SANDIA REPORT

SAND2006-4466
Unlimited Release
Printed July 2006

The Effect of Boundary Conditions within Pressure Convection–Diffusion Preconditioners †

Victoria E. Howle, Jacob Schroder, Ray Tuminaro

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation,
a Lockheed Martin Company, for the United States Department of Energy's
National Nuclear Security Administration under Contract DE-AC04-94-AL85000.

Approved for public release; further dissemination unlimited.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2006-4466
Unlimited Release
Printed July 2006

The Effect of Boundary Conditions within Pressure Convection–Diffusion Preconditioners [‡]

Victoria E. Howle [§] Jacob Schroder [¶] Ray Tuminaro [‖]

Abstract

We explore choices of boundary conditions within pressure convection–diffusion preconditioners for the incompressible Navier–Stokes equations. While these methods have been shown to be efficient, choosing the proper boundary conditions for the preconditioning operator is not well understood. In this paper, we first explore the effect of having "ideal" boundary conditions within the preconditioner. While not computationally feasible, the ideal boundary condition results highlight the importance of suitable boundary conditions. The remainder of the paper explores somewhat more practical approximations to the ideal conditions based on ILU factorizations and probing [5].

[*]This work was supported by Sandia National Labs. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

[†]This work was supported by Sandia National Labs. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

[‡]This work was supported by Sandia National Labs. Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

[§]Sandia National Laboratories, PO Box 969, MS 9159, Livermore, CA 94551, vehowle@sandia.gov.

[¶]Department of Computer Science, University of Illinois at Urbana-Champaign, 201 N Goodwin Ave, Urbana, IL 61801, jschrod@uiuc.edu.

[‖]Sandia National Laboratories, PO Box 969, MS 9159, Livermore, CA 94551, rstumin@sandia.gov.

1 Introduction

We explore choices of boundary conditions within pressure convection–diffusion preconditioners for the incompressible Navier–Stokes equations. While these methods have been shown to be efficient, choosing the proper boundary conditions for the preconditioning operator is not well understood. In this paper, we first explore the effect of having "ideal" boundary conditions within the preconditioner. While not computationally feasible, the ideal boundary condition results highlight the importance of suitable boundary conditions. The remainder of the paper explores somewhat more practical approximations to the ideal conditions based on ILU factorizations and probing [5].

Consider the Navier–Stokes equations

$$\begin{aligned} \alpha \mathbf{u}_t - \nu \nabla^2 \mathbf{u} + (\mathbf{u} \cdot \operatorname{grad}) \mathbf{u} + \operatorname{grad} p &= \mathbf{f} \\ -\operatorname{div} \mathbf{u} &= 0 \end{aligned} \tag{1}$$

on $\Omega \subset \mathbb{R}^d$, $d = 2$ or 3. Here, $\mathbf{u}$ is the d-dimensional velocity field, which is assumed to satisfy suitable boundary conditions on $\partial\Omega$, p is the pressure, and ν is the kinematic viscosity, which is inversely proportional to the Reynolds number. The value $\alpha = 0$ corresponds to the steady-state problem and $\alpha = 1$ to the transient case. Linearization and discretization of (1) by finite elements, finite differences, or finite volumes lead to a sequence of linear systems of equations of the form

$$\begin{bmatrix} F & B^T \\ B & -\frac{1}{\nu} C \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ p \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ g \end{bmatrix}. \tag{2}$$

These systems, which are the focus of this paper, must be solved at each step of a nonlinear (Picard or Newton) iteration or at each time step. Here, B and B^T are matrices corresponding to discrete divergence and gradient operators, respectively, and F operates on the discrete velocity space. This paper only considers $C = 0$ corresponding to *div-stable* discretizations (see, e.g., [1]), which satisfy an inf-sup condition.

