Parallel Adaptive Finite Element Analysis of Viscous Flows Based on a Combined Compressible-Incompressible Formulation

Dan Givoli* **Joseph E. Flaherty**

Department of Computer Science

Rensselaer Polytechnic Institute, Troy, NY 12180

Mark S. Shephard

Scientific Computation Research Center

Rensselaer Polytechnic Institute, Troy, NY 12180

November 6, 1996

*On leave from the Department of Aerospace Engineering, Technion — Israel Institute of Technology, Haifa 32000, Israel

Abstract

A new finite element scheme is described for the large-scale analysis of compressible and incompressible viscous flows. The scheme is based on a combined compressible-incompressible Galerkin Least-Squares (GLS) space-time variational formulation. Three-dimensional unstructured meshes are employed, with piecewise-constant temporal interpolation, local time-stepping for steady flows, and linear continuous spatial interpolation in all the variables. The scheme incorporates automatic adaptive mesh refinement, with a choice of various error indicators. It is implemented on a distributed-memory parallel computer, and includes an automatic load-balancing procedure. The ability to solve both compressible and incompressible viscous flow problems using the parallel adaptive framework is demonstrated via numerical examples. These include Mach 3 flow over a flat plate, and a divergence-free buoyancy-driven flow in a cavity. The latter is a model for the steady melt flow in a Czochralski crystal growth process.

Keywords: Parallel, Adaptive, Finite element, Galerkin Least Squares, Crystal growth, Czochralski.

1. Introduction

During the last few years there has been a tremendous increase in the development and application of finite element technology for the solution of fluid flow problems, particularly on adaptively defined unstructured meshes. Some of the current finite element codes can handle large-scale three-dimensional simulations of compressible or incompressible flows with very complicated geometry, general boundary conditions, and hundred of thousands to millions of degrees of freedom [1, 2].

A successful general analysis code that attempts to handle realistic problems, must be equipped with four main properties: modeling flexibility, numerical stability, accuracy and high speed. Flexibility in modeling general geometries, material properties, boundary conditions, etc., is an important capability which requires special non-trivial treatment, especially in three dimensions. Stability and accuracy are essential ingredients in establishing the reliability of the numerical results. High computing speed is a critical necessity for practical reasons of obtaining results within acceptable times. Various tools have been devised in recent years in the context of finite element analysis to address each of these issues. The scheme described in this paper incorporates a number of tools which cover all four aspects, and thus leads to a powerful solution technique.

Traditionally compressible and incompressible viscous flow problems are handled by two separate codes, since these two types of problems typically involve different concerns and numerical difficulties. However, recently Hauke and Hughes [3] showed how the compressible and incompressible cases can be combined, by approaching the incompressible limit while using either primitive (pressure-velocity-temperature) variables or entropy variables (but not conservation variables). In this work, we have adopted this approach and have developed a finite element scheme capable of solving both the compressible and incompressible Navier-Stokes equations. While a direct approach to incompressible problems may be more efficient in some cases [1], the combined compressible-incompressible approach provides a unified and convenient analysis framework, and is especially attractive when buoyancy is introduced into the incompressible Naver-Stokes equations via the Boussinesq approximation.

In the present paper, we propose a new finite element framework for the large-scale analysis of compressible and incompressible viscous flows. The scheme is based on a combined compressible-incompressible Galerkin Least-Squares (GLS) space-time variational formulation. Unstructured three-dimensional spatial meshes of tetrahedral elements are employed, with linear spatial interpolation in all the variables. This setup enables straight-forward generalization to higher-order elements. Piecewise-constant interpolation is used in time, with local time-stepping for steady flows. The scheme incorporates automatic adaptive mesh refinement, with a choice of various error indicators. It is implemented on a distributed-memory MIMD parallel computer, and includes an automatic load-balancing procedure to uniformly distribute the load among the processors. The nonsymmetric system of algebraic equations is solved iteratively using preconditioned GMRES.

This computational framework includes some of the features that have been used in [4], [2] and [3], as shown in Table 1. It combines the capability to analyze viscous flows [4] using a combined compressible-incompressible formulation [3], with parallel adaptive capabilities in three dimensions using unstructured meshes [2]. In addition, the scheme is capable of solving the Navier-Stokes equations with the Boussinesq approximation. This allows for the analysis of buoyancy-driven flows associated with divergence-free velocity fields, which are typical, for example, in certain crystal growth applications. Also, a "boundary layer error indicator" is incorporated in the adaptive procedure, which is based on the local gradient of the velocity magnitude.

We apply the new scheme to two model viscous flow problems. The first problem concerns the Mach 3 flow of air over a flat plate, and is intended to test the performance of the scheme in analyzing compressible supersonic flows. The results are compared to those presented in [4], which are obtained using a serial code with a uniform structured two-dimensional mesh.

The second problem is that of a divergence-free buoyancy-driven flow in a cavity. This is a model for the steady melt flow in a Liquid Encapsulated Czochralski (LEC) crystal growth process [5, 6]. There have been many attempts to model Czochralski (CZ) and LEC processes with various levels of complexity. See, e.g., the local (single-phase) analyses in [7]–

[11] and the global (multi-phase) analyses in [12]–[15], which use various finite volume and finite element schemes, as well as the recent review [16], and references therein. Most of the computational works on CZ and LEC assume axi-symmetric solutions, but some works are based on three-dimensional analyses [8], [9]–[11].

Most of the finite element analyses performed in crystal growth simulations are based on a standard Galerkin weighted residual formulation. For example, the scheme in [12]–[14] uses continuous biquadratic interpolation for velocities and temperature, and discontinuous bilinear interpolation for pressure. This combination is successful in that it satisfies the Babuška-Brezzi stability condition [17], although it is not easily generalized to higher-order interpolations. There has been an attempt to generalize this formulation to a Petrov-Galerkin framework in [18]. Here we perform a local analysis of steady-state flow in indium phosphide (InP) melt under LEC conditions, using a rather simple model, to check the effectiveness of applying the new computational framework, including the GLS formulation and the parallel adaptive capabilities, to problems of this type. The governing equations are the Navier-Stokes equations with the Boussinesq approximation.

In the next section, we describe the flow problems under consideration. In Section 3 we discuss the finite element formulation, and in Section 4 we describe the computational methodology and its various ingredients in some detail. In Section 5, we present some numerical results; first for the Mach 3 flow of air over a flat plate, and then for buoyancy-driven flows in InP melt. We close the paper with some remarks.

2. Statement of the Problem

The compressible Navier-Stokes equations can be written in the general conservation form (see, e.g., [19]),

$$\dot{U} + F^{\mathrm{adv}}_{i,i} = F^{\mathrm{diff}}_{i,i} + S(U) \ . \tag{1}$$

Here, U is the vector of conservation variables, F^{adv}_i and F^{diff}_i are the advective and diffusive flux vectors in the ith direction, and S is the source vector. A superposed dot indicates

differentiation with respect to time, a comma denotes partial differentiation, and the summation convention with respect to repeated indices is enforced. In the three dimensional case, (1) is a set of five equations for mass, momentum (three scalar equations), and energy conservation.

