Fast three dimensional r-adaptive mesh redistribution

Article

Accepted Version

Fast three dimensional r-adaptive mesh redistribution

P.A. Browne[1,*], C.J. Budd[2], C. Piccolo[3], and M. Cullen[3]

[1]Department of Meteorology, University of Reading, UK
[2]Department of Mathematical Sciences, University of Bath, UK
[3]Met Office, Fitzroy Road, Exeter, EX1 3PB, UK
[*]Correspondence to p.browne@reading.ac.uk

June 24, 2014

Abstract

This paper describes a fast and reliable method for redistributing a computational mesh in three dimensions which can generate a complex three dimensional mesh without any problems due to mesh tangling. The method relies on a three dimensional implementation of the parabolic Monge-Ampère (PMA) technique, for finding an optimally transported mesh. The method for implementing PMA is described in detail and applied to both static and dynamic mesh redistribution problems, studying both the convergence and the computational cost of the algorithm. The algorithm is applied to a series of problems of increasing complexity. In particular very regular meshes are generated to resolve real meteorological features (derived from a weather forecasting model covering the UK area) in grids with over 2×10^7 degrees of freedom. The PMA method computes these grids in times commensurate with those required for operational weather forecasting.

This work was funded by EPSRC EP/H500103/1 Knowledge Transfer Grant - University of Bath.

1 Introduction

1.1 Overview

Many physical problems exhibit a variety of different spatial scales and feature localised small scale structures embedded within a much larger scale geometry. Examples include the boundary layers frequently encountered in fluid mechanics and gas dynamics, meteorological inversion layers [1], weather fronts, combustion layers and shock waves. Computations on such problems using a uniform

computational mesh may encounter problems when the computational mesh size is too large to resolve the small scale structures. When such a computation is part of a computational fluid dynamics (CFD) calculation then this may lead to large truncation errors [2]. In the data assimilation context, an adaptive mesh is a convenient way of representing anisotropic spatially varying correlation structures in a flow dependent manner, which would otherwise be represented by spurious isotropic correlations. It is thus often important, both for accuracy and for computational efficiency, to use a computational mesh which is adapted in some manner to the small scales in the underlying problem. This is relatively easy in one spatial dimension with many excellent examples of successful implementations both in PDE calculations [3] and in data assimilation, [4] leading to significant increases in accuracy and computational efficiency. However, the computational difficulties of (dynamically) adapting a mesh for a three dimensional problem and coupling it to a solver, are considerable [5]. Furthermore, fully three dimensional adapted meshes can take a significant time to generate [6]. In this paper, we will describe an algorithm for *adaptive mesh redistribution* based on optimal transport ideas, which is both fast to implement, avoids mesh tangling and gives excellent three dimensional meshes for some large and challenging problems. We demonstrate the effectiveness of this procedure on a number of problems, including large meteorological calculations based on real data. These methods have the potential for relatively easy coupling to both CFD codes and data assimilation procedures.

1.2 An outline of adaptive mesh redistribution

Broadly speaking adaptive meshes fall into three types. The most commonly used is *Adaptive Mesh Refinement*, AMR or h-adaptivity, in which a structured mesh is locally refined (or possibly de-refined) by the addition (or subtraction) of new mesh points [7] when some local refinement condition is satisfied [8]. This is closely related to p-adaptive methods [9] in which the order of the elements used in the computation is locally increased, again prompted by some local refinement condition. Both of these methods have the advantages of a degree of maturity in implementation and flexibility of use. However they also suffer from various disadvantages. The complex and evolving data structures needed to describe the mesh and its changing connectivity [10] can make it difficult to couple them to other software. Furthermore the very local nature of the mesh refinement, can lead to meshes with poor global structures, without good alignment or regularity. An alternative procedure, a specific version of which is described in this paper, is *Adaptive Mesh Redistribution*, also known as r-adaptivity (or more simply as a moving mesh method). In this procedure a *fixed number* of mesh points in a *constant connectivity structure* is redistributed so that the fine-scale features of interest are best resolved. A powerful method for doing this is to move the points so that the *point density* is controlled by equidistributing an appropriate scalar or matrix *monitor function*. This procedure has certain similarities to Lagrangian methods in which the velocity of the mesh points is coupled to convective features of the underlying solution. However, it avoids the

mesh tangling problems often associated with such methods [11]. Whilst less mature than AMR type methods, adaptive mesh redistribution offers potential advantages. Firstly, the constant data structure makes them straightforward both to use in their own right and to couple to existing software. Secondly, the fact that all of the points in the mesh are calculated together means that both local refinement and global regularity of the mesh can be treated together, leading to potentially very regular meshes. (Indeed it is possible to build a degree of global regularity directly into the implementation of the method [11].) Thirdly, the mesh points can inherit underlying dynamical features of the problem such as symmetries and self-similarity. Various methods for implementing adaptive mesh redistribution of varying levels of complexity include Geometric Conservation Law methods, Harmonic maps, and variational methods. See the reviews in [12], and [13]. All of these methods consider adaptivity in at most two-dimensions. An alternative method based on Optimal Transport ideas is described in [11], [14], [6], [15], and takes a differing approach, coupling equidistribution to global mesh regularity and calculating an appropriate scalar *mesh potential* from which the mesh can be determined. Optimal transport based methods are relatively cheap to implement and have been coupled successfully to computations of incompressible flows in two-dimensions [16], and also to large scale data assimilation calculations [1, 4]. Objections to adaptive mesh redistribution methods include the possibilities of mesh tangling and mesh skewness, leading to elements with small angles and the loss of balance relationships when representing certain fluid motions. Whilst these objections are often valid, it is certainly the case that optimally transported meshes can be computed cheaply, even in three dimensions, they have provable regularity [11],[16], they do not suffer from mesh tangling, the reduction in errors due to improved resolution can outweigh the extra errors given by mesh skewness, and skewness can also be an advantage if it leads to better alignment of the mesh with the underlying solution [17], [13]. Finally the preservation of balance laws can be built into the mesh construction through the construction of the monitor function.

In this paper we show how the optimal transport method, coupled to a simple to implement, and robust, relaxation approach, can be implemented practically to deal with large three dimensional problems with severe geometric distortion. We then test this method on a series of challenging problems including large scale meteorological systems, and we study its convergence in each case. In this implementation the calculation of a three dimensional meteorological grid with 21772800 degrees of freedom could be accomplished in under four minutes on a laptop computer. In principle these meshes can be coupled to data assimilation codes using methods of [1, 4].

The remainder of this paper is structured as follows. In Section 2 we describe some of the underlying theory of r-adaptive mesh redistribution and the optimal transport method of doing this, leading to a single equation (the Monge-Ampère equation) describing the mesh. In Section 3 we describe a relaxation method for solving this equation. In Section 4 we describe a simple, practical and effec-

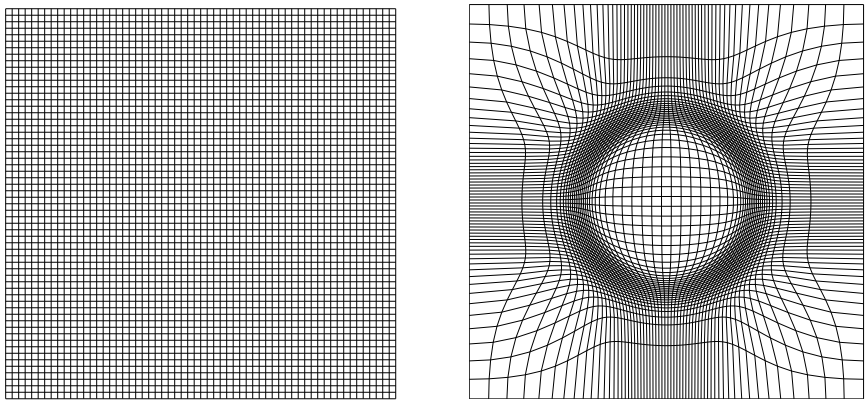
tive method for discretising this equation and calculating a three dimensional mesh. In Section 5 we consider various static mesh redistribution problems including some which use meteorological data from the Met Office UK4 forecast system. Finally in Section 6 we consider an evolving problem with dynamic mesh redistribution.

2 Adaptive mesh redistribution in three dimensions

Adaptive mesh redistribution methods work by keeping the number of mesh points and the topology of the mesh fixed but redistribute the mesh in space. For a time evolving problem the mesh can then evolve with the solution of the underlying problem. The simplest three dimensional mesh $\mathcal{T}_C$ comprises a regular subdivision of the unit cube into identical smaller cubes. We denote the unit cube by $\Omega_C = [0,1]^3$, and it represents a reference or computational space. We can then map the mesh $\mathcal{T}_C$ into any other logically (or topologically) cuboid mesh $\mathcal{T}_P$ occupying a *physical* space $\Omega_P \subset \mathbb{R}^3$, through the map

$$\mathbf{F}(.,t) : \Omega_C \to \Omega_P.$$

The mesh points in $\mathcal{T}_P$ are therefore the images of the corners of the cuboids in $\mathcal{T}_C$ and these points redistribute as the time t evolves. For clarity we define a point in Ω_C by $\xi \in \Omega_C = (\xi, \eta, \zeta)$. Similarly we denote a point $\mathbf{x}$ in the physical space Ω_P by $\mathbf{x} \in \Omega_P = (x, y, z)$. An example of a section of mesh $\mathcal{T}_C$ in Ω_C and a section of its image $\mathcal{T}_P$ in Ω_P is given in Figure 1.



(a) A mesh $\mathcal{T}_C$ in computational space Ω_C, denoted $\xi = (\xi, \eta, \zeta)$

(b) A mesh $\mathcal{T}_P$ in physical space Ω_P, denoted $x = (x, y, z)$

Figure 1: A mesh $\mathcal{T}_C \in \Omega_C$ and its image $\mathcal{T}_P \in \Omega_P$.

For redistribution to be effective we need to concentrate mesh points so that they have a high density in certain regions of Ω_P. The value of this mesh density

is taken to be proportional to the size of a monitor function $m(\mathbf{x},t) > 0$, so that if A is *any* set in Ω_C (such as a small cube) of fixed volume ϵ, and if the image of A in Ω_P is the set $F(A,t)$ then regardless of the location and orientation of A in Ω_C we have

$$\epsilon \equiv \int_A \mathbf{d}\xi = \frac{\int_{F(A,t)} m(\mathbf{x})\ \mathbf{d}x}{\int_{\Omega_P} m(\mathbf{x})\ \mathbf{d}x} = \frac{\int_A m(\mathbf{F}(\xi))|J(\xi)|\ \mathbf{d}\xi}{\int_{\Omega_P} m(\mathbf{x})\ \mathbf{d}x}$$

where $|J(\xi,t)|$ is the determinant of the Jacobian of the map from Ω_C to Ω_P given (in 3 dimensions) by

$$|J(\xi,t)| = \begin{vmatrix} x_\xi & x_\eta & x_\zeta \\ y_\xi & y_\eta & y_\zeta \\ z_\xi & z_\eta & z_\zeta \end{vmatrix}. \tag{1}$$

As this applies for all sets A it follows that the map must satisfy

$$m(\mathbf{x},t)\ |J(\xi,t)| = \int_{\Omega_p} m(\mathbf{x},t)\ \mathbf{d}x. \tag{2}$$

We call this the *equidistribution equation*. Its performance relies on a suitable choice of monitor function, which is often taken to be a measure of the error (eg. interpolation error) made when using the mesh in the calculation of the numerical approximation of the solution to a problem. In one dimension the equidistribution equation uniquely defines the map $\mathbf{F}$ and a number of methods exploit this, most particularly the moving mesh PDE methods listed in [18]. In higher dimensions additional conditions are required to define the map uniquely. Noting that for many computations there are signficant advantages to using a uniform mesh, it makes initial sense to look for meshes which are close to being uniform in some sense. In other words we seek functions $\mathbf{F}$ which are close to the identity in some measure. A convenient such measure is the Wasserstein metric I given by

$$I = \int_{\Omega_C} |\mathbf{F}(\xi,t) - \xi|^2\ d\xi \tag{3}$$

Definition 1. A map $\mathbf{F}$ which minimises I is over all invertible mappings satisfying (2) called an *optimally transported map*. The resulting mesh $\mathcal{T}_P$ is an *optimally transported mesh.*

Finding such a map is an example of a *Monge-Kantorovich problem* (see [19]). Equation (2) defines two measures on real space with ratio $|J|$, one of which is standard Lebesgue measure L. Then the Monge-Kantorovich problem finds the optimal map that pushes forward $|J|L$ to L with the quadratic cost given by (3). Although the condition of minimising I appears to be a coarse global restraint on the mesh $\mathcal{T}_P$, it not only leads to a system which is easy to calculate, but also to meshes with provably excellent regularity, good mesh grading and good

mesh alignment [11], [16], [15]. We now seek to solve the Monge-Kantorovich problem to determine the optimal mesh $\mathcal{T}_P$. The key underlying result which allows us to compute this mesh is the following

Theorem 1 (Brenier [19])**.** *There exists a* **unique** *optimally transported map* $\mathbf{F}(\xi, t)$ *which minimises* I*, and the Jacobian of which satisfies the equidistribution equation (2). This map has the same regularity as the monitor function* m*. Furthermore,* $\mathbf{F}(\xi, t)$ *can be written as the gradient (with respect to* ξ*) of a convex scalar (mesh) potential* $P(\xi, t)$*, so that*

$$(x, y, z) \equiv \mathbf{x}(\xi, t) = \nabla_\xi P(\xi, t), \qquad H_\xi(P(\xi, t)) \succ 0. \tag{4}$$

Finding the (three dimensional) map $\mathbf{F}$ and the associated mesh $\mathcal{T}_P$ is thus reduced to the simpler problem of finding the scalar mesh potential P. As $\mathbf{x} = \nabla_\xi P$ it follows immediately that $J(\xi) = H(P)$ where $H(P)$ is the Hessian matrix of P. Hence the Jacobian $J(\xi)$ is a *symmetric matrix* which imposes certain restrictions on $\mathbf{F}$. For example it cannot be a plane rotation. Such maps are called *Legendre Transformations* and play an important role in many fields including fluid mechanics and image processing [20] In 3-dimensions the determinant of the Hessian of P is given by

$$|H(P)| = \begin{vmatrix} P_{\xi\xi} & P_{\xi\eta} & P_{\xi\zeta} \\ P_{\eta\xi} & P_{\eta\eta} & P_{\eta\zeta} \\ P_{\zeta\xi} & P_{\zeta\eta} & P_{\zeta\zeta} \end{vmatrix}. \tag{5}$$

The equidistribution equation (2) then becomes the following equation for P:

$$m(\nabla_\xi P, t)|H(P)| = \int_{\Omega_P} m \, \mathbf{d}x \tag{6}$$

which is a Monge-Ampère equation. To fully specify the mesh we need to impose boundary conditions on P. Typically we require that the boundary Γ_C of Ω_C is mapped to the boundary Γ_P of Ω_P. If the latter is given implicitly by the condition

$$\Gamma_P = \{(x, y, z) : G(x, y, z) = 0\}$$

then we have the nonlinear Neumann boundary condition

$$G(\nabla_\xi P) = 0 \quad \text{if} \quad \xi \in \Gamma_P. \tag{7}$$

Observe that this procedure allocated points to the boundary, but does not prescribe their precise location. If Ω_P is a cuboid domains so that, for example, one face of Ω_P is given by the plane $x = 0$, then the nonlinear condition (7) simplifies to the simpler linear Neumann condition

$$P_\xi = 0. \tag{8}$$

For certain problems, for example a number of problems in meteorology, it is natural and convenient to use periodic boundary conditions instead. See [16] for an example.

