Comput. Methods Appl. Mech. Engrg. 200 (2011) 89–100

Contents lists available at ScienceDirect

Comput. Methods Appl. Mech. Engrg.

journal homepage: www.elsevier.com/locate/cma

Conservative interpolation between volume meshes by local Galerkin projection

P.E. Farrell [a,b,*], J.R. Maddison [c]

[a] Applied Modelling and Computation Group, Department of Earth Science and Engineering, Royal School of Mines, Imperial College London, London SW7 2AZ, UK
[b] Institute of Shock Physics, Royal School of Mines, Imperial College London, London SW7 2AZ, UK
[c] Atmospheric, Oceanic and Planetary Physics, Department of Physics, University of Oxford, Oxford OX1 3PU, UK

ARTICLE INFO

Article history:
Received 10 November 2009
Received in revised form 22 May 2010
Accepted 26 July 2010
Available online 21 August 2010

Keywords:
Interpolation
Conservation
Supermesh
Discontinuous Galerkin
Galerkin projection

ABSTRACT

The problem of interpolating between discrete fields arises frequently in computational physics. The obvious approach, consistent interpolation, has several drawbacks such as suboptimality, non-conservation, and unsuitability for use with discontinuous discretisations. An alternative, Galerkin projection, remedies these deficiencies; however, its implementation has proven very challenging. This paper presents an algorithm for the local implementation of Galerkin projection of discrete fields between meshes. This algorithm extends naturally to three dimensions and is very efficient.

1. Introduction

Interpolation between discrete fields arises frequently in computational physics. For example, the use of adaptive remeshing [35,33] frequently necessitates the interpolation of fields between meshes. Interpolation problems also arise in areas such as model coupling, model initialisation and visualisation.

This paper discusses the efficient local implementation of Galerkin projection of discrete fields between meshes. Firstly, we motivate the use of Galerkin projection by considering the more usual approach, consistent interpolation.

Consistent interpolation is the interpolation derived from the solution values of the donor mesh. Let $\mathcal{T}_D$ be the donor mesh with nodes $\mathcal{N}_D$, and let $\mathcal{T}_T$ be the target mesh with nodes $\mathcal{N}_T$. For each node $n_T \in \mathcal{N}_T$, a containing element K_D is identified in the donor mesh $\mathcal{T}_D$, and the solution q_D is evaluated at the physical location of the node n_T. Such an element K_D may be identified by an advancing front algorithm [28] or by an R-tree spatial indexing algorithm [21].

Consistent interpolation suffers from several serious drawbacks. Firstly, the interpolation is not conservative; the integral of the output field q_T will not in general be the same as the integral of the input field q_D. For some applications, conservation is a non-negotiable requirement. Secondly, the minima and maxima of the field will in general be eroded during consistent interpolation [10]. Thirdly, consistent interpolation is not suitable for discontinuous fields q_D which are not pointwise well-defined. Such fields arise in discontinuous Galerkin discretisations, which are becoming increasingly popular. While this may be circumvented by the application of a Clément-type pseudo-interpolation operator [6,4], the output would be restricted to the continuous subspace of the target function space.

Galerkin projection enjoys several key advantages over consistent interpolation. Firstly, it is the optimally accurate projection for the L_2 norm; the Galerkin projection is that function which minimises the L_2 norm of the interpolation error. From this, conservation follows naturally. Furthermore, it is well-defined for discontinuous fields. The properties of the Galerkin projection are well-known; however, its general implementation has proven challenging. This paper discusses the first efficient local implementation in two and three dimensions via the construction of an intermediate mesh, known as the supermesh.

George and Borouchaki [18] discuss the necessity of solution interpolation after adaptive remeshing, and propose the use of the Galerkin projection from mesh to mesh by means of mesh intersection. Although they comment that in their experience this provides a satisfactory algorithm for solution transfer, they give no examples. The reader is referred to a technical report by R. Ouachtaoui to be published in 1997 for further discussion; it appears, however, that this technical report was never published. Geuzaine et al. [19] also discuss the Galerkin projection between two-dimensional meshes; however, rather than integrating over the

* Corresponding author at: Applied Modelling and Computation Group, Department of Earth Science and Engineering, Royal School of Mines, Imperial College London, London SW7 2AZ, UK.
E-mail address: patrick.farrell06@imperial.ac.uk (P.E. Farrell).
URL: http://amcg.ese.ic.ac.uk.

0045-7825/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cma.2010.07.015

supermesh, the integrals appear to be computed over the target mesh. This is less accurate than assembling over the supermesh, and therefore should be referred to as an approximate Galerkin projection. A similar approach is taken by Parent et al. [34].

El Hraiech et al. [13] describe the desirability of the Galerkin projection for structural analysis, but comment that the construction of the supermesh was not yet feasible. Therefore, the integral is performed over a subdivision of the target mesh, with the hope being that computing the inner products of the basis functions on the refined target mesh is sufficiently accurate to assemble a useful Galerkin system. However, the basis functions of the donor mesh are (in general) discontinuous piecewise polynomials over any given element of the target mesh, which are very difficult to integrate by numerical quadrature schemes.

Farhat et al. [14] discuss a conservative load transfer algorithm for fluid–structure interactions. This was followed by the work of Heinstein and Laursen [23] and Jiao and Heath [25], who developed the natural extension to this algorithm, which is to assemble the mixed mass matrix over the supermesh. The supermesh is constructed by projecting the points of one surface onto their associate on the other and intersecting the resulting meshes. Both of these implementations only deal with two-dimensional surface meshes.

Farrell et al. [16] was the first to present the application of supermeshing to adaptive remeshing, and the first to describe a bounded variant of the Galerkin projection for piecewise linear fields. A technical report from INRIA has also been published on the application of supermeshing to two-dimensional simulations [1]. The projection algorithm described is not the optimally-accurate Galerkin projection and is specific to piecewise linear fields; however, it is conservative and bounded.

This paper presents a novel, local method for assembling the Galerkin projection system. Following from this, it presents the first implementation of Galerkin projection between three-dimensional meshes. The efficiency of the approach is demonstrated and shown to be sufficiently fast for practical applications.

2. Galerkin projection

Consider interpolation of a field q between a donor mesh $\mathcal{T}_D$ with $\mathcal{N}_D$ basis functions $\phi_D^{(i)}$, and a target mesh $\mathcal{T}_T$ with $\mathcal{N}_T$ basis functions $\phi_T^{(i)}$. Let q_T be the optimal interpolant in the L_2 norm [18,24]:

$$\|q_D - q_T\|_2 = \min_{q \in \mathcal{V}_T} \|q_D - q\|_2, \tag{1}$$

where $\mathcal{V}_T = \text{span}\left\{\phi_T^{(i)}\right\}$ for $i \in \{1, \ldots, \mathcal{N}_T\}$.

The L_2 norm of $q_D - q$ is minimised if the derivative of $\|q_D - q\|_2$ with respect to the coefficients of q is zero. Expanding the definition of the L_2 norm,

$$\frac{\partial}{\partial q^{(i)}} \int_\Omega (q_D - q)^2 \, dV = 0, \quad \forall i \in \{1, \ldots, \mathcal{N}_T\} \tag{2}$$

$$\Rightarrow \int_\Omega \frac{\partial}{\partial q^{(i)}} (q_D - q)^2 \, dV = 0, \quad \forall i \in \{1, \ldots, \mathcal{N}_T\}. \tag{3}$$

Expanding q in terms of the basis functions, $q = \sum_{i=1}^{\mathcal{N}_T} q^{(i)} \phi_T^{(i)}$

$$\Rightarrow \int_\Omega 2\phi_T^{(i)} (q_D - q) \, dV = 0, \quad \forall i \in \{1, \ldots, \mathcal{N}_T\}. \tag{4}$$

Since this value of q minimises $\|q_D - q\|_2$, $q = q_T$ and hence

$$\int_\Omega q_D \phi_T^{(k)} \, dV = \int_\Omega q_T \phi_T^{(k)} \, dV, \quad \forall k \in \{1, \ldots, \mathcal{N}_T\}. \tag{5}$$

This interpolation is referred to as Galerkin projection because it is optimal in the L_2 norm.

Conservation of the Galerkin projection follows naturally from this weak equality:

$$\int_\Omega q_D \, dV = \int_\Omega q_T \, dV, \tag{6}$$

if the constant function 1 is contained in the span of $\phi_T^{(i)}$ for $i \in \{1, \ldots, \mathcal{N}_T\}$.

Expanding q_D and q_T in terms of the basis functions, $q_D = \sum_{i=1}^{\mathcal{N}_D} q_D^{(i)} \phi_D^{(i)}$ and $q_T = \sum_{i=1}^{\mathcal{N}_T} q_T^{(i)} \phi_T^{(i)}$, yields

$$\int_\Omega \sum_{i=1}^{\mathcal{N}_D} q_D^{(i)} \phi_D^{(i)} \phi_T^{(k)} \, dV = \int_\Omega \sum_{j=1}^{\mathcal{N}_T} q_T^{(j)} \phi_T^{(j)} \phi_T^{(k)} \, dV. \tag{7}$$

This gives rise to the matrix equation

$$M_T q_T = M_{TD} q_D, \tag{8}$$

where

$$(M_T)_{ij} = \int_\Omega \phi_T^{(i)} \phi_T^{(j)} \, dV, \quad i,j \in \{1, \ldots, \mathcal{N}_T\} \tag{9}$$

and

$$(M_{TD})_{ij} = \int_\Omega \phi_T^{(i)} \phi_D^{(j)} \, dV, \quad i \in \{1, \ldots, \mathcal{N}_T\},\ j \in \{1, \ldots, \mathcal{N}_D\}. \tag{10}$$

M_{TD} represents a mixed mass matrix between meshes $\mathcal{T}_D$ and $\mathcal{T}_T$. Assuming the mixed mass matrix may be assembled, Eq. (8) may then be solved using a standard iterative solver to compute q_T.