Consider a transformation $Y = Y(U)$, and convert (1) to the quasi-linear form

$$A_0 \dot{Y} + A_i Y_{,i} = (K_{ij} Y_{,j})_{,i} + S(Y) \ , \tag{2}$$

where A_0, A_i and K_{ij}, $i, j = 1, 2, 3$, are Y-dependent matrices. Assume that the source term S can be written in the form,

$$S = S_0 - CY \ , \tag{3}$$

where S_0 is a constant vector, and C is a Y-dependent matrix. For future reference, write (2) and (3) in the form

$$\mathcal{L} Y = S_0 \ , \tag{4}$$

where the differential operator $\mathcal{L}$ is defined by,

$$\mathcal{L} = A_0 \frac{\partial}{\partial t} + A_i \frac{\partial}{\partial x_i} - \frac{\partial}{\partial x_i} \left(K_{ij} \frac{\partial}{\partial x_j} \right) + C \ . \tag{5}$$

Three possible choices for Y are: (i) entropy variables [19], for which the system (2) becomes *symmetric*, (ii) pressure-primitive variables, and (iii) density-primitive variables [3, 20]. Hauke and Hughes [3, 20] recently showed that the *incompressible limit* of (2) is well defined, and a well-behaved variational formulation of the problem can be derived, when Y is chosen as either the entropy variables or the pressure-primitive variables. Thus, with either of these two variable sets, the compressible and incompressible cases may be combined into a unified framework.

For fully incompressible flows, this approach seems to be less efficient than the direct one, because there is complete decoupling of the energy equation from the momentum and continuity equations. Moreover, in many cases one is interested only in the velocity and pressure fields, and not in the temperatures. Under such circumstances, the full solution of (2) for incompressible flows entails superfluous computational effort. Indeed, traditionally compressible and incompressible viscous flow problems are handled by two separate

codes [1]. However, the combined compressible-incompressible approach becomes attractive when buoyancy is considered, and introduced into the incompressible Navier-Stokes equations via the Boussinesq approximation [21]. In this case, S in (2) is temperature-dependent, and all the equations are coupled, as with compressible flows. We adopt the combined compressible-incompressible approach here, using entropy variables for Y.

The time-dependent Navier-Stokes equations with the Boussinesq approximation can be written in the explicit form

$$\rho_0 \left(\dot{v} + v \cdot \nabla v \right) = \nabla \cdot \text{œ} + \rho_0 \beta g \left(T - T_0 \right) e_z \ , \tag{6}$$

$$\nabla \cdot v = 0 \ , \tag{7}$$

$$\rho_0 c_p \left(\dot{T} + v \cdot \nabla T \right) - \nabla \cdot \left(\kappa \nabla T \right) = 0 \ . \tag{8}$$

Here, v is the velocity vector, T is the temperature, œ is the stress tensor, ρ_0 is a constant reference density of the fluid, T_0 is a reference temperature (corresponding to the density ρ_0), β is the thermal expansivity of the fluid, g is the gravitational acceleration, e_z is a unit vector in the z direction, c_p is the heat capacity, and κ is the heat conductivity of the fluid. Eqs. (6), (7) and (8) are, respectively, the momentum, continuity and energy equations. The last term on the right side of (6) is the buoyancy term, which is responsible for the coupling between the temperature and velocity fields. Gravity is assumed to act in the $-z$ direction. The stress-pressure-velocity relations are given, for a Newtonian fluid, by

$$\text{œ} = -pI + \mu(\nabla v + \nabla v^T) \ , \tag{9}$$

where I is the identity tensor, and μ is the fluid viscosity.

Equations (6)–(9) are recast in the vector form (2), or (4), to fit the combined compressible-incompressible formulation. These equations are accompanied by appropriate boundary and initial conditions, to complete the statement of the problem. In the steady-state case, the time-derivative terms in (6), (8), (2) and (5) are dropped.

3. Variational Formulation

Our starting point is the transformed time-dependent Navier-Stokes equations (2). These equations will be considered even when steady solutions are sought; in this case we shall employ the commonly-used relaxation procedure, in which we discretize the time-dependent problem (2) in space and time, and march rapidly in time to reach a steady state (see Section 4.2).

We define a weak form of (2) based on the time-discontinuous Galerkin Least Squares (GLS) method. This formulation was originally developed for two-dimensional compressible flows [4], and was also used in other configurations by [2] and [3]; see Table 1. The time-discontinuous GLS method is well suited for incorporation in an automated parallel adaptive environment. The method has a firm mathematical foundation, and its stability and accuracy properties have been rigorously established (see [4] and references therein). Moreover, it has the ability to naturally handle moving boundary problems by means of space-time deforming meshes [22].

First we introduce some notation. Given a spatial domain Ω with boundary Γ, consider a space-time domain $\Omega \times I$, where the time interval I is divided into subintervals (time steps), $I_n = (t_n, t_{n+1})$. Let $Q_n = \Omega \times I_n$ (a "space-time slab"), let $P_n = \Gamma \times I_n$, and let $^\circ$ be the unit outward normal on P_n. Also, let W_n^- and W_n^+ denote the values of the time-discontinuous quantity W as $t \to t_n$ from below and above, respectively. The slab Q_n is decomposed into $(N_{el})_n$ space-time elements, denoted Q_n^e. Finally, let $\S(Q_n)$ and $\mathcal{S}_0(Q_n)$ denote the appropriate trial and test spaces, respectively, defined on the slab Q_n.

With this notation, the variational formulation of (2) is stated as follows: within each slab Q_n, find $Y \in \S(Q_n)$ such that for all $W \in \mathcal{S}_0(Q_n)$,

$$\begin{aligned}\int_{Q_n} \left(-\dot{W} \cdot U(Y) - W_{,i} \cdot F_i^{\text{adv}}(Y) + W_{,i} \cdot K_{ij} Y_{,j} - W \cdot S\right) dQ \\ + \int_\Omega \left(W_{n+1}^- \cdot U(Y_{n+1}^-) - W_n^+ \cdot U(Y_n^-)\right) d\Omega \\ + \sum_{e=1}^{(N_{el})_n} \int_{Q_n^e} (\mathcal{L}W) \cdot \tau(\mathcal{L}Y - S_0)\, dQ\end{aligned}$$

$$+ \sum_{e=1}^{(N_{el})_n} \int_{Q_n^e} \delta g^{ij} W_{,i} \cdot A_0 Y_{,j} \, dQ$$

$$= \int_{P_n} W \cdot \left(-F_i^{\text{adv}}(Y) + F_i^{\text{diff}}(Y) \right) \nu_i \, dP \qquad (10)$$

The first and last integrals in (10) are the standard Galerkin terms. The second integral weakly enforces continuity of the solution in time from one slab to the next. Thus, the first, second and last integrals together constitute the discontinuous-Galerkin method [23]. The use of finite element shape functions which are temporally discontinuous enables simple stepping in time, similar to the time marching procedure performed in finite difference schemes. The third and fourth integrals are stabilizing terms; the third is a least squares term, and the fourth is a discontinuity-capturing term. The variational form (10) constitutes the time-discontinuous GLS method (with discontinuity capturing).

