CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2003; **00**:1–17 *Prepared using* cpeauth.cls *[Version: 2002/09/19 v2.02]*

Parallel iterative methods for Navier-Stokes equations and application to eigenvalue computation

Ivan G. Graham[1,*], Alastair Spence[2] and Eero Vainikko[3]

[1] *Department of Mathematical Sciences, University of Bath, E-mail: I.G.Graham@bath.ac.uk*
[2] *Department of Mathematical Sciences, University of Bath, E-mail: A.Spence@bath.ac.uk*
[3] *Department of Mathematical Sciences, University of Bath, E-mail: E.Vainikko@bath.ac.uk*

SUMMARY

We describe the construction of parallel iterative solvers for finite element approximations of the Navier-Stokes equations on unstructured grids using domain decomposition methods. The iterative method used is FGMRES, preconditioned by a parallel adaptation of a block preconditioner recently proposed by Kay, Loghin and Wathen. The parallelisation is achieved by adapting the technology of our domain decomposition solver `DOUG` (previously used for scalar problems) to block-systems. The iterative solver is applied to shifted linear systems that arise in eigenvalue calculations. To illustrate the performance of the solver, we compare both theoretically and practically several strategies for the calculation of eigenvalues of large sparse nonsymmetric matrices arising in the assessment of stability of flow past a cylinder.

KEY WORDS: Parallel iterative methods, domain decompostion, eigenvalues, incompressible Navier Stokes equations.

Introduction

When steady solutions of complex physical problems are computed numerically it is often crucial to compute their *stability* in order to, for example, check that the computed solution is "physical", or carry out a sensitivity analysis, or help understand complex nonlinear phenomena

*Correspondence to: Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, United Kingdom, tel: +441225386343 fax: +441225386492
Contract/grant sponsor: UK Engineering and Physical Sciences Research Council; contract/grant number: GR/M59075

near a bifurcation point. An important example is in the computation of fluid flows governed by the steady-state Navier-Stokes equations. Suppose that a velocity field $\mathbf{w}$ has been computed for some particular parameter values. To assess its stability it is necessary to solve the PDE eigenvalue problem:

$$\left.\begin{array}{rcl} -\epsilon\Delta\mathbf{u} + \mathbf{w}\cdot\nabla\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{w} + \nabla p &=& \lambda\mathbf{u} \\ \nabla\cdot\mathbf{u} &=& 0\ , \end{array}\right\} \tag{1}$$

for some eigenvalue $\lambda \in \mathbb{C}$ and nontrivial eigenfunction $(\mathbf{u}, p)$, satisfying suitable homogeneous boundary conditions. Here the parameter ϵ is the viscosity, which is inversely proportional to the Reynolds number Re. The eigenfunction $(\mathbf{u}, p)$ consists of the velocity $\mathbf{u}$, a vector field in 2D (resp. 3D) and the pressure p (a scalar field) both defined on a 2D (resp. 3D) computational (usually finite) domain. (See, for example [1].) For ease of exposition we shall restrict here to the 2D case.

The computation of the reference velocity $\mathbf{w}$ is, in itself, a large task, especially for 3D problems, or for 2D problems with multiple scales. The eigenvalue problem is generally accepted to be at least as hard as this, and in many cases considerably harder. Thus it is of great interest to find reliable and efficient algorithms for this task.

Restricting for convenience to 2D applications, discretisation of (1) with mixed finite elements yields the algebraic generalised eigenvalue problem:

$$A\mathbf{x} = \lambda M\mathbf{x} \tag{2}$$

where $\mathbf{x} = (\mathbf{U}_1^T, \mathbf{U}_2^T, \mathbf{P}^T)^T$ is a vector of n degrees of freedom approximating $(u_1, u_2, p)^T$, and the matrices A and M take the form:

$$A = \begin{bmatrix} F_{11} & F_{12} & B_1^T \\ F_{21} & F_{22} & B_2^T \\ B_1 & B_2 & 0 \end{bmatrix}\ ,\ M = \begin{bmatrix} M_u & 0 & 0 \\ 0 & M_u & 0 \\ 0 & 0 & 0 \end{bmatrix}\ . \tag{3}$$

Here, for example, F_{11} is the finite element stiffness matrix corresponding to the operator $u_1 \to -\epsilon\Delta u_1 + \mathbf{w}\cdot\nabla u_1 + (\partial w_1/\partial x)u_1$ and M_u is the mass matrix corresponding to the identity operator operating on a typical velocity component. Overall, A is unsymmetric, M is positive semi-definite, both are sparse and in real applications n will be very large ($10^5 - 10^9$ is common). Full details of this type of discretisation can be found, for example, in [1].

In stability analysis only a small number of "dangerous" eigenvalues are required, often those (possibly complex) eigenvalues nearest the imaginary axis. In this context it is usually necessary to perform "shift-invert" iterations, which require repeated solution of systems of the form

$$(A - \sigma M)\mathbf{y} = M\mathbf{x}, \tag{4}$$

for some *shift* σ (which may be near a spectral point) and for various right-hand sides $\mathbf{x}$. Examples of algorithms which require solves of the form (4) include classical inverse iteration, its subspace variant, or Arnoldi's method, based on Krylov spaces. In this paper we shall consider the Implicitly Restarted Arnoldi method [2], and inexact inverse iteration (see [3, 4, 5]).

If σ is close to an eigenvalue of (2) then the condition number of $A - \sigma M$ will be large. Consequently one might think that if an iterative method were used to solve (4), then convergence would be slow. In fact for the eigenvalue problem there are likely to be only a small number of isolated eigenvalues near zero and it is well known that in such cases the convergence of Krylov solvers is hardly affected. Indeed we shall present numerical results showing that for inexact inverse iteration it is advantageous to let the shift tend to the desired eigenvalue. An alternative approach is the Jacobi-Davidson method (see [6]) which seeks to remove the near singularity in the linear solve. We have not chosen this approach.

Previous algorithms developed at Bath [7] were able to find the eigenvalues of (2) nearest the imaginary axis on the assumption that systems like (4) could be solved exactly. These algorithms form a basis of the stability assessment module in the commercial finite element code ENTWIFE [8]. However, requiring exact solutions of (4) is impracticable for large problems.

