CFD methods for three-dimensional computation of complex flow problems

Tayfun E. Tezduyar

Abstract

This paper provides an overview of some of the CFD methods developed by the Team for Advanced Flow Simulation and Modeling (TAFSM) [http://www.mems.rice.edu/TAFSM/]. The paper also provides many examples of three-dimensional flow simulations carried out with these CFD methods and advanced parallel supercomputers. The methods and tools described in this paper include: stabilized finite element formulations; formulations for flows with moving boundaries and interfaces; mesh update methods; iterative solution techniques for large nonlinear equation systems; and parallel implementation of these methods. Our target is to be able to address effectively certain classes of flow simulation problems. These include: unsteady flows with interfaces; fluid–object interactions; fluid–structure interactions; airdrop systems; aerodynamics of complex shapes; and contaminant dispersion. © 1999 Elsevier Science Ltd. All rights reserved.

Keywords: CFD methods; TAFSM; Three-dimensional flow simulations

1. Introduction

We provide an overview of some of the advanced computational methods developed by the Team for Advanced Flow Simulation and Modeling (TAFSM) [http://www.mems.rice.edu/TAFSM/]. We also provide many examples of three dimensional flow simulations carried out by using these methods and advanced parallel computing platforms. Several components of the advanced methods and tools developed for flow simulation are described in this paper. These include: stabilized
finite element formulations for both compressible and incompressible flows; formulations for flows with moving boundaries and interfaces; mesh update methods for computation of moving boundaries and interfaces; iterative solution techniques for large nonlinear equation systems that need to be solved at every time step of a computation; and parallel implementation of all these methods. All methods developed are for flow problems involving complex geometries, and all software was developed and implemented on parallel platforms by the T★AFSM. In developing this set of methods and tools, our target is to be able to address effectively certain classes of flow simulation challenges. These targeted challenges include: unsteady flows with interfaces; fluid–object interactions; fluid–structure interactions; airdrop systems; aerodynamics of complex shapes; and contaminant dispersion.

Unsteady flows with interfaces can involve a free surface or an interface between two different liquids or a liquid and a gas. In this class of problems the location of the interface is also an unknown and must be determined together with the solution of the Navier–Stokes equations. This class of problems is a subset of a larger class flow problems with moving boundaries and interfaces. Fluid–object interactions too are subsets of flow problems with moving boundaries and interfaces. They involve interactions between the solid objects and the fluid they are moving in. It also involves interactions between the objects themselves, such as collisions and groupings. The flow is governed by the Navier–Stokes equations. The 3D dynamics of the objects is governed by the Newton’s laws. The fluid forces acting on these particles are calculated from the computed flow field. Fluid-structure interactions involve fluids with moving boundaries and unsteady interfaces between the fluid and the structure. This is somewhat similar to fluid–object interactions. However, now the “objects” are deformable, and these deformations need to be determined together with the solution of the Navier–Stokes equations.

Examples of airdrop systems are: aerodynamics of ram-air parachutes; interaction between a parachuter and an aircraft; an a parachuter/parachute system in the wake of an aircraft. These simulations involve fluid–object and fluid structure interactions. They also involve aerodynamics of complex shapes. Examples of contaminant dispersion are: how a contaminant introduced near a vehicle or a cluster of buildings spreads around that vehicle or those buildings; and how a contaminant spreads inside a building. These simulations involve solution of the Navier–Stokes equations with complex geometries. After the flow field is determined, using the velocity field computed, a time-dependent advection–diffusion equation is solved to compute the time-evolution of the passive contaminant.

All simulations, except those for testing a new method, are carried out in 3D. Furthermore, all computations are performed on parallel computing platforms: CRAY T3E and T3D, Thinking Machines CM-5, and multi-processor SGI POWER CHALLENGE and ONYX.

In Section 2 we review the governing equations used in the computations. An overview of the stabilized finite element formulations and methods for computation of moving boundaries and interfaces is provided in Section 3. The deformable-spatial-domain/stabilized space-time (DSD/SST) formulation is reviewed in Section 4. The mesh update methods for the DSD/SST formulation are described in Section 5.
Section 6 provides an overview of the iterative solution methods and parallel implementations. In Section 7 we report several examples of flow simulations, and end the paper with concluding remarks in Section 8.