The least squares (LS) term is the main stabilization mechanism. Numerical stability is always an issue in finite element formulations of the Navier-Stokes equations. A necessary condition for stability is the Babuška-Brezzi condition, and it is well known that it is not easily satisfied with standard Galerkin methods [24]. For example, equal-order interpolation in space for all the variables leads to a highly unstable numerical scheme, which typically yields non-convergent solutions even for relatively simple problems. The LS term enhances the stability of the method, and allows the use of equal-order interpolation in space, while maintaining the optimal order of accuracy. Equal-order interpolation has the advantage that it easily accommodates high-order elements and p-adaptivity. Stabilization is also needed to eliminate global pollution, in the form of spurious oscillations, due to unresolved boundary layers, large-gradient regions and discontinuities in the exact solution, and also pollution associated with advection-dominated flows [25].

The matrix ϕ appearing in the LS term in (10) is designed to balance stability and accuracy in both the diffusive and advective limits. Definitions of ϕ for the compressible and incompressible Navier-Stokes equations can be found in [4] and [26], respectively. A way to combine both ϕ's is proposed in [20].

Although the LS term eliminates the global pollution due to discontinuities, overshoots

and undershoots may still be present in the immediate vicinity of such features. The fourth integral in (10) is a nonlinear discontinuity capturing (DC) operator, which controls these local oscillations. For more details, as well as the definitions of δ and g^{ij}, see [4, 20].

As can be seen from (10), the LS and DC operators are applied at the element level. The method is consistent in the sense that (10) is satisfied by the exact solution of the problem. This is ensured since both stabilization terms depend on the residual $\mathcal{L}Y - S_0$ (see (4)) in the element, and vanish with this residual. Note also that the weighting $\mathcal{L}W$ in the LS term includes the Y-dependent part of the source S, but does not include its constant part S_0 (see (3)).

Essential and natural boundary conditions can be applied in conjunction with (10). Essential boundary conditions are imposed directly, after expressing them in terms of the Y variables. Natural boundary conditions are imposed by substituting the prescribed value for the corresponding flux in the (last) boundary integral of (10). If no boundary conditions are imposed on some part of the boundary, the fluxes are evaluated there from the current values of Y.

4. Computational Scheme

Fig. 1 summarizes the overall computational scheme. First, an initial mesh is generated. Then, the domain is decomposed into partition subdomains, so as to balance the initial load among the parallel processors. Then, the main analysis loop is started. This loop includes five main phases: (a) the "form phase," in which the element matrices and vectors and the global equations are formed; (b) the "solve phase," where the global algebraic equations are solved; (c) the error estimation phase, where some specified measure of the local error (the error indicator) is calculated for all the elements; (d) the mesh adaptation phase, in which the mesh is locally refined and/or coarsened based on the error indicator values; (e) the load balancing phase, where elements are migrated between processors in order to redistribute the load in a uniform manner among the processors.

The procedures used for the mesh generation, mesh adaptation, domain partitioning, load

balancing and element array formation are all linked to the so-called Parallel Mesh Database and Attribute Manager, which store, handle and supply information on mesh entities and physical data.

Now we shall describe in some detail the various ingredients of this computational scheme.

4.1. Initial Mesh Generation and the Attribute Manager

Spatial meshes are generated via the Finite Octree technique [27], which constructs general unstructured meshes of tetrahedral elements. The first step in meshing a model region is to develop an octree; this is a multi-level tree whose basic unit on each level is a mesh cell divisible to eight subcells (octants) on a lower level. This octree is constructed so that it reflects the mesh information and is consistent with the triangulation on the boundary of the model region. A one level difference of octants sharing one or more edges is enforced during this process to control smoothness of the mesh gradations. Once the octree is generated, the octants are classified as interior or boundary octants. Some interior octants are reclassified as boundary octants if they are sufficiently close to mesh entities on the boundary. The next step consists of meshing the interior octants. The last step in the mesh generation is to connect the boundary triangulation to the interior octants, using an operation called "face removal." The Finite Octree mesh generation procedure has been proved to be an extremely strong tool for general three-dimensional geometries.

The Attribute Manager consists of a special procedure for storing and handling the physical attribute information required to support the analysis, e.g., boundary conditions, initial conditions and material properties. This information is tied to the geometric model definition, rather than to the discrete model, and is defined in a general hierarchical form [28]. This is in contrast with the common procedure of defining the physical attribute information directly in terms of the discrete model. The Attribute Manager is especially effective in an adaptive environment.

4.2. Interpolation in Time and Space

The choice of finite element shape functions in space and time, in conjunction with the GLS time-space variational formulation described in Section 3, determines the definition of the element arrays, and thus completes the information needed for the "form phase."

In the present scheme, piecewise-constant interpolation is used in time. This choice is permitted in the time-discontinuous GLS method, and is very convenient and efficient for steady-state analysis. (It is less appropriate for transient analysis). A single Newton iteration is performed at each time step.

In transient analysis, a time-step size is specified which may vary in time, but is uniform in space. In steady-state analysis, the time-step need not be uniform throughout the mesh, but is determined locally, e.g., to achieve a specified element Courant number to reach steady state rapidly. The Courant number measures the estimated number of elements over which the information propagates in a single time step. Using this procedure, the flow information propagates at a nearly optimal rate throughout the spatial domain.

In space, continuous piecewise-linear spatial interpolation is used in all the variables. This convenient choice, which is never possible in standard mixed Galerkin formulations, is made possible by the LS stabilization. In addition to programming convenience, a major advantage of an equal-order interpolation is that it is amenable to easy generalization to higher-order elements. As a consequence, the present scheme can be extended, in a straightforward manner, to general *hp*-adaptive strategies.

4.3. Parallel Procedures

The finite element scheme used here is designed to work in an automatic parallel adaptive environment. One important aspect of parallelization is the partitioning of data among the processors, given the irregular communication patterns that characterize unstructured meshes. Domain decomposition is used for partitioning the computational domain into subdomains, each being assigned to a processing node. The subdomains exchange data with each other through the subdomain boundaries.

The partitioning of the initial mesh is done using a tree-based moment-of-inertia Recursive Bisection (RB) algorithm [29]. The whole mesh is first loaded into one processor and then recursively split in half and sent to other processors in parallel. The splitting is done by the linear time median algorithm. In this way, the time that initial partitioning by parallel RB takes is $O(N)$, where N is the number of elements.

The message passing paradigm is employed in the implementation of the scheme. Parallel implementation of the form phase is straightforward, with no communication among the processors. The solve phase is based on the use of the preconditioned GMRES algorithm; see Section 4.6 for the details.

