ELSEVIER

Comput. Methods Appl. Mech. Engrg. 184 (2000) 449–466

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Automotive design applications of fluid flow simulation on parallel computing platforms

Steven M. Rifai *, Jeffrey C. Buell, Zdeněk Johan, Thomas J.R. Hughes

Centric Engineering Systems, Inc., 624 East Evelyn Avenue, Sunnyvale, CA 94086, USA

Abstract

This article describes the use of fluid flow simulation on parallel computing platforms to solve design problems in the automotive industry. The fluid flow formulation in the Spectrum™ solver, which is used for these simulations, is described. The finite element treatment of fluid flow in this work is based on the Galerkin-Least-Squares (GLS) method with discontinuity capturing operators. The Arbitrary-Lagrangian–Eulerian (ALE) method is utilized to account for deformable fluid domains. Automatically generated tetrahedral grids are used to ease and expedite the analysis process. The multiphysics architecture used in this work lends itself naturally to high-performance parallel computing. By taking advantage of automatic mesh generation and parallel computing, dramatic reduction in turnaround time for flow analysis is achieved. Several applications are presented which demonstrate the utility and accuracy of finite element solutions in automotive engineering problems and highlight the scalability of the software.

1. Motivation

The competitive nature of the automotive industry has driven manufacturers and suppliers to push the boundaries of simulation methods for design applications [1]. This evolution is driven by the need to shorten design cycles, reduce cost, meet increasingly stringent government regulations, improve quality and safety, and reduce environmental impact. An increased need for accurate product and component simulation has pressed analysts for simulations of unprecedented scale and complexity. The application of fluid flow simulation in automotive design analysis has consequently gained a considerable following.

In this article, we describe the fluid flow formulation in the Spectrum™ solver and some of its applications in the automotive industry. Fig. 1 depicts three examples of such applications:

- Steady flow over a radiator cooling fan, simulated to predict the torque required to turn the fan at a given operating condition.
- Unsteady flow in an exhaust manifold of a six cylinder engine during its firing sequence, simulated to evaluate the pressure drop across the manifold, the heat transfer and resulting thermal stress in the solid casing.
- System-level simulation of flow exterior to and under the hood of vehicles with operating engine and powertrain components, simulated to predict the cooling efficacy of different product configurations.

The increased demand on the scale and complexity of simulations in the automotive industry has created a need for substantial computational resources. While single processor computers improve every year, the quantum increase in computational needs requires a quantum increase in computational power. The hardware industry is answering this increased need with affordable high-performance parallel architectures

* Corresponding author.