2. Governing equations

The governing equations used in the computations are the time-dependent Navier–Stokes equations of compressible and incompressible flows. According to the notation used here, $\Omega_i$ and $(0, T)$ will denote the space and time domains, where $\Gamma_i$ is the boundary of $\Omega_i$. In general, the spatial domain may change with respect to time, and the subscript $t$ indicates such time dependence. This might happen in computation of flows with moving boundaries and interfaces. In fact, this will be the case if in the formulation the spatial domain is defined to be the part of the space occupied by the fluid(s). The symbols $\rho(x, t)$, $u(x, t)$, $p(x, t)$ and $e(x, t)$ represent the density, velocity, pressure and the total energy, respectively. The external forces (e.g., the gravity) are represented by $f(x, t)$.

2.1. Compressible flows

The Navier–Stokes equations of compressible flows can be written in the following vector form:

$$\frac{\partial U}{\partial t} + \frac{\partial F_i}{\partial x_i} - \frac{\partial E_i}{\partial x_i} = 0 \quad \text{on } \Omega_i \quad \forall t \in (0, T),$$

where $U = (\rho, \rho u_1, \rho u_2, \rho u_3, \rho e)$ is the vector of conservation variables, and $F_i$ and $E_i$ are, respectively, the Euler and viscous flux vectors defined as

$$F_i = \begin{pmatrix} u_i \rho \\ u_i \rho u_1 + \delta_{i1} p \\ u_i \rho u_2 + \delta_{i2} p \\ u_i \rho u_3 + \delta_{i3} p \\ u_i (\rho e + p) \end{pmatrix},$$

$$E_i = \begin{pmatrix} 0 \\ [T]_{i1} \\ [T]_{i2} \\ [T]_{i3} \\ -q_i + [T]_{ia} u_a \end{pmatrix},$$

Here $[T]_{ij}$ are the components of the Newtonian viscous stress tensor:

$$T = 2\mu \varepsilon(u),$$
where $\mu$ is the dynamic viscosity and $\varepsilon$ is the strain rate tensor, and $q_i$ are the components of the heat flux vector. The equation of state is modeled with the ideal gas assumption.

Eq. (1) can also be written in the following form:

$$\frac{\partial \mathbf{U}}{\partial t} + A_i \frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial}{\partial x_i} \left( K_{ij} \frac{\partial U}{\partial x_j} \right) = 0 \quad \text{on } \Omega_t \quad \forall t \in (0, T),$$

(5)

where

$$A_i = \frac{\partial F_i}{\partial \mathbf{U}}.$$  

(6)

$$K_{ij} \frac{\partial U}{\partial x_j} = E_i.$$  

(7)

Appropriate sets of boundary and initial conditions are assumed to accompany Eq. (5).

2.2. Incompressible flows

The Navier–Stokes equations of incompressible flows can be written in the following vector form:

$$\rho \left( \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} - \mathbf{f} \right) - \nabla \cdot \sigma = 0 \quad \text{on } \Omega_t \quad \forall t \in (0, T),$$

(8)

$$\nabla \cdot \mathbf{u} = 0 \quad \text{on } \Omega_t \quad \forall t \in (0, T),$$

(9)

where $\rho$ is assumed to be constant, and

$$\sigma = -\rho I + T.$$  

(10)

This equation set is completed with an appropriate set of boundary conditions and an initial condition consisting of a divergence-free velocity field specified over the entire domain:

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0, \quad \nabla \cdot \mathbf{u}_0 = 0 \quad \text{on } \Omega_0.$$  

(11)