In recent years, there has been considerable activity in the development of efficient iterative methods for the numerical solution of the stationary and fully-implicit versions of this problem. These are based on new preconditioning methods derived from the structure of the linearized discrete problem given in (2). A complete overview of the ideas under consideration can be found in the monograph [3]. The key to attaining fast convergence lies with the effective approximation of the inverse of the Schur complement operator

$$S = BF^{-1}B^T + \tfrac{1}{\nu} C, \tag{3}$$

which is obtained by algebraically eliminating the velocities from the system. With $C = 0$,

$$S = BF^{-1}B^T. \tag{4}$$

One approach of interest is the *pressure convection–diffusion* preconditioner proposed by Kay, Loghin, and Wathen [4] and Silvester, Elman, Kay and Wathen [6]. In this method, the Schur complement matrix is approximated as

$$S \approx M_S = A_p F_p^{-1} Q, \tag{5}$$

where Q is the pressure mass matrix associated with the pressure discretization (or a spectrally equivalent approximation to it), and A_p and F_p are discrete Laplace and convection–diffusion operators defined on the pressure space. Although effective for solving the system (2), this method has the drawback of requiring users to provide the discrete operators A_p and F_p.

It is fairly well understood how to compute a suitable F_p and A_p operator within the domain interior. However, the situation is less clear for boundary conditions. Currently, there is an accumulated experimental knowledge of what boundary condition setup works best with a specific type of problem. For instance with a backward facing step, the inflow boundary conditions for both F_p and A_p should be Dirichlet, while the other sides should be set to Neumann. As will be demonstrated, other boundary condition setups can result in very

poor convergence. But, there is no general understanding of how to compute F_p and A_p such that the boundary condition setup, i.e., what boundaries are Neumann or Dirichlet, does not strongly affect convergence. In fact, we suspect that problems associated with F_p and A_p at boundaries sometimes cause mesh dependent convergence.

The general idea of this paper is to manipulate (5) so that an expression is obtained for updating either F_p or A_p at the boundary. These updates will occur after an initial F_p and A_p have been formed with somewhat naïve boundary conditions. Specifically, we consider updates based on the following row-based formulas:

$$A_p(bcs,:) = S(bcs,:)(Q^{-1}F_p) \tag{6}$$

and

$$F_p(bcs,:) = Q(bcs,:)(S^{-1}A_p)\,, \tag{7}$$

where $S = BF^{-1}B^T$ and *bcs* refers to the set of all of the boundary rows. Column-based analogs to (6) and (7) can also be used. These column-based formulas generally give similar convergence rates to the row-based methods and may be more appropriate within practical code. In particular, the column-based updates are more efficient because the first matrix operations are at the boundary columns of the right-most operand and this reduces the size and cost of all the subsequent matrix operations significantly.

While computations of (6) and (7) are expensive, we first use these "ideal" boundary conditions to study their effect on numerical convergence. Once this effect is established, we explore more computationally tractable approximations to (6) and (7) based on ILU factorization and probing.

While our results are for a particular problem where suitable boundary conditions are already known, we hope to gain insight into how to develop a more general pressure convection–diffusion preconditioning scheme that works well in more complex situations, where it is less clear how to choose the boundaries.

2 Results

2.1 Test Problem

Our test problem is a backward facing step generated by the IFISS 2.0 package for MATLAB by Silvester, Elman and Ramage [7]. A typical solution is shown in Figure 1.