When calculating a mesh, particularly when using the relaxation methods we will introduce presently, it is useful to have a measure of the mesh quality. If we assume that an ideal mesh is one which perfectly equidistributes the monitor function m then an appropriate such measure is given by the deviation away from such an equidistributed state, and is given as follows.

Definition 2. We define the *equidistribution error* ε to be

$$\varepsilon(t) := \mathrm{CV}\left[m(x,t)|J(\xi,t)|\right] \equiv \frac{\left(\mathrm{Var}\left[m(x,t)|J(\xi,t)|\right]\right)^{0.5}}{\overline{[m(x,t)|J(\xi,t)|]}}, \tag{9}$$

where the Coefficient of Variation, CV, is the quotient of the standard deviation and the mean taken over all the gridpoints in the domain.

Note that we use the coefficient of variation as it is a dimensionless quantity, and is equivalent to the L_2 norm when the monitor function in question has been normalised so that $\int_{\Omega_P} m \, \mathbf{d}x = 1$. We will use this as a measure of the convergence of the relaxation methods. However, we observe at this stage that this is a relatively crude measure of the quality of a mesh, and in practice many other measures are important such as the skewness and the alignment of the mesh [17].

3 The Parabolic Monge-Ampère formulation

Equation (6) is a fully non-linear elliptic PDE which is challenging to solve exactly. There is a significant literature describing various solution techniques both for the equation in its own right [21], as part of a meteorological calculation [22, 23] and as part of a mesh generation algorithm [6],[14]. Typically these methods use a careful finite difference or finite element discretisation of (6) which is then solved using an iterative Newton-type algorithm which is terminated when a specified condition is met, for example a measure of the equidistribution of the mesh. In [6] a fast multi-grid method is used to perform these calculations. In the context of mesh generation, we do not necessarily want to invest too much effort in solving (6) as the function of this calculation is to generate a mesh which is then used for other calculations. In this context an accurate solution of (6) is unnecessary, provided that the resulting mesh is sufficiently regular and aligned, and exhibits the correct compression properties that we desire. Accordingly, there are certain advantages in the context of mesh generation, of using methods to solve (6) which are relatively simple to implement, robust, and for which each computational step is relatively cheap. An example of such is a simple explicit relaxation method, implemented cheaply using a Forward Euler method. Such a relaxation method can be terminated at any time when the mesh generated is sufficiently regular for subsequent computations. In two-dimensions it has been demonstrated [11], [16], that such a parabolic relaxation of the Monge-Ampère equation, the Parabolic Monge-Ampère equation (PMA), is effective for generating meshes. We now extend

this method to higher dimensions and demonstrate that it continues to be effective as a mesh generator as well as considering its convergence properties and scalability. In this formulation we initially consider the true time t to be fixed during the computation of the mesh, and introduce a *pseudo-time* $\tau \in [0, \infty)$ and a corresponding pseudo-time dependent function $Q(\xi, \tau)$ so that $\nabla_\xi Q \to \nabla_\xi P$ as $\tau \to \infty$ where P solves (6).

Definition 3 (PMA)**.** The Parabolic Monge-Ampère equation in $d-$dimensions is defined by

$$LQ_\tau \equiv (I - \gamma \Delta_\xi) Q_\tau = (\hat{m}(\nabla_\xi Q)|H(Q)|)^{\frac{1}{d}} \tag{10}$$

where γ is a scalar parameter defining the amount of smoothing applied. The function $\hat{m}$ is a filtered version of the monitor m obtained by averaging m over several mesh points. (The necessity for such filtering for data assimilation problems is carefully illustrated in [1].) Q_τ is the pseudo-time derivative of the mesh potential Q.

We will use a discrete approximation to this equation, in both time and space, to solve this equation and hence to find the mesh. In this equation the application of L^{-1} acts as a smoothing preconditioning operator (described first in [24]) which leads to more regular meshes. Furthermore the action of L^{-1} on the discrete form of the right hand side of (10) acts to damp out certain (mesh dependent) chequer-board instabilities [25] and appears to increase the robustness of the method. It can be rapidly calculated for cuboid domains by using the FFT or the Fast Cosine Transform (depending upon whether we have periodic or Neumann boundary conditions). The operator $(H(Q))^{1/d}$ is used on the RHS (instead of $H(Q)$) as it has the property that $(H(\lambda Q))^{1/d} = \lambda (H(Q))^{1/d}$. Thus both sides of (10) scale linearly. This is useful both to ensure global existence of the solutions of (10) and to give it certain desirable scaling properties [11]. It is further shown in [11] that the equation (10) is *locally stable* so that, if $\nabla_\xi Q$ is sufficiently close to $\nabla_\xi P$ then $\nabla_\xi Q \to \nabla_\xi P \quad \text{as} \quad \tau \to \infty$. with standard linear convergence. Furthermore, during the evolution of (10) both $H(P)$ and $\nabla^2 Q$ are bounded away from zero. This prevents mesh tangling provided that the equation (10) has a sufficiently fine discretisation [11], although as we shall see in Section 5.5, tangling may occur if too large a temporal step size is used when finding an approximate solution to (10), and we will discuss estimates for this largest step size in that section.

The convergence of the above relaxation method can be determined either by monitoring the equidistribution error $\varepsilon(\tau)$ defined in (9), or by monitoring the change in ∇Q. Indeed, we can define a convergence measure, $r(\tau)$, for the PMA equation as the Wasserstein distance between $\nabla_\xi \tilde{Q}$ at two successive timesteps τ and $\tau + \delta\tau$. This allows us to measure when $\nabla_\xi \tilde{Q}$ has converged. As $r \to 0$, $\nabla_\xi Q \to \nabla_\xi P$ and hence $\varepsilon \to 0$ and so the resulting mesh will satisfy the equidistribution equation.

The evolutionary system (10) is subject to the same boundary conditions as (6). It is convenient when solving the PMA equation, especially when using periodic boundary conditions, to consider instead of Q the difference between it and the function $|\xi|^2/2$. Consider the displacement of the periodic potential, $\tilde{Q}$, such that

$$\tilde{Q} = Q - \frac{|\xi|^2}{2}. \tag{11}$$

This gives

$$\nabla_\xi \tilde{Q} = \nabla_\xi Q - \xi \tag{12}$$

and hence

$$\mathbf{x} = \nabla_\xi \tilde{Q} + \xi \tag{13}$$

as $\mathbf{x} = \nabla_\xi Q$. The PMA equation can then be rewritten as

$$(I - \gamma \Delta_\xi)\tilde{Q}_\tau = (\hat{m}(\nabla_\xi \tilde{Q} + \xi)|I + H(\tilde{Q})|)^{\frac{1}{d}} \tag{14}$$

In the absence of a better initial guess, we use the initial conditions for (14) $\tilde{Q}(0) = 0$. In the case of a dynamically evolving monitor function, it is substantially more efficient to evolve $\tilde{Q}$ starting from the most recently computed value of $\tilde{Q}$. If the monitor function $\hat{m}$ is known then a corresponding mesh can be found by evolving (14) in time, either until a steady state is reached or until the resulting mesh is sufficient, in compression and regularity, for solving any coupled PDE or data assimilation problem. This latter option results in very significant time savings.

If the mesh is used to solve a time dependent PDE then the monitor function $m(t)$ will evolve in the true time t. In this case the mesh is evolved in the pseudo-time until it is adapted to the solution of the PDE. The solution of the PDE is then interpolated onto the new mesh. The true time is then advanced by an appropriate amount and the new solution to the PDE, and hence the new value of m is calculated. The process of finding the new mesh by evolution in pseudo-time is then repeated. We now consider the practical issues with solving (14) forwards in pseudo-time on the assumption that the monitor function is known a-priori. In our examples we will consider cases both where m is fixed and also where m evolves in time.

4 Implementation and convergence analysis

When implementing a discrete version of (14) to find $\tilde{Q}$ and hence the mesh, it is essential that the algorithm used is fast and robust as it will typically be part of a much larger solution process. For example, the UK4 model, a model with 4km resolution over the UK used by the Met Office for both numerical weather prediction and for data assimilation, has dimension $288 \times 360 \times 70 = 7257600$ grid points. Each of these has 3 degrees of freedom (latitudinal, longitudinal

and vertical) and each degree of freedom is stored in double precision and thus requires 8 bytes of storage. Hence to store one grid requires $288 \times 360 \times 70 \times 3 \times 8 = 174182400$bytes $= 166.11$MB. This shows the scale of the problem we are considering and why an efficient implementation of the algorithm to redistribute the mesh is essential. However, for mesh generation it need not be especially accurate provided that the mesh generated is sufficiently regular for computations.

Accordingly when calculating $\tilde{Q}$, we seek an explicit method where possible, for both time and memory considerations. One such method uses a forward Euler discretisation of (10) with step size $\delta\tau$ to evolve $\tilde{Q}$ so that

$$\tilde{Q}(\tau + \delta\tau) = \tilde{Q}(\tau) + \delta\tau \tilde{Q}_\tau(\tau) \tag{15}$$

where $\tilde{Q}_\tau(\tau)$ is given by

$$\tilde{Q}_\tau = L^{-1}(\hat{m}(\nabla_\xi \tilde{Q} + \xi)|I + H(\tilde{Q})|)^{\frac{1}{d}}. \tag{16}$$

We discuss the choice of $\delta\tau$ and the convergence of this algorithm presently.

To compute the RHS of (16) we discretise the Hessian operator in (16). This can be done most simply by using a finite difference scheme in the computational space Ω_C. We assume that Ω_C is divided into regular cuboids with the values of $\tilde{Q}$ given at the vertices of the cuboid. The location (x, y, z) of the mesh in the physical space Ω_P at these vertices can then be recovered from $\tilde{Q}$ by taking a discrete gradient (most simply by using central differences). The d-dimensional mesh can then be stored as d d-dimensional arrays, each containing one of the degrees of freedom of the mesh. So in a 2-dimensional case, with n_x grid points in the x-direction and n_y grid points in the y-direction, the mesh is stored as 2 $n_x \times n_y$ arrays. The first of which contains the x coordinates of the grid and the second containing the y coordinates. Similarly in the three dimensional case there are 3 arrays, x, y and z, each of size $n_x \times n_y \times n_z$ where n_z is the number of grid points in the z-direction. The connectivity of the grid is then implicitly defined by the relationship within the d-dimensional array. Algorithms 1 and 2 outline the steps taken to find a solution of the Monge-Ampère equation (6) and determine the corresponding mesh in the static and dynamic situations respectively. Due to memory constraints for the meteorological test problem, these algorithms to solve the PMA equation were implemented in Fortran95.

When the monitor function $m(t)$ itself evolves in time (for example if it is computed from a time evolving solution to a PDE) then we must augment Algorithm 1 (which evolves the mesh in pseudo-time) with an outer loop that evolves it in real time. This leads to Algorithm 2.

Note that Algorithm 1 is the basic one for a static application and Algorithm 2 is the natural choice for a time-dependent problem.