For problems not involving moving boundaries and interfaces, the spatial domain does not need to change with respect to time. Therefore, we can drop the subscript $t$ from $\Omega_t$ and $\Gamma_t$. Even for computation of moving boundaries and interfaces, in some formulations, the spatial domain is not defined to be the part of the space occupied by the fluid(s). Consequently, the spatial domain does not need to change with respect to time. For example, we can select a fixed spatial domain, and model the fluid–fluid interfaces by assuming that the domain is occupied by two immiscible fluids, A and B, with densities $\rho_A$ and $\rho_B$ and viscosities $\mu_A$ and $\mu_B$. An interface function $\phi$ serves as a marker identifying Fluid A and B with the definition $\phi = \{1$ for Fluid A and 0 for
Fluid B. The interface between the two fluids is approximated to be at $\phi = 0.5$. In this context, $\rho$ and $\mu$ are defined as

$$\rho = \phi \rho_A + (1 - \phi)\rho_B, \quad (12)$$

$$\mu = \phi \mu_A + (1 - \phi)\mu_B. \quad (13)$$

The evolution of the interface function $\phi$ is governed by a time-dependent advection equation

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0 \quad \text{on } \Omega \quad \forall t \in (0, T). \quad (14)$$

3. Finite element formulations and methods for moving boundaries and interfaces

The finite element formulations reviewed here are either semi-discrete formulations to solve problems with fixed spatial domains, or space–time formulations [1–3] when we need to solve problems with changing spatial domains, such as those encountered in flows with moving boundaries and interfaces.

In the space–time method, the finite element formulation of the problem is written over its associated space–time domain. This automatically takes into account the motion of the boundaries and interfaces. At each time step of a computation, the locations of the boundaries and interfaces are determined as part of the overall solution. The space–time formulation is an interface-tracking method. It requires meshes which “track” the interfaces. Formulations for moving boundaries and interfaces can also be based on fixed spatial domains. In that case, an interface function, such as the one described in Section 2, needs to be computed to “capture” the interface. The interface is captured within the resolution of the finite element mesh covering the area where the interface is.

Both the interface-tracking and capturing methods described in this paper are based on stabilization techniques such as the streamline-upwind/Petrov–Galerkin (SUPG) [4–8], pressure-stabilizing/Petrov–Galerkin (PSPG) [9,10], and Galerkin/least-squares (GLS) [11,12,1,2] formulations. The SUPG method is one of the earliest and most popular stabilized methods. It was introduced in Ref. [4]. The SUPG formulation for compressible flows, on the other hand, was first introduced, in the context of conservation variables, in 1983 in Ref. [5]. Following that, several researches designed and studied SUPG-like methods for compressible flows. For example, the Taylor–Galerkin method, which appeared in the literature in 1984, is very similar, and under some conditions identical, to one of the SUPG methods introduced in Ref. [5]. The PSPG formulation was introduced in Ref. [9] and assures numerical stability when we use equal-order interpolation functions for velocity and pressure and other unknowns. The stabilization techniques prevent numerical oscillations and instabilities when the flow involves high Reynolds and/or Mach numbers and strong shocks and boundary layers. The stabilized finite element formulations developed also allow the use of equal-order interpolation functions for velocity and
pressure and other unknowns. In all cases, the stabilization is accomplished without introducing excessive numerical dissipation.

4. DSD/SST formulation

The DSD/SST formulation begins with the weak form of the governing equations being written over the associated space–time domain of the problem. This is done by dividing the domain into a sequence of \( N \) space–time slabs \( Q_n \), where \( Q_n \) is the slice of the space–time domain between the time levels \( t_n \) and \( t_{n+1} \). The integrations involved in the weak form are then performed over \( Q_n \). The finite element interpolation functions used are continuous in space but discontinuous across time levels. To reflect this situation, we use the notation \((\cdot)_n^–\) and \((\cdot)_n^+\) to denote the function values at \( t_n \) as approached from below and above, respectively. Each space–time slab \( Q_n \) is decomposed into space–time elements \( Q_{en} \), where \( e = 1,2,\ldots,(n_{el})_n \). The subscript \( n \) used with \( n_{el} \) is to account for the general case in which the number of space–time elements may change from one space–time slab to other. In our computations, we use first-order polynomials as interpolation functions.