Dirichlet boundary conditions can be imposed upon this projection in the strong sense via removal of the basis functions associated with boundary condition nodes. However, this does not in general result in a conservative projection.

3. Supermeshes

The desirable properties of the above projection have been known for some time. However, its implementation has proven challenging. To assemble the right-hand-side of Eq. (8), it is necessary to integrate the products of the basis functions of $\mathcal{T}_D$ and $\mathcal{T}_T$. Over each element of $\mathcal{T}_T$, the basis functions of $\mathcal{T}_D$ are possibly discontinuous piecewise polynomials. Therefore, if the products of the basis functions are evaluated at each of the quadrature points of $\mathcal{T}_T$, the integrals will not in general be exact, as quadrature schemes are exact for a specified polynomial order, not piecewise polynomials. This error in the assembly of M_{TD} causes the loss of conservation and accuracy properties, which is highly undesirable. An adaptive quadrature scheme for the assembly of M_{TD} is considered later in Section 3.2 and found to be impractical.

Over each element of the target mesh, the donor basis functions are possibly discontinuous piecewise polynomials. However, over each element of the donor mesh, the donor basis functions are simply polynomials. Therefore, over the intersection of a target element and donor element, both sets of basis functions are polynomials and therefore so is their product. This is the key observation that makes possible the assembly of M_{TD}. Therefore, we define a supermesh as a mesh of the intersections of the target and donor elements, illustrated for simple quadrilateral meshes in Fig. 1 [16].

3.1. Local supermeshing

Previously, in [16], a global approach to supermesh construction was proposed; this exploited a relationship between supermesh construction and the constrained Delaunay triangulation to form the supermesh in its entirety. In this paper, an alternative approach called local supermeshing is proposed. Here, the supermesh is

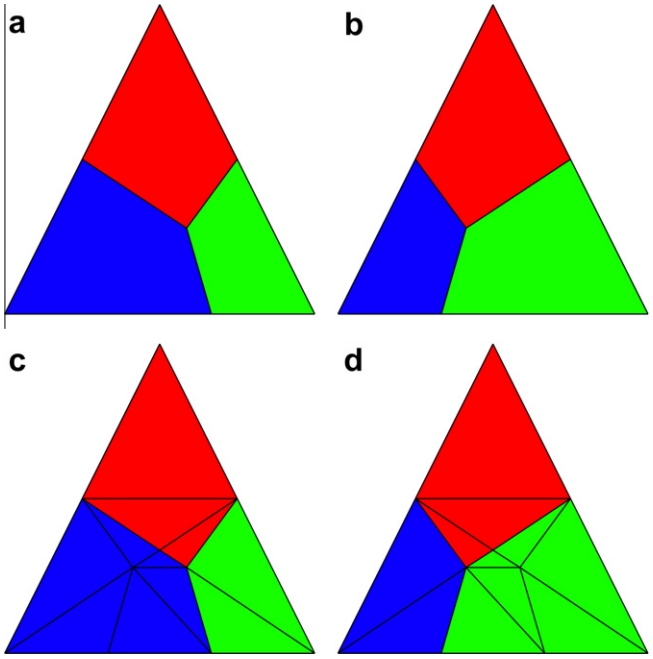

Fig. 1. (a) and (b) Two quadrilateral meshes. (c) A triangular supermesh of (a) and (b), coloured to show the elements of (a). (d) The same supermesh of (a) and (b), coloured to show the elements of (b). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

formed by meshing each intersection in turn. This approach is significantly more efficient and scalable in memory usage. The algorithm also naturally extends to three dimensions, unlike the approach proposed in [16]. The newly proposed method is particularly suited to interpolation between spaces of discontinuous functions, as both the assembly and solve can be performed entirely locally on a given element of $\mathcal{T}_T$ by exploiting the block-diagonal structure of the mass matrix.

We summarise the algorithm as follows:

1. Identify the intersecting pairs of elements.
2. For each element $K \in \mathcal{T}_T$:
   (a) Assemble the contribution to the mass matrix M_T.
   (b) For each intersecting element $K_D \in \mathcal{T}_D$:
      i. Form $\mathcal{T}_S^K$, the mesh of the region of intersection.
      ii. Assemble the contribution to the mass matrix M_{TD} by integrating the basis functions of K and K_D over $\mathcal{T}_S^K$.
      iii. Apply this to q_D to form the contribution to the right-hand-side of Eq. (8).
   (c) If $\mathcal{T}_T$ is discontinuous, perform the local solve of Eq. (8).
3. If $\mathcal{T}_T$ is continuous, perform the global solve of Eq. (8).

This procedure therefore uses local intersection-by-intersection integration to assemble the right-hand-side of Eq. (8).

The minimally diffusive bounded conservative interpolation scheme of Farrell et al. [16] requires only this right-hand-side, together with the lumped and consistent target mesh mass matrices, and hence this bounding method can be applied using a local supermeshing approach. For discontinuous $\mathcal{T}_T$ the block-diagonal structure of the mass matrix can again be exploited, and the bounding procedure is an element-wise local operation that can be performed as a further step immediately following the local solve, without requiring assembly of the full global system. Further details of the conservative bounding algorithm can be found in [16].

3.1.1. Intersection identification

To form a supermesh intersection-by-intersection, it is first necessary to identify the intersecting pairs of elements of $\mathcal{T}_D$ and $\mathcal{T}_T$. The existence of a suitable geometric intersection predicate is assumed (for more details, see [31]).

The naïve brute-force approach performs $O(|\mathcal{T}_D||\mathcal{T}_T|)$ intersection tests, which is clearly undesirable. The element pairs may be first filtered by an R-tree algorithm, which only considers pairs with intersecting bounding boxes for intersection testing [21,29]. The construction of the R-tree takes $O(|\mathcal{T}_D| \log |\mathcal{T}_D|)$ time, and the query for each element of $\mathcal{T}_T$ takes on average $O(\log |\mathcal{T}_D|)$ time, for a total time of $O((|\mathcal{T}_D| + |\mathcal{T}_T|) \log |\mathcal{T}_D|)$. This figure ignores the actual bounding box intersection tests performed ascending and descending the tree, which will render it sensitive to the number of intersections between the meshes. Neither of these algorithms exploits the mesh-based structure of $\mathcal{T}_D$ and $\mathcal{T}_T$. By exploiting the fact that the elements are not merely sets of polytopes but that they form meshes of known connectivity, it is possible to develop a novel algorithm based upon advancing fronts for intersection identification in connected domains. This algorithm is summarised as follows:

1. Choose an element a_1 in $\mathcal{T}_D$, and find an element b_1 in $\mathcal{T}_T$ intersecting a_1 (the seed) by a brute force method
2. While there are unprocessed elements in $\mathcal{T}_D$:
   (a) Find the elements I_i in $\mathcal{T}_T$ intersecting a_i by:
      i. $I_i \leftarrow \{b_i\}$
      ii. $F \leftarrow N(b)$, where $N(b_i)$ are the neighbouring elements of b_i in $\mathcal{T}_T$
      iii. While $|F| \neq 0$:
         A. Remove a neighbour n from F
         B. If $r \cap n \neq \emptyset$: $I_i \leftarrow I \cup \{n\}$, $F \leftarrow F \cup N(n)$
   (b) Choose a new unprocessed element a_{i+1} in $\mathcal{T}_D$ neighbouring a processed element a_k.
   (c) Find a new seed b_{i+1} by searching through I_k.

This algorithm is illustrated in Fig. 3. The algorithm performs $O(|\mathcal{T}_D| + k)$ intersection tests, where k is the number of intersecting elements between $\mathcal{T}_D$ and $\mathcal{T}_T$. If an initial seed is supplied to the algorithm, this reduces to $O(k)$. A complete formal proof of the linear scaling of this algorithm is given in [15].

For practical reasons, a bounding box intersection predicate was used in the implementation of this algorithm. This reports false

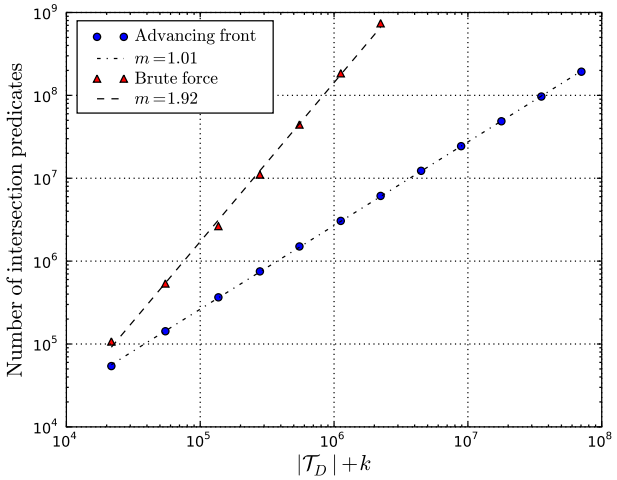

Fig. 2. Scaling properties of the advancing front intersection finding algorithm tested in a 3D cubic shell domain [15]. The number of intersection predicates performed is linear in $|\mathcal{T}_D| + k$.