The Parallel Mesh Database (see Fig. 1) stores, handles and supplies information on mesh entities. It has three main roles: (a) to provide a common interface and a single library for all the mesh related application, namely mesh generation, mesh refinement/coarsening and finite element analysis; (b) to provide information on adjacency relations among shared mesh entities on different processors; (c) to provide a mesh migration algorithm which facilitates arbitrary mobility of mesh entities on processors. The Mesh Database defines the mesh in terms of the topological entities of the mesh (vertices, edges, faces and regions) with pointers between them.

The data structures used in the scheme provide fast query of subdomain boundary information. Such information includes adjacencies for entities located on more than one partition, the number and list of adjacent processors given an entity type adjacency, and so on. In addition to these queries, fast update procedures are used for the refinement/coarsening and element migration/load balancing components of the solver. The fast query and update are achieved via a topological-entity hierarchy data structure, which provides a two-way link between the mesh entities of consecutive order, i.e., regions$\leftrightarrow$faces, faces$\leftrightarrow$edges and edges$\leftrightarrow$vertices. Using this hierarchy, any entity relationship can be derived by local traversals. The entities on subdomain boundaries are augmented with links, which point to the location of the corresponding entity on the neighboring processor. Each subdomain boundary entity can have attached to it either the complete or the minimal set of interprocessor links. A switching mechanism is employed to allow both representations to be used disjointly.

Special procedures are used to migrate elements between processors for the purpose of redistributing the mesh in order to achieve *load balance* [30, 31]. The migration routines are implemented in three stages: (i) the element mesh and its attribute data is packed into messages and sent, (ii) packed elements are received and unpacked, (iii) the inter-processor links are updated. Provided procedures allow each processor to send and receive multiple migrations of elements. An element-based dynamic load balancing scheme is used, which iteratively migrates elements from heavily loaded to less loaded processors. It is based on letting each processor request load from a heavily loaded neighbor. The hierarchic load request is represented by a tree, upon which a scan operation is performed. The complexity of this scan is $O(\log T)$, where T is the number of nodes in the tree [30].

These parallel procedures result in a scalable scheme (see Section 4.7) that runs on a distributed-memory MIMD parallel computer. The simulations described in Section 5 are performed on an IBM SP-2 computer, at Rensselaer Polytechnic Institute (36 processors), and at Cornell University (512 processors).

4.4. Error Estimation

Local error control of nonlinear hyperbolic and parabolic systems based on rigorous mathematical foundations, i.e., on local a-posteriori error estimates, is very difficult in general (as opposed to the linear elliptic case, for example). Traditionally, error indicators are used in place of error estimators when the former are not available. Although error indicators do not necessarily measure the real discretization error, they do indicate, if chosen carefully, local regions of the computational domain associated with high errors, and thus can be very useful in an adaptive environment.

Two types of error indicators are currently implemented in the scheme described here; both can be applied to one or more of the physical variables (typically density, pressure or temperature). The first error indicator is the magnitude of the gradient of the variable chosen. This indicator is not based on solid mathematical ground, but it is useful in practice when relatively small local regions of large gradients of the solution are present. Error

indicators based on the magnitude of density or pressure gradients are usually effective in tracking shocks in compressible supersonic flows. A nodal error indicator which is appropriate when thin boundary layers are present is

$$e_I = A_I(|\nabla v|) \quad , \quad v \equiv |v| \tag{11}$$

where v_I is the velocity vector at node I, and $A_I(\cdot)$ is the average value of a discontinuous quantity at node I. This error indicator measures the local change in the magnitude of the nodal velocity. It should be noted that v, the velocity magnitude, is taken to be linearly interpolated in each element (which is actually inconsistent with the linear interpolation in each of the velocity components). In the simulations of Section 5.2, we use a combination of this "boundary layer error indicator" (11) and one which measures temperature gradient.

The second error indicator which is implemented in the scheme is related to the magnitude of the second derivatives of the variable chosen [32]. This indicator is appropriate for linear elements, since it attempts to measure the error by estimating the leading (quadratic) term truncated. It has the form

$$e_I = \frac{h^2 H_I(\phi)}{h A_I(|\nabla \phi|) + \epsilon M_I(|\phi|)} \, , \tag{12}$$

where h is the mesh size parameter, ϕ is the solution variable being monitored, $H_I(\phi)$ is a measure of the nodal magnitude of the second derivatives of ϕ, $M_I(\cdot)$ is the weighted average of a quantity over all the nodes which surround node I, ϵ is a tuning parameter, and $A_I(\cdot)$ is defined as before. The second derivative measure H_I is computed using a variational recovery technique [32].

For each edge, the value of the error indicator is computed by averaging the associated two nodal values of the indicator. These edge values are then used to control mesh adaptation. Maximum and minimum thresholds, M and m, are supplied for the error indicators, so that the edge is refined if the error associated with at least one of the selected physical variables is higher than M, and is collapsed if the error is smaller than m for all selected variables. In some cases, especially if sudden changes in the character of the solution are expected, m is specified as zero, to turn off mesh coarsening. The threshold M is chosen so that only about 5%-15% of the elements in the mesh are refined, to prevent an excessive computational effort

in the following analysis cycle. Parallel implementation of the error indicator calculation is straight forward.

The adaptive process starts from an initial mesh (see Fig. 1), which is generated using the Finite Octree technique. This mesh should be fine enough to capture the main features of the solution, but ideally not much finer than this. Starting from an initial mesh which is too crude is potentially dangerous, because it may lead to situations where a certain feature of the exact solution (say, small local oscillations), which occurs at a scale much smaller than that of the initial mesh size parameter, may remain undetected by the error estimation phase.

In transient analysis, an adaptive step, which includes error estimation and mesh adaptation, is invoked every specified number of time steps. In a steady-state analysis, the procedure adopted in the present work is as follows: First, an initial mesh is generated. Then, local time-stepping is performed with this mesh for a sufficient number of time steps, until the residual norm is reduced below a certain specified level, indicating that a steady state is reached. Then, one step of mesh adaptation is performed, using one of the error indicators mentioned above. After this step, another cycle of local time stepping is performed, until steady state is reached again. This procedure is repeated for a desired number of cycles. A typical behavior of the residual norm is: possible initial oscillations, then reducing monotonically until the first mesh adaptation, then slightly increasing, and again reducing monotonically until the next adaptation step, and so on. In cases where a more oscillatory behavior is observed, this typically indicates that a steady state was not reached successfully, possibly because for the values of the physical parameters used one does not exist or is unstable.

4.5. Mesh Adaptation

The mesh level adaptive scheme combines local refinement, coarsening and triangulation optimization using local retriangulations. The refinement step makes use of subdivision patterns (templates). All possible patterns are considered and implemented to allow for

speed and avoid possible over-refinement. The coarsening step is based on the edge collapsing technique. This procedure does not require storage of any history information and therefore does not depend on the refinement procedure. Triangulation optimization is necessary to prevent mesh quality degradation. It is based on the iterative local retriangulation of well defined polyhedra, using the two dual techniques of edge removal and multi-face removal [33].

