Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computational Physics 205 (2005) 24–47

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

Performance of fully coupled domain decomposition preconditioners for finite element transport/reaction simulations ☆

J.N. Shadid, R.S. Tuminaro, K.D. Devine, G.L. Hennigan, P.T. Lin *

Sandia National Laboratories, Computational Sciences, P.O. Box 5800, MS 0316, Albuquerque, NM 87185-0316, USA

Received 11 May 2004; received in revised form 26 October 2004; accepted 26 October 2004
Available online 10 December 2004

Abstract

In this paper, we describe an iterative linear system solution methodology used for parallel unstructured finite element simulation of strongly coupled fluid flow, heat transfer, and mass transfer with nonequilibrium chemical reactions. The nonlinear/linear iterative solution strategies are based on a fully coupled Newton solver with preconditioned Krylov subspace methods as the underlying linear iteration. Our discussion considers computational efficiency, robustness and a number of practical implementation issues. The evaluated preconditioners are based on additive Schwarz domain decomposition methods which are applicable for totally unstructured meshes. A number of different aspects of Schwarz schemes are considered including subdomain solves, use of overlap and the introduction of a coarse grid solve (a two-level scheme). As we will show, the proper choice among domain decomposition options is often critical to the efficiency of the overall solution scheme. For this comparison we use a particular spatial discretization of the governing transport/reaction partial differential equations (PDEs) based on a stabilized finite element formulation. Results are presented for a number of standard 2D and 3D computational fluid dynamics (CFD) benchmark problems and some large 3D flow, transport and reacting flow application problems.
© 2004 Elsevier Inc. All rights reserved.

Keywords: Newton–Krylov; Fully coupled solvers; Schwarz domain decomposition; Two-level methods; Multilevel methods; Parallel methods; Stabilized finite element methods

☆ This work was partially funded by the US Department of Energy's Mathematical Information and Computational Sciences Division and the Department of Homeland Security, Advanced Scientific Computing Program, and was carried out at Sandia National Laboratories. 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-ACO4-94AL85000.

* Corresponding author.

E-mail addresses: jnshadi@sandia.gov (J.N. Shadid), ptlin@sandia.gov (P.T. Lin).

0021-9991/$ - see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2004.10.038

1. Introduction

Modern computational fluid dynamics simulations often require the solution of strongly coupled interacting physics in complex three-dimensional (3D) geometries with high resolution unstructured meshes to capture all the relevant length scales. After suitable spatial discretization and linearization, these simulations can produce large linear systems of equations with on the order 10^5–10^8 unknowns. As a result efficient and robust parallel iterative solution methods are required to make such simulations tractable for use in analysis or in engineering design cycle times. Preconditioned Krylov iterative methods are among the most robust and fastest iterative solvers over a wide variety of CFD applications [9,20,31,24,36,39]. In the last decade, there has been a significant amount of work on parallel Krylov methods, and a number of general purpose Krylov solver libraries have been developed [10,15,21]. In general, these Krylov methods are relatively straightforward to implement, highly parallel, and are often "optimal" in some sense. While the convergence characteristics of specific Krylov methods remains a topic of research interest, it is now clear that the key factor influencing the robustness and efficiency of these solution methods is preconditioning.

The focus of this study is to evaluate several different domain decomposition preconditioner variants for the computational solution of incompressible and low Mach number variable density reacting and nonreacting fluid flows with unstructured mesh finite element methods. These flow problems are characterized by both locally elliptic and nearly hyperbolic behavior, localized steep gradients, and often strongly coupled interactions between the flow velocities, hydrodynamic pressure, temperature and chemical species. This strong coupling results from the nonlinear transport terms in the governing PDEs as well as the localized chemical reaction source terms (see Table 1). The discussion and comparison of the proposed methods are framed in the context of steady-state solutions to the governing flow and transport equations. This context provides a more numerically challenging comparison since the absence of the transient terms (in general) produces a less well conditioned system of equations. The evaluated preconditioners fall into the family of Schwarz domain decomposition methods [3,4,10,26]. These schemes partition the original domain into subdomains and approximately solve the discrete problems corresponding to the individual subdomains in parallel. Among Schwarz schemes, there are a number of choices which can greatly affect the overall solution time and robustness. These choices include the subdomain size, the amount of overlap between

Table 1
Governing transport/reaction PDEs

Momentum	$\mathbf{R_m} = \rho \frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u} \cdot \nabla \mathbf{u}) - \nabla \cdot \mathbf{T} - \rho g$
Total mass	$\mathrm{R_P} = \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u})$
Thermal energy	$\mathrm{R}_T = \rho \hat{C}_p \left[\frac{\partial T}{\partial t} + \mathbf{u} \cdot \nabla T\right] + \nabla \cdot \mathbf{q}_c - \phi - \dot{Q} + \sum_{k=1}^{N_s} \mathbf{j}_k \cdot \hat{C}_{p,k} \nabla T + \sum_{k=1}^{N_s} h_k W_k \dot{\omega}_k + \nabla \cdot \mathbf{q_r}$
Species mass fraction for species k	$\mathrm{R}_{Y_k} = \rho \left[\frac{\partial Y_k}{\partial t} + \mathbf{u} \cdot \nabla Y_k\right] + \nabla \cdot \mathbf{j}_k - W_k \dot{\omega}_k, \quad k = 1,2,\ldots,N_s$

subdomains, and the partitioning metric which can alter the shape and aspect ratio of subdomains (see, e.g. [7,10–12,26]). The choices also include the selection of subdomain solver such as an incomplete LU factorization (ILU) (with further options for dropping nonzeros in the factorizations and ordering equations within a subdomain [1]), and the introduction of a coarse grid solve [9,35]. Therefore among these issues we explore the effects of the choice of subdomain solver, the amount of subdomain overlap, and the effect of exact and inexact coarse grid solvers for the two-level methods. The numerical studies include a variety of fluid flow, transport, and a challenging reacting flow application. As will be demonstrated, the proper choice among domain decomposition options is often critical to the efficiency of the overall solution scheme.

The remainder of the paper is organized as follows. After presenting the transport equations in Section 2, a brief overview of the parallel Newton–Krylov methodology is presented in Section 3 with a presentation of the domain decomposition preconditioners. Section 4 briefly describes the problems used to generate the numerical examples to evaluate the solution methods. The results of this comparison study are then presented in Section 5. Section 6 presents a brief discussion of the parallel efficiency of these methods. Application of the methods to large-scale problems is presented in Section 7. Finally, in Section 8, a number of conclusions are drawn.

2. The governing equations and numerical formulation

The governing transport PDEs describing fluid flow, thermal energy transfer, mass transfer and non-equilibrium chemical reactions are presented in Table 1 in residual form. In these equations, the unknown quantities are $\boldsymbol{u}$, P, T and Y_k; these are, respectively, the fluid velocity vector, the hydrodynamic pressure, the temperature, and the mass fraction for species k. The equations include constitutive relations for a Newtonian stress tensor $\mathbf{T}$, the Fourier law for the heat flux vector $\mathbf{q}_c$, and the Fickian diffusion fluxes $\mathbf{j}_k$. Additional quantities are the density, ρ, and the specific heat at constant pressure, C_p. W_k is the molecular weight of species k, $\dot{\omega}_k$ is the volumetric reaction source term for species k, and h_k is the enthalpy of formation for species k. In the heat equation ϕ is the volumetric heat source term from viscous dissipation, $\dot{Q}$ is a volumetric source term and $\mathbf{q}_r$ is the radiation heat flux vector (see [27] for more details). In the stabilized FE formulation of the equations, Φ is a generic FE basis function, in our case these are linear basis functions on quads (2D) and hex elements (3D). These residual definitions are used in the subsequent brief discussion of the discretization technique based on a stabilized finite element (FE) formulation.

The continuous problem, defined by the transport equations, is approximated by a stabilized finite element formulation [14,26,30,33]. This formulation allows for equal order interpolation of pressure and velocity (without spurious pressure solutions), and for stabilization of highly convected flows. The resulting stabilized FE equations are shown in Table 2.

The stabilization parameters τ_m, τ_T and τ_{Y_k} are given in [14,30,33]. For clarity in our later discussion of the solution methods and linear algebra, the Newtonian stress tensor $\mathbf{T}$ is expanded to include the pressure P and the viscous stress tensor term Υ. The resulting stabilized FE total mass residual equation in expanded form is given in the following equation:

$$F_{\mathrm{P}} = \int_{\Omega} \Phi\left(\frac{\partial\rho}{\partial t} + \nabla\cdot(\rho\mathbf{u})\right)\mathrm{d}\Omega + \int_{\Omega_e} \rho\tau_m\nabla\Phi\cdot\left[\rho\frac{\partial\mathbf{u}}{\partial \mathrm{t}} + \rho\mathbf{u}\cdot\nabla\mathbf{u} + \nabla P - \nabla\cdot\Upsilon - \rho \mathrm{g}\right]\mathrm{d}\Omega. \tag{1}$$

This expansion exhibits the weak form of a Laplacian operator acting on pressure

$$\int_{\Omega_e} \rho\tau_m\nabla\Phi\cdot\nabla P\,\mathrm{d}\Omega \tag{2}$$

Table 2
Stabilized FE formulation of transport PDEs

Momentum	$F_{\mathrm{m},i} = \int_\Omega \Phi \mathrm{R}_{\mathrm{m},i} \mathrm{d}\Omega + \int_{\Omega_e} \rho\tau_m(\mathbf{u}\cdot\nabla\Phi)\mathrm{R}_{\mathrm{m},i}\mathrm{d}\Omega$
Total mass	$F_\mathrm{P} = \int_\Omega \Phi \mathrm{R}_\mathrm{P} \mathrm{d}\Omega + \int_{\Omega_e}(\rho\tau_m\nabla\Phi\cdot\mathbf{R_m})\mathrm{d}\Omega$
Thermal energy	$F_T = \int_\Omega \Phi \mathrm{R}_T \mathrm{d}\Omega + \int_{\Omega_e} \rho\hat{C}_p\tau_T(\mathbf{u}\cdot\nabla\Phi)\mathrm{R}_T\mathrm{d}\Omega$
Species mass fraction for species k	$F_{Y_k} = \int_\Omega \Phi \mathrm{R}_{Y_k} \mathrm{d}\Omega + \int_{\Omega_e} \rho\tau_{Y_k}(\mathbf{u}\cdot\nabla\Phi)\mathrm{R}_{Y_k}\mathrm{d}\Omega, \quad k = 1,2,\ldots,N_s$

produced by the stabilized FE formulation of the total mass conservation equation. In addition to the pressure stabilization provided by this operator, this term also enables the development of effective Krylov based solvers with various preconditioners [24,26]. Finite element (FE) discretization of the stabilized FE equations gives rise to a system of coupled, nonlinear, nonsymmetric algebraic equations, the numerical solution of which can be very challenging. These equations are linearized using an inexact form of Newton's method [6]. A formal block matrix representation of these discrete linearized equations is given in Eq. (3) where the block diagonal contribution of the stabilization procedure has been highlighted by a specific ordering. In this representation, the vector $\mathbf{v}'$ contains the Newton updates to all the nodal solution variables with the exception of the nodal pressures P'. The block matrix A corresponds to the combined discrete convection, diffusion and reaction operators for all the unknowns; the matrix B corresponds to the discrete divergence operator with its transpose the gradient operator; the diagonal matrix R results from the group FE expansion of the density and velocity; and the matrix K corresponds to the discrete "pressure Laplacian" operator discussed above. The vectors $F_\mathbf{v}$ and $F_\mathbf{P}$ contain the right-hand side residuals for Newton's method.

