INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng (2016)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/nme.5271

Basis mapping methods for forward and inverse problems

Martin Schweiger*,† and Simon Arridge

Department of Computer Science, University College London, London WC1E 6BT, UK

SUMMARY

This paper describes a novel method for mapping between basis representation of a field variable over a domain in the context of numerical modelling and inverse problems. In the numerical solution of inverse problems, a continuous scalar or vector field over a domain may be represented in different finite-dimensional basis approximations, such as an unstructured mesh basis for the numerical solution of the forward problem, and a regular grid basis for the representation of the solution of the inverse problem. Mapping between the basis representations is generally lossy, and the objective of the mapping procedure is to minimise the errors incurred. We present in this paper a novel mapping mechanism that is based on a minimisation of the L^2 or H^1 norm of the difference between the two basis representations. We provide examples of mapping in 2D and 3D problems, between an unstructured mesh basis representative of an FEM approximation, and different types of structured basis including piecewise constant and linear pixel basis, and blob basis as a representation of the inverse basis. A comparison with results from a simple sampling-based mapping algorithm shows the superior performance of the method proposed here. © 2016 The Authors. *International Journal for Numerical Methods in Engineering* Published by John Wiley & Sons Ltd.

Received 14 December 2015; Revised 4 April 2016; Accepted 4 April 2016

KEY WORDS: finite element methods; optimisation; inverse problems

1. INTRODUCTION

Inverse problems arise in almost all areas of science whenever the parameters of a model need to be inferred from observed data. A standard approach is to define a model-based reconstruction problem as the inversion of a mapping $f : \mathrm{X} \mapsto \mathrm{Y}$ between function spaces X (solution space) and Y (data space). The topology of these spaces is determined by the physical nature of the parameters and data for a given problem. Except in a limited set of special cases, the mapping f cannot be inverted explicitly, and the problem is set in variational form. The task of the reconstruction is then to find parameters $\hat{x}$ that minimise an objective function $\psi(x)$, defined as a norm between model and measurement data y^{obs}:

$$\psi(x) = \mathcal{D}\left(y^{\text{obs}}, f(x)\right) + \tau\mathcal{R}(x) \tag{1}$$

$$\hat{x} = \arg\min_{x} \psi(x) \tag{2}$$

where $\mathcal{D}$ is a *data fit* function (typically the negative log-likelihood) and $\mathcal{R}$ is *regularisation* term with regularisation parameter τ.

In many inverse problems in imaging, the forward operator represents the propagation of waves or radiation, such as electromagnetic, acoustic or optical, and needs to be modelled computationally. For many problems, f is then represented by a partial differential equation (PDE) in a domain of interest Ω, with appropriate boundary conditions on boundary $\partial\Omega$, and a projection operator that

*Correspondence to: M. Schweiger, Department of Computer Science, University College London, London WC1E 6BT, UK.

†E-mail: M.Schweiger@ucl.ac.uk

maps the solution of the PDE to a discrete set of measurements. The inverse problem is then one of *parameter identification* and is often non-linear and strongly ill-posed. Examples include the following: Electrical Impedance Tomography, where the forward model is the generalised Laplace Equation (a low-frequency approximation of Maxwell's Equations) and the parameter of the inverse problem is admittivity [1]; Diffuse Optical Tomography, and its variants including Fluorescence Diffuse Optical Tomography where the forward model is the diffusion equation (a high-scattering approximation of the Radiative Transfer Equation) and the parameter of the inverse problem is absorption and/or diffusivity [2]; and Seismic Tomography, where the forward model is a hyperbolic equation for acoustic propagation (a simplification of the Elasticity Equation), and the parameter of the inverse problem represents various mechanical characteristics of rock such as density and/or stiffness [3].

In most practical applications, evaluation of f requires a numerical solution strategy that discretises the continuous into a finite-dimensional problem and represents the parameters $x(\mathbf{r})$ and field variables by a finite-dimensional basis expansion $\mathbb{U}$. We denote the discretised forward problem by

$$f^u : \mathrm{X}^u \mapsto \mathrm{Y} \tag{3}$$

where x^u is the approximation of spatial parameters x in basis $\mathbb{U}$. The most common numerical approaches include the finite element method (FEM), finite difference method and boundary element method.

In this paper, we consider FEM as the numerical realisation of the forward problem, where Ω is subdivided into an unstructured mesh of non-overlapping elements of simple shape, such as triangles in 2D problems and tetrahedra in 3D problems, that define $\mathbb{U}$ as a local, piecewise polynomial basis expansion.

The choice of basis for the forward problem is governed by considerations of computational stability and accuracy of the numerical model. This is often not an optimal choice for the inverse problem, where it may be more appropriate to adapt the basis expansion to physical limits of resolution of the reconstruction and to enforce smoothness constraints in the admissible solutions.

It is therefore often desirable to represent the inverse problem Equation 2 in a different basis representation $\mathbb{V}$ at a different resolution and with different basis functions:

$$\hat{x}^v = \arg\min_{x^v} \psi(x^v) \tag{4}$$

For example, $\mathbb{V}$ might be a regular pixel or voxel grid representing an image of the reconstructed parameter distribution or a blob basis that imposes smoothness constraints on the reconstruction.

Where $\mathbb{U}$ and $\mathbb{V}$ are different, a mapping must be devised to transfer the expansion of a variable $x(\mathbf{r})$ between basis representations x^u and x^v. For example, in an iterative minimisation approach, an updated parameter set x^v must be mapped to x^u to evaluate the forward model for computing ψ. The mapping algorithm should be optimal in the sense that the discretisation error of the mapped basis expansion is minimal. In general, the mapping is non-trivial, because the basis expansions are non-conforming and require the integration of basis functions over partial support areas. Previously, [4] we have presented a simple mapping method that relied on $\mathbb{U}$ being of higher resolution than $\mathbb{V}$ such that basis functions of $\mathbb{V}$ can be adequately represented by an expansion in $\mathbb{U}$, reducing the mapping problem to operations in $\mathbb{U}$ that can be performed with standard FEM tools.

In this paper, we propose an optimisation-based approach to define the mapping, which avoids these limitations; either or both bases may be of arbitrary, variable or multi-resolution. The key technique is the splitting of mesh elements comprising basis $\mathbb{U}$ such that the subdivided elements align with the boundaries of support of the basis functions of $\mathbb{V}$. This allows to compute the integrals of products of mixed basis functions. The integrals are evaluated with a quadrature rule that is either exact, where $\mathbb{V}$ is piecewise polynomial, or approximate in the general case.

2. METHOD

2.1. *Finite element basis*

We consider the *model* to be a finite element discretisation of a PDE over a simply connected domain $\Omega \subset \mathbb{R}^d$, $d = \{2, 3\}$ with boundary $\partial\Omega$. Ω is divided into P non-overlapping elements e_i, $i = 1..P$

with element subdomain $S(e_i) \subset \tilde{\Omega}$ such that $\cup_{i=1}^{P} S(e_i) = \tilde{\Omega}$, where $\tilde{\Omega}$ is an approximation of Ω that replaces boundary $\partial\Omega$ with a polygonal or polyhedral approximation $\partial\tilde{\Omega}$.

We define a piecewise polynomial basis expansion $\mathbb{U}^{(k)}$ of order k composed of N basis functions $u_i(\mathbf{r}),\ i = 1..N,\ \mathbf{r} \in \tilde{\Omega}$ with limited support, such that a continuous variable $x(\mathbf{r})$ in Ω is represented by the basis approximation

$$x(\mathbf{r}) \approx x^u(\mathbf{r}) = \sum_{i=1}^{N} X_i^u u_i(\mathbf{r}),\ \mathbf{r} \in \Omega \tag{5}$$

where $\mathbf{X}^u = \{X_1^u, \dots, X_N^u\} \in \mathbb{R}^N$ is the coefficient vector representing x in basis $\mathbb{U} = \{u_1, \dots, u_N\}$. For $k = 0$, we have $N = P$ and u_i are piecewise constant, $u_i(\mathbf{r}) = 1$ if $\mathbf{r} \in S(e_i);\ 0$ otherwise. More frequently, orders $k \geqslant 1$ are chosen that provide a piecewise C^k continuous basis expansion. In this case, each u_i is associated with a node $n_i \in \mathbb{R}^d,\ i = 1..N$ that participates in a subset of elements $e_{j(i)}$, and u_i is a piecewise polynomial function of order k with support $S(u_i) = \cup_{j(i)} S(e_j) \subset \tilde{\Omega}$. For $k = 1$, the nodes are given by the element vertices. For $k > 1$, additional nodes are required along element edges, faces and volumes to define u_i. For simplicity, we restrict the discussion in this paper to polynomial basis functions, and most frequently to $k = 1$. To simplify notation, we abbreviate $\mathbb{U} = \mathbb{U}^{(1)}$. The methods discussed after this can be directly applied to other choices of FEM basis expansion.

2.2. *Inverse problem basis*

While the choice of model basis $\mathbb{U}$ is governed by considerations of numerical stability and accuracy of the FEM forward model, it may not be suitable to represent the results of the solution of an inverse problem, nor be a natural representation of the parameters of the forward model. We therefore introduce a separate inverse problem basis $\mathbb{V}$ consisting of M basis functions $v_j(\mathbf{r}),\ j = 1..M,\ \mathbf{r} \in \Omega$ of limited support $S(v_j) \subset \Omega$. $\mathbb{V}$ may be chosen to match the physical resolution limits of the problem or to impose a specific level of smoothness in the images it can represent.

$$x(\mathbf{r}) \approx x^v(\mathbf{r}) = \sum_{i=1}^{N} X_i^v v_i(\mathbf{r}),\ \mathbf{r} \in \Omega \tag{6}$$

A large number of choices for $\mathbb{V}$ can be considered. In the simplest case, $\mathbb{V}$ may represent a piecewise constant pixel or voxel image, such that $v_j(\mathbf{r}) = 1$; if $\mathbf{r} \in$ pixel j; 0 otherwise. Alternatively, in analogy to the unstructured FEM basis, $\mathbb{V}$ can be continuous polynomial basis with v_j associated with the vertices of the basis grid and support over neighbouring pixels according to the order of expansion.

Further, $\mathbb{V}$ can represent a *blob basis*, consisting of radially symmetric basis functions that depend only on the distance $d = ||\mathbf{r} - \mathbf{R}_j||$ from a reference point $\mathbf{R}_j$, and are truncated to a support radius $d_{\max}$, such that

$$S(v_j) = \left\{\mathbf{r} \in \Omega : ||\mathbf{r} - \mathbf{R}_j|| \leqslant d_{\max}\right\} \tag{7}$$

and $v_j(\mathbf{r}) = 0$ if $d > d_{\max}$. The blob centres $\mathbf{R}_j$ may be arranged in a regular grid or some other distribution [5, 6]. Blobs are often preferred in image reconstruction problems because they enforce an inherent smoothness in the solution that can be controlled by the choice of v_i and by selecting specific shape parameters and the support radius [7].

Let $v_j(d)$ denote the radial profile of $v_j(\mathbf{r})$. Different choices for $v_j(d)$ exist [4], including

- Kaiser-Bessel window functions,

$$v_j(d) = \frac{1}{I_m(\alpha)} \left[\sqrt{1 - (d/d_{\max})^2}\right]^m I_m\left[\alpha\sqrt{1 - (d/d_{\max})^2}\right],\ d \leqslant d_{\max}, \tag{8}$$

where I_m is the modified Bessel function of order m and α is an adjustable shape parameter,

- Gaussian with truncated support,

$$v_j(d) = \exp\left(-d^2/\sigma^2\right),\ d \leqslant d_{\max}, \tag{9}$$

  where σ defines the shape,
- cubic B-spline,

$$v_j(d) = 4\sum_{k=0}^{4} \frac{(d-d_k)_+^3}{\omega(d_k)},\ d \leqslant d_{\max}, \tag{10}$$

  where $d_k = (k-2)d_{\max}/2$ and

$$\omega(d_k) = \Pi_{j=0,j\neq k}^{4}\left(d_k - d_j\right) \tag{11}$$

$$(d-d_k)_+^3 = \begin{cases} (d-d_k)^3 & \text{if } d > d_k \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

2.3. Basis mapping

Having defined forward model basis $\mathbb{U}$ and inverse problem basis $\mathbb{V}$, we need to define mappings $\mathbb{U} \to \mathbb{V}$ and $\mathbb{V} \to \mathbb{U}$ that map the basis representation x^u in $\mathbb{U}$ of a field variable $x(\mathbf{r})$ to the corresponding representation x^v in $\mathbb{V}$ and vice versa. In general, the mapping will not be lossless, because x^u cannot be represented exactly in $\mathbb{V}$. We therefore define the mapping problem as follows: given a coefficient vector $\mathbf{X}^u$ for basis $\mathbb{U}$, find the corresponding vector $\mathbf{X}^v$ of basis coefficients in $\mathbb{V}$ such that a norm of the difference $||x^u - x^v||_C$, for some choice of a normed linear space C, is minimised. We have previously [4] presented a simplistic sampling-based method of basis mapping, where the coefficients given in one basis expansion are projected onto the other basis expansion by sampling at the node or grid positions of the target basis. The mapping $\mathbb{V} \to \mathbb{U}$ is then given by

$$X_i^u \approx x^v(n_i) = \sum_{j=1}^{M} X_j^v v_j(n_i) \tag{13}$$

where $v_j(n_i)$ denotes the value of basis function v_j on node n_i of basis $\mathbb{U}$. Likewise, the mapping $\mathbb{U} \to \mathbb{V}$ is given by

$$X_j^v \approx x^u(g_j) = \sum_{i=1}^{N} X_i^u u_i(g_j) \tag{14}$$

where g_j denotes the position of the grid point associated with basis function v_j. We can thus define a *projection matrix* $\mathsf{G}^{uv} \in \mathbb{R}^{N\times M}$ for mapping the coefficients of the two basis representations from $\mathbb{V}$ to $\mathbb{U}$ (Algorithm 1).

