varepsilon = 1$ for $0 < t < T$, where T is a fixed time. In this first calculation t is an *artificial time* during which the uniform mesh evolves toward an equidistributed mesh. It follows from the earlier results that such a mesh exists and is stable. The relatively large value of ε prevents this calculation from being unnecessarily stiff, and T is chosen large enough to allow the mesh to relax toward the equidistributed state.
- 2 We then solve (3.36) with the time dependent monitor function $M(\mathbf{X}, t)$

Figure 3.7. Example 1. Snapshots at $t = 0, 1/4$ in the top row and $t = 1/2, 10$ in the bottom. As the solution is advected about the grid follows the maxima of the solution and does not ever fall behind or become distorted.

with t now *actual time*. For this calculation we now set $\varepsilon = 0.01$. It follows from the results above that the mesh resulting from taking the gradient of the solution of the Parabolic Monge-Ampère problem will, as time evolves, be ε -close to a mesh which exactly equidistributes $M(\mathbf{X}, t)$, provided that M does not change too rapidly with time). As this procedure starts from a mesh which exactly equidistributes $M(\mathbf{X}, 0)$ the resulting differential equations are not especially stiff.

Examples of meshes generated using the PMA method

We consider some meshes generated using the PMA method, choosing examples which can be compared with other methods presented both in the literature and in other sections of this article. All examples have been run in Matlab on a desktop computer in under five minutes.

Example 1 We first take a case motivated by (Cao, Huang, and Russell 2002) (see also Chapter 4) with a monitor function localised over a moving circle of the form

$$M_1(x, y, t) = 1 + 5 \exp(-50|(x - 1/2 - 1/4 \cos(2\pi t))^2 + (y - 1/2 - \sin(2\pi t)/2)^2 - 0.01|).$$

This is described in (Cao, Huang, and Russell 2002) as a severe test of a moving mesh method, and the meshes calculated from this using the (velocity based) Geometric Conservation Law method (Cao, Huang, and Russell 2002) have a high degree of skewness. To compute a corresponding moving mesh using the Parabolic Monge-Ampère method (and also the solutions in all of the examples in this subsection) we use a uniform computational mesh to solve (3.36) with $N = 30$ mesh points in each direction and mapping the unit square to the unit square. We see from this calculation that the resulting mesh closely follows the moving circle with now evidence of skewness or irregularity. Note the orthogonality and regularity of the mesh at the boundary of the domain. Notice further that in Figure 3.7 the solution partially exits the domain with no ill consequence and that the grid at $t = 10$ is virtually indistinguishable from that at $t = 0$. This is to be contrasted with a similar figure in (Cao, Huang, and Russell 2002) in which the grid is significantly distorted after the circle has passed out of the domain. It is clear from these figures that the mesh generated has excellent regularity. Observe the high degree of mesh uniformity close to the solution maximum.

Example 2 In an example taken directly from (Cao, Huang, and Russell

Figure 3.8. Example 2. Snapshots at $t = 0, 1/3$ in the top row and $t = 2/3, 1$ in the bottom. In this figure we see both the smoothness and the periodic (in time) form of the mesh.

2002) we consider a monitor function localised on a sine-wave of the form

$$M_2(x, y, t) = 1 + 5 \exp(-50|(y - 1/2 - 1/4 \sin(2\pi x) \sin(2\pi t))|).$$

and proceed to use exactly the same method as described in Example 1. In this example we again see that the PMA method has generated a very regular and periodic (in time) mesh. This is again in stark contrast to the GCL method which resulted in non-smooth and non-periodic meshes.

Example 3 In this example we consider a monitor function localised on the support of two travelling linear fronts moving toward each other and show the resulting mesh in Figure 3.9. For this we take

$$\begin{aligned} x_0 &= t, & y_0 &= 0.2 + t/2 \\ u_0 &= \gamma \operatorname{sech}(\lambda(x - x_0 + y - y_0)) \\ x_1 &= 1 - t, & y_1 &= 0.8 - t/2 \\ u_1 &= \gamma \operatorname{sech}(\lambda(x - x_1 + y - y_1)) \end{aligned}$$

$$M_3(x, y, t) = 1 + u_0 + u_1$$

with $\gamma = 5$ and $\lambda = 100$. We note that the fronts pass through each other without generating spurious oscillations in the mesh. Note also that the solution points do not follow the front (moving in and out of the region of high mesh density), but that the *density* of the solution points does. Observe that the mesh automatically aligns itself along the front. A close-up of the mesh close to the front is given in Figure 3.10. This shows very good resolution of the local structure close to the front and a smooth transition from a uniform mesh to one refined at the front. If γ is large then close to the front we see a mesh compression proportional to γ orthogonal to the front, with a local skewness of $s = \gamma$. The PMA method in this case thus generates a smooth mesh, for which the degree of skewness can be controlled via the choice of γ . More details of this calculation are given in (Walsh, Budd and Williams 2009). We can also see the effects of mesh orthogonality close to the boundaries of the domain. As mentioned earlier, it is in principle possible to reduce the loss of resolution here by increasing the value of M close to the boundary, although we have not done this here. We see therefore from this example that

Example 4 In this example we look at a monitor function localised on the support of two fronts meeting at an angle. The behaviour of the mesh

Figure 3.9. Example 3. Snapshots at $t = 0, 0.495$ in the top row and $t = 0.525, 0.75$ in the bottom. Here we have simulated two fronts passing through each other in parallel, and see no difficulties with the resulting mesh. Note the way that the mesh automatically aligns itself parallel to the front.

Figure 3.10. Example 3. A close-up of the mesh close to the front, showing the transition from a uniform mesh to one compressed by a factor γ orthogonal to the front.

close to the front is similar to that in the last example. Significantly there is no mesh tangling as the two fronts intersect. In Figure 3.3 we show a close-up of the intersection region showing the high degree of mesh regularity there.

$$\begin{aligned} x_0 &= t, & y_0 &= 0.2 + t/2 \\ u_0 &= \gamma \operatorname{sech}(\lambda(x - x_0 + y - y_0)) \\ x_1 &= 1 - t, & y_1 &= 0.8 - t \\ u_1 &= \gamma \operatorname{sech}(\lambda(x - x_1 + (y - y_1)/2)) \\ M_4(x, y, t) &= 1 + u_0 + u_1 \end{aligned}$$

Further examples in which we couple the PMA algorithm to generate moving meshes for certain partial differential equations are presented in Chapter 5.

3.4. Adaptive discretisation of PDEs in higher dimensions

We now extend the discussion earlier in this section in which we looked at coupling an MMPDE to a one-dimensional PDE to look at the rather harder problem of coupling a moving mesh method (derived either from a variational approach or from a Monge-Ampère based approach) to a partial differential equation in several spatial dimensions. Obviously in this case the errors in using a non-uniform mesh are more pronounced, and the additional convective terms introduced by the moving mesh are potentially

Figure 3.11. Example 4. Snapshots at $t = 0, 0.2$ in the top row and $t = 0.525, 1.4$ in the bottom. Here we have simulated two linear travelling fronts passing through each other at an angle and see no difficulties in generating and moving the mesh.

Figure 3.12. Example 4. A close-up of the region where the two fronts intersect.

destabilising. A moving mesh method becomes an adaptive method when it is coupled to a discretisation of a partial differential equation. There is no unique way to perform this coupling, and it depends upon whether the PDE is discretised in the *computational domain* (typically with a finite difference method) or in the physical domain (typically with a finite element or a finite volume method). The nature of the coupling also depends upon whether or not the adaptive method is going to be coupled to existing software (such as a standard CFD method) The former usually involves some form of interpolation of the solution to map it onto a mesh suitable for the existing solver to use. The coupling of the mesh to the underlying PDE should also preserve important structures of the PDE, in particular any conservation laws or solution symmetries should ideally be preserved in the coupled system. In general, the function M is not known explicitly, but is a given function of the solution of the underlying PDE which we assume has the form.

$$u_t = f(t, \mathbf{X}, u, \nabla_X u, \Delta_X u) \quad (3.38)$$

This equation is coupled with the moving mesh equations to evolve the solution and mesh together. This can be done in various ways, see (Tang 2005), (?) for reviews of these. The *Quasi-Lagrangian* approaches (which avoid interpolation) allow directly for the mesh movement and express the PDE in Lagrangian variables which generalises the expression (3.16).

$$\dot{u} = f(t, \mathbf{X}, u, \nabla_X u, \Delta_X u) + \nabla_X u \cdot \dot{\mathbf{X}} \quad (3.39)$$

where the \dot{u} , $\dot{\mathbf{X}}$ indicates differentiation with respect to time with the computational variable ξ fixed. The two equations (3.36) and (3.39) can then both be discretised on the *computational mesh* and solved *simultaneously*. As described earlier, this procedure is generally the method of choice when used in calculations in one spatial dimension (Huang, Ren and Russell 1994). However it is much harder to use in higher dimensions as the coupled system can be rather stiff and the equations are very nonlinear. This method has the advantages that there is no need to interpolate a solution from one mesh to the next as the mesh evolves and also that the mesh can inherit useful dynamical properties of the solution such as scaling structures (Budd, Huang and Russell 1996), (Budd and Williams 2006), (Baines et. al. 2006). Alternatively the two equations (3.36) and (3.39) can be solved *alternately* (?), (?), (Ceniceros and Hou 2001). This reduces the stiffness problems but can lead to a lag in the mesh movement.

Alternatively, a *rezoning* method can be used (Tang 2005). In such methods

equations similar to (3.36) are solved to advance the mesh by one time step. The current solution u is then interpolated onto this new mesh in the physical domain, and the original PDE (3.38) solved on the new mesh in this domain (often using a finite element or finite volume method). The advantage of this method is that standard software can be used to solve the PDE, but at the expense of an interpolation step.

We now describe these methods in more detail.

Simultaneous solution in the computational domain

We now briefly detail the implementation of the *simultaneous solution method* solving the mesh equation (3.36) and a relatively simple parabolic underlying PDE (3.39) by using finite differences in the *computational* domain. To do this we associate each mesh point in the fixed computational mesh with a solution point $U_{i,j}(t)$. To discretise (3.39) we transform the derivatives in the physical domain to ones in the computational domain. For example, in two dimensions, if we have general transformation F with Jacobian J then (see (Ceniceros and Hou 2001) for details) the derivatives of u can be expressed in terms of the computational variables in the following manner

$$\begin{aligned} u_x &= \frac{1}{J} ((y_\eta u)_\xi - (y_\xi u)_\eta), \\ u_y &= \frac{1}{J} (-(x_\eta u)_\xi + (x_\xi u)_\eta), \\ u_{xx} &= \frac{1}{J} \left((J^{-1} y_\eta^2 u_\xi)_\xi - (J^{-1} y_\eta y_\xi u_\eta)_\xi - (J^{-1} y_\eta y_\xi u_\xi)_\eta + (J^{-1} y_\xi y_\xi u_\eta)_\eta \right), \\ u_{yy} &= \frac{1}{J} \left((J^{-1} x_\eta^2 u_\xi)_\xi - (J^{-1} x_\eta x_\xi u_\eta)_\xi - (J^{-1} x_\eta x_\xi u_\xi)_\eta + (J^{-1} x_\xi x_\xi u_\eta)_\eta \right). \end{aligned} \tag{3.40}$$

As the computational domain has a uniform mesh, and the Jacobian J may be determined directly from the mesh mapping F , it follows that (3.40) can be discretised to high accuracy in space using a standard finite-difference or finite-element discretisation. The solution u is then determined by solving (3.36) and (3.39) together using an appropriate ODE solver such as SDIRK or a predictor-corrector method. Examples of the use of this method are given in Chapter 5.

Alternating solution in the computational domain

The alternating solution method is often used in higher dimensional calculations to alternatively solve the system and to move the mesh. This method avoids the highly nonlinear coupling of the mesh and the physical solution and preserves many structures such as ellipticity and sparsity in each of the mesh and physical PDEs. This can lead to significant efficiency gains, but at the disadvantage of a possible lag in the movement of the mesh and possible mesh instabilities.

Suppose that the physical solution u^n the mesh X^n and a solution time-step Δt^n are known at a time t^n . The alternating solution procedure is typically implemented as follows.

- 1 The monitor function $M^n(x) = M(t^n, u^n, x^n)$ is calculated using u^n and x^n .
- 2 The appropriate MMPDE is discretised in space and then integrated over the time $[t^n, t^n + \Delta t^n]$ to give a new mesh X^{n+1} at the time $t^n + \Delta t^n$. The underlying solution u^n is not changed during this calculation. This calculation can be done by using using the (fixed) monitor function M^n as in Ceniceros and Hou (2001), or by updating it during the integration by using linear interpolation (Huang 2007).
- 3 The physical PDE (3.39) is then discretised in (computational) space and integrated using a SDIRK or similar method. In this calculation, the convective terms involving X_t use the approximation

$$\dot{X} = \frac{X^{n+1} - X^n}{\Delta t^n}.$$

The mesh used in the discretisation is calculated by using linear interpolation so that

$$X(t) = X^n + (t - t^n)\dot{X}.$$

- 4 It may be necessary to use the new solution u^{n+1} to update the monitor function M^n and to iterate from 2 above in order to gain better control of the grid. In this case repeat these steps until the new mesh does not change.
- 5 This procedure is then repeated from 1 above.

In Ceniceros and Hou (2001) this method together with the higher dimensional form of the MMPDE (3.9) is used (without updating the monitor function between time-steps) to solve problems involving vortex singularities in the Bousinessq equations.

Rezoning in the physical domain

There are various problems associated with solving the Lagrangian form of the PDE (3.39) in the computational domain. A significant one of these is that certain properties of the original PDE may be lost when it is put into the Lagrangian form and coupled to a moving mesh equation. Two examples of this occurring are the loss of a conservation form of the equation (for example when adaptive methods are used to solve the Euler equations) or the loss of a Hamiltonian structure when solving problems such as the KdV or the NLS equations. These problems can be avoided by completely decoupling the solution of the PDE and the moving mesh equations over each time step, so that when solving the PDE the mesh is regarded as being

static. The PDE can then be solved over that time step using a method (for example the finite volume method for problems with a conservation law) that preserves the significant features of the solution. The rezoning method can be briefly described as follows

- At time step t^n let the solution be u^n and the mesh x^n . Calculate the corresponding monitor function M^n .
- Using this monitor function solve the moving mesh PDE over the time interval $[t^n, t^n + \Delta t^n]$ to give a new mesh X^{n+1} .
- *Interpolate* the solution u^n onto the *new mesh* X^{n+1} , to give an interpolated solution \hat{u}^n .
- If necessary, repeat steps 2 and 3, updating M until the new mesh does not change.
- Starting from \hat{u}^n solve the original PDE (3.38). Typically in the *physical domain* using high-resolution standard software such as the Finite Volume method) over the time interval $[t^n, t^n + \Delta t^n]$ doing all the calculations on the new mesh X^{n+1} .
- Repeat this from 1.