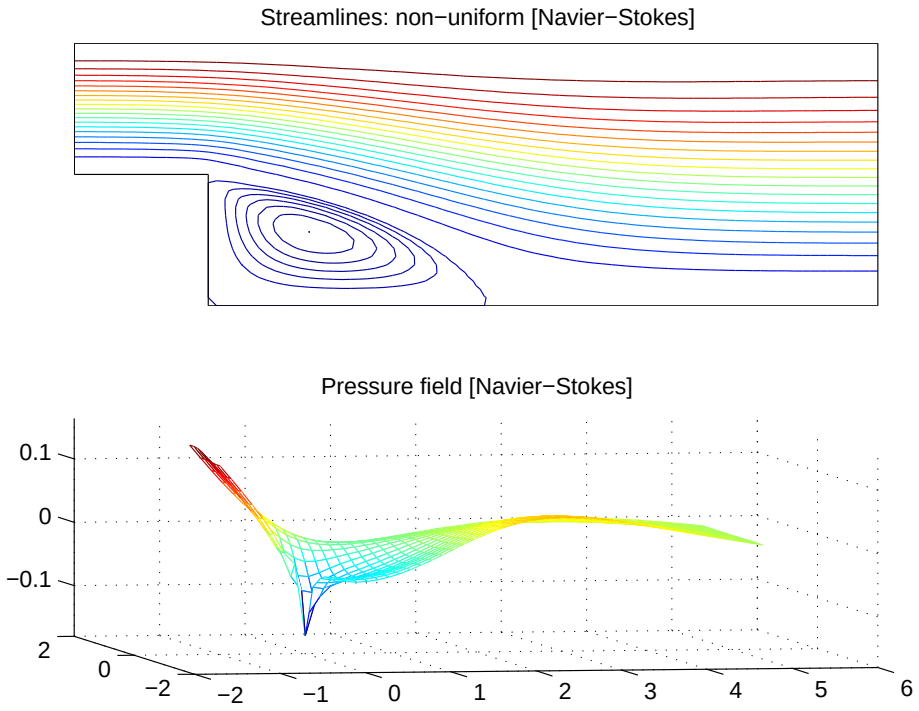

Figure 1. Typical Grid 5 Solution with $\mathfrak{Re} = 100$

The problem setup has the following the parameters:

1. The domain consists of a rectangle ([-1, 5] $\times$ [-1, 1]) with a square removed from the lower left corner.
2. Grid Parameters used were 4, 5 and sometimes 6, depending on the problem's computational load. See Table 1 for grid dimensions.

	Grid 4	Grid 5	Grid 6
Pressure Space	24×8	48×16	96×32
Velocity Space	48×16	96×32	192×64

Table 1. Test Problem Grid Sizes

3. The following Reynolds numbers are considered: $\mathfrak{Re}$ = 10, 100, 200. The Reynolds number is related to the IFISS viscosity parameter, ν, such that $\nu = \frac{2}{\mathfrak{Re}}$.

4. A Q2-Q1 discretization for the velocity - pressure spaces is used.

5. The system solved by GMRES is the final linear system generated by Picard iterations with a nonlinear tolerance of 10^{-5} and a maximum iteration count of 9.

All code and pseudo-code that follows is in MATLAB form.

2.2 Testing Strategy

The IFISS package initially builds F_p and A_p with Neumann boundary conditions on all sides. In a second stage, IFISS normally then modifies the inflow boundary conditions for A_p and F_p so that they correspond to Dirichlet boundaries. For our overall testing strategy, we modify IFISS to experiment with different combinations of Dirichlet boundary conditions on the top, bottom, inflow and outflow sides of the domain. Our intention here is to mimic situations where it is less clear how to choose boundary conditions. We then measure the effect of these changes on convergence with GMRES while using different preconditioning strategies to update the rows that correspond to all the boundary points in the F_p or A_p matrices. Our goal is to find a preconditioner that is mesh independent, insensitive to the initial boundary condition setup, and still yields good convergence.

We define "inflow only Dirichlet" to mean that the inflow side of the domain is set to Dirichlet while all the other sides of the domain are set to Neumann. "Outflow only Dirichlet" is defined analogously. We present in this paper convergence data for only these two experiment types, although 8 combinations of Dirichlet boundary conditions were tried for each preconditioning strategy. These combinations correspond to the following set of Dirichlet boundary conditions:

1. Inflow
2. Outflow
3. Top
4. Bottom
5. Inflow and outflow
6. Inflow and bottom
7. Inflow and top
8. Inflow, top, and bottom

The data for inflow only Dirichlet and outflow only Dirichlet captures the range of performance observed for all 8 combinations. Inflow only Dirichlet is generally the best and is considered the correct boundary condition setup based on accumulated experimental knowledge within the pressure convection–diffusion community. Outflow only Dirichlet, on the other hand, is generally the worst boundary condition setup. The convergence rate of the other combinations generally fell in between these two boundary condition setups.

2.3 IFISS Data without A_p and F_p Updates

In Table 2, GMRES (unrestarted) iterations are given for the standard IFISS code applied to the backward facing step. In Table 3, we make one modification to IFISS so that the outflow boundary conditions are changed to Dirichlet and the inflow conditions are Neumann. Note that 500+ refers to the fact that the experiment did not converge after 500 GMRES iterations.

