Massively parallel linear stability analysis with P_ARPACK for 3D fluid flow modeled with MPSalsa*

R.B. Lehoucq[1] and A. G. Salinger[1]

Sandia National Laboratories**
P.O. Box 5800, MS 1110
Albuquerque, NM 87185-1110
{rlehoucq,agsalin}@cs.sandia.gov

Abstract. We are interested in the stability of three-dimensional fluid flows to small disturbances. One computational approach is to solve a sequence of large sparse generalized eigenvalue problems for the leading modes that arise from discretizating the differential equations modeling the flow. The modes of interest are the eigenvalues of largest real part and their associated eigenvectors. We discuss our work to develop an efficient and reliable eigensolver for use by the massively parallel simulation code MPSalsa. MPSalsa allows simulation of complex 3D fluid flow, heat transfer, and mass transfer with detailed bulk fluid and surface chemical reaction kinetics.

1 Introduction

Galerkin least squares finite element discretization (GLS FE) of time-dependent Navier–Stokes modeling of incompressible fluid flow (see [8]) produce the matrix system

$$\begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{N} & \mathbf{0} \end{pmatrix} \begin{bmatrix} \dot{\mathbf{u}} \\ \dot{\mathbf{p}} \end{bmatrix} + \begin{pmatrix} \mathbf{L} & -\mathbf{C}^T \\ -\mathbf{C}\mathbf{R} + \mathbf{G} & \mathbf{K} \end{pmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \mathbf{d}_2 \end{bmatrix}$$

where $\mathbf{u}$ is the fluid velocity and possible temperature, $\mathbf{p}$ is the pressure, $\mathbf{M}$ is the symmetric positive definite matrix of the overlaps of the finite element basis functions, $\mathbf{N}$ is an up-winded mass matrix, $\mathbf{L}$ is the sum of the discretized diffusion, nonlinear convection and any possible reaction operators, $\mathbf{C}$ is the discrete gradient, $\mathbf{C}^T$ is the discrete divergence operator, $\mathbf{R}$ diagonal matrix representing variable density, and $\mathbf{G}$ and $\mathbf{K}$ (pressure Laplacian) are stabilization terms arising from the GLS FE.

* This work was in part supported by the United States Department of Energy Applied Mathematical Sciences Program.

** Sandia is a multiprogram laboratory operated by Sandia Corportation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Linearizing about the steady state gives rise to the following generalized eigenvalue problem

$$\begin{pmatrix} \mathbf{L} & -\mathbf{C}^T \\ -\mathbf{CR}+\mathbf{G} & \mathbf{K} \end{pmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} = \begin{pmatrix} \mathbf{M} & \mathbf{0} \\ \mathbf{N} & \mathbf{0} \end{pmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{p} \end{bmatrix} \lambda \tag{1}$$

or

$$\mathbf{Jx} = \mathbf{Bx}\lambda.$$

Here, $\mathbf{J}$ is the Jacobian matrix associated with the steady state or base flow and $\mathbf{B}$ is mass matrix, that is singular for incompressible flows. We denote the order of the matrices $\mathbf{J}$ and $\mathbf{B}$ by n.

The steady state is stable if $\mathrm{Real}(\lambda) < 0$ for all the eigenvalues of (1). Hence, computing approximations to the right-most eigenvalues determines the stability of the steady state. The goal of our paper is to present preliminary results on providing a massively parallel eigensolver for computing these rightmost eigenvalues.