In large applications, systems (4) have to be solved iteratively. For the rest of this paper we concentrate on describing a parallel solver for (4) in the special case when A and M arise from the Navier-Stokes equations, as in (3). At the end of the paper we describe some sample eigenvalue calculations using Arnoldi's method [2] and inexact inverse iteration, each iteration of which requires a solve of the form (4). We shall show that an approach based on a combination of Arnoldi's method and inexact inverse iteration can be more efficient than using only Arnoldi's method.

A preliminary version of this paper appeared in [9].

Iterative Method

Our iterative method for (4) is the Flexible Generalised Minimal Residual algorithm (FGMRES), which requires the repeated computation of matrix-vector multiplication with $A - \sigma M$ and converges with a rate determined by the distribution of the spectrum of $A - \sigma M$. The rate of convergence can be accelerated by introducing a (right) preconditioner C and solving, instead of (4), the system

$$(A - \sigma M)C\tilde{\mathbf{y}} = M\mathbf{x} , \tag{5}$$

after which $\mathbf{y}$ is retrieved by calculating $\mathbf{y} = C\tilde{\mathbf{y}}$. For best performance, C should firstly be a good approximate inverse for $A - \sigma M$. When solving (5) we require repeated matrix-vector multiplication with $(A - \sigma M)C$ which means secondly that this multiplication operation also needs to be sufficiently fast.

Our experience suggests that even though the shift σ might be near an eigenvalue λ of (2), a good preconditioner for $A - \sigma M$ in the PDE context can be found by using a good preconditioner for A (essentially since A represents a higher order operator than M). Thus in the next section we concentrate on describing our preconditioner for A.

Note that we choose to discuss the case of right preconditioning here because it fits in with the standard formulation of FGMRES [10].

Preconditioner for A

Block preconditioners for A

An ideal (right) preconditioner for A suggested by Elman and Silvester [11] is of the form

$$P = \begin{bmatrix} F_{11} & F_{12} & B_1^T \\ F_{21} & F_{22} & B_2^T \\ 0 & 0 & -X \end{bmatrix}, \tag{6}$$

where

$$X = \mathbf{B}\mathbf{F}^{-1}\mathbf{B}^T, \ \ \mathbf{F} = \begin{bmatrix} F_{11} & F_{12} \\ F_{21} & F_{22} \end{bmatrix}, \ \ \mathbf{B} = \begin{bmatrix} B_1 & B_2 \end{bmatrix}.$$

This lead to a preconditioned matrix AP^{-1} with a minimal polynomial of degree two, thus guaranteeing convergence in two iterations. This "perfect" preconditioner requires assembling and inverting the Schur complement X and inverting the 2×2 block matrix $\mathbf{F}$ and is not a practical choice. Elman [12] suggested the more practical approximation for X^{-1}:

$$X_B^{-1} = (\mathbf{B}\mathbf{B}^T)^{-1}(\mathbf{B}\mathbf{F}\mathbf{B}^T)(\mathbf{B}\mathbf{B}^T)^{-1}. \tag{7}$$

To motivate the approximation X_B^{-1} note that since $\mathbf{B}$ is rectangular there is no simple formula for X in terms of a product of inverses of its factors. On the other hand $\mathbf{B}\mathbf{B}^T$ acts on the pressure space. Thus X_B^{-1} is a plausible approximation of X^{-1}, in that it combines (i) the action of F and (ii) the inverse of $\mathbf{B}\mathbf{B}^T$.

However, numerical experiments with X_B^{-1} indicate that with this choice, the number of iterations degrades considerably as $h \to 0$ and as the Reynolds number Re increases. Kay, Loghin and Wathen [13, 14] considered the Oseen problem (equivalent to deleting the term $\mathbf{u} \cdot \nabla \mathbf{w}$ in (1)) and for this they proposed instead

$$X_F^{-1} = M_p^{-1} F_p A_p^{-1}, \tag{8}$$

where F_p and A_p are discrete versions of $-\epsilon\Delta + \mathbf{w} \cdot \nabla$ and Δ respectively and M_p is the pressure mass matrix. With this choice the preconditioner is observed to be independent of h and has a milder dependence on Re. But the drawback is that (8) requires rediscretisation of the convection-diffusion operator $-\epsilon\Delta + \mathbf{w} \cdot \nabla$ and Laplacian Δ on the pressure space. One of the outcomes of this study is a (parallel) preconditioner analogous to (8), which does not need rediscretisation of any of the operators and uses only submatrices of A. We restrict here to the case of the $Q_2 - Q_1$ finite element discretisation (see, for example [1]) and use a suitable restriction of F_{11} to the pressure space instead of explicitly forming F_p. Since $Q_1 \subset Q_2$ we can define a linear interpolation operator Π from pressure to velocity freedoms by simple injection and this leads to the following approximation of X^{-1}:

$$X_\Pi^{-1} = \Pi^T F_{11} \Pi (\mathbf{B}\mathbf{B}^T)^{-1}. \tag{9}$$

This is another plausible approximation of X^{-1} based on the same discussion as that following equation (7). However realisation of the action of X_Π^{-1} requires only operations with block matrices already present in A. Action of the whole block preconditioning step $(\mathbf{u}, p)^T = P^{-1}(\mathbf{w}, r)^T$ with P defined in (6) is achieved with the following algorithm:

(i) Solve $(\mathbf{B}\mathbf{B}^T)s = r$,

(ii) apply $p = -\Pi^T F_{11} \Pi s$,

(iii) apply $\mathbf{v} = \mathbf{w} - \mathbf{B}^T p$,

(iv) solve $\mathbf{F}\mathbf{u} = \mathbf{v}$.

Note that we assemble the matrix $\mathbf{B}\mathbf{B}^T$ by a sparse matrix multiplication algorithm and the product remains sparse. In our implementation we replace $\mathbf{F}$ in (iv) with a block diagonal approximation $\mathrm{diag}(F_{11}, F_{22})$.

This "block" preconditioner calls for the fast inversion of F_{ii} and $\mathbf{B}\mathbf{B}^T$. These are essentially discrete scalar PDEs and for the parallelisation we use the DOUG technology [16].