This method is of course closely related to the Quasi-Lagrangian approach. To see this in a one-dimensional example, suppose that the mesh velocity is X_t and the underlying solution is u with $u_i^n \approx u(X_i^n)$, then in step 3 we have

$$\hat{u}_i^n \approx u(X_i^{n+1}) = u(X_i^n + \delta t_n X_t) \approx u^n + \delta t_n u_x X_t.$$

Thus if we use a simple Forward Euler method to approximate the solution of (3.38) we obtain

$$u_i^{n+1} = \hat{u}_i^n + \delta t_n f_i = u^n + \delta t_n u_x X_t + \delta t_n f_i = u_n + \delta t_n (f_i + u_x X_t)$$

which is of course the result of applying the Forward Euler method to the Lagrangian form of the PDE given by (3.39).

The key to the success of this approach is the interpolation step 3. and an interpolation scheme that preserves some quantities of the solution is often necessary. We summarise some rezoning methods based on this approach for solving conservation laws in one and two dimensions using the Finite Volume method, and which are described in (Tang 2005), (Tang and Tang 2003).

One-dimensional calculation

Suppose that the i th new grid point at time t^{n+1} is X_i^{n+1} . In a Finite Volume method the key quantities are the *cell averages* given by

$$u_{j+1/2} = \frac{1}{X_{j+1} - X_j} \int_{X_j}^{X_{j+1}} u \, dx, \quad X_{j+1/2} = \frac{1}{2}(X_j + X_{j+1}).$$

If these are known on the mesh X^n then they can be interpolated onto the new mesh X^{n+1} . Suppose that $X_{j+1/2}^{n+1} \in [X_{k-1/2}^n, X_{k+1/2}^n]$ then naively this can be done via the formula

$$\hat{u}_{j+1/2}^n = u_{k+1/2}^n + \frac{u_{k+1/2}^n - u_{k-1/2}^n}{X_{k+1/2}^n - X_{k-1/2}^n} (X_{j+1/2}^{n+1} - X_{k+1/2}^n).$$

Unfortunately, this simple linear interpolation *does not conserve discrete solution mass* in the sense that

$$\sum \hat{u}^n_{j+1/2} (X_{j+1}^{n+1} - X_j^{n+1}) \neq \sum u^n_{j+1/2} (X_{j+1}^n - X_j^n)$$

and consequently leads to unsatisfactory results (Tang and Tang 2003) when used to solve hyperbolic conservation laws. An improved method in Tang and Tang (2003) mimics in part the discussion above for the relation between the Quasi-Lagrangian method and the rezoning method. Suppose that $X^{n+1} = X^n + \delta t \dot{X}$ then it follows from an application of Reynold's transport theorem, that to leading order

$$\begin{aligned} (X_{j+1}^{n+1} - X_j^{n+1}) \hat{u}_{j+1/2}^n &\approx \int_{X_j^{n+1}}^{X_{j+1}^{n+1}} \hat{u}^n dx \\ &\approx (X_{j+1}^n - X_j^n) u_{j+1/2}^n + \Delta t ((\dot{X} u^n)_{j+1} - (\dot{X} u^n)_j). \end{aligned}$$

This prompts the use of the (to leading order) conservative interpolation formula given by

$$(X_{j+1}^{n+1} - X_j^{n+1}) \hat{u}_{j+1/2}^n = (X_{j+1}^n - X_j^n) u_{j+1/2}^n + \Delta t ((\dot{X} u^n)_{j+1} - (\dot{X} u^n)_j). \quad (3.41)$$

which automatically conserves discrete mass. In Tang and Tang (2003) this method is used with success to solve conservation laws of the form

$$u_t + f(u)_x = 0$$

with the PDE being integrated using a MUSCL method (?) and using **MMPDE5** (3.7) to advance the mesh.

Two-dimensional calculations

In two-dimensions, Tang and Tang (2003) uses the moving mesh method described in Cenicerros and Hou (2001) to advance the mesh (X, Y) from time t^n to time t^{n+1} using the equations in ((?)). In this calculation we suppose that

$$(X^{n+1}, Y^{n+1}) = (X^n, Y^n) + \Delta t (\dot{X}, \dot{Y}).$$

The cell averages at time t^n over a (time evolving mesh cell) of area $A_{j+1/2, k+1/2}^n$ are now given by $u_{j+1/2, k+1/2}^n$ etc. and the normal mesh speed relative to

the surfaces of the mesh cell is given by

$$V = \dot{X}n_x + \dot{Y}n_y \quad \text{where the unit normal is given by } (n_x, n_y).$$

The conservative interpolation scheme proposed is

$$A_{j+1/2,k+1/2}^{n+1} \hat{u}_{j+1/2,k+1/2}^n = A_{j+1/2,k+1/2}^n u_{j+1/2,k+1/2}^n + \Delta t [(Vu^n)_{j+1,k+1/2} + (Vu^n)_{j,k+1/2}] + [(Vu^n)_{j+1/2,k+1} + (Vu^n)_{j+1/2,k}] \quad (3.42)$$

This scheme preserves discrete mass to leading order. Again a MUSCL method can be used to advance the solution of the PDE. This method is then used to solve the double-Mach reflection problem and various other problems with contact discontinuities arising in the solution of the Euler equations.

4. Velocity based moving mesh methods

In this chapter we will look in some more detail at *velocity based methods* for moving meshes. These are also called *Lagrangian methods* and they rely on calculating the *mesh point velocities* and from this the mesh point locations. In some ways these methods are very natural, as in (say) fluid mechanics calculations, natural solution features are often convected with the flow and it is natural to evolve the mesh points to follow the flow itself. (Note the huge popularity of the semi-Lagrangian and the characteristic Galerkin methods). However, velocity based methods have severe problems in implementation. These include significant mesh tangling, with associated skewness and also a tendency to create meshes which lag behind the solution. Indeed it is very possible for such meshes to have unstable movement, to move well away from equidistributed solutions and to lead to permanently distorted skewed and even frozen meshes, even after the significant solution structures have long gone. Some examples of this type of behaviour will be presented in our discussion of the GCL method. For these reasons the overall performance of these methods is, in general, not as good as that of the position based methods described in the last chapter and hence we will spend less time discussing them. We now describe three velocity based methods: the moving finite element method MFE, the geometric conservation law method GCL and the Deformation Map method.

4.1. Moving finite element (MFE) methods

The MFE method was originally developed by Miller and Miller (1981), Miller (1981) and represent a very important class of velocity based moving mesh methods with much underlying theory and many applications. See for example (Adjerid and Flaherty 1986), (Baines et. al. 2005), (Beckett, Mackenzie and Robertson 2001), (Cao, Huang and Russell 1999), (Carlson and Miller 1998a), (Carlson and Miller 1998b), (Di, Li, Tang and Zhang 2005), (Lang, Cao, Huang and Russell 2003), (Li, Tang and Zhang 2002), (Wathen and Baines 1985). A very complete survey of these and related methods is given in Baines (1994) and we will only describe them briefly here. The MFE method determines a mesh velocity $\dot{\mathbf{x}} = \mathbf{v}$ through a variational principle coupled to the solution of a PDE by using a finite element method. Specifically, if we consider the time dependent PDE

$$\frac{\partial u}{\partial t} = \mathcal{L}u, \quad (4.1)$$

with \mathcal{L} a spatial differential operator. The continuous version of the MFE determines the *solution and the mesh together* by minimising the residual

given by

$$\min_{\mathbf{v}, \frac{Du}{Dt}} I \left[\mathbf{v}, \frac{Du}{Dt} \right] \equiv \int_{\Omega_P} \left(\frac{Du}{Dt} - \nabla u \cdot \mathbf{v} - \mathcal{L}u \right)^2 W dx. \quad (4.2)$$

Here the function W is a weight function for which

$$W = 1$$

in the usual version of MFE described in Miller and Miller (1981) and Miller (1981), alternatively we can take

$$W = \frac{1}{1 + |\nabla u|^2},$$

for the weighted form of MFE described in Carlson and Miller (1998a) and Carlson and Miller (1998b). Observe that MFE is naturally trying to advect the mesh along with the solution flow. Of course in practice this equation is discretised using a Galerkin method.

This method is elegant and when tuned correctly works well (Baines 1994). However, it does have significant disadvantages. One of these is that the functional derivative of I with respect to \mathbf{v} can become singular and regularization is needed in practice.

4.2. Geometric conservation law (GCL) based methods

The Geometric Conservation Law (GCL) methods are based upon a direct differentiation of the equidistribution equation with respect to time, to derive an equation for the mesh velocity. In its simplest form the method assumes that the monitor function M is normalised so that the integral of M over Ω_P is constant (typically unity). If A is an arbitrary measurable set in Ω_C it follows that

$$I = \int_A \mathbf{d}\xi = \int_B M \mathbf{d}x,$$

where $B = F(A)$. Now, even if A is fixed then, as the mesh is moving the set B will typically change with time, with the points on the boundary of B moving with velocity \mathbf{v} . An application of Reynold's transport theorem implies that

$$\frac{d}{dt} \int_B M \mathbf{d}x = \int_B M_t \mathbf{d}x + \int_{\partial B} M \mathbf{v} \cdot \mathbf{d}S = \int_B (M_t + \nabla \cdot (M \mathbf{v})) \mathbf{d}x.$$

However, as A is fixed, it follows that $dI/dt = 0$. Furthermore the set A is arbitrary. It follows that M and hence \mathbf{v} must satisfy the (geometric) conservation law

$$M_t + \nabla \cdot (M \mathbf{v}) = 0. \quad (4.3)$$

If M is known, then (4.3) gives an equation for the (mesh) velocity \mathbf{v} . This equation must be augmented with the boundary condition

$$\mathbf{v} \cdot \mathbf{n} \quad \text{on} \quad \partial\Omega_P. \quad (4.4)$$

The equations (4.3,4.4) have a unique solution in one-dimension, but many solutions in higher dimensions. To determine \mathbf{v} uniquely additional conditions must be imposed. In the derivation of the optimal transport methods we saw the use of an additional condition on the curl of the solution in the computational space. In the various forms of the geometric conservation law (GCL) methods the curl of the velocity is imposed in the *physical space*. In particular, for a suitable weight function w and a background velocity field \mathbf{u} the condition

$$\nabla \times w(\mathbf{v} - \mathbf{u}) \quad (4.5)$$

is imposed, so that for an appropriate potential function ϕ we have

$$\mathbf{v} = \mathbf{u} + \frac{1}{w} \nabla \phi.$$

Here ϕ is unknown, and we presume that \mathbf{u} is specified in advance. Substituting into the conservation law, it then follows that ϕ satisfies the *elliptic partial differential equation*

$$\nabla \cdot \left(\frac{M}{w} \nabla \phi \right) = -M_t - \nabla \cdot (M\mathbf{u}). \quad (4.6)$$

with the boundary condition on $\partial\Omega_P$ given by

$$\frac{\partial \phi}{\partial n} = -w\mathbf{u} \cdot \mathbf{n}.$$

This equation is a *scalar linear equation* for ϕ when posed (and solved) in the computational space. Unlike the MFE methods this partial differential equation is well-defined for all time. In a similar manner to the optimal mesh methods we need only solve a scalar equation, however unlike the optimal mesh methods this equation is linear in the physical coordinates. Observe further, than unlike certain of the MMPDE and variational based methods described in Chapter 3, this system automatically deals with the mesh location on the boundaries. Having determined ϕ from this equation, the mesh velocity \mathbf{v} can be determined directly, and the mesh then found from integrating the equation $\mathbf{x}_t = \mathbf{v}$ with respect to time using, for example, an SDIRK method. This time integration to give \mathbf{x} has the disadvantage of possibly introducing mesh tangling, and of mesh points moving out of the domain during the course of the integration, if an appropriately coarse discretisation is used. This method is described in (Cao, Huang, and Russell 2002)

An alternative formulation, also described in (Cao, Huang, and Russell 2002)

considers a variational formulation, where it is shown that the solution of (4.3) is also the minimiser of the functional

$$I[\mathbf{v}] = \frac{1}{2} \int_{\Omega_P} \left(|\nabla \cdot (M\mathbf{v}) + M_t|^2 + \left(\frac{M}{w}\right)^2 |\nabla \times w(\mathbf{v} - u)|^2 \right) \mathbf{d}x. \quad (4.7)$$

This equation can be discretised using a finite element method using basis functions defined over the mesh in the physical space, and \mathbf{v} found using a simple Galerkin method. The mesh points can then also be found from \mathbf{v} through a Galerkin calculation. Details of these calculations are given in Baines et. al. (2005) and Baines et. al. (2006) where the GCL method is coupled to an ALE (Arbitrary Lagrangian Eulerian) method for discretising the underlying PDE.

In the usual implementation of the GCL method the weight function

$$w = 1$$

is taken. This gives an irrotational mesh (in the physical coordinates) when the background velocity $\mathbf{u} = 0$. In many implementations of GCL $\mathbf{u} = 0$. However, for certain applications, such as in computational fluid dynamics, the background velocity can be taken to be the flow velocity.

The appeal of the GCL methods is that to find ϕ (and hence \mathbf{v}) we need only solve a scalar linear elliptic partial differential equation. This seems to have the advantage over the optimal transport methods which require the solution of a nonlinear equation. However, like other velocity based methods it has the potential disadvantages of having problems with mesh tangling and mesh skewness as the mesh points follow the moving features in a monitor function. Furthermore, we require the solution of the mesh equations in the physical domain rather than the computational domain. This loses some of the speed advantages gained (by for example the use of spectral methods) when solving the mesh equations on a very uniform mesh in the computational domain.

We now consider two examples taken from (Cao, Huang, and Russell 2002) of the application of the GCL method which allow direct comparison with the optimal transport based methods described in Chapter 3.

Example 1. This first example looks at the mesh generated by a monitor function which concentrates points around a moving disc. This is considered to be a difficult test problem for a moving mesh method.

$$M_1(x, y, t) = \frac{d(x, y, t)}{\int_{\Omega_P} d(\tilde{x}, \tilde{y}, t) d\tilde{x}d\tilde{y}}$$

where

$$d(x, y, t) = 1 + 5 \exp(-50|(x - \frac{1}{2} - \frac{1}{4} \cos(2\pi t))^2 + (y - \frac{1}{2} - \frac{1}{4} \sin(2\pi t))^2 - .01|).$$

Results are shown in Fig. 4.13.

Figure 4.13. Example 1: Moving meshes at time $t = 0, 0.5, 0.75, 1$

If we compare the calculated mesh to that of the identical Example 1 in the application of the PMA method in Chapter 3 we see that the GCL method has not performed as well in this case. In particular we see some significant effects of mesh distortion and of mesh points lagging behind with the Lagrangian based method leading to mesh skewness and eventually to tangling and singular behaviour. This is in contrast to the much more regular mesh generated by the position based PMA method.