The existence of the well-conditioned nonzero matrix K in the stabilized FE discretization of the equations allows the solution of the linear systems with a number of algebraic and domain decomposition type preconditioners. This is in contrast to other formulations, such as Galerkin methods using mixed interpolation, that produce a zero block on the total mass continuity diagonal. The difficulty of producing robust and efficient preconditioners for the Galerkin formulation has motivated the use of many different types of solution methodologies. A number of these use two-level iteration schemes, penalty methods, pseudo-compressibility techniques or decoupled/segregated solvers [36,39]. A detailed presentation of the characteristics of current solution methods is far beyond the scope of this manuscript. However, the intent of our method of fully coupling the transport PDEs in the nonlinear solver is to preserve the inherently strong coupling of the physics with the goal to produce a more robust solution methodology. Preservation of this strong coupling, however, places a significant burden on the linear solution procedure to solve the fully coupled algebraic systems.

$$\begin{bmatrix} A & -B^\mathrm{T} \\ BR & K \end{bmatrix} \begin{bmatrix} \mathbf{v}' \\ \mathrm{P}' \end{bmatrix} = \begin{bmatrix} -F'_\mathrm{v} \\ -F'_\mathrm{P} \end{bmatrix}. \tag{3}$$

Our current linear algebra solution procedure uses a specific ordering of the unknowns locally at each FE node with each degree of freedom ordered consecutively ([u, v, w, P, T, Y_i]). A single coupled matrix problem, $Js;= -F$, is solved at each Newton step with sophisticated algebraic domain decomposition and multilevel preconditioned Krylov methods to solve this system as described below.

3. Overview of parallel Newton–Krylov implementation

In this section, a brief overview of the parallel numerical solution procedure is presented. This discussion provides context for the evaluation of the domain decomposition (DD) preconditioners. References are provided to more complete sources on each of the topics.

3.1. Problem partitioning

Chaco [13], a general graph partitioning tool, is used to partition the FE mesh into subdomains and make subdomain-to-processor assignments. Chaco constructs partitions and subdomain mappings that have low communication volume, good load balance, few message start-ups and only small amounts of network congestion. It supports a variety of new and established graph partitioning heuristics. For the results in this paper, multilevel methods with Kernighan–Lin improvement were used. For a detailed description of parallel FE data structures and a discussion of the strong link between partitioning quality and parallel efficiency, see [9,26].

3.2. Newton–Krylov methods

A Newton–Krylov method [2,16,25] is an implementation of Newton's method in which a Krylov iterative solution technique is used to approximately solve the linear systems that are generated at each step of Newton's method. Specifically, to solve the nonlinear system $\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0}$, we seek a zero of F: $\boldsymbol{R}^n \to \boldsymbol{R}^n$ where $\boldsymbol{x} \in \boldsymbol{R}^n$ is a current approximate solution. The Krylov iterative solver is applied to determine an approximate solution of the Newton equation

$$\boldsymbol{J}(\boldsymbol{x})\boldsymbol{s} = -\boldsymbol{F}(\boldsymbol{x}), \tag{4}$$

where $\boldsymbol{J}(\boldsymbol{x})$ is the Jacobian matrix of $\boldsymbol{F}$ at $\boldsymbol{x}$. A Newton–Krylov method is usually implemented as an inexact Newton method [6,28]. That is, in approximately solving Eq. (4), one chooses a forcing term $\eta \in [0,1)$ and then applies a Krylov method until an iterate $\boldsymbol{s}_k$ satisfies the inexact Newton condition

$$\|\boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{J}(\boldsymbol{x})\boldsymbol{s}_k\| \leqslant \eta\|\boldsymbol{F}(\boldsymbol{x})\|. \tag{5}$$

Intuitively one would assume that in the initial stages of the Newton iteration, when the current approximation is far from the true solution, there would be no benefit from solving too accurately the Newton equations with the inaccurate Jacobian matrix $\boldsymbol{J}(\mathbf{x}_k)$ that is currently available. Normally our inexact Newton method formulation uses an adaptive convergence criteria to reduce the amount of over-solving that occurs and thereby to produce a more computationally efficient nonlinear solution procedure. To improve robustness, a back-tracking algorithm can be used. This globalization method selects an update vector s_k by scaling a Newton step as needed to ensure that the nonlinear residual has been reduced adequately before the step is accepted. The details of this inexact Newton implementation can be found in [28]. In the discussion of results that follows we attempt to separate the issues associated with the nonlinear solution methods and the linear systems solved by preconditioned Krylov methods. In this context we select a constant, moderately small, value for the convergence criteria. Unless it is stated otherwise, we chose the convergence criteria to be 10^{-4} to focus on the role of the preconditioners in the linear solver rather than allow adaptive selection of the criteria.

3.3. Parallel preconditioned Krylov implementation

The linear subproblems generated from the inexact Newton method are solved by preconditioned Krylov methods as implemented in our Aztec parallel iterative solver library [15]. The Krylov algorithms implemented in Aztec include techniques such as the restarted generalized minimal residual [GMRES(k)] and transpose-free quasi-minimal residual techniques for nonsymmetric systems. All Krylov methods rely on a small, well defined set of basic kernel routines. These kernel routines consist of parallel matrix–vector, vector–vector, vector inner-product, and preconditioning operations. In our previous papers [24,26,29], we discussed in detail the implementation and parallel efficiency of these Krylov kernel routines with the exception of the preconditioners. In this paper, we focus on the preconditioner issues. To focus on the role of the preconditioning technique we present results only with the restarted GMRES(k) [22] Krylov solver (as these results are representative for the other solvers as well).

It is well known that the overall performance of Krylov methods can be substantially improved when one uses preconditioning [22]. The basic idea is that instead of solving the system $Ax = b$, the system $AM^{-1}y = b$ is solved, where M^{-1} is an approximation to A^{-1} and is easily computed, Since only matrix–vector products are needed, it is not necessary to explicitly form AM^{-1} (only software to solve $Mv = y$ is needed). We note that the preconditioning described here corresponds to "right" preconditioning; it is also possible to precondition on the "left" (i.e., $M^{-1}A$). In this paper, only right preconditioning is considered as the comparisons are more straightforward. Specifically, when left preconditioning is used the computed residual corresponds to a preconditioned residual. Thus, if convergence is based on the size of the residual, changing the preconditioner effectively changes the convergence criteria.

The preconditioners that are considered are based on algebraic additive Schwarz domain decomposition (DD) preconditioners with variable overlapping between subdomains. For comparison purposes, we also include some brief results based on classical iterative methods (Jacobi, block Jacobi and polynomial expansions) as preconditioners. More details of these preconditioners can be found in [24,26,29].

3.4. Additive Schwarz domain decomposition preconditioner

A formal description of the variable overlap additive Schwarz preconditioner can be described by considering the following linear system:

$$Au = f, \tag{6}$$

where A is an $n \times n$ nonsymmetric matrix and the matrix entry in the ith row and jth column is given by a_{ij}. This matrix induces a directed graph which can be defined in the following way. Each row of A corresponds to a vertex and each $a_{ij} \neq 0$ corresponds to an edge incident from node i to j. We denote the set of graph vertices by $V(A)$ and similarly the set of edges by $E(A)$. Throughout the rest of this discussion, the argument A in $V(A)$ and $E(A)$ will be dropped to facilitate the presentation. The set of edges and vertices defines a matrix graph $G(V,E)$.