4.1. DSD/SST formulation for compressible flows

In the DSD/SST formulation of compressible flows, for each slab \( Q_n \), we first define appropriate finite-dimensional space–time function spaces \( \mathcal{S}_n^h \) and \( \mathcal{V}_n^h \) corresponding to the trial solutions and weighting functions, respectively. The superscript \( h \) implies that these are finite-dimensional function spaces. The subscript \( n \) implies that corresponding to different space–time slabs we can have different spatial discretizations. The stabilized space–time formulation of Eq. (5) can then be written as follows: given \((U_n^h)\), find \( U_n^h \in \mathcal{S}_n^h \) such that \( \forall W_n^h \in \mathcal{V}_n^h \):

\[
\int_{Q_n} W_n^h \left( \frac{\partial U_n^h}{\partial t} + A^h_i \frac{\partial U_n^h}{\partial x_i} \right) dQ + \int_{Q_n} \left( \frac{\partial W_n^h}{\partial x_i} \right) \cdot \left( K_{ij}^h \frac{\partial U_n^h}{\partial x_j} \right) dQ \\
+ \int_{\Omega_n} \left( W_n^h \right)_n^+ \cdot \left( (U_n^h)_n^+ - (U_n^h)_n^- \right) d\Omega \\
+ \sum_{e=1}^{(n_{el})_n} \int_{Q_e} \left( \tau (A_{ij}^h)^T \frac{\partial W_n^h}{\partial x_k} \right) \cdot \left[ \frac{\partial U_n^h}{\partial t} + A^h_i \frac{\partial U_n^h}{\partial x_i} - \frac{\partial}{\partial x_i} \left( K_{ij}^h \frac{\partial U_n^h}{\partial x_j} \right) \right] dQ \\
+ \sum_{e=1}^{(n_{el})_n} \int_{Q_e} \delta \left( \frac{\partial W_n^h}{\partial x_i} \right) \cdot \left( \frac{\partial U_n^h}{\partial x_i} \right) dQ = \int_{P_n} W_n^h \cdot H_n^h dP. \tag{15}
\]

Here \( H_n^h \) represents the Neumann-type boundary condition, \( P_n \) is the lateral boundary of the space–time slab, and \( \tau \) and \( \delta \) are the stabilization parameters.
The solution to Eq. (15) is obtained sequentially for all space–time slabs \( Q_0, Q_1, Q_2, \ldots, Q_{N-1} \), and the computations start with

\[
(U^h)_0^- = U_0^h,
\]

where \( U_0^h \) is the specified initial value of the vector \( U \).

In the formulation given by Eq. (15), the first three integrals, together with the right hand side, represent the time-discontinuous Galerkin formulation of Eq. (5). The third integral enforces, weakly, the continuity of the conservation variables in time. The first series of element-level integrals are the SUPG stabilization terms, and the second series are the shock-capturing terms added to the formulation. The details regarding the stabilization and the space–time formulation can be found in Ref. [3]. For problems not involving moving boundaries and interfaces, Eq. (15) can be reduced to a semi-discrete formulation by dropping the third integral, and by converting all space–time integrations to spatial integrations.

4.2. DSD/SST formulation for incompressible flows

In the DSD/SST formulation of incompressible flows, the trial function spaces will be denoted by \( (\mathcal{S}^h_{u})_n \) and \( (\mathcal{S}^h_{p})_n \) and the weighting function spaces by \( (\mathcal{V}^h_{u})_n \) and \( (\mathcal{V}^h_{p})_n \). The stabilized space–time formulation of Eqs. (8) and (9) can then be written as follows: given \( (u_h^k)^- \), find \( u_h^k \in (\mathcal{S}^h_{u})_n \) and \( p_h^k \in (\mathcal{S}^h_{p})_n \) such that \( \forall u_h^k \in (\mathcal{V}^h_{u})_n \) and \( \forall q_h^k \in (\mathcal{V}^h_{p})_n \):

