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

August 6, 2013

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 Knowledge Transfer Grant XXX-XXXX-XXXX.

1 Introduction

1.1 Overview

Many physical problems exhibit 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 Piccolo2012 [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 calculation then this may lead to large truncation errors [2]. In the data assimilation context, an adaptive mesh is a convenient way of representing spatially varying correlation structures which would otherwise have to be represented by an unmanageably large correlation matrix. 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, 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 points are optimally placed to resolve the fine-scale features of interest. 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], 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 [15], 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],[15], 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 [16], [?], [?]. 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 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. In this implementation the calculation of a three dimensional meteorological grid with 21772800 degrees of freedom could be accomplished in five 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 effective 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.

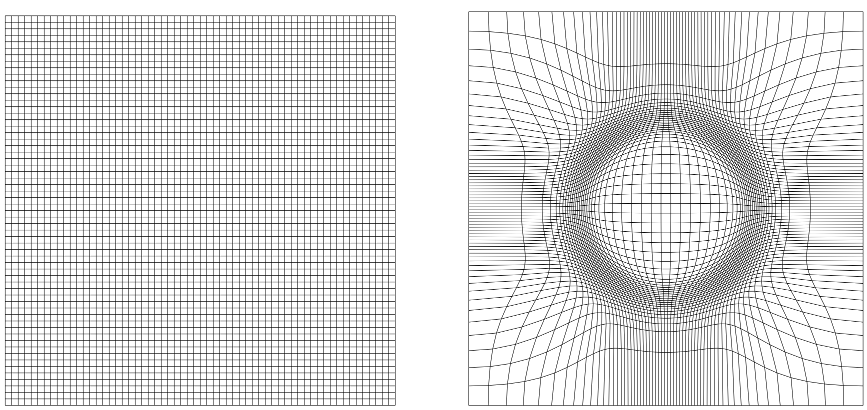
2 Adaptive mesh redistribution in three dimensions

amr

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 $x = (x, y, z)$

radaptexample

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

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

then

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

We call this the *equidistribution equation.* 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 [17]. 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 [18]). 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], [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 [18]). *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 $\mathbf{x} = \nabla P$ 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 [19] 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 [15] for an example.

3 Parabolic Monge-Ampère formulation

Equation (6) is a fully non-linear elliptic PDE which is challenging to solve. There is a significant literature describing various solution techniques both for the equation in its own right [?], as part of a meteorological calculation [20, 21] 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 a Newton-type algorithm. In [6] the resulting linear equations are in turn solved using a multi-grid method. These methods are necessarily computationally expensive and use significant computer time to give an accurate answer. However, in the context of mesh generation, we do not want to invest much effort in solving (6) as its function 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, a simple, explicit, relaxation method for solving (6) has certain advantages in this context. Firstly it can be implemented cheaply using a Forward Euler method, and the mesh calculated rapidly. Secondly, the relaxation method can be terminated at any time when the mesh generated is sufficient for subsequent computations rather than when we have an exact solution of (6). This gives a very significant speed increase in the mesh generation algorithm (with times reduced from hours to minutes). In two-dimensions it has been demonstrated [11], [15], that 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. 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 $\nabla_\xi Q \to \nabla P$ as $\tau \to \infty$ where P solves (6).

Definition 2 (PMA)**.** The Parabolic Monge-Ampère equation in $d-$dimensions is defined by

$$LQ_\tau \equiv (I - \gamma \Delta_\xi) Q_\tau = (\hat{m}(\nabla_\xi Q) |H(Q)|)^{\frac{1}{d}} \tag{9}$$

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].)

In this equation the application of L^{-1} acts as a smoothing operator (described first in [22]) which leads to more regular meshes. Furthermore the action of L^{-1} on the discrete form of the right hand side of (9) acts to damp out certain (mesh dependent) chequer-board instabilities [23]. 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 (9) scale linearly. This is useful both to ensure global existence of the solutions of (9) and to give it certain desirable scaling properties [11]. It is further shown in [11] that the equation (9) is *locally stable* so that, if ∇Q is sufficiently close to ∇P then $\nabla_\xi Q \to \nabla P_\xi \quad \text{as} \quad \tau \to \infty.$ with standard linear convergence. Furthermore, during the evolution of (9) both $H(P)$ and $\nabla^2 Q$ are bounded away from zero. This prevents mesh tangling provided that the equation (9) has a sufficiently fine discretisation [11].

The evolutionary system (9) 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

$$\tilde{Q} = Q - \frac{|\xi|^2}{2}. \tag{10}$$

This gives

$$\nabla_\xi \tilde{Q} = \nabla_\xi Q - \xi \tag{11}$$

and hence

$$\mathbf{x} = \nabla_\xi \tilde{Q} + \xi \tag{12}$$

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

$$(I - \gamma\Delta_\xi)\tilde{Q}_\tau = (\hat{m}(\nabla_\xi \tilde{Q} + \xi)|I + H(\tilde{Q})|)^{\frac{1}{d}} \tag{13}$$

tildeQ

In the absence of a better initial guess, we use the initial conditions for (13) $\tilde{Q}(0) = 0$. 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 (13) 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 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 (13) forwards in pseudo-time on the assumption that the monitor function is known a-priori. In our examples we will consider cases where m is fixed and also where m evolves in time.

4 Implementation

When implementing (13) 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.

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 (9) to evolve $\tilde{Q}$ so that

$$\tilde{Q}(\tau + \delta\tau) = \tilde{Q}(\tau) + \delta\tau \tilde{Q}_\tau(\tau) \tag{14}$$

model_evo

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

$$\tilde{Q}_\tau = L^{-1}(\hat{m}(\nabla_\xi \tilde{Q} + \xi)|I + H(\tilde{Q})|)^{\frac{1}{d}}. \tag{15}$$

mesh_evo

To compute the RHS of (15) we discretise the Hessian operator in (15). 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.

PMAalgo_static

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

Read initial mesh $\xi = (\xi, \eta, \zeta)$

$\tau \leftarrow 0$

Initialise $\tilde{Q}(\tau) = \tilde{Q}_0$

Store the grid $\mathbf{x}(\tau) = (x(\tau), y(\tau), z(\tau))$ as

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

while $r > tol$ & $\tau < \tau_{\max}$ **do**

Compute $\tilde{Q}_\tau(\tau)$ via:

- Compute the monitor function at the current grid points $m(\mathbf{x}(\tau))$. This may be analytically defined of 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

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

Take a Forward Euler step

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

Compute the finite difference approximations to $\frac{\partial \tilde{Q}(\tau)}{\partial \xi}$, $\frac{\partial \tilde{Q}(\tau)}{\partial \eta}$ and $\frac{\partial \tilde{Q}(\tau)}{\partial \zeta}$

Store the new grid as

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

Compute the change in the mesh

$$r \leftarrow ||\nabla_\xi \tilde{Q}(\tau + \delta\tau) - \nabla_\xi \tilde{Q}(\tau)||_2$$

$\tau \leftarrow \tau + \delta\tau$

end while

For completeness we now augment Algorithm 1 with an outer loop that is applied in the case of a time-dependent monitor function, giving Algorithm 2.