Domain decomposition methods rely on approximate solutions on subdomains. These subdomains are defined in terms of vertex subsets. To discuss vertex subsets we use the notation V_k^i to denote the kth subdomain in an i-overlap method. For now, let us assume that V_k^i is defined and corresponds to a subset of vertices. The following edge set can be associated with the vertex subset

$$E_k^i = \{e_j = (x,y) \in E \mid x \in V_k^i, y \in V\}. \tag{7}$$

Intuitively, E_k^i includes all edges emanating from V_k^i. To complete the definition of the Schwarz method, we must now define the vertex subsets. Assume that the vertices have been partitioned into p disjoint sets V_i^0 such that

$$V = \bigcup_{i=1}^{p} V_i^0 \tag{8}$$

and

$$V_i^0 \cap V_j^0 = \emptyset \quad \forall i \neq j. \tag{9}$$

This vertex partitioning corresponds to the distribution of the matrix over the processors via Chaco and effectively defines the 0th-overlap subdomains. To define the k-overlap subdomains, we use the edge sets associated with the $(k-1)$th-overlap subdomains:

$$V_k^i = \{x \in V \mid \exists y \in V, \quad \text{where } (x,y) \text{ or } (y,x) \in E_k^{i-1}\}. \tag{10}$$

To define overlap in an ith-overlap additive Schwarz method, we use the vertex sets V_k^i. Specifically, consider the restriction matrix I_k^i of size mxn where m is the number of nodes in V_k^i, n is the total number of nodes in V, and

$$I_k^i(l,j) = \begin{cases} 1 & j \text{ is the } \quad l\text{th node in } V_k^i, \\ 0 & \text{otherwise,} \end{cases} \quad 1 \leqslant k \leqslant p. \tag{11}$$

These I_k^i operators essentially map from the entire space to the kth subdomain. The ith-overlap additive Schwarz preconditioner is now given by

$$M^{-1} = \sum_{k=1}^{p} I_k^i((I_k^i)^{\mathrm{T}} A I_k^i)^{-1} (I_k^i)^{\mathrm{T}}. \tag{12}$$

This method corresponds to projecting the equations onto a series of overlapping subdomains defined by the vertex sets and solving each subsystem. Since these subdomain solves are independent, they can be performed concurrently. Intuitively overlapping can be view as means to increase robustness by expanding individual subdomains (to include FE nodes assigned to neighboring processors) by allowing more coupling between subdomains (processors). In a geometric sense, this overlap corresponds to increasing the size of the locally defined subdomain to include additional levels of FE nodes outside of the processor's assigned nodes. Thus a single level of overlapping uses only information from FE nodes that are connected by an edge (in the FE connectivity graph) that was cut by the original subdomain partition. Successive levels of overlap now use this method recursively by considering the previously overlapped points to now be *assigned* nodes to the subdomain. As described this method would be referred to as a one-level scheme. A two-level scheme uses not only the fine grid operator defined above but also adds an additional projection of the original equations onto a coarser grid. That is, a two-level domain decomposition method is given by

$$M^{-1} = \sum_{k=0}^{p} I_k^i((I_k^i)^{\mathrm{T}} A I_k^i)^{-1} (I_k^i)^{\mathrm{T}}, \tag{13}$$

where I_k^i is defined as above for $k > 0$ and I_0^i is an interpolation operator that maps solution vectors from the original FE mesh to an auxiliary coarser mesh that covers the same domain as the original but with significantly fewer grid points. Theoretically, the number of mesh points should be about the same size as the number of subdomains. When A is a symmetric positive definite discrete elliptic operator and a sufficient amount of overlap is used, the iterative method convergence using a domain decomposition preconditioner is independent of the number of unknowns in the matrix [32]. In cases where more modest overlap is used, the theoretical convergence depends mildly on the size of the subdomains. In the case where A is nonsymmetric, results have been obtained for single PDE systems [3]. Much less is known about coupled systems of nonsymmetric PDEs. It is important to notice that with the addition of the coarse grid solve, the domain decomposition method is no longer completely algebraic.

A number of practical simplifications can be made to the preconditioner described by Eq. (13). In particular, we rewrite the preconditioner as

$$M^{-1} = \sum_{k=0}^{p} I_k^i \hat{A}_k^{-1} (I_k^i)^{\mathrm{T}}, \tag{14}$$

where for $\hat{A}_0$ we could use an independent discretization of the PDE problem on a coarse mesh and for the $\hat{A}_k^{-1}$ an approximate solver method on each subdomain can be used. In our experiments, we use Aztec [15] to implement the one-level Schwarz method. Aztec automatically constructs the overlapping submatrices. While a direct factorization could be used on the subdomains, our experience indicates that this is rarely practical as the storage and time associated with this direct factorization is too high. Instead of solving the submatrix systems exactly we use an incomplete factorization technique on each subdomain (processor). In this paper, we use two specific ILU factorizations: the standard ILU(0) method with no fill-in as well as the ILUT(fill-in,drop) incomplete factorization [19] which allows specification of a user-specified fill-in parameter (fill-in $\geqslant$ 1.0) and a drop tolerance. In this nomenclature, a fill-in of 1.5 denotes an ILU factor with up to 1.5 times as many nonzeros as the original matrix. In the examples presented in Section 5, there are no entries dropped due to magnitude (drop = 0.0).

The ML multilevel solver library [35] is used to implement the coarse grid solve of the two-level domain decomposition method. There are several possibilities for generating I_0^i including completely algebraic techniques. In this paper, we use a technique based on the standard finite element basis functions. In particular, the user supplies two meshes: the original fine grid FE mesh and a coarser grid FE mesh. Additionally, users supply a call back function so that ML can evaluate the user's coarse grid FE basis functions. Using this call back function, ML constructs a grid transfer operator that corresponds to the interpolant associated with the FE basis functions. The $\hat{A}_0$ matrix can be generated by the matrix–matrix multiplication $(I_0^i)^{\mathrm{T}} A I_0^i$ or the user can supply it. In this paper, $\hat{A}_0$ is supplied by the application and corresponds to a finite element discretization on the coarser mesh.

4. Numerical experiments

To illustrate the robustness, convergence and efficiency of the fully coupled domain decomposition preconditioners, we present representative results for a number of 2D benchmark CFD type simulations as well as some more challenging 3D CFD simulations and a challenging 3D reacting flow problem. The important features of each of these test problems are summarized in Table 3. A more detailed description of each of these problems is given in Appendix A. The example problems include straightforward Stokes flow solutions to the momentum and continuity equations (without the convection operators); Navier–Stokes solutions to the momentum and continuity equations; thermal convection flows of the momentum,

Table 3
Summary of numerical example problems

Example problem	Description	Equations solved
1	2D Thermal convection in a square	Momentum, total mass, thermal energy
2	3D Thermal convection in a cube	Momentum, total mass, thermal energy
3	2D Lid driven cavity flow	Momentum, total mass
4	2D Internal channel flow about an obstruction	Momentum, total mass (Stokes, Navier–Stokes)
5	3D chemical vapor deposition (CVD) reactor for GaAs deposition	Momentum, total mass, thermal energy, mass species – no reactions
6	2D and 3D fluid flow and chemical agent transport in a building	Momentum, total mass, mass species – no reactions
7	3D CVD reactor for poly-silicon deposition	Momentum, total mass, thermal energy, mass species – with reactions

continuity, and the thermal energy equations; and full reacting flow simulations of the momentum, continuity, thermal energy and mass species equations with chemical reaction source terms. The corresponding FE discretizations of these problems include both structured and unstructured FE meshes. In general, the structured mesh examples correspond to a number of well accepted 2D and 3D benchmark CFD problems and provide a straightforward means of systematically increasing problems sizes for studying the robustness, algorithmic scaling (convergence rate) and parallel performance of the DD preconditioners. These results are then verified with the unstructured mesh examples that we present.

In the numerical studies comparing the one-level DD preconditioners and the block Jacobi and polynomial preconditioners, GMRES uses a restart value of 200, which was sufficiently large that GMRES stagnation did not become an issue for even the most difficult of the linear subproblems generated by the inexact Newton algorithm. We also allowed a maximum of 600 GMRES iterations at each inexact Newton step, after which the GMRES iterations were terminated and a new inexact Newton step started even if the convergence criteria did not hold. However, in cases where algorithmic scaling studies were carried out for the one- and two-level methods we employed nonrestarted GMRES to isolate the growth in iterations due to preconditioner performance as opposed to restarting effects. In these cases, the tables indicate that nonrestarted GMRES was used. In all cases, unless otherwise noted, the initial approximate solution was the zero vector.

5. Results and discussion

5.1. An illustration of preconditioner choice and robustness

To illustrate the favorable robustness properties of DD preconditioners over classical iterative schemes applied to fully coupled systems, we consider results for two 2D CFD benchmark problems (Example Problems 1 and 3). Each of these example problems is characterized by a nondimensional parameter that essentially controls the strength of the convection transport mechanism to the diffusion process. As the Rayleigh number *Ra* and the Reynolds number *Re* are increased, the nonsymmetry and the nonlinearity of the problem increases. In our results, we restrict our experiments to ranges of these parameters where the effects of the linear solvers can be more clearly separated from the nonlinear solvers [25,28]. In Table 4, we present a

Table 4
Comparison of various preconditioners on thermal convection Example Problem 1

Preconditioner	Mesh	*Ra*			
		10^2	10^3	10^4	10^5
DD&ILUT(1.0,0.0)	32×32	4/251/10.1	5/327/12.8	7/451/17.6	9/562/22.9
	64×64	4/470/45.6	5/748/66.0	7/1251/104.3	11/1701/147.5
	100×100	5/2265/349.2	5/2425/367.0	9/4211/644.4	11/4333/683.0
1 Step block –Jacobi	32×32	11/6316/106.7	Failed	Failed	Failed
	64×64	Failed	Failed	Failed	Failed
3 Step block –Jacobi	32×32	5/846/30.4	6/1386/47.5	8/1940/66.4	11/2396/82.8
	64×64	12/6807/656.5	Failed	Failed	Failed
5 Step block – Jacobi	32×32	4/327/18.4	6/455/25.6	8/662/36.7	11/966/53.2
	64×64	11/5977/854.1	10/5243/753.1	Failed	Failed
3 Order LS – Poly	32×32	5/1810/60.8	11/5307/174.6	Failed	Failed

Results: Newton steps/linear its./CPU time (s).

comparison of the number of Newton steps, the total number of restarted GMRES(128) iterations and the time to solution for Example Problem 1 for three choices of preconditioners. These results compare two classical iterative schemes used as preconditioners, a multi-step block Jacobi, a least squares polynomial expansion (order = 1), and a one-level domain decomposition preconditioner that uses the ILUT preconditioner within each processor subdomain with one level of subdomain overlap and no fill-in. The block Jacobi method uses the Jacobian entries that correspond to all of the unknown at a particular node of the FE mesh. The table presents the effect of increasing nonsymmetry (increasing *Ra*) and increasing mesh size. From these results, it is clear that the simple multi-step block-Jacobi preconditioner is not robust enough to solve the problem from a simple zero initial guess for the full range of *Ra* and meshes. The preconditioner based on a least squares polynomial expansion lacks robustness as well. While these two preconditioners are straightforward to implement in parallel, have low memory requirements and achieve high computational rates [26], the robustness for solving difficult systems direct-to-steady-state is lacking. In comparison, results are shown for the one-level DD ILUT preconditioner with no fill-in and one level of overlap. In this case, it is clear that the most efficient and robust solution is obtained by the domain decomposition preconditioner. These methods are also highly parallel (see Section 6) and can provide reasonably robust and efficient solutions for CFD applications. However, the one-level DD schemes do exhibit a detrimental growth of the required iterations-to-solution as the mesh size decreases (or number of unknowns increases). This decrease in convergence rate with increasing problem size has been theoretically predicted for single unknown PDE systems [4]. As a further illustration, we present Table 5 that demonstrates a similar lack of robustness for the multi-step Jacobi preconditioner as compared to the DD preconditioner as the Re is increased in Example Problem 3.