\[
\int_{Q_n} w_h^k \cdot \rho \left( \frac{\partial u_h^k}{\partial t} + u_h^k \cdot \nabla u_h^k - f \right) \, dQ + \int_{\Omega_n} \varepsilon(w_h^k) : \sigma(p_h^k, u_h^k) \, dQ \\
+ \int_{Q_n} q_h^k \nabla \cdot u_h^k \, dQ + \int_{\Omega_n} (w_h^k)^+ \cdot \rho ((u_h^k)^+ - (u_h^k)^-) \, d\Omega \\
+ \sum_{e=1}^{(n_e)} \tau_{\text{MOM}} \int_{Q_e} \frac{1}{\rho} \left[ \rho \left( \frac{\partial w_h^k}{\partial t} + u_h^k \cdot \nabla w_h^k \right) - \nabla \cdot \sigma(q_h^k, w_h^k) \right] \\
\cdot \left[ \rho \left( \frac{\partial u_h^k}{\partial t} + u_h^k \cdot \nabla u_h^k - f \right) - \nabla \cdot \sigma(p_h^k, u_h^k) \right] \, dQ \\
+ \sum_{e=1}^{(n_e)} \tau_{\text{CONT}} \int_{Q_e} \nabla \cdot w_h^k \rho \nabla \cdot u_h^k \, dQ = \int_{P_n} w_h^k \cdot h^k \, dP. \tag{17}
\]

Here \( h^k \) represents the Neumann-type boundary condition associated with the momentum equation, \( P_n \) is lateral boundary of the space–time slab, and \( \tau_{\text{MOM}} \) and \( \tau_{\text{CONT}} \) are the stabilization parameters.

The solution to Eq. (17) is obtained sequentially for all space–time slabs \( Q_0, Q_1, Q_2, \ldots, Q_{N-1} \), and the computations start with

\[
(u_h^k)^- = u_0^h. \tag{18}
\]
In the formulation given by Eq. (17), the first four integrals, together with the right-handside, represent the time-discontinuous Galerkin formulation of Eqs. (8) and (9). The fourth integral enforces, weakly, the continuity of the velocity field in time. The two series of element-level integrals in the formulation are the least-squares stabilization terms.

The reader can refer to Tezduyar et al. [1,2] and Behr and Tezduyar [13] for further details regarding the space–time formulation for incompressible flows, including definitions of the stabilization parameters. For problems not involving moving boundaries and interfaces, and for interface-capturing techniques, Eq. (17) can be reduced to a semi-discrete formulation by dropping the fourth integral and the term $\partial \mathbf{w}^h / \partial t$, and by converting all space–time integrations to spatial integrations.

5. Mesh update methods for DSD/SST formulations

Most real-world problems require simulations with rather complex geometries. Our approach to mesh generation depends on the complexity of the spatial domain and other factors, such as how frequently we need to generate a new mesh during the simulation.

A general-purpose approach is to use an automatic mesh generator. The automatic, 3D mesh generator we have developed [14] is quite efficient. It can generate 1.2 million tetrahedral elements in 6 min on an R4400 processor of an SGI workstation using only 90 Mbytes of memory (see Ref. [15]). Our automatic mesh generator has the capability to provide structured layers of elements around solid objects with reasonable geometric complexity. With this capability, we can fully control the mesh resolution near solid objects. This feature can be used for more accurate representation of the boundary layers.

Automatic mesh generation might become an overwhelming cost especially when the number of elements become very large or when frequency of remeshing (i.e., generating a new set of nodes and elements) is high. Special-purpose mesh generators designed for specific problems sometimes involve a high initial design cost, but minimal mesh generation cost. These can be used very effectively for simulations using the same class of problem geometries. For example, see Ref. [16].

In the DSD/SST formulation, as the spatial domain changes in time, the mesh needs to be updated to accommodate this change. The only rule the mesh motion needs to follow is that at the interface the normal velocity of the mesh has to match the normal velocity of the fluid. Beyond that, the mesh can be updated in any way desired, with the main objective being to reduce the frequency of remeshing. In 3D simulations, if remeshing requires calling an automatic mesh generator, the cost of automatic mesh generation becomes a major reason for trying to reduce the frequency of remeshing.

In some cases where the changes in the shape of the computational domain allow it, a special-purpose mesh moving method can be used in conjunction with a special-purpose mesh generator. In such cases, simulations can be carried out without calling an automatic mesh generator and without solving any additional equations to...
determine the motion of the mesh. An earlier example, 3D parallel computation of sloshing in a vertically vibrating container, can be found in Ref. [13].