Algorithm 1 Sampling-based mapping (SM) $\mathbb{V} \to \mathbb{U}$

```
1: for $i = 1..N$ do
2:     for $j = 1..M$ do
3:         $G_{ij}^{uv} = v_j(n_i)$
4:     end for
5: end for
6: $\mathbf{X}^u = \mathsf{G}^{uv}\mathbf{X}^v$
```

The construction of the reverse projection matrix $\hat{\mathsf{G}}^{vu} \in \mathbb{R}^{M\times N}$ for mapping from $\mathbb{U}$ to $\mathbb{V}$ is performed analogously.

In the following section, we will introduce a more rigorous basis mapping approach that provides the solution as the optimisation of an error functional.

2.4. Error norms for basis mapping

Define an objective function based on the L^2 norm of the difference between the basis representations,

$$\begin{aligned}\Phi_{L^2}(\mathbf{X}^u, \mathbf{X}^v) &= \frac{1}{2}\int_\Omega ||x^u(\mathbf{r}) - x^v(\mathbf{r})||^2 d\mathbf{r} \\ &= \frac{1}{2}\mathbf{X}^{u^T}\mathrm{B}^{uu}\mathbf{X}^u - \mathbf{X}^{u^T}\mathrm{B}^{uv}\mathbf{X}^v + \frac{1}{2}\mathbf{X}^{v^T}\mathrm{B}^{vv}\mathbf{X}^v\end{aligned} \tag{15}$$

where

$$\begin{aligned}\mathrm{B}^{uu} \in \mathbb{R}^{N\times N} \text{with } B^{uu}_{ij} &= \textstyle\int_\Omega u_i(\mathbf{r})u_j(\mathbf{r})d\mathbf{r} \\ \mathrm{B}^{vv} \in \mathbb{R}^{M\times M} \text{with } B^{vv}_{ij} &= \textstyle\int_\Omega v_i(\mathbf{r})v_j(\mathbf{r})d\mathbf{r} \\ \mathrm{B}^{uv} \in \mathbb{R}^{N\times M} \text{with } B^{uv}_{ij} &= \textstyle\int_\Omega u_i(\mathbf{r})v_j(\mathbf{r})d\mathbf{r}\end{aligned} \tag{16}$$

We define B^{uu} and B^{vv} as *self-mass-matrices*, that is, the symmetric positive definite mass matrices of the respective bases $\mathbb{U}$ and $\mathbb{V}$. We define B^{uv} as the *co-mass-matrix* containing mixed integrals of products of basis coefficients from $\mathbb{U}$ and $\mathbb{V}$.

The mapping of a function x^u in $\mathbb{U}$ to x^v in $\mathbb{V}$ is then given by minimising

$$\hat{\mathbf{X}}^v = \arg\min_{\mathbf{X}^v} \Phi(\mathbf{X}^u, \mathbf{X}^v) \tag{17}$$

which is found by solving

$$\mathrm{B}^{vv}\hat{\mathbf{X}}^v = \mathrm{B}^{uv^T}\mathbf{X}^u \tag{18}$$

Likewise, the mapping from $\mathbb{V}$ to $\mathbb{U}$ of function x^v is given by

$$\hat{\mathbf{X}}^u = \arg\min_{\mathbf{X}^u} \Phi(\mathbf{X}^u, \mathbf{X}^v) \tag{19}$$

$$\mathrm{B}^{uu}\hat{\mathbf{X}}^u = \mathrm{B}^{uv}\mathbf{X}^v \tag{20}$$

In the following, we will denote the mapping procedure given by Equations 18 and 20 as the *least squares mapping* (LSM) method, to distinguish it from the sampling mapping (SM) method outlined in Algorithm 1.

Instead of the previous L^2 norm, we also consider the Sobolev norm H^1:

$$\begin{aligned}\Phi_{H^1,\alpha}(\mathbf{X}^u, \mathbf{X}^v) &= \frac{1}{2}\left(\int_\Omega |x^u(\mathbf{r} - x^v(\mathbf{r})|^2 d\mathbf{r} + \alpha\int_\Omega |\nabla x^u(\mathbf{r}) - \nabla x^v(\mathbf{r})|^2 d\mathbf{r}\right) \\ &= \frac{1}{2}\left[\mathbf{X}^{u^T}(\mathrm{B}^{uu} + \alpha\mathrm{D}^{uu})\mathbf{X}^u\right] - \mathbf{X}^{u^T}(\mathrm{B}^{uv} + \alpha\mathrm{D}^{uv})\mathbf{X}^v \\ &+ \frac{1}{2}\left[\mathbf{X}^{v^T}(\mathrm{B}^{vv} + \alpha\mathrm{D}^{vv})\mathbf{X}^v\right]\end{aligned} \tag{21}$$

where α is a scalar, positive parameter and

$$\begin{aligned}\mathrm{D}^{uu} \in \mathbb{R}^{N\times N} \text{with } D^{uu}_{ij} &= \textstyle\int_\Omega \nabla u_i(\mathbf{r})\cdot\nabla u_j(\mathbf{r})d\mathbf{r} \\ \mathrm{D}^{vv} \in \mathbb{R}^{M\times M} \text{with } D^{vv}_{ij} &= \textstyle\int_\Omega \nabla v_i(\mathbf{r})\cdot\nabla v_j(\mathbf{r})d\mathbf{r} \\ \mathrm{D}^{uv} \in \mathbb{R}^{N\times M} \text{with } D^{uv}_{ij} &= \textstyle\int_\Omega \nabla u_i(\mathbf{r})\cdot\nabla v_j(\mathbf{r})d\mathbf{r}\end{aligned} \tag{22}$$

In analogy with the simpler least-squares case, we define D^{uu} and D^{vv} as *self-stiffness-matrices*, that is, the symmetric positive stiffness matrices of the respective bases $\mathbb{U}$ and $\mathbb{V}$ and D^{uv} as the *co-stiffness-matrix* containing mixed integrals of scalar products of the gradients of basis coefficients from $\mathbb{U}$ and $\mathbb{V}$.

2.5. Element subdivision

The computation of the co-mass-matrix B^{uv} in Equation 16 and the co-stiffness-matrix D^{uv} in Equation 22 requires the calculation of the integrals of mixed products $u_i v_j$ and their derivatives $\nabla u_i \cdot \nabla v_j$. These are generally not trivial to compute, because $\mathbb{U}$ and $\mathbb{V}$ are non-conforming, and the intersection of the areas of support of $S(u_i) \cap S(v_j)$ may not coincide with element boundaries in either basis. We solve this problem by element subdivision: split the elements representing the support of u_i into sub-elements along a cutting surface defined by the support boundary of v_j. Then compute the integrals over all sub-elements within the support of v_j and sum them up. For some choices of $\mathbb{U}$ and $\mathbb{V}$, the integrals may be evaluated analytically, for example, where both $\mathbb{U}$ and $\mathbb{V}$ are piecewise polynomial. Otherwise, a numerical quadrature scheme may be employed.

Element subdivision and integration over partial elements is a well-known problem in different applications of the FEM. It occurs for example in the computation of non-matching and overlapping meshes, where a mesh representing an object is overlayed on a mesh representing a background [8, 9], where elastic materials are inserted into an elastic background of different properties [10], or where an elastic body is immersed into a fluid [11]. Cut element problems also arise in ficticious domain problems, where physical domain boundaries may intersect a mesh [12, 13]. A common approach for treating interface problems on unfitted meshes is the use of Nitsche's method [14], which allows for discontinuities across element volumes. Dolbow *et al.* [15] and Hansbo *et al.* [16] use Nitsche's method for computing jumps in a field without re-aligning the mesh.

The subdivision of tetrahedral elements has been described in the context of physical cutting of materials, for example, soft tissue cutting during surgical interventions [17, 18], fracture modelling [19] or surgical reconstruction [20].

Where $\mathbb{V}$ represents a regular pixel or voxel basis, with planar boundaries between basis functions, the resulting subdivided elements are polyhedra, which can then be further subdivided into simple elements such as triangles and tetrahedra, and standard FEM techniques can be applied to compute the integrals. If $\mathbb{V}$ represents a more complex basis with non-planar boundaries, such as a blob basis composed of overlapping radially symmetric basis functions, the subdivided elements have non-trivial curved boundaries. In this case, we propose to represent the boundaries by a polygonal approximation and proceed as before by further subdividing into triangles and tetrahedra.

In general, for arbitrary basis definitions $\mathbb{V}$, a numerical quadrature scheme may have to be employed to compute B^{uv} and D^{uv}.

In the following subsections we describe the methods of element subdivision for 2D problems using an unstructured triangular basis and 3D problems using an unstructured tetrahedral basis for $\mathbb{U}$.

Algorithm 2 Element splitting and matrix assembly in 2D

```
 1: for $j = 1..M$ do
 2:     approximate $S(v_j)$ with polygonal approximation $\tilde{S}$, if required
 3:     for $i = 1..N$ do
 4:         $B_{ij}^{uv} \leftarrow 0,\ D_{ij}^{uv} \leftarrow 0$
 5:         for each element $e_k \subset S(u_i)$ do
 6:             compute $W = S(e_k) \cap S(v_j)$ using Sutherland-Hodgman
 7:             construct $P_k$ sub-triangles $\tilde{e}_m$ with $\cup_{m=1}^{P_k} S(\tilde{e}_m) = W$
 8:             for $m = 1..P_k$ do
 9:                 $B_{ij}^{uv} \leftarrow B_{ij}^{uv} + \int_{S(\tilde{e}_m)} u_i(\mathbf{r}) v_j(\mathbf{r}) d\mathbf{r}$
10:                 $D_{ij}^{uv} \leftarrow D_{ij}^{uv} + \int_{S(\tilde{e}_m)} \nabla u_i(\mathbf{r}) \cdot \nabla v_j(\mathbf{r}) d\mathbf{r}$
11:             end for
12:         end for
13:     end for
14: end for
```

2.5.1. Subdivision in 2D.. For 2D problems, we consider triangular elements with piecewise polynomial basis functions for the FEM basis $\mathbb{U}$, while $\mathbb{V}$ may be either a regular pixel basis with piecewise bi-polynomial basis functions or a blob basis with radially symmetric basis functions.

The element-splitting algorithm for the calculation of co-matrices B^{uv} and D^{uv} is shown in Algorithm 2. For each combination of basis functions $v_j,\ j = 1..M$ and $u_i,\ i = 1..N$, loop over all triangles $e_k \subset S(u_i)$. Calculate the intersection $W = S(e_k) \cap S(v_j)$, using a Sutherland–Hodgman [21] algorithm. If $S(v_j)$ is not polygonal, as is the case for blob basis functions, we replace $S(v_j)$ with an n-sided polygonal approximation $\tilde{S}(v_j)$ (Figure 1(b)). The accuracy of the mapping and its computational cost depend on n. For the computations in this paper, support areas for 2D blob basis functions were approximated by a regular polygon with $n = 32$. As both $S(e_k)$ and $S(v_j)$ (or its approximation) are polygonal and convex, W is also polygonal and convex. Let W be a $(P_k + 2)$-sided polygon with $P_k \geqslant 1$ (omitting the trivial case $W = \emptyset$). W can then easily be subdivided into P_k sub-triangles $\tilde{e}_m$ with $\cup_{m=1}^{P_k} S(\tilde{e}_m) = S(e_k)$ (Figure 1). The integrals $\int u_i v_j d\mathbf{r}$ for B_{ij}^{uv} and $\int \nabla u_i \cdot \nabla v_j d\mathbf{r}$ for D_{ij}^{uv} are computed over each $\tilde{e}_m$ and summed up.