Example 2. In this second example we consider a monitor function which concentrates mesh points around an oscillating front.

$$M_1(x, y, t) = \frac{d(x, y, t)}{\int_{\Omega_P} d(\tilde{x}, \tilde{y}, t) d\tilde{x}d\tilde{y}}$$

where

$$d(x, y, t) = 1 + 5 \exp(-50|y - \frac{1}{2} - \frac{1}{4} \sin(2\pi x) \sin(2\pi t)|).$$

The results of using the GCL method in this case are shown in Fig. 4.14.

Figure 4.14. Example 2: The moving meshes generated by the GCL method at the times $t = 0, 0.25, 0.75, 1$

We can compare this mesh to that generated in Example 2 of the application of the PMA method. We can see that the GCL method has successfully followed the moving sin-wave but unlike the mesh generated by the PMA method, there is a small degree of instability manifested by some oscillations of the mesh visible when $t = 1$ which are not present in the mesh when $t = 0$.

4.3. The Deformation Map Method

Deformation Map Methods (closely related to Harmonic Map Methods) take a strongly differential geometric approach to moving mesh generation.

Liao and his coworkers (Liao and Anderson 1992),(Semper and Liao 1995), Liao and Xue: J. Comput. Appl. Math. 2006) have proposed a moving mesh method based on *Deformation Maps*. Such maps were introduced by Moser (1965) and Dacorogna and Moser (1990) in their study of volume elements of a compact Riemannian manifold to prove the existence of a C^1 diffeomorphism with specified Jacobian. In this method, the mapping $\mathbf{x} = F(\xi, t) : \Omega_C \rightarrow \Omega_P$ is determined from the system of equations

$$\begin{cases} x_t = \frac{1}{M(x,t)} \nabla \phi(x, t) & \text{in } \Omega_P, \\ \Delta \phi = -\frac{\partial M}{\partial t} & \text{in } \Omega_P, \\ \frac{\partial \phi}{\partial n} = 0 & \text{on } \partial \Omega_P. \end{cases} \quad (4.8)$$

It is easy to see that (4.8) corresponds to the system (4.6) – the GCL method formulation which involves the potential function ϕ in the case where $\mathbf{u} = 0$ and $w = M$.

They also use the alternative formulation in Bochev, Liao and Pena (1996) given by

$$\begin{cases} x_t = \frac{\nu(v,t)}{M(x,t)} & \text{in } \Omega_P \\ \nabla \cdot \nu = -\frac{\partial M}{\partial t} & \text{in } \Omega_P \\ \nabla \times \nu = 0 & \text{on } \partial \Omega_P. \end{cases} \quad (4.9)$$

Note that letting $\nu = M\mathbf{v}$, this becomes the div-curl system (4.3) and (4.5) with $\mathbf{u} = 0$ and $w = M$.

The equation for x_t in both cases is nonlinear, and for simplicity explicit time integration schemes are used by Liao and his coworkers. For (4.8), the solution of the potential equation for ϕ is straightforward. However, an implicit integration of (4.8) requires interpolation of the potential function ϕ for values at points other than grid nodes, and the flux free boundary condition cannot generally be preserved. For (4.9), the div-curl system is solved using a least squares approach.

4.4. Some other velocity based methods

The natural idea of moving mesh points so that the velocity of the points reflects underlying dynamics of the solution has led to many other velocity based methods than those described above. In the velocity based methods of Anderson and Rai (1983), the mesh is moved according to attraction and repulsion pseudo-forces between nodes motivated by a spring model in mechanics.

In contrast Dorodnitsyn (1991), Dorodnitsyn (1993a), Dorodnitsyn (1993b) and his co-workers (Dorodnitsyn and Kozlov 1997), (Kozlov 2000) has derived a series of velocity based methods in which the underlying symmetries of the partial differential equation to be solved are used to guide the move-

ment of the grid points. Such methods have the potential to preserve all of the symmetric invariants of the underlying PDE in the discretised system, and indeed precisely these invariants are used in the calculation of the mesh points. They are closely related to the (position based) scale-invariant methods described in the next chapter. Such methods can also lead to conservation laws derived from a discrete version of Noether's theorem. They have been applied in Dorodnitsyn and Kozlov (1997) to solve a variety of nonlinear diffusion equations and in Budd and Dorodnitsyn (2001) to integrating the nonlinear Schrödinger equation. The main disadvantage of these methods is that they tend to be highly nonlinear, and the equations for the mesh points hard to solve. More details of the implementation and application of these methods are given in Budd and Piggott (2005).

5. Applications of moving mesh methods

In this final chapter we look at some applications of moving mesh methods to a variety of problems related to the solution of partial differential equations. As described in Chapter 1 there are a vast number of problems for which moving mesh methods have been used with great success and we cannot hope to summarise all of them in this chapter. For example, one of the most important set of application arises in fluid mechanics and reviews of these comparing many different methods and problems are given in (Baines 1994), Eisman (1985), Tang (2005), Tang and Tang (2003), Yanenko, Krosho, Lisejkin, Fomin, Shapev and Shitov (1976).

Instead, we look in this chapter at some specific problems and classes of problems which aim to highlight some of the special advantages and disadvantages of the moving mesh method for solving partial differential equations. In particular we look at problems where moving meshes can exploit natural solution scaling structures (including self-similarity); blow-up problems in which solution singularities arise in finite time; and problems such as Burgers' equation where moving meshes are used to capture the motion of moving fronts. We conclude this Chapter with a discussion of the link between moving mesh methods and the use of PDE based methods in image processing.

5.1. Symmetry, Self-Similar Solutions and Scale-Invariance

Many naturally occurring physical systems have an invariance under symmetries including *translations*, *rotations* and *changes of scale*, and this is reflected in the partial differential equations that describe them. The solutions of these equations may then either be themselves invariant under (combinations of) these symmetries, the *self-similar solutions*, or they can be transformed into other solutions through the application of symmetry operators. Conservation laws can be linked through Noether's theorem Olver (1986), (Dorodnitsyn 1993b) to many of the continuous symmetries. An example of such a system is Burgers' equation

$$u_t + u \cdot \nabla u = \nu \Delta u$$

which is invariant under *rotations* in space and *translations* in both space and time. It admits travelling wave solutions which are self-similar solutions coupling spatial and temporal translation, with the wave-speed giving the coupling. The waves can be at any orientation, and the action of a rotation is to map one wave to another. In other systems a rescaling of space, time and of the solution, leaves the partial differential equations governing the system invariant. Such changes of scale were first observed by Kepler in his studies of the solar system and summarised in his famous *third law*

in which he observed that if a planetary orbit existed with a solution (in polar coordinates) given by $(r(t), \theta(t))$ then there was also a solution of the form $(\lambda^3 r(\lambda^2 t))$ for any positive value of λ . Such scaling invariance also arises in the equations of fluid and gas dynamics, nonlinear optics and many biological systems. An excellent summary of scale-invariant systems with such properties is given in the book (Barenblatt 1996).

We can then ask the question of whether a numerical method can be constructed which is also invariant under symmetries. A non-adaptive method has difficulties here as the fixed spatial mesh does not admit any symmetry operation. More generally an adaptive method can be designed to exploit the symmetries in the problem. In a sense the answer to this question is obvious as we can always perform an a-priori scaling of the problem and then use a method in the scaled coordinates. However, this assumes more knowledge of the physical system than we may easily have available to us. Indeed, for some systems there are a number of different possible changes of scale and it is not a-priori obvious which one correctly describes the system evolution.

We now show how several of the moving mesh strategies described earlier, particularly those based on moving mesh partial differential equations, can be very effectively applied to such problems with symmetry. Such methods have several important properties. If carefully designed they can admit discrete self-similar solution, they can have (local truncation) errors which are invariant under changes in the scale of the problem, they can have discrete conservation laws and they can cope with symmetric singular structures. Such problems have been studied by a number of authors (Baines et. al. 2006), (Budd and Piggott 2005), (Ceniceros and Hou 2001), (Dorodnitsyn 1991), (Dorodnitsyn 1993b), (Kozlov 2000).

The underlying problem that we will consider is the partial differential equation

$$u_t = f(u, \nabla u, \Delta u, \dots). \quad (5.1)$$

Observe that this equation is invariant under translations in space and time and also rotations in space. We also assume it to be invariant under the action of the scaling symmetry (or symmetries)

$$t \rightarrow \lambda, \quad \mathbf{x} \rightarrow \lambda^\alpha \mathbf{x}, \quad u \rightarrow \lambda^\beta u. \quad (5.2)$$

The objective is now to derive an adaptive method which reflects the underlying symmetries. This can be achieved if (away from any finite boundaries, the equations describing the mesh (regardless of whether these are position or velocity based) are *invariant under the action of some symmetry operations as the underlying PDE* so that translations in space and time rotations and scalings of the form (5.1) leave the mesh equations invariant. This is

in many ways a very natural question to ask of an r -adaptive method, for which the mesh may be regarded as a dynamic object, amenable to the action of symmetries involving space and time. It is much harder to see how h and p type methods can be considered in this manner.

As an example of such a problem, (see also (Baines et. al. 2006) for the use of an adaptive ALE (Arbitrary Lagrangian Eulerian) method in higher dimensions, and (Dorodnitsyn 1993a),(Dorodnitsyn and Kozlov 1997) for a more abstract treatment) we consider the one-dimensional *porous medium equation* given by

$$u_t = (u^m)_{xx} \quad (5.3)$$

where we consider this equation posed on the whole real line, with mild decay conditions on the solution u at ∞ . This partial differential equation is invariant under two different, and independent, changes of scale, one of space and the other of the solution. These are given by

$$t \rightarrow \lambda t, \quad x \rightarrow \lambda^{1/2} x$$

and

$$t \rightarrow \mu t, \quad u \rightarrow \mu^{1/(m-1)} u.$$

Any (linear) combination of these two changes of scale will also leave the equation (5.3) invariant.

Whilst the PDE may be invariant under the action of a symmetry group, not all of the solutions have this property, although one solution can be mapped into another by the action of the symmetry operator. Those solutions which are themselves invariant under the action of the symmetry operator are termed the *self-similar solutions*. A self-similar solution satisfies the functional equation

$$u(\lambda t, \lambda^\alpha \mathbf{x}) = \lambda^\beta u(t, \mathbf{x}) \quad (5.4)$$

Such a solution can be described in terms of a new set of coordinates, typically of the form

$$u(t, \mathbf{x}) = t^\beta v(\mathbf{z}), \quad \mathbf{z} = \mathbf{x}/t^\beta. \quad (5.5)$$

In the case of the porous medium equation the self-similar solution with *constant mass* takes the form

$$u(t, x) = t^{-1/3} v(x/t^{1/3}).$$

The function $v(\mathbf{z})$ generally satisfies an ordinary differential equation, which is much simpler than the original PDE. The solutions of this ODE which correspond to solutions of the PDE are generally those with certain decay or boundedness conditions as $|\mathbf{z}| \rightarrow \infty$. This allows the construction of exact solutions to PDEs in many cases. In the example of the porous medium

equation we have the famous Barenblatt-Pattle solution (Barenblatt 1996) given by

$$u(x, t) = t^{-1/3}(a - x^2/t^{2/3})_+,$$

where a is a constant. Significantly such solutions are *globally attracting* (in the sense of L_1 -convergence (?)). An extensive description of such problems is given in Barenblatt (1996) and Olver (1986).

More generally, solutions of PDES with symmetries may also take the form

$$u(t, \mathbf{x}) = U(t)v(\mathbf{z}), \quad \text{where } \mathbf{z} = \mathbf{x}/L(t). \quad (5.6)$$

Here $U(t)$ and $L(t)$ are appropriate solution and length scales. In the case of self-similar solutions these are pure powers of t (or a translation of t) but they can take more general forms. For example in the case of the blow-up equation

$$u_t = u_{xx} + u^3$$

with $u \rightarrow \infty$ as $t \rightarrow T$ we have

$$U(t) = (T - t)^{-1/2} \quad \text{and} \quad L(t) = (T - t)^{1/2} |\log(T - t)|.$$

Such solutions are called *approximately self-similar solutions* (Galaktionov 1973). Examples of this type of behaviour also arise in the nonlinear Schrodinger equation in dimension two and in the Chemotaxis equations of mathematical biology.

The self-similar (and approximately self-similar) solutions of PDEs can play an important role in the description of the solution (beyond the fact that they lead to exact solutions). This is because they often describe very well the *intermediate asymptotics* of the solution, which is the behaviour of the solution after the transient effects of any initial conditions and before boundary terms become important. They are also often effective in describing certain singular types of behaviour such as the peaks in the blow-up and related problems, and also the interfaces in various problems in gas dynamics Barenblatt (1996). It is therefore useful to have numerical methods which can accurately reproduce self-similar behavior when it arises in applications. One method that has been used is to make an a-priori choice of (self-similar) variables, so that the PDE can be reduced to an ODE and to then solve this ODE numerically. This method however has a number of disadvantages. Firstly, it cannot deal with general initial and boundary conditions satisfied by the PDE. Secondly, there may often be several symmetry groups acting on a partial differential equation and it may often not be at all clear which (if any) leads to a self-similar solution. Indeed, there are many problems (for example the heat equation posed on a finite interval with an initially highly localised solution) which may have one form of self-similar behavior

for part of the evolution and another over longer times (for example when the solution of the heat equation interacts with the boundary). Problems of this form (called type II self-similar solutions in Barenblatt (1996)) are extremely hard to analyse in advance of any PDE calculation.

An alternative approach, which makes considerable use of the r -adaptive methods, is to use a numerical method which *admits the same scaling invariances as the original PDE away from any boundaries*. Such methods are called *scale-invariant* (Budd and Piggott 2005), (Baines et. al. 2006) as they perform identically under the scaling transformations.

The key to the design and implementation of such methods lies in the use of the moving mesh partial differential equations to describe the location of the mesh points. By an appropriate choice of monitor function it is often possible to construct such MMPDES to be invariant under the action of the scaling transformations. The advantages of such methods are:

- They usually have *discrete self-similar solutions* which inherit many of the properties of the underlying self-similar solutions.
- If designed carefully they may work for several types of scaling symmetry and thus can be used in the case of Type II self-similarity when the exact scaling group is not known in advance.
- They can be applied to problems with arbitrary initial and boundary conditions
- They can have relative truncation errors which are *independent of the scale of the solution* (Budd, Leimkuhler and Piggott 2001),(Baines et. al. 2006)

We give a partial proof of these results presently. A further, but less general, advantage of such methods is that they also often preserve the asymptotic properties of the (approximately) self-similar solutions. This is seen both in the global convergence towards the self-similar solutions of the porous medium equation, and the local convergence towards the singular profile described by the approximately self-similar solution of the blow-up equation.