5.2. Illustrations of the effectiveness of increasing ILU fill-in and overlap for DD preconditioners

In Tables 6 and 7, we present a comparison of the effectiveness of increasing fill-in and subdomain overlap for a domain decomposition preconditioner based on ILUT. These tables demonstrate that, in general, increasing fill-in and overlap can decrease iterations and often CPU time as well. Table 7 presents results for the fluid flow and transport calculations in the GaAs CVD reactor. In this case, we see a close connection between fill-in and robustness. The ability to adjust parameters such as the fill-in and amount of overlapping for Schwarz domain decomposition preconditioners allows solution algorithms to be adjusted to

Table 5
Comparison of multi-step block Jacobi and one-level DD preconditioners on lid driven cavity Example Problem 3

Preconditioner	*Re*		
	100	500	1000
5 Step block – Jacobi	9/3110/331.8	Failed	Failed
DD & ILUT(1.0,0.0)	6/404/41.3	10/809/75.3	13/1198/110.3

Results: Newton steps/linear its./CPU time (s).

Table 6
Effect of ILUT fill-in and level of overlap for thermal convection Example Problem 1, $Ra = 10,000$

Preconditioner	Level of overlap			
	1	2	3	4
ILUT(1.0,0.0)	4200/756	3960/715	3072/590	2680/545
ILUT(2.0,0.0)	1974/418	819/205	729/205	663/220
ILUT(3.0,0.0)	1853/438	792/238	700/249	620/271

Results: iterations/CPU time (s).

Table 7
Effect of fill-in and level of overlap for GaAs CVD transport Example Problem 5

Preconditioner	Level of overlap			
	1		2	
	Linear its.	CPU time (s)	Linear its.	CPU, s
ILUT(1.0,0.0)	Failed	Failed	Not run	Not run
ILUT(1.5,0.0)	1873	1336	477	447

produce more robust and often faster solutions. In general, while the number of iterations decreases with increasing fill-in and overlap, there is a point of diminishing returns and CPU time does not decrease monotonically. However, as can be seen in Table 7, the robustness of the increasing fill-in can sometimes mean the difference between a convergent method and a nonconvergent method. In the context of the one-level methods, this robustness comes at the expense of scalability. In the next section, the scalability of the one-level and two-level methods is described. It should be noted that these same one-level DD methods have been applied to reacting flow solutions for GaAs deposition in [26].

6. A comparison of one-level and two-level DD methods for 2D and 3D transport simulations

To illustrate the use of the two-level DD preconditioners as implemented in the ML library we present characteristic results for a standard 2D benchmark thermal convection flow problem (Example Problem 1) along with a generalization to a 3D problem (Example Problem 2). In this 2D benchmark problem [5], a thermal convection (or buoyancy-driven) flow in a differentially heated square box in the presence of gravity is modeled. The momentum transport, energy transport and total mass conservation equations defined in Table 1 are solved on a unit square. No-slip boundary conditions are applied on all walls. The temperature on the heated wall and other parameters are chosen so that the Rayleigh number Ra can be varied. The 3D problem adds two no-slip insulated walls in the third dimension to form a $1 \times 1 \times 1$ cube. A solution for this problem with $Ra = 1000$ is shown in Fig. 5. These simple geometries facilitate algorithmic/parallel studies as different mesh sizes can be easily generated. The results were obtained on the ASCI-Red Tflop computer at Sandia National Laboratories, which consists of nodes with 333 MHz Pentium II Xeon processors and 256 MB RAM.

In Figs. 1 and 2 and in Tables 8–10 results are presented for the algorithmic scaling of the one- and two-level DD schemes applied to the solution of the thermal convection example problems. Figs. 1 and 2 presents graphically the average iteration count per Newton step as a function of problem size (also processor count since the problem size per processor is fixed). In these computations the one-level method uses a DD preconditioner with an ILU preformed on each subdomain with one level of overlap between subdomains. The two-level methods employ a fine grid smoother which is based on a pointwise Gauss–Seidel iteration that is confined independently to each subdomain. The coarse grid solver for the two-level method is either based on a direct sparse solver (SuperLU) or an approximate coarse solver as described below. Clearly as the number of unknowns N is increased, the number of iterations to convergence for the one-level schemes increases significantly. This increase is roughly proportional to $N^{2/3}$ in 2D and $N^{1/2}$ in 3D. The two-level schemes are shown to be optimally convergent for the given fine-to-coarse grid ratio of 64 in 2D and 512 in 3D. The CPU time comparison indicates that while the two-level scheme can be faster, careful attention needs to be directed to the coarse grid solve times. In the 2D cases, the serial version of SuperLU is replicated on all processors to solve exactly the coarse grid problem; in the 3D case, a parallel version of SuperLU was invoked. Since using all P processors to factor this small system is not efficient,

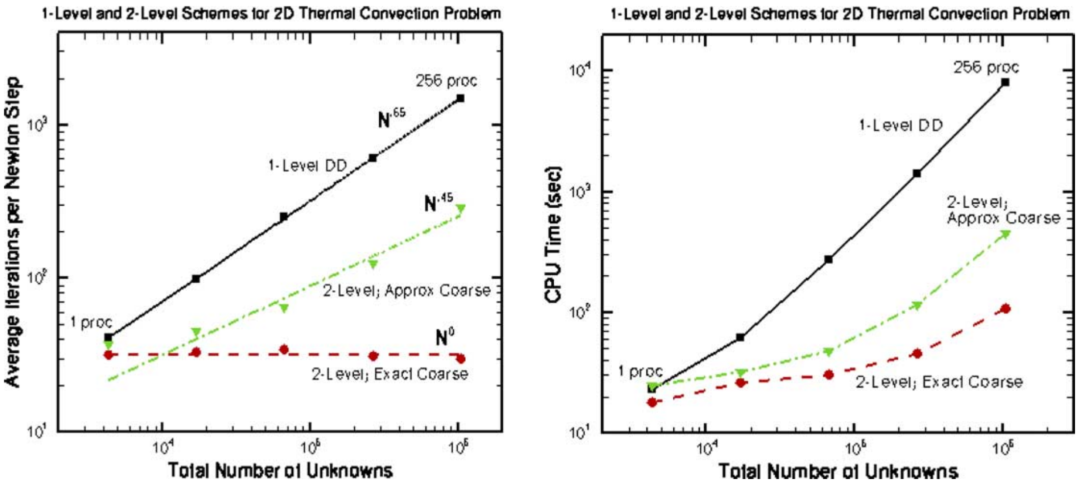


Fig. 1. Parallel and algorithmic scaling of iteration count and CPU time for one- and two-level DD preconditioners on 2D thermal convection problem Example Problem 1. $Ra = 1000$.

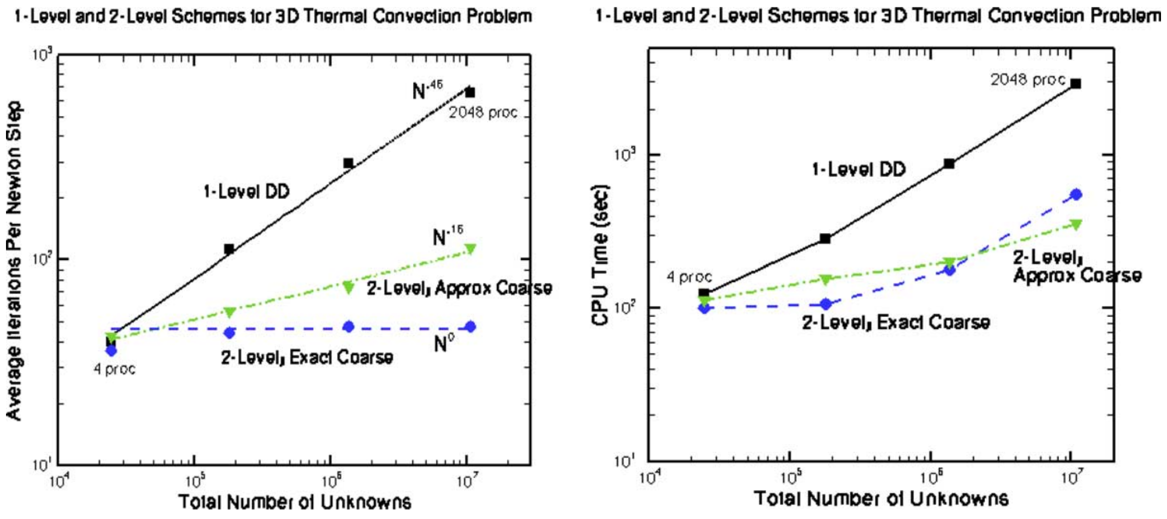


Fig. 2. Parallel and algorithmic scaling of iteration count and CPU time for one- and two-level DD preconditioners on 3D thermal convection Example Problem 2. $Ra = 1000$.

groups of approximately $P^{1/2}$ are utilized to solve a partially replicated linear system. Since the fine grid smoother is highly parallel [26] and the fine grid work per processor is fixed, it is the SuperLU performance on the increasingly larger coarse grid that causes an increase in the CPU time on the larger problems. Also in Figs. 1 and 2, CPU time results are presented for the 2D and 3D problems. In both cases, the CPU time is seen to increase as the coarse problem size grows. In the 3D case, the larger bandwidth or fill-in of the direct factorization is apparent even for moderately sized coarse grid problems. However, by using even coarser coarse grids or approximate solves instead of direct solves, it is possible to overcome the computational bottleneck associated with this direct solve. In this study, as an approximate solve, we have applied one-step of a DD ILU factorization with two levels of overlap between subdomains. These results, shown in

Table 8
Comparison of one-and two-level schemes for 2D thermal convection Example Problem 1

No. of processors	Fine grid size	Total unknowns	One-level method		Two-level method		
			Avg its. per Newton step	Total time (s)	Coarse grid size	Avg its. per Newton step	Total time (s)
1	32×32	4356	41	23	4×4	32	18
4	64×64	16,900	98	62	8×8	33	26
16	128×128	66,564	251	275	16×16	34	30
64	256×256	264,196	603	1399	32×32	31	46
256	512×512	1,052,676	1478	8085	64×64	30	107

Ra = 1000, *Pr* = 1. ASCI-Red, nonrestarted GMRES, one-level-ILU DD, two-level with 2 sweeps of Gauss–Seidel as Smoother, Superlu Coarse Grid Solver. The fine-to-coarse grid ratio is 64.