This data is not mesh independent, even for Table 2. Also as expected, inflow only Dirichlet was the best performer, and outflow only Dirichlet was the worst. The large difference in convergence rates; however, is surprising and disturbing. Obviously, how one chooses the boundary conditions is very important. This large

Grid Param	$\mathfrak{Re}$ = 10	100	200
4	22	31	43
5	25	33	41
6	30	42	47

Table 2. Inflow only Dirichlet, GMRES iterations

Grid Param	$\mathfrak{Re}$ = 10	100	200
4	36	59	79
5	44	500+	500+
6	54	500+	500+

Table 3. Outflow only Dirichlet, GMRES iterations

disparity argues strongly for the development of a general preconditioning strategy that eliminates the need for an accumulated experimental knowledge of what boundary condition setup works best for every specific problem type.

2.4 Concept Testing

Our concept testing consisted of explicitly forming the exact Schur complement and then using the Schur complement in the update for F_p or A_p's rows at all of the boundaries. We did this for only grid sizes 4 and 5, because this operation is too expensive for grid size 6. The F_p update is especially expensive because an LU factorization of the Schur complement must be formed and then the L and U factors are used to do a forward-solve and back-solve with the columns of A_p. While these experiments are much too expensive to be practical methods, they do test in an exact manner what we will later want to do in an approximate fashion.

Table 4 and Table 5 present the data for updating F_p based on (7). Table 6 and Table 7 give data for updating A_p based on (6).

Grid Param	$\mathfrak{Re}$ = 10	100	200
4	12	23	37
5	14	21	34

Table 4. Inflow only Dirichlet with F_p update, GMRES iterations

Grid Param	$\mathfrak{Re}$ = 10	100	200
4	12	22	39
5	14	20	32

Table 5. Outflow only Dirichlet with F_p update, GMRES iterations

Grid Param	$\mathfrak{Re} = 10$	100	200
4	14	30	51
5	15	25	41

Table 6. Inflow only Dirichlet with A_p update, GMRES iterations

Grid Param	$\mathfrak{Re} = 10$	100	200
4	21	46	70
5	23	500+	164

Table 7. Outflow only Dirichlet with A_p update, GMRES iterations

The most important observation is that the F_p row-update exhibits grid independence and excellent convergence in both cases. That is, a proper choice of boundary conditions appears to fix a loss of mesh independence that has been observed for pressure convection–diffusion preconditioners on the backward facing step. Further, the initial placement of Dirichlet boundaries within the A_p and F_p operators is less critical, if a suitable F_p update algorithm is used.

The situation for the A_p updates is less clear. Table 6 seems more mesh independent than either Table 2 or Table 3. However, Table 7, while outperforming Table 3, still shows a marked lack of mesh independence and very slow convergence for $\mathfrak{Re} = 100, 200$. Overall, the initial choice of Dirichlet boundary conditions is still very important in the A_p update case. While we are not completely sure why the A_p update results in slower convergence than the F_p update, it is worth noting one major difference between the A_p and F_p updates. In particular, while we update A_p, we use A_p^{-1} within the preconditioner. Thus, modifications to A_p's boundaries have a more global effect and must be performed carefully.

Since one of our main goals is to develop a preconditioning strategy that does not require any previous experimental knowledge of what boundary condition setup works best, we decided not to investigate A_p-based preconditioning strategies further. All subsequent preconditioning strategies are meant to be approximations of the exact solves and exact Schur complements in (7). Unfortunately the F_p updates are inherently much more expensive than the A_p updates, as they require the multiplication by the inverse Schur complement.

2.5 Employing Structured Probing and ILU to Update F_p

The exact application of (7) requires first a factorization of F followed by the computation of the exact Schur complement and then finally a factorization of the Schur complement. There are several possibilities for trying to approximate (7) inexpensively. In this paper, we explore two replacements for the inverses in (7) using ILU factorizations and probing ideas. We point out that there are several other possible replacements, including using simple iterative methods, coarse grid approximations or sparse approximate inverses.