To compute the right-most eigenvalues, a shift-invert spectral transformation [14] is typically used to transform (1) into the standard eigenvalue problem

$$\mathbf{T}_s\mathbf{x} = (\mathbf{J} - \sigma\mathbf{B})^{-1}\mathbf{Bx} = \nu\mathbf{x}, \quad \nu = \frac{1}{\lambda - \sigma}. \tag{2}$$

The above formulation maps the infinite eigenvalues of (1) (arising from singular $\mathbf{B}$) to zero. By selecting the pole near the imaginary axis, the right-most eigenvalues are mapped by $\mathbf{T}_s$ into those of largest magnitude. However, because $\mathbf{J}$ and $\mathbf{B}$ are real matrices, we only allow a real σ to keep the computation in real arithmetic. Although a natural choice is to select a zero pole, the resulting transformation might miss a Hopf bifurcation (complex conjugate pair of eigenvalues that is in the right half of the complex plane). This occurs, for instance, when the distance to the Hopf bifurcation is greater than the distance to other (perhaps stable) eigenvalues of (1). The paper [3] discusses these issues in some detail.

The computational burden is in solving the linear set of equations with coefficient matrix $(\mathbf{J} - \sigma\mathbf{B})^{-1}\mathbf{B}$. The standard approach is to use a sparse direct solver to factor $\mathbf{J} - \sigma\mathbf{B}$ and then solve linear sets of equations (see [3],[4],[1],[10]). In this paper, we use the related generalized Cayley spectral transformation [14] that is discussed in Section 3.

Because our interest is in three dimensional systems with coupled fluid flow, heat transfer, and mass transfer, we need methods that will work for system sizes of order over $n = 10^5$ and that will scale well up to systems of size $n = 10^8$. Hence, solving linear systems with sparse direct methods is not a viable alternative. They require $\mathcal{O}(n^2)$ operations plus a prohibitive amount of memory. Instead, this paper considers the use of iterative methods for the linear solves on massively parallel machines. Along with a scalable eigensolver, such an approach allows very large system eigenvalue problems to be solved.

2 The Players

In this section, we introduce the massively parallel software for modeling three-dimensional fluid flow, the large scale parallel sparse eigensolver, the iterative linear solver and the massively parallel supercomputer.

2.1 MPSalsa

MPSalsa [20],[18] simulates complex systems with coupled fluid flow, thermal energy transfer, mass transfer and non-equilibrium chemical reactions. Currently, computer simulations of these complex reacting flow problems are usually limited to idealized systems in one or two spatial dimensions when coupled with a detailed, fundamental chemistry model. The goal of the MPSalsa project is to develop, analyze and implement advanced massively parallel numerical algorithms that will allow high resolution three-dimensional simulations with an equal emphasis on fluid flow and chemical kinetics modeling.

MPSalsa uses a GLS FE formulation [8] and has been written for unstructured meshes in two and three dimensions and to run on distributed memory massively parallel computers. A fully-coupled Newton's method is used to solve the nonlinear equations, which requires the inversion of a large sparse matrix at every iteration. With these robust methods, it is often possible to reach steady-state solutions directly from a trivial initial guess without the need of following a time transient. Since the steady-state algorithm does not discriminate between stable and unstable steady states, linear stability analysis is a crucial capability for using MPSalsa as an analysis tool.

MPSalsa has been used to analyze a variety of reacting flow systems, including the analysis of 3D models of Chemical Vapor Deposition (CVD) reactors for the growth of gallium arsenide semi-conducting films [19].

2.2 P_ARPACK

P_ARPACK [12] software is capable of solving large scale symmetric, nonsymmetric, and generalized eigen-problems from significant application areas. The software is designed to compute a few eigenvalues with user specified features such as those of largest real part or largest magnitude. The software implements an implicitly restarted Arnoldi Method [21]. Restarting is used so that the number of Arnoldi iterations (the size of the corresponding Hessenberg matrix) remains fixed. An effective restarting scheme finds ever better starting vectors so that the subsequent Arnoldi run contains the desired eigenvalue approximations. Implicit restarting is a stable and efficient scheme; for more information see [11]. P_ARPACK uses a data parallel model of computation where each processor owns a number of rows of the Arnoldi vectors.

