

***Algorithm 866: IFISS, a Matlab toolbox for
modelling incompressible flow***

Elman, Howard C. and Ramage, Alison and Silvester,
David J.

2007

MIMS EPrint: **2005.45**

Manchester Institute for Mathematical Sciences
School of Mathematics

The University of Manchester

Reports available from: <http://eprints.maths.manchester.ac.uk/>

And by contacting: The MIMS Secretary
School of Mathematics
The University of Manchester
Manchester, M13 9PL, UK

ISSN 1749-9097

Algorithm 866: IFISS, A Matlab Toolbox for Modelling Incompressible Flow

HOWARD C. ELMAN

University of Maryland

ALISON RAMAGE

University of Strathclyde

and

DAVID J. SILVESTER

University of Manchester

IFISS is a graphical Matlab package for the interactive numerical study of incompressible flow problems. It includes algorithms for discretization by mixed finite element methods and a posteriori error estimation of the computed solutions. The package can also be used as a computational laboratory for experimenting with state-of-the-art preconditioned iterative solvers for the discrete linear equation systems that arise in incompressible flow modelling. A unique feature of the package is its comprehensive nature; for each problem addressed, it enables the study of both discretization and iterative solution algorithms as well as the interaction between the two and the resulting effect on overall efficiency.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*Linear systems (direct and iterative methods)*; G.1.8 [**Numerical Analysis**]: Partial Differential Equations—*Elliptic equations, finite element methods, iterative solution techniques, multigrid and multilevel methods*; G.4 [**Mathematical Software**]; J.2 [**Physical Sciences and Engineering**]

General Terms: Algorithms, Design, Documentation

Additional Key Words and Phrases: Matlab, finite elements, incompressible flow, iterative solvers, stabilization

The work of H. C. Elman was supported by the U.S. National Science Foundation under grant DMS0208015 and by the U.S. Department of Energy under grant DOEG0204ER25619.

Authors' addresses: H. C. Elman, Dept. of Computer Science, University of Maryland, College Park, MD 20742; email: elman@cs.umd.edu; A. Ramage, Dept. of Mathematics, University of Strathclyde, 26 Richmond Street, Glasgow G1 1XH, UK; email: A.Ramage@strath.ac.uk; D. J. Silvester, School of Mathematics, University of Manchester, Sackville Street, Manchester M60 1QD, UK; email: D.silvester@manchester.ac.uk.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or direct commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from the Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or permissions@acm.org. © 2007 ACM 0098-3500/2007/06-ART14 \$5.00 DOI 10.1145/1236463.1236469 <http://doi.acm.org/10.1145/1236463.1236469>

ACM Reference Format:

Elman, H. C., Ramage, A., and Silvester, D. J. 2007. Algorithm 866: IFISS, a matlab toolbox for modelling incompressible flow. *ACM Trans. Math. Softw.*, 33, 2, Article 14 (June 2007), 18 pages. DOI = 10.1145/1236463.1236469 <http://doi.acm.org/10.1145/1236463.1236469>

1. INTRODUCTION

This article describes the *Incompressible Flow Iterative Solution Software* (IFISS) package, a collection of over 290 `matlab` functions and `m`-files organised as a `matlab` toolbox. The toolbox can be used to compute numerical solutions of partial differential equations (PDEs) that are used to model steady incompressible fluid flow. It includes algorithms for discretization by finite element methods, fast iterative solution of the algebraic systems that arise from discretization, and *a posteriori* error analysis of the computed discrete solutions. The package was produced in conjunction with the monograph *Finite Elements and Fast Iterative Solvers* by Elman et al. [2005b], and it was used to perform the computational experiments described therein. It is structured as a stand-alone package for studying discretisation algorithms for PDEs and for exploring and developing algorithms in numerical linear and nonlinear algebra for solving the associated discrete systems. It can also be used as a pedagogical tool for studying these issues, or more elementary ones such as the properties of Krylov subspace iterative methods.

Four PDEs are treated: the *Poisson equation*, steady-state versions of the *convection-diffusion equation*, *Stokes equations*, and *Navier-Stokes equations*. The first two equations are ubiquitous in scientific computing, see for example Ockendon et al. [1999], Miller et al. [1995], Morton [1996] or Roos et al. [1996]. The latter two PDEs constitute the basis for computational modelling of the flow of an incompressible Newtonian fluid.

The first main feature of the package concerns problem specification and finite element discretization. For each of the equations listed, IFISS offers a choice of two-dimensional domains on which the problem can be posed, along with boundary conditions and other aspects of the problem, and a choice of finite element discretizations on a quadrilateral element mesh. The package allows the study of

- accuracy of finite element solutions,
- different choices of elements,
- a posteriori* error analysis.

In addition, special features associated with individual problems can be explored. These include the effects of boundary layers on solution quality for the convection-diffusion equation, and the effects of discrete inf-sup stability on accuracy for the Stokes and Navier-Stokes equations.

The second main feature of the package concerns iterative solution of the discrete algebraic systems, with emphasis on preconditioned Krylov subspace methods for linear systems of equations. The Krylov subspace methods are chosen to match each problem. For example, the discrete Poisson equation, which

has a symmetric positive definite coefficient matrix, can be treated by the *conjugate gradient method* (CG) [Hestenes and Stiefel 1952] whereas the discrete convection-diffusion and Navier-Stokes equations require a method such as the *generalized minimum residual method* (GMRES) [Saad and Schultz 1986], which is designed for nonsymmetric systems (see §3.1 for details). The key for fast solution lies in the choice of effective preconditioning strategies. The package offers a range of options, including algebraic methods such as incomplete LU factorizations, as well as more sophisticated and state-of-the-art multigrid methods designed to take advantage of the structure of the discrete linearized Navier-Stokes equations. In addition, there is a choice of iterative strategies, Picard iteration or Newton’s method, for solving the nonlinear algebraic systems arising from the latter problem.

A unique feature of the IFISS package is its comprehensive nature, where for each problem it addresses, it enables the study of both discretization and iterative solution algorithms as well as the interaction between the two and the resulting effect on solution cost. It is organized in a modular fashion, with separate program units (typically `matlab` functions) for individual tasks such as discretization, error analysis, iterative solution, and other aspects such as problem definition and visualization tools grouped in separate subdirectories. The package provides a convenient starting point for research projects that require the construction of new problem classes, discretizations or solution algorithms. A detailed description of how new problems can be incorporated into IFISS is given in Elman et al. [2005b].