Our general strategy for updating F_p is to first approximate S by some $\widetilde{S}$ and then to further approximate $\widetilde{S}^{-1}$. We found the most effective approximations of F_p updates were accomplished by using

1. The structured probing algorithm of [5] to approximate the Schur complement followed by using ILU($\widetilde{S}$) to approximate S^{-1}. ILU was also used with good results in [5] to approximate an inverse Schur complement produced by probing.
2. ILU(F) to approximate the Schur Complement $\widetilde{S}$ followed by using ILU($\widetilde{S}$) to approximate S^{-1}.

To test the potential effectiveness of approximating F_p, we initially tried sparsifying the "ideal" F_p by dropping values below a certain threshold. This was done such that the number of non-zeros in F_p was reduced by a factor of 5 to 70, and this lead to encouraging results. A reduction factor of 3-5 increases convergence by generally less than 5 iterations. A reduction factor of around 15 generally almost doubles the convergence rate, and a reduction factor of around 70 essentially causes a loss of convergence.

2.5.1 Updating F_p with a Probed S and ILU

The probing scheme used to calculate S is the structured probing algorithm of [5]. The idea of probing is to reconstruct an operator by repeatedly applying it to a carefully chosen sequence of vectors containing only 0's and 1's. For example, a simple probing method can be used to reconstruct a tridiagonal matrix exactly. This is illustrated by the following example

$$\begin{bmatrix} 1 & 2 & & & \\ 3 & 4 & 5 & & \\ & 6 & 7 & 8 & \\ & & 9 & 10 & 11 \\ & & & 12 & 13 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 4 & 5 \\ 8 & 6 & 7 \\ 10 & 11 & 9 \\ 12 & 13 & 0 \end{bmatrix}. \tag{8}$$

The structured probing algorithm of [5] is a more sophisticated extension of this idea that involves doing a graph coloring on a desired sparsity structure and using that coloring to choose probing vectors such that a matrix with the desired sparsity structure would be reproduced exactly. The carefully chosen probing vectors are then applied to a matrix-vector multiply routine to generate the approximate matrix. The structured probing code used in these experiments implements only one graph coloring algorithm, the greedy-sequential coloring algorithm.

To call the probing routine, we do the following:

- Form the LU factors *[L,U] = lu(F)* once.
- Provide a sparsity pattern which is used to generate a graph coloring and the probing vectors. This sparsity pattern is chosen to be identical to the A_p sparsity pattern provided by IFISS before the rows corresponding to the Dirichlet boundary conditions are zeroed.
- Provide a matrix-vector multiply function, *mat-vec*, that calculates $B(F^{-1}(B^T probe))$ for each probing vector, *probe*. This function is used by the probing software to build the approximate S after a coloring has been computed for the sparsity structure and the probing vectors have been defined from the coloring.

Table 8 and Table 9 contain GMRES iterations using F_p updates of the form:

$$\begin{aligned} &S = probing(mat\text{-}vec, SparsityStruct = A_p); \\ &[L,U] = luinc(S, optns); \qquad\qquad \%compute\ ILU \\ &Fp(bcs,:) = Q(bcs,:) * (U \backslash (L \backslash A_p)); \end{aligned} \tag{9}$$

where *optns* is *optns.droptol = 1e-2; optns.udiag = 1;* [1]. *Luinc('0')*, which performs ILU(0), does not work in this case because 0's appear on the diagonal, so these options have to be employed to replace 0's on the diagonal with *droptol*. *droptol* is also used by *luinc* as a drop tolerance. Using *optns* resulted in approximately the same number of non-zeros as using ILU(0).

The F_p row-update shows mesh independence at $\mathfrak{Re} = 100, 200$, but not at $\mathfrak{Re} = 10$. Encouragingly though, the convergence rate goes down significantly for $\mathfrak{Re} = 200$ with increasing grid size. Possibly, the

[1] The *udiag* option forces zeros on the diagonal to be replaced with optns.droptol.