PMAalgo_dynamic

Algorithm 2 The PMA algorithm in 3D for a dynamic monitor function

$t \leftarrow 0$
Apply Algorithm 1 with $m(\mathbf{x}) = m(\mathbf{x}, 0)$, $\tilde{Q}_0 \equiv 0$ and $\tau_{\max} = \infty$
while $t < t_{\max}$ **do**
    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
    $t \leftarrow t + \delta t$
end while

Now we elaborate on the details of the algorithms to show how the PMA method can be implemented in practice in 3 dimensions for a problem in which 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 with, respectively, n_x, n_y and n_z cubes in the the three coordinate directions, of corresponding side lengths h_x, h_y and h_z.

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

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

At the boundaries we invoke the Neumann boundary conditions so that

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

Derivatives with respect to other variables follow similarly.

4.2 Second order differentiation

Firstly, we note that the enforcing the Neumann boundary conditions implies that all mixed derivatives on the boundary are zero, i.e. on the boundary

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

For the non-mixed derivatives on the boundary, a simple forward (or backward) scheme is used, so that for example

$$\tilde{Q}_{\eta\eta}(:,1,:) = \frac{\tilde{Q}(:,1,:) - 2\tilde{Q}(:,2,:) + \tilde{Q}(:,3,:)}{4h_y^2}.$$

In the interior of the domain, central differencing is employed such that

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

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

$$\begin{aligned}\tilde{Q}_{\xi\zeta}(i,:,k) = \frac{1}{4h_x h_z}(&\tilde{Q}(i+1,:,k) - \tilde{Q}(i-1,:,k) - \\ &\tilde{Q}(i+1,:,k-1) + \tilde{Q}(i-1,:,k-1))\end{aligned}$$

for all $i \in \{2, \ldots, n_x - 1\}$ and $k \in \{2, \ldots, n_z - 1\}$.

Similar approximations can be used for the other second order derivatives of $\tilde{Q}$.

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:

$$\hat{m}(i,j,k) = \frac{\sum_{\ell_1=-1}^{1}\sum_{\ell_2=-1}^{1}\sum_{\ell_3=-1}^{1} m(i+\ell_1, j+\ell_2, k+\ell_3)\beta^{|\ell_1|+|\ell_2|+|\ell_3|}}{\sum_{\ell_1=-1}^{1}\sum_{\ell_2=-1}^{1}\sum_{\ell_3=-1}^{1} \beta^{|\ell_1|+|\ell_2|+|\ell_3|}}. \tag{16}$$

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 data assimilation contexts is as follows:

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

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 (9) 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 [24] 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

$$1/(1+\gamma(k_x^2+k_y^2+k_z^2)). \tag{18}$$

where the frequency-space coefficients k_x, k_y and k_z are 3D vector fields given by

$$k_x(i,j,k)=\frac{i-1}{n_x-1}\pi n_x, \quad k_y(i,j,k)=\frac{j-1}{n_y-1}\pi n_y, \quad \& \quad k_z(i,j,k)=\frac{k-1}{n_z-1}\pi n_z$$

for all $i\in\{1,\ldots,n_x\}$, $j\in\{1,\ldots,n_y\}$ and $k\in\{1,\ldots,n_z\}$. 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.

4.5 Choice of the tuning parameters

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 initially determine appropriate values for the three parameters used in the static case (Algorithm 1), namely $\delta\tau$ and γ. In this static case, 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. If it is too small then the system becomes overly stiff. This parameter can be controlled adaptively, however it is generally robust to being set at a small constant value. Noting that the intrinsic time-scale of this system is given by $m^{-1/d}$ a robust choice is to take

$$\delta\tau=\epsilon m^{-1/d}$$

where ϵ is a small constant value typically in the range $0.1\le\epsilon\le 1$. (This choice also has certain useful features when scaling symmetries act on the system [12].)

The parameter γ appears in the smoothing operator $L \equiv (I - \gamma\Delta_\xi)^{-1}$ as part of equation (15) and is applied in (18). 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. 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, δt corresponds to the natural time-scale of the model (ie. 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.

5 Static mesh results

We now present a series of examples chosen to demonstrate the performance of the PMA algorithm on various 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 and 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 $\tilde{Q}_0 = 0$.

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.

For all of the examples, the codes for the PMA algorithm were executed on a laptop with an Intel® Core$^{\mathrm{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

sec: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

$$s(\mathbf{x}) = s(x,y,z) = \sqrt{(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2}. \tag{19}$$

We then define the density of the ball via the function

$$f(\mathbf{x}) = f(x,y,z) = \begin{cases} 1 & \text{for } s(x,y,z) \le r_1 \\ \frac{1}{2}\cos\left(\frac{(s(x,y,z)-r_1)\pi}{r_2}\right) + \frac{1}{2} & \text{for } s(x,y,z) \le r_1 + r_2 \\ 0 & \text{for } s(x,y,z) > r_1 + r_2 \end{cases} \tag{20}$$

cartball

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

$$m(x,y,z) = \sqrt{(1 + c^2(f_x(x,y,z)^2 + f_y(x,y,z)^2 + f_z(x,y,z)^2))}. \tag{21}$$

ballmonitor3d

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

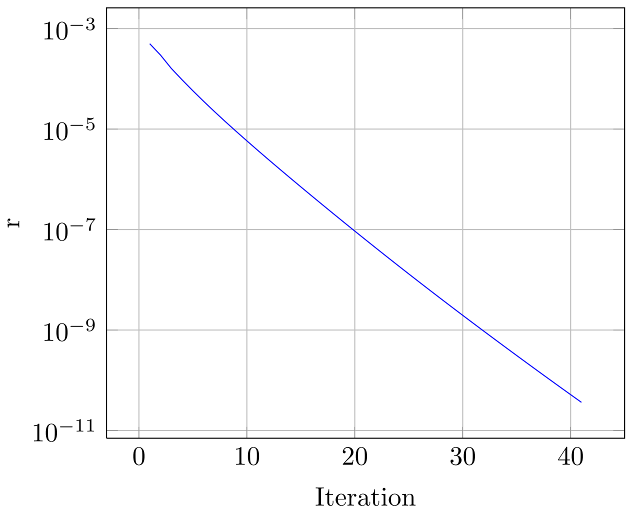


Figure 2: Plot of distance r that the mesh moved in each iteration (The Euclidean change in the mesh) for the shell problem.

conv_ball

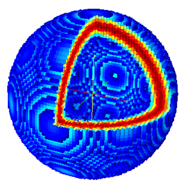
the width of the ball to be $r_1 = r_2 = \frac{1}{6}$, and centred in the domain so that

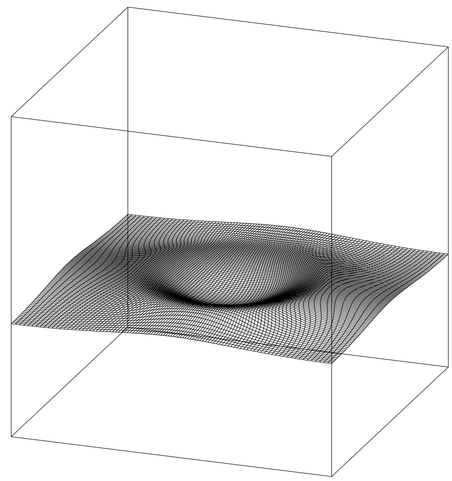
$$(x_0, y_0, z_0)^T = (\tfrac{1}{2}, \tfrac{1}{2}, \tfrac{1}{2})^T.$$

In the examples shown the computational domain $\Omega_C = [0,1]^3$ is split into a grid of $n_x \times n_y \times n_z$ points, with $n_x = n_y = n_z = 100$ 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 $5E - 11$ is shown in Figure 2. The calculation terminated after 41 iterations, taking 34 seconds on the laptop computer described earlier. From this figure we can clearly see the rapid, linear convergence of the algorithm.

