Journal of Scientific Computing, Vol. 24, No. 2, August 2005 (© 2005)
DOI: 10.1007/s10915-004-4614-x

The Optimisation of the Mesh in First-Order Systems Least-Squares Methods

Yves Tourigny[1]

Received November 30, 2003; accepted (in revised form) May 21, 2004

We describe an algorithm for optimising the mesh in the least-squares finite element discretisation of first-order systems of partial differential equations. The key feature of the method is that the optimisation process is based entirely on the solution of local PDE problems. We apply the algorithm to the Stokes equations for the flow of a viscous incompressible fluid, and to a convection diffusion equation where convection dominates.

KEY WORDS: Least-squares; finite element; adaptivity.

1. INTRODUCTION

There are two related aspects of adaptivity that dominate current research: one is *a posteriori* error estimation, whose ultimate aim is to compute a discrete approximation with an error that lies within a user-specified tolerance [1, 5]; the other concerns error *minimisation*, whose object is to obtain the best possible approximation for a given computational cost. Many adaptive schemes are based on determining the contribution of each computational cell to some *a posteriori* error estimate, and then seek to minimise the error by subdividing the cells with the largest contributions.

Mesh refinement is not the only mechanism available to enhance the accuracy of the approximation; mesh movement is a valuable alternative in which, for a fixed number of computational cells (elements, volumes), the vertices in the mesh are moved to new locations in the domain so as to reduce the error. Mesh movement, used in combination with local refinement, has a role to play in optimising the accuracy. This is

[1] School of Mathematics, University of Bristol, Bristol BS8 1TW, UK. E-mail: y.tourigny@bristol.ac.uk

particularly true in nonlinear problems where the areas of roughness of the solution (shocks, singularities, etc.) are not known beforehand [4].

The problem of optimising a mesh with very many degrees of freedom is often more difficult than that of solving the original PDE problem. For this reason, many mesh movement algorithms do not attempt to minimise the error directly; rather, they aim at smoothing or equidistributing some measure of the local error [3,4,14]. Nevertheless, there has recently been much interest in mesh optimisation based on minimising residuals. Roe and Nishikawa [16] explore this idea for some specific hyperbolic and elliptic problems arising in fluid dynamics, and provide a useful and lucid account of its potential benefits and of the difficulties of implementing it in practice.

In 1998, Tourigny and Hülsemann [17] described a novel approach to mesh optimisation for variational problems. The key feature of that algorithm is that the vertices in the mesh are updated by solving a succession of local partial differential problems. With this approach, the mesh can be improved—sometimes substantially—at a cost that is only a modest multiple of the cost of computing the global discrete solution on a given mesh. In the present paper, we show how the ideas of Tourigny and Hülsemann can be adapted to the case where the PDE is reformulated as a first-order system to which the principle of least-squares is applied.

In the next section, we consider a general boundary-value problem for a first-order system of partial differential equations, and discretise it by finite elements. The moving mesh algorithm is described in Sec. 3. The remaining sections are devoted to a few applications: we begin with a very simple one-dimensional example in Sec. 4. In Sec. 5, we discuss very briefly some of the implementation issues that arise in higher dimension. We then consider a convection-diffusion equation in Sec. 6, and the Stokes equations for the flow of a viscous incompressible fluid in Sec. 7. The paper ends with a summary of our conclusions.

2. A LEAST-SQUARES FINITE ELEMENT METHOD FOR A FIRST-ORDER SYSTEM

Let $\Omega \subset \mathbb{R}^d$ be a bounded open domain. Let $M, N \in \mathbb{N}$ and denote by $\mathbb{M}(M, N)$ the set of real $M \times N$ matrices. Given

- $\mathbf{f}:\Omega \to \mathbb{R}^M$,
- $\mathbf{g}:\partial\Omega \to \mathbb{R}^N$,
- $C,\ A_i:\Omega \to \mathbb{M}(M, N),\ 1 \leqslant i \leqslant d$,
- $B:\partial\Omega \to \mathbb{M}(M, N)$,

we seek a function $\mathbf{u}: \Omega \to \mathbb{R}^N$ such that

$$\mathcal{L}\mathbf{u} := \sum_{i=1}^{d} A_i(x) \frac{\partial \mathbf{u}}{\partial x_i} + C(x)\mathbf{u} = \mathbf{f}, \quad x \in \Omega \tag{2.1}$$

and

$$B\mathbf{u} = B\mathbf{g}, \quad x \in \partial\Omega\,. \tag{2.2}$$

For the least-squares formulation of this problem, we introduce a product function space

$$\mathbf{V} = \underbrace{H^1(\Omega) \times \cdots \times H^1(\Omega)}_{N \text{ times}}$$

and a functional $E: \mathbf{V} \to \mathbb{R}$ defined by

$$E(\mathbf{v}) = \frac{1}{2}\int_{\Omega} |\mathcal{L}\mathbf{v} - \mathbf{f}|^2\, dx + \frac{1}{2}\int_{\partial\Omega} |B(\mathbf{v} - \mathbf{g})|^2\, ds, \tag{2.3}$$

where $|\cdot|$ denotes the euclidean norm induced by the usual inner product $\langle \cdot, \cdot \rangle$ in $\mathbb{R}^M$. Then the problem of finding a solution of Eqs. (2.1) and (2.2) is replaced by that of finding $\mathbf{u} \in \mathbf{V}$ such that

$$E(\mathbf{u}) = \min_{\mathbf{v} \in \mathbf{V}} E(\mathbf{v})\,. \tag{2.4}$$

It should be emphasised that other least-squares formulations of the first-order system (2.1) are possible. For instance, one sometimes uses Sobolev norms to measure some components of the residual $\mathbf{f} - \mathcal{L}\mathbf{v}$. Another possibility would be to use some weighted inner product on $\mathbb{R}^M$ for the integrals. The results and techniques in this paper can be adapted to cater for many such variations. Our purpose in choosing this simple setting is to facilitate the exposition of the main ideas.

For the discretisation, we introduce a partition $\mathbb{T}_h$ of Ω into simplices κ. Here, h measures the coarseness of the partition. For an integer $r \geqslant 2$, we define the finite element space

$$V_h^r = \left\{ v \in C(\Omega) : \forall \kappa \in \mathbb{T}_h,\ v_{|_\kappa} \in P_r(\kappa) \right\},$$

where $C(\Omega)$ is the space of real-valued continuous functions defined on Ω and $P_r(\kappa)$ is the set of all polynomials of degree less than r defined on the simplex κ. Set

$$\mathbf{V}_h = V_h^{r_1} \times \cdots \times V_h^{r_N} \tag{2.5}$$

for some choice of r_n, $1 \leqslant n \leqslant N$. The finite element approximation $\mathbf{u}_h \in \mathbf{V}_h$ of $\mathbf{u}$ is then constructed by minimising E over $\mathbf{V}_h$, i.e.

$$E(\mathbf{u}_h) = \min_{\mathbf{v} \in \mathbf{V}_h} E(\mathbf{v}) . \tag{2.6}$$

We end the section with a brief discussion of the main theoretical issues associated with this least-squares method. We shall assume that the problem (2.1) and (2.2) has a unique solution.

Let us define a bilinear form $(\cdot, \cdot) : \mathbf{V} \times \mathbf{V} \to \mathbb{R}$ by

$$(\mathbf{v}, \mathbf{w}) = \frac{1}{2} \int_{\Omega} \langle \mathcal{L}\mathbf{v}, \mathcal{L}\mathbf{w} \rangle \, dx + \frac{1}{2} \int_{\partial\Omega} \langle B\mathbf{v}, B\mathbf{w} \rangle \, ds .$$

It follows from the uniqueness of the solution of (2.1) and (2.2) that the bilinear form is positive definite; hence it defines an inner product on $\mathbf{V}$ with corresponding norm

$$\| \cdot \| = \sqrt{(\cdot, \cdot)} .$$

We call this norm the *least-squares norm*. Note that

$$\forall \mathbf{v} \in \mathbf{V}, \quad E(\mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|^2 . \tag{2.7}$$

The least-squares finite element approximation $\mathbf{u}_h$ defined via Eq. (2.6) is therefore the best approximation in $\mathbf{V}_h$ in the sense of the least-squares norm:

$$\|\mathbf{u} - \mathbf{u}_h\| = \min_{\mathbf{v} \in \mathbf{V}_h} \|\mathbf{u} - \mathbf{v}\| . \tag{2.8}$$

Hence, the following *Galerkin orthogonality property* holds:

$$\forall \mathbf{v} \in \mathbf{V}_h, \quad (\mathbf{u} - \mathbf{u}_h, \mathbf{v}) = 0 . \tag{2.9}$$

One of the practical consequences of this property is that $\mathbf{u}_h$ can be computed by solving a linear system of equations that involves a symmetric, positive definite matrix. This is a very attractive feature of least-squares.