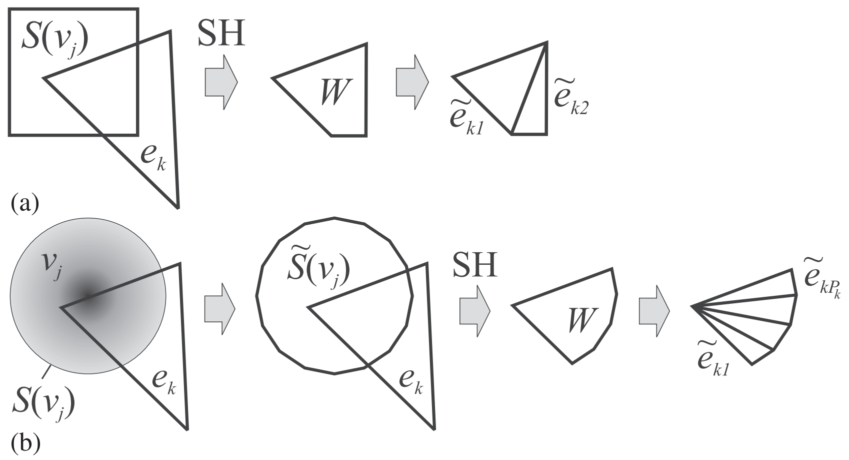

Figure 1. Basis support intersection computation with Sutherland–Hodgson (SH) and triangulation of intersection (a) for regular pixel basis function v_j and (b) for polygonal approximation of support area of a blob basis with circular support.

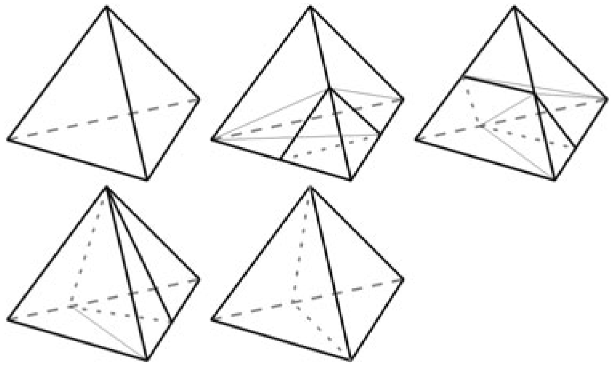

Figure 2. Splitting of a tetrahedron (top left) by a plane. The possible cases are as follows: (i) splitting plane cuts off a single vertex (top centre). This results in 1+3 subtetrahedra. (ii) Splitting plane cuts off two vertices (top right). This results in 3+3 subtetrahedra. Special cases are also considered, where (iii) a single vertex is located on the cutting plane (bottom left). This results in 1+2 subtetrahedra, and (iii) two vertices are located on the cutting plane (bottom centre). This results in 1+1 subtetrahedra. The case where three vertices are on the cutting plane is trivial and does not require splitting.

2.5.2. Subdivision in 3D. For 3D problems, we consider tetrahedral elements for the finite element basis $\mathbb{U}$. For basis $\mathbb{V}$, we use either a regular voxel grid with piecewise constant or trilinear basis functions, or a blob basis with radially symmetric basis functions.

We employ a 3D extension of the Sutherland–Hodgman algorithm for computing the intersections of the basis functions. In analogy to the 2D case, the spherical support area $S(v_j)$ of blob basis functions is approximated by a regular polyhedron. For the results presented in this paper, we used a 20-sided icosahedron as an approximation $\tilde{S}(v_j)$ to $S(v_j)$.

Algorithm 3 Element splitting and matrix assembly in 3D

```
1:  for $j = 1..M$ do
2:      approximate $S(v_j)$ with polyhedral approximation, if required
3:      for $i = 1..N$ do
4:          $B_{ij}^{uv} \leftarrow 0,\ D_{ij}^{uv} \leftarrow 0$
5:          for each element $e_k \subset S(u_i)$ do
6:              define single-element sub-mesh $m := \tilde{e}_1 := e_k$
7:              for each face $F_m$ of $S(v_j)$ do
8:                  for each element $\tilde{e}_n$ of $m$ do
9:                      if $\tilde{e}_n$ entirely inside $F_m$ then
10:                         continue
11:                     else if $\tilde{e}_n$ entirely outside $F_m$ then
12:                         remove $\tilde{e}_n$ from $m$
13:                     else
14:                         split $\tilde{e}_n$ along $F_m$ into sub-elements $\tilde{e}_{np}$
15:                         remove $\tilde{e}_n$ from $m$
16:                         for each sub-element $\tilde{e}_{np}$ inside $F_m$ do
17:                             add $\tilde{e}_{np}$ to $m$
18:                         end for
19:                     end if
20:                 end for
21:             end for
22:             for each element $\tilde{e}_n$ in $m$ do
23:                 $B_{ij}^{uv} \leftarrow B_{ij}^{uv} + \int_{S(\tilde{e}_n)} u_i(\mathbf{r}) v_j(\mathbf{r}) d\mathbf{r}$
24:                 $D_{ij}^{uv} \leftarrow D_{ij}^{uv} + \int_{S(\tilde{e}_n)} \nabla u_i(\mathbf{r}) \cdot \nabla v_j(\mathbf{r}) d\mathbf{r}$
25:             end for
26:         end for
27:     end for
28: end for
```

Algorithm 3 shows the process of element splitting employed for calculation of co-matrices B_{ij}^{uv} and D_{ij}^{uv} for 3D problems. The algorithm is similar to the 2D case, but in order to reduce the clipping process to a sequence of elementary operations of splitting a tetrahedron along a single intersecting plane, we perform the splitting along each face of $S(v_j)$ iteratively and re-form a clipped tetrahedral volume mesh at each step. The process starts with the single-element mesh $m = e_k$. For each face of $S(v_j)$, the elements of m are split by the plane defined by this face. Elements entirely external are discarded, elements entirely internal are retained and intersecting elements are split and replaced in m by their internal fragments.

There are two possible cases for the intersection of a tetrahedron at a single plane (Figure 2):

(i) The vertex nodes are split 1-3. In this case, we can split the tetrahedron into one subtetrahedron on one side of the cutting plane and three tetrahedra on the other side.
(ii) The vertex nodes are split 2-2. In this case, we can split the tetrahedron into three sub-tetrahedra on each side of the cutting plane.

The cases where one or more of the vertices of the tetrahedron coincide with the clipping plane are considered separately, to avoid the generation of degenerate subtetrahedra with coinciding vertices. If exactly one vertex is on the cutting plane, the tetrahedron can be split into three subtetrahedra. If two vertices are on the cutting plane, the tetrahedron can be split into two subtetrahedra. The trivial case where three vertices are on the cutting plane does not require subdivision.

Figure 3 shows the result of intersecting and subdividing a tetrahedron e_k with the icosahedral approximation of the support $S(v_j)$ of a blob basis function v_j.

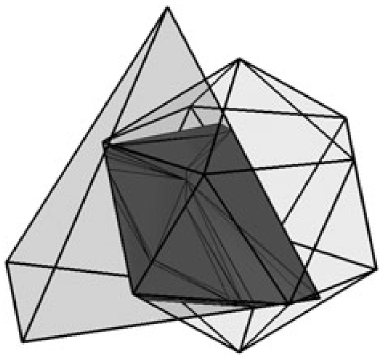

Figure 3. Example for the subdivision of a tetrahedron along the surface of an icosahedron, used to approximate the support surface of a spherically symmetric basis function v_j.

Algorithm 4 3D Element splitting and matrix assembly for regular voxel basis $\mathbb{V}$

```
1: for $i = 1..N$ do
2:     for each element $e_k \subset S(u_i)$ do
3:         define single-element sub-mesh $m := \tilde{e}_1 := e_k$
4:         assign label $l_{xyz}(\tilde{e}_1) = \{0, 0, 0\}$ to $\tilde{e}_1$
5:         for $d = \{x, y, z\}$ do
6:             for each inter-voxel plane $p_k(d)$ in direction $d$ do
7:                 for each element $\tilde{e}_n$ of $m$ do
8:                     if $\tilde{e}_n$ entirely left of $p_k(d)$ then
9:                         keep label $l_d(\tilde{e}_n)$
10:                    else if $\tilde{e}_n$ entirely right of $p_k(d)$ then
11:                        $l_d(\tilde{e}_n) \leftarrow l_d(\tilde{e}_n) + 1$
12:                    else
13:                        split $\tilde{e}_n$ at $p_k(d)$ and replace $\tilde{e}_n$ with the fragments
14:                        for all fragments right of $p_k(d)$, $l_d \leftarrow l_d + 1$
15:                    end if
16:                end for
17:            end for
18:        end for
19:        for $j = 1..M$ do
20:            find grid coordinate $g = \{x, y, z\}$ of voxel $j$
21:            for each element $\tilde{e}_n$ in $m$ do
22:                if $g = l(\tilde{e}_n)$ then
23:                    $B_{ij}^{uv} \leftarrow B_{ij}^{uv} + \int_{S(\tilde{e}_n)} u_i(\mathbf{r}) v_j(\mathbf{r}) d\mathbf{r}$
24:                    $D_{ij}^{uv} \leftarrow D_{ij}^{uv} + \int_{S(\tilde{e}_n)} \nabla u_i(\mathbf{r}) \cdot \nabla v_j(\mathbf{r}) d\mathbf{r}$
25:                end if
26:            end for
27:        end for
28:    end for
29: end for
```

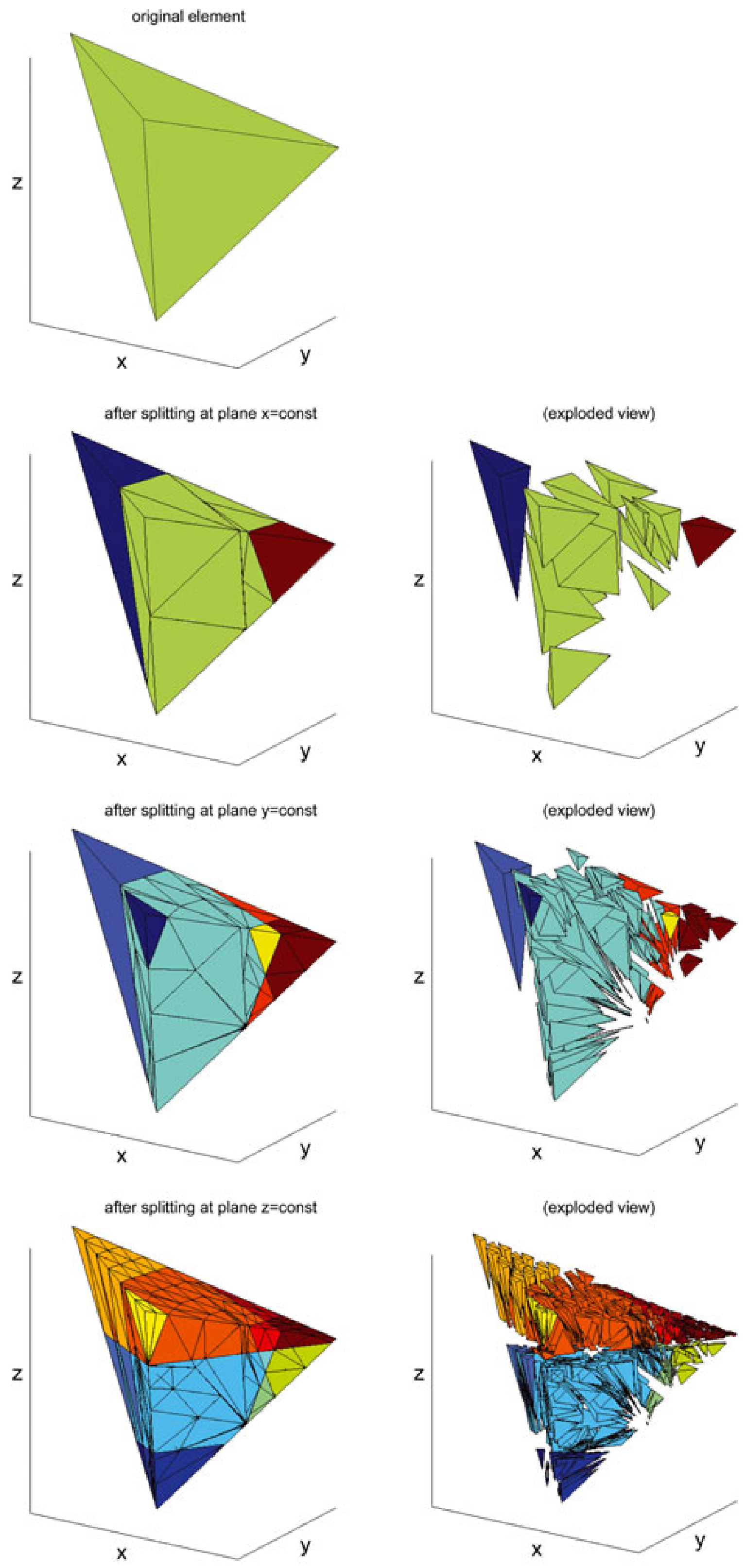
original element
after splitting at plane x=const
(exploded view)
after splitting at plane y=const
(exploded view)
after splitting at plane z=const
(exploded view)
x
y
z

Figure 4. Result of recursive subdivisions of a tetrahdron at voxel boundaries in the x, y and z planes, using Algorithm 4. Element colours indicate voxel labels. The images in the right column show an exploded view of the sub-elements.