The DOUG package

The general parallel iterative solver DOUG (Domain Decomposition on Unstructured Grids, see [16, 17]) for linear symmetric and unsymmetric systems in 2D or 3D has been developed at Bath University since 1996. Until the present work, the code was used for the efficient solution of scalar elliptic finite element problems defined in terms of element stiffness matrices on parallel computers. (See, for example [18].) The next version of the code will handle also systems with assembled matrices and block systems (like in the case of linearised Navier Stokes equations discussed here). Krylov subspace methods like CG, MINRES, BiCGSTAB and (F)GMRES are available with additive Schwarz type preconditioners with overlapping subdomains.

For parallel scalability in the presence of unstructured grids a nonmatching coarse grid technique is used. The coarse grid (if required) is computed automatically and adaptively so that it reflects the qualitative features of the fine grid.

DOUG is written in Fortran77, the parallelisation is based on MPI (Message Passing Interface, see for example [19]) and is written using a master-slave setup. The master node handles input and output and is responsible for the coarse problem solutions. The slaves are responsible for the subdomain solves. With N slaves the computational domain is split into N subdomains using the mesh partitioning software METIS [20]. To minimise computation time and to achieve efficient parallelisation, DOUG chooses automatically a suitable size of actual subdomains. For this purpose, the slaves may perform further mesh partitioning to achieve the desired subdomain size. Subdomain solves are performed exactly using the direct multifrontal solver UMFPACK [21]. The LU-factors are computed once at the beginning of computations and stored for each subproblem so that, during the course of the iterative solution, only the corresponding back-substitution phase needs to be called.

For efficient parallel implementation, computations are desired to overlap with communication. With this in mind, DOUG reorders the freedoms suitably for the matrix-vector operations so that freedoms at overlaps always come first in the freedom arrays. In the block preconditioner case, freedoms are additionally grouped together according to their block-rank. Thus, our parallel matrix-vector multiplication computations start with overlapping freedoms, after which simultaneously overlapped communication and multiplication operations on inner

Prepared using cpeauth.cls

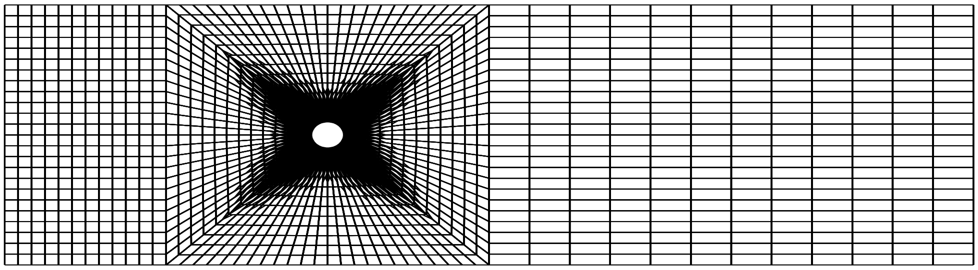

Figure 1. Typical discretisation grid of the flow past a cylinder.

freedoms are performed. For a more detailed overview of the parallel implementation please refer to [17].

Test results: a typical solve with A

The test problem of our choice is the problem of flow past a cylinder. A typical discretisation is given in Fig. 1. This mesh has 1728 elements. When the Navier Stokes equations are approximated using $Q_2 - Q_1$ elements on this mesh, the resulting system has 15286 freedoms. The applied velocity is $(1,0)^T$ on the top, bottom and left hand end of the domain, $(0,0)^T$ on the cylinder and $(\mathbf{u}, p)^T$ satisfies the outflow conditions $\epsilon \frac{\partial u_1}{\partial x} + p = 0 = \frac{\partial u_2}{\partial x}$ at the right hand end. To illustrate the performance of the preconditioners, we present, in Tables 1 and 2, the number of FGMRES iterations for solving (4) with $\sigma = 0$ with preconditioner

$$P = \begin{bmatrix} F_{11} & 0 & B_1^T \\ 0 & F_{22} & B_2^T \\ 0 & 0 & -X \end{bmatrix}$$

where X^{-1} is approximated using either (7) or (9). We employ the stopping criteria that the initial residual norm should be reduced by six orders of magnitude.

In the implementation of each solve with P it is required to implement (subproblem) solves with F_{ii}, $i = 1, 2$ and $\mathbf{B}\mathbf{B}^T$(the latter is done twice in the case of $X^{-1} \approx X_B^{-1}$). We have tested our preconditioners with exact and inexact subproblem solves. In the exact case all the solves with matrices $\mathbf{B}\mathbf{B}^T$, F_{11} and F_{22} are done directly. In the case of inexact subsolvers we apply K iterations of an inner FGMRES solver with two level additive Schwarz preconditioner. $K = 1$ corresponds to simply replacing the exact matrix to be inverted with one application of its additive Schwarz approximate inverse. In general, the larger the value of K the more precisely the subproblems are solved inside the preconditioner.

An example of the mesh partitioned into subdomains is given in Fig. 2 and an example of an adaptively refined coarse grid is given in Fig. 3. The results with $Re = 1.0$ and $Re = 25.0$ are presented in Table 1 and in Table 2 respectively. Note that the iteration counts in the

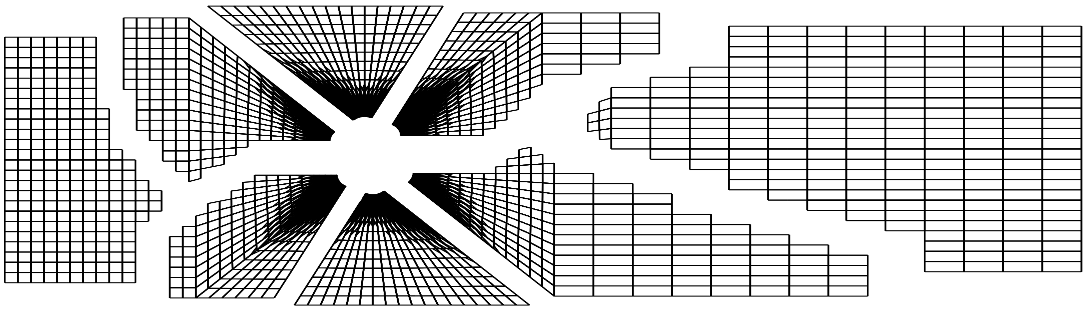

Figure 2. Flow past a cylinder: the grid partitioned into eight subdomains.