2.3 Aztec

Aztec [9] is a parallel iterative library for solving linear systems of equations. A globally distributed matrix allows a user to specify pieces (different rows for different processors) of the application matrix. Issues such as local numbering, ghost variables, and messages that are crucial for parallelizing an application can be ignored by the user and are instead computed by an automated transformation function. Efficiency is achieved by using standard distributed memory techniques; locally numbered sub-matrices, ghost variables, and message information computed by the transformation function are maintained by each processor so that local calculations and communication of data dependencies is fast. Additionally, Aztec takes advantage of advanced partitioning techniques (Chaco [7]) and utilizes efficient dense matrix algorithms when solving block sparse matrices.

Aztec allows the user to choose between several Krylov-based iterative solution techniques as well as several algebraic preconditioners. In this work, we have exclusively used the GMRES algorithm [17] with a sufficiently large Krylov space (so that no restarts were necessary) that is preconditioned using the ILU(0) domain decomposition preconditioner. The domains for the preconditioning are the same as the partitioning used for parallelizing the MPSalsa calculation, so that the number of domains is the same as the number of processors that the application is run on.

2.4 Computer

The calculations presented here were run on the Sandia-Intel Teraflop massively parallel supercomputer [13] (a.k.a. ASCI Red). This computer has 4,536 nodes, each with two 200Mhz Pentium Pro Processors and 128 MBytes of memory. The theoretical peak is 1.8 TFLOPS and over 1.0 TFLOPS has been achieved. Node to node bandwidth is 800 MBytes/sec. More information can be found at `www.sandia.gov/ASCI/TFLOP/Home_Page.html`.

In 1997, MPSalsa-Aztec was a finalist for the Gordon Bell prize for reaching a sustained rate of 210 Gflops on 3600 nodes of the Sandia-Intel Tflop computer [5].

3 Algorithm

We now briefly describe the algorithm used for computing several (say k) rightmost eigenvalues of (1). As we have already mentioned, the dominant cost of the computation is that of solving the linear systems of equations associated with the shift-invert spectral transformation. Although this transformation maps the eigenvalues near the pole to those of largest magnitude, the transformation also maps the eigenvalues far from the pole to zero. Hence, the spectral condition number (absolute value of the maximum ratio of the eigenvalues) of $\mathbf{T}_s$ can be quite large. The resulting linear systems will be difficult to solve because the rate of convergence with an iterative method [16],[6] depends strongly upon the spectral condition number.

A better conditioned linear set of equations is achieved when using a generalized Cayley [14] transformation resulting in

$$\mathbf{T}_c\mathbf{x} = (\mathbf{J} - \sigma\mathbf{B})^{-1}(\mathbf{J} - \mu\mathbf{B})\mathbf{x} = \nu\mathbf{x}, \quad \nu = \frac{\lambda - \mu}{\lambda - \sigma}. \tag{3}$$

We call μ the zero of the Cayley transform. In contrast to shift-invert transform, the Cayley transform maps any eigenvalues of (1) far from the pole close to one. If we are able to select a pole that is to the right of all the eigenvalues (1) along with $\sigma < \mu$, then the smallest eigenvalue of $\mathbf{T}_c$ is no smaller than one (in magnitude). Moreover, by judiciously choosing the pole, we can approximately bound the largest eigenvalue of $\mathbf{T}_c$ (in magnitude) resulting in a modest (say order ten) spectral condition number.

– Start P_ARPACK with the vector $\mathbf{v} = \mathbf{J}^{-1}\mathbf{B}\mathbf{x}$ where $\mathbf{x}$ is random vector. Select a pole and zero for $\mathbf{T}_c$.
  1. Compute m Arnoldi iterations for $\mathbf{T}_c$ with starting vector $\mathbf{v}$.
  2. Exit if the k eigenpairs satisfy the user specified tolerance.
  3. Implicitly restart the Arnoldi iteration.
  4. Pick a new σ and μ based on the current approximate eigenvalues.