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 ball. In Figure 3a we plot the values of the monitor function in three dimension, with part of the ball 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.



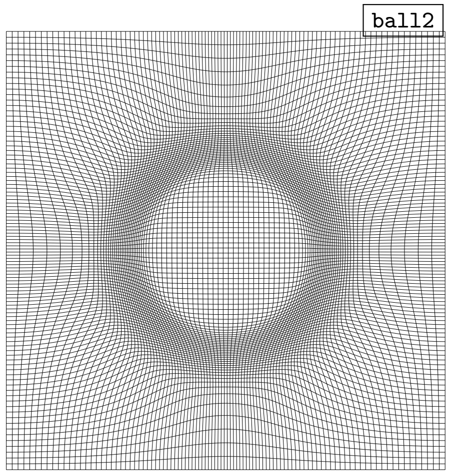


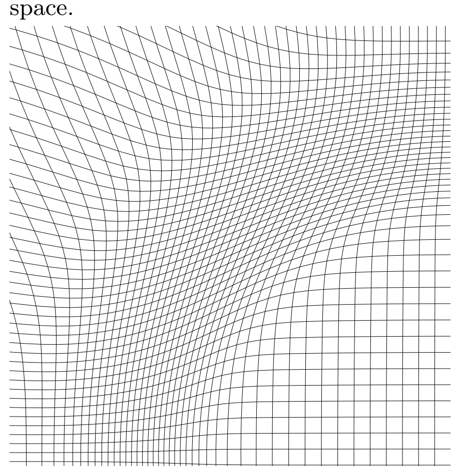
(a) 3D plot of the monitor function for $m > 1.05$

ball1

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

ball2





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

ball3

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

ball4

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

ballmesh

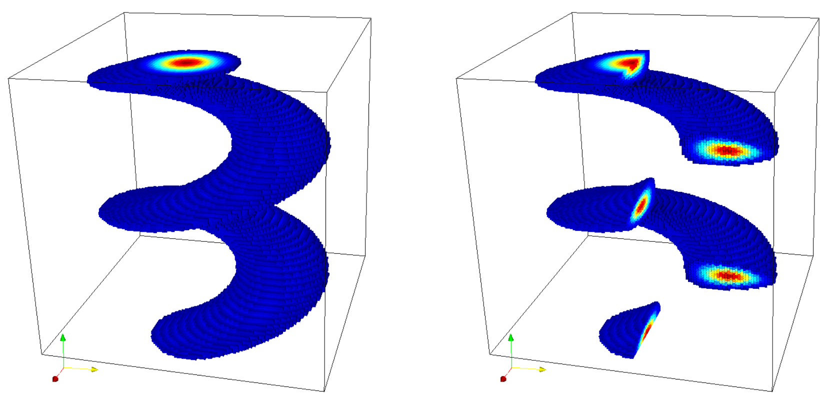
5.1.2 Example 2: A three dimensional helix

We next consider an analytically defined monitor function that describes a complex three dimensional helical surface without the symmetries of the shell. Taking $\mathbf{x} = (x, y, z)^T$ then a monitor function $m(\mathbf{x})$ which is large in a neighbourhood of such a helix is given by

$$m(x, y, z) = \exp(-w_1[(x - (w_2 \cos(4z\pi) + 0.5))^2 + (y - (w_2 \sin(4z\pi) + 0.5))^2]) \quad (22)$$

helixm

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 dimensional values of the monitor function are shown in Figure 4.



(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 $m > 0.05$

helix_mon

The PMA algorithm was applied to the helical problem with $\delta\tau = 0.1$ and $\gamma = 0.2$ and 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 non-monotone convergence of the mesh to an equidistributed state to a tolerance of $5E-11$ is shown in Figure 5. The calculation terminated after 473 iterations, taking 7.9 minutes on the laptop computer described earlier. We note that this convergence is significantly slower than for the shell in the example in Section 5.1.1. In Figure 6 we show the mesh generated by the PMA algorithm when applied to the problem taking m as defined in (22). In Figure 6b we show where the two horizontal planes in Figure 6a are mapped to in physical space.

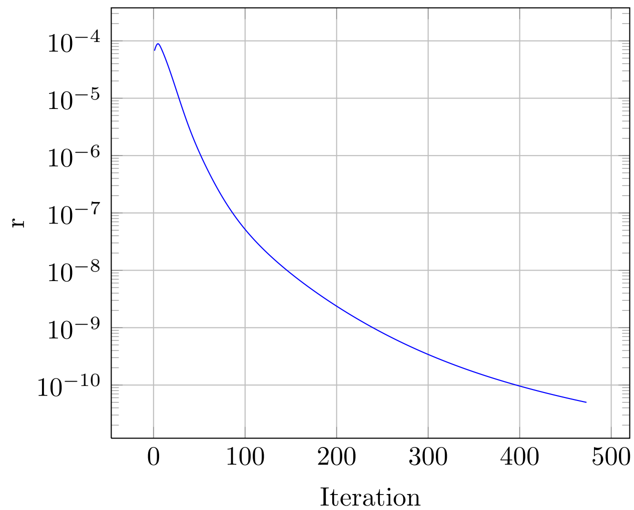
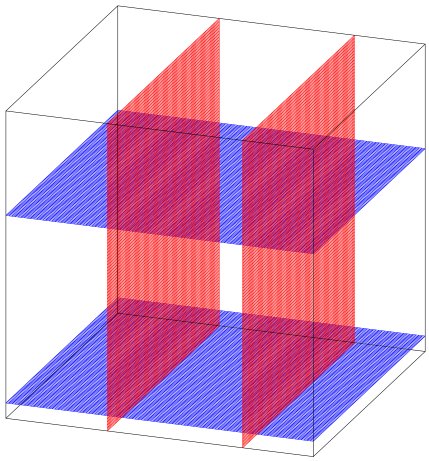


Figure 5: Plot of distance the mesh moved in each iteration (Euclidean change in mesh) for the helical problem.

conv_helix

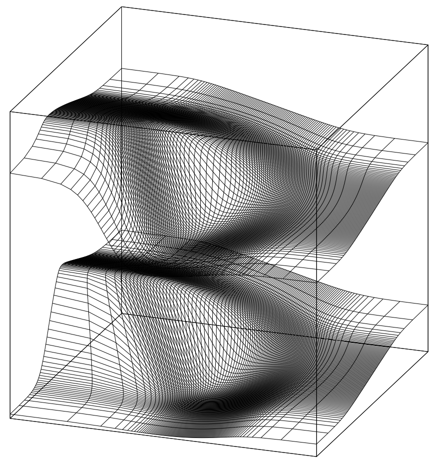
Similarly Figures 6c and 6d show where the vertical planes in Figure 6a are mapped to in physical space. These show that the redistributed grid is closely following the monitor function and very clearly show the fully 3D nature of the problem.