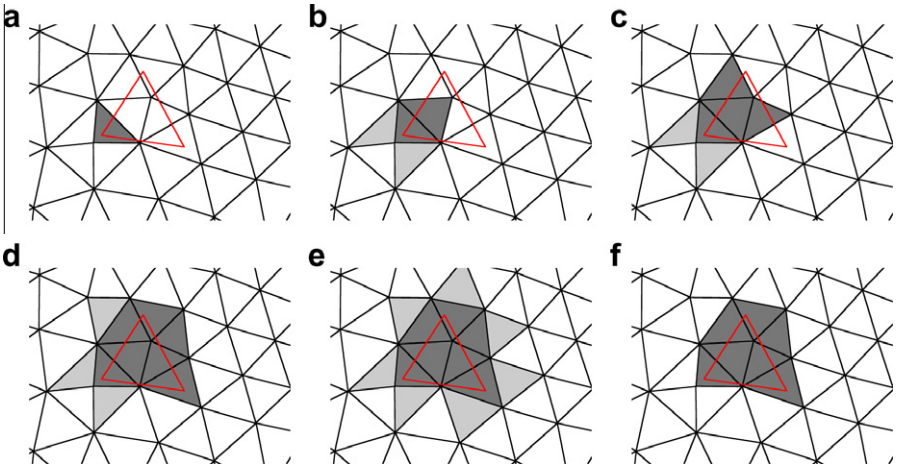

Fig. 3. Illustration of element intersection reporting via an advancing front method. The target mesh $\mathcal{T}_T$ is shown in black, and a donor element a in $\mathcal{T}_D$ is shown in red. (a) A seed element b in $\mathcal{T}_T$ intersecting a is supplied (shaded). (b) The neighbours of the seed are tested for intersection with a (intersecting elements shown in darker shading). (c)–(e) The untested neighbours of intersecting elements in $\mathcal{T}_T$ are successively tested, until no new intersections are found. (f) The resulting set of intersecting elements I (shaded). Seeds for the element intersection search for the neighbours of a (not shown) can be determined through a search over I. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

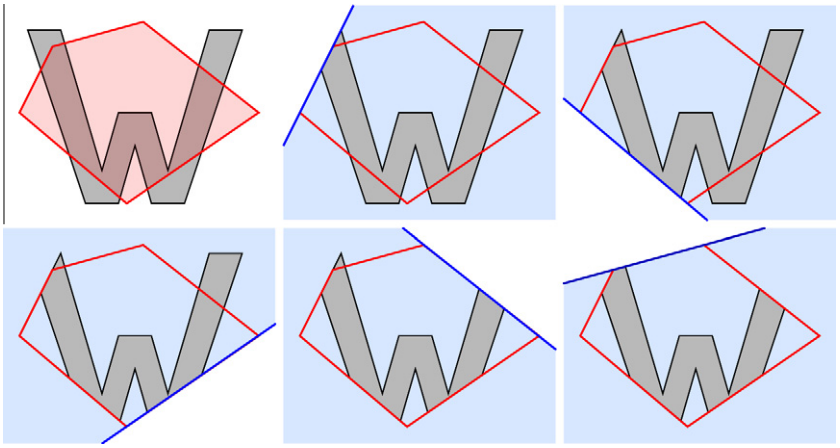

Fig. 4. The Sutherland–Hodgman clipping algorithm [41]. The subject polygon is initialised to the 'W' and the clipping polygon is the pentagon. Each edge of the clipping polygon discards the subject vertices outside it, possibly creating new vertices for any intersecting edges. Figure credit: Wikipedia (public domain).

positive intersections which are discarded at the intersection stage, described in the following section.

In order to demonstrate that the algorithm applies to higher dimensions and multiply connected domains, the algorithm was applied to a domain consisting of a cube of unit size, with a cubic region of half-unit size removed from its centre to form a cubic "shell". Unstructured meshes of varying node counts were generated by the addition of randomly distributed points throughout the shell, followed by constrained Delaunay tetrahedralisation using the three-dimensional mesh generator Tetgen [40].

Pairs of meshes were generated for input shell nodes counts of 2^n, $n \in \{8,\ldots,15\}$. For each pair, the advancing front intersection reporting algorithm was applied using a bounding box intersection predicate. The resulting set of intersections was verified for completeness by comparison against alternative intersection reporting algorithms. For shell node counts in the range 256–4096, the set of intersections was verified against a brute force reporting algorithm, and for shell node counts of 8192 and above was verified against an R-tree spatial indexing algorithm [21]. The number of bounding box intersections k was computed by summing the number of intersections for each element in $\mathcal{R}$. The scaling properties derived from this test are shown in Fig. 2. The scaling is observed to be linear in $\mathcal{T}_D + k$ as expected.

3.1.2. Intersection construction

Once the intersecting elements in $\mathcal{T}_D$ are identified for a given $K_T \in \mathcal{T}_T$, these intersections must be meshed so that the quadrature of the products of the basis functions may be performed.

Various intersection construction algorithms are available, depending on the specifics of the elements used. For general convex polytopes, algorithms are available in two dimensions [39] and three dimensions [5]. One of the simplest is the Sutherland–Hodgman clipping algorithm [41].

The Sutherland–Hodgman algorithm is illustrated in Fig. 4. The subject polygon is initialised to be one of the input polygons, and the other is designated the clipping polygon. Each edge of the clipping polygon is considered in turn. The edge, when extended, forms a line. An orientation test is performed for each vertex of the subject polygon to determine whether it is on the positive or negative side of the line, or collinear. If its orientation is positive or collinear, it is retained, while if it is negative, it is discarded. Furthermore, if one vertex of an edge in the subject polygon is

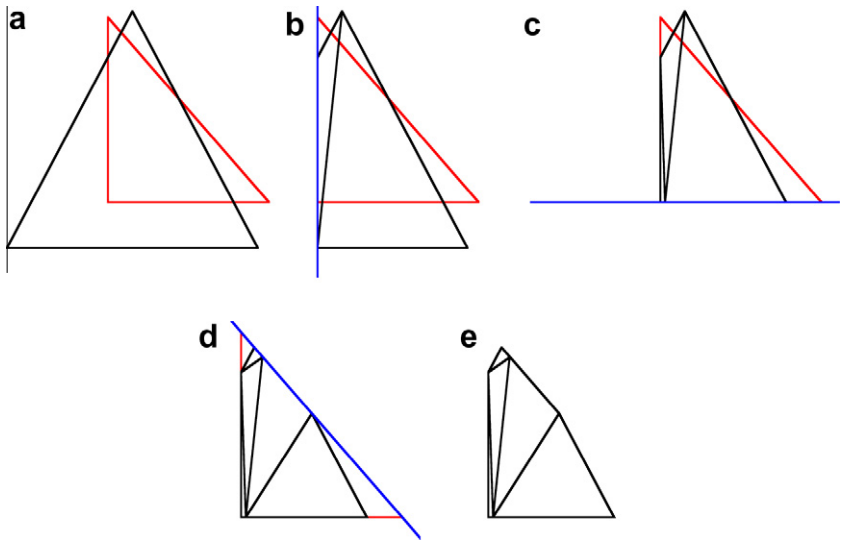

Fig. 5. An example of intersection construction via repeated clipping of simplex meshes. (a) Two triangles to be intersected. (b)–(d) The black triangle is successively clipped using each edge of the red triangle. In each clip the triangles from the previous stage are individually subdivided to form a new triangle mesh. Hence the intersection procedure is composed of a number intersections of triangles with half-spaces. (e) The resulting mesh of the intersection. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

retained while the other vertex is discarded, then that edge must intersect the clipping line, so the point of intersection between the subject edge and the clipping line is computed and inserted into the output vertices. The list of output vertices then forms the subject polygon to be compared to the next edge of the clipping polygon. In this manner, vertices of the subject polygon which are external to the clipping polygon are systematically removed.

A similar approach is taken in three dimensions. Each face of the clipping polyhedron is considered in turn. For each vertex of the subject polyhedron, an orientation test is performed to decide whether the vertex should be discarded or retained. Again, if an edge has one vertex discarded and one vertex retained, then a new vertex is formed at the intersection of the edge and the clipping plane. The new vertices produced are then connected with edges to form a new face of the subject polyhedron. This procedure is continued until all the faces of the clipping polyhedron have discarded those parts of the subject polyhedron outside the intersection.

A disadvantage of the Sutherland–Hodgman approach is that the algorithm only returns the vertices of the polytope of the intersection. In order to assemble the integrals of the mixed mass matrix, this polytope must be meshed. If the two input elements are convex, the intersection polytope is also convex, and thus the Delaunay triangulation of the vertices of the polytope is a mesh of the intersection.

An alternative approach, adopted in this work, naturally constructs the mesh of the polytope during the intersection procedure. This algorithm follows that of Eberly [11]. The list of output polytopes is initialised to be (a simplicial decomposition of) one of the input polytopes, and the other is designated the clipping polytope. Again, each edge of the clipping polytope is considered in turn. Each polytope in the output list is clipped against the edge; the clipping procedure returns a list of simplices that are to replace the considered polytope. In this manner, at every step of the intersection procedure, the intersection is represented as a list of simplices. Therefore, the output of the procedure consists of a list of simplices which constitute a mesh of the intersection. By design, the clipping procedure need only consider the intersection of a simplex with a half-space; this can be divided into a handful of possible cases and the solution for each devised on paper. This approach eliminates the need for costly post-processing of the resulting intersection polytope. An illustration of this intersection procedure is shown in Fig. 5.