Algorithm 1 The PMA algorithm in 3D for a static monitor function

1: Read initial mesh $\xi = (\xi, \eta, \zeta)$
2: $\tau \leftarrow 0$
3: Initialise $\tilde{Q}(\tau) = \tilde{Q}_0$
4: Store the grid $\mathbf{x}(\tau) = (x(\tau), y(\tau), z(\tau))$ as

$$x(\tau) \leftarrow \xi + \frac{\partial \tilde{Q}(\tau)}{\partial \xi}, \quad y(\tau) \leftarrow \eta + \frac{\partial \tilde{Q}(\tau)}{\partial \eta}, \quad z(\tau) \leftarrow \zeta + \frac{\partial \tilde{Q}(\tau)}{\partial \zeta}.$$

5: **while** $r >$ tol & $\tau < \tau_{\max}$ **do**
6: Compute $\tilde{Q}_\tau(\tau)$ via:

- Compute the monitor function at the current grid points $m(\mathbf{x}(\tau))$. This may be analytically defined or interpolated from a given data set
- Filter the monitor function

$$\hat{m}(\mathbf{x}(\tau)) \leftarrow m(\mathbf{x}(\tau))$$

- Compute the second derivatives of $\tilde{Q}(\tau)$ in the computational space by using via finite differences to give discrete approximations to:

$$\tilde{Q}_{\xi\xi}(\tau), \tilde{Q}_{\eta\eta}(\tau), \tilde{Q}_{\zeta\zeta}(\tau), \tilde{Q}_{\xi\eta}(\tau), \tilde{Q}_{\xi\zeta}(\tau), \tilde{Q}_{\eta\zeta}(\tau)$$

- Calculate the determinant, $\rho(\tau)$, of the Hessian of the mesh potential $\tilde{Q}(\tau)$ at every current grid point:

$$\rho(\tau) \leftarrow |I + H(\tilde{Q}(\tau))|$$

- Calculate the smoothing operator L^{-1} by applying the Fast Cosine Transform to the 3-dimensional array $(\hat{m}(\mathbf{x}(\tau))\rho(\tau))^{\frac{1}{3}}$, so

$$\tilde{Q}_\tau \leftarrow L^{-1}(\hat{m}(\mathbf{x}(\tau))\rho(\tau))^{\frac{1}{3}}$$

7: Take a Forward Euler step

$$\tilde{Q}(\tau + \delta\tau) = \tilde{Q}(\tau) + \delta\tau \tilde{Q}_\tau(\tau)$$

8: Compute the finite difference approximations to $\frac{\partial \tilde{Q}(\tau)}{\partial \xi}$, $\frac{\partial \tilde{Q}(\tau)}{\partial \eta}$ and $\frac{\partial \tilde{Q}(\tau)}{\partial \zeta}$
9: Store the new grid as

$$x(\tau) \leftarrow \xi + \frac{\partial \tilde{Q}(\tau)}{\partial \xi}, \quad y(\tau) \leftarrow \eta + \frac{\partial \tilde{Q}(\tau)}{\partial \eta}, \quad z(\tau) \leftarrow \zeta + \frac{\partial \tilde{Q}(\tau)}{\partial \zeta}$$

10: Compute the change in the mesh through the Wasserstein metric

$$r(\tau) \leftarrow \|\nabla_\xi \tilde{Q}(\tau + \delta\tau) - \nabla_\xi \tilde{Q}(\tau)\|_2 N^{-\frac{1}{2}}$$

11: $\tau \leftarrow \tau + \delta\tau$
12: **end while**

Algorithm 2 The PMA algorithm in 3D for a dynamic monitor function $m(t)$

1: $t \leftarrow 0$
2: Apply Algorithm 1 with $m(\mathbf{x}) = m(\mathbf{x}, 0)$, $\tilde{Q}_0 \equiv 0$ and $\tau_{\max} = \infty$
3: **while** $t < t_{\max}$ **do**
4: Apply Algorithm 1 with $m(\mathbf{x}) = m(\mathbf{x}, t)$ and the initial potential $\tilde{Q}_0$ given by the final value of $\tilde{Q}(\tau)$ from the previous iteration of Algorithm 1
5: $t \leftarrow t + \delta t$
6: **end while**

Now we elaborate on the details of the algorithms to show both how the PMA method can be implemented in practice in 3 dimensions and to discuss its reliability, convergence and complexity. For all problems we will assume that a cuboid region Ω_C of dimensions $[0, 1]^3$ is mapped to a corresponding cuboid region Ω_P of dimensions $[0, 1]^3$. As described in Section 2 this leads to a problem with Neumann boundary conditions of the form

$$\tilde{Q}_\xi(0, ., .) = \tilde{Q}_\xi(1, ., .) = \tilde{Q}_\eta(., 0, .) = \tilde{Q}_\eta(., 1, .) = \tilde{Q}_\zeta(., ., 0) = \tilde{Q}_\zeta(., ., 1) = 0.$$

For this implementation we assume that Ω_C has a regular cubic mesh (although in practice any suitable mesh could be used) with, respectively, n_ξ, n_η and n_ζ cubes in the the three coordinate directions, of corresponding side lengths h_ξ, h_η and h_ζ.

4.1 First order differentiation

With the mesh potential Q stored in an d-dimensional ordered array, computing the first order derivatives is straight forward to implement using a central differencing scheme. So for instance in the 3 dimensional case, the derivative with respect to ξ is given by

$$\tilde{Q}_\xi(j, :, :) \approx \frac{\tilde{Q}(j + 1, :, :) - \tilde{Q}(j - 1, :, :)}{2h_\xi}, \quad j = 2 : n_\xi - 1$$

At the boundaries we invoke the Neumann boundary conditions so that

$$\tilde{Q}_\xi(1, :, :) = \tilde{Q}_\xi(n_\xi, :, :) = 0.$$

Derivatives with respect to other variables follow similarly.

4.2 Second order differentiation

In the interior of the domain, central differences are employed to estimate the second derivatives, such that

$$\tilde{Q}_{\eta\eta}(:, j, :) \approx \frac{\tilde{Q}(:, j + 1, :) - 2\tilde{Q}(:, j, :) + \tilde{Q}(:, j - 1, :)}{h_\eta^2}, \quad j = 2 : n_\eta - 1$$

and similarly for mixed second derivatives away from the boundary, so that for example

$$\tilde{Q}_{\xi\zeta}(i,:,k) \approx \frac{1}{4h_\xi h_\zeta}(\tilde{Q}(i+1,:,k) - \tilde{Q}(i-1,:,k) - \tilde{Q}(i+1,:,k-1) + \tilde{Q}(i-1,:,k-1))$$

for all $i \in \{2,\ldots,n_\xi - 1\}$ and $k \in \{2,\ldots,n_\zeta - 1\}$.
Similar approximations can be used for the other second order derivatives of $\tilde{Q}$.

On the boundary planes, the Neumann boundary condition satisfied by $\tilde{Q}$ is exploited to determine the appropriate discretisation of the Hessian on each of the boundaries. This condition implies that certain mixed derivatives on the boundary are automatically zero. For example on the boundary plane given by $\eta = 0$ we have $\tilde{Q}_\eta = 0$ and hence

$$\tilde{Q}_{\xi\eta} = \tilde{Q}_{\eta\zeta} = 0.$$

The derivatives $\tilde{Q}_{\xi\xi}, \tilde{Q}_{\zeta\zeta}, \tilde{Q}_{\xi\zeta}$ on this boundary away from the edges, can be approximated by a standard second order difference scheme, and the final derivative, $\tilde{Q}_{\eta\eta}$ is then given (exploiting the Neumann boundary condition) by the one-sided second order approximation

$$\tilde{Q}_{\eta\eta}(:,1,:) \approx \frac{-7\tilde{Q}(:,1,:) + 8\tilde{Q}(:,2,:) - \tilde{Q}(:,3,:)}{2h_\eta^2}.$$

Similar approximations are used at the other interior points on the boundary planes.

Along the boundary edges at the intersection of the planes (and at the corners of the domain), slightly more care has to be taken, with one-sided approximations to the second derivatives taken in two directions.

4.3 Filtering of the monitor function

As described above, some form of filtering of the monitor function is required in practice [1], [11] to produces sufficiently smooth meshes in a reasonable time. This is typically achieved in numerical weather prediction and other similar applications by applying an appropriate low pass filter [11] to the monitor function m. For a three dimensional isotropic problem this most conveniently can take the form:

$$\hat{m}(i,j,k) = \frac{\sum_{\ell_1=-1}^{1}\sum_{\ell_2=-1}^{1}\sum_{\ell_3=-1}^{1} m(i+\ell_1, j+\ell_2, k+\ell_3)\beta^{|\ell_1|+|\ell_2|+|\ell_3|}}{\sum_{\ell_1=-1}^{1}\sum_{\ell_2=-1}^{1}\sum_{\ell_3=-1}^{1}\beta^{|\ell_1|+|\ell_2|+|\ell_3|}}. \tag{17}$$

Here β is a smoothing parameter such that $\beta \in [0,1]$. However, this type of filtering of the monitor function is not suitable for highly anisotropic cases, for

example the highly stratified flows treated in the data assimilation application of [1]. However, filtering only within horizontal atmospheric layers retains this stratified structure [1]. Thus a filtering operator that is more suitable for the data assimilation context that we consider is as follows:

$$\hat{m}(i,j,k) = \frac{\sum_{\ell_1=-1}^{1}\sum_{\ell_2=-1}^{1} m(i+\ell_1, j+\ell_2, k)\beta^{|\ell_1|+|\ell_2|}}{\sum_{\ell_1=-1}^{1}\sum_{\ell_2=-1}^{1}\beta^{|\ell_1|+|\ell_2|}} \tag{18}$$

This produces much sharper monitor functions and hence gives better refinement of the grid around the structures of interest. With real data this filtering has to be applied several times in order to get a monitor function which will produce a grid with sufficient regularity.

4.4 Applying the smoothing operator L^{-1}

For the solution of PMA on a domain with purely Neumann boundary conditions, the Fast Cosine Transform can be employed to calculate L^{-1} and hence to apply the smoothing operator of the left hand side of the PMA equation (10) in $\mathcal{O}(N\log(N))$ operations. In an d-dimensional problem this transform has to be applied d times; once along each dimension of the mesh. The freely available software FFTW [26] was used to apply the Fast Cosine transform as it has the ability to work on multidimensional arrays *in-place*. That is to say the data structures do not need to be manually altered to perform a Fast Cosine Transform along different dimensions. In the 3-dimensional case, the routine dfftw_plan_r2r_3d is used with the option FFTW_REDFT10 along each dimension to signify the forward fast cosine transform. When the forward transform has been applied, the transformed variable is multiplied by the factor

$$1/(1+\gamma(k_\xi^2+k_\eta^2+k_\zeta^2)). \tag{19}$$

where the frequency-space coefficients k_ξ, k_η and k_ζ are 3D vector fields given by

$$k_\xi(i,j,k) = \frac{i-1}{n_\xi-1}\pi n_\xi, \quad k_\eta(i,j,k) = \frac{j-1}{n_\eta-1}\pi n_\eta, \quad \& \quad k_\zeta(i,j,k) = \frac{k-1}{n_\zeta-1}\pi n_\zeta$$

for all $i \in \{1,\ldots,n_\xi\}$, $j \in \{1,\ldots,n_\eta\}$ and $k \in \{1,\ldots,n_\zeta\}$. Then the inverse Fast Cosine Transform is applied via dfftw_plan_r2r_3d used with the option FFTW_REDFT01 along each dimension. This whole operation is equivalent to applying the operator $(I-\gamma\Delta)^{-1}$ and can be seen to explicitly damp the higher order frequency components in the mesh, such as the potential chequer-board modes which can arise in the discretisation of the Hessian operator.

If the number of mesh points is N it follows from the above that the complexity of each time step of the PMA algorithm is $\mathcal{O}(N\log(N))$.

4.5 The overall convergence of the PMA method

The local convergence of the PMA algorithm is studied in [11]. It is shown in this paper that the convergence of ∇Q to the solution ∇P of the Monge-Ampère equation is locally exponential. In particular, if ∇Q is sufficiently close to ∇P then there are constants A and λ so that to leading order

$$\|\nabla Q - \nabla P\|_2 = Ae^{-\lambda\tau}, \tag{20}$$

where A and λ depend on the structure underlying problem (in particular the monitor function) and not on the mesh size N. It follows immediately that if the monitor function is calculated exactly that the equidistribution error, measuring the coefficient of variation of $m(\nabla Q, t)|H(Q)|$, has a similar behaviour, with the same decay rate, so that to leading order

$$\varepsilon(\tau) = Be^{-\lambda\tau}. \tag{21}$$

The Wasserstein measure of mesh movement $r(\tau)$ defined in Algorithm 1 is given by

$$r(\tau) = \|\nabla Q(\tau + \delta\tau) - \nabla Q(\tau)\|_2 N^{-\frac{1}{2}}.$$

It follows from (20) that to leading order

$$r(\tau) = A\lambda\delta\tau e^{-\lambda\tau}. \tag{22}$$

The numerical examples calculated presently will give support to the above convergence formulae. An immediate consequence of the estimates (20), (21) and (22) is that the rate of convergence of the two measures $r(\tau)$ and $\varepsilon(\tau)$ of the PMA algorithm are both independent of the mesh size N, which result is verified in the numerical examples as will be shown in Figure 2b and Table 2. This implies that the overall complexity of the algorithm depends mainly on the effort made at each computational step. This complexity will be discussed in Section 4.7.

4.6 Choice of the parameters $\delta\tau$ and γ

When applying the PMA algorithm we must make decisions on how rapidly the mesh must be updated, the degree of convergence at each iteration, and the degree of smoothing which must be applied. This requires us to determine appropriate values for the two parameters used in the static case (Algorithm 1), namely $\delta\tau$ and γ. Whilst the continuous PMA algorithm can be proven [11] to evolve the mesh without tangling, such behaviour is not necessarily found in the discrete implementation of this algorithm unless the time step $\delta\tau$ is taken sufficiently small. Indeed, if the time-step $\delta\tau$ is too large, the Hessian matrix H will typically become indefinite, leading to mesh crossing and other undesirable features, although, as we shall see, the PMA algorithm is actually robust to mesh tangling problems provided that $\delta\tau$ is small enough. In contrast, if $\delta\tau$ is too small then the whole system becomes overly stiff. The parameter $\delta\tau$ can

be controlled adaptively, however it is generally robust to being set at a small constant value. To estimate this value we note that the intrinsic time-scale of this system is given by $\left(\int m \, \mathrm{d}\xi\right)^{-1/d}$. A choice of time-step is to then take

$$\delta\tau = \epsilon \left(\int m \, \mathrm{d}\xi \right)^{-1/d} \tag{23}$$

where ϵ is a small constant value typically in the range $0.1 \leq \epsilon \leq 1$. In the numerical experiments we present in Section 5, we compare this estimate with the maximum value of $\delta\tau \equiv \delta\tau^*$ that can be taken before mesh tangling is observed for a number of different test cases, and will find empirically that a value of $\epsilon = 2/5$ works in these cases. We also note that the choice of $\delta\tau$ given by (23) also has certain useful features when scaling symmetries act on the system [12], leading to meshes which reproduce self-similar behaviour in the solution. We note that this is a fairly crude estimate of the maximum possible value of $\delta\tau$ as it does not take into account issues such as mesh skewness which are likely to affect mesh tangling. A more precise such estimate is the subject of further research.