For this helical problem, compared with the other examples considered in this paper, a substantially larger number of iterations are required in order to have a suitable mesh and for the algorithm to converge. This is due to the complex and highly non-uniform structure of the monitor function which we are considering and the fact that the original uniform mesh has to encounter significant deviation to wrap around the helical structure. Due to the twisted nature of the monitor function, if we were to be aggressive with our strategy to find a mesh for this, mesh tangling would easily occur on the path to the final mesh. To avoid this we decreased our step size ($\delta\tau$) in the forward Euler method and hence we increase the number of iterations required to reach the same tolerance as considered in the other examples. It is a significant achievement of the algorithm that it was able to evolve the mesh in this manner without ever encountering a state where the mesh was tangled.



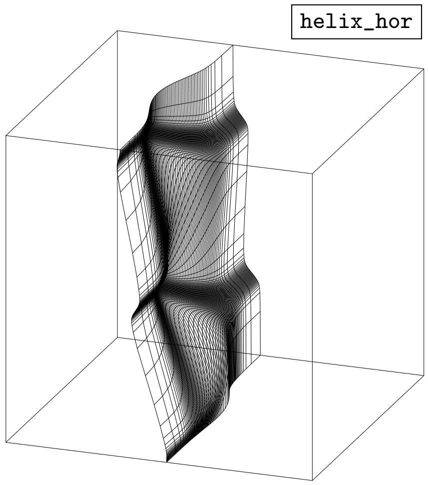
(a) Planes in the computational mesh showing where the meshes in Figures 6b–6d originate in computational space

helix_planes



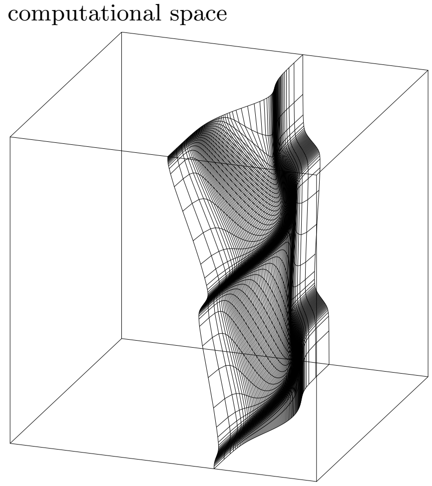
(b) Location of the two horizontal planes in the physical space corresponding to the horizontal planes shown in computational space

helix_hor



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

helix_ver1



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

helix_ver2

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

helix_res

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.

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 to determine an atmospheric state consistent both with observations and with the underlying physics of the atmosphere. In order to implement 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 Piccolo2011,Piccolo2012 [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 Piccolo2011,Piccolo2012 [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, 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. As a rule of thumb, a mesh which takes more than five minutes to generate is not useful operationally. 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, and thus to ease the finding of the minimum in the variational problem as in Piccolo2012,Piccolo2011 [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 [20]. 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 [20]. 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

$$\tilde{m}_{i,j,k} = \frac{\sum_{\ell_1=-1}^{1} \sum_{\ell_2=-1}^{1} m_{i+\ell_1, j+\ell_2, k} \beta^{|\ell_1|+|\ell_2|}}{\sum_{\ell_1=-1}^{1} \sum_{\ell_2=-1}^{1} \beta^{|\ell_1|+|\ell_2|}} \tag{23}$$

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. 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 $5E - 11$. 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 shows the convergence results from the three test cases.

Test case	Iterations	CPU time (minutes)
Stable boundary layer	22	4.0
Scattered showers	22	4.2
Frontal system	21	5.4

tab:met

Table 1: Results for the three meteorological test cases

Observe, that even in these large data sets, the PMA algorithm converges rapidly. 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.

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 7 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 greater than one and further apart when it is smaller than one. 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 8. Again the mesh (latitudes versus height levels) follows the structure of the corresponding monitor function and captures local variability both vertically and horizontally.

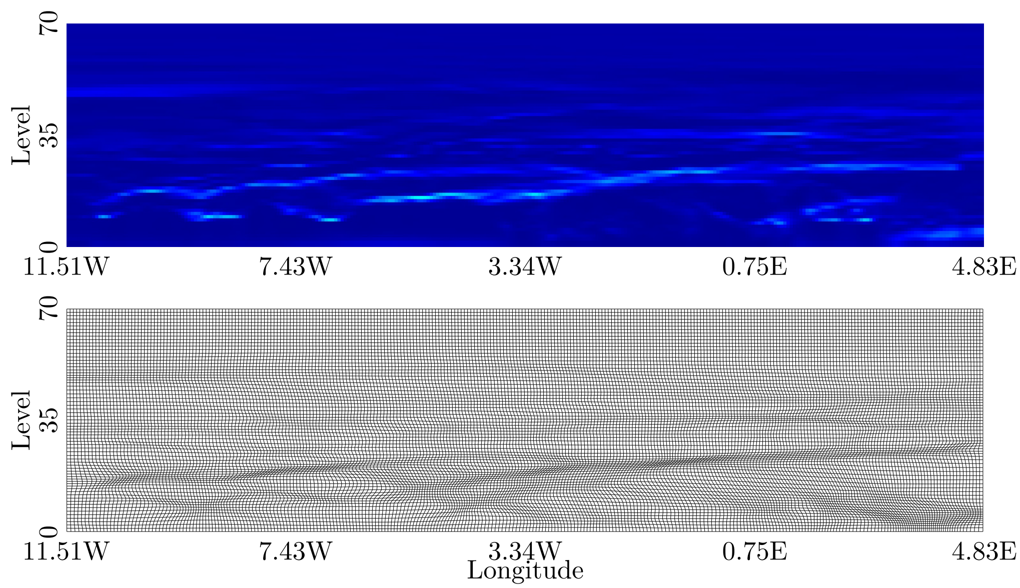


Figure 7: The monitor function and the resulting mesh for the stable boundary layer system at a 94^{th} latitude increment ad with increasing longitude. The function is shown in the vertical plane from $(50.68N, 11.51W)$ to $(50.80N, 4.84E)$

fig7

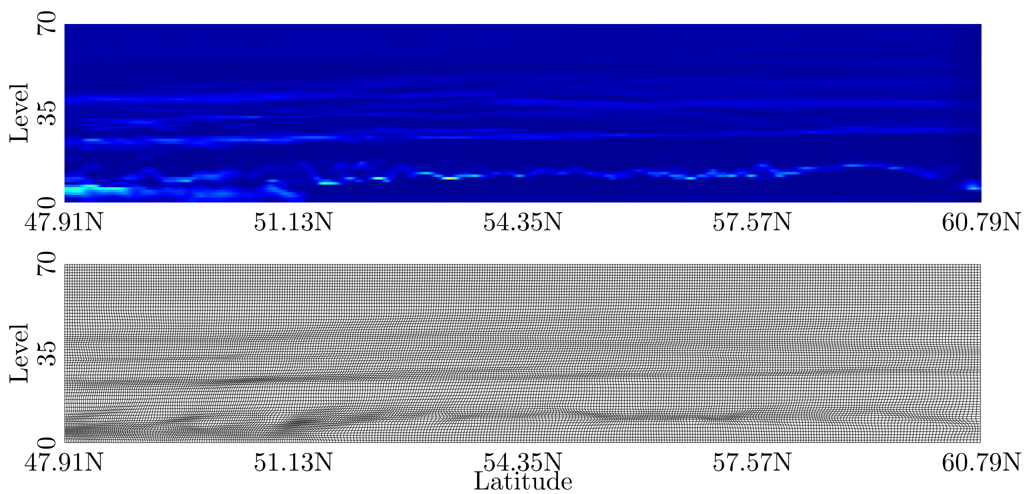


Figure 8: The monitor function and the mesh for the stable boundary layer system at a 260^{th} longitude increment. In the vertical plane from $(47.91N, 2.89E)$ to $(60.79N, 4.86E)$.

