Ph.D. Candidacy Prospectus
Methods for Block Preconditioning the Navier-Stokes Equations

Robert Shuttleworth

December 7, 2004

1 Motivation

The main goal for this project is to develop new preconditioning techniques for the incompressible Navier-Stokes equations. The objectives of these techniques are to:

1. develop fully implicit, robust, scalable algorithms, and
2. overcome the computational bottlenecks that arise in the current solution of large scale fluid flow simulations.

The hypothesis being tested is that it is possible to use information about the underlying physical problem to more robustly and efficiently solve the linear subproblems that arise from temporal and spatial discretizations in fluid flow simulations.

The new algorithm will implicitly solve the incompressible Navier-Stokes equations using efficient, scalable and robust preconditioning techniques. Moreover, these techniques are based upon using the solution of certain subproblems as building blocks to solve the coupled Navier-Stokes equations.

2 Background

The robust solution of partial differential equations (PDEs) is of vital interest in technology, engineering and national security. Scientific interest in computational fluid dynamics (CFD) includes the desire to model physical fluid phenomena that cannot be easily simulated or measured with a physical experiment. More specifically, incompressible flows are useful for modeling diverse phenomena such as, combustion, pollution, chemical reactions, and manufacturing processes. These flows can be modeled using the Navier-Stokes equations:

$$\alpha \mathbf{u}_t - \nu \nabla^2 \mathbf{u} + (\mathbf{u} \cdot \text{grad})\mathbf{u} + \text{grad } p = f \tag{1}$$

subject to the incompressibility constraints:

$$-\text{div } \mathbf{u} = 0$$

where $\mathbf{u}$ is the velocity of the fluid, ν the viscosity, and p the pressure. The value $\alpha = 0$ corresponds to the steady-state problem and $\alpha = 1$ to the transient case. A stable finite element or finite volume discretization and corresponding linearization (via Picard or Newton iteration) of the Navier-Stokes equations leads to a linear saddle point system[1] of equations of the form:

$$A = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} \tag{2}$$

where F is the convection diffusion operator, B^T is the gradient operator, and B is the divergence operator. The linear system found in (2) must be solved at each step. The cost for solving this system can be very high, and takes upwards of 80% of the CPU time for a given simulation. Therefore, reducing this CPU time will allow scientists and engineers to pose and solve more realistic problems. One way to reduce this computational time is by coupling iterative solvers with preconditioners to solve (2).

[1] *Note: Saddle point systems of this form are also found in electrical networks, structural networks, optimal control problems, and computer graphics.*

Over the years, numerous techniques have been developed to efficiently solve saddle point linear systems (2) and the Navier-Stokes equations (1), examples of which can be found in ([1],[2], and [5]). However, these methods have many shortcomings, i.e. they decouple the two equations, converge slowly, or only work well for small time steps.

3 Solution Methods

Modeling realistic types of fluids problems, in three dimensions, with the associated chemistry and physics requires a robust set of efficient and massively parallel algorithms. For challenging problems in scientific computation, preconditioning is the most crucial component in developing efficient and robust solution techniques. Furthermore, the ability to transform problems that appear intractable into ones that can be approximated rapidly, are of paramount interest.

3.1 General Preconditioning Techniques

To understand the need for preconditioners, we will first examine solving a linear system, $Ax = b$. Direct methods, such as Gaussian Elimination, are expensive in terms of operation counts and memory as the problem size grows. Large multiphysics simulations, with hundreds of million unknowns, are intractable with direct methods. Currently, iterative techniques, such as Krylov subspace methods, are the only available option to solve these types of problems. These techniques use far less CPU memory and operations than direct methods. However, iterative methods are not as reliable as direct methods, nor do they always converge to a solution.