Table 9
Comparison of one- and two-level schemes for 3D thermal convection problem

No. of processors	Fine grid size	Total unknowns	One-level method		Two-level method		
			Avg its. per Newton step	Total time (s)	Coarse grid size	Avg its. per Newton step	Total time (s)
1	$8 \times 8 \times 8$	3645	18	38	$1 \times 1 \times 1$	19	24
8	$16 \times 16 \times 16$	24,565	47	62	$2 \times 2 \times 2$	38	48
64	$32 \times 32 \times 32$	179,685	114	150	$4 \times 4 \times 4$	45	66
512	$64 \times 64 \times 64$	1,373,125	308	521	$8 \times 8 \times 8$	48	110

Ra = 1000, *Pr* = 1., nonrestarted GMRES, one-level-DD ILU, two-level with 2 Gauss–Seidel sweeps as a Smoother, Superlu Coarse Grid Solver.
The fine-to-coarse grid ratio is 512.

Table 10
Comparison of one- and two-level schemes for 3D thermal convection problem

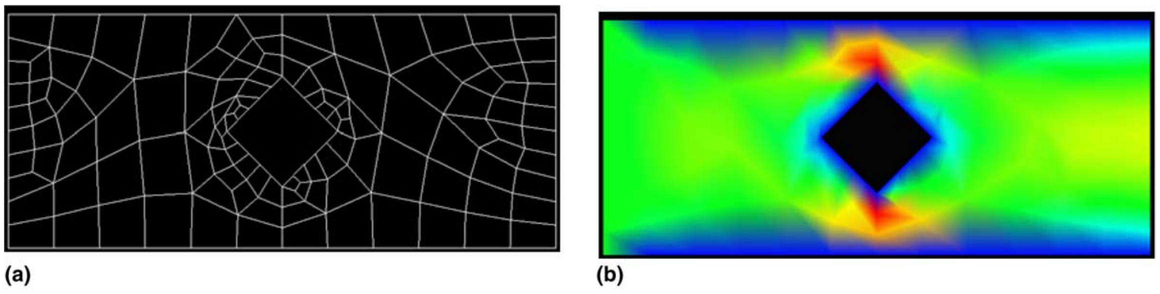
No. of processors	Fine grid size	Total unknowns	One-level method		Two-level method		
			Avg its. per Newton step	Total time (s)	Coarse grid size	Avg its. per Newton step	Total time (s)
4	$16 \times 16 \times 16$	24,565	40	123	$2 \times 2 \times 2$	36	101
32	$32 \times 32 \times 32$	179,685	112	282	$4 \times 4 \times 4$	44	107
256	$64 \times 64 \times 64$	1,373,125	296	863	$8 \times 8 \times 8$	47	179
2048	$128 \times 128 \times 128$	10,733,445	650	2915	$16 \times 16 \times 16$	47	546
4	$16 \times 16 \times 16$	24,565	40	123	$2 \times 2 \times 2$	42[a]	112[a]
32	$32 \times 32 \times 32$	179,685	112	282	$4 \times 4 \times 4$	56[a]	156[a]
256	$64 \times 64 \times 64$	1,373,125	296	863	$8 \times 8 \times 8$	73[a]	200[a]
2048	$128 \times 128 \times 128$	10,733,445	650	2915	$16 \times 16 \times 16$	114[a]	358[a]

Ra = 1000, *Pr* = 1., nonrestarted GMRES, one-level-DD ILU, two-level with two Gauss–Seidel sweeps as a Smoother, Superlu Coarse Grid Solver.
The fine-to-coarse grid ratio is 512.
[a] Coarse grid solve corresponds to a DD ILU factorization/backsolve in parallel with two levels of overlap.

the lower entries of Table 10, indicate that even this inexact coarse grid solve provides a suitable correction to the fine grid problem to accelerate convergence. However, for this case, the optimal convergence property is not obtained and a modest increase in the number of iterations is evident.

The next example considers flow in a channel with an obstruction (Example Problem 4). The results demonstrate the two-level Schwarz capability on an unstructured mesh problem for which the fine mesh is not a refinement of the coarse mesh. In this study the meshes were independently generated and therefore totally unrelated in structure (see Fig. 3). Since each mesh is partitioned separately by an automatic tool, Chaco [13], the resulting mesh partitions are not aligned in any way. In this example, the one-level solver uses a domain decomposition ILUT preconditioner with roughly twice as many nonzeros (fill-in = 2.0) as the original matrix and two levels of subdomain overlap [26]. The two-level solver uses the standard DD-ILU solver as a smoother with one level of overlap and employs a direct sparse solver (SuperLU) on the coarse grid solution which is replicated on each processor.

The first example of flow for Example Problem 4, a Stokes flow, considers the scaling of the methods on unstructured meshes. A typical coarse and fine mesh for this problem is presented in Fig. 3 along with solutions on each grid. Clearly while the coarse grid solution is under-resolved, there is significant information about the fine grid solution structure for this problem. As is evident from the convergence results presented in Tables 11 and 12, optimal convergence rates are obtained along with faster solution times for the



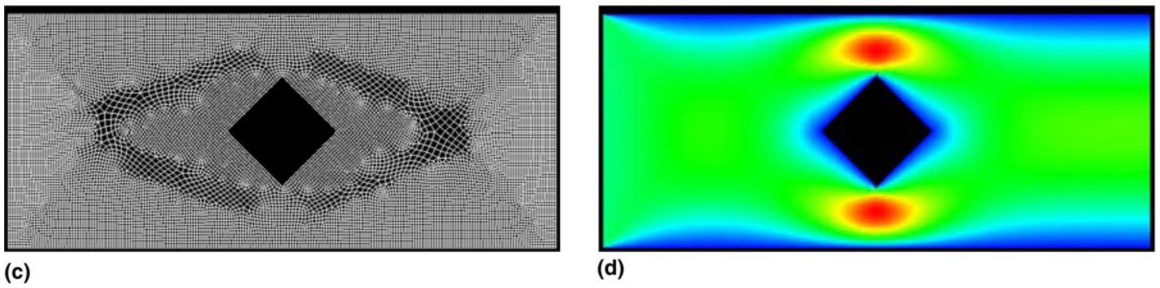


Fig. 3. Stokes flow about an obstruction, Example Problem 4 . Two-level DD method uses two unrelated meshes. (a) Coarse mesh, ~150 elements; (b) coarse mesh solution, x-velocity contour plot; (c) fine mesh, ~15,000 elements; and (d) fine mesh solution, x-velocity contour plot.

Table 11
Stokes flow Example Problem 4

Procs	Unknowns	One-level		Two-level	
		Iterations	Time (s)	Iterations	Time (s)
16	4704	32	5.9	25	6.0
64	13,008	73	9.9	33	6.6
256	55,008	316	48.3	34	14.1

GMRES solver: one-level-preconditioner = ILUT(2.,.2), two-level smoother = Gauss–Seidel, coarse solver Superlu, fine/coarse mesh ratio = 64.

Table 12
Stokes flow Example Problem 4

Procs	Unknowns	One-level		Two-level	
		Iterations	Time (s)	Iterations	Time (s)
16	18,240	52	34.1	41	33.1
64	50,976	136	60.9	59	34.5
256	217,920	704	390.2	77	70.5

GMRES solver: one-level-preconditioner = ILUT(2.,.2), two-level smoother = Gauss–Seidel, coarse solver Superlu, fine/coarse mesh ratio = 256.

two-level method for sufficiently large coarse grids. The CPU time scaling is again nonoptimal but still provides faster solutions than the corresponding one-level methods. As a final example of flow about the diamond obstruction, we consider a Navier–Stokes flow in the same flow geometry and examine the role of the fine grid smoother for the two-level methods. Table 13 presents the iteration count and CPU time to solution as a function of increasing Reynolds number *Re*. Clearly as *Re* is increased, the robustness and efficiency of the two-level methods require a more robust smoother for the fine grid subdomains. The development of appropriate smoothers for higher *Re* flows is an active area of research for multigrid as well as two-level DD methods [34,38].

7. A comparison of one- and two-level methods for large-scale flow, transport and reaction simulations

In this section, we demonstrate the performance of the one- and two-level domain decomposition preconditioners on two classes of large-scale applications. The first is a flow and transport problem that simulates the dispersion of a chemical agent in a large-scale indoor structure (Example Problem 6). A comparison of the performance of the one- and two-level methods is presented in Table 14. As demonstrated in this table the two-level preconditioners can provide substantial increases in performance even for modest sized problems. In 2D, execution time is reduced by a factor of roughly 90, and in 3D, by a factor of about 2.3 for problems with less than one million unknowns. Additionally we demonstrate solutions of very large problems with about 10 and 30 million unknowns in 2D and 3D, respectively, with the two-level methods. Specifically for the 3D 30 million unknown flow and transport problem, there is a very respectable solution time of less than two hours for a direct-to-steady-state calculation on 256 processors of the Sandia Cplant machine. Cplant consists of nodes with Dec Alpha 500 MHz processors and 1 GB RAM connected by Myrinet. For both the 2D and 3D cases, the convergence criterion for the linear solve was chosen to be 4×10^{-4}.

Table 13
Comparison of one- and two-level schemes for 2D Navier–Stokes flow past diamond obstruction (Example Problem 4)

Smoother	Two-level with Jacobi (2 sweeps)			Two-level with Gauss–Seidel (2 sweeps)			Two-level with ILUT			One-level DD ILUT		
Re	Newton steps	Linear its.	Time (s)	Newton steps	Linear its.	Time (s)	Newton steps	Linear its.	Time (s)	Newton steps	Linear its.	Time (s)
1.0	4	996	3376.8	4	534	2065.7	4	301	645.9	4	831	2045.3
10.0	5	1442	5172.9	5	760	3001.6	5	415	917.6	5	1065	2633.0
100.0	–	–	–	12	5721	27439.7	8	687	1564.8	8	1460	3310.0

The fine mesh has 96,768 unknowns and the coarse mesh has 468 unknowns.