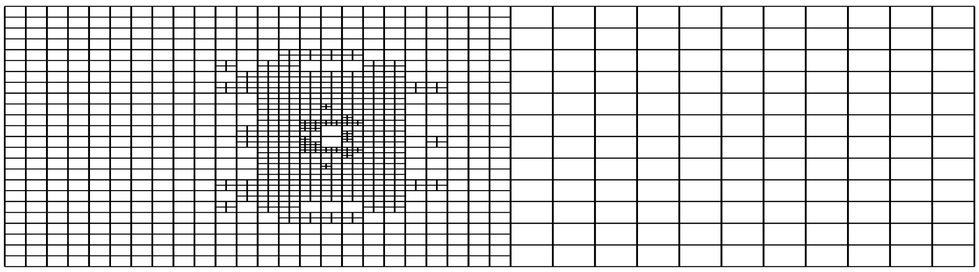

Figure 3. Flow past a cylinder: an adaptively refined coarse grid.

$Re = 1.0$	X_B^{-1}						X_Π^{-1}				
#dof \ K	exact	16	8	4	2	1	exact	8	4	2	1
6734	54	55	60	130	170	133	50	50	53	66	114
27294	87	104	350	>800	>800	234	63	65	78	259	171
61678	116	188	791	-	-	311	73	78	173	781	205
109886	139	352	-	-	-	382	81	91	281	>800	250

Table 1. Number of iterations for exact and inexact solvers with $Re = 1.0$.

$Re = 25.0$	X_B^{-1}						X_Π^{-1}				
#dof \ K	exact	16	8	4	2	1	exact	8	4	2	1
6734	123	126	136	170	338	244	144	150	191	273	295
27294	194	235	529	>800	-	335	133	146	174	275	384
61678	258	346	>800	-	-	475	132	162	173	261	407
109886	309	621	>800	-	-	593	127	153	183	272	459

Table 2. Number of iterations for exact and inexact solvers with $Re = 25.0$.

#slaves	time sec. (# it.)	relative speedup
1	1922.6 (644)	-
2	857.5 (611)	2.24
4	414.1 (578)	4.64
8	250.9 (564)	7.66
12	195.5 (607)	9.73

Table 3. Relative speedups for solving a moderate size problem on a Linux cluster (for $K = 1$).

tables reflect also the exact number of matrix vector operations in FGMRES (since we use a zero initial guess). Generally, the method with X_Π^{-1} converges better. Particularly, with exact subsolves we observe that the preconditioner with X_B^{-1} has some dependence on the meshsize but for X_Π^{-1} with $Re = 1.0$ such dependence is very small and with $Re = 25.0$ does not appear at all. Similar observations appear in [13, 14] for preconditioner (8). Similarly, the preconditioner with X_B^{-1} is much more demanding on the accuracy of the subsystem solves – with the number of degrees of freedom increasing, more subsolve iterations K are needed to improve the iteration counts obtained with $K = 1$. Too few inner solves can damage the convergence. (For the smallest problem, $K \geq 4$ is needed to improve on the result with $K = 1$ and at least $K \simeq 16$ inner iterations are needed for the largest problem.) For X_Π^{-1} with $Re = 1.0$ the same behaviour is observed but is less significant. Finally, in the case of the inexact subsolves, X_Π^{-1} with $Re = 25.0$ gives essentially no increase in iteration counts with mesh refinement for each $K > 1$. This is a particularly good feature for stability calculations as in many problems computations over a large range of Re values are required.

We have tested the parallel efficiency of our code in the case of solving a system with 109886 freedoms and $Re = 25$. We have measured the total time for the solution of the whole problem (including initialisation and data distribution phases). The same problem is solved in the case of one up to eight slaves. Since the DOUG code optimises the choice of subdivisions even for one processor, we can think of the time for one processor as "best" in this computation. The timings for a 1.8 GHz AMD Athlon Linux cluster are presented in Table 3 . We conclude that the algorithm has a good parallel efficiency.

Computation of Eigenvalues

PARPACK and Inexact Inverse Iteration

To compute the dangerous eigenvalues (i.e. those closest to the imaginary axis) of the Navier-Stokes system (1), we consider two algorithms both of which use PARPACK – the parallel version of the ARPACK package – which is based on the Implicitly Restarted Arnoldi Method [2] and is suitable for large scale eigenvalue problems. The software provides the option to compute a few (k) eigenvalues with user specified features of a general $n \times n$ matrix. For example, one can ask that eigenvalues of largest real part or largest magnitude be computed. The storage requirements of the algorithm are of the order of $n * k$ locations. From (4) it follows that for our purposes the matrix under consideration is $(A - \sigma M)^{-1}M$, and we call PARPACK in the shift-invert mode. (P)ARPACK uses a reverse communication interface which in this case requires calling one of the following three operations on each outer iteration step (depending on a PARPACK subroutine output parameter value):

a) $\mathbf{y} = (A - \sigma M)^{-1}M\mathbf{x}$,
b) $\mathbf{y} = (A - \sigma M)^{-1}\mathbf{x}$,
c) $\mathbf{y} = M\mathbf{x}$.

After one of these calls control is looped back to the PARPACK subroutine. When convergence is achieved another PARPACK subroutine is called to extract the corresponding computed eigenvalues. Throughout we used the PARPACK computational mode 'LI' – as we know from some preliminary investigations that the desired eigenvalues have large imaginary part.

Our first strategy is the straightforward application of PARPACK. We input a shift σ, a required relative eigenvalue tolerance, *etol* and a linear system relative tolerance, *tol*, which we take to be of the same order as *etol*.

Our second strategy is to use PARPACK to obtain a coarse approximation to the desired eigenvalue and eigenvector, and then feed these approximations to inexact inverse iteration. So our second strategy is PARPACK with refinement by inverse iteration with a fine relative tolerance for the eigenvalue approximation.

First we give the version of inexact inverse iteration used in our numerical experiments. (Note that throughout this paper, $\|\cdot\| = \|\cdot\|_2$, the usual 2-norm.)