The remainder of the article is organized as follows. Section 2 identifies the PDEs treated, describes the finite element methods used to approximate them, and outlines the strategies used for *a posteriori* error estimation. Section 3 describes the iterative solution algorithms provided for the linear systems that arise for each of the benchmark problems. Finally, Section 4 describes some of the design features of the package.

2. APPROXIMATION

A key feature of the IFISS package is the facility to construct various finite element solutions for a range of PDE problems and to carry out *a posteriori* error analysis of the computed discrete solutions. The theory of finite element methods is well established (see for example Braess [1997]; Brenner and Scott [1994]; Ciarlet [1978]; Elman et al. [2005b]; Gunzburger [1989]; and Johnson [1987]). An outline of the specific approximation methods implemented in IFISS is presented below for each PDE in turn.

2.1 The Poisson Equation

The problem considered here is

$$\begin{aligned} -\nabla^2 u &= f && \text{in } \Omega, \\ u &= g_D \text{ on } \partial\Omega_D, && \frac{\partial u}{\partial n} = 0 \text{ on } \partial\Omega_N, \end{aligned} \tag{1}$$

where Ω is a two-dimensional domain with boundary $\partial\Omega$ consisting of two nonoverlapping segments $\partial\Omega_D \cup \partial\Omega_N$. The function f is a given source (or load) function and g_D is given Dirichlet boundary data. In addition, \vec{n} is the outward-pointing normal to the boundary and $\frac{\partial u}{\partial n}$ denotes the directional derivative in the normal direction. The weak formulation of the PDE problem has a solution u in the Sobolev space $\mathcal{H}^1(\Omega)$ that satisfies the Dirichlet boundary condition together with the condition that

$$(\nabla u, \nabla v) = (f, v) \quad \forall v \in V := \{v \in \mathcal{H}^1(\Omega) | v = 0 \text{ on } \partial\Omega_D\},$$

where (\cdot, \cdot) denotes the L_2 inner product on Ω . Given some finite-dimensional subspace $V_h \subset V$, the associated discrete solution u_h satisfies $u_h = g_D$ on $\partial\Omega_D$ and

$$(\nabla u_h, \nabla v_h) = (f_h, v_h) \quad \forall v_h \in V_h, \quad (2)$$

where f_h is the $L_2(\Omega)$ orthogonal projection of f into V_h . For a standard finite element grid with n_u interior nodes and n_∂ Dirichlet boundary nodes, we choose a set of basis functions ϕ_i , $i = 1, \dots, n_u$ for V_h , and look for an approximate solution of the form

$$u_h(x, y) = \sum_{i=1}^{n_u} u_i \phi_i(x, y) + \sum_{i=n_u+1}^{n_u+n_\partial} u_i \phi_i(x, y), \quad (3)$$

where the boundary coefficients u_i , $i = n_u + 1, \dots, n_u + n_\partial$ are chosen to ensure that $\sum_{i=n_u+1}^{n_u+n_\partial} u_i \phi_i(x, y)$ interpolates the boundary data g_D on $\partial\Omega_D$. This leads to an $n_u \times n_u$ system of linear equations

$$\mathbf{A}\mathbf{u} = \mathbf{f}, \quad (4)$$

where the entries of the solution \mathbf{u} are the unknown coefficients u_i in (3).

Two standard finite element approximation methods are implemented in IFISS:

- Bilinear quadrilateral \mathbf{Q}_1* : The isoparametrically mapped square element with bilinear basis functions of the form $(ax + b)(cy + d)$. There are four unknowns per element.
- Biquadratic quadrilateral \mathbf{Q}_2* : The bilinearly mapped square element with biquadratic basis functions of the form $(ax^2 + bx + c)(dy^2 + ey + f)$. There are nine unknowns per element, corresponding to the terms $1, x, y, x^2, xy, y^2, xy^2, x^2y$ and x^2y^2 .

These finite element approximations typically have a discontinuous normal derivative across interelement boundaries. Consequently, it is convenient to define the *flux jump* across edge E adjoining elements T and S as

$$\left[\left[\frac{\partial v}{\partial n} \right] \right] := (\nabla v|_T - \nabla v|_S) \cdot \vec{n}_{E,T} = (\nabla v|_S - \nabla v|_T) \cdot \vec{n}_{E,S}, \quad (5)$$

where $\vec{n}_{E,T}$ is the outward normal with respect to E and $\nabla u_h \cdot \vec{n}_{E,T}$ is the discrete (outward-pointing) normal flux. An a posteriori estimate of the *discretization error* $e = u - u_h$ may be computed from the *equidistributed* interior

edge flux jump $R_E := \frac{1}{2} \llbracket \frac{\partial u_h}{\partial n} \rrbracket$, together with the *interior element residual*

$$R_T := \{f + \nabla^2 u_h\}|_T.$$

This is a consequence of the fact that the error satisfies

$$\sum_{T \in \mathcal{T}_h} (\nabla e, \nabla v)_T = \sum_{T \in \mathcal{T}_h} \left[(R_T, v)_T - \sum_{E \in \mathcal{E}(T)} \langle R_E, v \rangle_E \right] \quad (6)$$

for all $v \in V$, where $\mathcal{E}(T)$ is the set of edges associated with element T ; $\langle \cdot, \cdot \rangle_E$ denotes the L_2 inner product on E ; and \mathcal{T}_h is the set of all elements. See Elman et al. [2005b, p. 50] for further details.