A few remarks are in order. The starting vector is chosen so that it does not contain any components [15] in the null-space of $\mathbf{B}$. The orthogonality of the m Arnoldi vectors is maintained at machine precision. At step 2, the eigenvalues of the m by m Hessenberg matrix are mapped back to the system defined by (1) via the inverse Cayley transformation. The eigen-computation is terminated when these k rightmost approximate eigenvalues satisfy the user specified tolerance. The check that must be satisfied is equivalent to computing $||\mathbf{J}\hat{\mathbf{x}} - \hat{\lambda}\mathbf{B}\hat{\mathbf{x}}||$ where $\hat{\mathbf{x}}$ and $\hat{\lambda}$ are the approximate eigenvector and eigenvalue computed. By implicitly restarting the Arnoldi iteration, we compute a new starting vector for the subsequent run (step 1). Finally, at step 4, the new pole and zero for the next Cayley transformation are selected so that the spectral condition number of $\mathbf{T}_c$ is of order ten.

4 Numerical Experiments and Discussion

The flow and heat transfer in a rotating disk CVD reactor is chosen as the targeted test problem for the linear stability analysis. It is an important engineering application where transitions from stable to unstable (and undesirable) flows have been seen experimentally [2]. The flow stability problem is complex because of competing flow effects due to forced convection at the inlet of the reactor, rotationally-driven convection at a spinning disk in the reactor, and buoyancy-driven flow due to large temperature gradients. The system—geometry, equations, and boundary conditions—is axisymmetric, but an understanding of the bifurcations to three-dimensional solutions is crucial for successful design of the next generation CVD reactor.

A three-dimensional, unstructured finite element mesh consisting of 43520 elements is used to discretize the 5 coupled PDEs that govern the flow and heat transfer in this system. The total number of unknowns in the problem, and correspondingly the rank of the eigenvalue problem, is 233925.

For the calculations presented, GMRES was the iterative solver used for all the linear systems along with an overlapping Schwarz domain decomposition preconditioner with ILU(0) on each sub-domain. All computations were done using double precision floating point arithmetic.

Since P_ARPACK is matrix-free (no specified storage format is assumed for the matrices), the VBR (variable block row) storage format provided by Aztec is used for the matrices. A node based distribution of the problem is used hence there are repeated elements (ghost nodes). All the necessary processor communication and associated work is done by Aztec.

The MPSalsa calculation of the steady solution from a trivial initial guess on 200 Processors required 7 Newton iterations and 305 seconds, 90% of which was spent in the linear solver. Each linear solve was required to reduce the residual by 10^{-4} and required Krylov subspaces ranging from 215 to 289, and used 40 seconds on average.

A calculation of 7 leading eigenvalues of this system was performed with the algorithm described above. The total time for the eigenvalue calculation was 1800 seconds. The following discussion presents the times for each step in the algorithm as well as some of the choices in parameters that were made. A tolerance of 10^{-5} used for P_ARPACK.

Only 3 seconds were required to calculate the matrices $\mathbf{J}$ and $\mathbf{B}$. The initial guess of $\mathbf{v} = \mathbf{J}^{-1}\mathbf{B}\mathbf{x}$ for a random vector $\mathbf{x}$ was computed where the linear solver decreased the residual by 10^{-12} in 58 seconds. Using $\sigma = 100$ and $\mu = 1000$ for the initial Cayley transformation, $m = 24$ Arnoldi iterations were computed. The preconditioner was calculated in 10 seconds and and reused in each of linear solves. GMRES required on average 20 seconds and a Krylov subspace size of 175 (without restarts) to reduce the residual by a factor of 10^{-10}.