In general, we can use an automatic mesh moving scheme [17] to move the nodal points. This is accomplished by solving modified equations of linear elasticity to determine the motion of the internal nodes. The boundary conditions for these equations are specified in such a way that they match the normal velocity of the fluid at the interface. Structured layers of elements generated around solid objects move “glued” to these objects. Because their deformations are not governed by the equations of elasticity, the motion of the nodes belonging those layers do not add to the cost of automatic mesh moving. Also, the user continues to have full control of the mesh resolution in these layers. For examples of automatic mesh moving combined with structured layers of elements, see Refs. [14,18]

6. Iterative solution methods and parallel implementations

Full discretizations of the finite element formulations described in earlier sections lead to coupled, nonlinear equation systems. These systems need to be solved at every step of the simulations. We might be using a space–time formulation with moving meshes or a semi-discrete formulation with non-moving meshes. In either case, we can represent the equation system as follows:

\[ N(d_{n+1}) = F. \]  (19)

Here \( d_{n+1} \) is the vector of nodal unknowns. In a semi-discrete formulation, this vector contains the unknowns associated with marching from time level \( n \) to \( n+1 \). In a space–time formulation, it contains the unknowns associated with the finite element formulation written for the space–time slab \( Q_n \). If the various time-marching formulations described in this chapter are being used for solving a steady-state flow problem, then Eq. (19) still needs to be solved at every time step. However, in such cases we call the time steps the “pseudo-time steps”. In most of our simulations the number unknowns in Eq. (19) is around 1–10 millions. In some of our computations we exceed 100 million equations.

We solve Eq. (19) with the Newton–Raphson method:

\[ \frac{\partial N}{\partial d} \bigg|_{d_{n+1}} (\Delta d_{n+1}^k) = F - N(d_{n+1}^k), \]  (20)

where \( k \) is the step counter for the Newton–Raphson sequence, and \( \Delta d_{n+1}^k \) is the increment computed for \( d_{n+1}^k \). The linear equation system represented by Eq. (20) needs to be solved at every step of the Newton–Raphson sequence. We can represent Eq. (20) as

\[ Ax = b. \]  (21)

In very large-scale simulations, this equation system would be too large to solve with a direct method. Instead, we solve it iteratively. At each iteration, we first compute the
residual of this system,

\[ r = b - Ax, \]  

(22)

and then compute a candidate correction to \( x \) as given by the expression

\[ \Delta y = P^{-1} r, \]  

(23)

where \( P \), the preconditioning matrix, is an approximation to \( A \). \( P \) has to be simple enough to form and invert efficiently. However, it also has to be sophisticated enough to yield a satisfactory convergence rate. How to update the solution vector \( x \) by using \( \Delta y \) is an issue by itself. Several update methods are available and we use the GMRES [19] method. We focused our iterative methods research on computing the residual \( r \) efficiently and selecting a good preconditioner \( P \). In doing this, we needed to keep in mind that all these need to be implemented on parallel computing platforms. The parallel computing methods we designed for the residual computations include those which are element-matrix-based [20], element-vector-based [20], and sparse-matrix-based [21].

In preconditioning design, we developed some sophisticated preconditioners such as the clustered-element-by-element (CEBE) preconditioner [22] and the mixed CEBE and cluster companion (CC) preconditioner [23]. We have implemented, with quite satisfactory results, the CEBE preconditioner in conjunction with an ILU approximation [21]. However, our typical computations are based on diagonal and nodal-block-diagonal preconditioners. These are very simple preconditioners, but are also very simple to implement on parallel platforms.

A small fraction of our parallel computations are based on shared-memory platforms, such as a 12-processor SGI POWER CHALLENGE and a 20-processor SGIONYX. Most of our simulations are carried out on distributed-memory platforms. In this category, our earlier computations were based on data-parallel paradigm on a 512-node Thinking Machines CM-5. The majority of our computations nowadays is based on message-passing paradigm (with MPI and PVM) on a 256-node CRAY T3E-1200. More on our parallel implementations can be found in Ref. [20].

7. Examples of flow simulations

In this section we present several examples of flow simulations. These simulations were carried out by using the advanced methods and computing platforms described in the earlier sections. Each case is described briefly. References are given to our earlier publications which provide more information. The purpose here is not to provide detailed physical analysis of each problem. The purpose is to demonstrate how advanced flow simulation methods, together with advanced computing platforms, enable us to solve a variety of challenging problems. Only in the first problem the governing equations are the Navier–Stokes equations of compressible flows. In the remaining problems, we use the Navier–Stokes equations of incompressible flows.