If $\mathbb{V}$ is a regular voxel basis with cuboid basis support $S(v_j)$, we can re-arrange the splitting process to make it more efficient. We exploit the fact that many basis functions v_j have coplanar faces and thus share common cutting planes. We discard the outer loop over basis functions v_j, which would result in re-calculating identical cutting planes for different basis functions, and instead split the object tetrahedron e_k on all parallel cutting planes it intersects, retaining both the subtetrahedra on the left and right of each cutting operation, and labelling the generated subtetrahedra according to the cutting slice they are located in. This process is repeated with the cutting planes in the other two spatial directions. The result is a sub-mesh over the object tetrahedron where each tetrahedral sub-element is labelled according to the voxel it is located in. The integral contributions for B^{uv}_{ij} and D^{uv}_{ij} are then computed for all participating voxels j (Algorithm 4).

Figure 4 shows the result of such a subdivision of an object tetrahedron (top image) along the cutting planes of a voxel basis in the x, y and z directions. The colours indicate the voxel labels of each of the generated sub-tetrahedra.

2.6. *Self-mass-matrix calculations*

The matrices B^{uu} in Equation 16 and D^{uu} in Equation 22 can be computed with standard tools of finite element analysis. For example, there exist analytic formulae for products of polynomial shape functions over simple elements such as triangles and tetrahedra [22]. More complicated element types, such as isoparametric elements with curved boundaries, may require a numerical quadrature scheme.

Algorithm 5 Computation of B^{vv} for general blob basis $\mathbb{V}$

```
1: for $i = 1..M$ do
2:     for $j = 1..M$ do
3:         $B^{vv}_{ij} \leftarrow 0$
4:         if $||\mathbf{R}_i - \mathbf{R}_j|| < d_{\max,i} + d_{\max,j}$ then
5:             Compute $W = \tilde{S}_i \cap \tilde{S}_j$ for polygonal approximations $\tilde{S}$ of $S$
6:             Subdivide $W$ into $n$ sub-elements $\tilde{e}_k$ with Algorithm 2 or 3
7:             for $k = 1..n$ do
8:                 $B^{vv}_{ij} \leftarrow B^{vv}_{ij} + \int_{S(\tilde{e}_k)} v_i(\mathbf{r}) v_j(\mathbf{r}) d\mathbf{r}$
9:             end for
10:        end if
11:    end for
12: end for
```

Algorithm 6 Computation of B^{vv} for the special case of a regular blob basis $\mathbb{V}$

```
1: Initialise lookup table $b = 0$
2: for $i = 1..M$ do
3:     for $j = 1..M$ do
4:         $B^{vv}_{ij} \leftarrow 0$
5:         $d = ||\mathbf{R}_i - \mathbf{R}_j||$
6:         if $d < d_{\max,i} + d_{\max,j}$ then
7:             if $d$ present in $b$ then
8:                 $B^{vv}_{ij} \leftarrow b(d)$
9:             else
10:                Compute $B^{vv}_{ij}$ with Algorithm 5
11:                Insert $b(d) = B^{vv}_{ij}$
12:            end if
13:        end if
14:    end for
15: end for
```

The computation of B^{vv} and D^{vv} is straightforward for polynomial basis $\mathbb{V}$. For blob bases, the computation of B^{vv} becomes non-trivial, because it requires the integral of the product of two radial basis functions v_i and v_j over their region of overlap $W = S(v_i) \cap S(v_j)$. We replace S by their polygonal approximations $\tilde{S}$, compute W with the method described in Algorithm 2 or 3, and divide the overlap into sub-elements. The integrals $\int v_i v_j$ are evaluated with a numerical quadrature scheme over the sub-elements and summed up. The process is shown in Algorithm 5.

If the blob basis is arranged in a regular grid, only a small number of different overlap configurations occur, which makes computation of B^{vv} computationally inexpensive compared with B^{uv}. The number of neighbours contributing to each row of B^{vv} depends on the size of $d_{\max}$ relative to the grid spacing. Algorithm 6 shows the modified process.

2.7. *Co-mass-matrix calculations*

In order to compute the integrals in Equation 16 for the elements of B^{uv}, we employ Algorithm 2 for 2D problems and Algorithm 3 for 3D problems to perform the required element splitting, and sum the integrals over the generated sub-elements.

If both $\mathbb{U}$ and $\mathbb{V}$ consist of polynomial basis functions, the integrals $\int u_i v_j$ over the sub-elements can be computed exactly. For more general choices, such as $\mathbb{V}$ representing a blob basis, a numerical quadrature scheme may be used.

In the simplest case, $\mathbb{V}^{(0)}$ represents a piecewise constant pixel or voxel basis

$$v_j(\mathbf{r}) = \begin{cases} 1 & \text{if } \mathbf{r} \in S(v_j) \\ 0 & \text{otherwise} \end{cases} \tag{23}$$

In this case, the components of B^{uv} simplify to

$$B^{uv}_{ij} = \sum_{k=1}^{N_i} \int_{S(e_k)\cap S(v_j)} u_i(\mathbf{r}) d^n r \tag{24}$$

where the sum runs over all elements $e_k \subset S(u_i)$. After subdivision of element e_k along the edges of pixel j into m sub-elements $\tilde{e}_{kl},\ l = 1..m$, excluding sub-elements outside voxel j,

$$B^{uv}_{ij} = \sum_{k=1}^{N_i} \sum_{l=1}^{m} \int_{S(\tilde{e}_{kl})} u_i(\mathbf{r}) d^n r \tag{25}$$

The integral of u_i over $\tilde{e}_{kl}$ can be computed in closed form when $\mathbb{U}$ is a polynomial basis.

If $\mathbb{V}$ consists of higher-order polynomial basis functions such as piecewise bilinear or trilinear functions, then the basis indices j refer to the vertex positions joining the voxels, rather than the voxels themselves, and each basis function has support over multiple pixels. For the computation of B^{uv}, the same element subdivision strategy can be employed, and a numerical quadrature rule is used for evaluating the integrals.

2.8. *Stiffness matrix calculations*

The calculation of the self and co-stiffness matrices in Equation 22 for the computation of the H^1 norm follows along the same lines as discussed for the mass matrix computation. Again, the computation of self-stiffness matrix D^{uu} is performed with standard finite element techniques, which include direct and exact solutions for polynomial basis functions.

Similarly, evaluation of self-stiffness matrix D^{vv} can make use of the same FEM machinery for certain basis function types, such as piecewise bi-polynomial and tri-polynomial basis functions. The basis functions must, however, be differentiable, which precludes piecewise constant bases. Where direct expressions of the integrals $\int \nabla v_i \cdot \nabla v_j$ over elements are not available, numerical schemes are employed. This includes blob bases, where the calculation of the intersection $W = S(v_i) \cap S(v_j)$ is identical to the process for B^{vv} as laid out in Algorithms 5 and 6.

Equally, for co-stiffness matrix D^{uv} the subdivision process is identical to B^{uv}, which means that the computation of the B and D matrices can be performed simultaneously on the subdivided elements, as indicated by Algorithms 2 and 3. The integrals $\int \nabla u_i \cdot \nabla v_j$ over the sub-elements are computed directly where available, or by numerical quadrature otherwise.

2.9. Direct mapping from pixel images

In addition to the mapping between bases $\mathbb{U}$ and $\mathbb{V}$, it may be useful to map from a high-resolution regular pixel or voxel image directly into $\mathbb{V}$. For example, a parameter distribution may be provided in the form of an image and must be expressed in $\mathbb{V}$ to form an initial condition for a reconstruction problem.

Let basis $\mathbb{W}$ be the piecewise constant pixel or voxel basis of dimension P representing the image. We require a mapping from $\mathbb{W}$ to $\mathbb{V}$. With the mapping tools defined earlier, we could map from $\mathbb{W}$ to a mesh basis $\mathbb{U}$ and then back from $\mathbb{U}$ to $\mathbb{V}$. However, the intermediate mapping to a mesh basis expansion would lead to a loss of accuracy. Instead, we provide a direct mapping from pixel values $\mathbf{X}^w$ in $\mathbb{W}$ to basis coefficients $\hat{\mathbf{X}}^v$ in $\mathbb{V}$:

$$\mathrm{B}^{vv}\hat{\mathbf{X}}^v = \mathrm{B}^{vw}\mathbf{X}^w \tag{26}$$

where

$$\mathrm{B}^{vw} \in \mathbb{R}^{M\times P} \text{ with } B_{ij}^{vw} = \int_{\Omega} v_i(\mathbf{r}) w_j(\mathbf{r}) d^n r = \int_{V_j} v_i(\mathbf{r}) d^n r \tag{27}$$

and V_j is the support area of pixel j. Similar to the mapping $\mathbb{U} \to \mathbb{V}$, the mapping $\mathbb{W} \to \mathbb{V}$ is performed by subdividing the support area of each v_i along the intersecting pixel edges and summing up the integrals over the contributing subregions.

2.10. Adaptive basis formulation

For efficient inverse problem solution, a basis with locally varying resolution is often desirable. The unstructured mesh basis $\mathbb{U}$ may be locally refined in accordance with an *a posteriori* error estimate. Likewise, the inverse basis $\mathbb{V}$ may be iteratively adapted to features that emerge during the reconstruction.

Starting from a choice of $\mathbb{V}$ with basis functions arranged in a regular grid, a natural choice for refinement is a quadtree or octree approach, where a pixel or voxel may be subdivided into four pixels or eight voxels of equal size. This type of adaptive refinement is well suited for the basis mapping techniques described in this paper, because the required element-splitting procedures are already in place, and the quad-and octree subdivisions of a voxel v_k in $\mathbb{V}$ affect the mapping matrices only locally, by replacing a column in B^{uv} with those of its descendants.

3. RESULTS

3.1. 2D basis mapping examples

Figure 5 shows an overview of the different basis representations and mappings between them in a chart. The original 'Lena' image $\mathbf{X}^w$ (a) consists of a piecewise constant 512×512 pixel image (basis $\mathbb{W}$). This is mapped into a regular bilinear 32×32 basis $\mathbf{X}^v$ (b) using the $\mathbb{W} \to \mathbb{V}$ mapping step and mapped back to the high-resolution pixel basis with mapping $\mathbb{V} \to \mathbb{W}$ (d). Using a mesh – in this example, a circular mesh consisting of 413 nodes with inhomogeneous node density (c) – we perform the mapping $\mathbb{V} \to \mathbb{U}$ (e), and map the result back to basis $\mathbb{W}$ with step $\mathbb{U} \to \mathbb{W}$ (f).

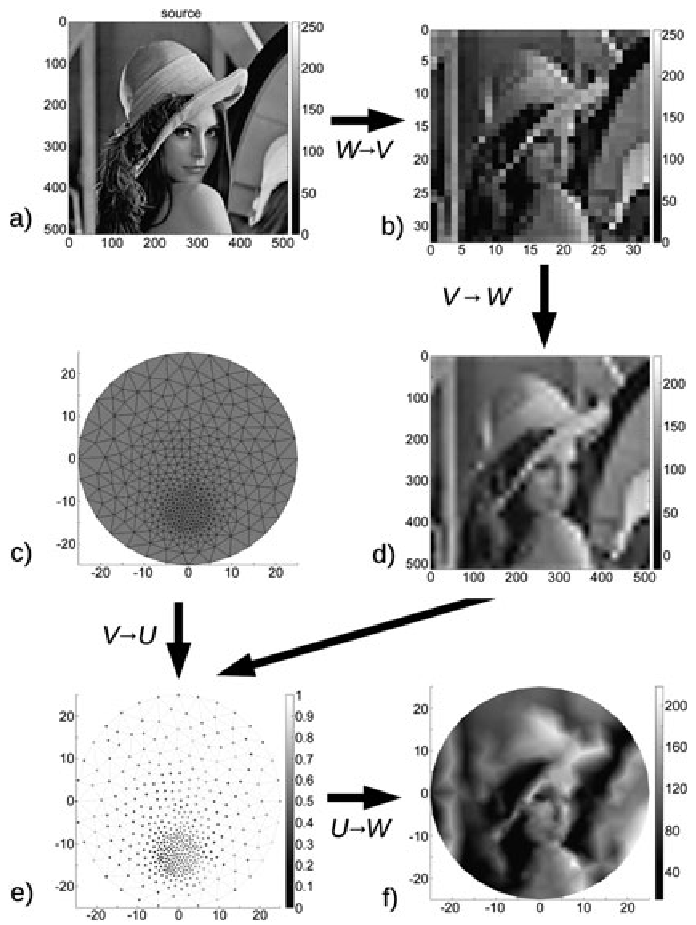

Figure 5. A chart of the mapping processes described in this paper.