The IFISS software computes an a posteriori error estimate whenever \mathbf{Q}_1 approximation is used. This is particularly straightforward to implement, in that R_E is piecewise linear and $R_T = f|_T$ can be approximated by a piecewise constant function R_T^0 by evaluating f at the element centroid. Consider the higher order correction space

$$\mathcal{Q}_T = \mathcal{Q}_T \oplus B_T,$$

so that \mathcal{Q}_T is the space spanned by biquadratic edge bubbles ψ_E , and B_T is the space spanned by interior biquadratic bubbles ϕ_T . The energy norm of the error is estimated by solving a 5×5 local Poisson problem posed over each element of the grid¹:

$$(\nabla e_T, \nabla v)_T = (R_T^0, v)_T - \sum_{E \in \mathcal{E}(T)} \langle R_E, v \rangle_E \quad \forall v \in \mathcal{Q}_T.$$

Once e_T has been computed, the local, elementwise, error estimator is given by $\eta_T = \|\nabla e_T\|_T$, and the global error estimator is given by

$$\eta := \left(\sum_{T \in \mathcal{T}_h} \eta_T^2 \right)^{1/2} \approx \|\nabla(u - u_h)\|.$$

This approach for error estimation was originally introduced in Bank and Weiser [1985]. It belongs to a class of methods referred to as *implicit estimators*, see Ainsworth and Oden [2000]. An analysis of its effectiveness as an estimator can be found in Elman et al. [2005b].

2.2 The Convection-Diffusion Equation

The equation

$$-\epsilon \nabla^2 u + \bar{w} \cdot \nabla u = 0 \quad \text{in } \Omega, \quad (7)$$

(with $\epsilon > 0$) arises in numerous models of flows and other physical phenomena. In a typical application, the unknown function u represents the concentration of a pollutant being transported (or ‘convected’) along a stream moving at velocity, \bar{w} and also subject to diffusive effects. The convection-diffusion equation

¹The local problem definition needs to be slightly modified for those elements having one or more edges on the Dirichlet boundary $\partial\Omega_D$.

considered in IFISS is (7) posed on a two-dimensional domain Ω , together with the boundary conditions

$$u = g_D \text{ on } \partial\Omega_D, \quad \frac{\partial u}{\partial n} = 0 \text{ on } \partial\Omega_N. \quad (8)$$

We will refer to the velocity vector \vec{w} as the *wind*.

It is well known that applying the Galerkin finite element method to (7) and (8) will often result in a discrete solution that exhibits nonphysical oscillations. This unwanted effect is minimized in IFISS through use of the *streamline diffusion method* of Hughes and Brooks [1979]: the discrete problem is stabilized by adding diffusion in the streamline direction. The resulting discrete formulation, which is the analogue of (2) for the convection-diffusion equation with discrete stabilization, is

$$\epsilon(\nabla u_h, \nabla v_h) + (\vec{w} \cdot \nabla u_h, v_h) + \sum_k \delta_k (\vec{w} \cdot \nabla u_h, \vec{w} \cdot \nabla v_h)_{\square_k} = 0 \quad \forall v_h \in V_h, \quad (9)$$

where the sum is taken over all elements \square_k in the grid.

To implement (9), we need a way of choosing locally defined parameters δ_k . One way of doing this is suggested by the analysis in Fischer et al. [1999] and Elman and Ramage [2002]. Specifically, we choose

$$\delta_k = \begin{cases} \frac{h_k}{2|\vec{w}_k|} \left(1 - \frac{1}{\mathcal{P}_h^k}\right) & \text{if } \mathcal{P}_h^k > 1 \\ 0 & \text{if } \mathcal{P}_h^k \leq 1. \end{cases} \quad (10)$$

Here, $|\vec{w}_k|$ is the Euclidean norm of the wind at the element centroid, h_k is a measure of the element length in the direction of the wind,² and $\mathcal{P}_h^k := |\vec{w}_k| h_k / (2\epsilon)$ is the so-called *element Peclet number*.

For all convection-diffusion problems solved in IFISS, the discretization is done using \mathbf{Q}_1 approximation on either uniform or stretched grids with stabilization parameter (10). There is also scope for a user-defined stabilization parameter so that the effect of adding streamline diffusion can be studied in detail. Further information specific to finite element modelling of convection-diffusion problems can be found in Elman et al. [2005b], Gresho and Sani [1998], Morton [1996], Quarteroni and Valli [1997] and Roos et al. [1996]. The a posteriori error estimation strategy built into the IFISS software is a natural extension of that used for the Poisson equation. Given the discrete solution u_h , the error $e = u - u_h \in V$ is estimated locally by solving the 5×5 local Poisson problem

$$\epsilon(\nabla e_T, \nabla v) = (R_T^0, v)_T - \epsilon \sum_{E \in \mathcal{E}(T)} \langle R_E, v \rangle_E \quad \forall v \in \mathcal{Q}_T$$

posed over each element of the grid, where the right hand side data is given by the equidistributed interior edge flux jumps $R_E := \frac{1}{2} \llbracket \frac{\partial u_h}{\partial n} \rrbracket$, together with the piecewise constant interior residuals

$$R_T^0 := \{f - \vec{w} \cdot \nabla u_h\}|_T.$$

²On a rectangle with sides of length h_x, h_y in which the wind forms an angle $\theta = \arctan(|w_y/w_x|)$ at the centroid, h_k is given by $\min(h_x/\cos\theta, h_y/\sin\theta)$.

In this case, the local, elementwise, error estimator is given by $\eta_T = \|\nabla e_T\|_T$, and the global error estimator is again given by

$$\eta := \left(\sum_{T \in \mathcal{T}_h} \eta_T^2 \right)^{1/2} \approx \|\nabla(u - u_h)\|.$$

This approach was originally suggested by Kay and Silvester [2001] and is a simplified version of the implicit estimator introduced and analyzed in Verfürth [1998].

2.3 The Stokes Equations

The Stokes equations, given by

$$-\nabla^2 \vec{u} + \nabla p = \vec{0}, \quad (11)$$

$$\nabla \cdot \vec{u} = 0, \quad (12)$$

together with boundary conditions (analogous to (8))

$$\vec{u} = \vec{w} \text{ on } \partial\Omega_D, \quad \frac{\partial \vec{u}}{\partial n} - \vec{n}p = \vec{0} \text{ on } \partial\Omega_N, \quad (13)$$

represent a fundamental model of viscous flow. The variable \vec{u} is a vector-valued function representing the velocity of the fluid, and the scalar function p represents the pressure. The equation (11) represents conservation of the momentum of the fluid (the *momentum equation*), and equation (12) enforces conservation of mass (the *incompressibility constraint*). The crucial modelling assumption is that the flow is ‘low speed,’ so that convection effects can be neglected.