Given a PDE problem, the main difficulty in designing a least-squares method is to find a first-order system formulation with a least-squares norm that provides a useful measure of the error. For instance, when dealing with elliptic PDEs, one may be interested in obtaining approximations that are accurate in the sense of a product $H^1(\Omega)$ norm, say

$$\|\mathbf{v}\|_{H^1} := \left[\sum_{n=1}^{N} \|v_n\|^2_{H^1(\Omega)} \right]^{1/2} ,$$

where v_n is the nth component of $\mathbf{v}$. In this case, it is desirable to design a first-order system with the property that there exist constants c and C such that

$$\forall \mathbf{v} \in \mathbf{V}, \quad c \|\mathbf{v}\|_{H^1} \leqslant \|\mathbf{v}\| \leqslant C \|\mathbf{v}\|_{H^1} \tag{2.10}$$

with $c \approx C \approx 1$. Such an equivalence between the least-squares and target norms is often very hard to achieve; lack of it can result in poor convergence rates with respect to the target norm.

Indeed, as pointed out by Deang and Gunzburber [10], while there is a vast and growing literature on the mathematical analysis of finite element least-squares methods, most of the results available so far assume an idealised setting (e.g. a smooth solution, a convex polygonal domain) and concern asymptotic convergence rates that hold in the limit as $h \to 0$. Our particular focus in this paper is precisely the opposite! We seek effective methods for solving difficult problems with rough solutions, using relatively coarse meshes. Some of the practical difficulties associated with the least-squares methodology will be illustrated in Sections 6 and 7, where we consider specific examples.

3. THE MESH MOVEMENT ALGORITHM

A rough description of our approach is as follows. Given an initial mesh, we seek new positions for the vertices such that the least-squares norm of the error is reduced. One iteration of the algorithm consists of a sweep through the vertices. For each vertex, say x_j, we compute a "descent direction", say $-\boldsymbol{\Phi}(x_j) \in \mathbb{R}^d$, and move the vertex in that direction, holding every other vertex—and the mesh topology—fixed.

It is an elementary fact of Calculus that the descent direction for the vertex x_j that would produce the largest reduction of $E(\mathbf{u}_h)$ locally is

$$-\frac{\partial}{\partial x_j} E(\mathbf{u}_h) .$$

In order to compute this direction, however, one requires the discrete solution $\mathbf{u}_h$. Hence, if we were to choose this as the descent direction, we would have to solve the global discrete problem (2.6) repeatedly as the position of the vertex x_j changes. Although this approach has been followed by some authors (see [11], for instance), it is not practical for anything but very coarse meshes.

The key features of our approach are:

- We choose as the descent direction

$$-\boldsymbol{\Phi}_j(x_j) = -\frac{\partial}{\partial x_j} E(\mathbf{u}^h),$$

  where $\mathbf{u}^h$ is an approximation of $\mathbf{u}$ obtained by setting up and solving a PDE problem that is *purely local*. Note that h appears as a superscript to distinguish $\mathbf{u}^h$ from the global discrete solution $\mathbf{u}_h$.
- The same local PDE problem is used to update the approximation $\mathbf{u}^h$ locally as the vertex is moved. Hence, *no separate global solution step is required.*
- Although in general

$$\mathbf{u}^h \neq \mathbf{u}_h$$

  the resulting algorithm has the *monotonicity property*, i.e. the least-squares functional is reduced as the vertices move.

We now turn to the details of defining the descent direction, and the recipe for updating the approximation as each vertex is moved. Let $\mathbb{T}_h = \{\kappa_i\}_{i=1}^I$ be the current mesh with vertices $\{x_j\}_{j=1}^J$. We denote by Ω_j the union of those cells that share the vertex x_j. We say that $\hat{x} \in \Omega_j$ is an *admissible position* for x_j if a partition $\hat{\mathbb{T}}_h = \{\hat{\kappa}_i\}_{i=1}^I$ may be obtained from $\mathbb{T}_h$ by substituting $\hat{x}$ for x_j(cf. Fig. 1). The corresponding finite element space will be denoted by $\hat{\mathbf{V}}_h$.

Let $\mathbf{u}^{h,\mathrm{old}}$ be some given element of $\mathbf{V}_h := \mathbf{V}_h(\mathbb{T}_h)$. The algorithm for updating the vertex x_j will depend on the particular choice of $\mathbf{u}^{h,\mathrm{old}}$, but

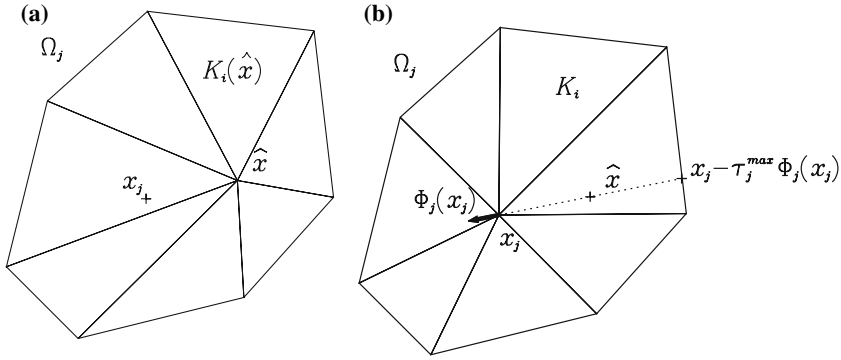

Fig. 1. (a) The old triangles κ_i in Ω_j and the descent path. (b) The triangles $\hat{\kappa}_i = \kappa_i(\hat{x})$ for the admissible location $\hat{x}$.

we shall postpone the precise definition. For the moment, one may simply think of $\mathbf{u}^{h,\mathrm{old}}$ as an approximation of $\mathbf{u}$ on the current mesh *which is not necessarily the global solution* $\mathbf{u}_h$ *of the discrete problem* (2.4). In order to emphasise this important distinction, the notation uses h as a subscript in the case of the global discrete solution $\mathbf{u}_h$, and as a superscript in the case of the approximation $\mathbf{u}^{h,\mathrm{old}}$.

Furthermore, we shall suppose that, for every admissible position $\hat{x}$, an approximation $\hat{\mathbf{u}}^h \in \hat{\mathbf{V}}_h$ is defined in such a way that

1. $\hat{\mathbf{u}}^h = \mathbf{u}^{h,\mathrm{old}}$ for $x \notin \overline{\Omega_j} := \Omega_j \cup \partial\Omega_j$.
2. For $\hat{x} = x_j$, $\hat{\mathbf{u}}^h = \mathbf{u}^{h,\mathrm{old}}$.

The first condition does not characterise $\hat{\mathbf{u}}^h$ fully; there are a few degrees of freedom associated with the elements contained in Ω_j that remain to be determined. Different strategies can be used to determine them, and we shall in due course consider two possible choices.

We shall seek a new admissible position $\hat{x} = x_j^{\mathrm{new}}$ for the jth vertex, with corresponding $\hat{\mathbf{u}}^h = \mathbf{u}^{h,\mathrm{new}} \in \hat{\mathbf{V}}_h = \mathbf{V}_h^{\mathrm{new}}$, such that

$$E(\mathbf{u}^{h,\mathrm{new}}) \leqslant E\left(\mathbf{u}^{h,\mathrm{old}}\right) \tag{3.1}$$

with equality if and only if $x_j^{\mathrm{new}} = x_j$. Suppose for the moment that x_j^{new} is given and such that

$$x_j(t) = t x_j^{\mathrm{new}} + (1-t)x_j$$

is admissible for all $0 \leqslant t \leqslant 1$. For each $t \in [0,1]$, we shall use $\mathbb{T}_h(t)$, $\kappa(t)$, $\mathbf{V}_h(t)$ and $\mathbf{u}^h(t)$ to denote $\hat{\mathbb{T}}_h$, $\hat{\kappa}$, $\hat{\mathbf{V}}_h$ and $\hat{\mathbf{u}}^h$ for $\hat{x} = x_j(t)$, respectively, although we shall often omit to indicate explicitly the dependence on t. As is well-known [2], in each cell $\kappa_i(t) \in \mathbb{T}_h(t)$, we have

$$\frac{d\mathbf{u}^h}{dt} = \dot{\mathbf{u}}^h - (\dot{x}_j \cdot \nabla \mathbf{u}^h)\,\phi_j. \tag{3.2}$$

In this expression, $\dot{\mathbf{u}}^h$ is the element of $\mathbf{V}_h(t)$ obtained by differentiating the coefficients in the representation of $\mathbf{u}^h(\cdot, t)$ in terms of the standard (Lagrange) finite element basis, $\phi_j(\cdot, t)$ denotes the piecewise linear function which equals unity at $x_j(t)$, and zero at all the other nodes, and the operator ∇ is applied to $\mathbf{u}^h$ componentwise. We shall make use of the following lemma, whose proof may be found in [17].