Figure 6 shows a comparison of different choices for the inverse basis $\mathbb{V}$, mapped from a common source image. The top row shows, from left to right, the checkerboard source image represented in piecewise constant 512×512 pixel basis $\mathbb{W}$, the unstructured FEM mesh representing basis $\mathbb{U}$ consisting of 2779 nodes and 5396 triangles, and the result of the mapping $\mathbb{W} \rightarrow \mathbb{U}$. Row 2 shows the basis coefficients of inverse basis $\mathbb{V}$ after the mapping $\mathbb{U} \rightarrow \mathbb{V}$ for 28×28 piecewise constant pixel basis (left), 28×28 piecewise linear pixel basis (centre) and 28×28 spline blob basis (right). A resolution of 28 pixels per axis was chosen to avoid alignment of the basis cells with the 8×8 checkerboard pattern, which could lead to artificially low mapping errors. Row 3 shows the result of mapping the basis coefficients for each of the basis expansions back to the source basis representation ($\mathbb{V} \rightarrow \mathbb{W}$), and the bottom row shows the differences between the mapped images in row 3 and the source image.

Figure 7 shows the equivalent basis mapping steps for a different source image, using a 512×512 piecewise constant pixel representation of the 'Lena' test image. The same mesh for representing $\mathbb{U}$ and the same three choices for $\mathbb{V}$ were used as for Figure 6.

3.2. *3D basis mapping examples*

The results of a 3D mapping problem are shown in Figure 8 for a 3D checkerboard source image, and Figure 9 for a 3D phantom source image containing spherical and ellipsoidal inclusions in a homogeneous background. In both cases, the source images (a) were represented in a $128 \times 128 \times 128$ piecewise constant voxel basis $\mathbb{W}$, the target domain was a cylinder, and the mesh representation for basis $\mathbb{U}$ consisted of 12,676 nodes and 55,296 elements (c).

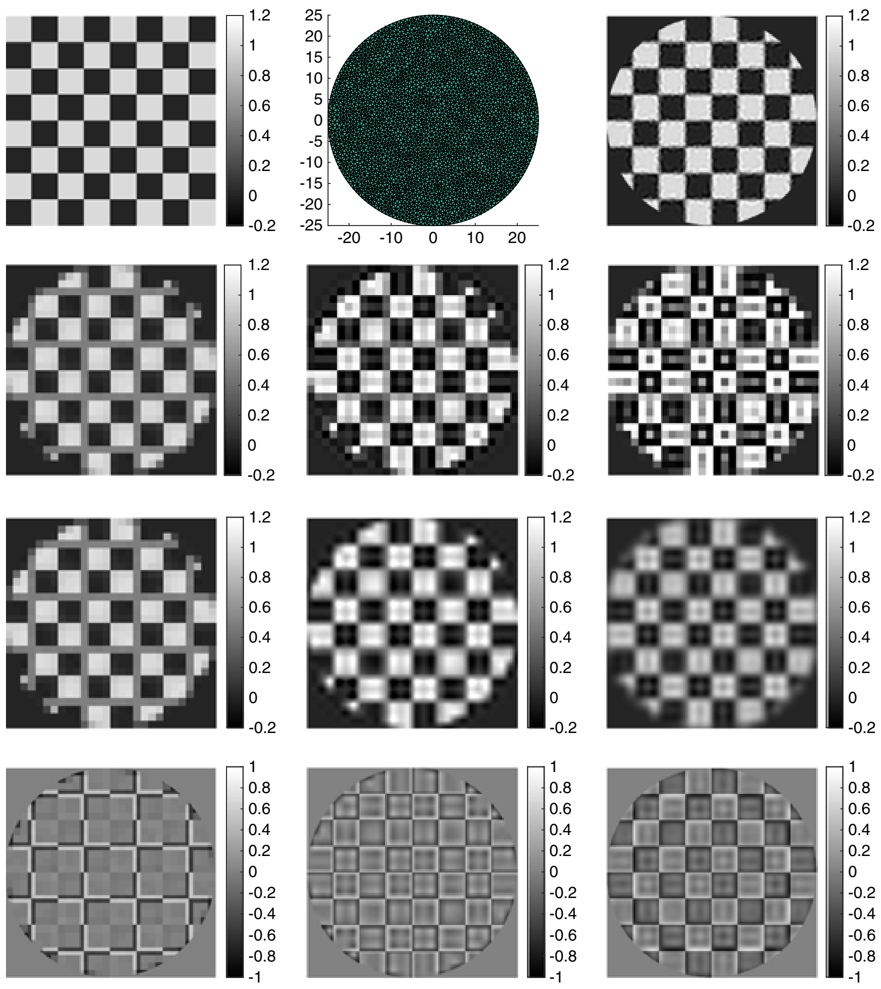

Figure 6. Comparison between inverse basis types. Top row, from left: 512×512 Source image, unstructured mesh, mapped source image $\mathbb{W} \to \mathbb{U}$. Second row: basis coefficients for 28×28 basis $\mathbb{V}$. From left: constant pixel basis, bilinear pixel basis, spline blob basis. Third row: basis representation after mapping back to high-resolution pixel basis, $\mathbb{V} \to \mathbb{W}$, for the three basis types. Bottom row: difference to source image.

The source image was first mapped into the mesh basis $\mathbb{U}$ (b) and then back into the original tri-linear voxel grid basis. Cross sections of the results for the SM (d) and LSM basis mapping algorithms (e) are shown. For both cases, the L^2 errors of the differences between the original basis coefficients (b) and the $\mathbb{V} \to \mathbb{U} \to \mathbb{V}$ mapped basis coefficients are smaller for the new LSM algorithm (e) than for the SM algorithm (d).

3.3. Comparison of mapping norms

To compare the effect of the choice of norm minimised by the mapping algorithm, we compare the mapping results using the L^2 norm (Equation 15) and the H^1 norm (Equation 21). Figure 10 shows a 128×128 constant pixel representation $\mathbb{W}$ of the Lena source image (a) and the mesh for representing basis $\mathbb{U}$ (b). Rows 2 to 4 show the results of basis mapping steps using the L^2 norm (left column) and the H^1 norm (right column), where row 2 contains the coefficients of a 32×32 spline blob basis after the mapping $\mathbb{W} \to \mathbb{V}$, row 3 contains the result of mapping to the mesh basis, $\mathbb{V} \to \mathbb{U}$ and row 4 shows the result of mapping back to the image basis ($\mathbb{U} \to \mathbb{W}$).

It can be seen that the use of the H^1 norm produces a smoother result in the mapped images, while the L^2 norm retains more high-frequency features of the original image.

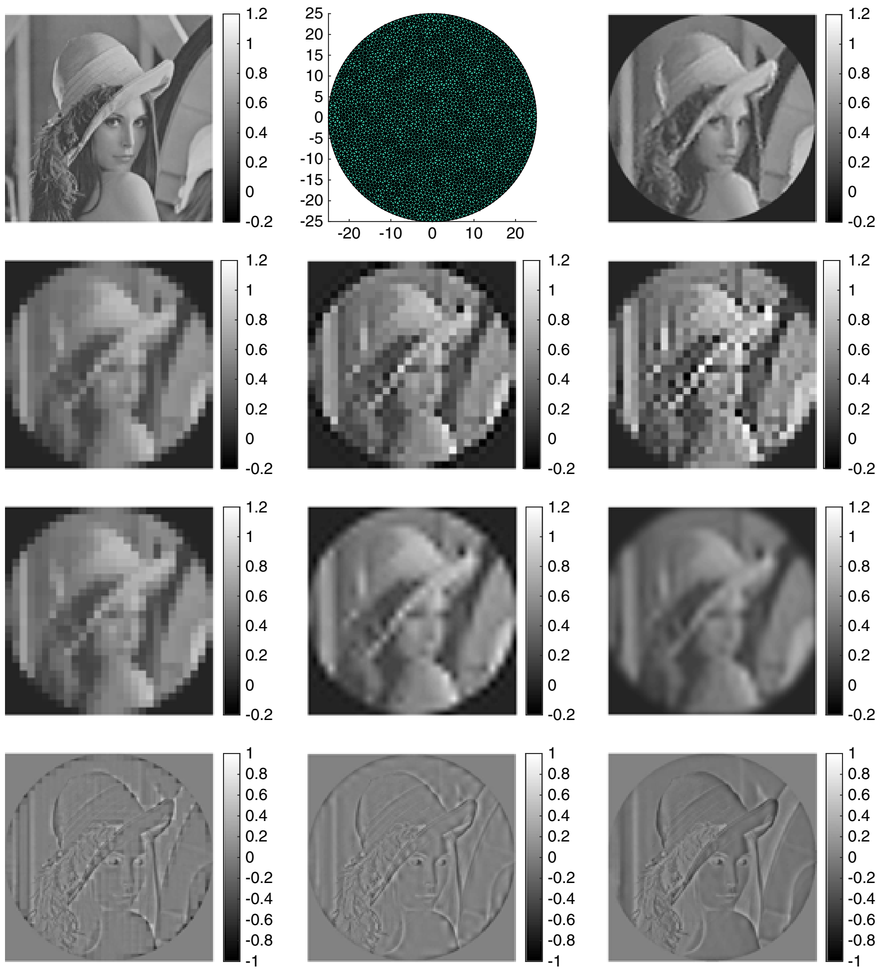

Figure 7. Comparison between inverse basis types. Top row, from left: 512×512 Source image, unstructured mesh, mapped source image $\mathbb{W} \to \mathbb{U}$. Second row: basis coefficients for 28×28 basis $\mathbb{V}$. From left: constant pixel basis, bilinear pixel basis, spline blob basis. Third row: Basis representation after mapping back to high resolution pixel basis, $\mathbb{V} \to \mathbb{W}$, for the three basis types. Bottom row: difference to source image.

3.4. *Mapping errors*

To demonstrate the advantage of the LSM mapping algorithm proposed in this paper compared with the SM method [4], we perform a mapping of a source image to different basis expansions and compare the mapping errors in the final mapped image with the source image. In particular, we compare the errors incurred by the mappings $\mathbb{V} \to \mathbb{U}$ and $\mathbb{U} \to \mathbb{V}$ for a range of choices of $\mathbb{V}$ in 2D and 3D examples.

Figure 11 shows the 512 × 512 pixel 2D checkerboard source image $\mathbf{X}^w$ (a) and an unstructured circular mesh, consisting of 682 nodes, used to define basis $\mathbb{U}$ (b).

For the first test, we perform the mapping $\mathbf{X}^w \overset{\mathbb{W}\to\mathbb{U}}{\longrightarrow} \mathbf{X}^u \overset{\mathbb{U}\to\mathbb{W}}{\longrightarrow} \tilde{\mathbf{X}}^w$. Figures 11(c) and (d) show $\tilde{\mathbf{X}}^w$ obtained with the SM and LSM methods, respectively. Figures 11(e) and (f) show the absolute differences $|(\tilde{\mathbf{X}}^w - \mathbf{X}^w)_i|$ for both cases. It can be seen that the new algorithm provides a better approximation of the internal structure of the source image, at the cost of some loss of homogeneity in the constant regions. The L^2 norms of the image difference for the SM and LSM mapping results are 122.7 and 103.8, respectively.

Figure 12 shows a corresponding comparison between the SM and LSM method, using the 'Lena' test image, and a higher-resolution mesh for basis $\mathbb{U}$. Again, it can be seen that the LSM algorithm provides an improved mapping result and generates smaller errors.

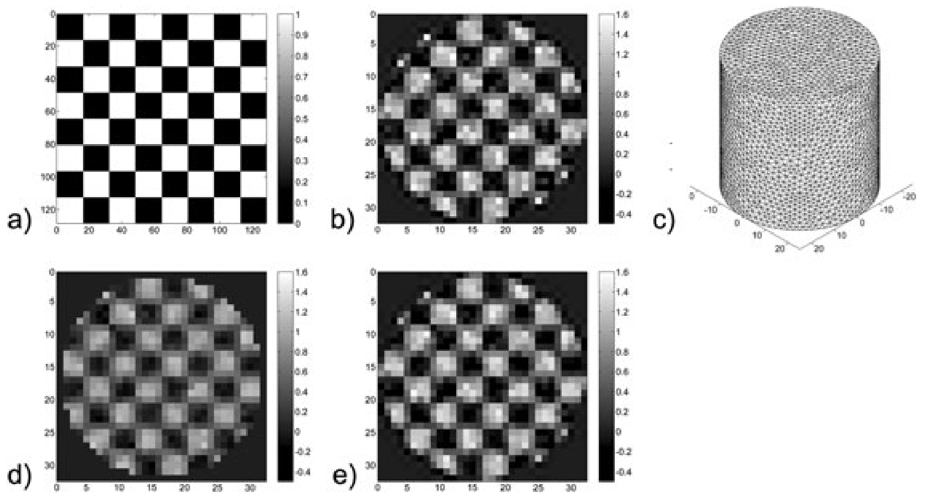

Figure 8. 3D mapping example. Cross section through 3D checkerboard source pattern (a). $\mathbb{W} \rightarrow \mathbb{V}$ Mapping of source image into $32 \times 32 \times 32$ trilinear voxel basis: cross section through basis coefficients (b). Cylindrical mesh for defining basis $\mathbb{U}$ (c). Result of mapping $\mathbb{V} \rightarrow \mathbb{U} \rightarrow \mathbb{V}$ using sampling mapping method (d) and least squares mapping method (e). L^2 errors of the difference images (d)–(c) and (e)–(c): 44.6 and 12.7, respectively.