Preconditioners are commonly used to increase the reliability and performance of iterative techniques. In brief, preconditioning refers to the process of transforming a linear system, $Ax = b$, into another $\hat{A}x = \hat{b}$, that has better properties with respect to the iterative solution strategies. In many cases, the preconditioning matrix is designed to improve the spectral properties of the original matrix. In other words, if Q is a matrix that approximates A, then

$$Q^{-1}Ax = Q^{-1}b$$

has the same solution as the original system, $Ax = b$, but should be easier to solve than $Ax = b$. This idea of transforming a problem from one that is hard to solve to one that is easier is not a new concept. For example, Jacobi (1845) used the concept of preconditioning when he was calculating planetary distances. However, the term preconditioning was not coined until Turing (1948) in a paper about roundoff error. A major computational breakthrough came in the 1970s when Meijerink and van der Vorst discovered the incomplete Cholesky Conjugate Gradient method [11].

Today, there are two major types of preconditioners, general and problem specific. General preconditioners include incomplete factorization (ILU), sparse approximate inverses, and, to some extent, domain decomposition, and multilevel multigrid techniques. However, these methods have many drawbacks. For example, incomplete factorization techniques do not parallelize well and are not scalable, while sparse approximate inverse techniques are not robust. The domain decomposition technique, which refers to the idea of breaking a computational domain/mesh into smaller domains that can then be preconditioned with an ILU or sparse approximate inverse technique parallelize well, but the requirement that

coarse grids be handled by direct methods causes it to scale poorly as the problem size increases. Newer techniques, based on multilevel multigrid-like methods show promise, but there are still many open questions about them[10].

3.2 Preconditioning Techniques for the Navier-Stokes Equations

Problem specific preconditioners use the underlying physical problem as motivation for the derivation of the methodology. Our focus is on preconditioning techniques for the Navier-Stokes equations. Historically, there are many solution techniques for the Navier-Stokes equations. These include fractional step methods, fully decoupled methods, and fully coupled methods. The fractional step methods, such as pressure projection or operator splitting ([1]), and the fully decoupled techniques, such as SIMPLE (Semi-Implicit Method for Pressure Linked Equations) and SIMPLER ([2], and [3]), do not preserve the coupling of physics, while the fully coupled convection-diffusion techniques do ([4],[5] and [6]). For our purposes, we will focus our discussion to two solution branches, mainly fully decoupled techniques, and pressure convection-diffusion preconditioners.

The SIMPLE algorithm begins by solving the momentum equation, then the continuity equation is solved to calculate the pressure component. In turn, this value is used to update the velocity component. An alternative derivation of SIMPLE, (in matrix form) can be seen from the LDU decomposition of (2). In the case of SIMPLE, the L and the D are grouped together. Then the corresponding preconditioner is of the form: $U^{-1}(LD)^{-1}$. The derivation is as follows:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} F & 0 \\ B & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix} \approx \begin{bmatrix} F & 0 \\ B & \hat{S} \end{bmatrix} \begin{bmatrix} I & \hat{F}^{-1}B^T \\ 0 & I \end{bmatrix} \tag{4}$$

where $S = -BF^{-1}B^T$, $\hat{F} = \text{diag}(F)$, and $\hat{S} = -B\hat{F}^{-1}B^T$ [2]. Then, a preconditioner can be defined using the inverse of the right hand side:

$$\left(\begin{bmatrix} F & 0 \\ B & \hat{S} \end{bmatrix} \begin{bmatrix} I & \hat{F}^{-1}B^T \\ 0 & I \end{bmatrix} \right)^{-1} = \begin{bmatrix} I & \hat{F}^{-1}B^T \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} F & 0 \\ B & \hat{S} \end{bmatrix}^{-1} \tag{5}$$

$$= \begin{bmatrix} I & -\hat{F}^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} F & 0 \\ B & \hat{S} \end{bmatrix}^{-1} \tag{6}$$

[3]. The strategy has many shortcomings. For example, the choice of $\hat{F} = diag(F)$ works well for small time steps, but is inefficient for small mesh sizes or convection dominated flows. Furthermore, these methods are based upon decoupling the Navier-Stokes equation which can lead to slow convergence because the coupling of physics is violated.