At this point, none of the computed eigenvalues satisfied the desired tolerance, and a new shift and pole were chosen as $\sigma = -6.86$ and $\mu = 63.5$. Another $m = 24$ Arnoldi iterations were computed and each linear system required 51 seconds and a Krylov subspace size of 345 to reduce the residual by a factor of 10^{-10}. The 7 converged eigenvalues and residuals are listed in Table 1. The numbers listed in the third and fourth columns are $\|\mathbf{J}\hat{\mathbf{x}} - \hat{\lambda}\mathbf{B}\hat{\mathbf{x}}\|/\|\mathbf{B}\hat{\mathbf{x}}\|$ (Residual) and $|\hat{\mathbf{x}}^H\mathbf{J}\hat{\mathbf{x}}/\hat{\mathbf{x}}^H\mathbf{B}\hat{\mathbf{x}} - \hat{\lambda}|$ (Rayleigh Quotient Error) where $\hat{\mathbf{x}}$ ($\|\hat{\mathbf{x}}\| = 1$) and $\hat{\lambda}$ are the approximate eigenvector and eigenvalue computed. In the case of complex conjugate pairs of eigenvalues, the successive rows list the real and imaginary parts of the vector and scalar quantities.

Calculations on this and similar systems showed that the residual reduction in the linear solver had to be near or below 10^{-10} to get accurate eigenvalues. Considering that the linear solves in the steady state calculation required over 200 GMRES iterations (no restarting) just to reach a 10^{-4} residual reduction, it was a concern that requiring 10^{-10} of the linear solver in the eigenvalue calculations was going to severely limit the algorithm. However, solving the Cayley systems proved to be much easier than solving the untransformed system for the steady state. This is seen in the above results, where it took 40 seconds to invert $\mathbf{J}$ to a tolerance of 10^{-4} in the steady state calculation and only 20 seconds to get 10^{-10} reduction when $\sigma = 100$ and $\mu = 1000$. As explained in Section 3 the Cayley transformed system maps most of the eigenvalues near 1, so that

Table 1. The 7 leading eigenvalues and the residuals are presented. The calculation for a system of rank 233925 required 1800 seconds on 200 Processors. The tolerance used for the eigensolver was 10^{-5}.

Real(λ)	Im(λ)	Residual	RQ errors
-13.867	0.0000	$9.3e-5$	$2.1e-7$
-15.908	10.144	$2.4e-5$	$3.0e-7$
-15.908	-10.144	$3.7e-5$	$1.3e-6$
-23.685	4.513	$4.8e-4$	$3.7e-6$
-23.685	-4.513	$1.7e-4$	$5.2e-6$
-24.137	21.809	$5.8e-5$	$7.5e-6$
-24.137	-21.809	$2.2e-4$	$7.2e-6$

the spectral condition number of the matrix is bounded. It turns out that the spectral condition number of the system $(\mathbf{J}-\sigma\mathbf{B})^{-1}(\mathbf{J}-\mu\mathbf{B})$ is just 8.9 for the initial Cayley transform and 11.0 for the second.

Another interesting algorithmic detail that proved important was the choice of orthogonalization used in the GMRES algorithm. Originally, a classical Gram-Schmidt scheme was used, but the lack of numerical stability prevented the GMRES algorithm from reaching the required 10^{-10} tolerance. Two alternative orthogonalization schemes were used successfully: two-step classical Gram-Schmidt (CGS) and a modified Gram-Schmidt (MGS). The CGS method uses two steps of orthogonalization (the second step is the correction for loss of orthogonality for the Arnoldi basis vectors) but the number of global communication points remains fixed (at two) independent of the GMRES iteration. On the other hand, the MGS scheme requires i communications to orthogonalize i vectors (at GMRES iteration i) but no additional floating point operations (flops). Both schemes reached the 10^{-10} tolerance and provided identical results in terms of the number of GMRES iterations needed.

There was a significant difference in the scalability of the two algorithms as the number of processors was changed. The time required to perform a single linear solve (using a pre-calculated preconditioner) was recorded for the same problem as above for the case of $\sigma = 10$ and $\mu = 50$, with the number of processors being varied from 50 to 500. The results of this calculation are most clear when presenting the total CPU time (calculated as the wall clock time multiplied by the number of processors) as shown in Figure 1. When the problem was run on 50 processors, the extra communications required by MGS were less expensive than the extra flops required by the CGS algorithm. However, as the number of processors is increased up to 500, it is seen that the communication time in MGS starts to dominate, while the total CGS time remains relatively flat. At 500 processors, the CGS routine uses less than half of the time of the MGS method.