The formal aspects of finite element discretization of the Stokes equations are the same as for the Poisson equation. However, two different types of test function are needed in this case, namely vector-valued velocity functions $\vec{v}_h = [v_x, v_y]^T$ and scalar pressure functions q_h . By extending this notation to the discrete case in an obvious way, the weak formulation can be written as

$$\begin{aligned} (\nabla \vec{u}_h, \nabla \vec{v}_h) - (p_h, \nabla \cdot \vec{v}_h) &= 0 & \forall \vec{v}_h \in \mathbf{X}_h, \\ (q_h, \nabla \cdot \vec{u}_h) &= 0 & \forall q_h \in M_h, \end{aligned}$$

where \mathbf{X}_h and M_h are appropriate finite dimensional subspaces of $\mathbf{H}^1(\Omega)$ and $L_2(\Omega)$, respectively. The fact that the velocity and pressure spaces are approximated independently leads to the nomenclature, *mixed approximation*. Implementation again entails defining appropriate bases for the velocity and pressure finite element spaces and constructing the associated finite element coefficient matrix (see Elman et al. [2005b, section 5.3]). In matrix form, the resulting linear system is

$$\begin{bmatrix} \mathbf{A} & B^T \\ B & O \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}. \quad (14)$$

This is a system of dimension $n_u + n_p$, where n_u and n_p are the numbers of velocity and pressure basis functions, respectively. The matrix \mathbf{A} is a 2×2 block

diagonal matrix with scalar Laplacian matrices defining the diagonal blocks, and the matrix B is an $n_p \times n_u$ rectangular matrix.

If (14) is to properly represent a continuous Stokes problem, then the finite dimensional approximation spaces need to be chosen carefully. The theory of how to choose basis functions appropriately is dealt with by the *inf-sup* stability condition. We omit a discussion of this analysis; full details can be found in Brezzi and Fortin [1991], and an accessible discussion is available in Elman et al. [2005b, Section 5.3.1]. Two inf-sup stable mixed methods are implemented in IFISS:

- \mathbf{Q}_2 – \mathbf{Q}_1 . The so-called *Taylor-Hood* method uses \mathbf{Q}_2 approximation for velocity and continuous \mathbf{Q}_1 approximation for pressure.
- \mathbf{Q}_2 – \mathbf{P}_{-1} . This uses \mathbf{Q}_2 approximation for velocity together with a *discontinuous* linear pressure. Specifically, the \mathbf{P}_{-1} element has a central node with three associated degrees of freedom, the pressure at the centroid, and its x and y derivatives.

It is an unfortunate fact of life that the simplest mixed approximations, such as the case when velocities and pressures are defined on the same set of grid points, are not inf-sup stable. In such cases however, it is possible to use *stabilization* techniques to circumvent the inf-sup condition. Details can be found in Elman et al. [2005b, section 5.3.2]. In practice, stabilisation of the lowest order methods leads to a Stokes system of the form

$$\begin{bmatrix} \mathbf{A} & B^T \\ B & -\beta C \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \quad (15)$$

for some $n_p \times n_p$ symmetric positive semidefinite matrix C and a stabilization parameter $\beta > 0$. Two stabilized element pairs are implemented in IFISS:

- \mathbf{Q}_1 – \mathbf{P}_0 . This is the most famous example of an unstable element pair, using bilinear approximation for velocity and a constant approximation for the pressure.
- \mathbf{Q}_1 – \mathbf{Q}_1 . Here the velocity and pressure degrees of freedom are defined at the same set of grid points. This is very appealing from the point of view of ease of programming and computational efficiency.

Stokes flow test problems can be solved with IFISS using any of the four discretizations listed above, with stabilization applied where necessary. An optimal value of the stabilization parameter is determined automatically, although the parameter value can also be changed by a user who wishes to investigate the effects of stabilization on solution accuracy.

An efficient strategy for a posteriori estimation for the Stokes flow problem can be realized by building on the Poisson error estimation strategy and solving an analogous local problem for each velocity component. Note that the velocity approximation typically has a discontinuous normal derivative across interelement boundaries, so it is natural to generalize the flux jump (5) to give a *stress jump* across edge E adjoining elements T and S :

$$\llbracket \nabla \tilde{u}_h - p_h \tilde{\mathbf{I}} \rrbracket := ((\nabla \tilde{u}_h - p_h \tilde{\mathbf{I}})|_T - (\nabla \tilde{u}_h - p_h \tilde{\mathbf{I}})|_S) \cdot \tilde{\mathbf{n}}_{E,T}.$$

If a C^0 pressure approximation is used, then the jump in $p_h \bar{\mathbf{I}}$ is zero. Exactly as for the characterization (6) for the Poisson problem, the mixed approximation error functions $\bar{e} := \bar{u} - \bar{u}_h \in \mathbf{X}$ and $\epsilon := p - p_h \in M$ can be shown to satisfy a set of *localized Stokes problems*:

$$\begin{aligned} \sum_{T \in \mathcal{T}_h} \{(\nabla \bar{e}, \nabla \bar{v})_T - (\epsilon, \nabla \cdot \bar{v})_T\} &= \sum_{T \in \mathcal{T}_h} \left[(\bar{R}_T, \bar{v})_T - \sum_{E \in \mathcal{E}(T)} \langle \bar{R}_E, \bar{v} \rangle_E \right] \\ - \sum_{T \in \mathcal{T}_h} (q, \nabla \cdot \bar{e})_T &= \sum_{T \in \mathcal{T}_h} (R_T, q)_T \end{aligned}$$

for all $(\bar{v}, q) \in \mathbf{X} \times M$. Here, the equidistributed stress jump is $\bar{R}_E := \frac{1}{2} \llbracket \nabla \bar{u}_h - p_h \bar{\mathbf{I}} \rrbracket$, and the elementwise interior residuals are given by $\bar{R}_T := \{\nabla^2 \bar{u}_h - \nabla p_h\}|_T$ and $R_T := \{\nabla \cdot \bar{u}_h\}|_T$, respectively.