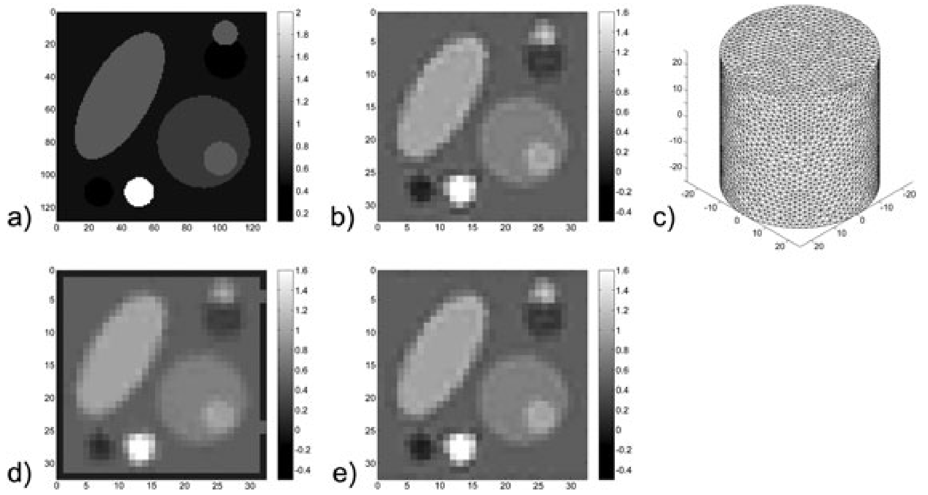

Figure 9. 3D mapping example. Cross section through 3D synthetic phantom source pattern (a). $\mathbb{W} \rightarrow \mathbb{V}$ Mapping of source image into $32 \times 32 \times 32$ trilinear voxel basis: cross section through basis coefficients (b). Cylindrical mesh for defining basis $\mathbb{U}$ (c). Result of mapping $\mathbb{V} \rightarrow \mathbb{U} \rightarrow \mathbb{V}$ using sampling mapping method (d) and least squares mapping method (e). L^2 errors of the difference images (d)–(c) and (e)–(c): 23.1 and 5.3, respectively.

Similar results are obtained for 3D mapping problems. We compare 3D mappings of the checkerboard and phantom source images, using both a trilinear voxel and spline blob basis $\mathbb{V}$, and perform the mapping by minimising both the L^2 and H^1 LSM error functionals. As the mapping from the image space ($\mathbb{W}$) to the inverse basis will incur a loss, we consider the result $\tilde{\mathbf{X}}^w$ of the mapping

$$\mathbf{X}^w \xrightarrow{\mathbb{W}\rightarrow\mathbb{V}} \mathbf{X}^v \xrightarrow{\mathbb{V}\rightarrow\mathbb{W}} \tilde{\mathbf{X}}^w \tag{28}$$

as the reference image and compare this with the result of mapping

$$\mathbf{X}^w \xrightarrow{\mathbb{W}\rightarrow\mathbb{V}} \mathbf{X}^v \xrightarrow{\mathbb{V}\rightarrow\mathbb{U}} \mathbf{X}^u \xrightarrow{\mathbb{U}\rightarrow\mathbb{V}} \tilde{\mathbf{X}}^v \xrightarrow{\mathbb{V}\rightarrow\mathbb{W}} \breve{\mathbf{X}}^w \tag{29}$$

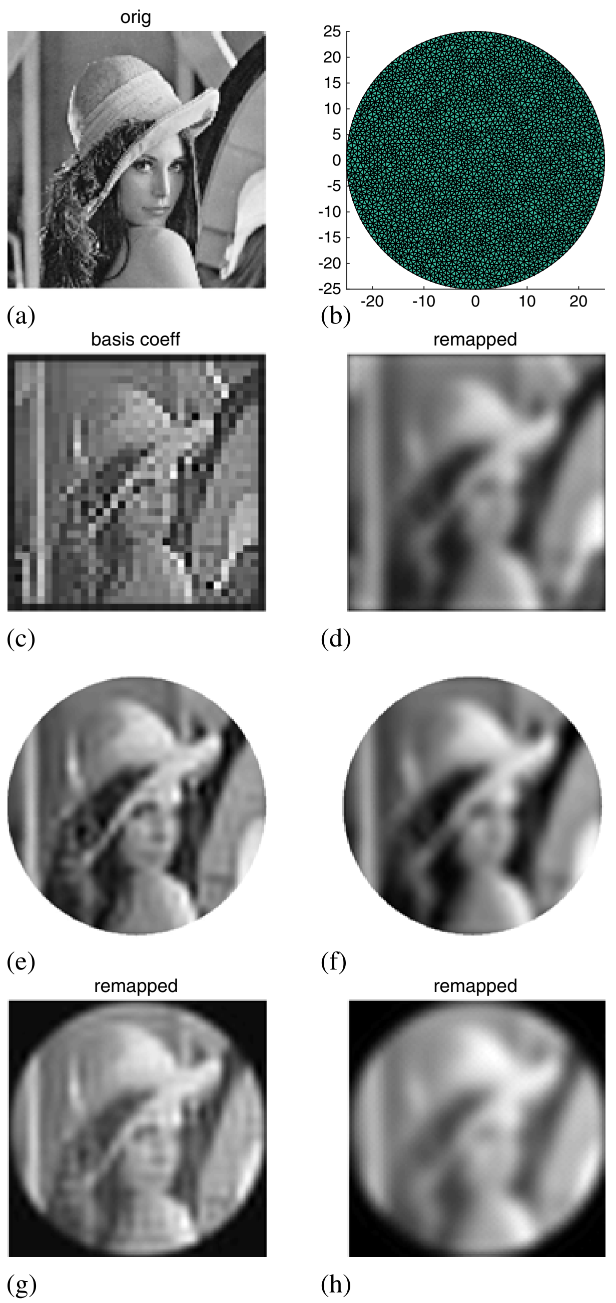

Figure 10. Comparison of the use of L^2 and H^1 norms in the least-squares minimisation of the mapping algorithm. 128×128 source image (a), FEM mesh representation of basis $\mathbb{U}$ (b). Result of mapping $\mathbb{W}\to\mathbb{V}$ for 32×32 spline blob basis $\mathbb{V}$, using the L^2 norm (c) and H^1 norm (d). Result of mapping $\mathbb{V}\to\mathbb{U}$ for both norms (e and f). Result of mapping back to the original image basis ($\mathbb{U}\to\mathbb{W}$) for both norms (g and h).

for different choices of $\mathbb{V}$ and $\mathbb{U}$, by comparing the norms of the image differences $||\tilde{\mathbf{X}}^w - \check{\mathbf{X}}^w||_D$, $D = \{L^2, H^1\}$. Figure 13 shows cross sections of the mappings Equation 28 and 29 for a 3D checkerboard pattern (a,b) and a 3D test phantom (c,d) using a trilinear voxel basis (a,c) and cubic spline blob basis (b,d). In each case, the first column shows the reference image $\tilde{\mathbf{X}}^w$, while columns 2 to 4 show $\check{\mathbf{X}}^w$ for SM, LSM(L^2) and LSM(H^1), respectively.

The error norms of the image differences between the $\mathbb{W}\to\mathbb{V}\to\mathbb{W}$ and $\mathbb{W}\to\mathbb{V}\to\mathbb{U}\to\mathbb{V}\to\mathbb{W}$ mappings are shown in Table I. Listed are the results for the SM algorithm and the LSM

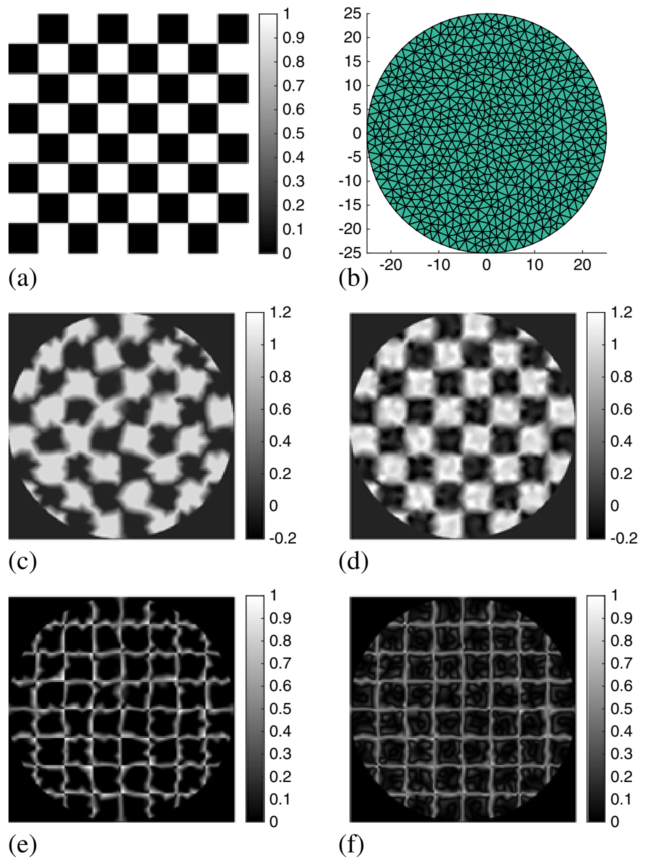

Figure 11. Comparison of $\mathbb{W} \rightarrow \mathbb{U} \rightarrow \mathbb{W}$ mapping errors between sampling mapping (SM) and least squares mapping (LSM) algorithm. (a) Source image in 512×512 piecewise constant pixel representation ($\mathbb{W}$). (b) 682-noded unstructured mesh representing basis $\mathbb{U}$. (c) Result of $\mathbb{W} \rightarrow \mathbb{U} \rightarrow \mathbb{W}$ mapping with sampling mapping algorithm, (d) result with LSM algorithm, (e) magnitude of image differences for SM algorithm and (f) for proposed LSM algorithm.

algorithm applying both L^2 and H^1 error functionals. The table lists both the L^2 and H^1 norms of the image differences. It can be seen that the LSM approach consistently achieves lower mapping errors than the SM method.

Finally, we explore the potential of reducing mapping errors by modifying basis $\mathbb{U}$ adaptively, in a 2D example. Figure 14 shows a 512×512 'Lena' source image (a) and an initial unstructured circular mesh with 682 nodes (b). Images (c) and (d) show the result of the $\mathbb{W} \rightarrow \mathbb{U} \rightarrow \mathbb{W}$ mapping for the SM and LSM algorithms, respectively. The L^2 norms for the image errors are 44.8 and 38.0. The pixel-wise image errors are then used to define a local node density target for re-meshing the circular domain with Delauny triangulation. The meshing parameters were set such that the total number of nodes was kept approximately constant. Images (e) and (f) show the resulting mesh structures, with node counts of 856 and 819 for the SM and LSM algorithms. The result of the $\mathbb{W} \rightarrow \mathbb{U}' \rightarrow \mathbb{W}$ mapping of the source image, where $\mathbb{U}'$ denotes the adaptive mesh basis, is shown in images (g) and (h), and the pointwise image errors in images (i) and (j). It can be seen that the proposed algorithm provides a superior mapping result, despite using a lower-dimensional mesh basis.