The results in Figure 1 show that the two-step classical Gram-Schmidt (CGS) scheme scales much better than the modified Gram-Schmidt (MGS) scheme. It

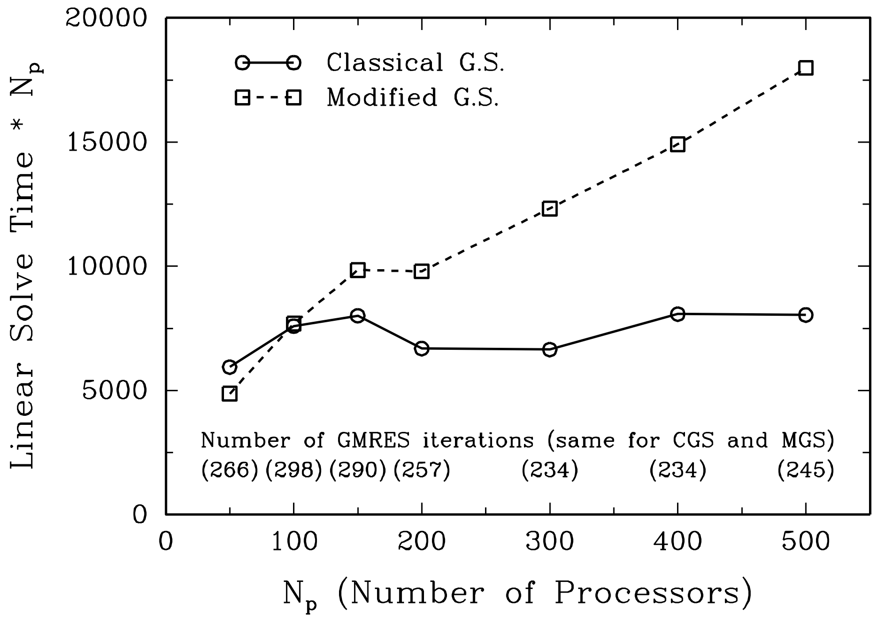

Fig. 1. Comparing the parallel efficiency of orthogonalization schemes used for GMRES. The number of GMRES iterations needed to to solve the linear systems to the specified tolerance using the given number of processors is shown in parenthesis.

should be pointed out that the inter-processor communication rate of the Sandia-Intel Tflop computer is very fast compared to more loosely coupled parallel machines, where we would expect the crossover point to be at significantly fewer than 100 processors as seen here.

In each case, the number of GMRES iterations is shown. The surprising result is that the number of GMRES iterations does not increase monotonically with the number of processors (and hence domains in the ILU(0) preconditioner). This is currently not understood.

5 Conclusions

The preliminary results presented here show that the algorithms can be used to successfully perform linear stability analysis on reacting flow systems of $\mathcal{O}(10^5)$ on massively parallel computers.

Our results showed that a judicious choice of a Cayley transformation along with a preconditioned GMRES iterative solver for the resulting linear systems allowed the P_ARPACK to compute accurate eigenvalues. Also crucial for the scalability of GMRES was the use of a two-step classical Gram-Schmidt scheme for the necessary orthogonalizations.

Acknowledgments

The authors would like to thank David Day, Louis Romero, Beth Burroughs, John Shadid, and Ray Tuminaro for their contributions to our understanding and the implementation of the algorithms as well as some of the text of this paper.