The IFISS software computes an estimate of the approximation error whenever \mathbf{Q}_1 velocity approximation is used (that is, if either stabilized \mathbf{Q}_1 - \mathbf{P}_0 or stabilized \mathbf{Q}_1 - \mathbf{Q}_1 mixed approximation is used). Using the space $\bar{Q}_T := (Q_T)^2$, the velocity errors are given by solving a pair of 5×5 Poisson problems. Specifically, $\bar{e}_T \in \bar{Q}_T$ is computed satisfying

$$(\nabla \bar{e}_T, \nabla \bar{v})_T = (\bar{R}_T, \bar{v})_T - \sum_{E \in \mathcal{E}(T)} \langle \bar{R}_E, \bar{v} \rangle_E \quad \forall \bar{v} \in \bar{Q}_T, \quad (16)$$

for every element in the grid. The local error estimator is given by the combination of the ‘energy norm’ of the velocity error and the L_2 norm of the element divergence error, that is,

$$\eta_T^2 := \|\nabla \bar{e}_T\|_T^2 + \|R_T\|_T^2. \quad (17)$$

The global error estimator is $\eta := \left(\sum_{T \in \mathcal{T}_h} \eta_T^2 \right)^{1/2}$.

This method of error estimation was introduced by Kay and Silvester [1999], and is a modification of an approach devised by Ainsworth and Oden [1997]. For an assessment of its effectiveness, see Elman et al. [2005b, section 5.4.2].

2.4 The Navier-Stokes Equations

The steady-state Navier-Stokes equations can be written as

$$\begin{aligned} -\nu \nabla^2 \bar{u} + \bar{u} \cdot \nabla \bar{u} + \nabla p &= \bar{f}, \\ \nabla \cdot \bar{u} &= 0, \end{aligned} \quad (18)$$

where $\nu > 0$ is a given constant called the *kinematic viscosity*. Associated boundary conditions are given by

$$\bar{u} = \bar{w} \text{ on } \partial\Omega_D, \quad \nu \frac{\partial \bar{u}}{\partial n} - \bar{n} p = \bar{0} \text{ on } \partial\Omega_N. \quad (19)$$

This system is the basis for computational modelling of the flow of an incompressible Newtonian fluid such as air or water. The presence of the nonlinear convection term $\bar{u} \cdot \nabla \bar{u}$ means that boundary value problems associated with the Navier-Stokes equations can have more than one solution.

The discrete formulation of the Navier-Stokes flow problem is determined almost exactly as in the previous section, the main difference being the presence of the convection term. An important point is that inf-sup stability is a necessary condition for the mixed approximation to be effective in the absence of stabilization.

Mixed finite element discretization of the weak formulation of the Navier-Stokes equations gives rise to a nonlinear system of algebraic equations. Two classical iterative procedures for solving this system are implemented in IFISS:

- Newton iteration.* With this approach, if the initial velocity estimate is ‘sufficiently close’ to a branch of nonsingular solutions, then quadratic convergence to a uniquely defined fixed point is guaranteed for large enough values of ν . See Girault and Raviart [1986, pp. 362–366] for details.
- Picard iteration.* This corresponds to a simple fixed point iteration strategy, with the convection coefficient evaluated at the current velocity. Although the rate of convergence of Picard iteration is only linear in general, the ball of convergence is often much bigger than that of Newton iteration.

Either strategy can be used in IFISS. The default option is a *hybrid* method that performs a small user-specified number of Picard steps (the default is 2) with the aim of generating a good starting value for Newton’s method, followed by Newton iteration.

At each step of the nonlinear iteration, a system of equations must be solved. The coefficient matrix has the form

$$\begin{bmatrix} \mathbf{F} & B^T \\ B & -\frac{1}{\nu}C \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix}, \quad (20)$$

where the structure of \mathbf{F} depends on the nonlinear solution algorithm. For Picard iteration, \mathbf{F} is a block diagonal matrix in which each of the diagonal blocks is a discrete convection-diffusion operator. For Newton iteration, \mathbf{F} has a more complex structure; the details can be found in Elman et al. [2005b, section 7.3]. The matrix C is the Stokes stabilization matrix in the case of $\mathbf{Q}_1-\mathbf{P}_0$ or $\mathbf{Q}_1-\mathbf{Q}_1$ mixed approximation, and is the zero matrix otherwise.

An efficient a posteriori estimation strategy for the Navier-Stokes equations can be realized by combining the convection-diffusion estimator with the Stokes error estimation strategy, and solving a local Poisson problem for each component of velocity. Further details are given in Elman et al. [2005b, section 7.4.2].

3. SOLVER FEATURES

The second feature of the IFISS code is its facility for exploring the performance of iterative solvers and preconditioning techniques for the solution of the finite element linear systems described in the previous section. Details of standard iterative methods can be found in many textbooks (see for example Axelsson [1994], Greenbaum [1997] or Saad [2003]) and so are not reproduced here.

3.1 Krylov Subspace Solvers

The iterative methods implemented in IFISS are all *Krylov subspace* methods. The two methods available in IFISS for solving symmetric systems are

- *The conjugate gradient method* (CG) [Hestenes and Stiefel 1952]. This method is applicable when the coefficient matrix A is symmetric positive definite. IFISS uses the standard (preconditioned) matlab routine `pcg`.
- *The minimum residual method* (MINRES) [Paige and Saunders 1975]. This is applicable to any symmetric system; in particular, MINRES is a robust algorithm for symmetric indefinite coefficient matrices. The implementation in preconditioned form used in IFISS is the standard matlab routine `minres`.

There are also two iterative methods for solving nonsymmetric systems:

- *The generalized minimum residual method* (GMRES) [Saad and Schultz 1986]. For nonsymmetric problems, this method represents the standard approach for constructing iterates satisfying an optimality condition. The implementation of GMRES with right preconditioning used in IFISS is a modification of a routine written by Kelley [1995].
- `BICGSTAB(ℓ)` [Sleijpen and Fokkema 1993]. In contrast to GMRES this algorithm does not satisfy an optimality condition but has fixed computational costs at each step. The implementation used in IFISS has $\ell = 2$ and is a modification of a routine written by Sleijpen.

The methods used in IFISS are tailored to the PDEs modelled, so not all of the methods are available for every type of PDE problem. For the benchmark problems described in Section 2, preconditioned versions of appropriate choices of these Krylov subspace methods are available, as follows:

- The Poisson equation: CG or MINRES.
- The convection-diffusion equation: GMRES or `BICGSTAB(ℓ)`.
- The Stokes equations: MINRES.
- The Navier-Stokes equations: GMRES or `BICGSTAB(ℓ)`.