The parameter γ appears in the smoothing operator $L \equiv (I - \gamma\Delta_\xi)^{-1}$ as part of equation (16) and is applied in (19). Larger values of γ correspond to higher smoothing of the calculated mesh. Typically we have found that the smaller the value of γ, the faster that PMA converges to an equidistributed mesh. However with γ too small mesh tangling can occur. Hence once the step length for the Euler method ($\delta\tau$) has been chosen above then γ is chosen to balance the speed of convergence with the robustness of the method. Although the smoothing does make an individual step more computationally expensive, the increase in the robustness of the method greatly compensates for this. Values of γ in the range $\gamma \in [0.1, 0.6]$ are typical and, as above, these could be set adaptively for best performance.

In the case of a dynamically evolving monitor function where we use Algorithm 2, δt corresponds to the natural time-scale of the model (i.e. the underlying solution of the PDE). If the PDE is calculated numerically then it is sensible (and usual) to take δt to be the same as the time-step used to evolve the solution of the PDE, although occasionally we might interpolate the value of m between time steps allowing us to use values of δt which are smaller than the time-step in the method. When the initial redistributed mesh has been found in step 2 of Algorithm 2, it is desirable that the mesh is updated more rapidly than the solution of the underlying PDE, so that it can track it effectively, but not much more rapidly, so that we are not working too hard to calculate the mesh. For the inner loop of Algorithm 2 (step 4), a value of $\delta\tau = 0.1\ \delta t$ is appropriate for many applications. In the inner loop of Algorithm 2 it is not always necessary to run the pseudotime iterations for a long time, as the mesh remains close to equidistribution provided δt is not too large. Instead we set $\tau_{\max} = \delta t$ and take K iterations of the inner inner loop with time-step of $\delta\tau = \delta t/K$. In correspondence with the above, a typical value of K may be in the range $[1, 10]$,

with larger values necessary if the difference $\|m(\mathbf{x}, t+\delta t) - m(\mathbf{x}, t)\|$ is large.

4.7 Complexity and Scalability of the PMA algorithm

Assuming that the problem is always posed on a finite domain, then it is clear that the finite difference calculations for each step require $\mathcal{O}(N)$ operations. Similarly the low-pass filter given in Section 4.3 and the calculation of the Hessian of the mesh potential $\rho(\tau)$ are also of $\mathcal{O}(N)$ in complexity.

As described above, the fast cosine transform used in the smoothing preconditioner, is known to be of complexity $\mathcal{O}(N \log N)$, and hence the complexity of applying the smoothing operator L^{-1} as given in Section 4.4 is $\mathcal{O}(N \log N)$. It may be possible to implement an optimal solver for this step however we have not considered this in this work as the amount of memory available constrained our problems before fftw lost efficiency. It should also be noted that the calculation of $\tilde{Q}_\tau$ can me made massively parallel. Minimal communication would be required for derivative calculations, whereas more would be required for the application of the fast cosine transform. However recent work has shown that this is possible very efficiently [27]. Thus the efficiency of the PMA method is limited only by the number of timesteps taken.

The total complexity to compute a single explicit Euler step is thus of $\mathcal{O}(N \log N)$. Hence the complexity to find a redistributed mesh using the PMA method is $\mathcal{O}(CN \log N)$ where C is the number of iterations used in the explicit Euler method. This number depends, of course, on the precise stopping criterion that we use for this method and the pseudo-timestep $\delta\tau$. If we use the equidistribution measure $\varepsilon(\tau)$ and compute until this reaches a threshold value ε^* then it follows from (21) that the pseudo-time $\tau*$ required to reach convergence is proportional to $|\log(\varepsilon)|/\lambda$ and is independent of N. We will see this behaviour in the examples given presently. Thus the number of Forward Euler steps is given by $\tau^*/\delta\tau$. As we will see in the following section, this constant is typically independent of N. Rigorously proving this is the subject of further research, as is the optimal choice for the step-size $\delta\tau$.

It is interesting to compare this scalability with that of other methods. The Newton-Raphson/multigrid method described in [6] scales (both theoretically and in the examples presented in their paper) as $\mathcal{O}(N)$, and has the rapid convergence advantages of the Newton method when it works. However, it is necessarily more complex to implement each step, than the PMA method, and of course requires a good initial guess. The PMKP (parabolic Monge-Kantorovich) algorithm described in [28] has a similar parabolic form to PMA (operating on the logarithm of the equidistribution measure) but does not employ the smoothing preconditioning operator at each time step, and is therefore of $\mathcal{O}(CN)$. However, as stated in [28], although each computational step is cheaper than PMA, they need to take more such steps. From the timings presented in the paper, it appears that $C \propto N$ resulting in an overall method with $\mathcal{O}(N^2)$ complexity. Another method described in [28] (for two-dimensional mesh generation) is

the FDMKP (Fluid Dynamics Monge-Kantorovich) method in which the velocity of the mesh points is determined through a fluid dynamics formulation of the Monge-Kantorovich problem, and this velocity integrated to give the mesh point location. This requires solving a minimisation problem to find the velocity field, involving solving a three dimensional Poisson equation, and its complexity is determined by the method used to perform this latter computation.

5 Static mesh results

We now present a series of examples chosen to demonstrate the performance of the PMA algorithm on various (large and) challenging problems. In particular the examples are chosen to investigate the correspondence of the symmetry and regularity of the mesh to that of the underlying monitor function, to demonstrate the avoidance of mesh tangling when calculating the meshes in three dimensions provided that $\delta\tau$ is chosen carefully, and to demonstrate the convergence and complexity of the algorithm. We will show in this section that the way of parabolising the Monge-Ampère equation presented in Sections 3 and 4 scales well for three dimensional problems. We also to show that the PMA algorithm can cope with very large problems for which the monitor function is defined only at data points. In this section results are presented for a series of time invariant test problems in which $m(\mathbf{x}, t) \equiv m(\mathbf{x})$ is taken to be a constant (in time) function, and only Algorithm 1 is used, starting from an initial potential $\tilde{Q}_0 = 0$. We note that simple analytical monitor functions have been used previously as test cases for adaptive mesh redistribution in two and three dimensions. One such paper [28], applied the PMKP method, which is related to PMA, and which seeks to solve a different form of the parabolic Monge-Kantorovich problem, as well as the FDMK method. This paper mainly considered numerical calculations for two-dimensional examples of varying size and also showed results when applied to a single three dimensional mesh with $41 \times 41 \times 41$ gridpoints; two orders of magnitude fewer degrees of freedom than some of the examples we consider in this paper and of a relatively simpler geometry.

The **first example** is a simple symmetrical case in which we present meshes generated by considering a monitor function which is large near the boundary of a sphere. This serves to show the symmetry preserving properties of the PMA equation and the regularity and alignment of the resulting meshes.

The **second example** is a more complicated, but still analytically determined, monitor function describing a helical feature. This will show more clearly the meshes which it is possible to construct which can represent a complex three dimensional geometry.

Finally in this section we will consider the very large and practical problem of generating adapted three-dimensional meshes for the purposes of meteorological data assimilation calculations. In this example we use forecast data from the Met Office UK4 model to define a monitor function based on an estimate of

the potential vorticity, looking at a sequence of different meteorological events. This example illustrates the effectiveness of the PMA algorithm to generate a mesh when used on a large scale practical three dimensional problem, with a monitor function defined by data values rather than an analytic function.

We will describe each of these examples in turn, and will then also study the relation between the largest usable value of $\delta\tau$ and the approximation (23).

For all of the examples, the codes for the PMA algorithm were executed on a laptop with an Intel® Core$^{\text{TM}}$2 Duo CPU P9400 @ 2.4Ghz with 4GB RAM running a 32-bit Linux OS and were compiled with the gfortran compiler in double precision. All reported times are wall-clock times measured using `system_clock`, averaged over 3 runs.

5.1 Simple test cases

5.1.1 Example 1: A three dimensional shell

We define the density $f(\mathbf{x})$ of a smooth three dimensional ball with a (graded) boundary of width r_2 and centred on the point (x_0, y_0, z_0) as follows. Let s be the distance of a point in our domain to the centre of the ball given by

$$s(\mathbf{x}) = s(x, y, z) = \sqrt{(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2}. \tag{24}$$

We then define the density of the ball via the function

$$f(\mathbf{x}) = f(x, y, z) = \begin{cases} 1 & \text{for } s(x, y, z) \leq r_1 \\ \frac{1}{2}\cos(\frac{(s(x,y,z)-r_1)\pi}{r_2}) + \frac{1}{2} & \text{for } s(x, y, z) \leq r_1 + r_2 \\ 0 & \text{for } s(x, y, z) > r_1 + r_2 \end{cases} \tag{25}$$

where r_1 and r_2 are scalars defining the width of the ball. For this problem we will consider generating a mesh which concentrates points close to the shell forming the boundary of the ball. This can be achieved by using a monitor function which is large when the derivatives of the density function $f(\mathbf{x})$ are also large. Accordingly, we define the monitor function $m(x, y, z)$ by

$$m(x, y, z) = \sqrt{(1 + c^2(f_x(x, y, z)^2 + f_y(x, y, z)^2 + f_z(x, y, z)^2))}. \tag{26}$$

Here c is a regularisation constant, which we set in our examples to be $c = 0.75$. We now consider a three dimensional mesh, constructed within the unit cube, and adapted to this monitor function in which we set the parameters defining the width of the ball to be $r_1 = r_2 = \frac{1}{6}$, and centred in the domain so that

$$(x_0, y_0, z_0)^T = (\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2})^T.$$

In the examples shown the computational domain $\Omega_C = [0, 1]^3$ is split into a grid of $n_\xi \times n_\eta \times n_\zeta$ points, with $n_\xi = n_\eta = n_\zeta = 100$ ($N = 10^6$) and is mapped into

the same physical domain (so that the solution of the PMA equation satisfies Neumann boundary conditions).

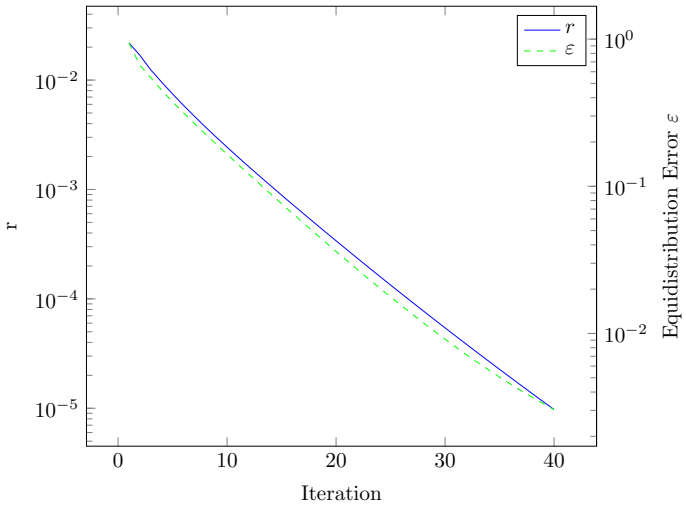
The PMA algorithm was applied to this problem with $\delta\tau = 0.2$ and $\gamma = 0.2$. The convergence of the mesh to an equidistributed state to a tolerance of $\varepsilon = 1E-5$ is shown in Figure 2a in which we plot both $\varepsilon(\tau)$ and $r(\tau)$. The calculation terminated after 41 iterations, taking 34 seconds on the laptop computer described earlier. From this figure we can clearly see the rapid, exponential convergence of the algorithm as predicted from (21), (22) with both $\varepsilon(\tau)$ and $r(\tau)$ converging at the same exponential rate. To determine the complexity of this calculation we repeated this calculation with a varying number of spatial mesh points N, keeping $\delta\tau$ fixed and computed until the tolerance threshold was reached. The number of iterations was computed and is shown in Figure 2b. We see that, as predicted from the analysis at the end of the last section, the number of iterations is essentially independent of N. As a consequence the computational complexity, and hence the CPU time, varies as $N\log(N)$ as can also be seen.

The resulting mesh is presented in Figure 3. From this simple test problem it is possible to see how the solution of the PMA equation is equidistributing the monitor function. There are many more grid points in the region where the monitor function is high than outside of that region, and the mesh shows excellent alignment with the boundary of the sphere. In Figure 3a we plot the values of the monitor function in three dimension, with part of the sphere cut away to show the variation in value across the shell. In Figure 3b we show a plane in the mesh that precisely follows the contours of the monitor function. Figures 3c and 3d show the grid from the centre of the computational domain projected onto the x–y plane in physical space. Figure 3d shows the regularity of the grid that is generated and that the PMA equation aligns the mesh with the contours of the monitor function. This elegant behaviour arises because symmetries in the monitor function lead to symmetries in the PMA equation and hence in the function Q.

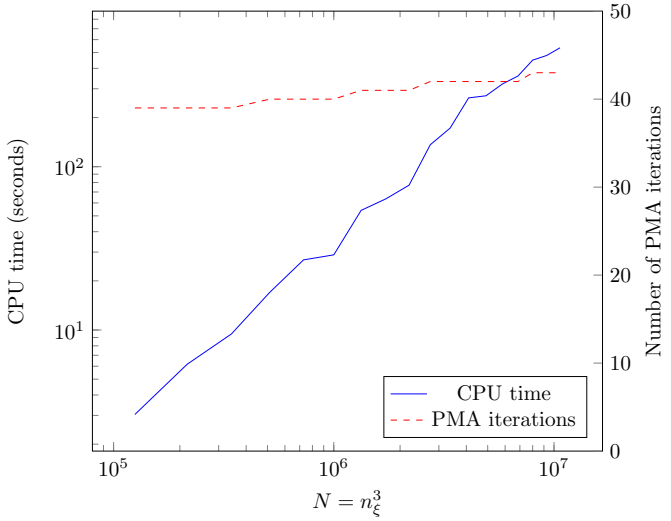
5.1.2 Example 2: A three dimensional helix

We next consider an analytically defined monitor function that describes a complex three dimensional helical surface without the symmetries of the shell. This problem was chosen as it leads to a very non uniform and twisted mesh, and it is thus a major challenge for the algorithm. In particular we might in principle expect to see more problems with mesh tangling. Taking $\mathbf{x} = (x, y, z)^T$ then a monitor function $m(\mathbf{x})$ which is large in a neighbourhood of such a helix, and regular elsewhere, is given by

$$m(x,y,z) = 5\exp(-w_1[(x-(w_2\cos(4z\pi)+0.5))^2+(y-(w_2\sin(4z\pi)+0.5))^2])+1 \tag{27}$$



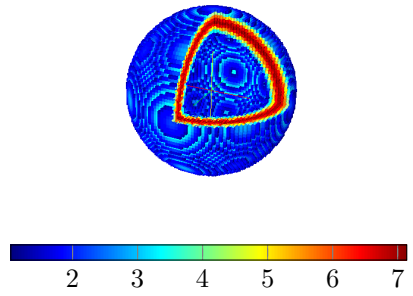
(a) Plot of the convergence for the shell problem with $n_\xi = 100$. Note the exponential convergence seen in this example, as predicted by (21).