Algorithm 1. Inexact Inverse Iteration
Given $\mathbf{x}^0$, σ^0, τ^0, *etol*
(i) Inexact solve of $(A - \sigma^i M)\mathbf{y}^i = M\mathbf{x}^i$, that is:
$\left\|(A - \sigma^i M)\mathbf{y}^i - M\mathbf{x}^i\right\| \le \tau^i \left\|M\mathbf{x}^i\right\|$.
(ii) Rescale: $\mathbf{x}^{i+1} = \mathbf{y}^i / \left\|\mathbf{y}^i\right\|$.
(iii) Update parameters σ^i and/or τ^i using:
$\sigma^{i+1} = \sigma^i$, or $\sigma^{i+1} = ({\mathbf{x}^{i+1}}^H A\mathbf{x}^{i+1})/({\mathbf{x}^{i+1}}^H M\mathbf{x}^{i+1})$
$\tau^{i+1} = \tau^i$, or $\tau^{i+1} = \left\|(A - \sigma^{i+1}M)\mathbf{x}^{i+1}\right\| / \left|\sigma^{i+1}\right|$.
(iv) Test for convergence:
if $\left\|(A - \sigma^{i+1}M)\mathbf{x}^{i+1}\right\| < etol \cdot \left|\sigma^{i+1}\right|$ stop;
otherwise $i \leftarrow i + 1$ and repeat.

	Strategy 1	Strategy 2	
	PARPACK: $k = 16$ $etol = 10^{-6}$, $tol = 10^{-6}$	PARPACK: $k = 16$ $etol = 0.07$, $tol = 0.05$	Inexact Inverse It^{n} $etol = 10^{-6}$
#solves	161	35	4
# $A\mathbf{x}$ ops.	54728	2663	1062
time (s)	2857	91	206
Tot. time	2857	297	

Table 4. A comparison of matrix operations and times for Strategy 1 and Strategy 2.

In Algorithm 1, we use our preconditioned parallel iterative solver to perform the inexact solve in step (i), and make the choice to update the shift σ^i using the Rayleigh Quotient but keep the value τ^i fixed. The mathematical details of a convergence analysis are given in section *Convergence for Inexact Inverse Iteration*, where this choice of σ^i and τ^i is referred to as "Option 2"; this choice of shift and tolerance is found empirically to be the best from three possible choices.

Example 1. In this example we compare the performance of PARPACK using $etol = 10^{-6}$, $tol = 10^{-6}$ and $k = 16$, against PARPACK using $etol = 0.07$, $tol = 0.05$, and $k = 16$ with estimates for the most dangerous eigenvalue then fed to Option 2 (Variable shift/Fixed Tolerance) of inexact inverse iteration with $etol = 10^{-6}$. Note that in our experiments we found that PARPACK needed a value of k of around 16, otherwise the eigenvalues nearest the imaginary axis were not found. In our experience when using PARPACK it is important not to try to reduce costs by choosing a small value of k. The number of unknowns of the flow past a cylinder problem is 27K and $Re = 25.0$. The value of the shift σ used for both application of PARPACK was taken to be zero.

In Table 4 we compare costs in time and calls to A for the two strategies to produce the required eigenvalue to a relative tolerance of 10^{-6}. Note that the right hand columns give the costs for PARPACK plus a single application of Algorithm 1 to find one complex eigenvalue. We see from the table that Strategy 2 is nearly 10 times faster than Strategy 1. This is probably sufficient for stability assignment where one is merely interested in the sign of the real part of the most dangerous eigenvalue. Clearly Strategy 2 is only worthwhile if a few accurate eigenvalues are sought. We note that if accurate values of the 8 most dangerous complex pairs were required then the costs using Strategy 2 could be estimated at $91 + 8 \times 206 = 1739$secs, almost exactly half the 2857secs required by the straight use of PARPACK for this problem.

These results suggest that when a small number of eigenvalues are needed to a reasonable accuracy then a good strategy is to use PARPACK to obtain rough approximations, and refine these approximations using a version of inexact inverse iteration.

Also relevant to the problem of finding the most dangerous eigenvalue is the relaxation strategy for Arnoldi's method proposed by [15] in which the inexact solves in Arnoldi's method applied to $(A - \sigma M)^{-1}M$ are solved with successively larger tolerances. We have not tested this method.

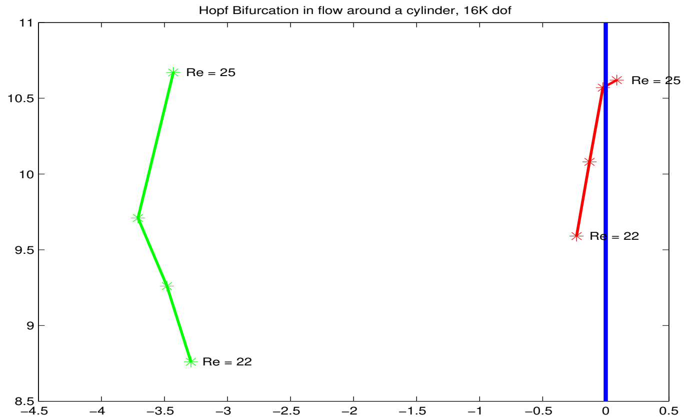

Figure 4. The paths of a few eigenvalues as Re increases, 16K dof.

Example 2. In this example we show the utility of our eigenvalue solvers to analyse the stability of fluid flow past a cylinder. The paths of a few dangerous eigenvalues as Re increases are depicted in Figs. 4 and 5 for two different discretisations. The figures illustrate the necessity to solve the eigenvalue problem with a large number of degrees of freedom, since using 16K dof and $Re = 24$ the most dangerous eigenvalue has negative real part, but for 109K dof the real part has crossed the imaginary axis. A Hopf bifurcation for this system occurs at $23 < Re < 24$. An incorrect stability assignment would result if one only examined the eigenvalues of the small system (Fig. 4).

Convergence for Inexact Inverse Iteration

Here we give a convergence analysis for inexact inverse iteration (see Algorithm 1 above) for the generalised eigenvalue problem

$$A\mathbf{x} = \lambda M\mathbf{x} \tag{10}$$

where A is nonsymmetric and M is symmetric positive definite. Note that λ and $\mathbf{x}$ could be complex.