Lemma 3.1. For each $0 \leqslant t \leqslant 1$, let $F(\cdot, t) \in C^1(\kappa(t))$ and suppose that F is continuously differentiable in t. Then

$$\frac{d}{dt}\int_{\kappa(t)} F(x,t)\,dx = \dot{x}_j \cdot \int_{\partial\kappa(t)} F(x,t)\,\phi_j(x,t)\,n\,ds + \int_{\kappa(t)} \frac{\partial F}{\partial t}(x,t)\,dx\,.$$

Now, by construction, $\mathbf{u}^h(0) = \mathbf{u}^{h,\mathrm{old}}$. We then have

$$E(\mathbf{u}^{h,\mathrm{new}}) = E(\mathbf{u}^{h,\mathrm{old}}) + \int_0^1 \frac{d}{dt} E(\mathbf{u}^h(t))\,dt\,. \tag{3.3}$$

Our plan is to choose x_j^{new} in such a way that the integral on the right-hand side of the last equality is as *negative* as possible, so that the least-squares error is minimised. With this in mind, we denote, for each admissible $\hat{x}$ in Ω_j, the counterpart of $\phi_j(t)$ by $\hat{\phi}_j$, and introduce the d-dimensional vector

$$\begin{aligned}\boldsymbol{\Phi}_j(\hat{x}) = \sum_{\hat{\kappa}\subseteq\Omega_j} \Bigg\{ &\int_{\partial\hat{\kappa}} \left[\frac{1}{2}\left|\mathcal{L}\hat{\mathbf{u}}^h - \mathbf{f}\right|^2 \hat{\phi}_j\,n - \left\langle A^*\cdot n\left(\mathcal{L}\hat{\mathbf{u}}^h - \mathbf{f}\right), \nabla\hat{\mathbf{u}}^h\hat{\phi}_j\right\rangle_*\right] ds \\ &- \int_{\hat{\kappa}} \left\langle \mathcal{L}^*\left(\mathcal{L}\hat{\mathbf{u}}^h - \mathbf{f}\right), \nabla\hat{\mathbf{u}}^h\hat{\phi}_j\right\rangle_* dx\Bigg\} \\ &- \int_{\partial\Omega\cap\partial\Omega_j} \left\langle B^*B\left(\hat{\mathbf{u}}^h - \mathbf{g}\right), \nabla\hat{\mathbf{u}}^h\hat{\phi}_j\right\rangle_* ds\,.\end{aligned} \tag{3.4}$$

In this expression, $\langle\cdot,\cdot\rangle_*$ denotes the usual inner product in $\mathbb{R}^N$, $n \in \mathbb{R}^d$ is the unit vector normal to $\partial\kappa$ and pointing outward,

$$A^*\cdot n := \sum_{i=1}^d n_i A_i^*,$$

A_i^*, B^* and C^* are the $(N\times M)$ matrix transposes of A_i, B, and C, respectively, and $\mathcal{L}^*$ is the formal adjoint of the partial differential operator $\mathcal{L}$. More precisely, if $\mathbf{v}$ is a function with values in $\mathbb{R}^M$, then

$$\mathcal{L}^*\mathbf{v} = C^*\mathbf{v} - \sum_{i=1}^d \frac{\partial}{\partial x_i}\left(A_i^*\mathbf{v}\right)\,.$$

We show in Appendix A that

$$\begin{aligned}\frac{d}{dt}E\left(\mathbf{u}^h\right) = &\int_{\Omega_j}\left\langle \mathcal{L}\mathbf{u}^h - \mathbf{f}, \mathcal{L}\dot{\mathbf{u}}^h\right\rangle dx + \int_{\partial\Omega_j\cap\partial\Omega}\left\langle B\left(\mathbf{u}^h - \mathbf{g}\right), B\dot{\mathbf{u}}^h\right\rangle ds \\ &+ \dot{x}_j\cdot\boldsymbol{\Phi}_j\left(x_j(t)\right)\,.\end{aligned} \tag{3.5}$$

In order to deal with the $\dot{\mathbf{u}}^h$ term in this expression, we now need to specify how $\hat{\mathbf{u}}^h$ is defined in $\overline{\Omega_j}$.

One possibility is to choose $\hat{\mathbf{u}}^h$ in such a way that the coefficients in its expansion with respect to the Lagrange basis functions are *independent of* $\hat{x}$ (and so equal to those of $\mathbf{u}^{h,\text{old}}$). This scheme would define $\hat{\mathbf{u}}^h$ completely and, recalling that $\mathbf{u}^h$ is $\hat{\mathbf{u}}^h$ for $\hat{x} = x_j(t)$, would yield $\dot{\mathbf{u}}^h \equiv 0$, so that the terms involving $\dot{\mathbf{u}}^h$ in Eq. (3.5) drop out. This scheme, however, does not make optimal use of the degrees of freedom available in the definition of $\hat{\mathbf{u}}^h$.

A better way of defining $\hat{\mathbf{u}}^h$ is to set up a local version of the discrete PDE problem (2.9): find $\hat{\mathbf{u}}^h \in \hat{\mathbf{V}}_h$ such that $\hat{\mathbf{u}}^h = \mathbf{u}^{h,\text{old}}$ for $x \notin \overline{\Omega_j}$ and

$$\int_{\Omega_j} \left\langle \mathcal{L}\hat{\mathbf{u}}^h - \mathbf{f}, \mathcal{L}\mathbf{v} \right\rangle dx + \int_{\partial\Omega_j \cap \partial\Omega} \left\langle B\left(\hat{\mathbf{u}}^h - \mathbf{g}\right), B\mathbf{v} \right\rangle ds = 0 \tag{3.6}$$

for all $\mathbf{v} \in \hat{\mathbf{V}}_h$ with support in $\overline{\Omega_j}$.

We emphasise that, given $\mathbf{u}^{h,\text{old}}$, the computation of $\hat{\mathbf{u}}^h$ involves only the determination of very few unknowns. Furthermore, this method of construction leads to the desired simplification in Eq. (3.5). Indeed, $\mathbf{u}^h$ agrees with $\mathbf{u}^{h,\text{old}}$ outside $\overline{\Omega_j}$. It follows that $\dot{\mathbf{u}}^h$ has support in $\overline{\Omega_j}$. Since $\mathbf{u}^h$ solves the local problem (3.6), we obtain

$$\frac{d}{dt} E\left(\mathbf{u}^h\right) = \dot{x}_j \cdot \boldsymbol{\Phi}_j\left(x_j(t)\right), \tag{3.7}$$

and Eq. (3.3) becomes

$$E\left(\mathbf{u}^{h,\text{new}}\right) = E\left(\mathbf{u}^{h,\text{old}}\right) + \dot{x}_j \cdot \int_0^1 \boldsymbol{\Phi}_j\left(x_j(t)\right) dt\,. \tag{3.8}$$

The significance of the map $\boldsymbol{\Phi}_j$ is now apparent; Eq. (3.7) shows that

$$\boldsymbol{\Phi}_j(\hat{x}) = \frac{\partial}{\partial \hat{x}} E(\hat{\mathbf{u}}^h)\,.$$

When x_j is a vertex inside Ω, it is therefore natural to seek x_j^{new} along the direction of "steepest descent", i.e.

$$x_j^{\text{new}} = x_j - \tau\, \boldsymbol{\Phi}_j(x_j) \quad \text{for some } \tau \in [0, \alpha_j\, \tau_j^{\max}], \tag{3.9}$$

where $0 < \alpha_j < 1$. In this expression,

$$\tau_j^{\max} = \sup\left\{\tau > 0 : x_j - \tau' \boldsymbol{\Phi}_j(x_j) \text{ is admissible } \forall\, 0 \leqslant \tau' \leqslant \tau\right\}.$$

Given $\mathbf{u}^{h,\text{old}} \in \mathbf{V}_h$, the algorithm for updating x_j is as follows:

- If $\boldsymbol{\Phi}_j(x_j) = 0$, set $x_j^{\text{new}} = x_j$.
- Otherwise, let $\tau \in [0, \alpha_j \tau_j^{\max}]$ be the number nearest to zero such that

$$\boldsymbol{\Phi}_j(x_j) \cdot \int_0^{\tau} \boldsymbol{\Phi}_j \left(x_j - \tau' \boldsymbol{\Phi}_j(x_j) \right) d\tau'$$

  is maximised either locally or globally.

When x_j is a vertex on the boundary of Ω, then the vector $\boldsymbol{\Phi}_j\left(x_j\right)$ in Eq. (3.9) must be replaced by its projection on the boundary; otherwise, the algorithm is the same, given that the boundary condition (2.2) is applied "weakly" through the use of the boundary integral in the residual functional E.

A complete iteration consists of a sweep through the vertices in the mesh. We can choose to update the vertices simultaneously at the end of each sweep, or one by one as soon as each new location is computed. By analogy with the well-known relaxation methods for the solution of linear systems, we shall refer to the former choice as a "Jacobi sweep" and to the latter choice as a "Gauss–Seidel sweep". It follows from the foregoing calculation that the functional $E(\mathbf{u}^h)$ must *decrease* after a Gauss–Seidel sweep, unless the initial mesh is locally optimal. By contrast, the Jacobi approach does not generally enjoy this monotonicity property.