fig8

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 monitor function. The convective activity is shown by the radar image in Figure 9. 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 10 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 9, this is because the monitor function is calculated from a T+3h forecast and not from 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 11 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.

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 12 shows the radar image for the front system on the 25 April 2012 at 03UTC. The horizontal cross section of the monitor function and the corresponding mesh for this case are shown in Figure 13. The front is clearly depicted in both pictures and the refinement of the mesh is high in correspondence with the front. Figure 14 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.

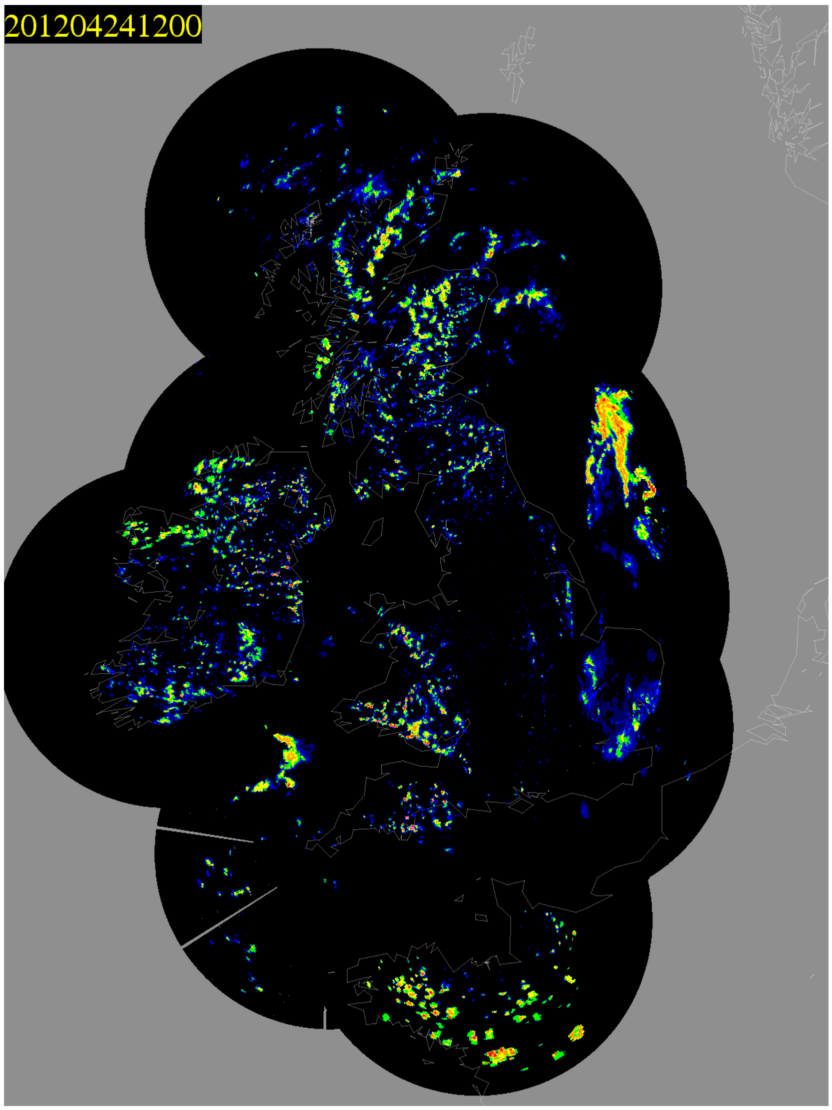


fig9

Figure 9: Radar image of the scattered showers system.

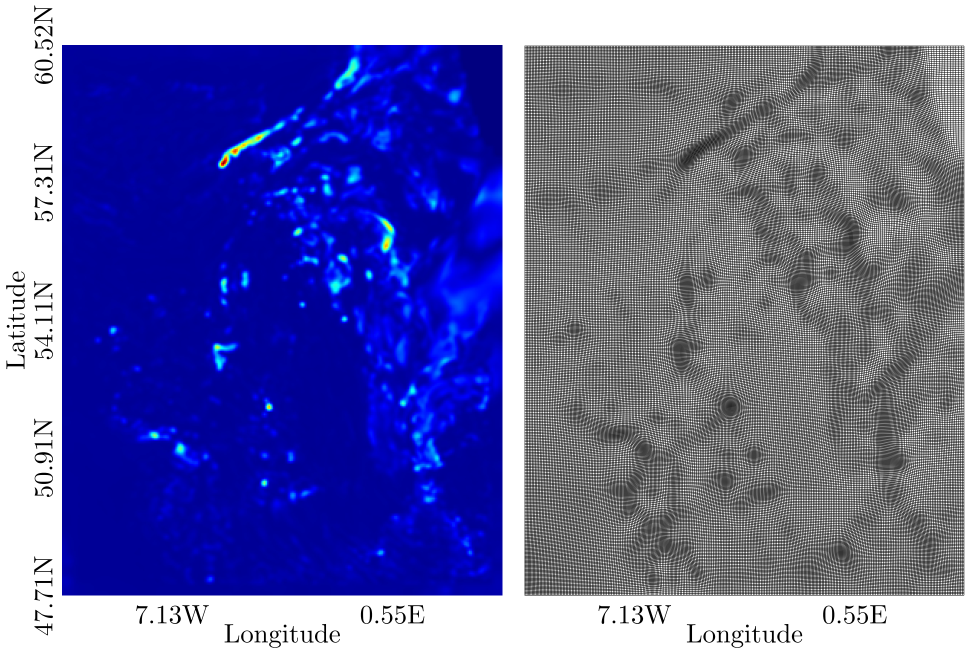


Figure 10: The monitor function and the mesh for the scattered shower system at a 8$^{\mathrm{th}}$ vertical level. At 261.7m