*Flow around two high-speed trains in a tunnel.* Two high-speed trains pass each other in a tunnel. Each train has a speed of 100 m/s. The Reynolds number based on the
train length is around 67 million. The mesh consists of 101888 hexahedral elements and 230982 space–time nodes, and leads to 900274 coupled, nonlinear equations that need to be solved at every time step. A Smagorinsky LES turbulence model is used. For the DSD/SST formulation, to accommodate the motion of the trains, we developed a special mesh moving and remeshing scheme. This adds almost no cost to the computations. This computation was performed on the Thinking Machines CM-5. Fig. 1 shows the trains and pressure distribution at different instants. For more on this simulation see Ref. [24].

Contaminant dispersion in a model subway station. The subway station has two entrances on each side and four vents located on the upper surface. This simulation is carried out on the CRAY T3D in two stages. First, the Navier–Stokes equations are solved to obtain the flow velocity. This velocity field is used in the second stage in the time-dependent contaminant advection–diffusion equation to obtain the concentration of the contaminant. The contaminant is released from a point source with constant strength. The unstructured mesh used consists of 187612 nodes and 1116992 tetrahedral elements. The steady-state solution of the flow equations is obtained by solving over 0.65 million coupled, nonlinear equations at every pseudo-time step. For the contaminant dispersion, at every time step, we solve a linear system with more
than 0.15 million equations. Fig. 2 shows the mesh and the contaminant concentration at an instant. For more on this simulation see Ref. [24].

**Flare maneuver of a large ram-air parachute.** This simulation was carried out on the T3D. The Reynolds number is about 10 million. The initial condition consists of the steady glide configuration of an unconstrained parachute with no flap deflection. The time for the flare maneuver and total flap deflection is obtained from test data. The parachute is treated as a solid body with changing shape. The shape of the parachute

![Image](image-url)
during the maneuver is interpolated from the initial and final flap configurations. A special mesh generator/mover was developed. With this the flare maneuver is simulated without any remesh. The DSD/SST formulation leads to 2258496 coupled, nonlinear equations that need to be solved at every time step. Fig. 3 shows the pressure distribution on the parachute surface during three instants of the flare maneuver. For more on this simulation see Ref. [24].

Dynamics of a parachuter jumping from a plane. This simulation was carried out on the T3D. The aircraft is traveling at 130 knots. The parachuter is allowed to translate and rotate freely governed by Newton’s laws of motion. We use the DSD/SST formulation in combination with our automatic mesh moving algorithm and remeshing as needed. The mesh has approximately 880000 tetrahedral elements for half of the domain (since the geometry is symmetric). A Smagorinsky turbulence model is used. The initial condition is the steady-state flow past the plane with the parachuter located within the open side door of the plane (a portion of the interior of the plane is also modeled). Fig. 4 shows the pressure distribution on the aircraft and also shows at an instant the pressure distribution on the parachuter and the streamlines. For more on this simulation see Ref. [24].

Fluid–object interactions with 1000 spheres falling in a liquid-filled tube. This simulation was carried out, in multi-platform computing setting, on the CM-5 and a SGI
Fig. 4. Dynamics of a parachuter jumping from a plane. This computation was performed on the CRAY T3D. Pictures show the pressure distribution on the cargo aircraft and also show at an instant the pressure distribution on the parachuter and the streamlines.

ONYX. While mesh partitioning, flow computations, and mesh movements are performed on the 512-node CM-5, mesh generation and projection is accomplished on a 2-processor (MIPS R10K) ONYX2. The spheres, in addition to interacting with the fluid, interact and collide with each other and with the tube wall. The average Reynolds number is around 8. The DSD/SST formulation is used in the simulation. The mesh is updated using our automatic mesh moving method with remeshing as needed. The mesh size is approximately 2.5 million tetrahedral elements, resulting in about 5.5 million coupled, nonlinear equations to be solved every time step. The number of time steps is around 1100 in the simulation. Fig. 5 shows the distribution of
the spheres at four instants during the simulation. The first picture shows the initial distribution. The colors are for identifying the individual spheres.