Assume that $M^{-1}A$ is diagonalisable with eigenvalues λ_j, $j = 1, \ldots, n$, and corresponding right and left eigenvectors $\mathbf{v}_j$ and $M\mathbf{u}_j$ normalised such that, for $j = 1, ..., n$,

$$A\mathbf{v}_j = \lambda_j M\mathbf{v}_j, \;\; \mathbf{u}_j^T A = \lambda_j \mathbf{u}_j^T M, \;\; \|\mathbf{v}_i\| = 1, \;\; \mathbf{u}_j^T M\mathbf{v}_j = \delta_{ij}. \tag{11}$$

We shall assume that λ_1 is the desired eigenvalue, that λ_2 is the eigenvalue closest to λ_1, and that all shifts σ^i satisfy

$$\left|\lambda_1 - \sigma^i\right| < \left|\lambda_2 - \sigma^i\right| < \left|\lambda_j - \sigma^i\right| \text{ for } j = 3, \ldots n. \tag{12}$$

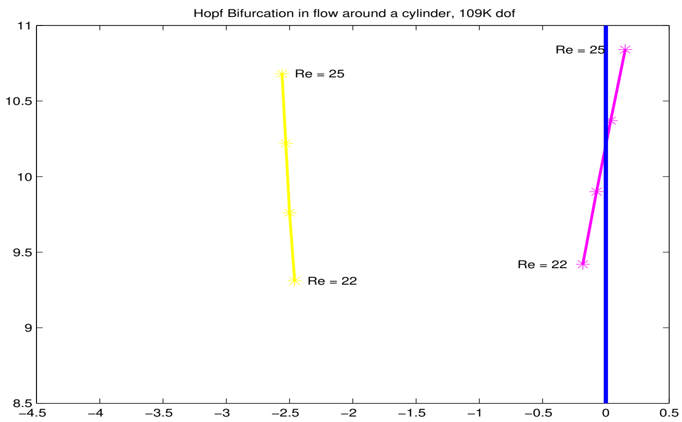

Figure 5. The paths of a few eigenvalues as Re increases, 109K dof.

Then $\mathbf{v}_1$ is the required (right) eigenvector corresponding to λ_1. Following the style in [22] (for symmetric eigenvalue problems) for all i, we write the inexact inverse iterate $\mathbf{x}^i$ as

$$\mathbf{x}^i = c^i \mathbf{v}_1 + s^i \mathbf{w}^i \tag{13}$$

where $\|\mathbf{w}^i\| = 1$ and $\mathbf{w}^i \in \operatorname{span}\{\mathbf{v}_j\}_{j=2}^n$, which implies

$$\mathbf{u}_1^T M \mathbf{w}^i = 0. \tag{14}$$

Here c^i and s^i are complex constants. Note that $|s^i|$ can be thought of as an error indicator, since

$$\|\mathbf{x}^i - c^i \mathbf{v}_1\| = |s^i|$$

showing that $|s^i|$ measures the amount by which $\mathbf{x}^i$ fails to be in the direction $\mathbf{v}_1$. Note also that if $|s^i| \to 0$ as $i \to \infty$, then $|c^i| \to 1$ since by Algorithm 1 (ii) $\|\mathbf{x}^i\| = 1$. The aim of the following analysis is to obtain the convergence result (25) below which relates the ratio $\left|\frac{s^{i+1}}{c^{i+1}}\right|$ to $\left|\frac{s^i}{c^i}\right|$ and from which rates of convergence for different choices of σ^i and τ^i can be deduced.

First note that with the Rayleigh Quotient defined by

$$\rho(\mathbf{x}) := \left(\mathbf{x}^H A \mathbf{x}\right) / \left(\mathbf{x}^H M \mathbf{x}\right) \tag{15}$$

it is readily shown that for small enough $|s^i|$ there are constants C_1 and C_2 such that

$$|\lambda_1 - \rho(\mathbf{x}^i)| \le C_1 |s^i| \tag{16}$$

and

$$\|(A - \rho(\mathbf{x}^i) M)\mathbf{x}^i\| \le C_2 |s^i|. \tag{17}$$

If $\mathbf{y}^i$ is the result of the inexact solve in Algorithm 1 (i), then

$$(A - \sigma^i M)\mathbf{y}^i - M\mathbf{x}^i = \mathbf{r}^i, \tag{18}$$

where the residual $\mathbf{r}^i$ satisfies

$$\left\|\mathbf{r}^i\ \right\| \leq \tau^i \left\|M\mathbf{x}^i\right\|$$

with τ^i as in Algorithm 1.

Analogous to (14), we can decompose $\mathbf{r}^i$ as

$$\mathbf{r}^i = \alpha^i M\mathbf{v}_1 + M\mathbf{d} \tag{19}$$

where $\mathbf{d} \in \operatorname{span}\{\mathbf{v}_j\}_{j=2}^n$. Then $\alpha^i = \mathbf{u}_1^T\mathbf{r}^i$ and $\mathbf{u}_1^T M\mathbf{d}$ =0. Note that

$$\left|\mathbf{u}_1^T\mathbf{r}^i\right| \leq \tau^i \ \text{ and } \ \|\mathbf{d}\| \leq C_3\tau^i. \tag{20}$$

for some constant C_3. Using Algorithm 1 (i), (ii), we can then write (18) as

$$\left\|\mathbf{y}^i\right\|(A - \sigma^i M)\mathbf{x}^{i+1} = M\mathbf{x}^i + \mathbf{r}^i. \tag{21}$$

Inserting (13) on each side of (21) we obtain

$$\left\|\mathbf{y}^i\right\|(A - \sigma^i M)(c^{i+1}\mathbf{v}_1 + s^{i+1}\mathbf{w}^{i+1}) = c^i M\mathbf{v}_1 + s^i M\mathbf{w}^i + \mathbf{r}^i. \tag{22}$$

Applying $\mathbf{u}_1^T$ on the left gives, using (14) and (19),

$$\left\|\mathbf{y}^i\right\|(\lambda_1 - \sigma^i)c^{i+1} = c^i + \mathbf{u}_1^T\mathbf{r}^i. \tag{23}$$

On the other hand the component of (22) in $\{\mathbf{u}_1\}^\perp$ is

$$\left\|\mathbf{y}^i\right\|(A - \sigma^i M)s^{i+1}\mathbf{w}^{i+1} = s^i M\mathbf{w}^i + M\mathbf{d}.$$