3.2. Adaptive quadrature approach

Clearly, supermeshing is not the only way to compute the inner products of the basis functions of the target and donor meshes. An alternative is to evaluate the basis functions of the donor mesh at the quadrature points of the target mesh and compute the integrals using numerical quadrature. In this vein, an experiment was performed to test the practicality of computing the integrals in this manner. A very similar approach was advocated in [13]; there, the authors subdivide the elements of the target mesh into several sub-elements to compute a more accurate approximation of the integrals.

The adaptive quadrature package `CUBPACK` [7] was chosen as the numerical quadrature scheme, as this appeared to offer implementations of the most recent research in numerical quadrature. The example used was a single projection between two two-dimensional $P2_{DG}$ meshes of 200 triangular elements. The adaptive quadrature algorithm was used to compute the inner products of the basis functions of the two meshes. The target relative error was set to 1% of the integral and the maximum number of integrand evaluations for each element in the target mesh was set to 10^5.

It was found that the integral computation performed for each element reached the maximum number of integrand evaluations, 10^5; the target relative error of 1% was never reached. As such, the system of equations was inaccurately assembled, resulting in conservation errors on the order of 10^{-4}, or 0.02%. By contrast, the integral error arising from local supermeshing was on the order of 10^{-16}, equivalent to double precision roundoff errors. The interpolant constructed with supermeshing took less than a second, while the adaptive quadrature approach took over 15 min. The adaptive quadrature approach was therefore over 10^3 times slower than projection by supermeshing, for a poorer result. The experiment was not attempted on other examples due to the prohibitive computational cost.

It is to be emphasised that the outcome of this experiment does not reflect on the quality of the algorithms developed in `CUBPACK`. Over each element, the integral to be computed is a discontinuous

piecewise polynomial; it is merely the case that supermeshing is a vastly more efficient method for computing these particularly difficult integrals to within an acceptable error, even for this simple case. However, it would appear that the method of El Hraiech et al. [13] is not practical for such integrals.

Seen in this light, local supermeshing may be interpreted as a very specific quadrature scheme for the inner products of basis functions associated with meshes.

3.3. The size of the supermesh

The size of the supermesh may be estimated as follows. The intersection of each intersecting pair of elements from the target and donor meshes must be representable as the union of supermesh elements; each intersection must in turn be triangulated to form a mesh over which quadrature may be performed. The number of intersections k is bounded by the cases where every element in $\mathcal{T}_D$ intersects with exactly one element in $\mathcal{T}_T$, and where every element in $\mathcal{T}_D$ intersects with every element in $\mathcal{T}_T$:

$$\max(|\mathcal{T}_D|, |\mathcal{T}_T|) \leqslant k \leqslant |\mathcal{T}_D||\mathcal{T}_T|. \tag{11}$$

In two dimensions, if the elements of the input meshes are convex polygons of n vertices, then the intersection is a convex polygon of at most $2n$ vertices [31]. A convex polygon of $2n$ vertices may be minimally triangulated into $2n-2$ triangles. In three dimensions, the problem is significantly harder. Given two tetrahedra, the intersection has at most 8 faces [37, Theorem 7.2]. A 3-polytope with 8 faces can have at most 12 vertices [38, p. 499]. Computing the size of the minimal triangulation of a convex polyhedron is NP-complete [2]. However, the size of the minimal triangulation of a polyhedron with n vertices is bounded above by $\binom{n}{2} - 2n + 3$[12], where $\binom{n}{2} = \frac{n!}{2!(n-2)!}$. Therefore, the minimal number of elements of the supermesh in the worst case is bounded above by $C_d|\mathcal{T}_D||\mathcal{T}_T|$, with $C_2 = 4$ and $C_3 = 45$. For most practical pairs of meshes, this bound is very pessimistic.

3.3.1. Error computation

An elegant feature of supermesh-based interpolation is that the error between the donor function and its interpolant is exactly computable (ignoring roundoff).

The supermesh provides a function superspace of the function spaces associated with the input meshes, because the basis functions of the input meshes are exactly representable upon it [15]. This implies that the projection operator obtained by consistent interpolation,

$$\Pi_{SP} : \mathcal{V}_P \to \mathcal{V}_S, \quad P \in \{T, D\} \tag{12}$$

is the identity operation, i.e. $\Pi_{SP}(q) = q$. By closure, the difference between two functions in $\mathcal{V}_S$ is also an element of $\mathcal{V}_S$, and therefore the projection error may be computed as

$$\eta = \Pi_{SD}(q_D) - \Pi_{ST}(q_T) = q_D - q_T. \tag{13}$$

The evaluation of Π_{SP} is trivial: since the parenthood mapping from each element in $\mathcal{T}_S^K$ is already stored to facilitate the construction of M_{TD}, no searching need be performed; only the evaluation of the parent basis functions is required.

The computation of η is exact, ignoring roundoff error in the evaluation of the projection operator Π_{SP}. Needless to say, this is a very attractive property.

As η is available as a function, any desired norm may be taken to quantify the error. Since the Galerkin projection is optimal in the L_2 norm, the L_2 norm is a sensible choice. If the projection were modified so that it were optimal in the H_1 norm, as in [24], then the H_1 norm should be chosen.

In practice, this error computation is more useful for discontinuous fields, as q_T is computable element-by-element and therefore so is η. The error computation can still be applied for continuous fields, but either the supermesh must be stored or recomputed, as q_T requires a global mass matrix solve and so the entire supermesh must be assembled before it is computable.

3.3.2. Numerical order of convergence

A numerical experiment was performed to investigate the observed order of convergence of the Galerkin projection. A given scalar field ζ is evaluated on the donor and target meshes. Although the donor and target meshes are topologically unrelated, they share the same characteristic mesh size h. The Galerkin projection from the donor mesh to the target mesh is computed. The error is then computed as described in Section 3.3.1. This process is repeated with pairs of meshes of different sizes. The meshes were generated with Gmsh [20].

Three functions were used:

$$\zeta_1(x, y) = 5y^3 + x^2 + 2y + 3, \tag{14}$$

$$\zeta_2(x, y) = \exp x^2 + 2y, \tag{15}$$

$$\zeta_3(x, y) = \sin x + \cos y. \tag{16}$$

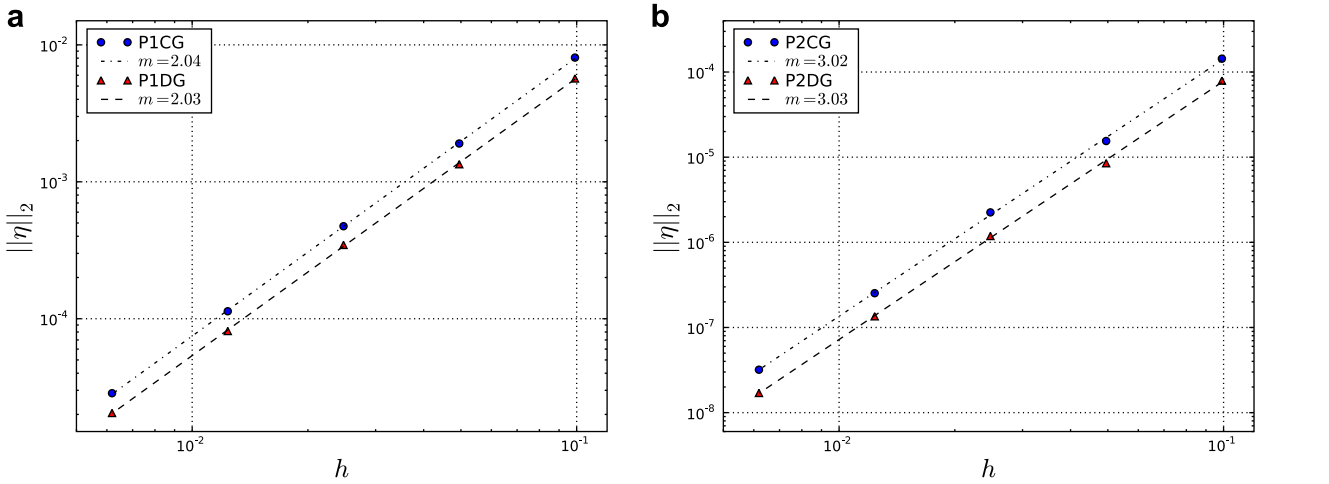

Fig. 6. Convergence results for the L_2 error of ζ_1 as a function of mesh sizing h for (a) linear basis functions and (b) quadratic basis functions. The error is $O(h^{p+1})$, as expected. Since ζ_1 is cubic, the error for higher-order basis functions is on the order of numerical zero.