Since refinement uses templates, its parallel implementation presents no difficulties. First, mesh faces on the partition boundary are triangulated. Face level interprocessor links are set up for the child faces of the mesh faces on the partition boundary. Then, mesh regions are triangulated without communication involved. Any mesh edge carrying minimal interprocessor links transfers link information to its two child edges.

The efficient parallelization of local adaptive mesh refinement and coarsening is discussed in [33]. Efficient parallel retriangulation of polyhedra is performed in three steps: (i) retriangulate polyhedra which are fully interior to the partition, (ii) shift the partition boundary using element migration techniques, (iii) retriangulate polyhedra that are now fully accessible due to the shift. Multiple processors can request the same off-processor region, and in such case the processor with the lowest identification number is the one which is granted priority. Triangulation optimization is an iterative process, and naturally, shifting the partition boundary constantly in one direction would quickly create a load imbalance. Therefore, after each adaptive step, a load balancing step is applied (see Section 4.3).

4.6. Equation Solving

The nonlinear system of algebraic equations obtained from finite element discretization in space and time is linearized using Newton iterations. The resulting nonsymmetric linear system is solved iteratively using preconditioned GMRES. Preconditioning is performed by means of a nodal block-diagonal scaling transformation. Given the nonsymmetric system

$$Tp = F \,, \tag{13}$$

the GMRES algorithm [34] starts from an initial guess p_0, and attempts to find an approximate solution $p_0 + z$ to (13), where z is a vector in the Krylov space $\mathcal{K} = (r_0,\ Tr_0,\ ...,\ T^{k-1}r_0)$

and $r_0 = F - Tp_0$. The vector z is found by solving the minimization problem

$$\min_{z \in \mathcal{K}} ||F - T(p_0 + z)|| \,, \tag{14}$$

using the QR algorithm. An orthogonal basis of $\mathcal{K}$ is obtained by a Gram-Schmidt procedure. This procedure involves matrix-vector multiplication and vector dot products, which constitute the computing-intensive part of the algorithm.

The matrix-vector multiplications are implemented in parallel and necessitate the exchange of data through the interprocessor boundaries. In order to overlap communication and computation for efficiency reasons, these operations are realized following a four-step procedure on each processor: (i) sending information on the inter-processor boundaries to each neighboring processor; (ii) performing computations involving only data associated with nodes that lie within the internal volume of the partition; (iii) receiving information on the inter-processor boundaries from all the neighboring processors; (iv) performing computations involving only data associated with nodes that lie on the inter-processor boundaries.

For the implementation of the vector dot product operations, nodes that lie on the inter-processor boundaries are randomly split, so that two partitions that share an internal boundary are assigned only a subset of the nodes of that boundary. Then, each processor performs the dot product involving nodes contained in its internal volume and its subset of nodes on the partition boundaries. The sum of all the processor results yields the global dot product result.

The minimization problem (14) is written in terms of an upper Hessenberg matrix, whose dimension is the same as the dimension of the space $\mathcal{K}$, and whose entries are the results of the dot products performed during the orthogonalization procedure. At the end of the Gram-Schmidt procedure, each processor has complete knowledge of the upper Hessenberg matrix and it is therefore able to perform the solution of (14) independently without any inter-processor communication. Similarly, once convergence is achieved in the GMRES solver, the Newton update (i.e., addition of the incremental solution to the solution from the previous Newton iteration) is performed by each processor independently with no communication.

4.7. Overall Performance

With the features described in the previous sections, the proposed scheme contains the four desired properties mentioned in Section 1: flexibility (mesh and physical attribute handling), stability (LS and DC operators), accuracy (adaptivity, ability to handle high-order elements), and speed (parallel environment, local time-stepping).

The parallel adaptive procedure enables the solution of large problems, with CPU times which are much smaller, for the same level of accuracy, than those required in a serial code using a uniform mesh. A single steady-state simulation of those reported in Section 5.2, with an initial mesh of about 140,000 elements, with two adaptive steps leading to a final mesh of about 400,000 elements, on 16 processors, took about 11 hours of computing, including the initial mesh generation and partitioning. See also the comparison given in Section 5.1.

The parallel environment is associated with an additional memory requirement compared to a serial code, but it is usually marginal. This additional storage is handled by the Parallel Mesh Database, which defines the mesh in terms of its topological entities with pointers between them. The mesh on each processor is stored, and in addition special treatment is given to those mesh entities which are on the boundary of the partition. For them, an entity neighbor is pointed by indicating the processor it is on and its local memory pointer on that processor. There is also a need for a linked list of the entities on the interprocessor boundaries to support the gather/scatter operations performed by the GMRES solver. This comprises the added memory needed in the parallel scheme, and it is typically small compared to the total memory volume needed. The size of the added memory is proportional to the number of mesh entities on the interprocessor boundaries.

The scheme has a high degree of scalability. This is demonstrated by Fig. 2, which shows the speed-up gained relative to a serial run as a function of the number of processors used. These results correspond to one of the simulations reported in Section 5.2. When the number of processors increases from 1 to 2, a speed-up of about 2 is obtained. When 16 processors are used, a speed-up of 14 is obtained. Thus, the slope of the scalability curve is about 0.875 (vs. the ideal unit slope). The 12% reduction in the slope is due to communication overhead.

Communication cost is mainly due to the load balancing procedure, which is dominated by interprocessor communications. However, this cost is maintained under a reasonable level (as Fig. 2 shows) owing to the special parallel procedures mentioned in Section 4.3. The communication cost is especially small for relatively large problems, where a large number of finite elements is used per processor.

5. Numerical Examples and Results

5.1. Mach 3 Flow Over a Plate

To demonstrate the capability of the scheme to handle compressible viscous flows, we consider steady flow of air over a flat plate, with inflow conditions corresponding to M= 3 and Re=1000 (see [4], [20]). From the leading edge of the plate, a thick boundary layer and a shock develop. The solution is two-dimensional, but here it is obtained using the three-dimensional computational domain $-0.2 \leq x \leq 1.2$, $0 \leq y \leq 0.8$, $0 \leq z \leq 0.1$, where the "thickness" 0.1 in the z direction is arbitrary, and is chosen for numerical convenience. The boundary of this domain is divided into seven faces, with the following boundary conditions:

$x = -0.2$ (inflow):	$\rho = 1$, $v_1 = 1$, $v_2 = v_3 = 0$, $T = 2.769 \cdot 10^{-4}$.
$y = 0.8$ (inflow):	same as above.
$y = 0$, $x \geq 0$ (plate surface):	no-slip, $T = 7.754 \cdot 10^{-4}$.
$y = 0$, $x < 0$ (symm.):	slip condition, zero normal heat flux.
$z = 0$ (symm.):	same as above.
$z = 0.1$ (symm.):	same as above.
$x = 1.2$ (outflow):	no boundary conditions.