3.5. Adaptive basis results

As an example for an adaptive basis refinement, Figure 15 shows an iterative quadtree subdivision of an initial 16×16 pixel basis, defined over a 512×512 source image. A pixel is subdivided if the contrast range of the area it covers in the source image is larger than a threshold value. The first

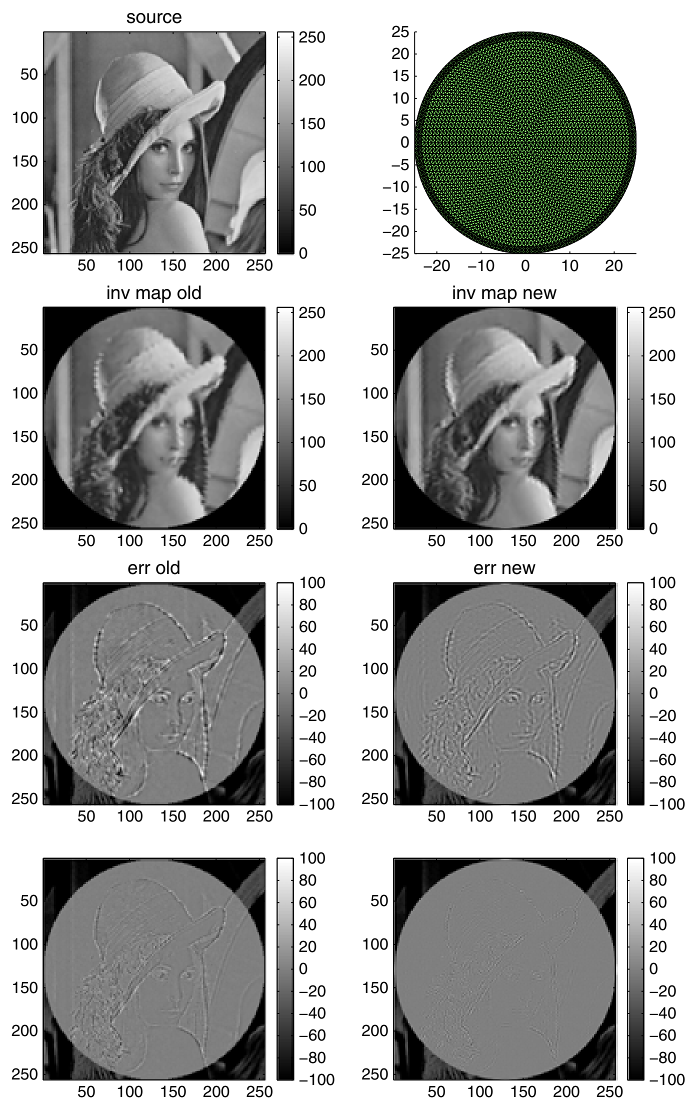

Figure 12. Basis mapping comparison: 256 × 256 'Lena' source image (top left) and circular mesh representing basis $\mathbb{U}$, consisting of 6840 triangular elements. Row 2: result of mapping source image into $\mathbb{U}$ and back into pixel basis, for sampling mapping algorithm (left) and new least squares mapping (right). Row 3: magnitude of differences between source and mapped images. Row 4: magnitude of differences when a higher-resolution mesh consisting of 27360 triangles was used.

column of images shows the structure of the quadtree mesh at each subdivision level. The second column shows the source image, mapped in the corresponding piecewise constant adaptive basis, and the last column shows the differences to the original image.

The motivation in adaptive basis definition is its computational efficiency combined with low error of the represented parameter. Table II shows a comparison of the adaptive meshes at the five levels shown in Figure 15 with a uniformly high-resolution mesh at the corresponding refinement level. It can be seen that the adaptive meshes reduce the basis dimension significantly while retaining a similar level of mapping error as the fully refined meshes.

3.6. Parallel computation

Computation of matrices B^{uv} and D^{uv} is computationally expensive due to the large number of element subdivisions and the need for a numerical quadrature over the sub-elements. To make the

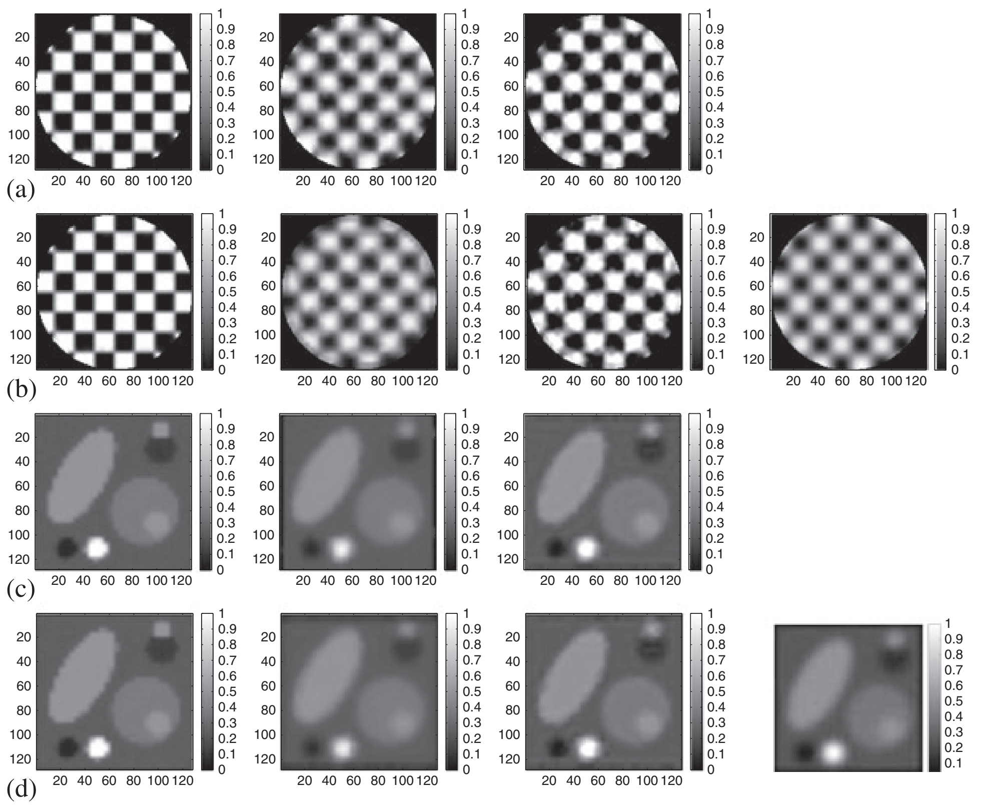

Figure 13. Comparison of basis mapping between SM and LSM L^2 and H^1 algorithms for a 3D target. The images show a cross section through a cylindrical domain containing a checkerboard pattern (a,b) and a phantom consisting of piecewise constant ellipsoidal inclusions (c,d). Rows (a) and (c) use a $32 \times 32 \times 32$ trilinear voxel basis $\mathbb{V}$, while rows (b) and (d) use a $32 \times 32 \times 32$ cubic spline blob basis $\mathbb{V}$. Columns from left to right: mapping $\mathbb{W} \to \mathbb{V} \to \mathbb{W}$, and mapping $\mathbb{W} \to \mathbb{V} \to \mathbb{U} \to \mathbb{V} \to \mathbb{W}$ for SM, LSM(L^2) and LSM(H^1), respectively.

Table I. Comparison of mapping errors between SM and LSM algorithm for 3D mapping examples in a cylindrical domain. For LSM, optimisations using both the L^2 and H^1 norms are considered. For all cases, the listed error values represent the L^2 and H^1 norms of the differences between the mapping $\mathbb{W} \to \mathbb{V} \to \mathbb{W}$ and $\mathbb{W} \to \mathbb{V} \to \mathbb{U} \to \mathbb{V} \to \mathbb{W}$.

		SM		LSM(L^2)		LSM(H^1)	
Target	Basis $\mathbb{V}$	$\|\|\cdot\|\|_{L^2}$	$\|\|\cdot\|\|_{H^1}$	$\|\|\cdot\|\|_{L^2}$	$\|\|\cdot\|\|_{H^1}$	$\|\|\cdot\|\|_{L^2}$	$\|\|\cdot\|\|_{H^1}$
checkerboard	$32 \times 32 \times 32$ trilinear	2.93	5.29	3.34	4.52		
checkerboard	$32 \times 32 \times 32$ spline	20.0	32.74	8.74	15.91	9.02	13.38
phantom	$32 \times 32 \times 32$ trilinear	12.42	13.28	10.89	13.11		
phantom	$32 \times 32 \times 32$ spline	6.31	9.16	4.32	6.74	4.37	5.60

SM, sampling mapping; LSM, least squares mapping.

problem tractable for larger meshes, we have devised a parallelised method for computation, using a threaded approach for shared-memory architectures. The mesh elements are distributed over the available threads, and element subdivision and integration are performed in parallel. Figure 16 shows the computation time for the mapping matrices for a mesh with 12,676 tetrahedral elements for basis $\mathbb{U}$, and a regular voxel grid of dimension $128 \times 128 \times 8$ using trilinear basis functions for basis $\mathbb{V}$, as a function of the inverse of the number of threads used. The relationship is approximately linear, demonstrating good scalability.

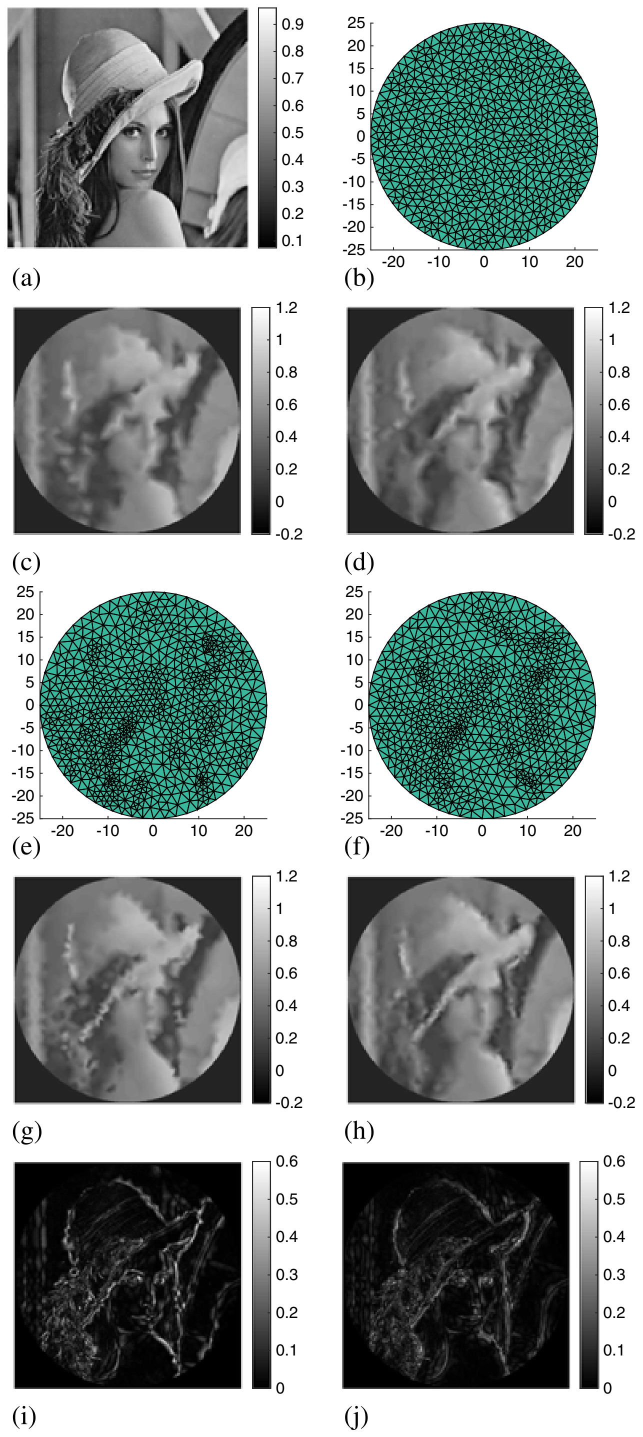

Figure 14. Effect of adaptive re-meshing on mapping errors. 512 × 512 source image (a), initial mesh (b), mapped images with sampling mapping and least squares mapping algorithm (c) and (d), adaptive meshes from image errors (e) and (f), mapped images using modified meshes (g) and (h), image errors after mapping into modified meshes (i) and (j).