4. A SIMPLE ONE-DIMENSIONAL EXAMPLE

It is instructive to consider a very simple one-dimensional example, namely

$$\frac{du}{dx} = f, \quad x \in \Omega := (0, 1) \tag{4.1}$$

with the boundary condition $u(0) = u_0$. For the discretisation, we use piecewise linear polynomials on a partition with vertices

$$0 = x_0 < x_1 < \cdots < x_N = 1,$$

and impose the boundary conditon $u_h(0) = u_0$ strongly. Equation (2.9) then yields

$$\int_0^1 \frac{du_h}{dx} \frac{d\phi_j}{dx} \, dx = \int_0^1 f \frac{d\phi_j}{dx} \, dx, \quad 1 \leqslant j \leqslant N, \tag{4.2}$$

where

$$\phi_j(x) = \begin{cases} \frac{x - x_{j-1}}{x_j - x_{j-1}} & \text{if } x_{j-1} \leqslant x \leqslant x_j, \\ \frac{x_{j+1} - x}{x_{j+1} - x_j} & \text{if } x_j \leqslant x \leqslant x_{j+1}, \\ 0 & \text{otherwise.} \end{cases}$$

The solution of Eq. (4.1) is of course

$$u(x) = \int_0^x f(x')\, dx'.$$

By noting that

$$\frac{u(x_j) - u(x_{j-1})}{x_j - x_{j-1}} = \frac{1}{x_j - x_{j-1}} \int_{x_{j-1}}^{x_j} f(x)\, dx, \tag{4.3}$$

it is relatively easy to see that *the discrete solution u_h is exact at the vertices x_j*, i.e.

$$u_h(x_j) = u(x_j), \quad 0 \leqslant j \leqslant N.$$

It follows that, if $u^{h,\text{old}}$ is the global discrete solution on the old mesh, then, for every $x_{j-1} \leqslant \hat{x} \leqslant x_{j+1}$, the solution $\hat{u}^h$ of the local problem (3.6) is also exact at $\hat{x}$, and so equals the global discrete solution on the mesh with $\hat{x}$ as jth vertex. We emphasize that this is a very special case; generally, the approximation $\hat{u}^h$ obtained by solving the local problem will differ from u_h. Our purpose in considering this special case is to gain some insight into the asymptotic behaviour of the mesh movement algorithm.

Now, Eq. (3.4) gives

$$\begin{aligned} \boldsymbol{\Phi}_j(\hat{x}) = {} & \frac{1}{2} \left| \frac{d\hat{u}^h}{dx}(\hat{x}-) - f(\hat{x}-) \right|^2 - \frac{1}{2} \left| \frac{d\hat{u}^h}{dx}(\hat{x}+) - f(\hat{x}+) \right|^2 \\ & - \int_{x_{j-1}}^{\hat{x}} \left(\frac{d\hat{u}^h}{dx} - f \right) \frac{d\hat{u}^h}{dx} \frac{d\hat{\phi}_j}{dx}\, dx + \int_{\hat{x}}^{x_{j+1}} \left(\frac{d\hat{u}^h}{dx} - f \right) \frac{d\hat{u}^h}{dx} \frac{d\hat{\phi}_j}{dx}\, dx. \end{aligned}$$

As noted earlier, $\hat{u}^h$ equals u at the vertices, and so it follows from Eq. (4.3) that the last two integrals vanish. Hence, we obtain

$$\boldsymbol{\Phi}_j(\hat{x}) = \frac{1}{2} \left[\int_{x_{j-1}}^{\hat{x}} \frac{x - x_{j-1}}{\hat{x} - x_{j-1}} \frac{df}{dx}\, dx \right]^2 - \frac{1}{2} \left[\int_{\hat{x}}^{x_{j+1}} \frac{x_{j+1} - x}{x_{j+1} - \hat{x}} \frac{df}{dx}\, dx \right]^2. \tag{4.4}$$

Clearly, $\boldsymbol{\Phi}_j(x_{j-1}) < 0$ and $\boldsymbol{\Phi}_j(x_{j+1}) > 0$. So the new location x_j^{new} solves the equation

$$\boldsymbol{\Phi}_j(x_j^{\text{new}}) = 0\,.$$

It is interesting to consider the particular case where $f(x) = x$ in more detail. Then

$$\boldsymbol{\Phi}_j(\hat{x}) = -\frac{x_{j+1} - x_{j-1}}{8}\left(x_{j-1} - 2\hat{x} + x_{j+1}\right)\,.$$

Hence

$$x_j^{\text{new}} = \frac{x_{j-1} + x_{j+1}}{2}\,.$$

Let us suppose for simplicity that we sweep through the vertices in the lexicographic order. Then the vertices x_j^k of the mesh generated after k Gauss–Seidel sweeps satisfy the recurrence relation

$$x_j^{k+1} = \frac{1}{2}\left(x_{j-1}^{k+1} + x_{j+1}^k\right), \quad 1 \leqslant j \leqslant N-1, \;\; x_0^k = 0, \;\; x_N^k = 1\,. \tag{4.5}$$

The limiting mesh is uniform, i.e. $x_j^\infty = j/N$, and we have

$$\max_{0 \leqslant j \leqslant N} \left| x_j^k - x_j^\infty \right| = O\left(\varrho_h^k\right) \quad \text{as } k \to \infty,$$

where

$$\varrho_h = \max_{0 < \ell < N} \cos^2(\ell h \pi)\,.$$

The recurrence relation (4.5) is familiar; it arises when one applies the Gauss–Seidel relaxation method to solve the linear system resulting from the three-point finite difference discretisation of the one-dimensional Laplace equation. It is well-known that, although the first few iterations of the Gauss–Seidel method are very effective in reducing the error—particularly on coarse meshes—the sequence eventually "stalls" because the contraction factor ϱ_h is close to unity for h small [7].

The mesh movement algorithm that we have described exhibits a similar behaviour; unless the mesh is very coarse, there is no practical advantage in performing more than a few sweeps.

5. SOME IMPLEMENTATION DETAILS

In this section, we make a few remarks on the practical implementation of the Gauss–Seidel version of the algorithm, with emphasis on the two-dimensional case.

Clearly, the results depend on the way the vertices are *ordered*. For the numerical experiments in this paper, we order the vertices according to the value of $|\boldsymbol{\Phi}_j(x_j)|$ before each sweep.

The monotonicity property holds provided that the descent direction is computed *exactly*. In practice, some quadrature rule is required to evaluate $\boldsymbol{\Phi}_j(x_j)$. In the next sections, we use Gaussian quadrature rules with three points for the triangles and five points for the line integrals.

The parameter α_j in Eq. (3.9) and the sequential nature of the algorithm make it possible to control the geometrical quality of the simplices. For triangles in $\mathbb{R}^2$, we follow Bank and Smith [4] and use the quality measure

$$q(\kappa) = \frac{4}{\sqrt{3}} \frac{|\kappa|}{\sum_{i=1}^{3} \ell_i^2(\kappa)}, \tag{5.1}$$

where $|\kappa|$ denotes the area of κ, and $\ell_i(\kappa)$ is the length of the ith edge. Note that q takes values in the interval $[0, 1]$ and equals unity if κ is equilateral. Let $q_{\min} \in (0, 1)$ and suppose that

$$\min_{\kappa \in \mathbb{T}_h} q(\kappa) \geqslant q_{\min}. \tag{5.2}$$

Then, it is an easy matter to select α_j in such a way that this inequality holds after the vertex x_j has been moved.

In practical applications, mesh movement should be combined with adaptive algorithms that optimise the *topology* of the mesh; for instance, with mesh refinement algorithms such as those studied in [1,5]. In this paper, we illustrate the combination of mesh movement and "local edge-swapping" [13]. In brief, for every edge in a partition of triangles, we consider the quadrilateral formed by the triangles that share the edge. The edge is a diagonal connecting two of the vertices. If the quadrilateral is convex, then an alternative triangulation is obtained by making the edge lie along the other diagonal. In the context of least-squares, we can choose to "swap" the edge if it leads to a reduction of the least-squares functional. The approach we use in this paper is analogous to that described in [17].

Finally, let us address the question of the *computational cost* of the algorithm. If we assume that the quadrature rules are fixed, and that the

topology of the meshes is such that the number of vertices is always proportional to the number of elements, then the computational complexity of a single Gauss–Seidel sweep is

$$O(J) = O(h^{-d}) \quad \text{as } h \to 0\,.$$

On the other hand, the computational cost of solving the gobal discrete problem is

$$O(h^{-\gamma d}) \quad \text{as } h \to 0,$$

where γ depends on the particular solver. For the ideal solver—an optimally efficient multigrid method—we have $\gamma = 1$ and so, in that case, the cost of a sweep is proportional to the cost of solving the global discrete problem. However, in this paper, we have found it more convenient to use the preconditioned conjugate gradient method; for this choice, we have, when $d = 2$,

$$\gamma = 5/4\,.$$

Hence, in this case, the relative cost of performing a fixed number of mesh movement sweeps decreases as the meshes are refined.