Flow past the spillway of the Olmsted Dam. This simulation was carried out on the CM-5. The model represents a 48 feet-wide section of the navigation pass crest and stilling basin. It includes a long upstream channel, the spillway crest, and a set of underwater obstacles designed to dissipate the flow energy. The DSD/SST formulation is used. The mesh is updated using our automatic mesh moving method. The mesh consists of 139352 space–time nodes and 396682 tetrahedral space–time elements. Fig. 6 shows the water pressure and streamlines, and the steady shape of the
Fig. 6. Flow past the spillway of the Olmsted Dam. This computation was performed on the CM-5. Picture shows the water pressure and streamlines, and the steady shape of the free surface achieved in the final stages of the simulation. For more on this simulation see Ref. [24].

Free-surface flow past a circular cylinder. This simulation was carried out on the CM-5. The Reynolds number based on upstream velocity is 10 million. The upstream Froude number is 0.564. The mesh consists of 230480 prism-based space–time elements and 129382 nodes. The DSD/SST formulation is used with an algebraic mesh update method. The free-surface height is governed by an advection equation and solved with a stabilized formulation. Fig. 7 shows, at an instant, the cylinder together with the free-surface color-coded with the velocity magnitude.

Airflow past an automobile. This simulation was carried out on the CM-5. The automobile is traveling at 55 miles/h. It is modeled after a Saturn SL2. The Reynolds number is 6.9 million. The mesh contains 448695 nodes and 281518 elements. The computed drag coefficient under wind-tunnel conditions is 0.35. The published drag coefficient for a Saturn SL2 is 0.34. Fig. 8 shows the pressure distribution on the surface of the car and the streamlines. For more on this simulation see Ref. [24].

Sloshing in a tanker driving over a bump. This simulation was carried out on the CRAY T3E. We use an interface-capturing technique. The tanker, moving at 10 m/s, drives over a bump 30 cm-high. The suspension system of the tanker absorbs the initial displacements due to the bump and transfers the generated forces to the tanker. The 3D rigid-body dynamics equations are coupled to the finite element formulation of the flow problem and solved simultaneously for the motion of the tanker. The fluid dynamics equations are written in a non-inertial frame. The mesh has 343560 hexahedral elements and 357911 nodes. At each time step, a coupled system of
Fig. 7. Free-surface flow past a circular cylinder. This computation was performed on the CM-5. Picture shows, at an instant, the cylinder together with the free-surface color-coded with the velocity magnitude. The bow wave in front of the cylinder, the hydraulic jump behind, and the V-shaped waves in the wake are characteristic of this flow problem.

Fig. 8. Airflow past an automobile. This computation was performed on the CM-5. Pictures show the pressure distribution on the surface of the car and the streamlines.
non-linear equations with 1704661 unknowns is solved. Fig. 9 shows, at different instants, the motion of the tanker, sloshing and pressure distribution. For more on this simulation see Ref. [25].

8. Concluding remarks

We provided an overview of some of the CFD methods developed by the T★AFSM. We reported many examples of three-dimensional flow simulations carried out with these CFD methods and advanced parallel supercomputers. The methods described include: stabilized finite element formulations for both compressible and incompressible flows; formulations for moving boundaries and interfaces; mesh update methods for computation of moving boundaries and interfaces; iterative solution techniques for large non-linear equation systems that need to be solved at every time step; and parallel implementation of these methods. All methods developed are for flow problems involving complex geometries, and all software was developed and implemented on parallel platforms by the T★AFSM. We demonstrated the power and potential of these methods by applying them to a number of complex flow problems.
problems. These examples come from the following classes of problems: unsteady flows with interfaces; fluid–object interactions; airdrop systems; aerodynamics of complex shapes; and contaminant dispersion.

9. Acknowledgements

This work was sponsored by ARO, by NASA, and by the Army HPC Research Center under the auspices of the Department of the Army, ARL cooperative agreement number DAAH04-95-2-0003 and contract number DAAH04-95-C-0008. The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References