Grid Param	$\mathfrak{Re}$ = 10	100	200
4	20	38	77
5	23	32	70
6	27	30	44

Table 8. Inflow only Dirichlet with F_p update, GMRES iterations

Grid Param	$\mathfrak{Re}$ = 10	100	200
4	25	50	82
5	31	41	72
6	34	50	65

Table 9. Outflow only Dirichlet with F_p update, GMRES iterations

physics is better captured numerically on the finer mesh. Overall, the convergence rates are good but not nearly as good as the "ideal" F_p updates, and the choice of boundary conditions had only a modest effect on convergence. In this case, updating F_p's rows did perform significantly better than updating columns.

We also did experiments where the *mat-vec* routine used by the probing software calculated F^{-1} using *luinc('0')* instead of *lu()*. This approximation had little effect on convergence rates and made the probing step noticeably quicker.

2.5.2 Updating F_p with ILU only

Another promising method of approximating the F_p updates is by using ILU twice, as shown below.

$$\begin{aligned} &[L,U] = luinc(F,\text{'0'}); && \%compute\ ILU \\ &S = B*(U\backslash(L\backslash B^T)); \\ &[L,U] = luinc(S,\text{'0'}); && \%compute\ ILU \\ &Fp(bcs,:) = Q(bcs,:)*(U\backslash(L\backslash A_p)); \end{aligned} \tag{10}$$

where $luinc(\text{'0'})$ refers to the MATLAB ILU(0) factorization.[2] Table 10 and Table 11 give the corresponding number of GMRES iterations. When comparing these with Table 4 and Table 5, it is apparent that the use of ILU does not seriously degrade convergence.

Grid Param	$\mathfrak{Re}$ = 10	100	200
4	14	26	40
5	16	24	33
6	20	24	31

Table 10. Inflow only Dirichlet with F_p update, GMRES iterations

The results exhibit grid independence, virtually no variability due to boundary condition choices and excellent convergence rates. Yet, we will see in 2.5.3 that this method is much more expensive than probing.

[2] Using ILU with *optns* as in (9) gives nearly identical convergence.

Grid Param	$\mathfrak{Re} = 10$	100	200
4	15	26	44
5	17	27	38
6	21	32	39

Table 11. Outflow only Dirichlet with F_p update, GMRES iterations

2.5.3 Efficiency

In order to better determine the feasibility of the methods outlined in 2.5.1 and 2.5.2, a modest attempt is made to examine the computational cost of their current form in MATLAB. This cost analysis could, however, be mitigated by the possibility that the F_p updates might not need to be done at every nonlinear step. Computational cost is measured by

- Timing the computation of S using probing or ILU with the *tic\toc* calls in MATLAB.
- An examination of the computational bounds on calculating S with probing or ILU.
- Recording the number of non-zeros in S from probing or ILU in order to gauge the cost of the *luinc(S)* call prior to updating F_P.

Timing Table 12 contains average times, in seconds, to calculate S using probing, *luinc('0')*, and *luinc(optns)*. The times for *luinc()* also reflect the matrix-matrix multiplications required to form $S = BF^{-1}B^T$.

Grid Param	Probing	*luinc('0')*	*luinc(optns)*
4	0.280	0.357	0.242
5	1.850	5.611	3.679

Table 12. Timing Data is Average Value in Seconds

Timing *luinc()* accurately was hampered by the fact that MATLAB 7 has unexpected and unexplainable timing results when *luinc()* is used with the parameters *luinc('0')* and *luinc(optns)*. Using *optns* speeds up the execution of the *luinc()* call by a factor of around 5. This speed up occurs even if the convergence rate of GMRES and the number of non-zeros in the resulting factors changes negligibly when comparing *'0'* with *optns*.[3] Nonetheless, the MATLAB timings do indicate that using ILU instead of probing is significantly more expensive, especially for larger grids.

Probing seems to scale much better to larger grid sizes, which indicates it would be a better method for larger problems. Also, substituting *luinc(optns)* for *lu()* in the *mat-vec* routine used by the probing software lowers the probing timings by 21% for grid size 4 and 46 % for grid size 5. Using *luinc(optns)* instead of *lu()* did not affect the convergence of GMRES significantly, although it did have a slight negative effect.