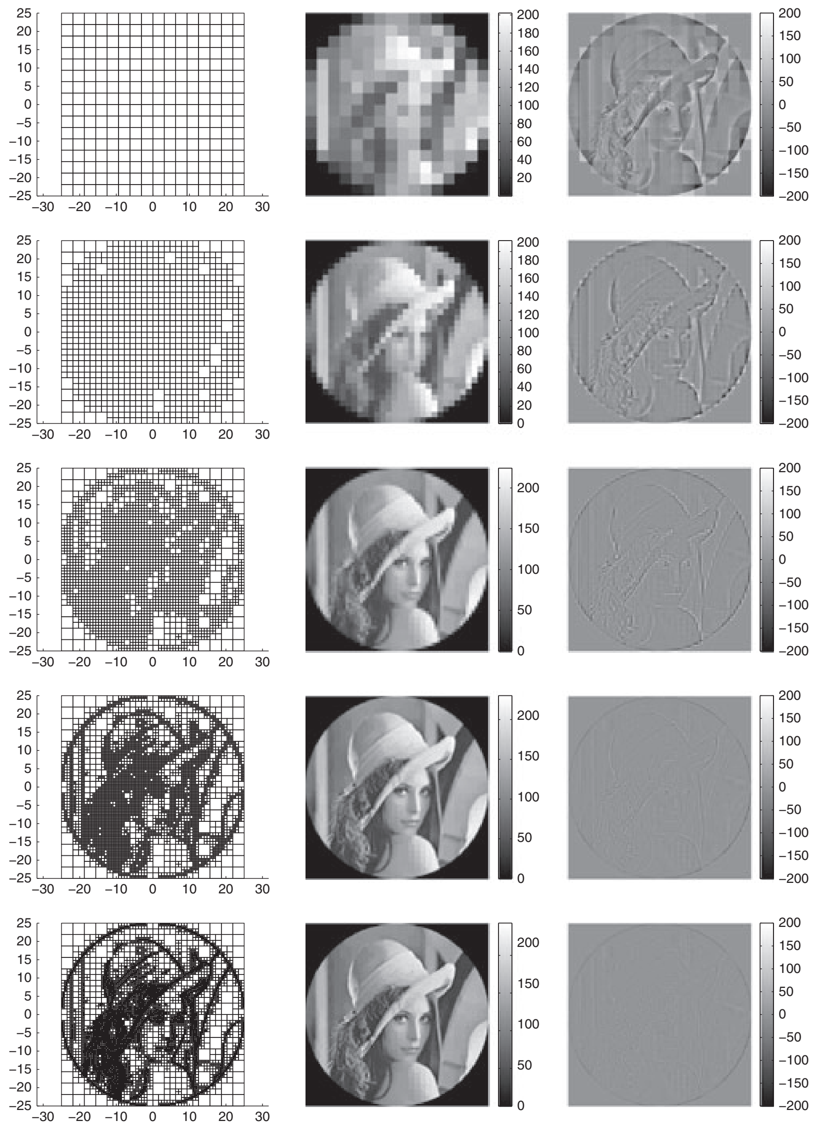

Figure 15. Adaptive refinement steps of a piecewise constant pixel basis. Left column: pixel distribution. Centre column: target image mapped into the basis using piecewise constant basis functions. Right column: pointwise squared differences between original and mapped images.

Table II. Comparison of adaptively refined with fully populated basis. M: basis dimension, ε: L^2 norm of difference of source and mapped image.

Res. level	M_{adapt}	M_{full}	$\varepsilon_{\text{adapt}}$	$\varepsilon_{\text{full}}$
1	256	256	6.4807	6.4807
2	928	1024	4.2030	4.2023
3	2887	4096	2.4591	2.4463
4	7813	16384	1.4331	1.3820
5	17080	65536	1.0715	0.9142

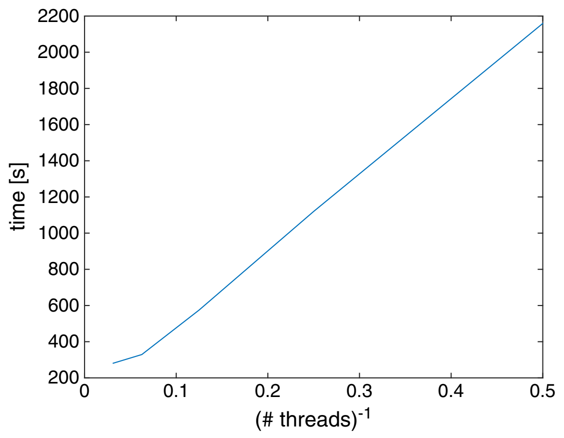

Figure 16. Runtimes for computation of basis matrices B^{uu}, B^{uv} and B^{uv} against inverse of thread count, using a multithreaded implementation.

4. CONCLUSIONS

We have presented in this paper a novel method for mapping a field variable $x(\mathbf{r})$, defined in a bounded domain Ω, from one finite-dimensional basis representation $x^{\mathbb{U}}(\mathbf{r})$ to a different representation $x^{\mathbb{V}}(\mathbf{r})$. Such mappings between basis expansions may arise in numerical analysis where different components of the computational pipeline operate on different representations of the same spatial variable. In this paper, we considered a problem typical for an inverse problem setting, where the problem parameters are expressed in different basis expansions for the forward and reconstruction problems. We assumed the forward model to operate on an unstructured piecewise polynomial finite element basis representation, while the solution of the inverse problem is represented by a regular grid of basis functions, be it a piecewise constant or polynomial pixel or voxel representation, or a blob basis representation with radially symmetric basis functions arranged in a regular grid.

Adaptive approaches to the numerical solution of inverse problems, such as those described in the introduction, that could potentially benefit from the basis mapping algorithms derived in this paper include electromagnetic imaging, where the forward model is given by Maxwell's equations [23, 24], diffuse optical tomography [4, 25] and fluorescence tomography [26, 27], electrical impedance tomography [28], as well as in adaptive inverse electrocardiography [29] and electroencephalography problems [30]. Applications can also be found in geophysics, using adaptive methods for seismic tomography [31], local earthquake tomography [32] and magnetic susceptibility mapping [33]. Furthermore, they may have relevance in other areas, besides inverse problems, such as in the fields of computational fluid dynamics [34] and in adaptive approaches to non-rigid image registration [35].

Our proposed method of basis mapping is based on an error-functional approach that minimises a norm of the difference between the two basis representations. We have compared this new LSM method with a previous approach (SM) that was based on sampling the source basis representation $\mathbb{U}$ at the nodal positions of the target basis $\mathbb{V}$. We have shown the new method presented here to achieve consistently superior mapping results in terms of the error functional residual than the SM method.

In addition, the LSM method offers a choice of norms for defining the error functional to be minimised, which can be used to emphasise certain image properties. In this paper, we have compared the L^2 and H^1 norms, and demonstrated that the latter allows to suppress high spatial frequencies in the mapped images.

The new mapping algorithm requires the computation of integrals of mixed products of basis functions, $\int u_i v_j$ or their derivatives, which is non-trivial because the support of u_i and v_j does not coincide. The calculation of these integrals is implemented by subdividing the elements of the unstructured mesh representing basis $\mathbb{U}$ along the boundaries of support of each v_j and performing the integrals by summing the integrals over the sub-elements. The subdivision adds a significant amount of computational cost, which can however be mitigated by parallel computation.

We have presented mapping results in both 2D and 3D problems, using unstructured meshes with locally varying node density and piecewise linear basis functions for basis $\mathbb{U}$, and regular voxel grids with piecewise constant, or piecewise bi-and tri-linear basis functions for basis $\mathbb{V}$. We have also demonstrated the use of a blob basis with radially symmetric basis functions. In this case, the limit of support of v_j is approximated by a polygonal or polyhedral surface to facilitate the element subdivision.

We have finally introduced the potential to apply more complex basis representations for $\mathbb{V}$, where a quadtree or octree representation was employed to construct a hierarchical basis expansion with locally varying resolution. This method could be used to perform adaptive refinement of the basis representation during reconstruction or to incorporate prior information about interal structure.

ACKNOWLEDGEMENTS

The research in this paper was supported by EPSRC Grants EP/J021318/1 and EP/K03829X/1.

REFERENCES

1. Borcea L. Electrical impedance tomography. *Inverse Problems* 2002; **18**(6):R99–R136.
2. Arridge SR, Schotland JC. Optical tomography: forward and inverse problems. *Inverse Problems* 2009; **25**(12):123010.
3. Symes WW. The seismic reflection inverse problem. *Inverse Problems* 2009; **25**(12):123008.
4. Schweiger M, Arridge SR. Image reconstruction in optical tomography using local basis functions. *Journal of Electronic Imaging* 2003; **12**(4):583–593.
5. Lewitt RM. Multidimensional digital image representations using generalized Kaiser-Bessel window functions. *Journal of the Optical Society of America A* 1990; **7**:1834–1846.
6. Matej S, Lewitt RM. Practical considerations of 3-D image reconstruction using spherically symmetric volume elements. *IEEE Transaction Medical Imaging* 1996; **15**(1):68–78.
7. Lewitt RM. Alternatives to voxels for image representation in iterative reconstruction algorithms. *Physics in Medicine and Biology* 1992; **37**(3):705–716.
8. Chesshire G, Henshaw WD. Composite overlapping meshes for the solution of partial differential equations. *Journal of Computational Physics* 1990; **90**(1):1–64.
9. Mayer UM, Popp A, Gerstenberger A, Wall WA. 3D Fluid-structure-contact interaction based on a combined XFEM FSI and dual mortar contact approach. *Computational Mechanics* 2010; **46**(1):53–67.
10. Jirasek M, Belytschko T. Computational resolution of strong discontinuities. *Fifth World Congress on Computational Mechanics*, Vienna, 2002; 7–12.
11. Baiges J, Codina R. The fixed-mesh ALE approach applied to solid mechanics and fluid-structure interaction problems. *International Journal for Numerical Methods in Engineering* 2009; **81**:1529–1557.
12. Glowinski R, Pan T, Hesla T, Joseph D, Periaux J. A ficticious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies Application to particulate flow. *Journal of Computational Physics* 2001; **169**:363–426.
13. Burman E, Hansbo P. Fictitious domain finite element methods using cut elements II. A stabilized Nitsche method. *Applied Numerical Mathematics* 2012; **62**:328–341.
14. Nitsche J. über ein variationsprinzip zur lösung von dirichlet-problemen bei verwendung von teilräumen, die keinen randbedingungen unterworfen sind. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg* 1971; **36**:9–15.
15. Dolbow J, Harari I. An efficient finite element method for embedded interface problems. *International Journal for Numerical Methods in Engineering* 2009; **78**:229–252.
16. Hansbo A, Hansbo P. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Computer Methods in Applied Mechanics and Engineering* 2002; **191**:5537–5552.
17. Bielser D, Maiwald VA, Gross MH. Interactive cuts through 3-dimensional soft tissue. *Computer Graphics Forum* 1999; **18**(3):31–38.
18. Steinemann D, Harders M, Gross M, Szekely G. Hybrid cutting of deformable solids. *Proceedings of IEEE Computer Society Conference on Virtual Reality (VR 06)*, Alexandria VA, USA, March 26-29, 2006; 35–42.
19. O'Brien JF, Hodgins JK. Animation of brittle fracture. *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques*, ACM Press/Addison-Wesley Publishing Co. Graphical, New York, NY, USA, 1999; 137–146.
20. Metzger MC, Gissler M, Asal M, Teschner M. Simultaneous cutting of coupled tetrahedral and triangulated meshes and its application in orbital reconstruction. *International Journal of CARS* 2009; **4**:409–416.

21. Sutherland I, Hodgman GW. Reentrant polygon clipping. *Communications of the ACM* 1974; **17**:32–42.
22. Zienkiewicz OC, Taylor RL. *The Finite Element Method* (4th edn). McGraw-Hill: London, 1987.
23. Wang ZJ, Przekwas AJ, Liu Y. A FV-TD electromagnetic solver using adaptive cartesian grids. *Computer Physics Communications* 2002; **148**:17–29.
24. Beilina L. Adaptive finite element method for a coefficient inverse problem for Maxwell's system. *Applicable Analysis* 2011; **90**(10):1461–1479.
25. Naser MA, Patterson MS, Wong JW. Algorithm for localized adaptive diffuse optical tomography and its applications in bioluminescence tomography. *Physics in Medicine and Biology* 2014; **59**(8):2089–2109.
26. Joshi EM, Sevick-Muraca A, Bangerth W. Adaptive finite element based tomography for fluorescence optical imaging in tissue. *Optics Express* 2004; **12**(22):5402–5417.
27. Zhou L, Yazici B. Discretization error analysis and adaptive meshing algorithms for fluorescence diffuse optical tomography in the presence of measurement noise. *IEEE Transactions on Image Processing* 2010; **20**(4):1094–1111.
28. Molinari M, Blott BH, Cox SJ, Daniell GJ. Optimal imaging with adaptive mesh refinement in electrical impedance tomography. *Physiological Measurement* 2002; **23**(1):121–128.
29. Johnson CR, MacLeod RS. Adaptive local regularization methods for the inverse ECG problem. *Progress in Biophysics and Molecular Biology* 1998; **69**(2–3):405–423.
30. Schimpf PH, Liu H, Ramon C, Haueisen J. Efficient electromagnetic source imaging with adaptive standardized LORETA/FOCUSS. *IEEE Transactions on Biomedical Engineering* 2005; **52**(5):901–908.
31. Zhang H, Thurber C. Adaptive mesh seismic tomography based on tetrahedral and Voronoi diagrams: application to Parkfield, California. *Journal of Geophysical Research* 2005; **110**(b04303).
32. Thurber C, Eberhart-Phillips D. Local earthquake tomography with flexible gridding. *Computers & Geosciences* 1999; **25**(7):809–818.
33. Davis K, Li Y. Fast solution of geophysical inversion using adaptive mesh, space-filling curves and wavelet compression. *Geophysical Journal International* 2011; **185**:157–166.
34. Darmofal DL, Fidkowski KJ. Review of output-based error estimation and mesh adaptation in computational fluid dynamics. *AIAA Journal* 2011; **49**(4):673–694.
35. Haber E, Heldmann S, Modersitzki J. Adaptive mesh refinement for nonparametric image registration. *SIAM Journal of Scientific Computing* 2008; **30**(6):3012–3027.