fig10

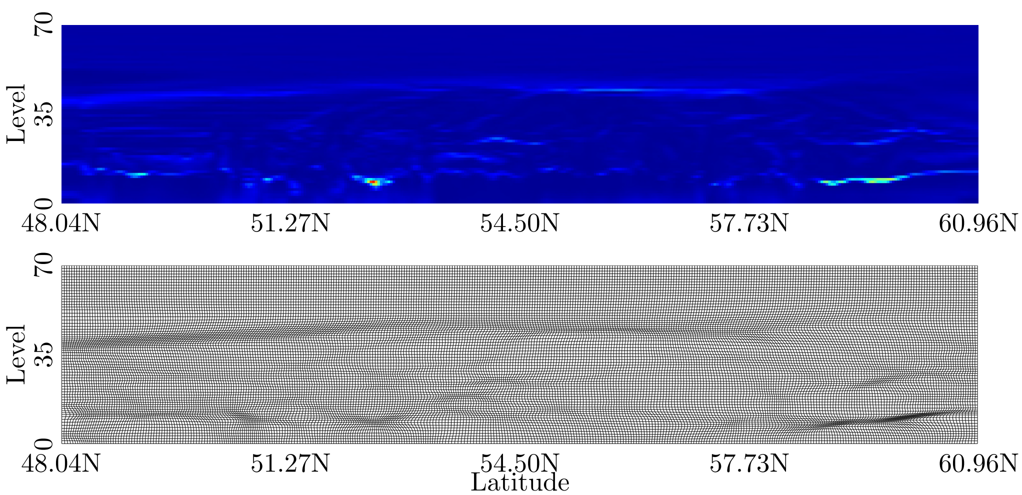


Figure 11: The monitor function and the mesh for the scattered shower system at a 135$^{\mathrm{th}}$ longitude increment. Vertical plane from $(48.04N, 3.81W)$ to $(60.96N, 4.29W)$.

fig11

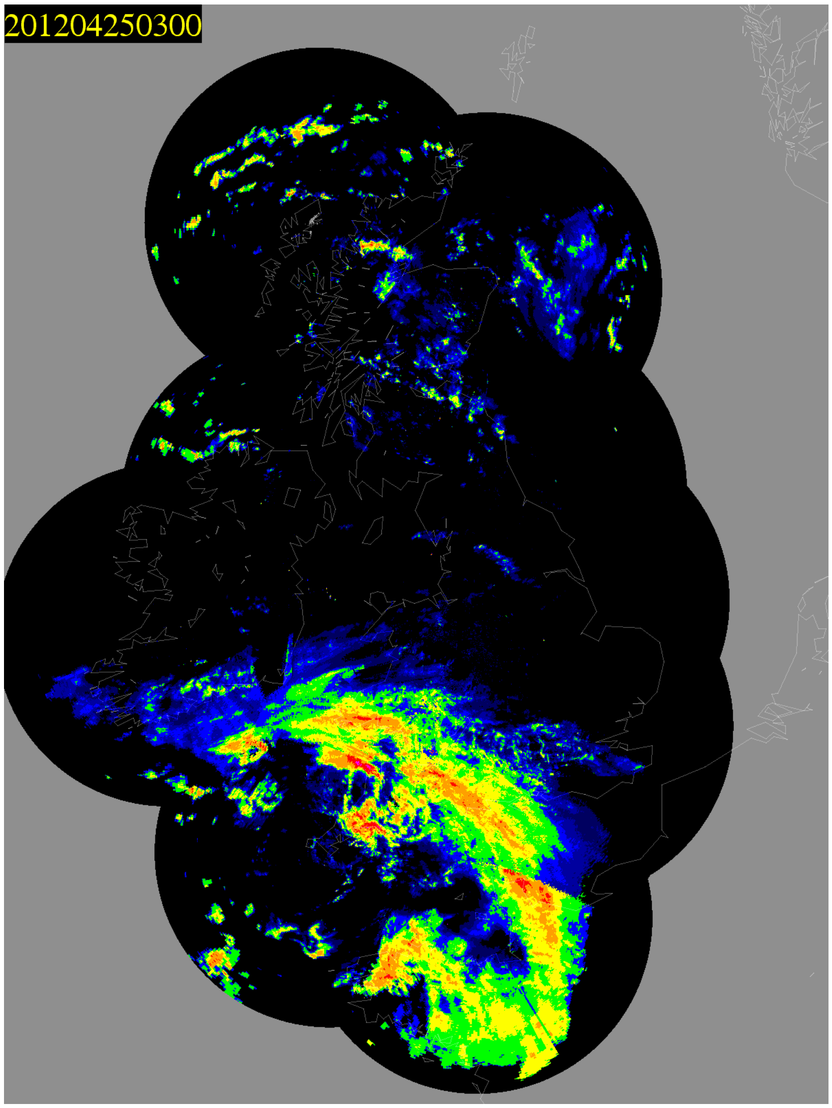


fig12

Figure 12: The radar image of the frontal system crossing the South West coast of the British Isles

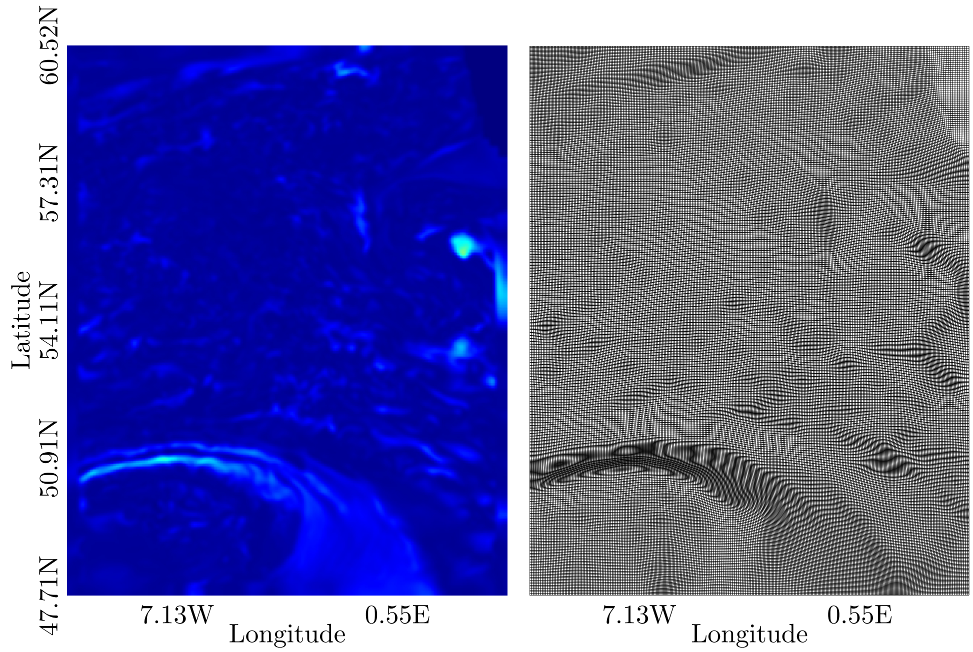


Figure 13: The monitor function and the mesh of the frontal system at a 23$^{\text{rd}}$ vertical level at 1911.7m