3.2 Preconditioners

Whatever Krylov subspace method is used, it is usually necessary to accelerate convergence by applying *preconditioning*. The various preconditioners M implemented in IFISS are described below.

When solving PDEs, the underlying physical problem plays an important role in dictating the efficiency (or otherwise) of a particular preconditioning strategy. The following preconditioners are available for the scalar (Poisson and convection-diffusion) equations implemented in IFISS:

- *Diagonal scaling*. M is the diagonal matrix whose entries are those on the main diagonal of the system coefficient matrix.
- *Incomplete Cholesky and LU factorizations* [Meijerink and van der Vorst 1977]. The preconditioner is the matrix $M = LU$, where L and U are the

incomplete Cholesky (if A is symmetric) or LU (if A is nonsymmetric) factors of A . The factors have no fill: the nonzero structure of $L + U$ is the same as that of the coefficient matrix.

- Geometric multigrid preconditioning* (GMG). The `IFISS` code uses one multigrid V-cycle as a preconditioner, with direct approximation at the coarse grid level, and bilinear interpolation and its transpose, as grid transfer operators. (See, for example, Briggs et al. [2000], Hackbusch [1985], or Wesseling [1992] for definitions and properties of multigrid.) There is a choice of smoothing operator: weighted Jacobi, ILU and point or line versions of Gauss-Seidel. The number of pre- and post-smoothing steps is selected by the user. For the convection-diffusion equation on a square domain, a ‘multi-directional’ line Gauss-Seidel smoother is available to handle recirculating flows.³
- Algebraic multigrid preconditioning* (AMG). When gaining access to geometric data is difficult, purely algebraic multigrid processes can be employed instead; see Briggs et al. [2000] or Trottenberg et al. [2001] for details. The `IFISS` code provides an interface to the algebraic multigrid solver of the commercial finite element code `FEMLAB`,⁴ which enables the use of AMG as a preconditioner.

For the block systems arising from the Stokes (15) and Navier-Stokes (20) equations, several preconditioners specifically designed for these problems are available. All Stokes preconditioners in `IFISS` have the structure proposed by Wathen and Silvester [1993] and Silvester and Wathen [1994], namely,

$$M = \begin{bmatrix} \mathbf{P} & 0 \\ 0 & T \end{bmatrix}. \quad (21)$$

- Stokes ideal preconditioning*. Here, $\mathbf{P} = \mathbf{A}$ and $T = Q_p$ where \mathbf{A} and Q_p are the velocity Laplace and pressure mass matrices, respectively.
- Stokes diagonal preconditioning* $\mathbf{P} = \text{diag}(\mathbf{A})$ and $T = \text{diag}(Q_p)$.
- Stokes multigrid preconditioning*. \mathbf{P} is defined implicitly by the condition that \mathbf{P}^{-1} corresponds to one GMG V-cycle applied to the Laplacian component, and $T = \text{diag}(Q_p)$.

The Navier-Stokes preconditioners implemented in `IFISS` are all of the form

$$M = \begin{bmatrix} \mathbf{M}_F & B^T \\ 0 & -M_S \end{bmatrix} \quad (22)$$

where \mathbf{M}_F approximates the convection-diffusion operator \mathbf{F} and M_S approximates the Schur complement $S = B\mathbf{F}^{-1}B^T + \frac{1}{\nu}C$.

- Pressure convection-diffusion preconditioning* [Silvester et al. 2001; Kay et al. 2002]. Here, $M_S = \hat{Q}_p F_p^{-1} A_p$, where $\hat{Q}_p = \text{diag}(Q_p)$ is the diagonal of the mass matrix on the pressure space, and F_p and A_p are discrete convection-diffusion and Laplace operators, respectively, on the pressure space.

³GMG may also be applied as a stand-alone solver without Krylov subspace iteration.

⁴Available from <http://www.comsol.com>.

—*Least-squares commutator preconditioning* [Elman 1999; Elman et al. 2006]. With this preconditioner, M is also of the form (22) but here, for stable discretizations, the Schur complement matrix is approximated by

$$M_S = (B\hat{Q}_v^{-1}B^T)(B\hat{Q}_v^{-1}\mathbf{F}\hat{Q}_v^{-1}B^T)^{-1}(B\hat{Q}_v^{-1}B^T),$$

where \hat{Q}_v is the diagonal of the velocity mass matrix. The analogue for stabilized elements is discussed in Elman et al. [2006].

With both of these strategies, for \mathbf{M}_F , there is a choice of using *ideal preconditioning*, with $\mathbf{M}_F = \mathbf{F}$, or replacing \mathbf{M}_F with an operator defined by one step of GMG or AMG iteration to approximate the action of \mathbf{F}^{-1} . Similarly, the action of A_p^{-1} (or $(B\hat{Q}_v^{-1}B^T)^{-1}$) required for M_S^{-1} can be replaced by a multigrid approximation.

4. DESIGN FEATURES

We now discuss some important aspects of the design and organization of the IFISS package. More details can be found in the user manual [Elman et al. 2005a].

4.1 Model Problems

As stated in the introduction, IFISS focuses on four specific PDEs. Easy investigation of approximation properties and behaviour of solvers for each of these is facilitated via a set of built-in drivers that set up and solve four reference test problems for each PDE. The test problems have been selected to illustrate the interesting features of examples of each type of equation, and to allow users to experiment with various aspects of the modelling and solution processes. A full description of the reference problems can be found in Elman et al. [2005b]. Further details can be found in the user manual accompanying the software [Elman et al. 2005a], along with a description of how the model problems can be easily adapted to deal with new domains, PDE features or boundary conditions.

4.2 Modules and Directory Structure

The IFISS package groups the code required to define and solve each individual PDE problem into modular components. In particular, specification of the discrete version of each of the PDEs requires choices of domain, finite element, and boundary conditions; each of these operations is defined in a set of self-contained modules. For example, the choice of whether discretization is based on linear or quadratic elements is independent of the choice of boundary conditions in the problem. Moreover, once a problem has been posed, the choice of iterative method to solve it also includes a set of independent modules defining the preconditioning operators.

The IFISS directory is organized into subdirectories according to tasks performed. The individual PDE reference problems each have their own directory:

- \ifiss\diffusion\ the Poisson equation,
- \ifiss\convection\ the convection-diffusion equation,
- \ifiss\stokes_flow\ the Stokes equations,
- \ifiss\navier_flow\ the Navier-Stokes equations.