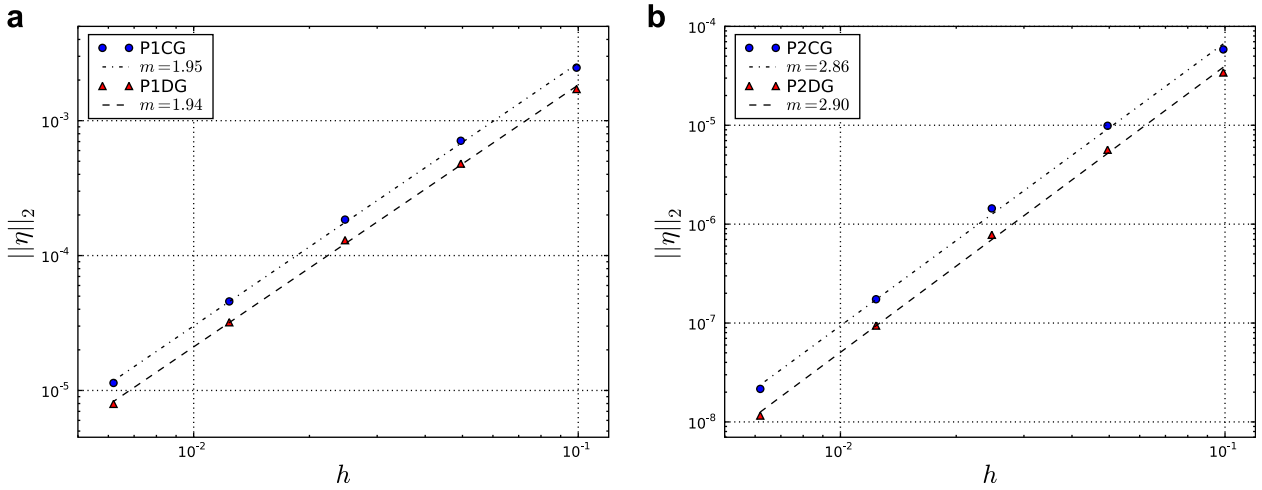

Fig. 7. Convergence results for the L_2 error of ζ_2 as a function of mesh sizing h for (a) linear basis functions and (b) quadratic basis functions. The error is $O(h^{p+1})$, as expected.

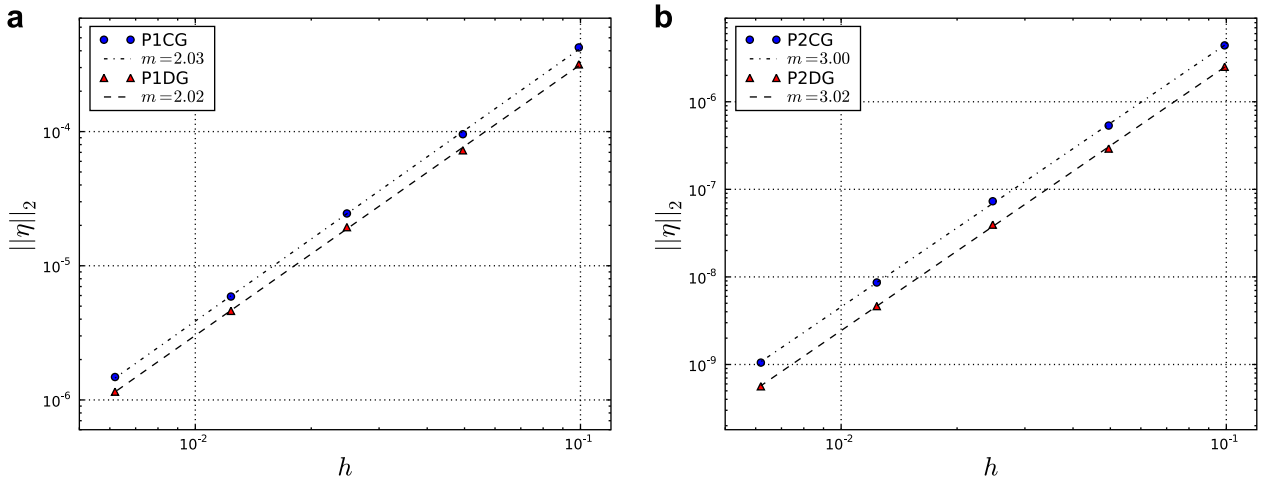

Fig. 8. Convergence results for the L_2 error of ζ_3 as a function of mesh sizing h for (a) linear basis functions and (b) quadratic basis functions. The error is $O(h^{p+1})$, as expected.

Convergence results for functions ζ_1–ζ_3 with finite element basis function degree $p = 1,2$ are shown in Figs. 6–8. The $O(h^{p+1})$ expected order of convergence is observed in the numerical results.

4. Examples

4.1. Profiling results

An experiment was conducted to investigate the scaling of the proposed local supermeshing scheme with problem size. For a given size of problem, two different unstructured meshes were generated used the Gmsh mesh generator [20], and a $P1_{DG}$ field interpolated between them via Galerkin projection 10 times. The mesh edge lengths in the target mesh were, in each case, 10% smaller than those in the donor mesh. An example of these meshes is shown in Fig. 9. The time taken for this problem size was computed as the total projection CPU time (measured by the `mpi_wtime` MPI routine) divided by the number of projections. Therefore, the reported timings include the cost of the intersection finder, constructing the supermesh, assembling the Galerkin system and solving it.

The experiment was conducted in serial on a two-processor quad-core Intel X5355 2.66 GHz machine with 16 Gb of RAM. The implementation of the algorithm was compiled with version 10.1 of the Intel compiler suite and used the Hoard memory allocator [3].

The results are shown in Fig. 10. As can be seen, the time taken by the algorithm is linear in the size of the input meshes. In two dimensions, the algorithm takes approximately 0.13 ms of CPU time per element in the mesh, or 15 μs per bounding box element intersection reported by the intersection finder; in three dimensions, 0.66 ms of CPU time per element, or 10 μs per reported element intersection. Despite the inherent complexity of supermeshing in three dimensions, the procedure is faster per reported intersection in three dimensions than in two; this is because significant development effort was invested in optimising the three-dimensional intersector using OProfile [27]. For the largest test with tetrahedral elements, with 819,758 elements in the donor mesh and 1,161,175 elements in the target mesh (136,917 and 192,515 degrees of freedom, respectively), the Galerkin projection took 665 s, which is sufficiently fast for practical use. Note that in the context of adaptive remeshing, the cost could be reduced dramatically by supplying the projection algorithm with information about unchanged elements from the adaptive remeshing algorithm. If only 10% of the elements of the mesh change, then it is unnecessary to supermesh or assemble the mixed mass matrix over the other 90%, and thus the cost of Galerkin projection would be reduced by a factor of approximately 10.

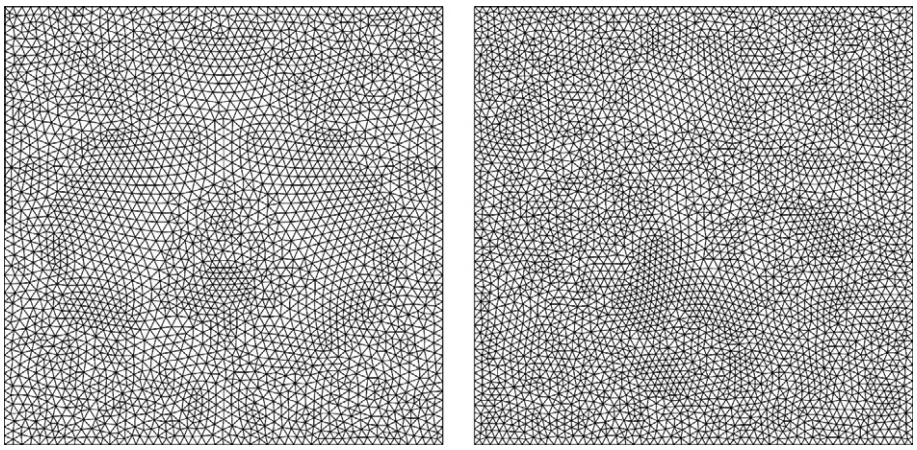

Fig. 9. The lowest resolution two-dimensional meshes used in the profiling analysis of Galerkin projection. Left: Donor mesh. Right: Target mesh.

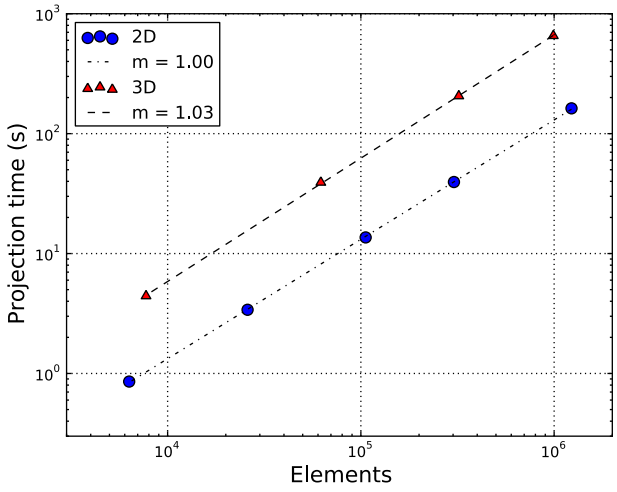

Fig. 10. Profiling results for the experiment described in Section 4.1 in two (blue circles) and three (red triangles) dimensions. The number of elements plotted is the mean of the number of elements in the donor and target meshes. The time taken by the algorithm is linear in the size of the inputs. In two dimensions, the algorithm takes approximately 0.13 ms of CPU time per element in the mesh; in three dimensions, 0.66 ms of CPU time per element. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

4.2. Two-dimensional lock exchange

With the development of Galerkin projection, combinations of techniques that were previously infeasible become not just possible but useful.