Now note that $(A - \sigma^i M)$ is invertible when considered as a mapping from $\operatorname{span}\{\mathbf{v}\}_{j=2}^n$ into $\{\mathbf{u}_1\}^\perp = \operatorname{span}\{M\mathbf{v}_j\}_{j=2}^n$. Moreover, recalling (12) we have, on $\operatorname{span}\{\mathbf{v}\}_{j=2}^n$, $\left\|(A - \sigma^i M)^{-1}M\right\| \leq C_4\left|\lambda_2 - \sigma^i\right|^{-1}$ for some constant C_4. Hence

$$\left\|\mathbf{y}^i\right\| s^{i+1}\mathbf{w}^{i+1} = (A - \sigma^i M)^{-1}M(s^i\mathbf{w}^i + \mathbf{d})$$

and, taking norms,

$$\left\|\mathbf{y}^i\right\| \left|s^{i+1}\right| \leq \frac{C_4}{\left|\lambda_2 - \sigma^i\right|}(\left|s^i\right| + \|\mathbf{d}\|). \tag{24}$$

We now combine (23) and (24) and using (20) to obtain, for small enough τ^i,

$$\left|\frac{s^{i+1}}{c^{i+1}}\right| \leq C_4\left|\frac{\lambda_1 - \sigma^i}{\lambda_2 - \sigma^i}\right| \left\{\frac{\left|s^i\right| + C_3\tau^i}{\left|c^i\right| - \tau^i}\right\}. \tag{25}$$

This equation provides the basis for our understanding of the convergence of inverse iteration. Note if $\tau^i = 0$, that is we solve linear systems exactly, we obtain the standard convergence result. For example, if A is symmetric, $C_4 = 1$, and if $M = I$ we recover equation (4.9) in [22].

Result (25) can be used to provide theorems for the convergence of various different strategies. Here we consider three different options for choosing σ^i and τ^i in Algorithm 1.

Option 1 (Fixed shift/Variable tolerance). Choose $\sigma^i = \sigma$, a constant, and τ^i as the relative eigenvalue residual, that is,

$$\tau^i = \left\|(A - \sigma^i M)\mathbf{x}^i\right\| / \left|\sigma^i\right| \leq C_2 \left|s^i\right| / \left|\sigma^i\right|$$

(see (17)). Inserting into the right-hand side of (25), we see that we have convergence if $\left|s^i\right| \to 0$. So this choice involves reducing τ^i as the outer iteration proceeds. (Note that an easy consequence of (25) is that there is little gain in choosing $\tau^i = o(\left|s^i\right|)$). With these choices for τ^i and σ^i we readily obtain that for small enough $\left|s^i\right|$ there is a scalar $\alpha_1 < 1$ such that

$$\left|\frac{s^{i+1}}{c^{i+1}}\right| \leq \alpha_1 \left|\frac{s^i}{c^i}\right| .$$

Hence with this choice of σ^i and τ^i Algorithm 1 will give *linear* convergence.

Option 2 (Variable shift/Fixed tolerance). If we choose $\tau^i = \tau$; a constant, and $\sigma^i = \rho(\mathbf{x}^i)$, the Rayleigh Quotient as defined in (15), then inserting (16) into the right-hand side of (25) we readily obtain that, for small enough $\left|s^i\right|$ and τ, there is a scalar $\alpha_2 < 1$ such that

$$\left|\frac{s^{i+1}}{c^{i+1}}\right| \leq \alpha_2 \left|\frac{s^i}{c^i}\right| .$$

Hence Algorithm 1 with this choice of σ^i and τ^i will give *linear* convergence.

Option 3 (Variable shift/Variable tolerance). Here we choose σ^i as the Rayleigh Quotient as in Option 2, and τ^i as the relative eigenvalue residual as in Option 1. It is readily shown that in this case

$$\left|\frac{s^{i+1}}{c^{i+1}}\right| = O\left(\left|\frac{s^i}{c^i}\right|^2\right)$$

so that we achieve *quadratic* convergence.

A detailed account of the convergence of inexact inverse iteration for symmetric eigenvalue problems using MINRES as the inexact solver is in [23] and [24], which also contains an analysis of the nonsymmetric eigenvalue problem.

Example 3. Here we compare the performances of the three options for inverse iteration on the matrix with 27K dof for $Re = 25$. In all cases the starting values for σ^0 and $\mathbf{x}^0$ were these from PARPACK with $etol = 0.07$ and $tol = 0.05$ used in Example 1 above. All three options are methods involving “inner/outer” iterations, with the inner iterations being concerned with the inexact solve and the outer iteration described by the index i in Algorithm 1.

In Table 5 we show the total number of (matrix)×(vector) operations, denoted by $A\mathbf{x}$, for each inner iteration. For Option 1 we see that the number of $A\mathbf{x}$ operations increases as the outer iteration proceeds which is to be expected since the relative tolerance τ^i decreases. For Option 2 we again see an increase in $A\mathbf{x}$ operations as the outer iteration proceeds since the shift is tending to the desired eigenvalue. However, as is seen in the table, the increase is modest. This is consistent with some more detailed analysis done at Bath on symmetric eigenvalue problems that indicates it is in fact advantageous to use a technique where the shift

outer iteration	Option 1 $\sigma^i = \sigma^0$ $\tau^i = \min\{\delta^i, 0.1\}$	Option 2 $\sigma^i = \rho(\mathbf{x}^i)$ $\tau^i = 0.1$	Option 3 $\sigma^i = \rho(\mathbf{x}^i)$ $\tau^i = \min\{\delta^i, 0.1\}$
1	287	192	281
2	392	238	397
3	438	276	499
4	484	356	420
5	521		
tot.$A\mathbf{x}$-ops.	2122	1062	1597
tot. time (s)	564	206	406

Table 5. A comparison of matrix-vector multiplications and times for the three options of inexact inverse iteration. (In Option 1 and Option 3 we denote $\delta^i = \left\| \left(A - \rho(\mathbf{x}^i)M\right)\mathbf{x}^i \right\|$.)

tends to the desired eigenvalue. This analysis will be reported in [23] and [24]. We observe that Option 2 outperforms Option 1 significantly. This behaviour is observed in other experiments and is why we chose Option 2 in experiment 1. For this example there is clearly no gain in using Option 3 which in theory has quadratic rather than linear convergence. Though there are fewer outer iterations the combination of the lower tolerance $\tau^i$and shift converging to the desired eigenvalue makes the cost of each inner iteration more expensive than for Option 2 and both methods take the same number of outer iterations. However, in other problems we have observed that Option 3 is sometimes superior to Option 2 particularly when very high accuracy is required and when the quadratic convergence produces a significantly smaller number of outer iterations.