Within each of these directories, there is a subdirectory `\test_problems\` containing all function files that define the boundary conditions and coefficients associated with the reference PDE problems. There are also task-specific directories for other purposes:

```

—\ifiss\grids\      domain geometry and grid generation,
—\ifiss\graphs\    visualization tools,
—\ifiss\solvers\   iterative solution algorithms.

```

Two directories are used to store data accumulated during execution of the program:

```

—\ifiss\datafiles\  finite element matrices or multigrid data
                    that can be reused,
—\ifiss\plotfiles\  repository for graphical displays.

```

All files are viewable using the matlab editor.

4.3 Implementation Issues

An effort has been made to implement the underlying mathematical ideas in a manner that produces efficient matlab code while at the same time taking advantage of matlab features that enable flexibility and clarity of code. We highlight two examples of this philosophy below.

—*Long inner loops (vectorization)*. In standard finite element implementations, the assembly of global stiffness matrices entails an outer loop over elements with local stiffness matrices constructed and accumulated into the global matrix. A high level depiction of the conventional construction of the global matrix A from 4×4 local matrices $\{A_{loc}(k, :, :)\}$ defined on element k is

```

for k=1:#elements
  for j=1:4
    for i=1:4
      A(P(k,i),P(k,j)) = A(P(k,i),P(k,j)) + Aloc(k,i,j)
    end
  end
end
end

```

Here P maps local to global indices, and the local matrices require construction via quadrature. The short inner loops (including those used for quadrature) make this inefficient in matlab. To circumvent this, the `IFISS` implementation constructs a three-dimensional array of local stiffness matrices $Ae(\#elements, 4, 4)$ using a loop of the form:

```

for j=1:4
  for i=1:4
    construct Ae(:,i,j)
  end
end
end

```

The global matrix (of order n) is then assembled via

```

for i=1:4
    nrow=ev(:,i);
    for j=1:4
        ncol=ev(:,j);
        A = A + sparse(nrow,ncol,Ae(:,i,j),n,n);
    end
end
end

```

Here, for each local row and column index pair (i, j) , the arrays `nrow` and `ncol` delineate the row and column indices in `A` where the local matrix entries (one for each element) are accumulated. The ‘inner loop’ in both of these constructions is a vectorized computation performed on all elements and is very efficient in `matlab`.

—*Structures for preconditioned solvers.* The portion of the package concerned with iterative solvers is designed to facilitate experimentation with different benchmark problems and preconditioning operators. This is done through the use of `matlab` structures, by means of which, information about the problem and preconditioners is passed to the Krylov subspace solvers. In particular, the nonsymmetric solvers accept two parameters, `afun_par`, which contains all of the information associated with the coefficient matrix, and `mfun_par`, containing all information associated with the preconditioning operator. For example, solution of the convection-diffusion equation by `GMRES` with geometric multigrid preconditioning entails a function call of the form

```
[x_it,flag,iter,resvec] = gmres_r(afun_par,mfun_par,fsupg,params,x0);
```

The argument `afun_par` is a structure that defines the coefficient matrix:

```
afun_par = struct('Afun','a_cdt','A',Asupg);
```

where `a_cdt` is the name of the `matlab` function that performs a matrix-vector product, and `Asupg` is the coefficient matrix. The argument `mfun_par` defines the preconditioning operator:

```
mfun_par = struct('Mfun','m_mg','mgdata',mgdata,...
    'smooth_data',smooth_data, 'nc',nc,...
    'npre',npre, 'npost',npost, 'sweeps',sweeps);
```

Here, `m_mg` is the name of the `matlab` function that performs one GMG step, which is used as a preconditioner; `mgdata` contains all hierarchical multigrid data such as coarse grid and grid transfer operators; and `smooth_data` contains the multigrid smoothing operators. Note that `mgdata` and `smooth_data` are structures themselves. The remaining arguments are other multigrid parameters, such as the number of presmoothing and postsmoothing steps to perform. Using this naming convention, we have found it easy to augment the package with new preconditioning operators.⁵

⁵Systems with symmetric coefficient matrices arising from the diffusion and Stokes equations use the built-in `matlab` `pcg` and `minres` functions. For these, we found it easier to pass the coefficient matrices explicitly rather than with structures.

4.4 Graphics and Visualization

For each model problem, graphical output includes depictions of

- the computational domain and finite element grid,
- contour and mesh plots of the computed solutions,
- contour and mesh plots of error estimates, and
- for iterative solution, plots of the behavior of the residual during the course of the iteration.

4.5 Running Jobs in batchmode

The model problems available with the package can be generated interactively by running one of the driver routines provided. Sometimes, however, it is more convenient to do this without having interactive input from the user. To this end, `IFISS` also provides a `batchmode` facility via which data may be input from a preprepared file rather than directly from the terminal. The specific parameters that need to be input will of course vary from problem to problem, and the input file must be prepared accordingly. Sample input files for each of the model problems are provided; these can be easily modified by the user for a particular run. The names of these input files must have the form “*_batch.m” where “*” begins with one of “P”, “CD”, “S” or “NS” for the Poisson, convection-diffusion, Stokes or Navier-Stokes equations, respectively. For example, typing the command

```
batchmode('NS2')
```

uses the file `NS2_batch.m`.

A similar `batchmode` facility is available for running the driver `it_solve` without interactive input after a discrete system has been generated in `batchmode`. Input files must have names of the form `itsolve*_batch.m`, and a template is provided within the software package.