We start this calculation by looking at a moving mesh method in one-dimension for which the moving mesh PDE is given by MMPDE1 and for which the monitor function is a function of u and u_x so that

$$(M(u, u_x)x_\xi)_\xi = 0.$$

If this is to be invariant under the action of the scaling symmetry $t \rightarrow \lambda t, x \rightarrow \lambda^\beta x, u \rightarrow \lambda^\gamma u$ we require that

$$(M(\lambda^\gamma u, \lambda^{\gamma-\beta} u_x)\lambda^\beta x_\xi)_\xi = 0.$$

This is satisfied (for all β) provided that the monitor function satisfies the

functional equation

$$M(\lambda^\gamma u, \lambda^{\gamma-\beta} u_x) = \lambda^\theta M(u, u_x), \quad (5.7)$$

where θ is arbitrary. Many monitor functions do not satisfy this functional equation, for example the simple arc-length monitor $M = \sqrt{1 + u_x^2}$ (although it approximately satisfies it when $|u_x|$ is large.) However, it is certainly possible to find function which do, and a simple example is given by

$$M(u, u_x) = u^\delta$$

for some choice of δ . Observe that this monitor function is invariant under a very arbitrary set of scaling symmetries and using it poses no explicit a-priori scaling on the solution. It is thus very useful when considering self-similar solutions of Type II.

As an example we consider the porous medium equation in the form

$$u_t = (uu_x)_x \quad |u| \rightarrow 0 \quad \text{as} \quad |x| \rightarrow \infty. \quad (5.8)$$

It is easy to see that the first integral of this solution is constant, and we may scale the solution so that for all t we have

$$\int_{-\infty}^{\infty} u \, dx = 1.$$

It is then natural to choose $M = u$ so that the monitor function has unit integral over the physical domain. It follows immediately that

$$Mx_\xi = 1. \quad (5.9)$$

Furthermore, from the geometric conservation law given by

$$M_t + (M\dot{x})_x = 0$$

we have

$$u_t + (u\dot{x})_x = 0.$$

Substituting for u_t and integrating gives

$$u(u_x + \dot{x}) = 0,$$

so that the Lagrangian equation for the mesh points is given by

$$\dot{x} = -u_x. \quad (5.10)$$

This equation is used in Dorodnitsyn (1993a) and Dorodnitsyn and Kozlov (1997) as the equation of motion of all of the mesh points. Note that this is also the equation for the movement of the leading edge of the front of those solutions of the porous medium equation which have compact support. The same monitor function is used in Baines et. al. (2006) to compute solutions of the porous medium equation using the scale-invariant ALE method.

In the usual manner, either of the equations (5.9, 5.10) can be discretised and solved simultaneously with the porous medium equation (5.8) (see Budd, Collins, Huang and Russell (1999) for more details) so that the discrete solution and mesh points are given by

$$U_i \approx u(X_i, t), \quad X_i = x(i\Delta\xi).$$

A similar procedure can also be used with a variational formulation in higher dimensions (Baines et. al. 2006)

It is *immediate* (see Budd, Collins, Huang and Russell (1999)) that any such discretisation admits a *discrete self-similar solution* of the form

$$U_i = t^{-1/3}V_i, \quad X_i = t^{1/3}Z_i. \quad (5.11)$$

It can also be shown (Budd and Piggott 2005) that such self-similar solutions are not only locally stable but they are also *global attractors* so that they correctly organise the qualitative long-term dynamics of the solution and they even obey a discrete maximum principle.

Construction of scale-invariant MMPDEs

The porous medium equation has relatively benign dynamics, which allows us to use a relatively simple moving mesh equation to evolve the mesh. In the case of systems with more extreme forms of dynamics, such as that arising in shocks or localised singularities, it is usually necessary to use a more sophisticated moving mesh equation to avoid mesh instabilities.

In one-dimension two possible equations for the mesh are the moving mesh PDES MMPDE5 and MMPDE6 given by

$$\epsilon x_t = (Mx_\xi)_\xi, \quad -\epsilon x_{\xi\xi t} = (Mx_\xi)_\xi. \quad (5.12)$$

Suppose that these are used to solve a system which has intrinsic solution, length and time scales given respectively by U , L and T and a derived scale Λ for the monitor function M . As the computational variable ξ is *independent of scale*, the left hand side of each of these two MMPDEs scales as L/T and the right hand as ΛL . These two balance (so that the mesh evolves at the same rate as the underlying solution) provided that

$$\frac{1}{T} = \Lambda. \quad (5.13)$$

Observe, that this condition is independent of the spatial length scale L . The implication of this is that if the monitor function $M(u, x)$ depends upon u and x then this satisfies (5.13) provided that

$$M(Uu, Lx) = \frac{1}{T}M(u, x). \quad (5.14)$$

This is a more severe condition than the condition (5.14) for MMPDE1 but is necessary to ensure that the mesh calculation does not destabilise the calculation of the solution of the PDE. It has the useful property that it often gives a precise characterisation of the necessary form of the monitor function appropriate to one scaling transformation. A disadvantage of this approach is, however, that it may not always (or indeed generally) be possible to capture *all* possible scaling transformations in a *single monitor function* satisfying (5.14). Thus some a-priori knowledge of the expected solution behaviour might be necessary in this case.

As an example we will consider the radially symmetric solutions of the cubic nonlinear Schrödinger equation. This equation has the form

$$iu_t + u_{rr} + \frac{n-1}{r}u_r + u|u|^2 = 0 \quad (5.15)$$

where $r = |x|$ and n is the spatial dimension. If $n \geq 2$ this can have solutions which develop singularities (in amplitude and phase) in a finite time. The equation (5.15) is invariant under the action of the scaling group

$$t \rightarrow \lambda t, \quad r \rightarrow \lambda^{1/2}r \quad u \rightarrow \lambda^{-1/2}u$$

as well as the unitary multiplicative group

$$t \rightarrow t, \quad r \rightarrow r, \quad u \rightarrow e^{i\phi}u, \quad \phi \in R.$$

This system develops singularities in a finite time T with a natural time-scale of $(T-t)$ a length scale of $L = (T-t)^{1/2}$ and a solution scale of $U = (T-t)^{-1/2}$. For the moving mesh PDE based on a simple monitor function of the form $M(u)$, to be invariant under the action of the unitary multiplicative group we require that

$$M(u) = M(|u|).$$

The condition (5.14) then implies that

$$M((T-t)^{-1/2}u) = \frac{1}{(T-t)}M(u).$$

Both conditions are satisfied if

$$M(u) = |u|^2. \quad (5.16)$$

Calculations using a regularised form of this monitor function are described in the next section.

Scale invariant moving mesh methods for a general class of scale-invariant PDEs can also be constructed in higher dimensions $\mathbf{x} \in R^n$. We consider two cases, firstly the method of Ceniceros and Hou (2001), and then the optimal transport method. The first of these describes a two-dimensional

moving mesh generated by the PDE system

$$x_t = \nabla_{\xi} \cdot (M \nabla_{\xi} x), \quad y_t = \nabla_{\xi} \cdot (M \nabla_{\xi} y).$$

This system scales in an identical manner to both MMPDE5 and MMPDE6 and consequently is scale invariant provided that the monitor function M satisfies the condition (5.14). In the case of optimal transport in an n -dimensional system, the PMA equation gives a mesh from $\nabla_{\xi} Q$ where Q satisfies the PDE

$$(I - \gamma \Delta) Q_t = (M(u) H(Q))^{1/n}. \quad (5.17)$$

Now, if the underlying problem has the natural scaling symmetry $\mathbf{x} \rightarrow \lambda \mathbf{x}$ then this is equivalent to the scaling symmetry $Q \rightarrow LQ$. It is immediate that

$$H(LQ) = L^n H(Q).$$

It then follows immediately that (5.17) is invariant under the action of the scaling symmetries provided that the monitor function satisfies the functional equation

$$M(Uu, L\mathbf{x})^{1/n} = \frac{1}{L} M(u, \mathbf{x}). \quad (5.18)$$

We note that (in two-dimensions) the scaling structure of the function Q also implies that the mesh skewness s as defined in Chapter 3 by the relation

$$s = \frac{\Delta(Q)^2}{H(Q)} - 2$$

is *invariant under the scale change* $Q \rightarrow LQ$. This implies that if the mesh is generated by a scale invariant PMA type method then any mesh regularity is *preserved under scaling*.

Discrete self-similar and approximately self-similar solutions

As mentioned above, a significant benefit of using a scale-invariant adaptive scheme is that it admits discrete self-similar solutions. One reason for this is the almost trivial, yet very important, observation that

the actions of discretisation and of rescaling commute

where here a discretisation can be any of a finite difference, collocation, finite element or a finite volume method. (This means that a discretisation of a rescaled solution will be identical to a rescaling of a discrete solution). More formally, if the PDE is invariant under the action of the scaling group

$$t \rightarrow \lambda t, \quad \mathbf{x} \rightarrow \lambda^{\beta} \mathbf{x}, \quad u \rightarrow \lambda^{\alpha} u$$

then a *continuous* self-similar solution takes the form

$$u(x, t) = t^\alpha v(\mathbf{y}) \quad \mathbf{y} = \mathbf{x}/t^\beta. \quad (5.19)$$

In terms of the computational variables this becomes

$$u(\mathbf{x}(\xi, t) = t^\alpha v(\xi), \quad \mathbf{x}(\xi, t) = t^\beta \mathbf{y}(\xi),$$

for appropriate functions v and y . This leads immediately to a *discrete self-similar solution* of the form

$$U_i(t) = t^\alpha V_i, \quad X_i(t) = t^\beta Y_i, \quad (5.20)$$

For convenience we now study this discrete self-similar solution in the context of a prototypical example a one-dimensional system governed by a semi-linear second order PDE of the form

$$u_t = u_{xx} + f(u) \quad \text{or in Lagrangian form} \quad u_t = u_{xx} + f(u) + \dot{x}u_x$$

This PDE is invariant under the action of the scaling group

$$t \rightarrow \lambda t, \quad x \rightarrow \lambda^{1/2} x, \quad u \rightarrow \lambda^\alpha u$$

(so that $\beta = 1/2$) provided that the function $f(u)$ satisfies the functional equation

$$f(\lambda^\alpha u) = \lambda^{\alpha-1} f(u).$$

Substituting in the expression for the self-similar solution (5.19), and setting $\beta = 1/2$ we see that the function $v(y)$ must satisfy the *ordinary differential equation*

$$\alpha v - \frac{y}{2} v_y = v_{yy} + f(v) \quad \text{so that} \quad \alpha v = v_{yy} + f(v) + \frac{y}{2} v_y. \quad (5.21)$$

Now consider a simple centred finite-difference discretisation of the Lagrangian form of the underlying PDE which takes the form

$$\dot{U}_i = \frac{\frac{U_{i+1}-U_i}{X_{i+1}-X_i} - \frac{U_i-U_{i-1}}{X_i-X_{i-1}}}{\frac{1}{2}(X_{i+1}-X_{i-1})} + f(U_i) + \dot{X}_i \frac{U_{i+1}-U_{i-1}}{(X_{i+1}-X_{i-1})}$$

We can substitute (5.20) directly into this expression to give

$$\alpha V_i = \frac{\frac{V_{i+1}-V_i}{Y_{i+1}-Y_i} - \frac{V_i-V_{i-1}}{Y_i-Y_{i-1}}}{\frac{1}{2}(Y_{i+1}-Y_{i-1})} + f(V_i) + \frac{Y_i}{2} \frac{V_{i+1}-V_{i-1}}{(Y_{i+1}-Y_{i-1})}. \quad (5.22)$$

It is immediately obvious that (5.22) is a consistent discretisation of the ordinary differential equation (5.21) so that the function $v(y)$ will be approximated by V_i at the (time-independent computational) mesh point Y_i . Note further that the discretisation error in approximating v by V_i is *independent of the solution scale*. In the case of systems such as the blow-up

problems of the next section this implies that the asymptotic form of the singularity at the peak will be approximated with uniform accuracy for peaks with very small spatial scales (and correspondingly large solution scales). Although we note that other errors do arise at points where the (rapidly) moving mesh following the evolving peak matches to a nearly stationary mesh in the regions closer to the boundary of the domain.

To determine the location of the points Y_i in terms of the computational variables we must apply the MMPDE used to evolve the mesh. Again, to give an example we consider MMPDE5. Substituting (5.20) into a standard discretisation of MMPDE5 gives the following discrete equation for Y_i

$$\frac{1}{2}Y_i = \frac{1}{2\Delta\xi^2} (M_{i+1/2}(Y_{i+1} - Y_i) - M_{i-1/2}(Y_i - Y_{i-1})). \quad (5.23)$$

Crucially we note that the functional equation (5.14) satisfied by the monitor function M allows this rescaling to be made. Similar discrete equations arise for other choices of MMPDE provided that M satisfies (5.14).

We note that exactly the same rescalings are possible in higher-dimensional problems using moving meshes generated by any other methods described in this section and for any other form of discretisation (provided that the processes of discretisation and rescaling commute).

It is also possible to construct approximate discrete self-similar solutions in cases where the underlying solution is better described by approximate self-similar variables. Details of the calculations in this case are given in Budd and Williams (2006).

As an example of this we return to the porous medium equation in one-dimension. For this problem, for all constants $C > 0$ there is a self-similar solution $u_s(x, t)$ of the form

$$u_s(x, t) = (t + C)^{-1/3}v(y), \quad x = (t + C)^{1/3}y,$$

where the function $v(y)$ is given by the Barenblatt-Pattle profile (Barenblatt 1996). It is also well known that any positive initial data leads to a solution $u(x, t)$ which converges towards to a self-similar solution in the sense that

$$t^{1/3}u(t^{1/3}y, t) \rightarrow v(y)$$

Similarly it is shown in Budd, Collins, Huang and Russell (1999) that if the monitor function is chosen to give a scale-invariant scheme (for example $M = u$), then this scheme admits a set of *discrete self-similar solutions* on a moving mesh which take the form

$$U_i(t) = (t + C)^{-1/3}V_i, \quad X_i(t) = (t + C)^{1/3}Y_i.$$

Figure 5.1. Convergence of the solution in the computational domain to the discrete self-similar solution. Note the invariance of the solution profile in this domain. We also show the evolution of the mesh and note that this scales as $t^{1/3}$

Note that the product

$$W_i \equiv U_i X_i = V_i Y_i$$

is invariant in time. In Fig. 5.1 we show the results of a computation presented in the computational domain, using a scale invariant moving mesh method with $M = u$ and a centred finite-difference discretisation. In this calculation the initial data at $t = 0$ was taken to be an irregular function, and results are plotted at times $t = 0$ and $t = 10$. We also show (dashed) two discrete self-similar solutions with values of C chosen so that they initially lie above and below the solution. Note that the solution calculated is sandwiched between these two functions. We also present the corresponding mesh $X_i(t)$ and the scaled mesh $Y_i(t) = X_i(t)/t^{1/3}$. It is clear from these figures that the computed solution converges towards the discrete self-similar solutions (in correspondence with the continuous theory) and that the moving mesh tends towards the one for which Y_i is constant in time so that $X_i(t)$ scales asymptotically as $t^{1/3}$. Similar figures for solutions of the porous medium equation computed using a scale-invariant ALE method in two-dimensions are given in Baines et. al. (2006).