To demonstrate this, a two-dimensional lock exchange simulation was conducted in the domain $\Omega = [0,0.8] \times [0,0.1]$. The lock exchange problem consists of two fluids of different density that are initially separated by a gate or 'lock'. The gate is removed at $t = 0$ and the buoyancy force drives two gravity currents that propagate in opposite directions, with the denser fluid flowing under the lighter fluid.

The equations solved were the incompressible Navier–Stokes equations subject to the Boussinesq approximation for velocity and pressure, and the advection–diffusion equation for temperature. The equation set was closed with the addition of a linear equation of state. A no-slip boundary condition was enforced at the bottom boundary and no-normal flow was enforced on all other boundaries. The kinematic viscosity coefficient ν was set to 10^{-6}, to give a Reynolds number of 790 by the definition of Özgökmen et al. [32]. The initial condition for temperature (Fig. 11) was set to

$$T(x,y,t=0) = \begin{cases} -1/2, & \text{if } x < 0.4, \\ 1/2, & \text{otherwise.} \end{cases} \tag{17}$$

The velocity and pressure fields were discretised with the $P1_{DG}$–P2 finite element discretisation of Cotter et al. [9]. This element pair promises to be excellent for geophysical applications, as it satisfies the Ladyzhenskaya–Babuška–Brezzi stability condition and excellently represents geostrophic balance [8]. The advection–diffusion equation is discretised using a control volume scheme with the Sweby limiter [42,44]. Crank–Nicolson time-stepping was used with a timestep Δt of 0.025. The simulation was terminated after 24 units of simulation time. The mesh was adapted every 5 timesteps with the algorithm of Vasilevskii and Lipnikov [43]. A metric used to control the L_∞ norm of the interpolation error associated with temperature was computed [33,36]:

$$M = \frac{|H|}{\epsilon}, \tag{18}$$

where H is the Hessian of the temperature field and ϵ is a configurable adaptivity tolerance. The adaptivity tolerance was set to 0.025. A minimum edge length of 5×10^{-4} and a maximum edge length of 0.5 were enforced on the metric.

This simulation requires Galerkin projection for two reasons. Firstly, consistent interpolation is undefined for the discontinuous velocity field, so if one wishes to combine adaptive remeshing with a discontinuous discretisation then an alternative to consistent interpolation is necessary. Here, Galerkin projection is used for velocity. Secondly, the discretisation of the advection–diffusion equation preserves the integral of temperature to machine

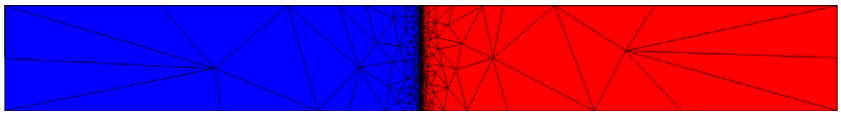

Fig. 11. Initial condition for temperature for the two-dimensional lock exchange problem. The bounds on temperature are $[-0.5,0.5]$.

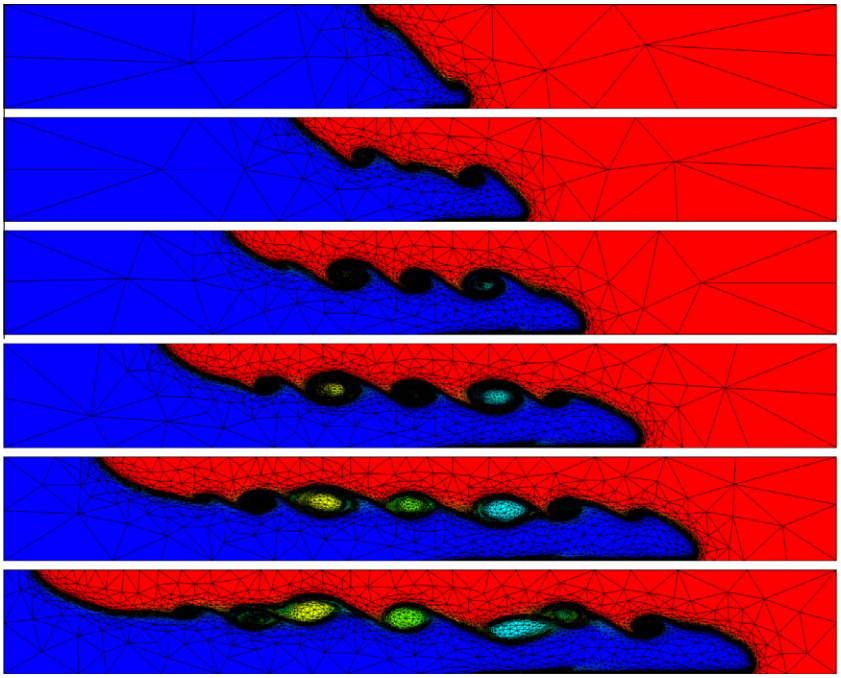

Fig. 12. Simulation results for the lock exchange at times 4, 8, 12, 16, 20, 24 time units. The bounds for temperature are $[-0.5, 0.5]$. Note the mesh adapting to resolve the temperature interface. The combination of adaptive remeshing and discontinuous Galerkin methods would be impossible with consistent interpolation.

precision and the bounds approximately; therefore, it is highly desirable to preserve these properties through the interpolation step. Therefore, the bounded Galerkin projection of Farrell et al. [16] is employed for temperature.

Simulation results are displayed in Fig. 12. The simulation spends 0.2% of runtime identifying intersecting elements using the algorithm of Section 3.1.1 and 6.9% of runtime in the assembly and solve of the Galerkin projection. Of the time spent performing the Galerkin projection, 59% of runtime was spent intersecting triangles, 0.2% performing the local solves for the projection of the discontinuous velocity, 3% performing the global solves for the projection of the continuous pressure and temperature, and 12% applying the diffusive bounding algorithm to the temperature field. Other projection costs are attributed to the time spent interpolating basis functions onto the intersections, and the time spent performing local assembly. The adaptive remeshing algorithm ensures that the mesh resolution is appropriately placed to resolve the temperature interface. The velocity of the front is in good agreement with the benchmark data of Härtel et al. [22]. As can be seen in Fig. 13, both the discretisation and the interpolation step are conservative. Galerkin projection supplies a key component in rendering possible the combination of discontinuous discretisations and adaptive remeshing.

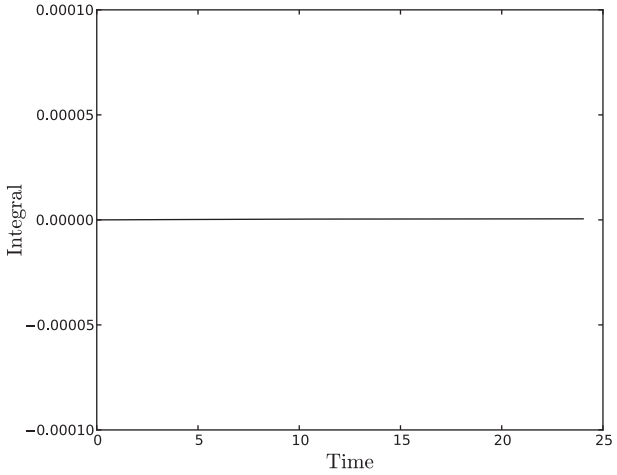

Fig. 13. Integral of the temperature field for the two-dimensional lock exchange simulation.

4.3. Three-dimensional water collapse

Simulations of multimaterial flows are numerically challenging. The interfaces of the material volume fractions recording the materials are sharp and anisotropic, and this must be reflected in the mesh upon which these flows are discretised. Furthermore, the discretisation must be conservative and bounded; otherwise nonphysical phenomena such as mass exchange may occur.

Adaptive remeshing was applied to an unsteady multimaterial simulation of a water column collapsing in air under gravity. The equations solved were the incompressible Navier–Stokes equation and the advection equation for the evolution of the material volume fraction.

The simulation was conducted in the domain $\Omega = [-0.5, 0.5] \times [-0.5, 2] \times [-0.5, 0.5]$. A material volume fraction representing water is initialised to be 1 in the region $[-0.5, -0.25] \times [-0.5, 0] \times [-0.5, 0]$ and zero elsewhere. No-normal flow was imposed on velocity on all boundaries except for the top. At the top, a homogeneous Neumann boundary condition was imposed on velocity, and a homogeneous Dirichlet boundary condition was imposed on pressure. The $P0_{DG}$–$P1_{CV}$ element pair was used for the velocity–pressure discretisation; the HyperC control volume face value algorithm was used for the advection equation [26]. The motivation for this element pair and the discretisation is described in Wilson [44]. Crank–Nicolson time-stepping was used, with an initial timestep of $\Delta t = 2 \times 10^{-4}$. The timestep was adapted through the simulation to maintain a CFL number of 2.5. The simulation was terminated at $t = 0.9$. The material volume

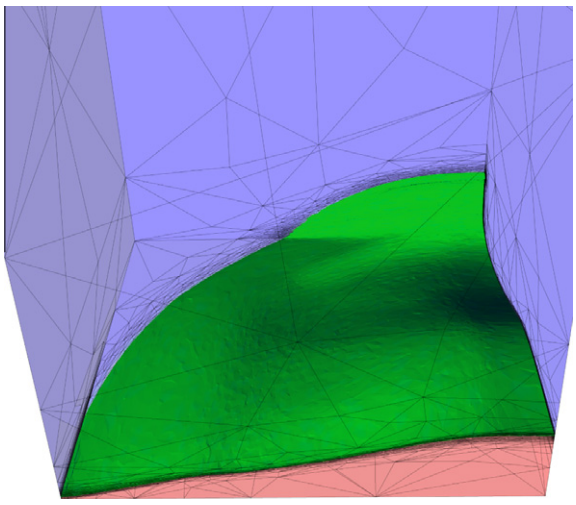

Fig. 14. Isosurface of the material volume fraction field at time $t = 0.43$ for the water collapse simulation.

fraction representing water was chosen as the field to guide adaptivity. A three-dimensional extension of the metric formation algorithm of Formaggia and Perotto [17]; Micheletti and Perotto [30] was used to control the H_1-seminorm of the interpolation error; the target error was chosen to be $\tau = 25$. A minimum edge length of 0.001 was enforced to constrain the adaptive algorithm. The mesh was adapted every 10 timesteps. To spread resolution ahead of the dynamics, the metric tensor formed was advected forward for one adaptivity period and superimposed with itself. This has the effect of extending resolution to where the interface will be over the course of the adaptivity period. Since the velocity field was discontinuous, Galerkin projection was used to interpolate it from each previous mesh to the corresponding adapted mesh. As conservation and boundedness of the material volume fraction are crucial, the bounded Galerkin projection of Farrell et al. [16] was employed for this field.