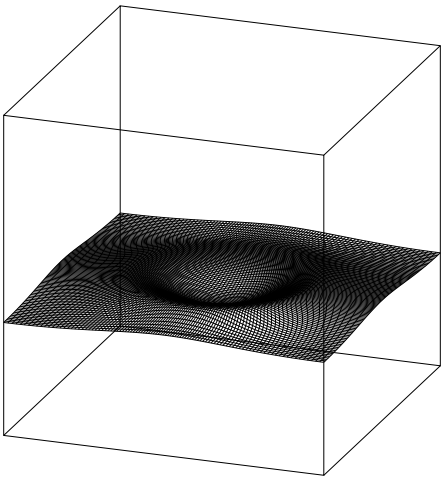
(b) The CPU time and number of iterations plotted as a function of the number of gridpoints N for the shell problem. Note the almost constant number of PMA iterations taken.

Figure 2: Performance plots for the shell problem.

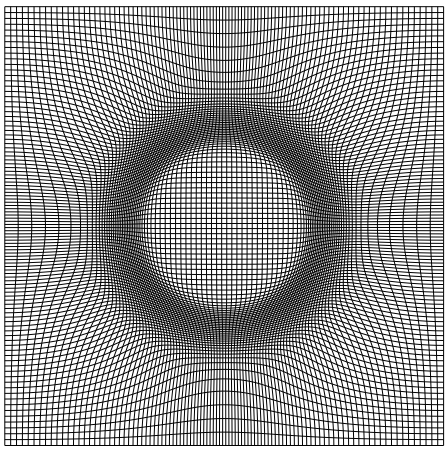
Here the parameter w_1 describes the width of this boundary neighbourhood, and the parameter w_2 gives the width of the helix. These are set to be $w_1 = 100$ and $w_2 = \frac{1}{4}$. The domain is split into $100 \times 100 \times 100$ grid points and the three



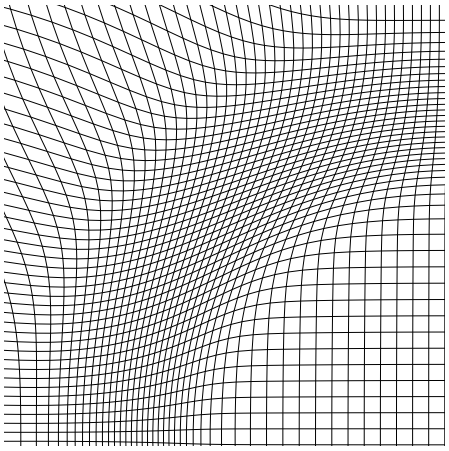
(a) Cut away 3D plot of the monitor function for $m > 1.05$. Note that this monitor function ranges from 1 to 7.14.



(b) 3D view of grid in physical space of the grid from $\zeta = 1/3$ in computational space.



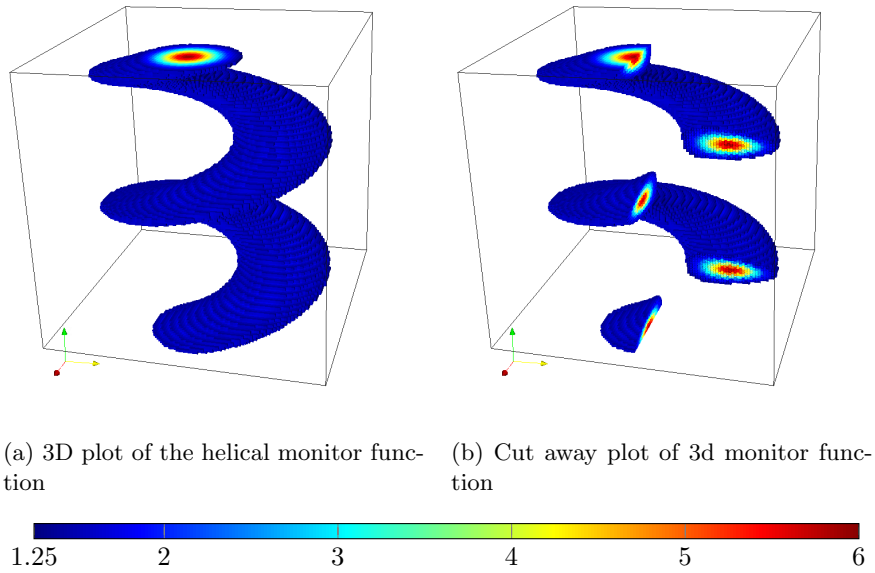
(c) Projected view of a plane of the mesh that was at $\zeta = 49/99$ in computational space



(d) Zoomed view of projected mesh from $\zeta = 49/99$ around the high monitor function.

Figure 3: The monitor function and the resulting sections from the mesh for the shell test problem.

dimensional values of the monitor function are shown in Figure 4.
The PMA algorithm was applied to the helical problem with $\delta\tau = 0.2$ and $\gamma = 0.2$. For these parameter values it was successful in generating a highly non-uniform mesh without any evidence of mesh tangling at any stage of the application of the algorithm. The exponential convergence of the mesh to an equidistributed state to a tolerance of $1E-05$ is shown in Figure 5. The



(a) 3D plot of the helical monitor function

(b) Cut away plot of 3d monitor function

Figure 4: 3D plots of the helical monitor function showing only those points with $m > 1.25$. Note that this monitor function ranges from 1 to 6.

calculation terminated after 24 iterations, taking 20.7 seconds on the laptop computer described earlier. In Figure 6 we show the mesh generated by the PMA algorithm when applied to the problem taking m as defined in (27). In Figure 6b we show where the two horizontal planes in Figure 6a are mapped to in physical space. Similarly Figures 6c and 6d show where the vertical planes in Figure 6a are mapped to in physical space. These show that the redistributed grid is closely following the monitor function and very clearly show the fully 3D nature of the problem.

5.2 Meteorological test problems

We now consider a large scale meteorological problem for which the monitor function is not given as an analytic function, but is instead defined at a set of discrete data points. This is a commonly encountered situation both in the numerical solution of PDEs or (as in this case) of function approximation where the function is only known at discrete points. Note that in this example we are not evolving a PDE, but simply redistributing a mesh around data derived from the solution of a PDE.

Data assimilation is the technique of matching noisy data to models of a process which also may have error. It is widely, and successfully, used in meteorology

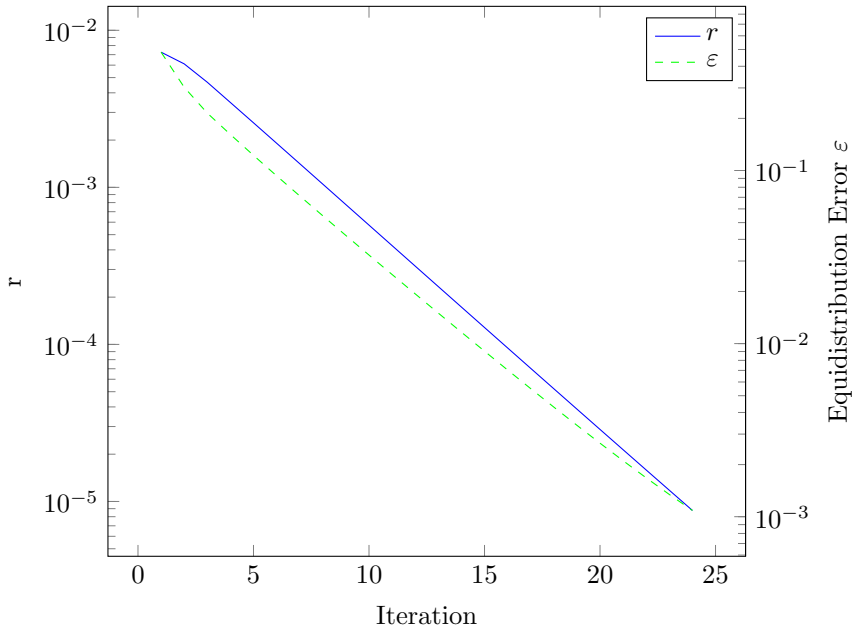
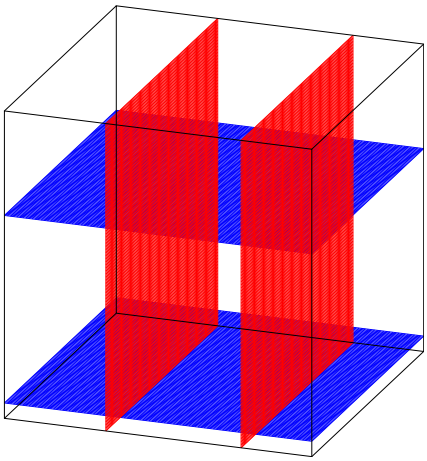


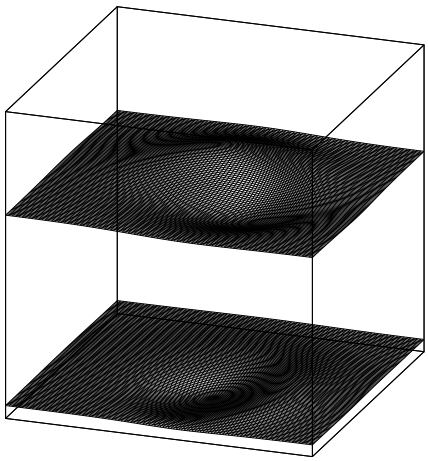
Figure 5: Plot of the convergence for the helical problem showing $r(\tau)$ and $\varepsilon(\tau)$. Again we see exponential convergence in 24 iterations.

to determine an atmospheric state consistent both with observations and with the underlying physics of the atmosphere. In order to implement variational data assimilation methods effectively, it is important that the underlying covariance matrix of the errors is well represented. This matrix is too large to store explicitly. In this context adaptive mesh redistribution can be applied to create a simplified and thus manageable representation of the background error covariance matrix, and in particular include a reasonable representation of the spatially varying structure of the covariances [4, 1]. The Met Office data assimilation system already implements a 1D adaptive meshing procedure for the vertical component of their grid used for their data assimilation algorithms. The improvement in data correlations represented by doing this has resulted in a measurable increase in forecasting accuracy [4, 1]. In this paper we consider the first step of extending this work by considering how to use the PMA algorithm to generate a suitable 3D mesh for data assimilation in a variety of meteorological conditions. A discussion of the implementation and testing of the adapted meshes within the data assimilation system will follow in a later paper.

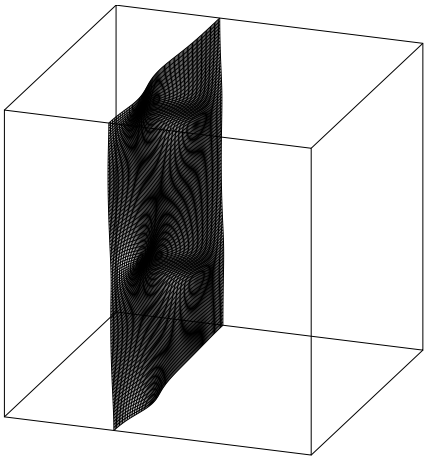
To be effective within the context of a data assimilation calculation, the mesh generation code must be both fast and robust to use, and must also be easily linked to the existing data assimilation software. For the Met Office application,



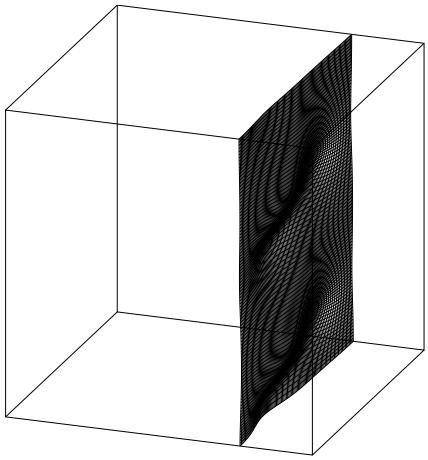
(a) Planes in the computational mesh showing where the meshes in Figures 6b–6d originate in computational space



(b) Location of the two horizontal planes in the physical space corresponding to the horizontal planes shown in computational space



(c) Location of the plane in physical space corresponding to $\eta = 1/3$ in computational space



(d) Location of the plane in physical space corresponding to $\eta = 7/9$ in computational space

Figure 6: 3D plots of the mesh generated by the helical monitor function at various slices.

the goal is to produce a weather forecast after using data assimilation to get a best guess for the current state of the atmosphere. This imposes an immediate operational time restriction on the time-frame in which the computations can be made, as a forecast delivered after the event is useless. This paper considers adapting the UK4 grid (4km horizontal spacing local area model over the

British Isles) with efficiency a key consideration for any future operational implementation. As a code for an operational centre, the meshes produced will have to run automatically and hence be robust to all weather conditions. Thus it is essential to have a monitor function which is well scaled to maintain good global resolution while still refining sufficiently around features of interest.

This specific application of adaptive meshing is as an aide to help calculate the background error covariance matrix within the data assimilation algorithm, as in [1, 4].