The gas is assumed to be ideal, with Sutherland's viscosity law, $\mu = 0.0906T^{1.5}/(T + 0.0001406)$, and Pr=0.72. The initial mesh used is shown in Fig. 3(a). It consists of 6486 tetrahedral elements and 1770 nodal points (with 5 degrees of freedom per node). The figure shows the trace of this mesh on the xy plane. Two cycles of adaptive mesh refinements

were applied (see Section 4.4). The magnitude of the density gradient served as an error indicator. The final mesh thus obtained is shown in Fig. 3(b). Mesh refinement at and near the boundary layer and the shock is clear.

Figs. 4(a)–(d) show the contours of the finite element solution for the density, Mach number, temperature and pressure, respectively. The results agree well with those obtained in [4] and [20]. The latter results were obtained using a serial, two-dimensional code, with a uniform structured mesh which is as fine as the finest level in the mesh of Fig. 3(b). Thus, we obtain a similar solution to that of [4, 20] while achieving significant saving in the number of degrees of freedom. It is apparent from Figs. 3(b) and 4(a)–(d) that the strong features in the solution, namely the boundary layer and the shock, are captured very effectively by the adaptive scheme.

It should be noted that the final mesh is not uniform in the "thickness direction" z of the plate; elements in the fine regions are small in all three dimensions, since the tetrahedral elements generated during the adaptive step are well-proportioned. The coarse regions of the mesh consist of two elements in the z direcition, whereas in the fine regions there are up to eight elements in this direction.

This analysis was performed with 8 and 16 processors of an IBM SP/2 computer. Figs. 5(a) and 5(b) show, respectively, the subdivision of the initial mesh and of the final mesh, in the 8-processor case. As more elements are concentrated in the regions of the shock and the boundary layer, the processor subdivision is changed by the dynamic load balancing procedure discussed in Section 4.3.

5.2. LEC Melt Flows

The finite element scheme is now applied to a problem of divergence-free buoyancy-driven flow in a cavity. This is a simplified single-phase model for the steady melt flow in a LEC crystal growth process (see Section 1). The governing equations are the Navier-Stokes equations with the Boussinesq approximation, (6)–(9). We use this model problem to check the effectiveness of applying the new computational framework to problems of this type.

We consider buoyancy driven flow in InP melt contained in a small cylindrical crucible with a curved bottom, with radius r_c and height H. The upper surface of the melt (including the melt-crystal and melt-encapsulant interfaces) is assumed to be flat. We introduce a cylindrical system of coordinates (r, θ, z), with the z axis coinciding with the crucible axis and pointing upwards. We denote the part of the boundary defined by the crucible walls and bottom by Γ_c. The melt-solid interface is defined by the circular domain $z = H$, $0 \leq r \leq r_s$, and is denoted Γ_s. The melt-encapsulant interface is defined by the annular domain $z = H$, $r_s < r \leq r_c$, and is denoted Γ_f.

The boundary conditions considered are as follows. A no-slip condition is applied on the whole boundary. On Γ_c, a given constant temperature T_c is prescribed. On Γ_s, the temperature is equal to the melting-point temperature T_{mp}. On Γ_f, zero normal heat flux is assumed. The no-slip condition at the melt-encapsulant interface is reasonable since the encapsulant is much more viscous than the melt. The values of the parameters are: $r_c = 3.4$cm, $r_s = r_c/2$, $H = r_c$, $\rho_0 = 5.05$ g/cm^3, $\mu = 0.0081$ g/cm-s, $\kappa = 0.23$ W/cm-K, $c_p = 0.42$ J/g-K, $\beta = 4.44 \cdot 10^{-4}$ K^{-1}, $T_0 = T_{mp} = 1335$ K. For later reference, we define the Grashof number, which is the ratio of buoyancy to viscous forces,

$$\mathrm{Gr} = \rho_0^2 \beta g r_c^3 (T_c - T_{mp})/\mu^2 \ . \tag{15}$$

We control the value of Gr by assigning an appropriate value to the crucible wall temperature, T_c. It should be noted that although the length scale of the problem is realistic for InP crucibles (they are much smaller than silicon growth crucibles), the Gr values considered here are smaller than the realistic ones in a LEC process [35, 6]. This simplification is not so uncommon in crystal growth modeling, since CZ and LEC systems with high Gr numbers are associated with several complicated interacting flow instability phenomena [16].

The flow problems are solved using 8, 16 and 32 parallel processors. As initial conditions, we set the velocity equal to zero and the temperature equal to the crucible wall temperature T_c, everywhere. The adaptive procedure described in Section 4.4 is used, with an error indicator based on a combination of the boundary layer indicator (11) and the magnitudes of the temperature gradient. After each adaptive mesh-refinement step, time marching is

performed, with a local time step which is determined by setting the algorithm Courant number equal to 10, until the residual norm is reduced by a factor of 10^{-3}. The dimension of the Krylov space used in the GMRES solver is chosen to be 25. In these simulations the use of the DC term is suppressed.

Fig. 6(a) shows the initial mesh used, which contains about 140,000 elements. Figs. 6(b) and 6(c) show, respectively, vertical cross-sections through the meshes obtained after the first and second adaptive refinements. These meshes contain, respectively, about 230,000 and 400,000 elements, and similar numbers of degrees of freedom. The latter is the final mesh used in this example. Fig. 6(d) is a three-dimensional view of this final mesh. In each refinement step, about 8% of the elements in the current mesh associated with the largest error indicator values are refined.

The mesh refinement near the walls of the crucible, and especially the top boundary, where thin boundary layers are present, is apparent. The asymmetric refinement seen in the final mesh (Figs. 6(c) and 6(d)) is due to the fact that the original mesh, being an unstructured mesh of tetrahedral elements, is not axisymmetric in nature. The error indicator calculation is slightly sensitive to the local shape and size of the elements, and thus leads to small perturbations in the error indicator values around the specified threshold value, and hence to a slightly asymmetric refinement. However, this asymmetry is reduced as the mesh is further refined, and it does not lead to significant asymmetry in the solution itself.

Figs. 7(a) and 7(b) show, respectively, the initial and final mesh partitioning, when 8 processors are used. The differences between the two are due to the dynamic load balancing procedure employed. It is clear that the final processor distribution is such that more effort is concentrated near the top surface of the melt, where most of the refinement has taken place.