Table 14
Comparison of one- and two-level preconditioners for steady-state solve of fluid flow and transport of a chemical agent in a 2D and 3D large-scale indoor structure for laminar flow conditions

	Preconditioner	Smoothers/coarse solver	Fine mesh unknowns	Coarse mesh unknowns	Avg. its. per Newton step	Total time (s)	Computing hardware
2D	1-level DD	ILU	619,300	10,720	500	10,460	20 1-GHz P3
	2-level DD	ILU/SuperLU	619,300	10,720	17	118	20 1-GHz P3
	2-level DD	ILU/SuperLU	9.8M	10,720	293	3266	128 procs CPlant
3D	1-level DD	ILU	872,000	17,360	118	1801	16 1-GHz P3
	2-level DD	ILU/SuperLU	872,000	17,360	21	795	16 1-GHz P3
	2-level DD	ILU/GMRES	28.9M	68,320	40	5536	256 procs CPlant

Example Problem 6.

Our last example problem is a 3D reacting flow simulation for the deposition of poly-Silicon in a horizontal rotating disk reactor (Example Problem 7). This problem is a full reacting flow problem with unknowns for the three velocity components, hydrodynamic pressure, temperature, and 3 chemical species (H_2, $SiCl_3H$, HCl). We present results for a simple continuation run, incrementally increasing the reactor thermodynamic pressure from 0.6 atm to the operating pressure of 0.85 atm. This type of continuation procedure is a common technique to follow physically realistic paths through a complex nonlinear solution space as often encountered in reacting flow simulations. In this context, it should be noted that, in some sense, the one-level scheme benefits (or conversely the two-level method loses some of its advantage) because the coarse scales or global modes of the solution are approximately set by using the 0.6 atm solution as a starting guess for the nonlinear scheme, and consequently in the linear subproblems that are generated by Newton's method. Preliminary results for this calculation are shown in Table 15, which presents the convergence and CPU times for a set of three differently sized meshes for this problem. The problems sizes range from 0.6 to 38 M unknowns and all of the two-level solvers use the same large (87,400 unknowns) coarse mesh. The convergence criterion for the linear solve was chosen to be 3×10^{-4}. The results indicate reasonable performance for the largest of problems with the two-level method being on the order of 25% faster that the one-level method. The solution to the very large 38M unknown problem requires about 2.25 h. on 1000 processors of CPlant. While the ratio of CPU time for the one-level to two-level method is not as

Table 15
Comparison of one- and two-level preconditioners for steady-state solve of reacting flow simulation of epitaxial silicon deposition in 3D CVD horizontal spinning disk reactor

Levels of Ref.	Preconditioner	Smoother/coarse solver	Fine mesh unknowns	Avg. its. per Newton step	Total time (s)	Computing hardware
1	1-level DD	ILU	636,168	99	262	64 3-GHz P4
	2-level DD	ILU/ILU	636,168	50	249	64 3-GHz P4
2	1-level DD	ILU	4,845,640	184	2971	64 3-GHz P4
	2-level DD	ILU/ILU	4,845,640	84	2204	64 3-GHz P4
1	1-level DD	ILU	636,168	111	384	128 Procs CPlant
	2-level DD	ILU/ILU	636,168	53	406	128 Procs CPlant
2	1-level DD	ILU	4,845,640	243	1186	1000 Procs CPlant
	2-level DD	ILU/ILU	4,845,640	124	1152	1000 Procs CPlant
3	1-level DD	ILU	37,806,984	482	10,973	1000 Procs CPlant
	2-level DD	ILU/ILU	37,806,984	230	8174	1000 Procs CPlant

The coarse problem has 87,400 unknowns. Example Problem 7.

impressive as the fluid flow or fluid flow with transport calculations presented earlier, there is still a benefit to using the two-level method. In addition the same scaling or trend is visible in the results as the problem size increases with the two-level method becoming faster. When interpreting these results it must be kept in mind that this is a particularly difficult problem to solve and a direct-to-steady-state fully coupled solver is being used. In this context, the convergence of the two-level preconditioner is encouraging. It should also be noted that the two-level method is not always faster than the one-level method on this problem for the small levels of refinement between the fine and coarse mesh. For one level of refinement, the fine mesh has eight times as many elements as the coarse mesh, and the cost of the coarse mesh solve is significant compared to the fine mesh solve, perhaps offsetting any gains from the reduction in iteration count.

When interpreting the results of one- and two-level methods there are at least two particular issues about the current multilevel solution methodology that should be kept in mind to explain the less than impressive results on the 3D CVD reactor problem. First, due to the limitation of accurately meshing the complex 3D reactor geometry, the coarse problem is quite large. This required using an approximate ILU coarse grid solver instead of the direct sparse solver for the coarse grid problem and most likely contributes to the only moderate reduction from 480 to 230 average linear solver iterations per Newton step. In general this limitation of producing a coarse grid that reflects the underlying geometric complexity is important. To eliminate this requirement we are pursuing a new aggressive coarsening algebraic multigrid technique based on graph partitioning for coupled systems of equations [18]. Secondly, we believe that the possible indefiniteness of the chemical reaction source terms for this system might also be contributing to the only moderate reduction of the iteration count. Currently we are considering, in more detail, this issue and exploring multi-level formulations which attempt to handle these type of systems [37]. Clearly, further analysis and experimentation with these two-level methods in the reacting flow context is required and is currently being pursued. However, we consider these modest steps to be encouraging.

8. Conclusions

This set of numerical studies has presented a number of important issues related to efficient and robust solution of unstructured FE flow solutions with heat and mass transport by parallel fully coupled Newton–Krylov solution methods. As presented, careful attention to problem formulation and the use of an inexact Newton method with domain decomposition preconditioned Krylov based solvers can produce efficient and robust solution algorithms on large-scale parallel supercomputers. These methods allow the selection of algorithmic parameters such as the Krylov subspace dimension, the fill-in for incomplete factorizations, and the level of subdomain overlapping for Schwarz domain decomposition preconditioners to tailor the robustness and computational efficiency of the resulting CFD solution method. As presented, the results verify that these methods can produce robust and efficient solution methods. As the problem size increases, the scaling of the average iteration count per Newton step for the one-level fully coupled domain decomposition preconditioners is roughly proportional to $N^{2/3}$ in 2D and $N^{1/2}$ in 3D. This adverse scaling for large problems has been demonstrated to be reduced by the use of two-level methods. Using these techniques we have demonstrated optimal algorithmic scaling for a range of fluid flow and transport problems. The scaling of CPU time has not yet been demonstrated to be optimal. However, the results have been very encouraging for fully coupled solution methods and have allowed solution of large problems of $O(10^7)$ unknowns for direct-to-steady-state computations in very respectable times. The use of approximate coarse grid solvers for the two-level methods has been demonstrated to be effective to reduce the total CPU time for the large coarse-grid problems. These results are very encouraging for the simulation of fluid flow with transport, an important subproblem of reacting flow calculations. Currently, the one-level method is the standard preconditioner that we use for reacting flow simulations [26]. The reacting flow results presented in this study are the first applications of two-level methods to our reacting flow problems; they are however, preliminary. For these reacting flows we have demonstrated convergence of the one- and two-level methods for a large-scale reacting flow epitaxial-Silicon CVD simulation as a final example. This problem with approximately 40M unknowns was solved direct-to-steady-state in less than 2.5 h on a large-scale parallel machine, a very encouraging result. While the two-level method was not significantly faster than the one-level method as in some of the nonreacting flow cases, it did provide a performance increase over the one-level method in some of the computations. In future work, we will consider further studies of these methods for reacting flows and study the effect of strong convection (anisotropic behavior) and reaction (indefinite source terms) on convergence and scaling of these methods.

Acknowledgement

The authors thank Andrew Salinger and Roger Pawlowski for continued development of the MPSalsa code which was used to generate the results in this study. Also we thank Andrew Salinger for use of the 3D Epitaxial-Silicon deposition example mesh and problem definition. We also thank Colin O'Rourke for help with making a number of computational runs that are presented here.

Appendix A. Example problem description

In the following section, we describe the test problems listed in Table 3 that make up the numerical studies. These examples consist of two 2D test problems (thermal convection, lid driven cavity) which are standard test problems, a 3D generalization of the 2D thermal convection problem, a 2D flow about a diamond shaped body, two CVD reactor simulations and a fluid flow and transport calculation in a large-scale internal structure. In Table 3, a summary is provided of the specific governing PDE equations that are solved for each example problem.

Example Problem 1. (*2D thermal convection*). In this standard 2D benchmark problem [5], a thermal convection (or buoyancy-driven) flow in a differentially heated square box in the presence of gravity is modeled. The momentum transport, energy transport and total mass conservation equations defined in Table 1 are solved on a unit square. No-slip boundary conditions are applied on all walls. The temperature on the heated wall and other parameters are chosen so that the Rayleigh number Ra can be varied. In Fig. 4, we show a typical solution of this problem with $Ra = 1{,}000{,}000$. A second parameter, the Prandtl number Pr, is taken to be $Pr = 1$, implying that the diffusion of momentum and thermal energy are roughly equivalent.

Example Problem 2. (*3D thermal convection*). The 3D Thermal convection example problem adds two no-slip insulated walls in the third dimension to form a $1 \times 1 \times 1$ cube from Example Problem 1. A typical solution is shown in Fig. 5.

Example Problem 3. (*Lid driven cavity*). In this standard 2D benchmark problem [8,23], the momentum transport and total mass conservation equations defined in Table 1 are solved on a unit square to simulate confined flow driven by a moving boundary on the upper wall. No-slip boundary conditions are applied on all other walls. As the Reynolds number Re is increased, the nonlinear inertial terms in the momentum equation become more dominant and the solution becomes more difficult to obtain. A typical solution for this problem is shown in Fig. 6.