5.3 Defining a monitor function

In this example, the physical coordinates $\mathbf{x} = (x, y, z)$ correspond to longitude, latitude and vertical levels respectively. The vertical levels are defined using a terrain-following coordinate η which is a monotone function of height. It is plausible to assume that the correlation structure is isotropic in geostrophic and isentropic coordinates, which implies the use of the semi-geostrophic potential vorticity as a monitor function [22]. The PV is the Jacobian of the transformation from physical to geostrophic and isentropic coordinates. This is given in terms of the primitive variables u, v and θ by

$$PV = \begin{vmatrix} f + v_x & v_y & v_z \\ -u_x & f - u_y & -u_z \\ g\theta_x/\theta_0 & g\theta_y/\theta_0 & g\theta_z/\theta_0 \end{vmatrix}$$

where f is the Coriolis parameter (assumed constant), u and v are the wind velocities in the longitudinal and latitudinal directions respectively, g is the force due to gravity, θ is potential temperature and θ_0 a reference potential temperature [22]. Since the PV calculated from real data may not be positive, we use only the dominant diagonal terms of semigeostrophic potential vorticity to form the basis for the monitor function which we use to control the adapted mesh. Each of the diagonal terms is regularised to take account of the typical scale of the individual terms and ensure positivity. This resulting monitor function then has the following form

$$m = \begin{vmatrix} \sqrt{1 + c_1(1 + \frac{v_x}{10f})^2} & 0 & 0 \\ 0 & \sqrt{1 + c_2(1 - \frac{u_y}{10f})^2} & 0 \\ 0 & 0 & \sqrt{1 + c_3(\frac{\theta_z}{\theta_0})^2} \end{vmatrix}.$$

Note that the wind gradients u_y and v_x have been rescaled by a factor of 10 to remove some of the greater variability in the wind speeds than in the potential temperature. The constants c_1, c_2 and c_3 are regularisation parameters which allow for different weightings to be given to the different components. With a great deal of testing, it was found that all the normalisation parameters equal 0.75 gave good results. Note that $c_1 = c_2 = 0$ reduces this three dimensional

monitor function to the one dimensional static stability based monitor function, which is currently used operationally [4, 1].

In the application to atmospheric data assimilation it is important to respect the stratified structure of the atmosphere. Though the monitor function should be smoothed to avoid computational difficulties caused by rapid grid variations, the smoothing should be applied only in the horizontal and not the vertical. Thus the filtering operator that is applied is

$$\tilde{m}_{i,j,k} = \frac{\sum_{\ell_1=-1}^{1} \sum_{\ell_2=-1}^{1} m_{i+\ell_1,j+\ell_2,k} \beta^{|\ell_1|+|\ell_2|}}{\sum_{\ell_1=-1}^{1} \sum_{\ell_2=-1}^{1} \beta^{|\ell_1|+|\ell_2|}} \tag{28}$$

This produces much sharper monitor functions and hence gives better refinement of the grid around the structures of interest.

5.4 Test cases

In our calculations we considered three different meteorological data sets to test the grid generation capabilities of the 3D PMA algorithm. These data sets were actual forecast data provided by the UK Met Office for periods of very different weather conditions, in particular: (a) a stable boundary layer, (b) scattered showers, and (c) a frontal system. The dimension of this problem is $288 \times 360 \times 70 = 7257600$ gridpoints and hence 21772800 degrees of freedom. Over the scales we are interested in, the atmosphere is shallow, i.e. the vertical scale of the domain is much smaller than the horizontal scale. This presents computational issues for the solution of the atmospheric dynamics equations. However, we rescale the vertical "altitude" component of the physical domain into terrain following "level" coordinates. This rescaling removes the computational issues associated with the PDEs of atmospheric dynamics and allows us to work unhindered on the computational domain $[0, 1]^3$. In keeping with the possible operational restrictions on adapted grid generation, all parameters used in the subsequent results will be fixed across all cases to show the robustness of the method. In all of these calculations, the parameters used were $\delta\tau = 0.5, \gamma = 0.5$ and the convergence tolerance was set to $1E-05$. The PMA algorithm performed very well in each case and the meshes obtained captured all the features of the underlying localised systems (identified by the monitor function). Consequently we are confident that the resulting meshes should perform very well when used for data assimilation calculations. The table below and the plots in Figure 7, shows the convergence results from the three test cases.

Observe from Figure 7, that even in these large data sets, the PMA algorithm converges rapidly. Note that the final ε achieved here is greater than for the analytically defined monitor functions considered previously. This is a consequence of interpolation error in sampling the monitor function away from the given data points. We now show the resulting meshes in each case. For each figure we give the monitor function and the mesh at appropriate sections through the domain.

Test case	Iterations	CPU time (minutes)	Range of m	Final ε
Stable boundary layer	21	3.26	1.1–21.8	$2.11E-02$
Scattered showers	20	3.14	1.0–18.2	$2.02E-02$
Frontal system	20	3.12	1.0–13.8	$1.76E-02$

Table 1: Results for the three meteorological test cases

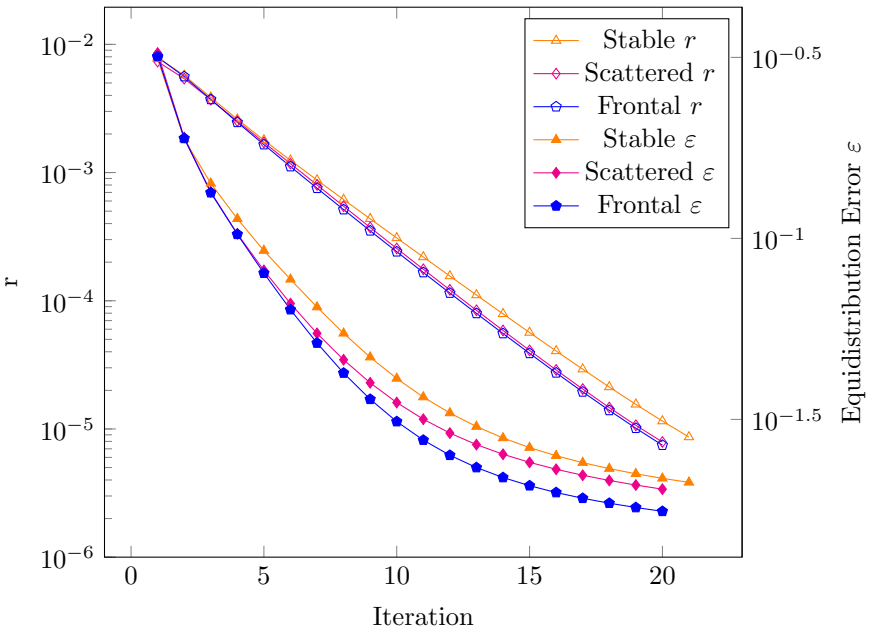


Figure 7: Plot of the convergence of PMA for the meteorological test cases. Note that the decay of r is exponential, but that ε decays in a different manner. This is because the monitor function is evaluated at discrete data points rather than being defined continuously.

5.4.1 Stable boundary layer

This test case uses the same UK4 model data described in [1], representing a scenario when UK was mainly covered by low-level clouds. The synoptic situation over the UK at the time (3rd January 2011 at 00UTC) was characterised by a weak flow within a large anticyclone of 1030 hPa surface pressure. Observed vertical profiles show saturated boundary layer below an inversion of 850 hPa.There is a warm front in the south-west with some likely enhancements from a vorticity anomaly aloft. This is associated with extensive low clouds particularly in the south-west. Figure 8 shows a cross section (longitude versus levels) of the monitor function described in Section 5.3 for 3 January 2011

at 00 UTC and the corresponding mesh. The three dimensional monitor function clearly captures the vertical structures in the troposphere which indicates the presence of clouds at different levels in agreement with the results showed when using the one dimensional static stability monitor function described in [1]. The mesh follows the monitor function by moving the vertical height levels further together when the monitor function is large and further apart when it is small. This is in agreement with the one dimensional results. In addition the three dimensional monitor function moves the mesh horizontally capturing more realistically local variations of the cloud layering.

Another cross section is shown in Figure 9. Again the mesh (latitudes versus height levels) follows the structure of the corresponding monitor function and captures local variability both vertically and horizontally.

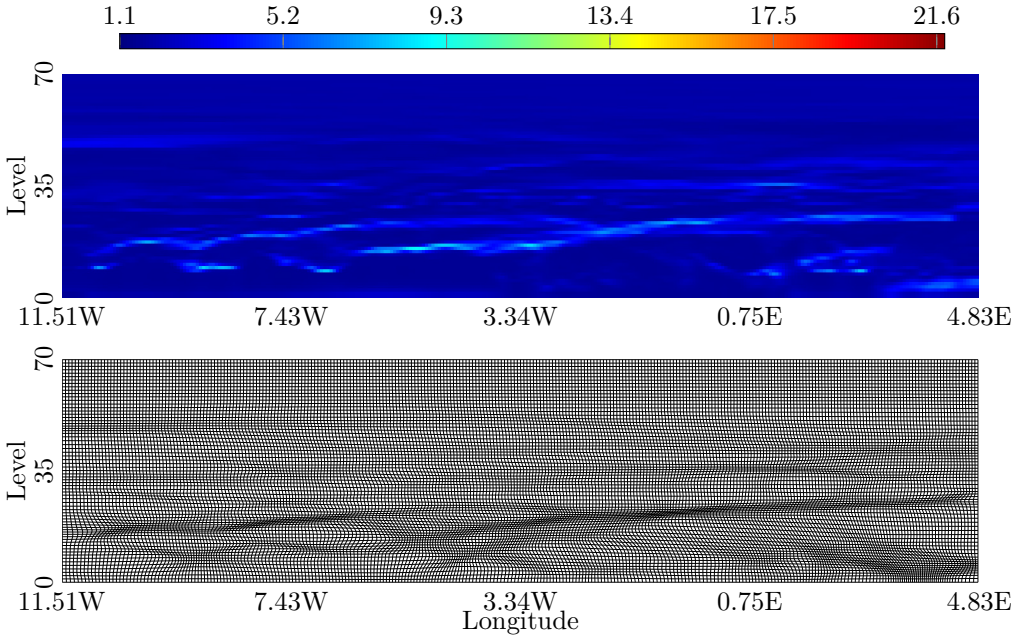


Figure 8: The monitor function and the resulting mesh for the stable boundary layer system at a 94^{th} latitude increment ad with increasing longitude. The function is shown in the vertical plane from $(50.68N, 11.51W)$ to $(50.80N, 4.84E)$

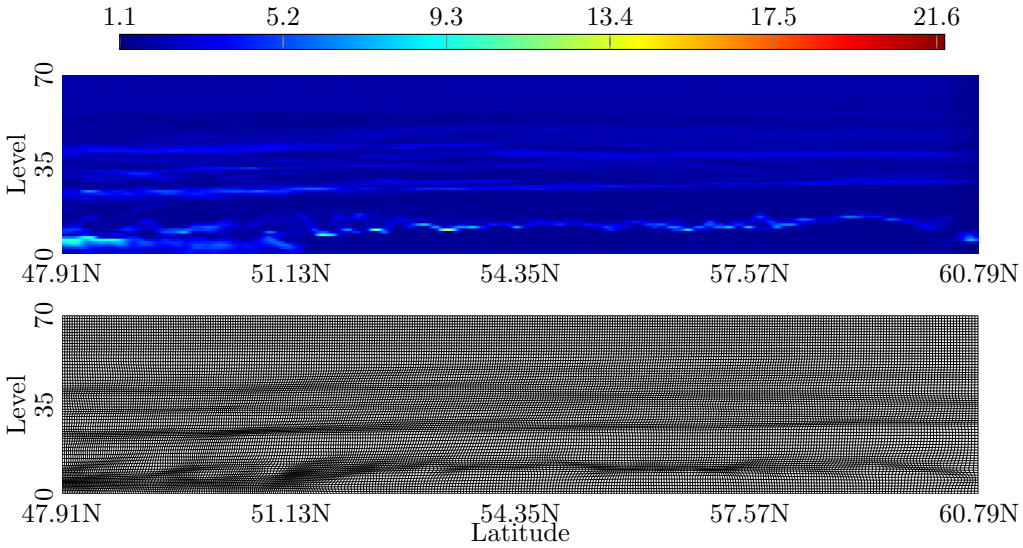


Figure 9: The monitor function and the mesh for the stable boundary layer system at a 260^{th} longitude increment. In the vertical plane from $(47.91N, 2.89E)$ to $(60.79N, 4.86E)$.

5.4.2 Scattered showers

The next two cases have been selected to test the capability of the scheme to capture two different extremes, i.e. localised convective activity as in the scenario of scattered showers and a large scale weather system as in the case of a front. The synoptic situation over the UK on the 24 April 2012 at 12UTC was characterised by a weak flow within a large scale upper trough with an upper filament of vorticity in the south-west of England giving focus to the convective activity. The latter gives large values of the (potential vorticity based) monitor function. The convective activity over the UK is shown by the radar image in Figure 10 in which the intensity of the rain showers shows up in the figure as regions of more intense colour. (In this figure the UK and Ireland occupy most of the region, with Scotland at the top. The most intense convective activity is over the North Sea just to the East of the NE cost of England.) The adaptive mesh scheme here needs to pick up very small and localised showers scattered over the UK as well as the response to the large scale forcing over SW England.

Figure 11 shows an horizontal cross section of the monitor function on the left and the corresponding mesh on the right for a low height level of the model. The monitor function tends to capture local and small scale phenomena. These do not coincide with the radar image in Figure 10, this is because the monitor function is calculated from a T+3h forecast and not from current observations. The monitor function does not respond to the random showers over Ireland, but does pick up the area with no showers over central England. The mesh follows the monitor function behaviour and clustered mesh points near the high values of the monitor function. When the showers are better organised and less random, like the filament over North Scotland, the mesh nicely aligns with this feature. Figure 12 shows instead a vertical cross section (latitudes versus height levels) for the same case. As well as capturing the small scale variations due to the showers the monitor function picks up the upper filament of vorticity (around level 35) and the lower filament over north Scotland (around level 8). The mesh nicely follows the behaviour of the monitor function both horizontally and vertically.