6. A CONVECTION-DIFFUSION PROBLEM

Let

$$\Omega = [0, 1] \times [0, 1], \quad f \in L^2(\Omega), \quad a : \Omega \to \mathbb{R}^2 \quad \text{and} \quad \varepsilon > 0\,.$$

We consider the problem of finding a function $u : \Omega \to \mathbb{R}$ such that

$$-\varepsilon \Delta u + a \cdot \nabla u = f, \quad x \in \Omega \tag{6.1}$$

with the boundary condition

$$u = 0, \quad x \in \partial\Omega\,. \tag{6.2}$$

For small $\varepsilon > 0$, the solution can have a large gradient near the "outflow boundary"

$$\{x \in \partial\Omega : a \cdot n(x) > 0\}\,.$$

In order to obtain a method that is accurate uniformly in ε, we use the first-order system reformulation proposed by Fiard *et al.* [12]. First, we introduce functions $\alpha_1, \alpha_2 : \Omega \to \mathbb{R}^2$ such that

$$\left(\frac{\partial \alpha_1}{\partial x_1}, \frac{\partial \alpha_2}{\partial x_2}\right) = \frac{1}{\varepsilon} a\,.$$

Using the convenient notation

$$\alpha_+ = \frac{\alpha_1 + \alpha_2}{2} \quad \text{and} \quad \alpha_- = \frac{\alpha_1 - \alpha_2}{2}.$$

Equation (6.1) may be written in the form:

$$-\mathrm{e}^{\alpha_-} \frac{\partial}{\partial x_1}\left[\mathrm{e}^{-\alpha_1} \frac{\partial u}{\partial x_1}\right] - \mathrm{e}^{-\alpha_-} \frac{\partial}{\partial x_2}\left[\mathrm{e}^{-\alpha_2} \frac{\partial u}{\partial x_2}\right] = \frac{1}{\varepsilon} \mathrm{e}^{-\alpha_+} f\,. \tag{6.3}$$

Set

$$\mathbf{u} = \begin{pmatrix} u \\ \mathrm{e}^{-\alpha_1} \frac{\partial u}{\partial x_1} \\ \mathrm{e}^{-\alpha_2} \frac{\partial u}{\partial x_2} \end{pmatrix}.$$

Then Eq. (6.1) can be recast in the form (2.1) with

$$A_1 = \begin{pmatrix} -\mathrm{e}^{-\alpha_+} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & -\mathrm{e}^{\alpha_-} & 0 \\ 0 & 0 & -\mathrm{e}^{-\alpha_-} \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 0 & 0 \\ -\mathrm{e}^{-\alpha_+} & 0 & 0 \\ 0 & 0 & -\mathrm{e}^{-\alpha_-} \\ 0 & \mathrm{e}^{\alpha_-} & 0 \end{pmatrix},$$

$$C = \begin{pmatrix} 0 & \mathrm{e}^{\alpha_+} & 0 \\ 0 & 0 & \mathrm{e}^{-\alpha_-} \\ 0 & 0 & 0 \\ 0 & \mathrm{e}^{\alpha_-} \frac{\partial \alpha_1}{\partial x_2} & -\mathrm{e}^{-\alpha_-} \frac{\partial \alpha_2}{\partial x_1} \end{pmatrix} \quad \text{and} \quad \mathbf{f} = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{\varepsilon} \mathrm{e}^{-\alpha_+} f \\ 0 \end{pmatrix}.$$

The first two equations express the defining relationship between u and the other two unknowns; the third equation corresponds to Eq. (6.3), and the last equation essentially states that

$$\frac{\partial^2 u}{\partial x_2 \partial x_1} = \frac{\partial^2 u}{\partial x_1 \partial x_2}\,.$$

The boundary conditions are

$$u = \mathrm{e}^{-\alpha_1} \frac{\partial u}{\partial x_1} = 0 \quad \text{for } x_2 = 0,\ 1 \quad \text{and} \quad u = \mathrm{e}^{-\alpha_2} \frac{\partial u}{\partial x_2} = 0 \quad \text{for } x_1 = 0,\ 1\,.$$

Using the same scaling as for the PDE, this can be expressed in the form (2.2) with $\mathbf{g} = \mathbf{0}$ and

$$B = \frac{1}{\varepsilon} \mathrm{e}^{-\alpha_+} \begin{pmatrix} 1 & 0 & 0 \\ 0 & b_{22} & b_{23} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},$$

where

$$b_{22} = \begin{cases} 1 & \text{if } x_2 = 0 \text{ or } 1 \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad b_{23} = \begin{cases} 1 & \text{if } x_1 = 0 \text{ or } 1, \\ 0 & \text{otherwise.} \end{cases}$$

For the numerical experiment, we follow one of the examples in [12] and take

$$a = (-1, 0) \quad \text{and} \quad \varepsilon = \frac{1}{32},$$

choosing f so that

$$u = x_1 x_2 (1 - x_1)(1 - x_2) \mathrm{e}^{-\frac{x_1}{\varepsilon}}.$$

Thus, u has a boundary layer at $x_1 = 0$. The components of the corresponding solution $\mathbf{u}$ are plotted in Fig. 2.

For the least-squares method, we use a discrete space $\mathbf{V}_h$ such that every component of $\mathbf{v} \in \mathbf{V}_h$ is piecewise linear. The global discrete problem is solved by a preconditioned conjugate gradient method.

The meshes shown in Fig. 3 were obtained as follows: starting with a coarse uniform mesh of 64 triangles (here h is half the square root of the number of triangles in the mesh), we computed the global discrete solution, and then performed 10 sweeps of the mesh movement algorithm. The resulting mesh was then refined uniformly, and the procedure repeated on the refined mesh. For this case, no edge-swapping was performed. We used the constraint (5.2) with

$$q_{\min} = \frac{1}{2}.$$

The meshes obtained when mesh movement is used in combinition with edge swapping are qualitatively similar.

The value of the least-squares functional at each refinement level is given in Table I. It is apparent that

$$\|\mathbf{u} - \mathbf{u}_h\| = \sqrt{E(\mathbf{u}_h)} = O(h) \quad \text{as } h \to 0,$$

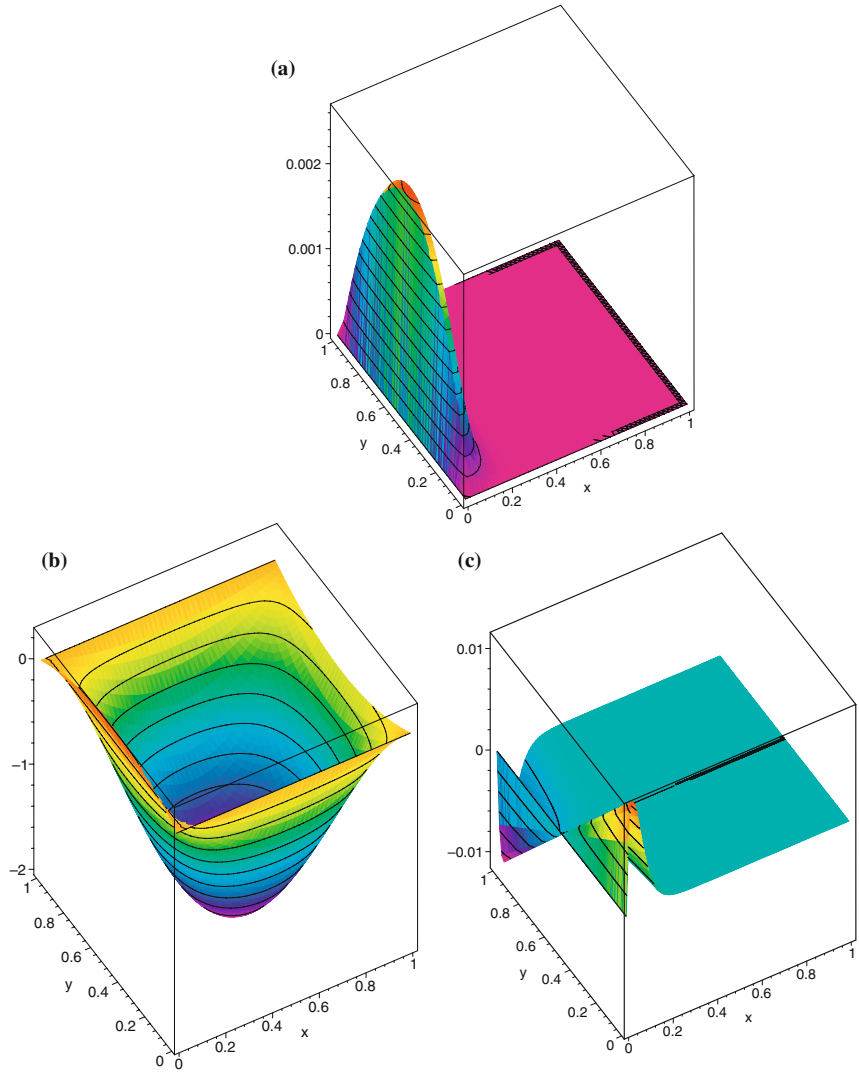

Fig. 2. The components (a) u_1, (b) u_2 and (c) u_3 of the solution of the first-order system for the convection-diffusion problem.