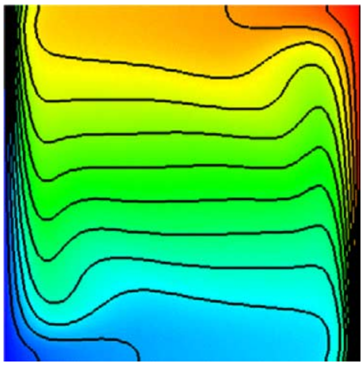


Fig. 4. Contour plot of the temperature for Example Problem 1 at $Ra = 1{,}000{,}000$.

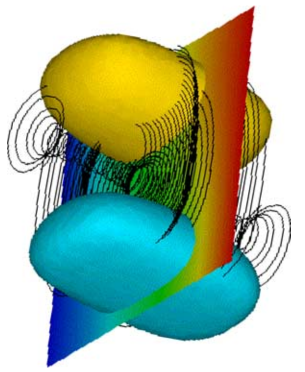


Fig. 5. Constant x-velocity iso surfaces with streamlines and temperature contours on slice plane. $Ra = 1000$. $Pr = 1$, Example Problem 2.

Example Problem 4. (*2D internal channel flow about an obstruction*). In this internal channel flow example the flow is accelerated about a diamond shaped obstruction at the center. The momentum transport and total mass conservation equations defined in Table 1 are solved on a complex unstructured mesh. The Stokes flow solution exhibits the isotropic flow field, whereas the Navier–Stokes solution at $Re = 100$ clearly shows the nonisotropic effect of the convection operator in the governing transport/reaction equations (see Fig. 7).

Example Problem 5. (*Flow and transport in a 3D tilted chemical vapor deposition (CVD) reactor*). This example problem involves computing the 3D solution for fluid flow, heat transfer and the mass transfer of three chemical species in a horizontal tilted chemical vapor deposition (CVD) reactor. Fluid enters in the larger cross sectional area inlet and accelerates up the inclined surface with the inset rotating heated disk. At the elevated disk temperature, chemical reactions are initiated to deposit gallium arsenide (GaAs). In this example, we only transport the precursors for this reaction (tri-methylgallium, $Ga(CH_3)_3$, arsine, AsH_3) and a carrier gas (hydrogen, H_2) and do not allow chemical reactions. In our example calculation, the inlet velocity is 100 cm/s, the inlet temperature is 600 K, and the disk rotates at 200 rpm and is heated to 900 K. To simulate the deposition process, we use a Dirichlet condition on the reactants that introduces significant diffusion gradients and boundary layers that approximate the average behavior of the full reacting system depositing GaAs on the rotating disk. In practice, CVD reactors are run at low pressures and

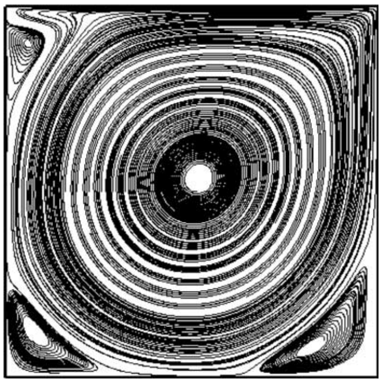


Fig. 6. Lid driven cavity. Contour plot of the stream function for Example Problem 3 at $Re = 4000$.

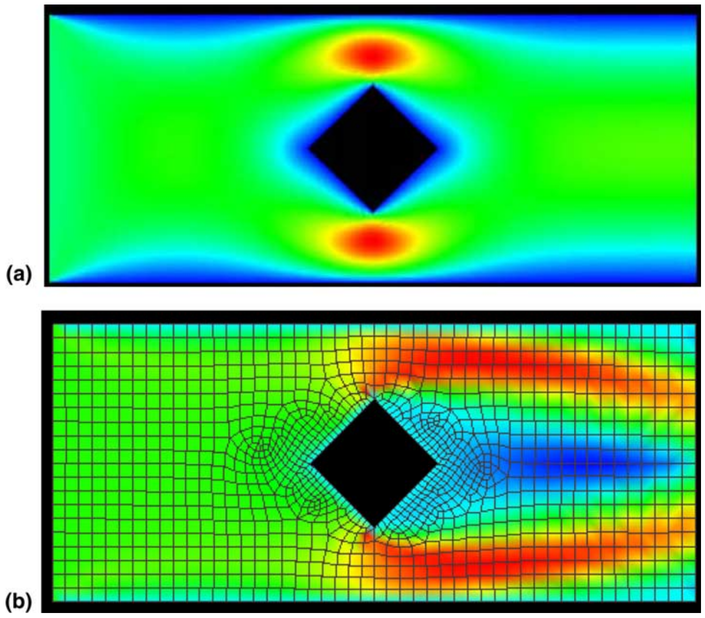


Fig. 7. Example Problem 4. Flow about a diamond shape obstruction. (a) Stokes flow solution. (b) Navier–Stokes flow solution and FE mesh at $Re = 100.0$. x-velocity field contour plot shown.

fluid velocities, and thus the Reynolds numbers are small ($Re < 1$). In addition, for gases at these temperatures and pressures, the Prandtl number and the Schmidt number for mass transport (analogous to the Prandtl number) are approximately one as well. A typical reacting flow solution is shown in Fig. 8, where the streamlines show the effect of the counterclockwise rotation of the disk. Included is a contour plot of the concentration of tri-methylgallium at the heated surface [26]. This contour plot is from the full reacting flow solution. For these experiments, the number of unknowns for the discretized problem was 384,200. The number of Intel TFlop processors used was 48. The GMRES restart value was 150, with a maximum of 600 GMRES iterations allowed at each inexact Newton step.

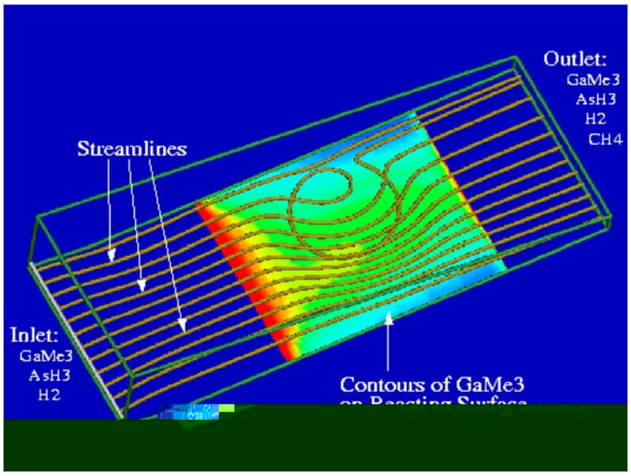


Fig. 8. Solution for the tilted CVD reactor used in Example Problem 5. Streamlines and filled contours of tri-methyl gallium are shown.

Example Problem 6. (*Fluid flow and transport of a chemical agent in a building*). This example problem simulates the flow and transport of a chemical agent in a large-scale indoor structure [18]. There are 2D and 3D versions of this problem. The 2D structure is 11 m high and 105 m wide. The 3D structure is 11 m $\times$ 105 m $\times$ 24 m. In these example problems, *Re* is restricted to a lower than normal operating condition of one order of magnitude in the 2D case and two orders of magnitude in the 3D case based on an inlet duct reference length and flow. The low *Re* allows us to quickly solve direct-to-steady-state with the DD solvers and simplifies the presentation of the results. In addition, these lower Reynolds number solutions can sometimes be used as initial guesses to attempt to solve mean-steady RANS type turbulence models in the same geometries at higher *Re*. While we do not present the results here, we have also used the one- and two-level methods to solve for transient turbulent large eddy simulations (LES) at the standard operating conditions of the ventilation systems. In Fig. 9 the upper inset figure shows the detailed steady-state streamlines of the flow in the 2D structure along with color contours of a neutrally buoyant chemical agent (in this case SF_6) that is released at two locations on the lower floor of the structure. There are inlets on the sidewalls of both floors, the ceiling of the lower level, and the floor of the upper level. Outlets are arranged on the ceiling of the upper floor, and the main outlet collector is above an atrium (or opening) between the two floors. The 3D structure is shown in the lower inset figure and exhibits three iso-surfaces of SF_6 from two source locations on the lower floor. There is a similar arrangement of inlets and outlets in the 3D model as well, with the addition of extra inlets on the 105 m length face.

Example Problem 7. (*Flow, transport and reaction in a* 3*D horizontal epitaxial-silicon CVD reactor*). This is an example of reacting flow through a CVD reactor and involves the solution for fluid flow, heat and mass transfer of three chemical species. A mixture of trichlorosilane ($SiCl_3H$), HCl, and H_2 enters from the left, flows over a forward facing step, and over an inset rotating disk. The disk is heated to 1398 K, which initiates chemical reactions to deposit silicon on the wafer. The reaction model of Kommu et al. [17] was used. The top left figure in Fig. 10 shows the different side sets for the solid model. The top right figure shows an example of the mesh partitioned for 48 processors. The bottom figure shows the contour plot of HCl mass fraction on the rotating disk surface and the flow streamlines. Three different sized meshes were used for this calculation to give 0.6, 4.8, and 37.8 M unknowns.

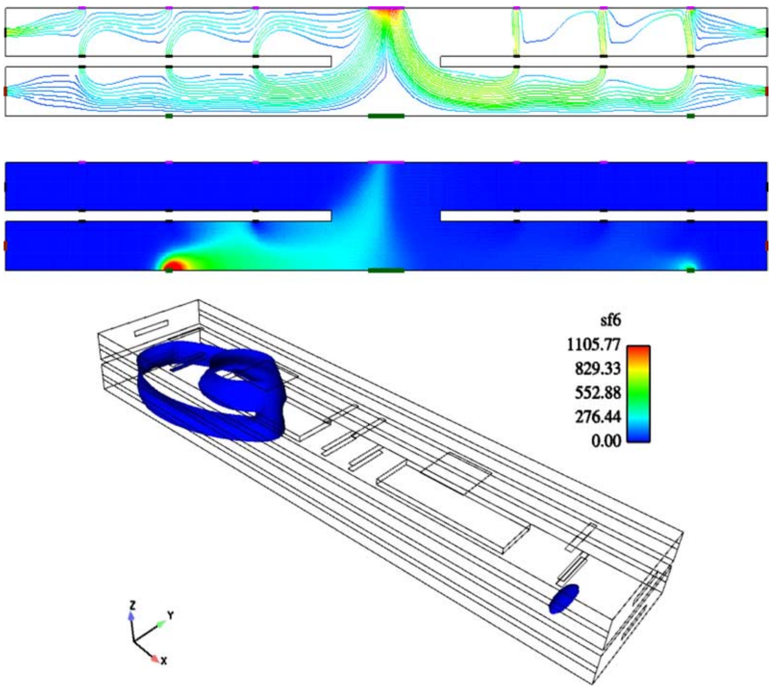


Fig. 9. 2D and 3D fluid flow and chemical agent transport in a building Example Problem 6.