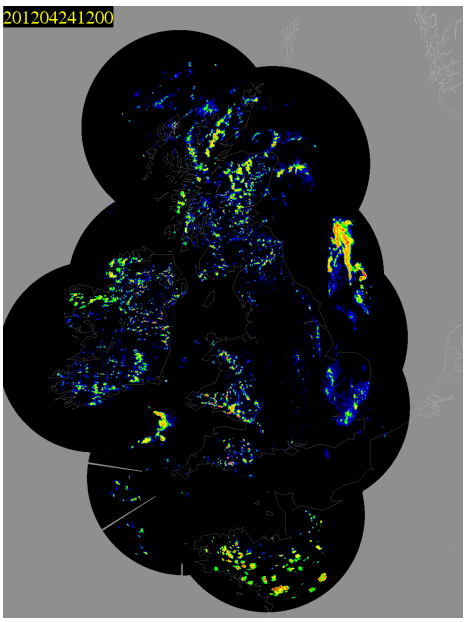


Figure 10: Radar image of the scattered showers system over the UK and Ireland showing isolated areas of high convective activity indicated by intense colours. These colours are areas of high reflectivity which correspond to rainfall.

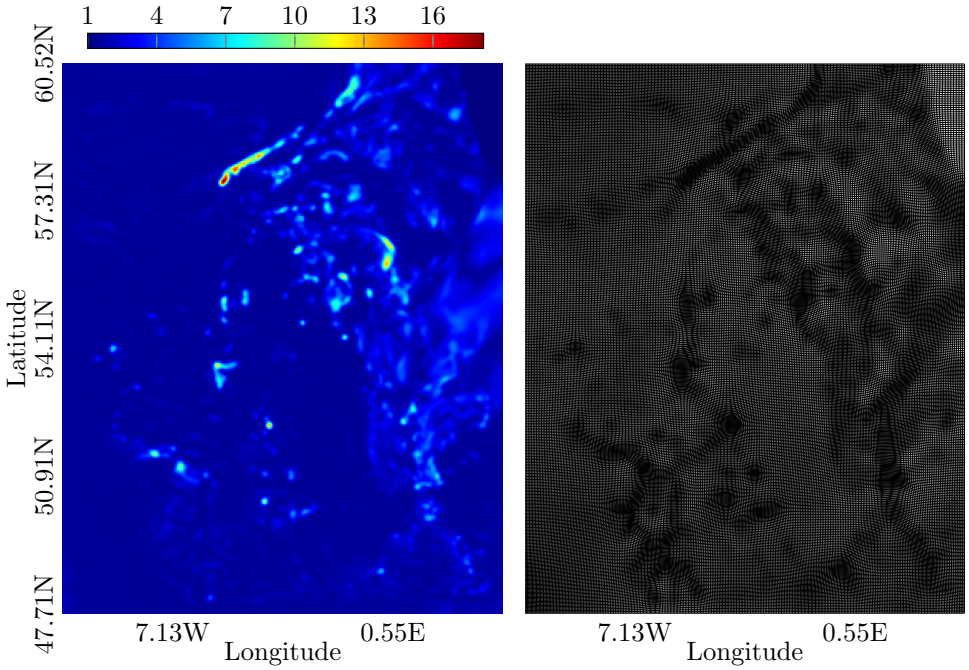


Figure 11: The monitor function and the mesh for the scattered shower system at the 8^{th} vertical level, or 261.7m

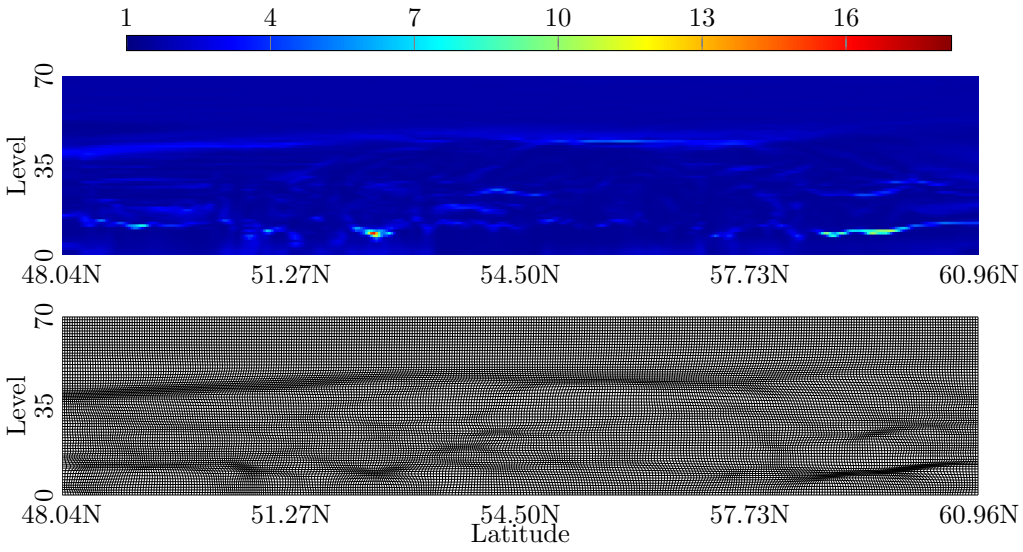


Figure 12: The monitor function and the mesh for the scattered shower system at a 135^{th} longitude increment. Vertical plane from $(48.04N, 3.81W)$ to $(60.96N, 4.29W)$.

5.4.3 A Frontal system

The last case described in this section follows from the scattered showers weather system. The large upper trough described in the previous section extends south and by 00UTC on the 25 April 2012 it drives the surface cyclonic system eastward bringing a warm front system into the south-west of UK. The activity on the front is strongly enhanced by vorticity forcing at 250 hPa. Figure 13 shows the radar image for the frontal system on the 25 April 2012 at 03UTC. Again in this figure the UK and Ireland occupy most of the region and a strong front can be seen in the South West crossing Devon. The horizontal cross section of the monitor function and the corresponding mesh for this case are shown in Figure 14. The front is clearly depicted in both pictures and the refinement of the mesh is high in correspondence with the front. Figure 15 shows the vertical cross section (latitude versus levels) of the monitor function and the resulting mesh. It clearly picks up the three dimensional structure of the front (around latitude 50N) as a function of height and latitude. The monitor function also displays extra vertical structures over the UK. Again the mesh nicely follows the behaviour of the monitor function both horizontally and vertically.

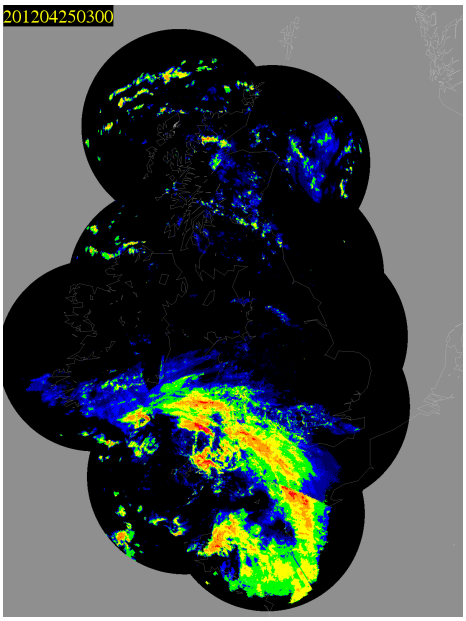


Figure 13: The radar image of the frontal system crossing the South West coast of the British Isles. The region of high reflectivity indicates the well organised rain band ahead of the front.

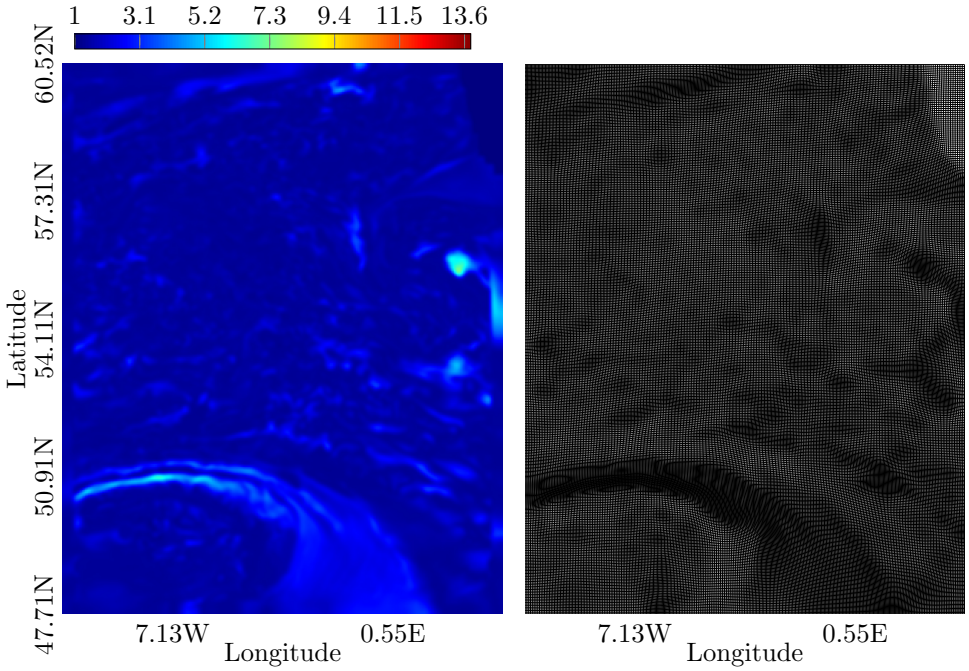


Figure 14: The monitor function and the mesh of the frontal system at the 23^{rd} vertical level, or 1911.7m

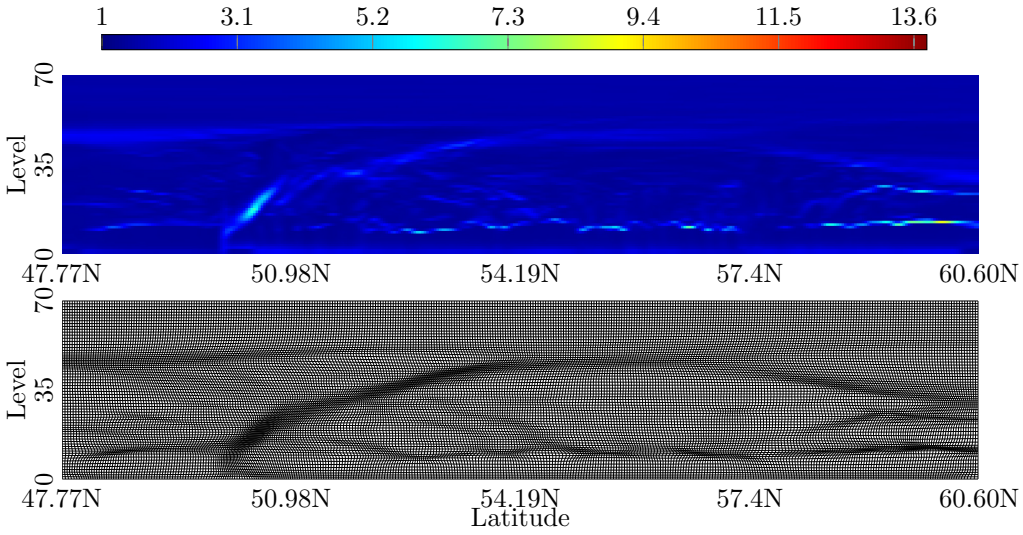


Figure 15: Monitor function and mesh of frontal system at 16^{th} longitude increment. Vertical plane from $(47.77N, 10.17W)$ to $(60.60N, 12.94W)$.

5.5 Estimates for the maximum step size $\max \delta\tau$

As described earlier, whilst in the limit of small $\delta\tau$ we expect to see an absence of mesh tangling, it is certainly the case that the PMA algorithm fails, due to mesh tangling, at a maximum value $\max \delta\tau$. In each of the test problems described in this section we kept N fixed and increased $\delta\tau$ until mesh tangling was observed at some stage in the application of the algorithm. This maximum value appeared from our experiments to be independent of N, but did depend on the example problem. In Figure 16 we show, for all the static examples considered in this paper, the numerically estimated largest value of $\delta\tau$. These are all plotted as a function of the estimate of inverse of the monitor function m given by $m^* = \left(\int m \, \mathrm{d}\xi\right)^{-1/3}$ as described earlier in (23). We can see from this figure that there is a reasonably good correlation between m^* and $\max \delta\tau$, and that m^* is of the correct magnitude for all of these examples. In these cases, $\max \delta\tau = \epsilon \left(\int m \, \mathrm{d}\xi\right)^{-1/3}$ where ϵ varies within the range 0.42 to 0.64.

For the choices of $\delta\tau$ which do converge, our numerical tests strongly imply that the constants predicted in (21) and (22) are indeed independent of $\delta\tau$.

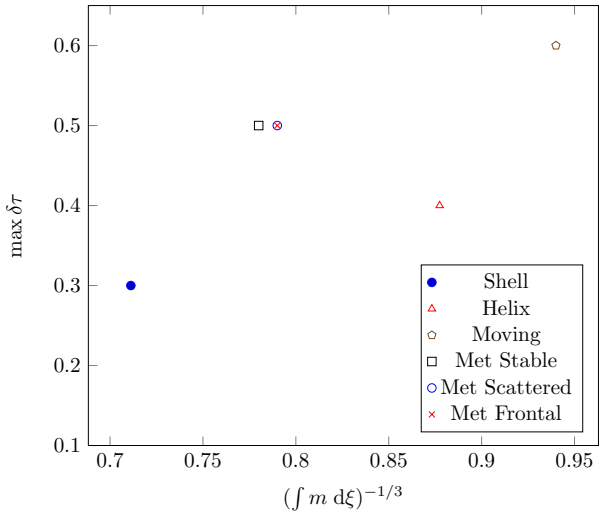


Figure 16: Maximum values of $\delta\tau$ as compared to the estimate $\delta\tau^*$ given in (23).