fig13

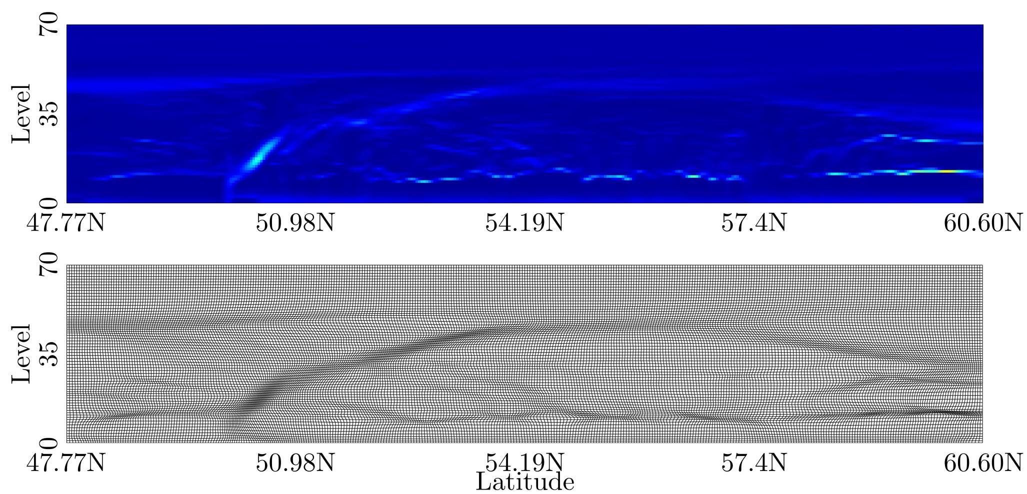


Figure 14: Monitor function and mesh of frontal system at 16$^{\text{th}}$ longitude increment. Vertical plane from $(47.77N, 10.17W)$ to $(60.60N, 12.94W)$.

fig14

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:

$$m(x,y,z,t) = 1 + 4\exp\left(-r(x,y,z)^2\left(\frac{\cos^2(\kappa(x,y,z,t))}{\sigma_x^2} + \frac{\sin^2(\kappa(x,y,z,t))}{\sigma_y^2}\right)\right) \tag{24}$$

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

$$\kappa(x,y,z,t) = \arctan\left(\frac{y-\frac{1}{2}}{x-\frac{1}{2}}\right) + 1.6\sin(\pi z)\max[(\tfrac{1}{2} - r)r, 0]t. \tag{25}$$

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].

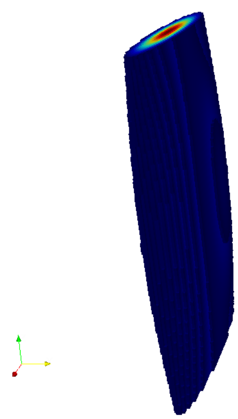
Firstly at time $t = 0$ Algorithm 2 sets the monitor function $m(\mathbf{x}, 0)$ and, starting from a uniform mesh, the system (14) is evolved forward in pseudo-time using Algorithm 1 with $m(\mathbf{x}, 0)$ fixed until the mesh satisfies the equidistribution condition to a high tolerance. For this calculation we take $\delta\tau = 0.1$, $\gamma = 0.2$ and $tol = 5E - 11$.

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.

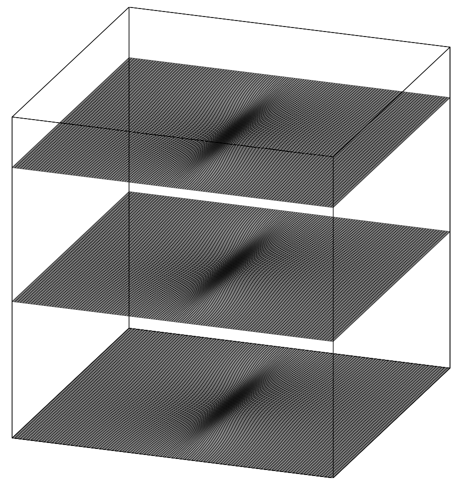
Some of the resulting meshes for the case of a $128 \times 128 \times 128$ mesh are presented as follows. In Figure 15 we show the monitor function and the resulting mesh at the initial time $t = 0$. In Figures 16 and 17 we then show the evolved meshes at the later times $t = 50$ and $t = 100$.

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.

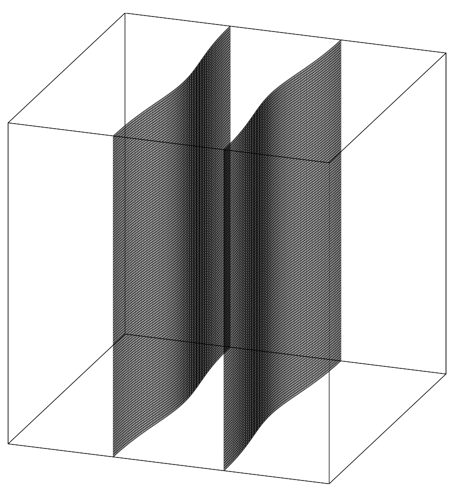
We next consider the computational cost of calculating these meshes. To do this the unit cube is discretised into a grid of $N \times N \times N$, where $N = 32, 64, 128$,



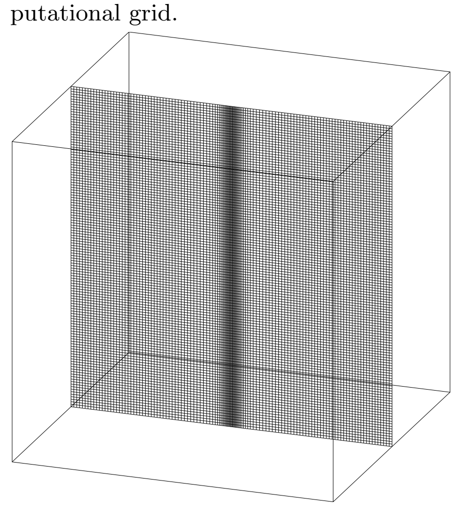
(a) The monitor function at time $t = 0$



(b) Planes $z \in \{0, \frac{54}{127}, \frac{107}{127}\}$ in the computational grid.



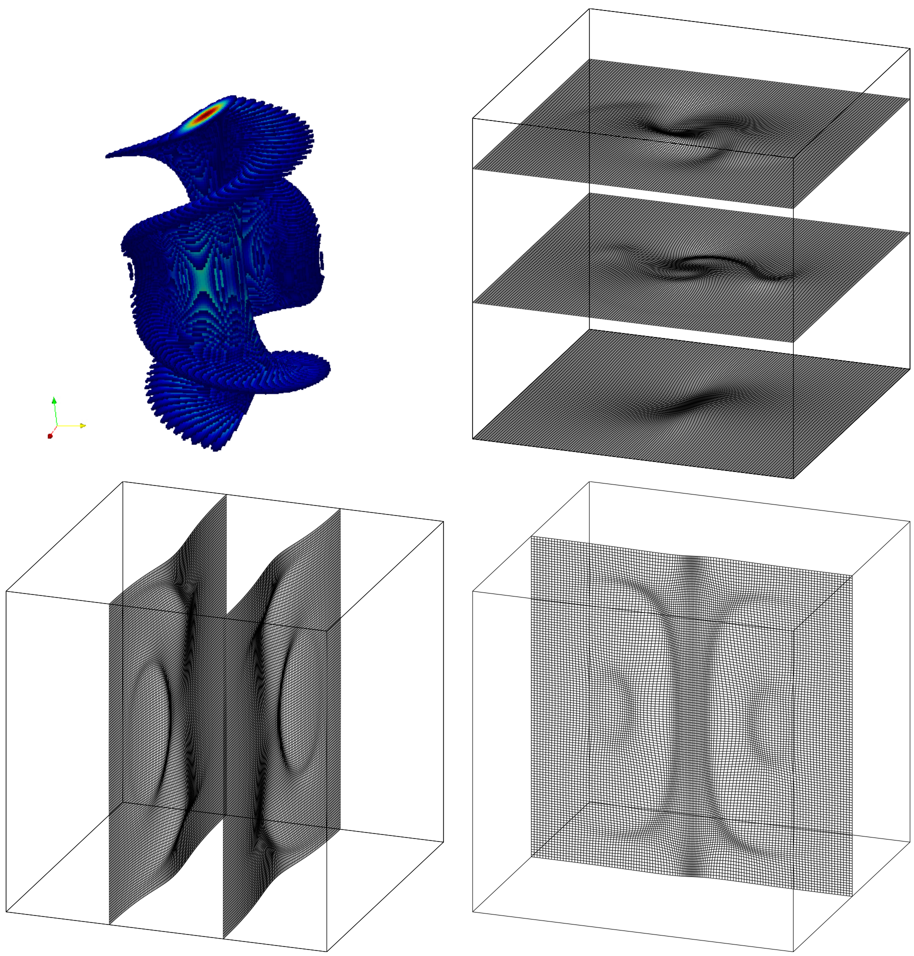
(c) Planes $y \in \{\frac{40}{127}, \frac{87}{127}\}$ in the computational grid.