Figs. 8–10 show the finite element results. Fig. 8 is the temperature contour plot in the vertical yz plane, for $\mathrm{Gr}=3.3 \cdot 10^4$. Figs. 9(a), 9(b) and 9(c) show the velocity pattern in a vertical plane for $\mathrm{Gr}=3.3 \cdot 10^4$, $3.3 \cdot 10^5$ and $3.3 \cdot 10^6$, respectively. Fig. 10 shows the velocity distribution in the horizontal plane $z = 3.2$, slightly below the top surface of the melt. These results are similar in principle to those presented in [13], [7] and [8]. At the heated vertical

wall of the crucible the melt rises due to buoyancy, and then turns radially inwards at the melt surface. This gives rise to the Rayleigh-Bénard toroidal cell flow seen in Figs. 9(a)–(c). The flow cell is not centered; its center is closer to the crucible wall, as in the "bulk-flow" model in [13]. The radial flow shown in Fig. 10, which is the upper part of the toroidal cell, exists below the boundary layer adjacent to the top surface of the melt. In CZ flows, where no encapsulant is used, this radial flow appears on the free surface of the melt.

We note, however, that these results are expected to differ significantly from those obtained in a fully-coupled LEC simulation. This has been demonstrated in [13], where the results of a simple CZ "bulk-flow" model and of a coupled thermal-capillary model, were compared in various configurations. The significant differences in the results of the two models, especially at high Gr numbers, show that flow coupling effects are important, and thus the uncoupled model is too simple to yield accurate information.

Although our finite element scheme solves three-dimensional problems, the steady-state results shown here are all axisymmetric. Both experimental evidence and numerical calculations with realistic value of Gr, show that CZ and LEC processes always involve flows which are time-dependent and fully three-dimensional. For silicon with buoyancy-driven flows, transition to time-dependent three-dimensional solutions typically occurs at $\mathrm{Gr} \approx 5 \cdot 10^6$ [7, 8].

The extension of the current methodology to the three-dimensional regime of realistic Gr values is currently underway. Time-dependent simulations and some preliminary three-dimensional results for large Gr are reported in [36]. These results show that transition to unsteady three-dimensional flows occurs at about $\mathrm{Gr} = 10^7$, and that these flows are characterized by short transient, followed by periodic oscillations with period of about 3.5 seconds. The simulations in [36] also include flows in silicon and InP melts with the combined effects of natural convection due to buoyancy and forced convection due to crystal and crucible rotation.

6. Concluding Remarks

In this paper we have described a new parallel adaptive finite element scheme for the large-scale analysis of viscous flows. We demonstrateded how this scheme can be applied to both compressible and incompressible flows, and to problems governed by the Navier-Stokes equations with the Boussinesq approximation.

Further developments are currently underway. First, the spatially linear finite elements used here may be replaced by higher-order elements. This extension is rather straight forward, and would lead to a highly accurate and stable scheme. Stability is guaranteed owing to the GLS formulation. Incorporating a general hierarchy of high-order finite elements would enable the use of *hp*-adaptive strategies, which have known advantages over *h*-adaptive schemes in many cases.

Second, the capability of efficient time-dependent analysis is being added to the present scheme. Effective time integration, which is essential in a large-scale nonlinear analysis, will be achieved using high-order singly implicit Runge-Kutta (SIRK) methods [37].

Finally, we intend to extend and apply the methodology described in this paper to the coupled multiphase transient analysis of InP high-pressure crystal growth systems. The difficulties entailed in the realistic analysis of the full LEC system including all the important effects are enormous, and we believe that the current methodology serves as an excellent framework for such simulations.

Acknowledgments

This work is partly supported by AFOSR, as part of the ARPA/AFOSR Consortium for Crystal Growth Research, under grant No. F49620-95-1-0407. The authors would also like to thank Dr. Carlo Bottasso, Dr. Simon Brandon, Prof. Leo Franca and Prof. Alex Ostrogorsky for their very helpful remarks.

References

[1] S. Mittal and T.E. Tezduyar, "Massively Parallel Finite Element Computation of Incompressible Flows Involving Fluid-Body Interactions," *Comp. Meth. Appl. Mech. Engng.*, 112, 253–282, 1994.

[2] C.L. Bottasso, H.L. de Cougny, M. Dindar, J.E. Flaherty, C. Ozturan, Z. Rusak and M.S. Shephard, "Compressible Aerodynamics Using a Parallel Adaptive Time-discontinuous Galerkin Least-Squares Finite Element Method," in *Proc. 12th AIAA Applied Aerodynamics Conference*, no. 94-1888, Colorado Springs, CO, June 1994, AIAA.

[3] G. Hauke and T.J.R. Hughes, "A Unified Approach to Compressible and Incompressible Flows," *Comp. Meth. Appl. Mech. Engng.*, 113, 389–395, 1994.

[4] F. Shakib, T.J.R. Hughes and Z. Johan, "A New Finite Element Formulation for Computational Fluid Dynamics: X. The Compressible Euler and Navier Stokes Equations," *Comp. Meth. Appl. Mech. Engng.*, 89, 141–219, 1991.

[5] D.T.J. Hurle and B. Cockayne, "Czochralski Growth," in *Handbook of Crystal Growth*, D.T.J. Hurle, ed., Vol. 2A, Chap. 3, pp. 99–212, North-Holland, Amsterdam, 1994.

[6] G. Müller and A. Ostrogorsky, "Convection in Melt Growth," in *Handbook of Crystal Growth*, D.T.J. Hurle, ed., Vol. 2B, Chap. 13, pp. 709–819, North-Holland, Amsterdam, 1994.

[7] M.J. Crochet, P.J. Wouters, F.T. Geyling and A.S. Jordan, "Finite Element Simulation of Czochralski Bulk Flow," *J. Crystal Growth*, 65, 153–165, 1983.

[8] A. Bottaro and A. Zebib, "Three Dimensional Thermal Convection in Czochralski Melt," *J. Crystal Growth*, 97, 50–58, 1989.

[9] M. Mihelčić and K. Wingerath, "Instability of the Buoyancy Driven Convection in Si Melts During Szochralski Crystal Growth," *J. Crystal Growth*, 97, 42–49, 1989.

[10] H.-J. Leister and M. Perić, "Numerical Simulation of a 3D Czochralski Melt Flow by a Finite Volume Multigrid Algorithm," *J. Crystal Growth*, 123, 567–574, 1992.

[11] Q. Xiao and J.J. Derby, "Three-dimensional Melt Flows in Czochralski Oxide Growth: High-resolution, Massively Parallel, Finite Element Computations," *J. Crystal Growth*, 152, 169–181, 1995.

[12] P.A. Sackinger, R.A. Brown and J.J. Derby, "A Finite Element Method for Analysis of Fluid Flow, Heat Transfer and Free Interfaces in Czochralski Crystal Growth," *Int. J. Numer. Meth. Fluids*, 9, 453–492, 1989.

[13] Q. Xiao and J.J. Derby, "Bulk-flow Versus Thermal-capillary Models for Czochralski Growth of Semiconductors," *J. Crystal Growth*, 129, 593–609, 1993.

[14] J.J. Derby, S. Brandon, A.G. Salinger and Q. Xiao, "Large-scale Numerical Analysis of Materials Processing Systems: High-temperature Crystal Growth and Molten Glass Flows," *Comp. Meth. Appl. Mech. Engng.*, 112, 69–89, 1994.