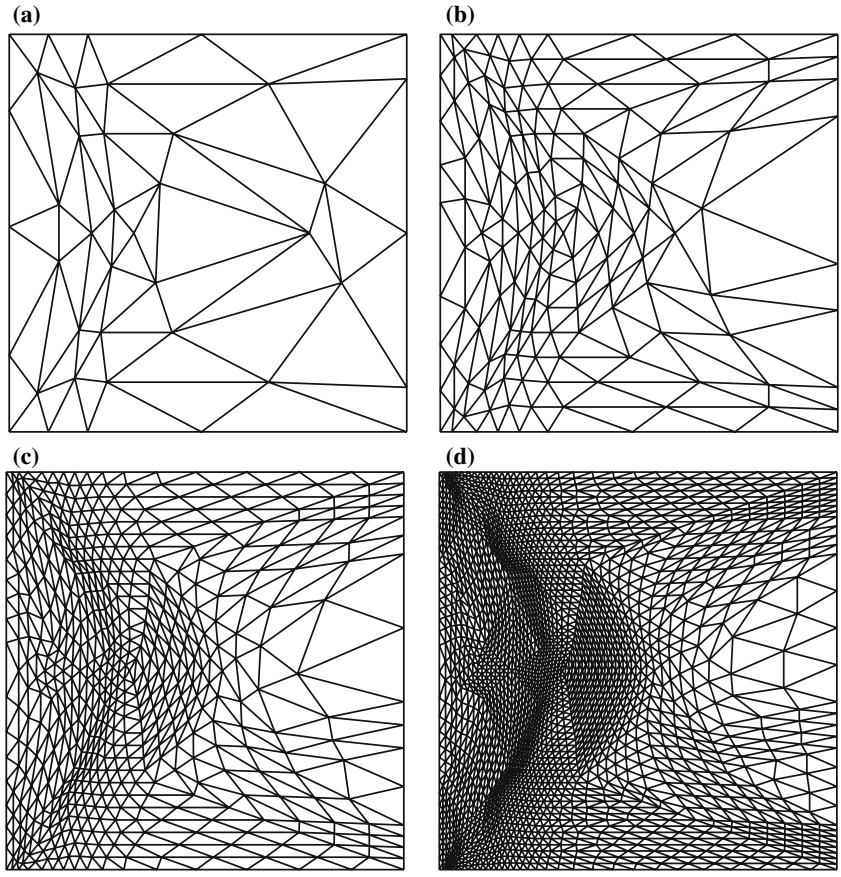

Fig. 3. Optimised meshes at successive refinement levels for the convection–diffusion problem: (a) $h = 1/4$, (b) $h = 1/8$, (c) $h = 1/16$, and (d) $h = 1/32$.

Table I. $E(\mathbf{u}_h)$ as h Decreases for the Convection-diffusion Problem

h	Uniform	Optimised	With swapping
1/4	2.96e-2	1.91e-2	1.92e-2
1/8	1.08e-2	6.03e-3	5.94e-3
1/16	3.17e-3	1.59e-3	1.44e-3
1/32	8.31e-4	4.00e-4	3.60e-4

whether or not the mesh is optimised. However, mesh movement reduces the least-squares functional by a factor of about two.

It would have been interesting to experiment with yet smaller values of the parameter ε. Fiard *et al.* [12] use a multigrid solver on meshes that are graded within the boundary layer and report results for $\varepsilon = 1/256$. But we were unable to calculate the discrete solution by a preconditioned conjugate gradient method on uniform meshes for $\varepsilon < 1/32$. This suggests that, despite the rescaling of the original equations, the global discrete least-squares problem becomes increasingly ill-posed as $\varepsilon \to 0$.

7. A PROBLEM OF STOKES FLOW

Let $\Omega = [-1, 1] \times [0, 1] \subseteq \mathbb{R}^2$. We consider the problem of finding three scalar functions u, v and p such that

$$-\Delta u + \frac{\partial p}{\partial x_1} = 0, \tag{7.1}$$

$$-\Delta v + \frac{\partial p}{\partial x_2} = 0, \tag{7.2}$$

$$\frac{\partial u}{\partial x_1} + \frac{\partial v}{\partial x_2} = 0, \tag{7.3}$$

in Ω, with the boundary conditions

$$u = u_0,\ v = v_0, \quad x \in \partial\Omega, \tag{7.4}$$

where u_0 and v_0 are given.

The unknowns u and v represent the horizontal and vertical components of the velocity field of a very viscous incompressible fluid, and p represents the pressure field. The pressure is only determined up to a constant by these equations; we select the particular solution such that

$$p = 0 \quad \text{at } x = (-1, 0)\,. \tag{7.5}$$

Various least-squares methods for the numerical solution of the Stokes equations have been studied in the literature. Deang and Gunzburger have carried out an interesting survey of this research area, with a particular focus on the practical performance of the methods [10]. Here, we describe the widely used velocity–vorticity–pressure formulation.

This approach consists of introducing the *vorticity*

$$\omega = \frac{\partial v}{\partial x_1} - \frac{\partial u}{\partial x_2} \tag{7.6}$$

as additional unknown. We then have

$$\frac{\partial \omega}{\partial x_2} + \frac{\partial p}{\partial x_1} = 0, \tag{7.7}$$

$$-\frac{\partial \omega}{\partial x_1} + \frac{\partial p}{\partial x_2} = 0, \tag{7.8}$$

$$\frac{\partial u}{\partial x_1} + \frac{\partial v}{\partial x_2} = 0. \tag{7.9}$$

So we set

$$\mathbf{u} = \begin{pmatrix} u \\ v \\ p \\ \omega \end{pmatrix}$$

and rewrite these equations in the form (2.1) with

$$A_1 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Hence, the first-order system consists of four equations for four unknowns. The boundary condition (7.4) can be expressed in the form (2.2) by defining the vector $\mathbf{g}$ and the matrix B as follows:

$$B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{g} = \begin{pmatrix} u_0 \\ v_0 \\ 0 \\ 0 \end{pmatrix}.$$

Let us now remark that the functional E, as defined in Eq. (2.3), uses the L^2 norm to measure every component of the residual. However, in the present case, the mathematical analysis of the least-squares method indicates that one should ideally measure the last two components of the residual in the H^1 norm. Following Bochev and Gunzburger [6], we can use the mesh-dependent inner product

$$\langle \mathbf{x}, \mathbf{y} \rangle = x_1 y_1 + x_2 y_2 + |\mathbb{T}_h| x_3 y_3 + |\mathbb{T}_h| x_4 y_4, \tag{7.10}$$

where

$$|\mathbb{T}_h| = \#\{\kappa \in \mathbb{T}_h\} = O(h^{-2}) \quad \text{as } h \to 0,$$

to mimic the use of the H^1 norm for the last two components. Note that the inner product depends only on the *number* of elements in the mesh, so that the calculations in Sec. 3 remain valid. In order to ensure that the boundary condition contributes to the total residual, we also use the mesh dependent weight in the boundary integral. Finally, for the discretisation, we use piecewise linears for every unknown and, again, solve the resulting system by a preconditioned conjugate gradient method.

For the numerical experiment, we take

$$u_0 := \frac{3}{2}\sqrt{r}\,[\cos(\theta/2) - \cos(3\theta/2)],$$
$$v_0 := \frac{3}{2}\sqrt{r}\,[3\sin(\theta/2) - \sin(3\theta/2)]$$

for $x \in \partial\Omega$, where $r = |x|$ and $\theta = \arctan(x_2/x_1)$ denote the familiar polar coordinates in $\mathbb{R}^2$. This somewhat artificial boundary condition has the advantage that an exact solution is given explicitly by $u \equiv u_0$, $v \equiv v_0$,

$$p = -\frac{6}{\sqrt{r}}\cos(\theta/2) \quad \text{and} \quad \omega = -\frac{6}{\sqrt{r}}\sin(\theta/2), \quad x \in \Omega\,.$$

This analytical solution is plotted in Fig. 4. We note that $\mathbf{u}$ has a singularity at the origin; it is easy to see that

$$p,\ \omega \in H^{\alpha}(\Omega), \quad u, v \in H^{\alpha+1}(\Omega) \quad \text{with } 0 \leqslant \alpha < 1\,.$$

Hence, we expect a poor rate of convergence on uniform meshes.

The meshes shown in Fig. 5 were obtained in a way similar to that in Sec. 6: starting from a coarse uniform mesh of 16 triangles, we computed the global discrete solution, and then performed 10 sweeps of the moving mesh algorithm, interleaved with edge-swapping sweeps. The procedure was then repeated on a (uniform) refinement of the resulting mesh. We controlled the mesh quality by insisting that, at each refinement level, the quality of the mesh remain within 60% of the quality of the initial mesh at that level. The optimised meshes are remarkably regular; there is only a very mild clustering of triangles at the origin.

The computed errors for uniform and optimised meshes are shown in Table II. Given a measure e_h of the error, we postulate that there exist C and γ such that

$$e_h \sim Ch^{\gamma} \quad \text{as } h \to 0\,.$$

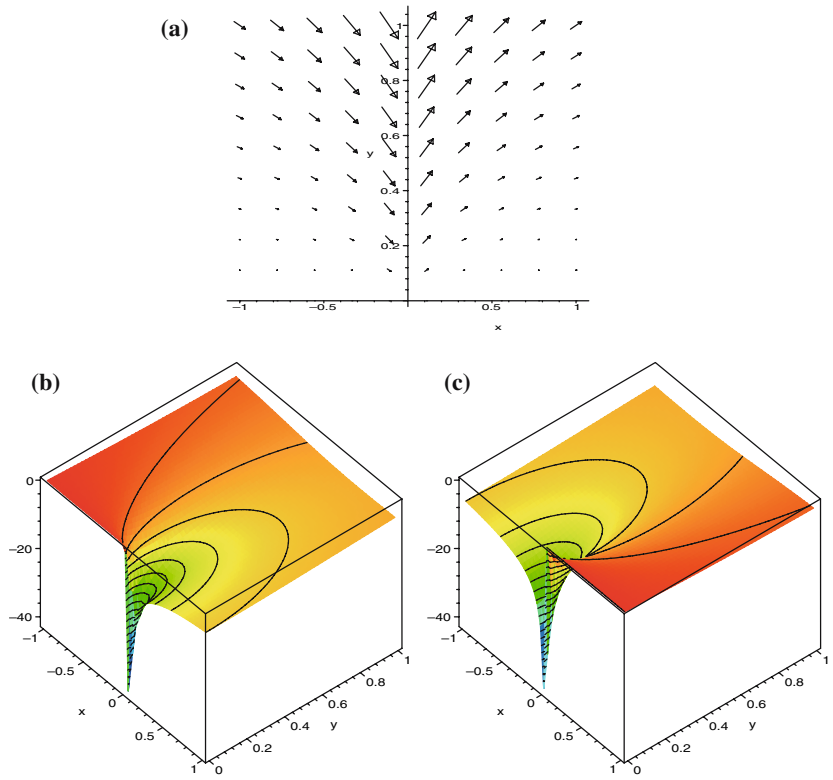

Fig. 4. The (a) velocity field, (b) pressure and (c) vorticity for the Stokes flow.