The fully coupled pressure convection-diffusion preconditioners are insensitive to mesh size, time step, and CFL number. For stationary problems, there is a slight dependence on Reynolds number, while transient problems have no Reynolds number dependence. Computationally, this preconditioner can be implemented as:

$$P = \begin{bmatrix} F^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & -B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & -S^{-1} \end{bmatrix}$$

[2] *The Schur complement cannot be computed explicitly because its a full dense matrix, therefore its intractable for these types of problems.*

where S is the Schur complement. From this decomposition, one can see that it takes two difficult operations to apply this preconditioner. These are that S^{-1} needs to be applied to a vector in the pressure space and F^{-1} needs to be applied to a vector in the velocity space. So a good preconditioner in this family will make good approximations to S^{-1} and F^{-1}. In the case of this preconditioner, the application of F^{-1} can be with a multigrid iteration ([10]), which is inexpensive. However, the choice of the approximations to S^{-1} are less straightforward. We are considering several methods. One of these is based upon defining an operator, F_p, that is defined on the pressure space (analogous to the convection diffusion matrix, F, that is defined on the velocity space) [4]. Another method, found in [5], approximates the Schur complement by removing the action of the inverse of F from the inside of the Schur complement. Both of these methods also have many open questions, such as the form of the preconditioner for stabilized finite element discretizations.

In preliminary work under the direction of my advisor, Dr. Howard Elman, along with Vicki Howle and Ray Tuminaro of Sandia National Laboratory, we have implemented (in C++) a few Navier-Stokes preconditioners from the branches described above. Furthermore, we have coupled this parallel code to MPSalsa, a massively parallel, chemically reactive fluid flow code. In preliminary work, a number of serial and parallel trials have been run for the lid driven cavity problem over a range of Reynolds numbers.

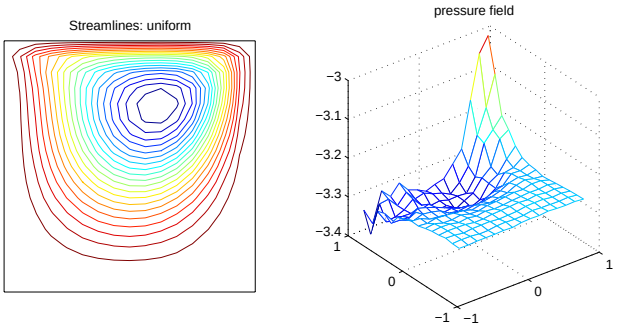

Figure 1: Lid Driven Cavity - This example represents flow in a square cavity with the top of the lid moving from left to right. The boundary conditions are $\mathbf{u} = 0$ on $\partial\Omega$ except $u_1(x, 1) = 1$ at the top of Ω.

There are many computational issues involved in efficiently implementing these types of techniques while mitigating the computational bottlenecks. The high performance computing issues that we will focus on include message passing, storage and partitioning of sparse matrix operators, and memory management. To implement these techniques for use in massively parallel environment, the message passing interface (MPI), [9], will be used to handle the communication between processors. Efficient techniques to store the sparse matrix operators can be found in [7]. While methods for partitioning matrices for parallel computations or multilevel multigrid solutions can be found in [8]. All of these techniques are beneficial to building a robust, valid, and efficient block preconditioner for the Navier-Stokes equations.

The remaining steps for developing a robust, scalable software package for the incompressible Navier-Stokes equations are:

1. *Theoretical:* The methods described above have been developed for stable spatial discretizations. However, many production level codes (including MPSalsa) use an

explicit stabilization procedure. Therefore, the saddle point matrix becomes:

$$\begin{bmatrix} F & B^T \\ B & -C \end{bmatrix}.$$

So, appropriate techniques need to be developed for other preconditioners in the pressure convection-diffusion family. These techniques also need to be expanded to handle more general models that include, mainly temperature or chemistry. Currently, problems have only been tested with enclosed flow, we would like to expand this to cover more realistic test problems, including those with inflow and outflow capabilities.

2. *Software:* The code will need to be modified to handle stabilized meshes and other models, including temperature or chemistry. This will allow us to link our code to Fuego, a fire prediction code developed at Sandia National Laboratory. We will also test these techniques on spectral element spatial discretizations. Finally, we will create an additional portion of code that automatically generates the F_p preconditioner, therefore making it easier for users to use these techniques.