5.2. Blow-up and related problems

Parabolic blow-up

As mentioned in the previous section, a significant success in the application of moving mesh methods occurs in the study of parabolic partial differential equations (and also of systems of PDES) which have solutions which blow-up (so that they, or a derivative of them, becomes infinite) in some in a finite time T . We now consider such problems in a little more detail. Blow-up in the solution often represents an important change in the properties of the model that the equation represents (such as the ignition of a heated gas mixture) and it is important that it is reproduced accurately in a numerical computation. A survey of many different types of blow-up problem is presented in Galaktionov (1973). Blow-up typically occurs on increasingly small time and length scales and hence it is usually essential to use both temporarily and spatially adaptive methods for any such computation. A survey of methods and underlying numerical theory for blow-up type problems is given in Budd, Huang and Russell (1996) and ? with more recent references in Budd and Williams (2006) and Cenicerros and Hou (2001).

Mesh refinement (h-adaptive) methods have been used in some numerical studies of blow-up such as those described in (Berger and Kohn 1988), but moving mesh methods have proved to be more effective in both one and two-dimensions. This is due, in part, to the strong scaling symmetry properties of the solution close to blow-up (where the effects of boundary and initial conditions become progressively less important), and the way that this can be exploited by using the scale-invariant methods described in the previous sub-section.

A prototype example of a blow-up system is the parabolic partial differential equation

$$u_t = \Delta u + u^p, x \in \Omega, \quad p > 1, \quad u|_{\partial\Omega} = 0, \quad (5.24)$$

This has the property of *single point blow-up* in that for certain types of sufficiently large initial data, there is a point \mathbf{x}^* and a finite time T so that

$$u(x^*, t) \rightarrow \infty \quad \text{as} \quad t \rightarrow T.$$

Close to the point \mathbf{x}^* the solution develops a narrow peak which evolves in an approximately self-similar manner. It is not uncommon to consider solutions which change by ten orders of magnitude, with a reduction in the solution length scale by a similar amount. It is essential to use an adaptive method to capture such behaviour accurately.

Example 1 The first example that we consider is given by

$$\begin{cases} u_t = \Delta u + u^3, & \mathbf{x} = (x, y) \in \Omega = (0, 1)^2, \\ u(\mathbf{x}, t) = 0, & \mathbf{x} \in \partial\Omega, \\ u(\mathbf{x}, 0) = 5 \exp(-25(x - 0.45)^2 - 25(y - 0.35)^2). \end{cases} \quad (5.25)$$

This problem has the natural scaling symmetry

$$t \rightarrow \lambda t, \quad \mathbf{x} \rightarrow \lambda^{1/2} \mathbf{x}, \quad u \rightarrow \lambda^{-1/2} u,$$

it has a natural time-scale of $(T - t)$ and it is shown in (Galaktionov 1973) that the natural space and solution scales are given by the approximately self-similar variables

$$L = (T - t)^{1/2} |\log(T - t)|^{1/2} \quad U = (T - t)^{-1/2}.$$

To compute a solution, we augment this problem with the PMA equation (5.17) to determine the moving mesh with a monitor function of the form $M \equiv M(u)$. To obtain scale invariance for this system we require that M must satisfy the function equation (5.18) so that

$$M((T - t)^{-1/2} u)^{1/2} = \frac{1}{(T - t)} M(u).$$

A simple solution of this is $M(u) = u^4$. In practice this monitor function

Figure 5.2. **Example 1** Final grid for the blow-up example. Left: entire grid. Right: detail near the blow-up point. Note that the grid is quite regular in the vicinity of the singularity, with no evidence of any skewness or tangling.

can lead to instabilities due to placing too many points in the singular region and not enough closer to the boundary of the domain, and to overcome this we apply a McKenzie regularisation to give

$$M(u) = u(\mathbf{x}, t)^4 + \int_{\Omega_P} u(\mathbf{x}', t)^4 d\mathbf{x}'.$$

This choice of monitor leads to a mesh which automatically inherits the correct dynamic length scale of the underlying solution in the singular regions where u is large. We can then find a solution of the blow-up problem in the computational domain Ω_C using a finite difference method with a uniform $N = 30 \times 30$ mesh in the computational domain to discretise both the PMA equation and the Lagrangian form of underlying PDE. The resulting system of ODEs was then solved *simultaneously* using a BDF method. In this calculation, as the blow-up time was approached an adaptive time-step was used by applying the Sundman transformation as described in Budd, Leimkuhler and Piggott (2001). In this we take

$$\Delta t = \frac{1}{(\max u)^2}.$$

In Fig. 5.2 we show the final grid both over the whole domain and also close to the peak near the centre as well as the initial and final solutions. The integration was performed until $|u|_\infty = 10^{15}$, for which the peak has approximate length scale of 10^{-15} .

Over the course of the evolution we see a mesh compression (and a solution amplification) by a factor of 10^{12} in the physical domain. Note that this has been achieved with a very modest number of mesh points. The final mesh shows a similar gradation of mesh elements from size $\mathcal{O}(10^{-2})$ to size $\mathcal{O}(10^{-14})$. However, if we study the mesh close to the solution peak, as illustrated in Figure 5.2, we see that this shows a strong degree of local regularity, with no evidence of long thin elements or skewness in the region where the solution gradients are very large. This is exactly as predicted by the previous theory. (Away from the solution peak the mesh is less degree of smoothness, however in this region the solution gradients are much smaller than in the peak and the local truncation errors are thus lower.)

Example 2 In order to consider these results in the context of some of the error analysis presented in Chapters 2 and 3, it is instructive to look at a

second one-dimensional example in which we consider the blow-up problem

$$u_t = u_{xx} + u^3, \quad u_x(0) = u(1) = 0.$$

It is known (Galaktionov 1973) that this equation has solutions which blow up at the origin and in the peak the asymptotic blow-up profile of this solution takes the form

$$u(x, t) = \frac{1}{\sqrt{T-t}} \frac{1}{\sqrt{1+ax^2/L^2}} \quad (5.26)$$

where a is a constant (depending weakly on the initial conditions) which we make take equal to unity, and $L(t)$ is the natural length scale given by

$$L(t) = \sqrt{(T-t)|\log(T-t)|}.$$

As this is now a problem in one-dimension we initially take $M = u^2$ so that in the peak.

$$M = \frac{1}{(T-t)} \frac{1}{1+x^2/L^2}.$$

The integral of M is overwhelmingly dominated by the contribution in the peak so that

$$\theta = \int_0^1 M dx = \frac{L}{(T-t)} \tan^{-1}(1/L) \approx \frac{\pi L}{2(T-t)}.$$

We can then take a regularised monitor function of the form

$$\bar{M} = M + \theta, \quad \text{with} \quad \int_0^1 \bar{M} = 2\theta,$$

to ensure that we have a 50:50 mesh. In the peak, $\bar{M} \approx M$ and an equidistributed mesh satisfies the equation $Mx_\xi = 2\theta$ so that

$$x_\xi = \pi L(1+x^2/L^2).$$

It follows immediately that in the peak we have

$$x(\xi, t) = L(t) \tan(\pi\xi). \quad (5.27)$$

Observe that if $\xi < 1/2$ then $x = \mathcal{O}(L)$ and that this mesh matches naturally to one for which $x = \mathcal{O}(1)$ as $\xi \rightarrow 1/2$ so that we have (as required) half of the mesh points in the peak and half outside the peak. In Figure 5.3 we show the solution computed in the computational plane using a simple centred discretisation of the PDE using a uniform mesh with $N = 30$ points in the computational space. Observe the peak in the region $\xi < 1/2$ has a regular form with no evidence of instability.

In Fig. 5.4 we present the mesh evolving in time and also a plot of $\tau \equiv \log(T-t)$ as a function of $\log(x)$. Observe that the mesh in the first of these

Figure 5.3. **Example 2** Solution profile for the blow-up problem computed in the computational space up to a peak value of 10^{15} .

Figure 5.4. **Example 2** Blow-up mesh. Left: Grid in the natural coordinates. Right: Plotting $\tau = |\log(T - t)|$ as a function of $\log(X_i)$ at equally spaced values of ξ showing the contraction rate and approximately self-similar behaviour close to the origin.

figures has concentrated in the peak, but that there is still good resolution of the solution away from the peak. In the second figure we see that all of the grid cells move at the same rate close to the origin, demonstrating the self-similar form there.

The expression (5.27) allows a direct evaluation of the leading order terms in the truncation error Tr . Recall from (2.52) that the leading term in the truncation error in the discretisation of u_{xx} is given by

$$Tr = \Delta\xi^2 \left(\frac{1}{3} x_{\xi\xi} u_{xxx} + \frac{1}{12} x_{\xi}^2 u_{xxxx} \right).$$

From (5.27) we have

$$x = L \tan(\pi\xi), \quad x_{\xi} = \pi L \sec^2(\pi\xi), \quad x_{\xi\xi} = 2\pi^2 L \tan(\pi\xi) \sec^2(\pi\xi).$$

A straightforward calculation based on the scaling structures of u shows that in the peak region

$$\begin{aligned} u &= \mathcal{O}\left(\frac{1}{\sqrt{T-t}}\right), & u_{xx} &= \mathcal{O}\left(\frac{1}{L^2\sqrt{T-t}}\right), \\ u_{xxx} &= \mathcal{O}\left(\frac{1}{L^3\sqrt{T-t}}\right), & u_{xxxx} &= \mathcal{O}\left(\frac{1}{L^4\sqrt{T-t}}\right). \end{aligned}$$

If $\xi < 1/2$ we can also see that x and its derivatives are all of $\mathcal{O}(L)$. As a consequence, substituting into the expression for Tr , we have that in this region

$$Tr = \mathcal{O}\left(\frac{\Delta\xi^2}{L^2\sqrt{T-t}}\right) \quad \text{so that} \quad \frac{Tr}{u_{xx}} = \mathcal{O}(\Delta\xi^2).$$

We can see from this expression that the *relative truncation error* Tr/u_{xx} in discretising u_{xx} depends only on $\Delta\xi$ and not on the intrinsic solution scale. This analysis starts to break down as $\xi \rightarrow 1/2$ and the terms $x_{\xi\xi}$ etc. become large as the mesh adapted to the peak evolves into a uniform mesh

close to the boundary. However, the results shown above do not indicate and instability in the mesh in this limit.

Focusing solutions of the nonlinear Schrödinger equation

The nonlinear Schrödinger equation described in the previous section

$$iu_t + \Delta u + u|u|^2 = 0, \quad x \in R^n \quad (5.28)$$

is a model for the modulational instability of water waves and plasma waves, and is important in studies of nonlinear optics where the refractive index of a material depends on the intensity of a laser beam. In all dimensions it has the conserved quantities

$$\int |u|^2 dx \quad \text{and} \quad \int \left(|\nabla u|^2 - \frac{1}{2}|u|^4 \right) dx$$

corresponding to mass and energy respectively. In one-dimension (5.28) is integrable and can give rise to soliton type solutions. More generally it is an example of a Hamiltonian PDE. Many numerical (usually non-adaptive) methods have been derived to take advantage of this integrability (Budd and Piggott 2005), (McLachlan 1994). If it posed in n -dimensions where $n \geq 2$ then the PDE (5.28) is non-longer integrable and it may admit singular (focusing) solutions for certain initial data. A review of these is given in Sulem and Sulem (1999) which also describes some numerical computations using a moving mesh method based on Winslow's algorithm. Numerically this is a very difficult problem as high resolution in time and space is required to capture the strong self-focusing of the solution, to deal with the highly oscillatory nature of the solution 'tail' and to compute over a large domain to avoid boundary effects (Budd, Chen and Russell 1999), (Ceniceros 2002). In such singular solutions both the maximum value of the solution modulus, and its phase blow-up in a finite time T . The precise form of this blow-up and the initial conditions that lead to blow-up are the subject of much investigation (see the review in Sulem and Sulem (1999)) and much remains unresolved. Two significant open questions are: (1) what is the exact nature of the blow-up profile in 2-dimensions (where it is known to be approximately self-similar but the precise form is still unclear) and (2) do there exist radially symmetric self-similar solutions in three dimensions. In the latter case these are conjectured to take the form

$$u(r, t) = \frac{1}{\sqrt{(T-t)}} e^{ia \log(T-t)} Q(y), \quad r = \sqrt{T-t} y, \quad r = |x|,$$

where a is an unknown constant. The existence of such a solution can be addressed by a scale-invariant method applied to the (one-dimensional) class

Figure 5.5. The solution when $|u(0, t)| = 100000$ and 500000 note the narrow width of the peak.

Figure 5.6. The evolution of the mesh as the blow-up time T is approached. Observe that the product $X_i|u(0, t)|$ converges to a constant indicating that both the solution and the mesh are evolving in a self-similar manner.

of radially symmetric solutions. In this we take the monitor function

$$M = |u(r, t)|^2$$

derived in the previous section, and use a collocation based discretisation with $N = 81$ points. In such a calculation we would expect to observe a discrete self-similar solution for which the mesh points X_i should take the form $X_i = \sqrt{T - t}Y_i$ so that

$$|u(0, t)|X_i = Q(0)Y_i$$

is constant in time. In Fig. 5.5 we plot two solutions starting from initial data $u(r, 0) = 6\sqrt{2}\exp(-r^2)$ which has a computed blow-up time of $T = 0.0343013614215$. Observe the excellent resolution of the peak even when the peak amplitude is around 10^5 and the peak width is around 10^{-5} . In Fig. 5.6 we show the computed values of $W_i \equiv |u(0, t)|X_i$ as functions of $\tau = \log(T - t)$ for a range in which $|u|$ varies from 100 to 500000. These are clearly tending towards constants, indicating that both the solution and the mesh are evolving in a self-similar manner.

It is interesting to note that the scale-invariant methods are easy to use and give rather better results in this case than symplectic methods described in McLachlan (1994), despite the Hamiltonian structure of the problem. This is because the adaptive methods give much better resolution of the peak.