[15] H. Zhang and V. Prasad, "A Multizone Adaptive Process Model for Low and High Pressure Crystal Growth," *J. Crystal Growth*, 155, 47–65, 1995.

[16] F. Dupret and N. van den Bogaert, "Modelling Bridgman and Czochralski Growth," in *Handbook of Crystal Growth*, D.T.J. Hurle, ed., Vol. 2B, Chap. 15, pp. 875–1010, North-Holland, Amsterdam, 1994.

[17] F. Brezzi and M. Fortin, *Mixed and hybrid finite element methods*, Springer-Verlag, New York, 1991.

[18] P.M. Adornato and R.A. Brown, "Petrov-Galerkin Methods for Natural Convection in Directional Solidification of Binary Alloys," *Int. J. Numer. Meth. Fluids*, 7, 761–791, 1987.

[19] F. Chalot, T.J.R. Hughes and F. Shakib, "Symmetrization of Conservation Laws with Entropy for High-Temperature Hypersonic Computations," *Comp. Sys. Engng.*, 1, 495–521, 1990.

[20] G. Hauke, A unified Approach to Compressible and Incompressible Flows and A New Entropy-consistent Formulation of the $k - \epsilon$ Model, Ph.D. Thesis, Stanford University, 1995.

[21] B. Gebhart, Y. Jaluria, R.L. Mahajan and B. Sammakia, *Buoyancy-Induced Flows and Transport*, Hemisphere Publishing, Washington, 1988.

[22] T.E. Tezduyar, M. Behr, S. Mittal and J. Liou, "A New Strategy for Finite Element Computations Involving Moving Boundaries and Interfaces — The Deforming Spatial Domain/Space Time Procedure: I. The Concept and the Preliminary Tests," *Comp. Meth. Appl. Mech. Engng.*, 94, 353–371, 1992.

[23] C. Johnson, *Numerical Solutions of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, 1987.

[24] T.J.R. Hughes, *The Finite Element Method*, Prentice-Hall, New Jersey, 1987.

[25] T.J.R. Hughes, L.P. Franca and G. Hulbert, "A New Finite Element Formulation for Computational Fluid Dynamics: VIII. The Galerkin/Least-Squares Method for Advective-Diffusive Equations," *Comp. Meth. Appl. Mech. Engng.*, 73, 173–189, 1989.

[26] L.P. Franca and S.L. Frey, "Stabilized Finite Element Methods: II. The Incompressible Navier-Stokes Equations," *Comp. Meth. Appl. Mech. Engng.*, 89, 141–219, 1992.

[27] M.S. Shephard and M.K. Georges, "Automatic Three-dimensional Mesh Generation by the Finite Octree Technique," *Int. J. Numer. Meth. Engng.*, 32, 709–749, 1991.

[28] M.S. Shephard, "The Specification of Physical Attribute Information for Engineering Analysis," *Eng. with Comp.*, 4, 145–155, 1988.

[29] M.S. Shephard, J.E. Flaherty, H.L. de Cougny, C. Ozturan, C.L. Bottasso and M.W. Beall, "Parallel Automated Adaptive Procedures for Unstructured Meshes," in *Parallel Computing in CFD*, Vol. R-807, pp. 6.1–6.49, AGARD, Nevilly-sur-Seine, France, 1995.

[30] H.L. de Cougny, K.D. Devine, J.E. Flaherty, R.M. Loy, C. Ozturan and M.S. Shephard, "Load Balancing for the Parallel Solution of Partial Differential Equations," *Applied Numerical Mathematics*, 16, 157–182, 1994.

[31] C. Ozturan, *Distributed Environment and Load Balancing for Adaptive Unstructured Meshes*, PhD thesis, Rensselaer Polytechnic Institute, Troy, NY, 1995.

[32] R. Lohner, "An Adaptive Finite Element Scheme for Transient Problems in CFD," *Comp. Meth. Appl. Mech. Engng.*, 61, 323–338, 1987.

[33] C. Ozturan, H.L. de Cougny, M.S. Shephard and J.E. Flaherty, "Parallel Adaptive Mesh Refinement and Redistribution on Distributed Memory Machines," *Comp. Meth. Appl. Mech. Engng.*, 119, 123–137, 1994.

[34] Y. Saad and M. Schultz, "A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems," *SIAM J. Sci. Stat. Comp.*, 7, 856–869, 1986.

[35] G.W. Iseler, "Advances in LEC Growth of InP Crystals," *J. Electronic Materials*, 13, 989–1011, 1984.

[36] D. Givoli, J.E. Flaherty and M.S. Shephard, "Simulation of Czochralski Melt Flows Using Parallel Adaptive Finite Element Procedures," *Modelling Simul. Mater. Sci. Eng.*, to appear.

[37] P.K. Moore and J.E. Flaherty, "High-order Adaptive Finite Element-Singly Implicit Runge-Kutta Methods for Parabolic Differential Equations," *BIT*, 33, 309–331, 1993.

Feature	**Shakib** *et al.* [4]	**Bottasso** *et al.* [2]	**Hauke & Hughes [3]**	**Present**
Space-time GLS	X	X	X	X
Viscous flows	X		X	X
N.S. eqs. + Boussinesq approx.				X
Compressible-incompressible			X	X
Three dimensions		X		X
Unstructured meshes		X		X
Parallel implementation		X		X
Adaptivity		X		X
Boundary-layer error indicator				X

Table 1. Comparison of computational features used in three previous works and in the present work.

Captions for Figures

Fig. 1. Flowchart of the overall solution procedure.

Fig. 2. Scalability of the numerical scheme.

Fig. 3. Flow over plate: finite element meshes. (a) Initial mesh, (b) Final mesh, after two steps of adaptive refinement.

Fig. 4. Flow over plate: finite element results. Contour lines are shown for (a) density, (b) Mach number, (c) temperature, (d) pressure.

Fig. 5. Flow over plate: partitioning of the mesh to 8 parallel processors. (a) Initial mesh partitioning, (b) final mesh partitioning.

Fig. 6. Melt flow of InP: finite element meshes. (a) Initial mesh, (b) vertical section through mesh after one adaptive refinement, (c) vertical section through mesh after a second adaptive refinement, (d) final mesh.

Fig. 7. Melt flow of InP: mesh partitioning, with 8 processors. (a) Initial mesh partitioning, (b) final mesh partitioning.

Fig. 8. Melt flow of InP: temperature contour lines in the vertical yz plane, for Gr=$3.3 \cdot 10^4$.

Fig. 9. Melt flow of InP: velocity pattern in the vertical yz plane. (a) Gr=$3.3 \cdot 10^4$, (b) Gr=$3.3 \cdot 10^5$, (c) Gr=$3.3 \cdot 10^6$.

Fig. 10. Melt flow of InP: velocity distribution in the horizontal plane $z = 3.2$, slightly below the top surface of the melt.