Computational Bounds on Probing and ILU Computational bounds for structured probing are provided in [5]. The expensive steps in structured probing are

[3] MathWorks technical support explained that using *optns* with a *droptol* executes a completely different algorithm than *'0'*. Using *optns* with a *droptol* executes an algorithm which is more concerned with maintaining the large values in the incomplete factors, while using *'0'* executes an algorithm that maintains the sparsity structure of the original matrix exactly. How this makes *'0'* significantly slower is still not clear, however.

1. $O(nv^2)$ operations for graph coloring

2. $O(nvp)$ operations for matrix vector multiplies to build the resulting approximate matrix

where n is the dimension of the $n \times n$ matrix, S, which is to be approximated, v is the number of non-zeros per row, and p is the number of colors used. For all of our problems regardless of grid size and Reynolds number, $p = v = 9$. In our experiments, we chose the probing sparsity pattern to match the sparsity pattern of A_p. A_p never has more than 9 non-zeros in a row, so $p = v = 9$.

Using ILU to approximate S requires computations of the form:

$$\begin{aligned} [L,U] &= luinc(F,\text{'0'}); \qquad \%compute\ ILU \\ S &= B(U\backslash(L\backslash B^T)); \end{aligned} \tag{11}$$

In examining the cost of (11), the most important operations to understand are $(U\backslash(L\backslash B^T))$. The main problem is that a back-solve and forward-solve produce a dense result. This makes not only the repeated back-solves and forward-solves expensive, but also the subsequent matrix-matrix multiplication with B. The computational cost of (11) is

1. $O(k^2 r)$ operations to compute an ILU on F, where k is the number of non-zeros in a row of F and r is the number of rows in F. It should be noted that in certain circumstances performing an ILU(F) might be natural for the full version of the preconditioner and may be already available.

2. *O(k r)* operations to do a single back-solve or forward-solve with U and L, respectively, where kr is the total number of non-zeros in the factors which is also approximately equal to the number of non-zeros in F when ILU(0) is done. The back-solve and forward-solve will have to be done for every column in B^T, so this effectively makes this operation cost $O(krs)$, where s is the number of columns in B^T and s equals the number of pressure variables.

3. *O(x y m)* operations for matrix-matrix multiplication of B and the resulting operand to its right, where x is the number of non-zeros in a row of B, y is the number of non-zeros in a column of $(U\backslash(L\backslash B^T))$ and m is the number of non-zeros in the resulting matrix, S. Note that $y \gg 0$ and hence $m \gg 0$.

It should be clear that doing ILU to approximate S is more computationally expensive. With probing, we have operations on the order of *O(81 n)*. With ILU, we have operations on the order of $O(xym)$. In our experiments, m is in the thousands to hundreds of thousands, depending on the grid (see Table 13), $xy \gg 81$ and n is in the hundreds to thousands, depending on grid size.

Non-zeros in *S* In order to gauge the cost of the *luinc(S)* call that both (9) and (10) make, we present the number of non-zeros in each method's approximate S. The number of non-zeros for each method was the same for all Reynolds numbers and only varied with respect to grid size.

The number of non-zeros in S from (10) did not change if *luinc(optns)* was used instead of *luinc('0')*. Using method (10) also yielded an equal number of non-zeros as the actual Schur complement, despite the fact that the incomplete factors are much sparser than the full LU factors. The forward-solves and back-solves against the columns of B^T in forming the approximate S from (10) are the steps where the density is restored to that of the exact S. A possible alternative to using ILU(F) to calculate S that would avoid such a dense result would be sparse approximate inverses. We do not explore this option in this paper.

Probing results in far fewer non-zeros and again scales better than ILU when moving from grid size 4 to 5. Probing scales better in the sense that its number of non-zeros increases at a much slower rate .