In contrast, further computations of focusing solutions of the nonlinear Schrödinger equation are given by Cenicerros (2002). In this case the following highly dispersive problem was studied

$$i\epsilon u_t + \frac{1}{2}\epsilon^2 u_{xx} + u|u|^2 = 0, \quad u(x, 0) \equiv u_0 = A_0(x)e^{iS_0(x)/\epsilon}, \quad (5.29)$$

with $\epsilon \ll 1$. In this calculation a moving mesh method closely based on that in Cenicerros and Hou (2001) with smoothing was used as described in Chapter 3. This took the form

$$(1 - \gamma\partial_{\xi\xi}^2)\dot{x} = (Mx_\xi)_\xi \quad (5.30)$$

with $\gamma = \max(M)$. In Cenicerros (2002) the Lagrangian form of the PDE (5.29) was discretised in the computational domain and solved alternatively with the MMPDE (5.30) using the methods described in Chapter 3. In particular a semi-implicit BDF method was used for the time integration of both the moving mesh equation and the underlying PDE. A monitor function of the form

$$M = \sqrt{1 + \beta^2 |u_\xi|^2 + \alpha^2 |u|^4}$$

was employed with

$$\beta = (2L)^2 \frac{\|u_{0,x}\|_\infty}{\|u_0\|}$$

and L a measure of the size of the computational domain. In this calculation, particular care had to be taken with the spatial discretisation of the mesh movement terms of the form $\dot{x}u_x$ arising in the Lagrangian form of (5.29) as the dispersive nature of (5.29) meant that there was no natural damping to the high frequency terms in the errors associated with a central difference discretisation. This is in contrast to the highly dissipative nature of the discretisation of (5.30) which took the form

$$\begin{aligned} \frac{X_j^{n+1} - X_j^n}{\Delta t} &= \gamma \frac{X_{j+1}^{n+1} - 2X_j^{n+1} + X_{j-1}^{n+1}}{\Delta \xi^2} - \gamma \frac{X_{j+1}^n - 2X_j^n + X_{j-1}^n}{\Delta \xi^2} \\ &+ \frac{1}{\Delta \xi^2} \left[M_{j+1/2}^n (X_{j+1}^n - X_j^n) - M_{j-1/2}^n (X_j^n - X_{j-1}^n) \right]. \end{aligned}$$

To avoid some of the resulting instabilities reported in Li, Petzold and Ren (1998) and discussed in the next section in the context of the solution of Burgers' equation, a high order (fourth order central difference) expression was used to discretise the Lagrangian advective term $\dot{x}u_x$. The resulting semi-implicit BDF discretisation then took the form

$$\begin{aligned} &\frac{1}{2\Delta t} [3U_j^{n+1} - 4U_j^n + U_j^{n-1}] = \\ &i \frac{\epsilon \sigma_j^{n+1}}{2\Delta \xi^2} \left[\sigma_{j+1/2}^{n+1} (U_j^{n+1} - U_j^{n+1}) - \sigma_{j-1/2}^{n+1} (U_j^{n+1} - U_{j-1}^{n+1}) \right] \\ &+ 2 \left[\frac{i}{\epsilon} |U_j^n|^2 U_j^n + \dot{X}_j^n \frac{U_{j-2}^n - 8U_{j-1}^n + 8U_{j+1}^n - U_{j+2}^n}{X_{j-2}^n - 8X_{j-1}^n + 8X_{j+1}^n - X_{j+2}^n} \right] \\ &- \left[\frac{i}{\epsilon} |U_j^{n-1}|^2 U_j^{n-1} + \dot{X}_j^{n-1} \frac{U_{j-2}^{n-1} - 8U_{j-1}^{n-1} + 8U_{j+1}^{n-1} - U_{j+2}^{n-1}}{X_{j-2}^{n-1} - 8X_{j-1}^{n-1} + 8X_{j+1}^{n-1} - X_{j+2}^{n-1}} \right], \end{aligned}$$

with $\sigma = 1/x_\xi$. The resulting moving mesh method was then found to be highly effective in computing the focusing solutions.

5.3. Some problems with moving fronts

A classical problem leading to the formation of sharp fronts is Burgers' equation given (in two-dimensions) by

$$u_t + \frac{1}{2}(u^2)_x + \frac{1}{2}(u^2)_y = \nu \Delta u, \quad \nu \ll 1. \quad (5.31)$$

This equation has been used as a benchmark for a number of different moving mesh algorithms (Mackenzie and Mekwi 2007), (Zhang and Tang 2002), (Tang and Xu 2007), (Li, Petzold and Ren 1998).

Example 1 As a first calculation we consider a problem with the initial and boundary values chosen over the unit square so that (5.31) has the exact solution given by

$$u(x, y, t) = \left(1 + e^{(x+y-t)/2\nu}\right)^{-1}. \quad (5.32)$$

This solution has a sharp moving front. For the purposes of this example we consider coupling this system to the PMA algorithm described in Chapter 3, to generate a moving mesh which can both compute an approximation to this solution and follow the front as it evolves over the time interval $t \in [1/4, 2]$. To do this we discretise (5.31) in the Lagrangian form

$$u_t = \nu \Delta u - \left(\frac{1}{2}(u^2)_x + \frac{1}{2}(u^2)_y\right) + \dot{x}u_x + \dot{y}u_y, \quad \nu \ll 1. \quad (5.33)$$

with all discretisations made in the computational variables. The conservation form of the equation above is used for this calculation so that, for example, the advective term u_x^2 is rescaled as

$$u_x^2 = \frac{1}{J} [y_\eta u_\xi^2 - y_\xi u_\eta^2],$$

which is then discretised using a central difference scheme. For this calculation we also take similar discretisations for the other advective terms. The equation posed in the computational variables is then coupled to the discretised form of PMA equation (5.17) with the values of the approximation $U_{i,j}$ to $u(X_i, Y_j)$ and of Q given on the mesh vertices. In these computations we take an $N \times N$ computational mesh (typically $N = 40$), a viscosity of $\nu = 0.005$ and use the arc-length monitor function

$$M = \sqrt{1 + \alpha (u_x^2 + u_y^2)}$$

with $\alpha = 1$. A number of different strategies could be used to evolve this

Figure 5.7. Example 1. The solution of Burgers' equation and the corresponding mesh at the time $t = 1$.

coupled system forward in time, but in practice a simple scheme which solved the PMA equation and (5.33) *simultaneously* using a simple Forward Euler method is effective with a time-step Δt as given in (Zhang and Tang 2002) determined by the CFL condition. In the PMA equation we used $\epsilon = 1, \gamma = 0.335$ to find the initial mesh (before evolving the solution PDE) when $t = 1/4$ and then $\epsilon = 0.01, \gamma = \sqrt{\max(M)}$ to follow the front up to $t = 2$. More details are given in (Walsh, Budd and Williams 2009). We note also that solving the PMA equation coupled to (5.33) using an alternating solution strategy coupled with an up-winding discretisation, has also been shown to be effective (Mohamed 2008). In Fig. 5.3 we present the results of a series of computations using this method, when $N = 40$, showing both the solution for (5.31) and the corresponding mesh. In this figure we see excellent resolution of the solution at the front.

A quantitative measure of the error in the computed solution at $t = 1$ can be given by determining the L_2 -norm of the difference between it and the exact solution. We consider this error both for a uniform and a moving mesh for various values of N

N	L_2 -error on a uniform mesh	L_2 -error on the moving mesh
20	6e-2	8e-3
40	2e-2	2e-3
80	6e-3	1e-3

In (Mohamed 2008) a similar table is obtained by using a PMA method in which the convective terms are calculated by using an upwind method and the PMA system and the underlying PDE solved alternately. It is also interesting to compare this calculation with the results presented in (Zhang and Tang 2002). In this paper a Harmonic mapping method of the form described in Chapter 4 was used to generate the mesh, and this was then coupled to a finite volume discretisation of (5.31.) We see from the table that a useful reduction in the error for calculating the solution to (5.31) has been by using the PMA method in this case. This reduction is similar to that observed in (Zhang and Tang 2002) for using the Harmonic Map method.

Example 2 In closely related calculation we can compare these calculations with a similar moving mesh calculation of the solution of Burgers' equation

Figure 5.8. The computed solution of Burgers' equation in one-dimension at times $t = 0, 0.05, 0.2, 1, 1.4, 2$.

Figure 5.9. The motion of the mesh points when solving Burgers' equation in one-dimension.

over the interval $x \in [0, 1]$ when $\nu = 1e - 4$ and with initial data $u_0 = \sin(2\pi x) + 1/2 \sin(\pi x)$. In this case we compute the mesh using a one-dimensional form of PMA with the arc-length monitor function and $N = 30$ mesh points, central differencing in the computational domain for all spatial derivatives, and solve the resulting ODEs using the Matlab routine `ode15s`. We start with an initially uniform mesh and evolve the mesh and solution together using PMA with $\epsilon = 1$. In this calculation and with this choice of ϵ , the mesh evolves from being uniform to one which is equidistributed, over a time $t \approx 0.05$. In this time period the underlying solution remains fairly smooth. At the time $t \approx 0.2$ the solution develops a sharp front which is well approximated, and then followed, by the evolving mesh. The solution is presented in the physical domain at a series of different times Fig. 5.3. and the resulting mesh trajectories are presented in Fig. 5.3. Observe the manner in which the mesh points resolve the front with no oscillations in this case (due in part to the regularity of the initial data).

It is interesting to compare this calculation with a very similar calculation made by Li and Petzold (1997) who looked at the solution of Burgers' equation when $\nu = 1e - 4$ with the less regular initial data given by

$$\begin{aligned} u(x, 0) \equiv u_0 &= 0.2, & x \leq 0.1, & & u_0 = 8x - 0.6, & 0.1 \leq x \leq 0.2 \\ u_0 &= 1, & 0.2 \leq x \leq 0.5, & & u_0 = -10x + 6, & 0.5 \leq x \leq 0.6, \\ u_0 &= 0, & 0.6 \leq x \leq 1. & & & \end{aligned}$$

This is a more severe test of the moving mesh method than the previous example as the initial data is much less smooth. This calculation employed an arc-length based monitor function together a regularised differential algebraic formulation of the moving mesh equations, very similar to using MMPDE6, and based on a method proposed in Adjerd and Flaherty (1986) with the mesh smoothing algorithm proposed by Dorfi and Drury (1987) described in Chapter 2. In Li and Petzold (1997) three different discretisations were considered, a central difference discretisation an ENO (Roe) method and a piecewise hyperbolic method (PHM) due to Marquina (1994). It was

found that in this case the central difference method tended to lead to spurious oscillations due (as commented on in the example of the nonlinear Schrödinger equation) to the destabilising effect of the anti-diffusive terms arising in the discretisation of the advective terms describing the mesh motion. In Li and Petzold (1997) it was shown that these oscillations were reduced with the ENO scheme and eliminated using the PHM method.

The related paper by Li, Petzold and Ren (1998) also considered applying the same moving mesh method as above to a number of other reaction diffusion problems including the Fisher equation

$$u_t = \beta u_{xx} + \alpha u(1 - u). \quad (5.34)$$

This equation has an exact solution with a moving front given by

$$u = (1 + \exp(\sqrt{\alpha/6}x - 5\alpha t/6))^{-2}.$$

Whilst it was found that the wave front was well resolved, the moving mesh method appeared to give worse results than for the fixed grid, with the front moving at the wrong speed (too fast). It was proposed in (Li, Petzold and Ren 1998) that this error was largely due to the effects of discretising the additional convective terms due to the moving mesh as discussed in Chapter 3 which can lead both to a high truncation error and instabilities in the solution. A possible solution is to use higher order methods or a curvature dependent monitor function. However, there is an interesting other possibility for the error. In problems such as (5.34) the speed of motion of the front is not so much determined by the shape of the front itself but by the nature of the exponential terms in the tails of the solution and how they interact with the boundaries. Similar behaviour arises in the Cahn-Hilliard equation and also in similar systems such as the Gray-Scott equation in chemistry. Somewhat paradoxically the solution needs to be well resolved in the boundary regions where it appears to be behaving smoothly in order for the front speed to be accurately resolved. Thus a uniform mesh may well perform better than a moving mesh in these cases as it will probably be placing more mesh points close to the boundary in these cases.

5.4. Problems involving a change of phase and/or combustion

A very rich set of examples of the use of moving mesh methods is given by Stefan-type problems involving phase changes and/or combustion, and examples of these can be found in Beckett, Mackenzie and Robertson (2001), Mackenzie and Mekwi (2007), Mackenzie and Robertson (2002), Miller, Gleyzer and Imhoff (1998), Huang and Zhan (2004), Zegeling (2005), (Tan 2007), Tan, Lim and Khoo (2007).

As an example we consider the combustion problem described in Cao, Huang

and Russell (1999) and Moore and Flaherty (1992). The mathematical model is a system of coupled nonlinear reaction-diffusion equations

$$\begin{aligned}\frac{\partial u}{\partial t} - \nabla^2 u &= -\frac{R}{\alpha\delta} u e^{\delta(1-1/T)}, \\ \frac{\partial T}{\partial t} - \frac{1}{Le} \nabla^2 T &= \frac{R}{\delta Le} u e^{\delta(1-1/T)},\end{aligned}$$

where u and T represent the dimensionless concentration and temperature of a chemical which is undertaking a one-step reaction. We consider the J -shape solution domain shown in Fig. 5.10. The initial and boundary conditions are

$$\begin{aligned}u|_{t=0} = T|_{t=0} &= 1, & \text{in } \Omega, \\ u|_{\partial\Omega} = T|_{\partial\Omega} &= 1, & \text{for } t > 0\end{aligned}$$

and the physical parameters are set to be $Le = 0.9, \alpha = 1, \delta = 20$, and $R = 5$. A resulting calculation obtained with a MMPDE finite element method as described in Chapter 3 (with details in Cao, Huang and Russell (1999)) is shown in Fig. 5.10.

Figure 5.10. The contour plot of the temperature T (where white represents 2.2 and black represents 1) and the moving meshes are shown at various times.

5.5. Convection driven problems in meteorology

A system of interest to meteorologists is the *Eady problem* which describes the evolution of (localised) extra-tropical (mid-latitude) cyclones. The Eady problem is a two-dimensional reduction of the Euler equations describing the (incompressible) air velocity (u, w) and pressure P in the $(x - z)$ plane, where $x \in [-L, L]$ is a horizontal coordinate along lines of constant Latitude, $z \in [0, H]$ represents height. (A shallow atmosphere model is used with $H \ll L$ together with an f-plane approximation which uses a locally flat approximation to the Earth's curvature.) This model also includes both the potential temperature θ (relative to a reference temperature θ_0) and the effects of Coriolis forces. It is described in detail in (Cullen 2006) and takes the form

$$\begin{aligned}\frac{Du}{Dt} - fv + P_x &= 0, \\ \frac{Dv}{Dt} + fu - \frac{Cg}{\theta_0}(z - H/2) &= 0,\end{aligned}$$

Figure 5.11. The (x, z) contours of the longitudinal velocity and potential temperature of the Eady problem close to the formation of a tropical storm.

$$\begin{aligned}\frac{D\theta}{Dt} - Cv &= 0, \\ \frac{Dw}{Dt} + P_z - \frac{g\theta}{\theta_0} &= 0, \\ u_x + w_z &= 0.\end{aligned}$$

Here D is the advective (total) derivative, f is the Coriolis parameter (assumed constant), g is the gravitational constant and $C = -\theta_y$ is assumed constant. All variables are periodic in x , with w and P_x vanishing at $z = 0$ and $z = H$. From certain initially smooth data (as described by ?) it is possible (Cullen 2006) for the solutions of the Eady problem to develop severe fronts in a small number of days, and some sort of adaptive mesh is needed to resolve the fine structure of the solution close to the fronts. In Fig. 5.5 we present the solution to the Eady problem close to the formation of a severe tropical storm. obtained by using a pressure correction method on a semi-staggered grid, looking at the contours in the horizontal and vertical coordinates (x, z) of the longitudinal v wind speed and potential temperature θ .