Remark. Note that we presented the convergence theory for inexact inverse iteration for $A\mathbf{x} = \lambda M\mathbf{x}$ where M is symmetric positive definite. For Navier-Stokes problems the M in (3) is only positive semidefinite. However, using the decomposition in Section 2 of [25] it is possible to show that (3) is equivalent to a *reduced* generalised eigenvalue problem $\hat{K}\hat{\mathbf{x}} = \lambda\hat{M}\hat{\mathbf{x}}$ where $\hat{M}$ is symmetric positive definite. Under the assumption that the technique used for the inexact solves of (4) retains the discrete incompressibility condition then there is also an equivalence between the inexact inverse iteration theory for (3) and that for the reduced problem. Hence in this case the theory in section *Convergence for Inexact Inverse Iteration* can be applied to the discretised Navier-Stokes problem.

Conclusions

In this paper we have discussed some possibilities of building up parallel preconditioners for the linearised Navier-Stokes problem. We investigated two types of preconditioners based on Schur complement techniques. The preconditioners are built up using only existing submatrices of the original problem stiffness matrix and are relatively cheap to construct. For parallelisation the Domain Decomposition technique is used and for the implementation the DOUG package is extended to block systems. The parallel preconditioner gives relatively good speedup results.

We also discussed two strategies for the computation of a small number of eigenvalues nearest the imaginary axis of the linearised Navier-Stokes equations. Both strategies required the application of our parallel solver to shifted linear systems. It was found that an efficient approach is to use Arnoldi's method with a coarse tolerance to obtain a rough approximation to the eigenvalue nearest the imaginary axis and then apply an inexact inverse iteration algorithm to obtain the accuracy desired. The design of such methods is subject to further research. Further investigation is also needed to extend the given technique to other types of finite element discretisations.

REFERENCES

1. Gresho PM, Sani RL. *Incompressible Flow and the Finite Element Method. Advection-diffusion and isothermal laminer flow.* John Wiley & Sons, 1998.
2. Lehoucq RB, Sorensen DC, Yang C, *ARPACK user's guide: Solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods.* http://www.caam.rice.edu/software/ARPACK [12 December 2002].
3. Smit P, Paardekooper M. The effects of inexact solvers in algorithms for symmetric eigenvalue problems*Linear Algebra Appl.* 1999; **287**:337-357.
4. Golub GH, Ye Q. Inexact inverse iteration for generalised eigenvalue problems. *BIT* 2000; **40**:671-684.
5. Simoncini V, Eldén L. Inexact Rayleigh Quotient-type methods for eigenvalue computations*BIT* 2002; **42**:159-182.
6. Sleijpen G, van der Vorst H. Jacobi-Davidson Methods. In *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide.* Bai Z, Demmel J, Dongarra J, Ruhe A, van der Vorst H (eds.). SIAM; Philadelphia, 2000; 88-105.
7. Cliffe KA, Garrett TJ, and Spence A, Eigenvalues of the discretized Navier-Stokes equations with application to the detection of Hopf bifurcations *Advances in Computational Mathematics* 1993; **1**:337-356.
8. ENTWIFE: http://www.entwife.com [12 December 2002].
9. Graham IG, Spence A, Vainikko E. Parallel iterative methods for Navier-Stokes equations and application to stability assessment. In *EupoPar2002 Parallel Processing,* Monien B, Feldmann R (eds.). Lecture Notes in Computer Science 2400, Springer-Verlag, Berlin, 2002:705 - 714.
10. Saad Y. A flexible inner-outer preconditioned GMRES algorithm *SIAM J. Sci. Comput.* 1993; **14**:461-469.
11. Elman HC, Silvester DJ. Fast nonsymmetric iterations and preconditioning for Navier-Stokes equations*SIAM J. Sci. Comp.* 1996; **17**: 33-46.
12. Elman HC. Preconditioning for the steady-state Navier-Stokes equations with low viscosity*SIAM J. Sci. Comp.* 1999; **20**:1299-1316.
13. Kay D, Loghin D. A Green's function preconditioner for the steady-state Navier-Stokes equations *Oxford Numerical Analysis Group Research Report* 1999; NA-99/06.
14. Kay D, Loghin D, Wathen A. A preconditioner for the steady-state Navier-Stokes equations*SIAM J. Sci. Computing* , to appear.
15. Bouras A, Frassé V. A relaxation strategy for the Arnoldi method in eigenproblems. *CERFACS Report.* 2000; TR/PA/00/16.
16. DOUG: http://www.maths.bath.ac.uk/~parsoft [12 December 2002].
17. Hagger MJ. Automatic domain decomposition on unstructured grids DOUG*Advances in Computational Mathematics* 1998; **9**:281-310.
18. Cliffe KA, Graham IG, Scheichl R, Stals L. Parallel computation of flow in heterogeneous media modelled by mixed finite elements. *Journal of Computational Physics*2000; **164**:258-282.
19. Gropp W, Lusk E, Skjellum A. *Using MPI - Portable Parallel Programming with the Message Passing Interface,* MIT Press, 2nd Edition, 1999.
20. Karypis G, Kumar V.METIS: Unstructured graph partitioning and sparse matrix ordering system Department of Computer Science. *University of Minnesota Report* 1995 Aug, Minneapolis.
21. Davis TA, Duff IS.A combined unifrontal/multifrontal method for unsymmetric sparse matrices *ACM Transactions on Mathematical Software*1999; **25**(1):1-19.
22. B. N. Parlett. *The Symmetric Eigenvalue Problem.* SIAM, Philadelphia, 1998.

23. Berns-Müller J, Graham IG, Spence A. *Inexact Inverse Iteration for the symmetric eigenvalue problem* (in preparation).
24. Berns-Müller J. *Inexact Inverse Iteration and Galerkin-Krylov Methods* PhD thesis, University of Bath (in preparation).
25. Cliffe KA, Garrett TJ, Spence A. Eigenvalues of Block Matrices arising from problems in Fluid Mechanics *SIAM J. Matrix Analysis*1994; **15**:1310-1318.