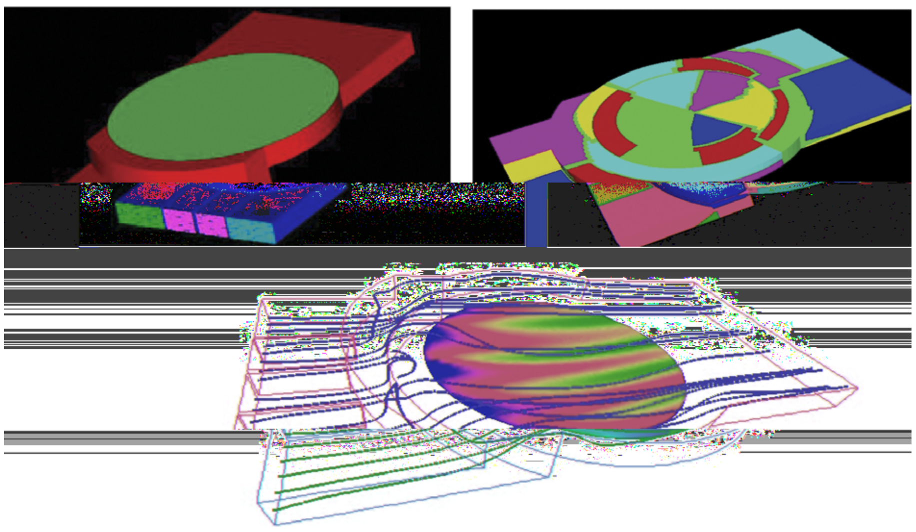


Fig. 10. Flow, transport and reaction in a 3D horizontal epitaxial-silicon CVD reactor, Example Problem 7. Upper images: solid model and partitioned subdomains for parallel execution. Lower image: flow streamlines and a color contour plot of HCl mass fraction on the rotating disk surface.

References

[1] M. Benzi, Preconditioning techniques for large linear systems: a survey, JCP 182 (2002) 418–477.

[2] P.N. Brown, Y. Saad, Hybrid Krylov methods for nonlinear systems of equations, SIAM J. Sci Stat. Comp. 11 (3) (1990) 450–481.

[3] X.-C, Cai, An additive Schwarz algorithm for nonselfadjoint elliptic equations, in: Tony F. Chan, Roland Glowinski, Jacques Periaux, Olof B. Widlund (Eds.), Third International Symposium on Domain Decomposition Methods for Partial Differential Equations, 1989, pp. 232–244.

[4] X.-C. Cai, W.D. Gropp, D.E. Keyes, Convergence rate estimate for a domain decomposition method, Num. Math. 61 (2) (1992) 153–169.

[5] G. de Vahl Davis, I.P. Jones, Natural convection in a square cavity: a comparison exercise, Int. J. Num. Meth. Fluids 3 (1983) 227–248.

[6] S.C. Eisenstat, H.F. Walker, Globally convergent inexact Newton methods, SIAM J. Optim. 4 (2) (1994) 393–422.

[7] C. Farhat, N. Maman, G.W. Brown, Mesh partitioning for implicit computations via iterative domain decomposition – impact and optimization of the subdomain aspect ratio, Int. J. Num. Meth. Eng. 38 (6) (1995) 989–1000.

[8] U. Ghia, K.N. Ghia, C.T. Shin, High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method, J. Comput. Phys. 48 (1982) 387–411.

[9] W.D. Gropp, D.K. Kaushik, D.E. Keyes, B.F. Smith, High-performance parallel implicit CFD, Parallel Comput. 27 (2001) 337–362.

[10] W.D. Gropp, B. Smith, Scalable, extensible, and portable numerical libraries, in: Proceedings of the Scalable Parallel Libraries Conference 87–93, IEEE, Los Alamitos, CA, 1994.

[11] B. Hendrickson, Load balancing fictions, falsehoods and fallacies, Appl. Math. Model. 25 (2) (2000) 99–108.

[12] B. Hendrickson, T.G. Kolda, Graph partitioning models for parallel computing, Parallel Comput. 26 (12) (2000) 1519–1534.

[13] B. Hendrickson, R. Leland, The Chaco User's Guide, Version 2.0., Sandia National Laboratories Technical Report, SAND95-2344, Albuquerque, NM, 1995.

[14] T.J.R. Hughes, L.P. Franca, M. Balestra, A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuska–Brezzi condition: a stable Petrov–Galerkin formulation of the Stokes problem accommodating equal-order interpolations, Comp. Meth. Appl. Mech. Eng. 59 (1986) 85–99.

[15] R.S. Tuminaro, M. Heroux, S.A. Hutchinson, J.N. Shadid, Official Aztec User's Guide Version 2.1, Sandia National Laboratories Technical Report, SAND99-8801J 1999.

[16] D.A. Knoll, P.R. McHugh, Newton–Krylov methods applied to a system of convection–diffusion-reaction equations, Comput. Phys. Commun. 88 (1995) 141–160.

[17] S. Kommu, D. Sinha, J. Knieling, A theoretical/experimental study of silicon epitaxy in horizontal single-wafer chemical vapor deposition reactors, J. Elect. Chem. Soc. 147 (4) (2000) 1538–1550.

[18] P.T. Lin, M. Sala, J.N. Shadid, R.S. Tuminaro, Performance of a geometric and an algebraic multilevel preconditioner for incompressible flow with transport, in: Z. Yao, M. Yuan, W. Zhong (Eds.), Computational Mechanics, Proceedings of WCCM VI, Tsinghua Univ. Press, Springer, 2004.

[19] Y. Saad, Iterative Solution Methods for Sparse Linear Systems, second ed., SIAM, 2003.

[20] Y. Saad, Krylov subspace methods on supercomputers, SIAM J. Sci. Stat. Comput. 10 (6) (1989) 1200–1232.

[21] Y. Saad, Gen-Ching Lo, S. Kuznetsov, PSPARSLIB Users Manual: A Portable Library of parallel Sparse Iterative Solvers, 18 January 1998.

[22] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856–869.

[23] R. Schreiber, H.B. Keller, Driven cavity flows by efficient numerical techniques, J. Comput. Phys. 49 (1983) 310–333.

[24] J.N. Shadid, A comparison of parallel preconditioners for solution of unstructured finite element fluid flow, heat and mass transfer simulations, Proceedings of the Fourth Japan–US Symposium on Finite Element Methods in Large-Scale Computational Fluid Dynamics, Nihon University, Tokyo, Japan, 1998, April 2–4.

[25] J.N. Shadid, A fully-coupled Newton–Krylov solution method for parallel unstructured finite element fluid flow, heat and mass transfer simulations, Int J. CFD 12 (1999) 199–211.

[26] J. Shadid, S. Hutchinson, G. Hennigan, H. Moffat, K. Devine, A.G. Salinger, Efficient parallel computation of unstructured finite element reacting flow solutions, Parallel Comput. 23 (1997) 1307–1325.

[27] J.N. Shadid, A.G. Salinger, R.C. Schmidt, T.M. Smith, S.A. Hutchinson, G.L. Hennigan, K.D. Devine, H.K. Moffat, MPSalsa Version 1.5: A Finite Element Computer Program for Reacting Flow Problems: Part 1 – Theoretical Development, Sandia National Laboratories Technical Report, SAND98–2864, Albuquerque, NM, 1999.

[28] J.N. Shadid, R.S. Tuminaro, H.F. Walker, An inexact Newton method for fully-coupled solution of the Navier–Stokes equations with heat and mass transport, J. Comput. Phys. 137 (1997) 155–185.

[29] J.N. Shadid, R.S. Tuminaro, A comparison of preconditioned nonsymmetric Krylov methods on a large-scale MIMD machine, SIAM J. Sci. Comput. 15 (2) (1994) 440–459.

[30] F. Shakib, T.J.R. Hughes, Z.A. Johan, New finite-element formulation for computational fluid-dynamics. 10. The compressible Euler and Navier–Stokes equations, Comp. Meth. Appl. Mech. Eng. 89 (1–3) (1991) 141–219.

[31] D. Silvester, A. Wathen, Fast iterative solution of stabilized stokes systems – Part II: using general block preconditioners, SIAM J. Num. Anal. 31 (5) (1994) 1352–1367.

[32] B.F. Smith, P.E. Bjorstad, W.D. Gropp, Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations, Cambridge University Press, Cambridge, 1996.

[33] T.E. Tezduyar, Stabilized finite element formulations for incompressible flow computations, Adv. Appl. Mech. 28 (1992) 1–44.

[34] U. Trottenberg, C. Oosterlee, A. Schuller, Multigrid, Academic Press, London, UK, 2001.

[35] R.S. Tuminaro, C.H. Tong, J.N. Shadid, K.D. Devine, D.M. Day, On a multilevel preconditioning module for unstructured mesh Krylov solvers: two-level Schwarz, Commun. Num. Meth. Eng. 18 (2002) 383–389.

[36] C. Vincent, R. Boyer, A preconditioned conjugate gradient uzawa-type method for the solution of the Stokes problem by mixed Q1-P0 stabilized finite elements, Int. J. Num. Meth. Fluids 14 (1992) 289–298.

[37] H. Waisman, J. Fish, R.S. Tuminaro, J.N. Shadid, The generalized global basis (GGB) method, Int. J. Num. Meth. Eng. 61 (8) (2004) 1243–1269.

[38] P. Wesseling, Principles of computational fluid dynamics, Computational Mechanics, vol. 29, Springer, Berlin, 2001.

[39] S.O. Wille, A preconditioned alternating inner-outer iterative solution method for the mixed finite element formulation of the Navier–Stokes equations, Int. J. Num. Meth. Fluids 18 (1994) 1135–1151.