We consider two different monitor functions coupled to PMA to find adaptive meshes for this problem. In the first case we take the arc-length monitor function

$$M_1 = \sqrt{1 + |\nabla v|^2}.$$

In the second case we take the monitor function M_2 to be an estimate of the *potential vorticity* q of the solution, so that M_2 is taken to be the maximum eigenvalue of the matrix

$$Q = \begin{pmatrix} v_x + f & v_z \\ \theta_x & \theta_z \end{pmatrix},$$

for which $q = \det(Q)$. The resulting meshes are shown in Figure 5.5. In both cases we see well structured and regular meshes with good resolution at the boundaries and of the front, and with no evidence of mesh tangling. However use of the potential vorticity monitor function M_2 leads to a mesh which more precisely follows the physical solution. Further details are given in (Walsh, Budd and Williams 2009).

Figure 5.12. Two meshes adapted to the solution of the Eady problem presented above. In the first figure we use the arc-length monitor function M_1 and in the second the potential vorticity monitor function M_2 .

5.6. Image processing

The inherent geometry implicit in calculating a mesh is linked very naturally to many PDE problems arising in image processing. Indeed, the optimal transport algorithms described in Chapter 3 is similar in many ways to methods used for image registration. It is thus appropriate to conclude this article with a description of some basic algorithms used in image processing and how they relate to moving methods. A review of PDE methods applied to image processing is given in Sapiro (2003). See also the recent reviews of image registration in Zitova and Flusser (2003), Mohamed (2008).

- Image registration

Image registration (or image warping) is the problem of mapping a (reference) image onto a (target) image. Typically this might involve geometrically aligning two images which may represent the same scene but taken from different viewpoints and/or in different conditions with different sensing devices. It is an essential technique in remote sensing, medical imaging and computer vision (Zitova and Flusser 2003). Such mapping methods aim to preserve and/or align specified features in the reference and target images. See ? for a direct link to optimal transport methods.

- Edge detection
- Denoising
- Morphing

REFERENCES

- S. Adjerid and J.E. Flaherty (1986), A moving finite element method with error estimation and refinement for one-dimensional time dependent partial differential equations, *SIAM J. Numer. Anal.*, **23**, 778–795.
- M. Ainsworth and J.T. Oden (2000), A-posteriori error estimation in finite element analysis. *Pure and Applied Mathematics (New York)*. Wiley-Interscience.
- V.F. Almeida, (1999) Domain Deformation Mapping: Application to Variational Mesh Generation. *SIAM J. Sci Comput.*, **20**, 1252–1275.
- D.A. Anderson and M.M. Rai (1983), The use of solution adaptive grids in solving partial differential equations, *In J.H. Thompson, editor Numerical Grid Generation*, 317–338.
- V. B. .Andreev and N. B. Kopteva (1998), On the convergence, uniform with respect to a small parameter, of monotone three-point difference approximations, *Diff. Urav.*, **34**, 921–929.

- I. Babuska and W.C. Rhienboldt (1979), Analysis of optimal finite element meshes in R^1 , *Math. Comp.*, **33**, 435–463.
- M. J. Baines (1994), *Moving Finite Elements*, Clarendon Press, Oxford.
- M. J. Baines, M. E. Hubbard, and P. K. Jimack (2005), A moving mesh finite strategy for the adaptive solution of time-dependent partial differential equations with moving boundaries, *Appl. Numer. Math.*, **54**, 450–469.
- M. J. Baines, M. E. Hubbard, P. K. Jimack, and A. C. Jones (2006), Scale-invariant moving finite elements for nonlinear partial differential equations in two dimensions, *Appl. Numer. Math.*, **56**, 230–252.
- M. J. Baines and S. L. Wakelin (1991), Equidistribution and the Legendre transformation, *Univ. of Reading, Numer. Anal. Rep.*, **4/91**
- M. L. Balinski (1986), ‘A competitive (dual) simplex method for the assignment problem’, *Math. Programming* **34**(2), 125–141.
- R.E. Bank and R.K. Smith (1997), Mesh smoothing using a posteriori error estimates, *SIAM J. Numer. Anal.*, **34**, 979–997.
- G.I. Barenblatt (1996), *Scaling, self-similarity and intermediate asymptotics*, Cambridge
- G. Beckett, J. A. Mackenzie and M.L. Robertson (2001), A moving mesh finite element method for the solution of two-dimensional Stefan problems. *J. Comput. Phys.*, **186**, 500–518.
- G. Beckett and J. A. Mackenzie (2000), Convergence analysis of finite difference approximations on equidistributed grids to a singularly perturbed boundary value problem, *Appl. Num. Math.*, **35**, 87–109.
- J. D. Benamou and Y. Brenier (2000), A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem, *Numer. Math.*, **84**.
- M. Berger and R. Kohn (1988), A rescaling algorithm for the numerical calculation of blowing-up solutions, *Comm. Pure. Appl. Math.*, **41**, 841–863.
- M. Berzins (1998), A solution-based triangular and tetrahedral mesh quality indicator, *SIAM J. Sci. Comput.*, **19**, 2051–2060.
- P. Bochev, G. Liao and G.d. Pena (1996), Analysis and computation of adaptive moving grids by deformation. *Numer. Meth. PDEs*, **12**, 489–506.
- C. de Boor (1973), *Good approximations by splines with variable knots II*. Springer Lecture note series, **363**, Berlin.
- J.U. Brackbill (1993), An adaptive grid with directional control, *J. Comput. Phys.*, **108**, 38–50.
- J.U. Brackbill and J.S. Saltzman (1982), Adaptive zoning for singular problems in two-dimensions *J. Comput Phys.*, **46**, 342–368.
- Y. Brenier (1991), Polar factorization and monotone rearrangement of vector-valued functions, *Comm. Pure Appl. Math.*, **44**, 375–417.
- C.J. Budd, S-N. Chen and R.D.R. Russell (1999), New self-similar solutions of the nonlinear Schrödinger equation, with moving mesh computations, *J. Comput. Phys.*, **152**, 756–789.
- C. J. Budd, R. Carretero-Gonzalez, and R. D. Russell (2005), Precise computations of chemotactic collapse using moving mesh methods, *J. Comput. Phys.*, **202**, 462–487.
- C.J. Budd, G.J. Collins, W.-Z. Huang and R.D. Russell (1999), Self-similar discrete

- solutions of the porous medium equation, *Philos. Trans. Roy. Soc. Lond, A*, **357**, 1047–1078.
- V.A. Dorodnitsyn and C.J. Budd (2001), Symmetry adapted moving mesh schemes for the nonlinear Schrödinger equation, *J. Phys. A.*, **34**, 103887–10400.
- C. J. Budd, W. Z. Huang, and R. D. Russell (1996), Moving mesh methods for problems with blow-up, *SIAM J. Sci. Comput.*, **17**, 305–327.
- C. J. Budd, B. Leimkuhler, and M. D. Piggott (2001), Scaling invariance and adaptivity, *Appl. Numer. Math.*, **39** 261–288.
- C. J. Budd and M. D. Piggott (2005), Geometric integration and its applications, *in handbook of Numerical Analysis ed. F. Cucker*.
- C. J. Budd, M. D. Piggott, and J. F. Williams (2008), Adaptive numerical methods and the geostrophic coordinate transformation, *Submitted to Monthly Weather Review*.
- C. J. Budd and J. F. Williams (2006), Parabolic Monge-Ampère methods for blow-up problems in several spatial dimensions, *J. of Physics A*, **39**, 5425–5444.
- C. J. Budd and J. F. Williams (2008), Mesh generation using the Parabolic Monge-Ampère method, submitted.
- L. A. Caffarelli (1992), The regularity of mappings with a convex potential, *J. Am. Math. Soc.*, **5**, 99–104.
- L. A. Caffarelli (1996), Boundary regularity of maps with convex potentials, *Ann. of Math.*, **3**, 453–496.
- X. Cai, D. Fleitas, B. Jiang and G. Liao (2004), Adaptive grid generation based on the least squares finite element method, *Comput. Math. Appl.*, **48**, 1077–1085.
- D.A. Calhoun, C. Helzel and R.J. LeVeque (2008), *SIAM Review*, **50**, 723–752.
- W. Cao (2005), On the error of linear interpolation and the orientation, aspect ratio and internal angles of a triangle, *SIAM J. Numer. Anal.*, **43**, 19–40.
- W. Cao (2007), An Interpolation Error Estimate on Anisotropic Meshes in R^n and Optimal Metrics for Mesh Refinement, *SIAM J. Numer. Anal.*, **45**, 2368–2391
- W. Cao (2007), Anisotropic Measures of Third Order Derivatives and the Quadratic Interpolation Error on Triangular Elements *SIAM J. Sci. Comput.*, **29**, 756–781.
- W. Cao, W. Huang, and R. D. Russell (1999a), An r-adaptive finite element method based upon moving mesh pdes, *J. Comput. Phys.*, **149**, 221–244.
- W. Cao, W. Huang, and R. D. Russell (1999b), A study of monitor functions for two-dimensional adaptive mesh generation, *SIAM J. Sci. Comput.*, **20**, 1978–1994.
- W. Cao, W. Huang, and R. D. Russell (2002), A moving mesh method based on the geometric conservation law, *SIAM J. Sci. Comput.*, **24**, 118–142.
- W. Cao, W. Huang, and R. D. Russell (2003), Approaches for generating moving adaptive meshes: location versus velocity, *Appl. Numer. Math.*, **47**, 121–138.
- G. Carey (1997), Computational grids: Generation, adaptation and solution strategies, *Taylor and Francis*.
- N. Carlson and K. Miller (1998a), Design and application of a gradient-weighted moving finite element code, Part I, in 1-D, *SIAM J. Sci. Comput.*, **19**, 728–765.
- N. Carlson and K. Miller (1998b), Design and application of a gradient-weighted

- moving finite element code, Part II, in 2-D, *SIAM J. Sci. Comput.*, **19**, 766–798.
- H. D. Ceniceros (2002), A Semi-implicit moving mesh method for the focusing non-linear Schrödinger equation, *Communications on pure and applied analysis*, **4**, 1–14.
- H. D. Ceniceros and T. Y. Hou (2001), An efficient dynamically adaptive mesh for potentially singular solutions, *J. Comput. Phys.*, **172**, 609–639.
- R. Chartrand, K. R. Vixie, B. Wohlberg and E. M. Bollt (2007), A gradient descent solution to the monge-kantorovich problem, <http://math.lanl.gov/Research/Publications/Docs/chartrand-2007-gradient.pdf>.
- L. Chen, P. Sun and J. Xu (2007), Optimal anisotropic meshes for minimising interpolation errors in the L^p -norm, *Math. Comput.*, **76**, 179–204.
- S. Chynoweth and M. J. Baines (1989), Legendre transform solutions to semi-geostrophic frontogenesis, in *Finite element analysis in fluids*, T. J. Chung and G. R. Kerr, eds., 697–703.
- S. Chynoweth and M. J. Sewell (1989), Dual variables in semigeostrophic theory, *Proc. R. Soc. Lond. A*, **424**, 155–186.
- M. J. P. Cullen (1989), Implicit finite difference methods for modelling discontinuous atmospheric flows, *J. Comput. Phys.*, **81**, 319–348.
- M. J. P. Cullen (2006), A Mathematical Theory of Large-Scale Atmosphere/Ocean Flow, *Imperial College Press*.
- M. J. P. Cullen, J. Norbury, and R. J. Purser (1991), Generalised Lagrangian solutions for atmospheric and oceanic flows, *SIAM J. Appl. Math.*, **51**, 20–31.
- M. J. P. Cullen and R. J. Purser (1984), An extended Lagrangian theory of semi-geostrophic frontogenesis, *J. Atmos. Sci.*, **41**, 1477–1497.
- B. Dacorogna and J. Moser (1990), On a partial differential equation involving the Jacobian determinant, *Ann. Inst. Henri Poincaré Analyse non lineaire*, **7**, 1–26.
- E. Dean and R. Glowinski (2003), ‘Numerical solution of the two-dimensional elliptic Monge–Ampère equation with Dirichlet boundary conditions: an augmented Lagrangian approach’, *Comptes rendus-Mathématique* **336**(9), 779–784.
- E. Dean and R. Glowinski (2004), ‘Numerical solution of the two-dimensional elliptic Monge–Ampère equation with Dirichlet boundary conditions: a least-squares approach’, *Comptes rendus-Mathématique* **339**(12), 887–892.
- G. Delzanno, L. Chacón, J. Finn, Y. Chung and G. Lapenta (2008), ‘An optimal robust equidistribution method for two-dimensional grid adaptation based on Monge–Kantorovich optimization’, *J. Comput. Physics* **227**(23), 9841 – 9864.
- Y. Di, R. Li, T. Tang and P. Zhang (2005), Moving mesh finite element methods for the incompressible Navier-Stokes equations, *SIAM J. Sci. Comput.*, **26**, 1036–1056.
- V.A. Dorodnitsyn (1991), Transformation groups in mesh spaces, *J. Sov. Math*, **55**, 1490–1517.
- V.A. Dorodnitsyn (1993a), Finite-difference models exactly inheriting symmetry of

- original differential equations, *In: N. Ibragimov et. al. editors. Modern Group Analysis: Advanced Analytical and Computational Methods in Mathematical Physics.*, **Kluwer (Dordrecht)**, 191–201.
- V.A. Dorodnitsyn (1993b), Finite difference analog of the Noether theorem, *Dokl. Akad. Nauk*, **328**, 678–690.
- V.A. Dorodnitsyn and R. Kozlov (1997), The whole set of symmetry preserving discrete versions of a heattransfer equation with a source. *Preprint 4/1997, NTNU, Trondheim*.
- E.A. Dorfi and L. O’c Drury (1987), Simple adaptive grids for 1-D initial value problems, *J. Comput. Phys.*, **69**, 175–195.
- A.S. Dvinsky (1991), Adaptive grid generation from harmonic maps on Riemannian manifolds, *J. Comput. Phys.*, **95**, 450–476.
- P.R. Eisman (1985), Grid generation for fluid mechanics computation, *Ann. Rev. Fluid Mech.*, **17**, 487–522.
- P.R. Eisman (1987), Adaptive grid generation, *Comput. Meth. Appl. Mech. Engrg.* **64**, 321–376.
- L. C. Evans (1999), Partial differential equations and Monge-Kantorovich mass transfer, in *Current developments in mathematics, 1997 (Cambridge, MA)*, Int. Press, Boston, MA, pp. 65–126.
- X. Feng and M. Neilan (2008), ‘Vanishing moment method and moment solutions for fully nonlinear second order partial differential equations’, *Journal of Scientific Computing*. DOI:10.1007/s10915-008-9221-9.
- S. Fulton (1989), ‘Multigrid Solution of the Semigeostrophic Invertibility Relation’, *Monthly Weather Review* **117**(9), 2059–2066.
- W. Gangbo and R. J. McCann (1996), The geometry of optimal transport, *Acta Math.*, **177**, 113–161.
- C. E. Gutiérrez (2001), *The Monge-Ampère equation*, Progress in Nonlinear Differential Equations and their Applications, 44, Birkhäuser Boston Inc., Boston, MA.
- S. Haker and A. Tannenbaum (2003), On the Monge-Kantorovich problem and image warping, in *Mathematical methods in computer vision*, Vol. 133 of *IMA Vol. Math. Appl.*, Springer, New York, pp. 65–85.
- D.F. Hawken, J.J. Gottlieb and J.S. Hansen (1991), Review of some adaptive node-movement techniques in finite element and finite difference solutions of PDEs, *J. Comput. Phys.*, **95**, 254–302.
- Y. He and W. Huang (2008), A posteriori error analysis for finite element solution of elliptic differential equations using equidistributing meshes, *submitted*
- W. Huang (2001), Practical aspects of formulation and solution of moving mesh partial differential equations, *J. Comput. Phys.*, **171**, 753–775.
- W. Huang (2001), Variational mesh adaption: isotropy and equidistribution, *J. Comput. Phys.*, **174**, 903–924.
- W. Huang (2005), Measuring mesh qualities and application to variational mesh adaption, *SIAM J. Sci. Comput.*, **26**, 1643–1666.
- W. Huang (2005), Metric tensors for anisotropic mesh generation, *J. Comput. Phys.*, **204**, 663–665.