References

1. N. Alleborn, K. Nandakumar, H. Raszillier, and F. Durst. Further contributions on the two-dimensional flow in a sudden expansion. *Journal of Fluid Mechanics*, 330:169–188, 1997.
2. W. G. Breiland and Greg H. Evans. Design and verification of nearly ideal flow and heat transfer in a rotating disk chemical vapor deposition reactor. *Journal of the Electrochemical Society*, 138:1806–1816, 1991.
3. K. N. Christodoulou and L. E. Scriven. Finding leading modes of a viscous free surface flow: An asymmetric generalized eigenproblem. *Journal of Scientific Computing*, 3:355–406, 1988.
4. K. A. Cliffe, T. J. Garratt, and A. Spence. Eigenvalues of the discretized navier-stokes equation with application to the detection of hopf bifurcations. *Advances in Computational Mathematics*, 1:337–356, 1993.
5. K. D. Devine, G. L. Hennigan, S. A. Hutchinson, A. G. Salinger, J. N. Shadid, and R. S. Tuminaro. High performance mp unstructured finite element simulation of chemically reacting flows. In *Proceedings of SC97*, 1997.
6. Anne Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM, Philadelphia, PA., 1997.
7. B. Hendrickson and R. Leland. The Chaco user's guide: Version 2.0. Technical Report SAND94–2692, Sandia National Labs, Albuquerque, NM, June 1995.
8. J. R. Hughes, L. P. Franca, and M. Balestra. A new finite element formulation for computational fluid dynamics: Circumventing the Babuska–Brezzi condition: A stable Petrov–Galerkin formulation of the Stokes problem accommodating equal-order interpolation. *Comp. Meth. App. Mech. and Eng.*, 59:85–99, 1986.
9. S. A. Hutchinson, L. V. Prevost, R. S. Tuminaro, and J. N. Shadid. Aztec user's guide: Version 2.0. Technical report, Sandia National Laboratories, Albuquerque, NM, 1998.
10. R. B. Lehoucq and J. A. Scott. Implicitly restarted Arnoldi methods and eigenvalues of the discretized Navier-Stokes equations. Technical Report SAND97-2712J, Sandia National Laboratories, Albuquerque, New Mexico, 1997.
11. R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK USERS GUIDE: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.* SIAM, Phildelphia, PA, 1998.
12. K. J. Maschhoff and D. C. Sorensen. P_ARPACK: An efficient portable large scale eigenvalue package for distributed memory parallel architectures. In Jerzy Wasniewski, Jack Dongarra, Kaj Madsen, and Dorte Olesen, editors, *Applied Parallel Computing in Industrial Problems and Optimization*, volume 1184 of *Lecture Notes in Computer Science*, Berlin, 1996. Springer–Verlag.
13. T. G. Mattson and G. Henry. An overview of the Intel TFLOPS supercomputer. *Intel Technology Journal*, 1, 1998.
14. K. Meerbergen, A. Spence, and D. Roose. Shift-invert and Cayley transforms for the detection of rightmost eigenvalues of nonsymmetric matrices. *BIT*, 34:409–423, 1994.

15. Karl Meerbergen and Alastair Spence. Implicitly restarted Arnoldi with purification for the shift–invert transformation. *Mathematics of Computation*, 218:667–689, 1997.
16. Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS, Boston, MA, 1996.
17. Y. Saad and M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, July 1986.
18. A. G. Salinger, K. D. Devine, G. L. Hennigan, H. K. Moffat, S. A. Hutchinson, and J. N. Shadid. Mpsalsa: a finite element computer program for reacting flow problems part 1 - user's guide. Technical Report SAND95–2331, Sandia National Laboratories, Albuquerque, NM, 1996.
19. A. G. Salinger, J. N. Shadid, S. A. Hutchinson, G. L. Hennigan, K. D. Devine, and H. K. Moffat. Analysis of gallium arsenide deposition in a horizontal chemical vapor deposition reactor using massively parallel computations. Technical Report SAND98–0242, Sandia National Laboratories, Albuquerque, NM, 1998.
20. J. N. Shadid, H. K. Moffat, S. A. Hutchinson, G. L. Hennigan, K. D. Devine, and A. G. Salinger. Mpsalsa: a finite element computer program for reacting flow problems part 1 - theoretical development. Technical Report SAND95–2752, Sandia National Laboratories, Albuquerque, NM, 1996.
21. D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Analysis and Applications*, 13(1):357–385, January 1992.