Then

$$\gamma_h := \frac{\ln \frac{e_{2h}}{e_h}}{\ln 2} \to \gamma \quad \text{as } h \to 0.$$

We can conclude from these results that the error is generally reduced by optimising the mesh, though the improvement is relatively minor. It is also apparent that these values of h are too large to reflect the asymptotic behaviour of the errors.

We have experimented with other least-squares methods for the Stokes problem. In particular, with the velocity-pressure-gradient of velocity method proposed by Cai *et al.* [8]. The analysis of the method shows that optimal convergence rates are obtained on convex polygonal domains with smooth solutions and, indeed, this is what we observed in practice. However, when the method was applied to the problem of computing

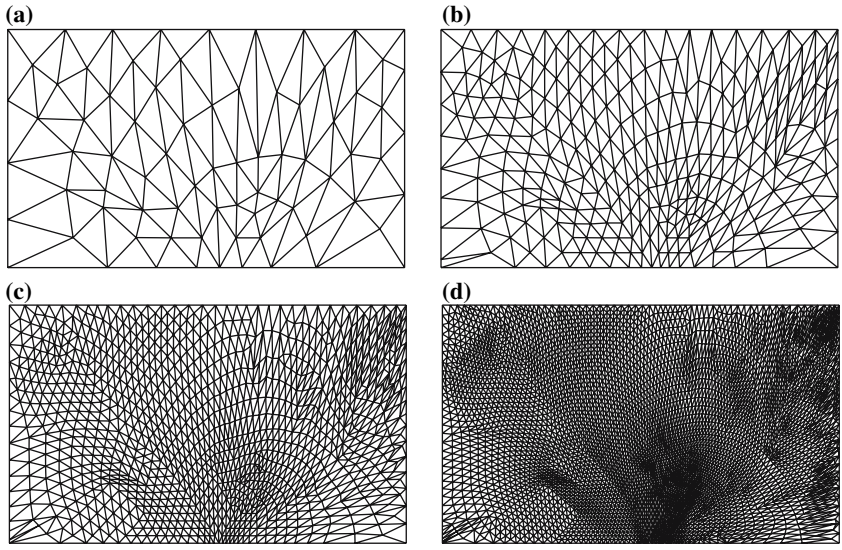

Fig. 5. Optimised meshes at successive refinement levels for the Stokes problem: (a) $h = 1/4$, (b) $h = 1/8$, (c) $h = 1/16$, and (d) $h = 1/32$.

the flow over a step—with a singular solution defined on a non-convex domain—we found that the approximation obtained was poor. In particular, because of the incompressibility condition, one expects approximate "mass conservation", i.e.

$$\int_{\partial\Omega} (u_h, v_h) \cdot n \, ds \to 0, \quad \text{as } h \to 0 .$$

We found that, on practical meshes, the quantity on the left-hand side converged very slowly. The issue of mass conservation in least-squares methods for the Stokes problem has been discussed by several authors [10].

8. CONCLUSION

In this paper, we have shown how the moving mesh algorithm developed by Tourigny and Hülsemann in [17] can be used in the context of least-squares. This algorithm is based on the solution of local PDE problems—making it relatively inexpensive to use. Hence, it provides a useful complement to the usual strategy of adaptive mesh refinement.

The least-squares methodology has many attractive features and, in principle, can be used to recast any given partial differential problem as

Table II. The Error as h Decreases for the Stokes Problem on Uniform and Optimised Meshes

h	1/2	1/4	1/8	1/16	1/32
$E(\mathbf{u}_h)$					
Uniform	1.28e+1	2.62e+1	4.65e+1	7.58e+1	1.20e+2
γ_h	–	−1.03	−0.83	−0.70	−0.66
Optimised	1.03e+1	1.99e+1	3.03e+1	4.49e+1	6.70e+1
γ_h	–	−0.95	−0.61	−0.57	−0.58
$\|u-u_h\|_{L^2(\Omega)}$					
Uniform	2.74e−1	1.80e−1	8.47e−2	3.75e−2	2.52e−2
γ_h	–	0.61	1.08	1.18	0.57
Optimised	2.53e−1	1.68e−1	8.21e−2	3.23e−2	1.85e−2
γ_h	–	0.59	1.03	1.34	0.80
$\|v-v_h\|_{L^2(\Omega)}$					
Uniform	3.70e−1	2.18e−1	1.10e−1	5.21e−2	2.80e−2
γ_h	–	0.76	0.99	1.08	0.90
Optimised	3.37e−1	1.82e−1	8.04e−2	3.58e−2	1.99e−2
γ_h	–	0.89	1.18	1.17	0.85
$\|p-p_h\|_{L^2(\Omega)}$					
Uniform	2.68e+0	2.13e+0	1.61e+0	1.24e+0	9.74e−1
γ_h	–	0.33	0.40	0.38	0.35
Optimised	2.56e+0	2.11e+0	1.42e+0	1.16e+0	8.64e−1
γ_h	–	0.28	0.57	0.29	0.43
$\|\omega-\omega_h\|_{L^2(\Omega)}$					
Uniform	6.60e+0	3.98e+0	2.47e+0	1.65e+0	1.18e+0
γ_h	–	0.73	0.69	0.58	0.48
Optimised	6.57e+0	3.59e+0	2.41e+0	1.65e+0	1.19e+0
γ_h	–	0.87	0.57	0.55	0.47

a minimisation problem for a computable functional (the residual). However, it is recognised that care must be taken when least-squares methods are used for problems that are not covered by the existing mathematical theory. In this paper, we have used the moving mesh algorithm to solve the Stokes equations with rough data, and a convection-diffusion equation where convection dominates. For these difficult applications, we have found that the approximation is improved when the algorithm is used, but that the improvement does not justify the additional cost incurred in optimising the mesh.

Nevertheless, we feel that the optimisation algorithm deserves further study. Future work will focus on combining the algorithm with adaptive

refinement strategies, and on comparing the results with those obtained by using mesh smoothing.

Appendix A: CALCULATION OF THE DESCENT DIRECTION

In this appendix, we derive Eq. (3.5).

In view of Eqs. (2.3) and (2.7), it is convenient to express $E(\mathbf{u}^h)$ as

$$E(\mathbf{u}^h) = \sum_{\kappa \in \mathbb{T}_h} E_\kappa(\mathbf{u}^h) + \frac{1}{2} \int_{\partial\Omega} |B(\mathbf{u}^h - \mathbf{g})|^2 ds,$$

where

$$E_\kappa(\mathbf{u}^h) = \frac{1}{2} \int_\kappa |\mathcal{L}\mathbf{u}^h - \mathbf{f}|^2 dx\,. \tag{A.1}$$

Hence

$$\frac{d}{dt} E(\mathbf{u}^h) = \sum_{\kappa \in \mathbb{T}_h} \frac{d}{dt} E_\kappa(\mathbf{u}^h) + \int_{\partial\Omega} \left\langle B(\mathbf{u}^h - \mathbf{g}),\, B \frac{d\mathbf{u}^h}{dt} \right\rangle ds\,. \tag{A.2}$$

First, we consider the sum on the right-hand side of this equation. The simplices outside Ω_j do not depend on t. Likewise, by assumption, $\mathbf{u}^h$ is independent of t for $x \notin \overline{\Omega_j}$. Therefore,

$$\sum_{\kappa \in \mathbb{T}_h} \frac{d}{dt} E_\kappa(\mathbf{u}^h) = \sum_{\kappa \subseteq \Omega_j} \frac{d}{dt} E_\kappa(\mathbf{u}^h)\,. \tag{A.3}$$

Using Lemma 3.1, we obtain from Eq. (A.1):

$$\frac{d}{dt} E_\kappa(\mathbf{u}^h) = \dot{x}_j \cdot \int_{\partial\kappa} \frac{1}{2} |\mathcal{L}\mathbf{u}^h - \mathbf{f}|^2 \phi_j\, n\, ds + \int_\kappa \left\langle \mathcal{L}\mathbf{u}^h - \mathbf{f},\, \mathcal{L} \frac{d\mathbf{u}^h}{dt} \right\rangle dx\,. \tag{A.4}$$