Simulation results are displayed in Figs. 14 and 15. The simulation spent 0.4% of runtime identifying intersecting elements using the algorithm of Farrell [15], and spent 12.3% of runtime in the Galerkin projections. Again, the adaptive remeshing ensures that the mesh resolution is focussed on the material interface.

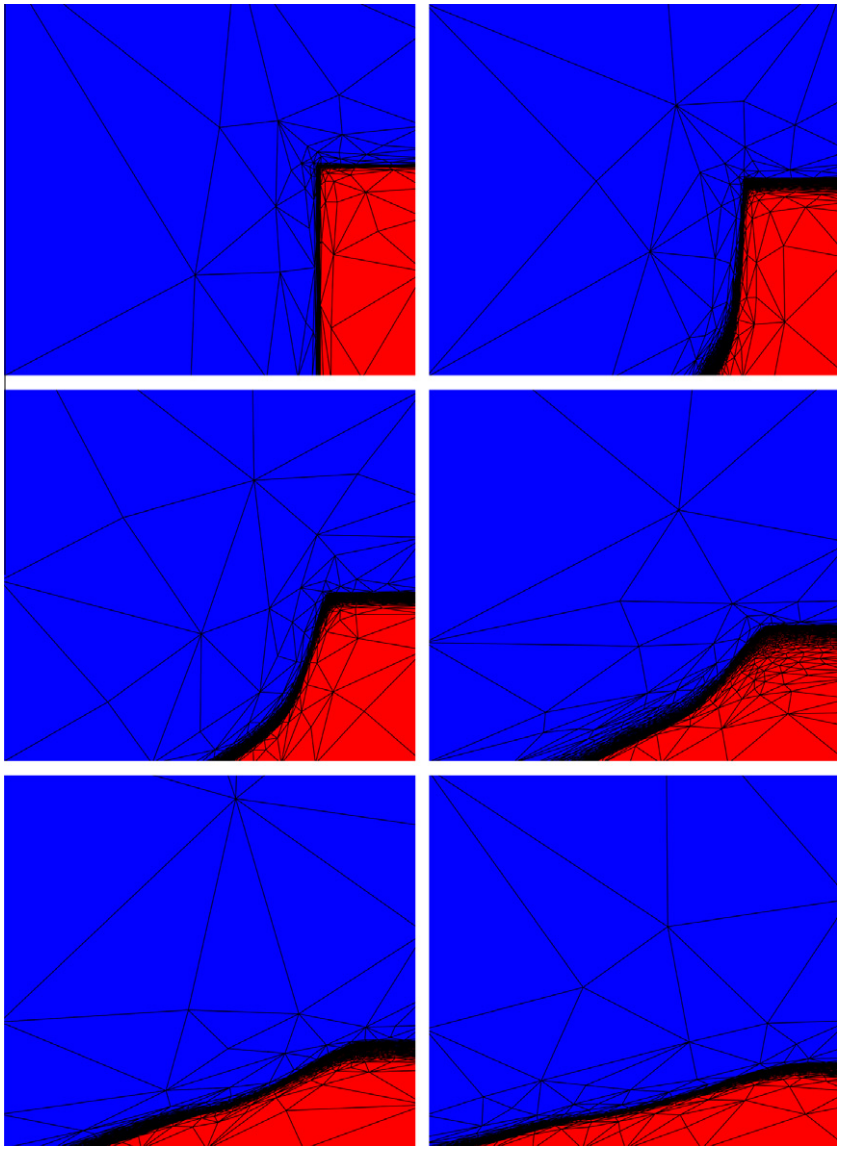

Fig. 15. The material volume fraction and mesh at times $t = 0, 0.09, 0.17, 0.25, 0.33, 0.39$ for the water collapse simulation, viewing the domain surface at $y = -0.5$.

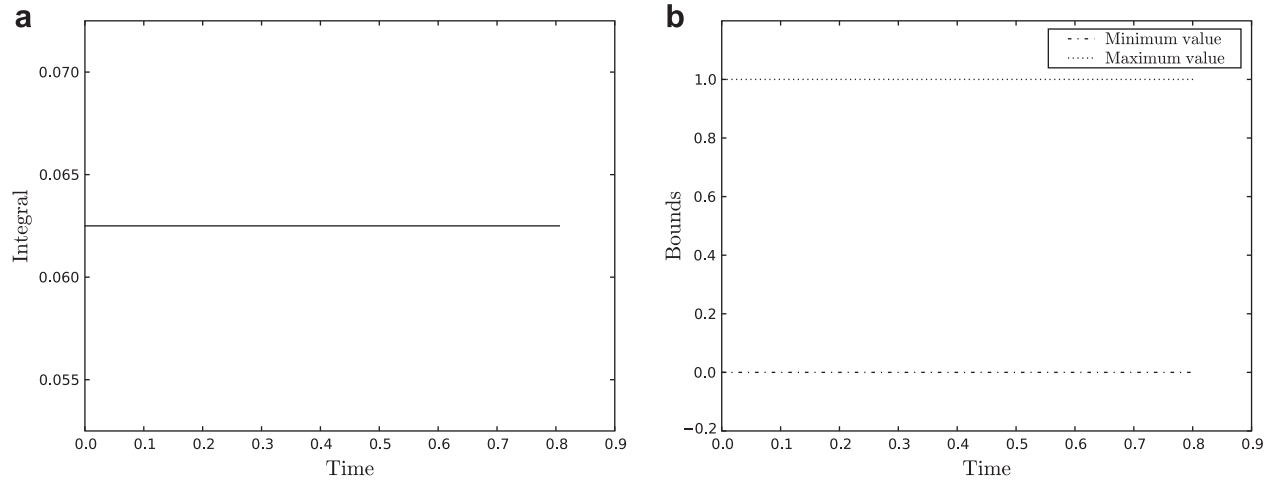

Fig. 16. (a) Integral and (b) bounds of the material volume fraction field for the water column collapse in three dimensions. Note that both the discretisation and interpolation are conservative and bounded.

Maintaining this accuracy in the material interface would be prohibitively expensive on any fixed mesh. As can be seen in Fig. 16, both the discretisation and the interpolation step preserve the integral and bounds of the material volume fraction. This combination of a discontinuous discretisation, conservative and bounded advection, and adaptive remeshing would have been impossible with consistent interpolation.

5. Conclusions

A robust, efficient supermesh construction algorithm has been proposed. With the development of local supermeshing, the algorithm can be applied in three dimensions, and can scale to larger problem sizes than those feasible using a global supermeshing approach. The computational cost of this method has been shown to have a linear scaling with problem size. Two numerical examples have been shown, a lock exchange simulation and a water collapse simulation, which demonstrate the practicality of this approach in two and three dimensions.

The method can be applied to both continuous and discontinuous fields. This enables mesh adaptive simulations using novel discretisations to be conducted at practical computational cost. In particular, the LBB stable $P1_{DG}$–P2 element pair, which has shown excellent geostrophic balance properties in geophysical simulations [8], can be used in mesh adaptive simulations using this interpolation method.

A extension of this method is to impose a Lagrange constraint upon the interpolation. For example, discrete incompressibility of a velocity field u can be imposed via:

$$M'_T u_T = M'_{TD} u_D + C\Psi, \quad C^T u_T = 0, \tag{19}$$

where $M'_T = \mathrm{diag}(M_T)$, $M'_{TD} = \mathrm{diag}(M_{TD})$ and Ψ a scalar potential. C is a discrete gradient matrix: $C = (C^1, \ldots, C^d)^T$ with $(C^k)_{ij} = \int_\Omega \left(\partial \phi_T^{(i)} / \partial x_k\right) \zeta^{(j)}\, dV$ for $i \in \{1, \ldots, \mathcal{N}_T\}$, $j \in \{1, \ldots, \mathcal{N}_\Psi\}$ and $k \in \{1, \ldots, d\}$, where d is the dimension of the domain and $\mathcal{N}_\Psi$ is the number of basis functions $\zeta^{(j)}$ for Ψ. This formulation is equivalent to the Galerkin projection of the Helmholtz decomposition of the donor field u_D, assembled directly on the target mesh. A topic for future research is the implementation and testing of such constrained projections.