6 A moving mesh test problem

We now consider the performance of the PMA algorithm when used to compute a time varying three-dimensional mesh when the monitor function $m(\mathbf{x}, t)$ is itself a function of time. This situation of course is closer to a typical implementation of a mesh redistribution method when it would be used to as part of the solution

of a time varying PDE. In this section the example considered is the same as that studied by Chacón et al. [6] which also considers calculating a mesh by solving the Monge-Ampère equation, but which uses a Newton method coupled with a multi-grid solver to do this. To find the mesh in this case we implement Algorithm 2 as described earlier.

The time-varying, analytically defined, monitor function considered is given by:

$$m(x,y,z,t) = 1 + 4\exp\left(-r(x,y,z)^2\left(\frac{\cos^2(\kappa(x,y,z,t))}{\sigma_x^2} + \frac{\sin^2(\kappa(x,y,z,t))}{\sigma_y^2}\right)\right) \tag{29}$$

where $r(x,y,z)$ is the distance to the centre of the domain at $\left(\frac{1}{2},\frac{1}{2},\frac{1}{2}\right)$, $\sigma_x = \sqrt{0.05}$, $\sigma_y = \sqrt{0.001}$ are scaling factors and

$$\kappa(x,y,z,t) = \arctan\left(\frac{y-\frac{1}{2}}{x-\frac{1}{2}}\right) + 1.6\sin(\pi z)\max[(\tfrac{1}{2}-r)r, 0]t. \tag{30}$$

The goal of this test problem is to find meshes at times $t \in \{0, 1, \ldots, 100\}$. The problem of finding the mesh for this time dependent system is then solved in two stages in a manner analogous to the MMPDE method described in [13].

Firstly at time $t = 0$ Algorithm 2 sets the monitor function $m(\mathbf{x}, 0)$ and, starting from a uniform mesh, the system (15) is evolved forward in pseudo-time using Algorithm 1 with $m(\mathbf{x}, 0)$ fixed until the mesh satisfies the equidistribution condition to a high tolerance. For this calculation we take $\delta\tau = 0.1$, $\gamma = 0.2$ and $tol = 1E-05$.

Secondly Algorithm 2 evolves the monitor function in real time, with the value of t increased in intervals of $\delta t = 1.0$. For each of these outer timesteps, we set $\tau_{\max} = \delta t$ and $\delta\tau = \delta t/5$, ensuring at least 5 pseudo-timesteps per inner loop. The initial value of Q at each stage of the inner loop is given by the previously converged value for ther last time step.

The resulting mesh at the final times $t = 100$ for the case of a $128 \times 128 \times 128$ mesh is presented in Figure 17.

We can see at time $t = 100$ that the mesh closely follows the contours of the monitor function and is very regular with no hint of mesh tangling.

We next consider the computational cost of calculating these meshes. To do this the unit cube is discretised into a grid of $N \times N \times N$, where $N = 32, 64, 128, 192$, and we list the number of iterations to converge to the given tolerance in the pseudo-time calculation at $t = 0$ and the total CPU time required to compute the 101 meshes until $t = 100$. These results are presented in Table 2. We note that the scaling of the CPU time is fully consistent with an estimated complexity of $\mathcal{O}(N\log(N))$.

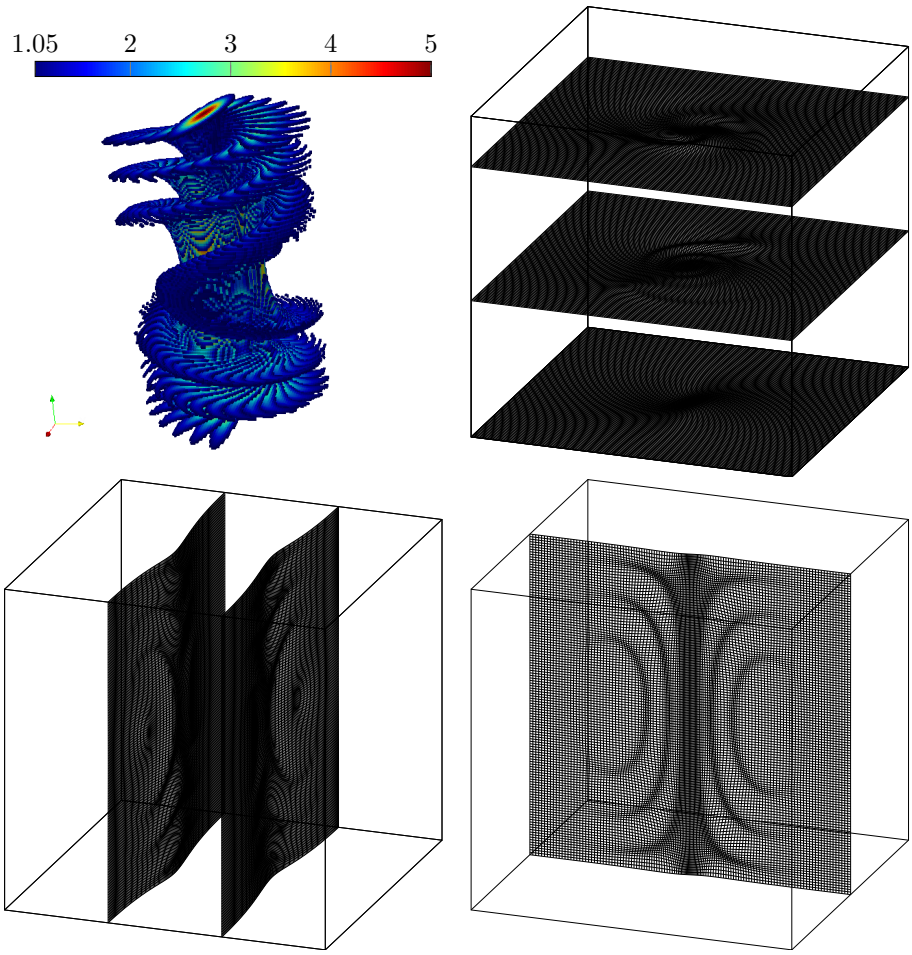
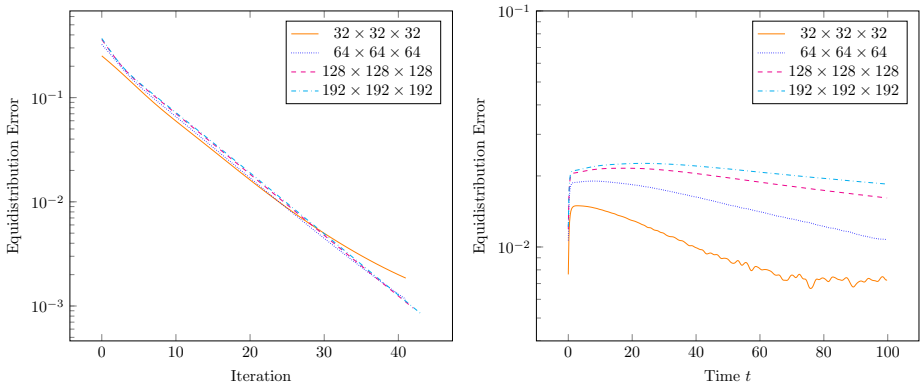


Figure 17: The monitor function and the resulting meshes at the time $t = 100$. Note that for all t, the monitor function ranges from 1 to 5.

Grid resolution N	DOFs	Initial iterations of static PMA	CPU(s)
$32 \times 32 \times 32$	98304	42	15.92099
$64 \times 64 \times 64$	786432	42	253.4290
$128 \times 128 \times 128$	6291456	43	2227.012
$192 \times 192 \times 192$	21233664	44	7604.040

Table 2: Timings for the evolution of the mesh to an equidistributed state for varying spatial discretisations



(a) Equidistribution error during pseudo-time convergence to mesh at $t = 0$

(b) Equidistribution error during real-time evolution of mesh

Figure 18: Equidistribution errors ε of the two-step process in redistributing a mesh for a dynamically evolving monitor function shown for various discretisation levels. Note the increase in the equidistribution errors in the dynamically evolving step, due to evolving the monitor function to $t + 1$ based on the value of the monitor function at time t.

7 Conclusion

In this paper we have demonstrated that the Parabolic Monge-Ampère algorithm can be extended from two dimensions to three, and that it is effective in generating meshes with good regularity in a short time. In particular it can deliver effective meshes for three dimensional meteorological data assimilation calculations using large data sets with 21 million degrees of freedom, in times commensurate with those required for actual weather forecasting. When applied to test problems it shows fast convergence, with meshes rapidly (and without any hint of tangling provided that the computational time step is taken sufficiently small) converging to an equidistributed state. We therefore think that this method should be considered seriously, alongside other techniques, as a fast and effective method for redistributing a large three dimensional mesh.

References

[1] Chiara Piccolo and Mike Cullen. A new implementation of the adaptive mesh transform in the Met Office 3D-Var System. *Quarterly Journal of the Royal Meteorological Society*, 138(667):1560–1570, July 2012.

[2] Jessica Gullbrand and Fotini Katopodes Chow. The effect of numerical errors and turbulence models in large-eddy simulations of channel flow, with and without explicit filtering. *Journal of Fluid Mechanics*, 495:323–341, November 2003.

[3] Weizhang Huang and Robert D Russell. A moving collocation method for solving time dependent partial differenial equations. *Applied Numerical Mathematics*, 20:101–116, 1996.

[4] Chiara Piccolo and Mike Cullen. Adaptive mesh method in the Met Office variational data assimilation system. *Quarterly Journal of the Royal Meteorological Society*, 137(656):631–640, April 2011.

[5] C.C. Pain, M.D. Piggott, A.J.H. Goddard, F. Fang, G.J. Gorman, D.P. Marshall, M.D. Eaton, P.W. Power, and C.R.E. de Oliveira. Three-dimensional unstructured mesh ocean modelling. *Ocean Modelling*, 10(1-2):5–33, January 2005.

[6] L. Chacón, G.L. Delzanno, and J.M. Finn. Robust, multidimensional mesh-motion based on MongeKantorovich equidistribution. *Journal of Computational Physics*, 230(1):87–103, January 2011.

[7] Jörn Behrens. Atmospheric and ocean modeling with an adaptive finite element solver for the shallow-water equations. *Applied Numerical Mathematics*, 26(1-2):217–226, January 1998.

[8] Hilary Weller. Predicting mesh density for adaptive modelling of the global atmosphere. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 367(1907):4523–4542, November 2009.

[9] Mark Ainsworth and Bill Senior. Aspects of an adaptive hp-finite element method: Adaptive strategy, conforming approximation and efficient solvers. *Computer Methods in Applied Mechanics and Engineering*, 150(1-4):65–87, December 1997.

[10] Jörn Behrens. *Adaptive atmospheric modeling: key techniques in grid generation, data structures, and numerical operations with applications*. Springer, 2006.

[11] C J Budd and J F Williams. Moving mesh generation using the parabolic Monge-Ampère Equation. *SIAM Journal on Scientific Computing*, 31(5):3438–3465, 2009.

[12] Chris J. Budd, Weizhang Huang, and Robert D. Russell. Adaptivity with moving grids. *Acta Numerica*, 18:1–131, May 2009.

[13] Weizhang Huang and Robert D Russell. *Adaptive moving mesh methods*, volume 174. Springer, 2011.

[14] G.L. Delzanno, L. Chacón, J.M. Finn, Y. Chung, and G. Lapenta. An optimal robust equidistribution method for two-dimensional grid adaptation based on MongeKantorovich optimization. *Journal of Computational Physics*, 227(23):9841–9864, December 2008.

[15] John M Finn, Gian Luca Delzanno, and Luis Chacón. Grid Generation and Adaptation by Monge-Kantorovich Optimization in Two and Three Dimensions. In *Proceedings of the 17th International Meshing Roundtable*, pages 551–568. Springer, 2008.

[16] C.J. Budd, M.J.P. Cullen, and E.J. Walsh. Monge-Ampére based moving mesh methods for numerical weather prediction, with applications to the Eady problem. *Journal of Computational Physics*, 236:247–270, March 2013.

[17] Weiming Cao. On the Error of Linear Interpolation and the Orientation, Aspect Ratio, and Internal Angles of a Triangle. *SIAM Journal on Numerical Analysis*, 43(1):19–40, January 2005.

[18] Weizhang Huang, Yuhe Ren, and Robert D. Russell. Moving Mesh Partial Differential Equations (MMPDES) Based on the Equidistribution Principle. *SIAM Journal on Numerical Analysis*, 31(3):709–730, 1994.

[19] Yann Brenier. Polar Factorization and Monotone Rearrangement of Vector-Valued Functions. *Communications on Pure and Applied Mathematics*, XLIV:375–417, 1991.

[20] M J Sewell. Some applications of transformation theory in mechanics. *Large Scale Atmosphere–Ocean Dynamics*, 2:143–223, 2002.

[21] Brittany Dawn Froese. *Numerical Methods for the Elliptic Monge-Ampère Equation and Optimal Transport*. Phd, Simon Fraser University, 2012.

[22] M.J.P. Cullen. *A Mathematical Theory of Large-Scale Atmosphere/Ocean Flow*. Imperial College Press, 2006.

[23] S. Chynoweth and J Sewell. A consise derivation of the semi-geostrophic equations. *Quarterly Journal of the Royal Meteorological Society*, 117(502):1109–1128, 1991.

[24] Hector D. Ceniceros and Thomas Y. Hou. An Efficient Dynamically Adaptive Mesh for Potentially Singular Solutions. *Journal of Computational Physics*, 172(2):609–639, September 2001.

[25] C J Budd and V A Galaktionov. On Self-similar blow-up in evolution equations of Monge-Ampère Type: A view from Reaction-Diffusion Theory. *IMA Journal of Applied Mathematics*, 2011.

[26] Matteo Frigo and Steven G. Johnson. The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

[27] Michael Pippig. Pfft: an extension of fftw to massively parallel architectures. *SIAM Journal on Scientific Computing*, 35(3):213–236, 2013.

[28] Mohamed Sulman, J.F. Williams, and R.D. Russell. Optimal mass transport for higher dimensional adaptive grid generation. *Journal of Computational Physics*, 230(9):3302–3330, May 2011.