Now, Eq. (3.2) gives

$$\begin{aligned} \int_\kappa \left\langle \mathcal{L}\mathbf{u}^h - \mathbf{f},\, \mathcal{L} \frac{d\mathbf{u}^h}{dt} \right\rangle dx &= \int_\kappa \left\langle \mathcal{L}\mathbf{u}^h - \mathbf{f},\, \mathcal{L}\dot{\mathbf{u}}^h \right\rangle dx \\ &\quad - \int_\kappa \left\langle \mathcal{L}\mathbf{u}^h - \mathbf{f},\, \mathcal{L}(\dot{x}_j \cdot \nabla \mathbf{u}^h\, \phi_j) \right\rangle dx. \end{aligned} \tag{A.5}$$

We wish to take the $\dot{x}_j$ term outside the last integral. To this end, we use integration by parts:

$$\begin{aligned}\int_\kappa \left\langle \mathcal{L}\mathbf{u}^h - \mathbf{f}, \mathcal{L}\left(\dot{x}_j \cdot \nabla \mathbf{u}^h \phi_j\right)\right\rangle dx &= \int_{\partial k} \left\langle A^* \cdot n\left[\mathcal{L}\mathbf{u}^h - \mathbf{f}\right], \dot{x}_j \cdot \nabla \mathbf{u}^h \phi_j \right\rangle_* ds \\ &\quad + \int_\kappa \left\langle \mathcal{L}^*\left(\mathcal{L}\mathbf{u}^h - \mathbf{f}\right), \dot{x}_j \cdot \nabla \mathbf{u}^h \phi_j \right\rangle_* dx \\ &= \dot{x}_j \cdot \left\{ \int_{\partial k} \left\langle A^* \cdot n\left[\mathcal{L}\mathbf{u}^h - \mathbf{f}\right], \nabla \mathbf{u}^h \phi_j \right\rangle_* ds \right. \\ &\quad \left. + \int_\kappa \left\langle \mathcal{L}^*\left(\mathcal{L}\mathbf{u}^h - \mathbf{f}\right), \nabla \mathbf{u}^h \phi_j \right\rangle_* dx \right\}. \quad \text{(A.6)}\end{aligned}$$

Hence, we have shown that

$$\begin{aligned}\sum_{\kappa \in \mathbb{T}_h} \frac{d}{dt} E_\kappa\left(\mathbf{u}^h\right) &= \int_{\Omega_j} \left\langle \mathcal{L}\mathbf{u}^h - \mathbf{f}, \mathcal{L}\dot{\mathbf{u}}^h \right\rangle dx \\ &\quad + \dot{x}_j \cdot \sum_{\kappa \subseteq \Omega_j} \left\{ \int_{\partial \kappa} \left[\frac{1}{2} \left| \mathcal{L}\mathbf{u}^h - \mathbf{f} \right|^2 \phi_j \, n - \left\langle A^* \cdot n\left(\mathcal{L}\mathbf{u}^h - \mathbf{f}\right), \nabla \mathbf{u}^h \phi_j \right\rangle_* \right] ds \right. \\ &\quad \left. - \int_\kappa \left\langle \mathcal{L}^*\left(\mathcal{L}\mathbf{u}^h - \mathbf{f}\right), \nabla \mathbf{u}^h \phi_j \right\rangle_* dx \right\}. \quad \text{(A.7)}\end{aligned}$$

Turning next to the boundary integral in Eq. (A.2), we readily obtain

$$\begin{aligned}\int_{\partial\Omega} \left\langle B\left(\mathbf{u}^h - \mathbf{g}\right), B\frac{d\mathbf{u}^h}{dt} \right\rangle ds &= \int_{\partial\Omega \cap \partial\Omega_j} \left\langle B\left(\mathbf{u}^h - \mathbf{g}\right), B\frac{d\mathbf{u}^h}{dt} \right\rangle ds \\ &= \int_{\partial\Omega \cap \partial\Omega_j} \left\langle B\left(\mathbf{u}^h - \mathbf{g}\right), B\dot{\mathbf{u}}^h \right\rangle ds \\ &\quad - \dot{x}_j \cdot \int_{\partial\Omega \cap \partial\Omega_j} \left\langle B^* B\left(\mathbf{u}^h - \mathbf{g}\right), \nabla \mathbf{u}^h \phi_j \right\rangle_* ds\,.\end{aligned}$$

Reporting this in Eq. (A.2) and making use of (A.7), we deduce

$$\begin{aligned}\frac{d}{dt} E\left(\mathbf{u}^h\right) &= \int_{\Omega_j} \left\langle \mathcal{L}\mathbf{u}^h - \mathbf{f}, \mathcal{L}\dot{\mathbf{u}}^h \right\rangle dx + \int_{\partial\Omega_j \cap \partial\Omega} \left\langle B\left(\mathbf{u}^h - \mathbf{g}\right), B\dot{\mathbf{u}}^h \right\rangle ds \\ &\quad + \dot{x}_j \cdot \sum_{\kappa \subseteq \Omega_j} \left\{ \int_{\partial\kappa} \left[\frac{1}{2} \left| \mathcal{L}\mathbf{u}^h - \mathbf{f} \right|^2 \phi_j \, n - \left\langle A^* \cdot n\left(\mathcal{L}\mathbf{u}^h - \mathbf{f}\right), \nabla \mathbf{u}^h \phi_j \right\rangle_* \right] ds \right. \\ &\quad \left. - \int_\kappa \left\langle \mathcal{L}^*\left(\mathcal{L}\mathbf{u}^h - \mathbf{f}\right), \nabla \mathbf{u}^h \phi_j \right\rangle_* dx \right\} \\ &\quad - \dot{x}_j \cdot \int_{\partial\Omega \cap \partial\Omega_j} \left\langle B^* B\left(\mathbf{u}^h - \mathbf{g}\right), \nabla \mathbf{u}^h \phi_j \right\rangle_* ds\,.\end{aligned}$$

ACKNOWLEDGMENTS

I am indebted to Professor Mike Baines and Dr Paul Houston for many useful discussions of the ideas reported in this paper. Some of the contents were presented in the mini-symposium on adaptive methods at ICIAM 2003. I thank Professor Tao Tang for inviting me to participate, and the Royal Society for the Conference Grant that enabled me to attend. This research was also supported by the Engineering and Physical Research Council (United Kingdom) under Grant GR/L56817.

REFERENCES

1. Ainsworth, M., and Oden, T. (1997). A posteriori error estimation in finite element analysis. *Comput. Methods Appl. Mech. Engrg.* **142**, 1–88.
2. Baines, M. J. (1994). *Moving Finite Elements*, Clarendon Press, Oxford.
3. Baines, M. J. (2002). Moving meshes, conservation laws and least-squares equidistribution. *Int. J. Numer. Meth. Fluids* **40**, 3–19.
4. Bank, R. E., and Smith, R. K. (1997). Mesh smoothing using a posteriori error estimates. *SIAM J. Numer. Anal.* **34**, 979–997.
5. Becker, R., and Rannacher, R. (2001). An optimal control approach to a posteriori error estimation in finite element methods. `http://gaia.iwr.uni-heidelberg.de`
6. Bochev, P., and Gunzburger, M. D. (1994). Analysis of least-squares finite element methods for the Stokes equations. *Math. Comput.* **63**, pp. 479–506.
7. Briggs, W. L. (1987) *A Multigrid Tutorial*. SIAM, Philadelphia.
8. Cai, Z., Manteuffel, T. A., and McCormick, S. F. (1997). First-Order Systems Least-Squares for the Stokes equations, with application to linear elasticity. *SIAM J. Numer. Anal.* **34**, 1727–1741.
9. Cai, Z., Lee, C.-O., Manteuffel , T. A., and McCormick, S. F. (2000). First-Order Systems Least-Squares for the Stokes and linear elasticity equations: further results. *SIAM J. Sci. Comput.* **21**, 1728–1739.
10. Deang, J., and Gunzburger, (1998). Issues related to finite element methods for the Stokes equations. *SIAM J. Sci. Comput.* **20**, 878–906.
11. Delfour, M., Payre, G., and Zolésio, J. P. (1985). An optimal triangulation for second-order elliptic problems, Comput. Methods Appl. Mech. Engrg. **50**, pp. 231–261.
12. Fiard, J. M., Manteuffel, T. A., and McCormick, S. F. (1998). First-Order System Least-Squares (FOSLS) for convection-diffusion problems: numerical results. *SIAM J. Sci. Comput.* **19**, 1958–1979.
13. Lawson, C. L. (1977). Software for C^1 Interpolation, In *Mathematical Software III*, J. R. Rice (ed.), Academic Press, New York, 161–194.
14. Li, R., Liu, W., Ma, H., and Tang, T. (2002). Adaptive finite element approximation for distributed elliptic optimal control problems. *SIAM J. Control Optim.* **41**, 1321–1349.
15. Li, R., Tang, T., and Zhang, P. (2002). A moving mesh finite element algorithm for singular problems in two and three dimensions. *J. Comput. Phys.* **177**, 365–393.
16. Roe, P., and Nishikawa, H. (2002). Adaptive grid generation by minimizing residuals. *Int. J. Numer. Meth. Fluids* **40**, 121–136.
17. Tourigny, Y. and Hülsemann, F. (1998). A new moving mesh algorithm for the finite element solution of variational problems. *SIAM J. Numer. Anal.* **35**, 1416–1438.