The availability of this algorithm makes the exploitation of Galerkin projection between unrelated unstructured meshes practical. Hence this approach enables the accurate, conservative, L_2 optimal interpolation of fields, in two and three dimensions for both continuous and discontinuous function spaces. This has applications in dynamically mesh adaptive numerical modelling, model initialisation and model coupling.

Acknowledgements

The authors wish to acknowledge support from the UK Natural Environment Research Council (Grants NE/C52101X/1, NE/C51829X/1 and NE/H527032/1) and support from the Imperial College High Performance Computing Service and the Oxford Supercomputing Centre. P.E. Farrell would like to thank AWE for their funding of his research through the Institute of Shock Physics.

References

[1] F. Alauzet, M. Mehrenberger, P1-conservative solution interpolation on unstructured triangular meshes, Tech. Rep. RR-6804, INRIA Rocquencourt, Rocquencourt, Le Chesnay, France, 2009, <http://hal.inria.fr/inria-00354509/en/>.
[2] A. Below, J.A. De Loera, J. Richter-Gebert, The complexity of finding small triangulations of convex 3-polytopes, J. Algorithms 50 (2) (2004) 134–167.
[3] E.D. Berger, K.S. McKinley, R.D. Blumofe, P.R. Wilson, Hoard: a scalable memory allocator for multithreaded applications, ACM SIGPLAN Notices 35 (11) (2000) 117–128.
[4] C. Carstensen, Clément Interpolation and Its Role in Adaptive Finite Element Error Control, Operator Theory: Advances and Applications, vol. 168, Birkhäuser Basel, 2006, pp. 27–43 (Chapter 2).
[5] B. Chazelle, An optimal algorithm for intersecting three-dimensional convex polyhedra, SIAM J. Comput. 21 (4) (1992) 671–696.
[6] P. Clément, Approximation by finite element functions using local regularization, RAIRO Anal. Numér. 9 (1975) 77–84.
[7] R. Cools, A. Haegemans, Algorithm 824: CUBPACK: a package for automatic cubature; framework description, ACM Trans. Math. Software 29 (3) (2003) 287–296.
[8] C.J. Cotter, D.A. Ham, C.C. Pain, A mixed discontinuous/continuous finite element pair for shallow-water ocean modelling, Ocean Model. 26 (1–2) (2009) 86–90.
[9] C.J. Cotter, D.A. Ham, C.C. Pain, S. Reich, LBB stability of a mixed Galerkin finite element pair for fluid flow simulations, J. Comput. Phys. 228 (2) (2009) 336–348.
[10] D.R. Davies, J.H. Davies, O. Hassan, K. Morgan, P. Nithiarasu, Investigations into the applicability of adaptive finite element methods to two-dimensional infinite Prandtl number thermal and thermochemical convection, Geochem. Geophys. Geosyst. 8 (5) (2007).
[11] D.H. Eberly, 3D Game Engine Design: A Practical Approach to Real-time Computer Graphics, Morgan Kaufmann, 2001.
[12] H. Edelsbrunner, F.P. Preparata, D.B. West, Tetrahedrizing point sets in three dimensions, J. Symb. Comput. 10 (3/4) (1990) 335–348.
[13] A. El Hraiech, H. Borouchaki, P. Villon, L. Moreau, Mechanical field interpolation, in: L.M. Smith, F. Pourboghrat, J. Cao, T.B. Stoughton, J.-W. Yoon, M.F. Shi, C.-T. Wang, L. Zhang (Eds.), Proceedings of the Sixth International Conference and Workshop on Numerical Simulation of 3D Sheet Metal Forming Process, AIP, Detroit, Michigan, 2005, pp. 201–208.

[14] C. Farhat, M. Lesoinne, P.L. Tallec, Load and motion transfer algorithms for fluid/structure interaction problems with non-matching discrete interfaces: momentum and energy conservation, optimal discretization and application to aeroelasticity, Comput. Methods Appl. Mech. Engrg. 157 (1–2) (1998) 95–114.
[15] P.E. Farrell, Galerkin projection of discrete fields via supermesh construction, Ph.D. Thesis, Imperial College London, 2009.
[16] P.E. Farrell, M.D. Piggott, C.C. Pain, G.J. Gorman, C.R.G. Wilson, Conservative interpolation between unstructured meshes via supermesh construction, Comput. Methods Appl. Mech. Engrg. 198 (33–36) (2009) 2632–2642.
[17] L. Formaggia, S. Perotto, Anisotropic error estimates for elliptic problems, Numer. Math. 94 (1) (2003) 67–92.
[18] P.L. George, H. Borouchaki, Delaunay Triangulation and Meshing: Application to Finite Elements, Hermes, 1998.
[19] C. Geuzaine, B. Meys, F. Henrotte, P. Dular, W. Legros, A Galerkin projection method for mixed finite elements, IEEE Trans. Magn. 35 (3) (1999) 1438–1441.
[20] C. Geuzaine, J.-F. Remacle, Gmsh: a 3-D finite element mesh generator with built-in pre- and post-processing facilities, Int. J. Numer. Methods Engrg. 79 (11) (2009) 1309–1331.
[21] A. Guttman, R-trees: a dynamic index structure for spatial searching, ACM SIGMOD Record 14 (2) (1984) 47–57.
[22] C. Härtel, E. Meiburg, F. Necker, Analysis and direct numerical simulation of the flow at a gravity-current head. Part I. Flow topology and front speed for slip and no-slip boundaries, J. Fluid Mech. 418 (1) (2000) 189–212.
[23] M.W. Heinstein, T.A. Laursen, A three dimensional surface-to-surface projection algorithm for non-coincident domains, Commun. Numer. Methods Engrg. 19 (6) (2003) 421–432.
[24] X. Jiao, M.T. Heath, Common-refinement-based data transfer between non-matching meshes in multiphysics simulations, Int. J. Numer. Methods Engrg. 61 (2004) 2402–2427.
[25] X. Jiao, M.T. Heath, Overlaying surface meshes, Part I: algorithms, Int. J. Comput. Geom. Appl. 14 (6) (2004) 379–402.
[26] B.P. Leonard, The ultimate conservative difference scheme applied to unsteady one-dimensional advection, Comput. Methods Appl. Mech. Engrg. 88 (1) (1991) 17–74.
[27] J. Levon, P. Elie, Oprofile: a system profiler for Linux, 2009, <http://oprofile.sourceforge.net>.
[28] R. Löhner, Robust, vectorized search algorithms for interpolation on unstructured grids, J. Comput. Phys. 118 (2) (1995) 380–387.
[29] Y. Manolopoulos, A. Nanopoulos, A. Papadopoulos, Y. Theodoridis, R-trees: Theory and Applications, Springer, 2005.
[30] S. Micheletti, S. Perotto, Reliability and efficiency of an anisotropic Zienkiewicz–Zhu error estimator, Comput. Methods Appl. Mech. Engrg. 195 (9–12) (2006) 799–835.
[31] D.M. Mount, Geometric intersection, in: J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, CRC Press, Inc., Boca Raton, FL, USA, 1997, pp. 615–630.
[32] T.M. Özgökmen, T. Iliescu, P.F. Fischer, A. Srinivasan, J. Duan, Large eddy simulation of stratified mixing in two-dimensional dam-break problem in a rectangular enclosed domain, Ocean Model. 16 (1–2) (2007) 106–140.
[33] C.C. Pain, A.P. Umpleby, C.R.E. de Oliveira, A.J.H. Goddard, Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations, Comput. Methods Appl. Mech. Engrg. 190 (29–30) (2001) 3771–3796.
[34] G. Parent, P. Dular, J.P. Ducreux, F. Piriou, Using a Galerkin projection method for coupled problems, IEEE Trans. Magn. 44 (6) (2008) 830–833.
[35] J. Peraire, M. Vahdati, K. Morgan, O. Zienkiewicz, Adaptive remeshing for compressible flow computations, J. Comput. Phys. 72 (2) (1987) 449–466.
[36] M.D. Piggott, C.C. Pain, G.J. Gorman, P.W. Power, A.J.H. Goddard, *h*, *r*, and *hr* adaptivity with applications in numerical ocean modelling, Ocean Model. 10 (1–2) (2005) 95–113.
[37] F.P. Preparata, M.I. Shamos, Computational Geometry: An Introduction, second ed., Springer, 1985.
[38] R. Seidel, Convex hull computations, in: J.E. Goodman, J. O'Rourke (Eds.), Handbook of Discrete and Computational Geometry, Chapman & Hall/CRC, 2004, pp. 495–512.
[39] M.I. Shamos, D. Hoey, Geometric intersection problems, in: Proceedings of the 17th Annual Symposium on Foundations of Computer Science, 1976, pp. 208–215.
[40] H. Si, K. Gärtner, Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations, in: B.W. Hanks (Ed.), Proceedings of the 14th International Meshing Roundtable, Springer, San Diego, California, 2005, pp. 147–163.
[41] I.E. Sutherland, G.W. Hodgman, Reentrant polygon clipping, Commun. ACM 17 (1) (1974) 32–42.
[42] P.K. Sweby, High resolution schemes using flux limiters for hyperbolic conservation laws, SIAM J. Numer. Anal. 21 (5) (1984) 995–1011.
[43] Y. Vasilevskii, K. Lipnikov, An adaptive algorithm for quasioptimal mesh generation, Comput. Math. Math. Phys. 39 (9) (1999) 1468–1486.
[44] C.R. Wilson, Modelling multiple-material flows on adaptive unstructured meshes, Ph.D. Thesis, Imperial College London, London, UK, 2009.