REFERENCES

- AINSWORTH, M. AND ODEN, J. 1997. A posteriori error estimates for Stokes' and Oseen's equations. *SIAM J. Numer. Anal.* 34, 228–245.
- AINSWORTH, M. AND ODEN, J. 2000. *A Posteriori Error Estimation in Finite Element Analysis*. Wiley, New York.
- AXELSSON, O. 1994. *Iterative Solution Methods*. Cambridge University Press, Cambridge.
- BANK, R. E. AND WEISER, A. 1985. Some a posteriori error estimators for elliptic partial differential equations. *Math. Comp.* 44, 283–301.
- BRAESS, D. 1997. *Finite Elements*. Cambridge University Press, London.
- BRENNER, S. C. AND SCOTT, L. R. 1994. *The Mathematical Theory of Finite Element Methods*. Springer-Verlag, New York.
- BREZZI, F. AND FORTIN, M. 1991. *Mixed and Hybrid Finite Element Methods*. Springer-Verlag, New York.
- BRIGGS, W., HENSON, V., AND MCCORMICK, S. 2000. *A Multigrid Tutorial*. SIAM, Philadelphia.
- CIARLET, P. G. 1978. *The Finite Element Method for Elliptic Problems*. North-Holland, Amsterdam.
- ELMAN, H., HOWLE, V. E., SHADID, J., SILVESTER, D., AND TUMINARO, R. 2006. Least squares preconditioners for stabilized discretizations of the Navier-Stokes equations. Tech. rep., MIMS Eprint 2006.55, Manchester Institute for Mathematical Sciences.

- ELMAN, H. C. 1999. Preconditioning for the steady-state Navier-Stokes equations with low viscosity. *SIAM J. Sci. Comput.* 20, 1299–1316.
- ELMAN, H. C. AND RAMAGE, A. 2002. An analysis of smoothing effects of upwinding strategies for the convection-diffusion equation. *SIAM J. Numer. Anal.* 40, 254–281.
- ELMAN, H. C., RAMAGE, A., AND SILVESTER, D. 2005a. Incompressible Flow Iterative Solution Software (IFISS) installation & software guide. http://www.cs.umd.edu/~elman/ifiss/ifiss_guide.pdf.
- ELMAN, H. C., SILVESTER, D. J., AND WATHEN, A. J. 2005b. *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluid Dynamics*. Oxford University Press, Oxford, UK.
- FISCHER, B., RAMAGE, A., SILVESTER, D., AND WATHEN, A. 1999. On parameter choice and iterative convergence for stabilised discretisations of advection-diffusion problems. *Comput. Methods Appl. Mech. Eng.* 179, 185–202.
- GIRAULT, V. AND RAVIART, P.-A. 1986. *Finite Element Methods for Navier-Stokes Equations*. Springer-Verlag, Berlin.
- GREENBAUM, A. 1997. *Iterative Methods for Solving Linear Systems*. SIAM, Philadelphia.
- GRESHO, P. AND SANI, R. 1998. *Incompressible Flow and the Finite Element Method: Volume 1: Advection-Diffusion*. John Wiley, Chichester.
- GUNZBURGER, M. D. 1989. *Finite Element Methods for Viscous Incompressible Flows*. Academic Press, San Diego.
- HACKBUSCH, W. 1985. *Multi-Grid Methods and Applications*. Springer-Verlag, Berlin.
- HESTENES, M. R. AND STIEFEL, E. 1952. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bureau Stand.* 49, 409–435.
- HUGHES, T. AND BROOKS, A. 1979. A multi-dimensional upwind scheme with no crosswind diffusion. In *Finite Element Methods for Convection Dominated Flows*, T. Hughes, Ed. AMD–vol 34, ASME, New York.
- JOHNSON, C. 1987. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, New York.
- KAY, D., LOGHIN, D., AND WATHEN, A. 2002. A preconditioner for the steady-state Navier-Stokes equations. *SIAM J. Sci. Comput.* 24, 237–256.
- KAY, D. AND SILVESTER, D. 1999. A posteriori error estimation for stabilized mixed approximations of the Stokes equations. *SIAM J. Sci. Comput.* 21, 1321–1336.
- KAY, D. AND SILVESTER, D. 2001. The reliability of local error estimators for convection-diffusion equations. *IMA J. Numer. Anal.* 21, 107–122.
- KELLEY, T. 1995. *Iterative Methods for Linear and Nonlinear Equations*. SIAM, Philadelphia.
- MELJERINK, J. A. AND VAN DER VORST, H. A. 1977. An iterative solution method for linear systems of which the coefficient matrix is a symmetric m-matrix. *Math. Comp.* 31, 148–162.
- MILLER, J. J. H., O’RIORDAN, E., AND SHISHKIN, G. I. 1995. *Fitted Numerical Methods for Singularly Perturbed Problems—Error Estimates in the Maximum Norm for Linear Problems in One and Two Dimensions*. World Scientific, Singapore.
- MORTON, K. W. 1996. *Numerical Solution of Convection-Diffusion Problems*. Chapman & Hall, London.
- OCKENDON, J., HOWISON, S., LACEY, A., AND MOVCHAN, A. 1999. *Applied Partial Differential Equations*. Oxford University Press, Oxford.
- PAIGE, C. AND SAUNDERS, M. 1975. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.* 12, 617–629.
- QUARTERONI, A. AND VALLI, A. 1997. *Numerical Approximation of Partial Differential Equations*. Springer-Verlag, Berlin.
- ROOS, H.-G., STYNES, M., AND TOBISKA, L. 1996. *Numerical Methods for Singularly Perturbed Differential Equations*. Springer-Verlag, Berlin.
- SAAD, Y. 2003. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia. Second ed.
- SAAD, Y. AND SCHULTZ, M. H. 1986. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* 7, 856–869.
- SILVESTER, D., ELMAN, H., KAY, D., AND WATHEN, A. 2001. Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow. *J. Comp. Appl. Math.* 128, 261–279.
- SILVESTER, D. AND WATHEN, A. 1994. Fast iterative solution of stabilised Stokes systems. Part II: Using general block preconditioners. *SIAM J. Numer. Anal.* 31, 1352–1367.

- SLEIJPEN, G. L. G. AND FOKKEMA, D. R. 1993. BICGSTAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum. *Elec. Trans. Numer. Math.* 1, 11–32.
- TROTTEBERG, U., OOSTERLEE, C., AND SCHÜLLER, A. 2001. *Multigrid*. Academic Press, London.
- VERFÜRTH, R. 1998. A posteriori error estimators for convection-diffusion equations. *Numer. Math* 80, 641–663.
- WATHEN, A. AND SILVESTER, D. 1993. Fast iterative solution of stabilised Stokes systems. Part I: Using simple diagonal preconditioners. *SIAM J. Numer. Anal.* 30, 630–649.
- WESSELING, P. 1992. *An Introduction to Multigrid Methods*. John Wiley & Sons, New York.

Received October 2005; revised June 2006; accepted October 2006