3. *Validation/Testing:* With a project this size, it is vital that the methods are accurate and working properly. Benchmark test problems will be chosen that test many features of realistic, harder flows, including inflow/outflow and singularities. This will aid in determining the extent to which these preconditioners provide accurate results.

References

Area of Specialization: Preconditioning the Navier-Stokes Equations

[1] A.J. Chorin, A numerical method for solving incompressible viscous problems, Journal of Computational Physics, 2:12-26,1967.

[2] S.V. Patankar, *Numerical heat transfer and fluid flow*, Hemisphere Pub. Corp., New York, 1980, pg. 124-135.

[3] M. Pernice and M.D. Tocci, A multigrid-preconditioned Newton Krylov method for the incompressible Navier-Stokes equations., SIAM J. Sci. Comput. 123:398-418, 2001.

[4] D. Kay, Loghin, D. and A.J. Wathen, A preconditioner for the steady-state Navier-Stokes equations. SIAM Journal of Scientific Computation 24:237-256, 2002.

[5] H. C. Elman, D. J. Silvester and A. J. Wathen, *Finite Elements and Fast Iterative Solvers*, Oxford University Press, to appear. *Chapters 3,4,7, and 8.*

[6] H. C. Elman, V. E. Howle, John Shadid and Ray Tuminaro, A parallel block multi-level preconditioner for the 3D Incompressible Navier-Stokes Equations. Journal of Computational Physics 187:504-523, 2003.

Related Area: High Performance Computing

[7] J. Dongarra, A. Lumsdaine, X. Niu, R. Pozo, and K. Remington. "A sparse matrix libray in C++ for high performance computing architectures," Proceedings of the Second Annual Object-Oriented Numerics Conference, 1994.

[8] G. Karypis and V. Kumar. "A Coarse-Grain Parallel Formulation of Multilevel k-way Graph Partitioning Algorithm," SIAM Conference on Parallel Processing for Scientific Computing,

[9] The MPI Fourum. *MPI:Message Passing Interface Standard*. University of TN, 1995.

Related Area: Sparse Linear Systems

[10] U. Trottenberg, C.W. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, San Diego, 2001. *Chapters 2 and 6.*

[11] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, 1997. *Chapter 6.*

4 Relevant Coursework

In this project, I will be developing a software package for block preconditioning the Navier-Stokes equations. My approach requires knowledge of techniques in mechanics, numerical linear algebra, and finite element/finite difference methods.

4.1 Relevant Science Courses

Science Courses have been chosen to provide "a foundation in my discipline." Therefore, courses in engineering and applied mathematics that achieve this goal are:

- ENME 670 - Mechanics of deformable bodies, deformation and strain measures, kinematics of continua with global and local balance laws. Thermodynamics of continua and an introduction to constitutive theory for elastic solids and viscous fluids.
- ENME 677 - Field equations and constitutive laws for linear elasticity, linearized boundary value problems, biharmonic equation, Airy stress function, Saint-Venant's principle, thermoelastic problems, and the Boussinesq problems.
- AMSC 698A (Computational Fluid Mechanics) - Techniques for discretizing the Navier-Stokes Equations; pressure Poisson solvers; voriticity formulation, projection methods, implict and explict numerical formulations.

4.2 Relevant Mathematics Courses

Mathematics courses have been chosen to provide exposure to "computational methods and mathematical modeling." Application courses that enhance my knowledge of numerical linear algebra, and computational methods and modeling in mechanics are:

- AMSC 698I (Numerical Methods in Math Finance) - Discrete models, The Black-Scholes Model, Finite Difference Methods, Finite Element Methods, and Levy Processes and Partial Differential-Integral Equations.
- AMSC 660 - Monte Carlo simulation, numerical linear algebra, nonlinear systems and continuation method, optimization, ordinary differential equations.
- AMSC 661- Fourier and wavelet transform methods, numerical methods for elliptic partial differential equations, numerical linear algebra for sparse matrices. Finite element methods, numerical methods for time dependent partial differential equations.