- W. Huang (2007), Anisotropic mesh adaption and movement, *in Adaptive Computations: Theory and Algorithms*, eds. T.Tang and J.Xu, Science Press, Beijing.
- W. Huang and B. Leimkuhler (1997), The adaptive Verlet method, *SIAM J. Sci. Comput.*, **18**, 239–256.
- W. Huang, Y. Ren, and R. D. Russell (1994), Moving mesh partial differential equations (MMPDES) based on the equidistribution principle, *SIAM J. Numer. Anal.*, **31**, 709–730.
- W. Huang and R. D. Russell (1996), A moving collocation method for solving time dependent partial differential equations, *Appl. Numer. Math.*, **20**, 101–116.
- W. Huang and R. D. Russell (1997), Analysis of moving mesh partial differential equations with spatial smoothing, *SIAM J. Numer. Anal.*, **34**, 1106–1126.
- W. Huang and R. D. Russell (1999a), A high dimensional moving mesh strategy, *Appl. Numer. Math.*, **26**, 63–76.
- W. Huang and R. D. Russell (1999b), A moving mesh strategy based on a gradient flow equation for two-dimensional problems, *SIAM J. Sci. Comput.*, **20**, 998–1015.
- W. Huang and R. D. Russell (2001) Adaptive mesh movement — the MMPDE approach and its applications, *J. Comp. Appl. Math.*, **128**, 383–398.
- W. Huang and D. Sloan (1994), A simple adaptive grid method in two dimensions, *SIAM J. Sci. Comp.*, **15**, 776–797.
- W. Huang and W. Sun (2003), Variational mesh adaption II: error estimates and monitor functions. *J. Comput. Phys.*, **184**, 619–648.
- W. Huang and X. Zhan (2004), Adaptive moving mesh modeling for two dimensional groundwater flow and transport. *In: Recent advances in adaptive computation. AMS Contemporary Mathematics*, **383**, 283–296.
- O.-P. Jacquotte (1988), A mechanical model for a new grid generation method in computational fluid dynamics, *Computer Methods in Applied Mechanics and Engineering*, **66**, 323–338.
- O.-P. Jacquotte and G. Coussement(1992), Structured mesh adaption: space accuracy and interpolation methods, *Computer Methods in Applied Mechanics and Engineering* **101**, 397–432.
- C. Johnson (1987), Numerical solution of partial differential equations by the finite element method, *Cambridge*
- T. Kaijser (1998), ‘Computing the Kantorovich distance for images’, *J. Math. Imaging Vision* **9**(2), 173–191.
- J. Kautsky and N.K. Nichols (1980), Equidistributing Meshes with Constraints, *SIAM J. Sci. Stat. Comput.*, **1**, 499–511.
- J. Kautsky and N.K. Nichols (1982), Smooth regridding of discretized data, *SIAM J. Sci. Stat. Comput.*, **3**, 145–159.
- P.M. Knupp (1996), Jacobian-weighted elliptic grid generation, *SIAM J. Sci. Comp.*, **17**, 1475–1490.
- P.M. Knupp (2001), Algebraic Mesh Quality Metrics, *SIAM J. Sci. Comput.*, **23**, 193–218.
- P.M. Knupp, L. Margolin and M. Shashkov (2002), Reference Jacobian optimization-based rezoning strategies for arbitrary Lagrangian Eulerian methods, *J. Comput. Phys.*, **176**, 93–128.

- P. Knupp and S. Steinberg (1994), Fundamentals of Grid Generation (1994), *CRC Press, Boca Raton*.
- N. Kopteva (2007), Convergence theory of moving grid methods, in *Adaptive Computations: Theory and Algorithms*, eds. T.Tang and J.Xu, Science Press, Beijing.
- N. Kopteva and M. Stynes (2001), A robust adaptive method for a quasilinear one-dimensional convection-diffusion problem, *SIAM J. Numer. Anal.*, **39**, 146–1467.
- R. Kozlov (2000), Symmetry applications to difference and differential-difference equations. *PhD Thesis, Institut for matematiske fag, NTNU, Trondheim*.
- J. Lang, W. Cao, W. Huang and R.D. Russell (2003), A two-dimensional moving finite element method with local refinement based on a posteriori error estimates, *Appl. Numer. Math.*, **46**, 75–94.
- R. Li, T. Tang, and P.-W. Zhang (2002), A moving mesh finite element algorithm for singular problems in two and three space dimensions, *J. Comput. Phys.*, **177**, 365–393.
- S.T. Li and L.R. Petzold (1997), Moving mesh methods with upwinding schemes for time dependent PDEs, *J. Comput Phys.*, **131**, 368–377.
- S.T. Li, L.R. Petzold and Y. Ren (1998), Stability of moving mesh systems of partial differential equations, *SIAM J. Sci. Comput.*, **20**, 719–738.
- G. Liao and D. Anderson (1992), A new approach to grid generation, *Appl. Anal.*, **44**, 285–297.
- V.D. Liseikin (1999), Grid generation methods, *Springer, Berlin*.
- A. Liu and B. Joe (1994a), Relationship between tetrahedron quality measures, *BIT*, **34**, 268–287.
- J. A. Mackenzie and W. R. Mekwi (2007), On the use of moving mesh methods to solve PDEs, in *Adaptive Computations: Theory and Algorithms*, eds. T.Tang and J.Xu, Science Press, Beijing.
- J. A. Mackenzie and M. L. Robertson (2002), A moving mesh method for the solution of the one-dimensional phase-field equations., *J. Comput. Phys.*, **181**, 526–544.
- A. Marquina (1994), Local piecewise hyperbolic resolution of numerical fluxes for nonlinear scalar conservation laws, *SIAM J. Sci. Comput.*, **15**, 894–904.
- R. I. Machlachlan (1994), Symplectic integration of Hamiltonian wave equations, *Numer. Math.*, **66**, 465–492.
- C. T. Miller, S. N. Gleyzer and P. T. Imhoff (1998), Numerical modeling of NAPL dissolution fingering in porous media, *In: Physical nonequilibrium in soils modeling and application*, Eds. H. M. Selim and L. Ma, Ann Arbor Press.
- K. Miller and R.N. Miller (1981), Moving finite elements I, *SIAM J. Numer. Anal.*, **18**, 1019–1032.
- K. Miller (1981), Moving finite elements II, *SIAM J. Numer. Anal.*, **18**, 1033–1057.
- M.H. Sulman (2008) Optimal mass transport for adaptivity and image registration, PhD Thesis, SFU.
- P. K. Moore and J. E. Flaherty (1992), Adaptive local overlapping grid methods for parabolic system in two space dimensions, *J. Comput. Phys.*, **98**, 54–63.
- J. Moser (1965), On the volume elements of a manifold, *Trans AMS*, **120**, 286–294.

- V. I. Oliker and L. D. Prussner (1988), ‘On the numerical solution of the equation $(\partial^2 z/\partial x^2)(\partial^2 z/\partial y^2) - ((\partial^2 z/\partial x\partial y))^2 = f$ and its discretizations. I’, *Numer. Math.* **54**(3), 271–293.
- P.J. Olver (1986), Applications of Lie groups to differential equations, *New York, Springer*
- L.R. Petzold (1982), A description of DASSL: A differential/algebraic system solver, *Tech report SAND82-8637, Sandia National Labs, Livermore, CA*
- L.R. Petzold (1987), Observations on an adaptive moving grid method for one-dimensional systems for partial differential equations, *Appl. Numer. Math.*, **3**, 347–360.
- J. Pryce (1989), On the convergence of iterated remeshing, *IMA J. Numer. Anal.*
- S. T. Rachev and L. Rüschemdorf (1998), *Mass transportation problems. Vol. I, Probability and its Applications* (New York), Springer-Verlag, New York. Theory.
- W. Ren and X. Wang (2000), An iterative grid redistribution method for singular problems in multiple dimensions, *J. Comput. Phys.*, **159**, 246–273.
- R. D. Russell, J.F. Williams, and X. Xu (2007), MOVCOL4: A Moving Mesh Code for Fourth-Order Time-Dependent Partial Differential Equations, *SIAM J. Sci. Comput.*, **29**, 197–220
- G. Sapiro (2003), Introduction to partial differential equations and variational formulations in image processing. *In: Foundations of Computational Mathematics. Editor F. Cucker*, **1**, 383–461.
- P. Saucez, A. Vande Vouwer and P.A. Zegeling (2005), Adaptive method of lines solutions for the extended fifth order Korteweg-De Vries equation, *J. Comput. Math.*, **183**, 343–357.
- B. Semper and G. Liao (1995), A moving grid finite-element method using grid deformation. *Numer. Methods in PDEs*, **11**, 603–615.
- M. J. Sewell (1978), On Legendre transformations and umbilic catastrophes, *Math. Proc. Camb. Phil. Soc.*, **83**, 273–288.
- R. Shewchuk (2002), Constrained Delaunay Tetrahedralizations and Provably Good Boundary Recovery, *IMR 2002*, 193–204.
- G.E. Shilov and B.L. Gurevich (1978), Integral measure and derivative: a unified approach. *Dover Publications*.
- J. Stockie, J. A. MacKenzie, and R. D. Russell (2000), A moving mesh method for one-dimensional hyperbolic conservation laws, *SIAM J. Sci. Comput.*, **22**, 1791–1813.
- C. Sulem and P. L. Sulem (1999), The nonlinear Schrodinger equation: self focusing and wave collapse, *Springer-Verlag*.
- Z. Tan (2007), Adaptive moving mesh methods for two- dimensional resistive magneto-hydrodynamic PDE models, *Computers and Fluids*, **36**, 758–771.
- Z. Tan, K.M. Lim and B.C. Khoo (2007), An adaptive mesh redistribution method for the incompressible mixture flows using phase-field model, *J. of Comput. Phys.*, **225**, 1137–1158.
- H. Z. Tang and T. Tang (2003), Adaptive mesh methods for one- and two- dimensional hyperbolic conservation laws, *SIAM J. Numer. Anal.*, **41**, 487–515.
- T. Tang (2005), Moving mesh methods for computational fluid dynamics, *Contemporary mathematics*, **383**, 141–173.

- T. Tang and J. Xu, editors (2007), Adaptive Computations: Theory and Algorithms, *Science Press, Beijing*.
- J. F. Thompson (1985), A survey of dynamically-adaptive grids in the numerical solution of partial differential equations, *Appl. Numer. Math.*, **1**, 3–27.
- J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin (1985), Boundary-fitted coordinate systems for numerical solution of partial differential equations - a review., *J. Comput. Phys.*, **47**, 1–108.
- J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin (1985), Numerical grid generation, *North-Holland*.
- J.F. Thompson and N.P. Weatherill (1992), Structured and unstructured grid generation, *Critical Reviews Biomed. Eng.*, **20**, 73–120.
- Y. Touringy (1995).
- A.E.P. Veldman and K. Rinzema (1992), Playing with nonuniform grids, *J. of Engineering Mathematics*, **26**, 119–130.
- J.G. Verwer, J.G. Blom, R.M. Furzeland and P.A. Zegeling (1989), A moving-grid method for one-dimensional PDEs based on the method of lines, in *Adaptive Methods for Partial Differential Equations*, edited by J.E. Flaherty, P.J. Paslow, M.S. Shepard and J.D. Vasilakis, SIAM.
- C. Villani (2003), Topics in Optimal Transportation, vol. 58 of Graduate Studies in Mathematics, *AMS*.
- E. Walsh, C.J. Budd and J.F. Williams (2009), The PMA method for grid generation applied to the Eady Problem in meteorology, *University of Bath report*.
- A.J. Wathen and M.J. Baines (1985), On the structure of the moving finite-element equations. *IMA J. Numer. Anal.*, **5**, 161–182.
- A. M. Winslow (1967), Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh, *J. Comput. Phys.*, **2**, 149–172.
- A. M. Winslow (1981), Adaptive mesh rezoning by the equipotential method *Technical Report UCID-19062, Lawrence Livermore Lab*.
- X. Xu, W.-H. Huang, R.D. Russell and J.F. Williams (2008), Convergence of de Boor's algorithm for generation of equidistributing meshes, *submitted*
- N.N. Yanenko, E.A. Krosko, V.V. Lisejkin, V.M. Fomin, V.P. Shapev and Y.A. Shitov (1976), Methods for the construction of moving grids for problems of fluid dynamics with big deformations. *Springer Lecture Notes in Physics, Berlin*, **59**
- P.A. Zegeling (1993), Moving-grid methods for time-dependent partial differential equations, *CWI Tract 94*.
- P.A. Zegeling (2005), On resistive MHD models with adaptive moving meshes *J. Sci. Comput.*, **24**, 263–284.
- P.A. Zegeling (2007), Theory and application of adaptive moving grid methods, in *Adaptive Computations: Theory and Algorithms, Science Press, Beijing*.
- P.A. Zegeling and H.P. Kok (2004), Adaptive moving mesh computations for reaction-diffusion systems, *J. Comput. Appl. Math.*, **168**, 519–528.
- Z-R Zhang and T. Tang (2002), An adaptive mesh redistribution algorithm for convection-dominated problems, *Comm. Pure and Applied Analysis*, **1**, 341–357
- B. Zitova and J. Flusser (2003), Image registration methods: a survey, *Image and vision computing*, **21**, 977–1000.

M. Zlamal (1968), On the finite element method, *Numer. Math.*, **12**, 394–409.