Grid Param	Probing	*luinc()*
4	1,681	43,681
5	6,529	591,361

Table 13. Number of Non-zeros in S

2.6 Other F_p Updates

We experimented with other F_p updates that did not yield reasonable or improved convergence. These were

1. Iterated solves of F_p or A_p, where both the rows and the columns of only F_p or only A_p were updated in succession. This generally had little noticeable impact on performance, and never made convergence faster than just updating only the rows or only the columns. However with probing, this method exhibited much worse convergence when compared to row only or column only updates.
2. Iterated F_p and A_p updates, where the rows or columns of F_p would be updated followed by updating the rows or columns of A_p, or vice versa. This had similar convergence to just updating F_p.
3. Probing twice, once for S and once for S^{-1}.
4. Using ILU to approximate F^{-1} in order to form an approximate S followed by probing to form an approximate S^{-1}.
5. Using the Least Squares Commutator formula [2] to form our own F_p. We used the following equation

$$\begin{aligned} S &= \left(B\left(G^{-1}B^T\right)\right) \\ F_p &= S^{-1}B\left(G^{-1}FG^{-1}B^T\right), \end{aligned} \tag{12}$$

   where G is the velocity mass-matrix.

Using probing to form an approximate S^{-1} in 3 and 4 yielded some of the worst convergence rates. Probing was used to form S^{-1} by simply changing the *mat-vec* function used by the probing software so that given a matrix, S, the function would form the L and U factors of S once and then output *(U\L \probe)* for any probing vector, *probe*.

3 Conclusions

We have experimented with pressure convection–diffusion preconditioners on a backward facing step problem. The choice of boundary conditions with the F_p and A_p operators can have a very significant effect on the convergence of the standard pressure convection–diffusion scheme. The use of "ideal" boundary conditions for F_p significantly reduces the convergence sensitivity to the boundary conditions chosen for A_p. These "ideal" conditions are defined by algebraically requiring that the preconditioned Schur complement on the boundary points correspond to the identity matrix. Further, "ideal" boundary conditions for F_p give rise to mesh independent convergence rates for the backward facing step. While mesh independent convergence rates have been reported for other problems, e.g., lid-driven cavity problems, they have not generally been observed for standard pressure convection–diffusion preconditioners on the backward facing step. Our results imply that the culprits for this loss of mesh independence are the boundary conditions used within the preconditioner. While significant improvements were also observed with "ideal" A_p boundary conditions, the results were generally less satisfying than ideal F_p boundary conditions.

Inexpensive approximations to the "ideal" boundary conditions have also been explored based on probing and ILU ideas. In general, the ILU methods give excellent convergence results, but are still too expensive to be used in practice. The probing approximations lead to a fairly practical algorithm in terms of computational cost. While the convergence with probing is good, it leads to noticeably inferior convergence rates when compared with ILU.

Overall, the effectiveness of 2.5.2 and 2.5.1 indicate promise in the direction of developing a general preconditioning strategy that does not require extensive experimental knowledge of which boundary conditions to choose. Further research should be performed to identify other approximations to the "ideal" boundary conditions based on either sparse approximate inverses, coarse grid approximations or more expensive probing variants.

4 Acknowledgments

We would like to thank Chris Siefert for providing us with the MATLAB structured probing code, which we used in our probing experiments.

References

[1] F. BREZZI AND M. FORTIN, *Mixed and Hybrid Finite Element Methods*, vol. 15, Springer–Verlag, Berlin, 1991.

[2] H. C. ELMAN, V. E. HOWLE, J. SHADID, R. SHUTTLEWORTH, AND R. TUMINARO, *Block preconditioners based on approximate commutators*, SIAM J. Sci. Comput., to appear.

[3] H. C. ELMAN, D. J. SILVESTER, AND A. J. WATHEN, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, Oxford, 2005.

[4] D. KAY, D. LOGHIN, AND A. WATHEN, *A preconditioner for the steady-state Navier–Stokes equations*, SIAM J. Sci. Comput., 24 (2002), pp. 237–256.

[5] C. SIEFERT AND E. DE STURLER, *Probing methods for generalized saddle-point problems*, Tech. Rep. UIUCDCS-R-2005-2540, University of Illinois, Urbana-Champaign, Urbana, IL 61801, 2005.

[6] D. SILVESTER, H. ELMAN, D. KAY, AND A. WATHEN, *Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow*, J. Comp. Appl. Math., 128 (2001), pp. 261–279.

[7] D. J. SILVESTER, H. C. ELMAN, AND A. R. RAMAGE, *Incompressible Flow Iterative Solution Software Package*. http://www.ma.umist.ac.uk/djs/ifiss/.

Sandia National Laboratories