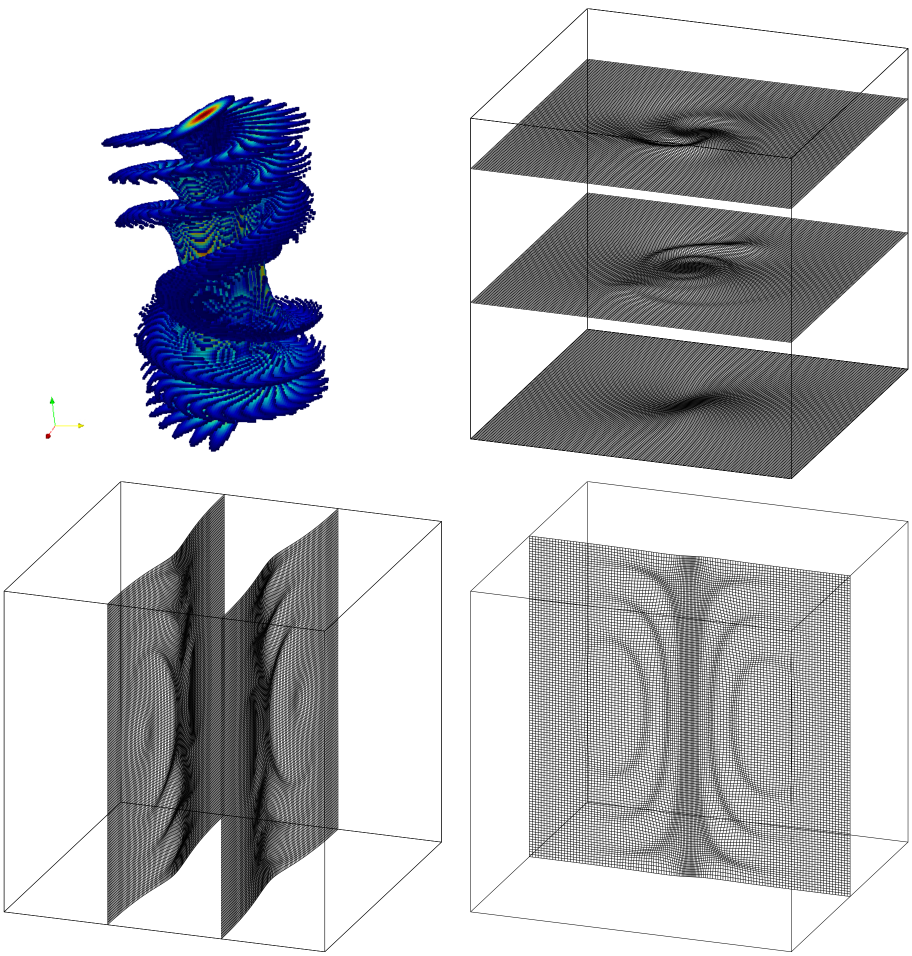
(d) Planes $x \in \{\frac{64}{127}\}$ in the computational grid.

cfig1

Figure 15: The monitor function and the resulting meshes at the time $t = 0$



cfig2 Figure 16: The monitor function and the resulting meshes at the time $t = 50$



cfig3 Figure 17: The monitor function and the resulting meshes at the time $t = 100$

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.

Grid resolution N	DOFs	Initial iterations of static PMA	CPU(s)
$32 \times 32 \times 32$	98304	44	15.91
$64 \times 64 \times 64$	786432	45	180.28
$128 \times 128 \times 128$	6291456	44	2356.29

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

A direct comparison with the results in Chacón et. al. [6] can be made with these results. In their paper they describe and implement a Newton-Krylov iteration using modern multigrid methods to solve exactly the fully non-linear Monge-Ampère equation. For the high resolution $128 \times 128 \times 128$ grid this computation required 32000s of CPU time. The methods produced by the PMA algorithm appear similar in structure, despite the significantly reduced cost of their calculation.

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 rapid convergence, with meshes rapidly (and without any hint of tangling) converging to an equidistributed state. In particular it is an order of magnitude faster in converging than other similar mesh generation methods. We therefore think that this method should be considered seriously as a fast and effective method for redistributing a large three dimensional mesh.

References

Piccolo2012 [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.

Gullbrand2003 [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.

Huang1996 [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.

Piccolo2011 [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.

Pain2005 [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.

Chacon2011 [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.

Behrens1998 [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.

Weller2009 [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.

Ainsworth1997 [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.

behrens2006 [10] Jörn Behrens. *Adaptive atmospheric modeling: key techniques in grid generation, data structures, and numerical operations with applications*. Springer, 2006.

Budd2009a [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.

Budd2009 [12] Chris J. Budd, Weizhang Huang, and Robert D. Russell. Adaptivity with moving grids. *Acta Numerica*, 18:1–131, May 2009.

huang2011 [13] Weizhang Huang and R Robert D Russell. *Adaptive moving mesh methods*, volume 174. Springer, 2011.

Delzanno2008 [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.

Budd2013 [15] 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.

Cao2005 [16] 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.

Huang1994a [17] 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.

Brenier1991 [18] Yann Brenier. Polar Factorization and Monotone Rearrangement of Vector-Valued Functions. *Communications on Pure and Applied Mathematics*, XLIV:375–417, 1991.

sewell2002 [19] M J Sewell. Some applications of transformation theory in mechanics. *Large Scale Atmosphere–Ocean Dynamics*, 2:143–223, 2002.

cullen2006 [20] M.J.P. Cullen. *A Mathematical Theory of Large-Scale Atmosphere/Ocean Flow*. Imperial College Press, 2006.

Chynoweth1991 [21] 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.

Ceniceros2001 [22] 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.

Budd2011 [23] 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.

fftw [24] 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".