0045-7825/00/$ - see front matter
PII: S0045-7825(99)00239-X

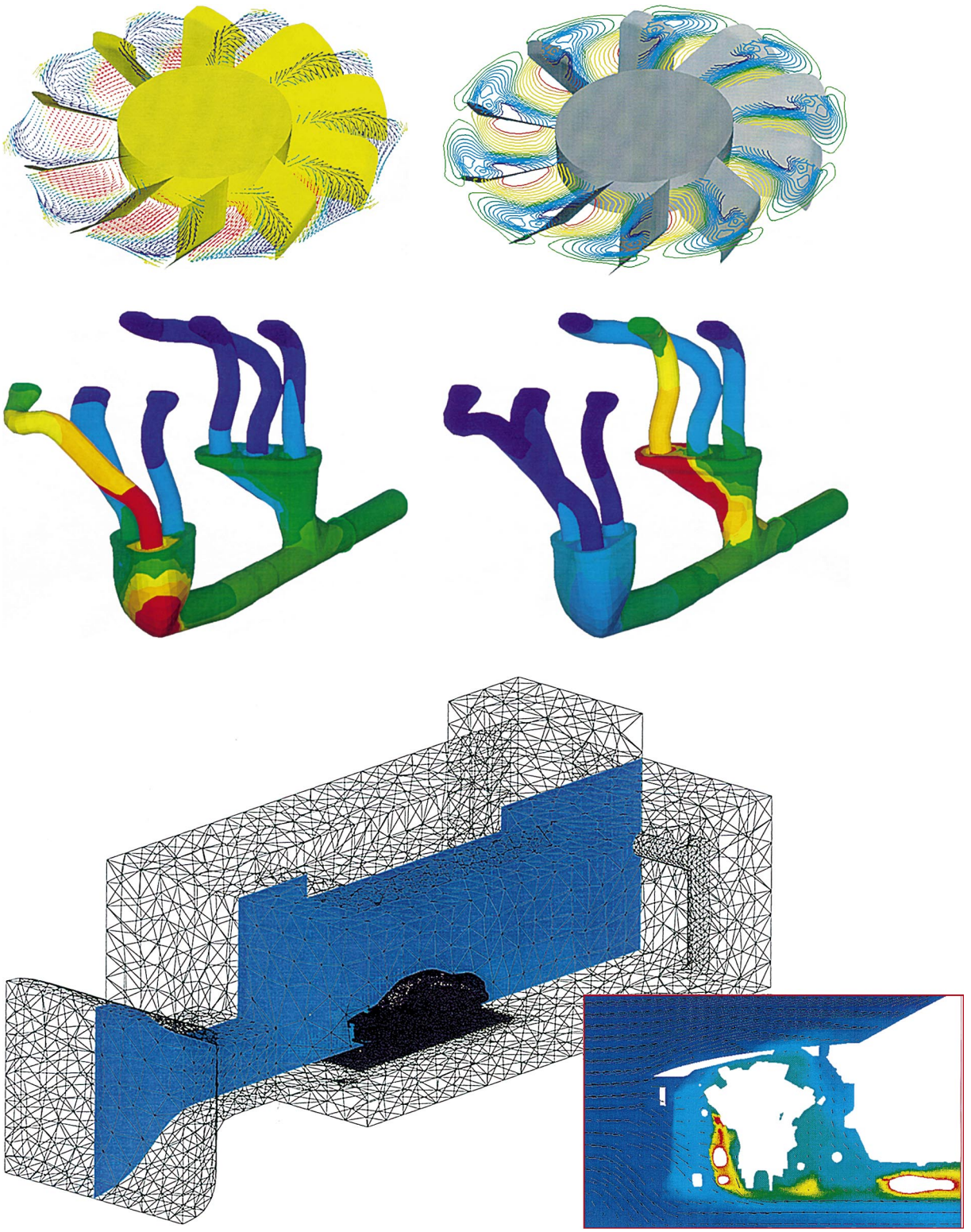

Fig. 1. Examples of automotive flow simulation applications: steady flow over a cooling fan, unsteady pulsatile flow within an exhaust manifold, and system level simulation of exterior flow and underhood cooling.

which are built on commodity parts and compatible with workstation systems. These architectures offer the level of computational performance and increased memory capacity needed for large-scale computing. In the context of these architectures, two parallel programming models have emerged as popular and supported approaches: data-parallel and Same-Program Multiple Data (SPMD). Because the underlying architecture is different than traditional uniprocessor or vector computers, the increased performance in both models is most fully available to codes designed to take advantage of the new architectures.

Although parallel processing has made a significant impact on the adoption of flow simulation by automotive analysts, it is the combination of scalable software and hardware with automatic mesh generation which makes simulation effective in design applications. The simulation process, as schematically illustrated in Fig. 2, begins with a definition of the computational geometry, followed by mesh generation, solution and visualization. The mesh generation process can be a tedious, manual and time consuming part of the flow simulation. For complex geometries, such as the vehicle interior and engine water jacket illustrated in Fig. 3, manual grid generation can occupy weeks or months of an analyst's time. This renders the entire simulation process untenable for use in the industry's design cycles. However, the availability of automatic mesh generation [17] combined with scalable software on parallel processing platforms automates and streamlines the two major bottlenecks in the analysis.

By taking advantage of commercially available automatic mesh generation software and parallel processing hardware, industry users have successfully applied the Spectrum solver to simulate many automotive flow problems.

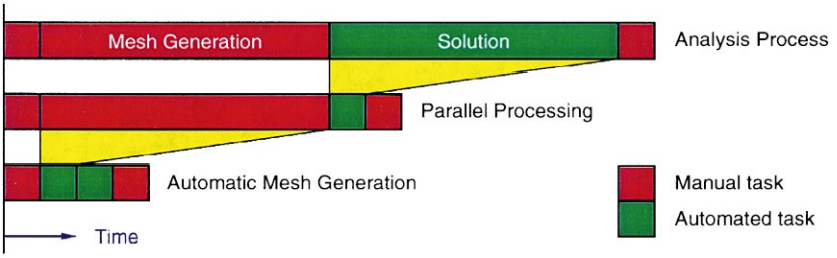

Fig. 2. Schematic illustration of the analysis process for finite element simulation. The combination of automatic mesh generation and parallel processing automates and streamlines the two major bottlenecks in this process.

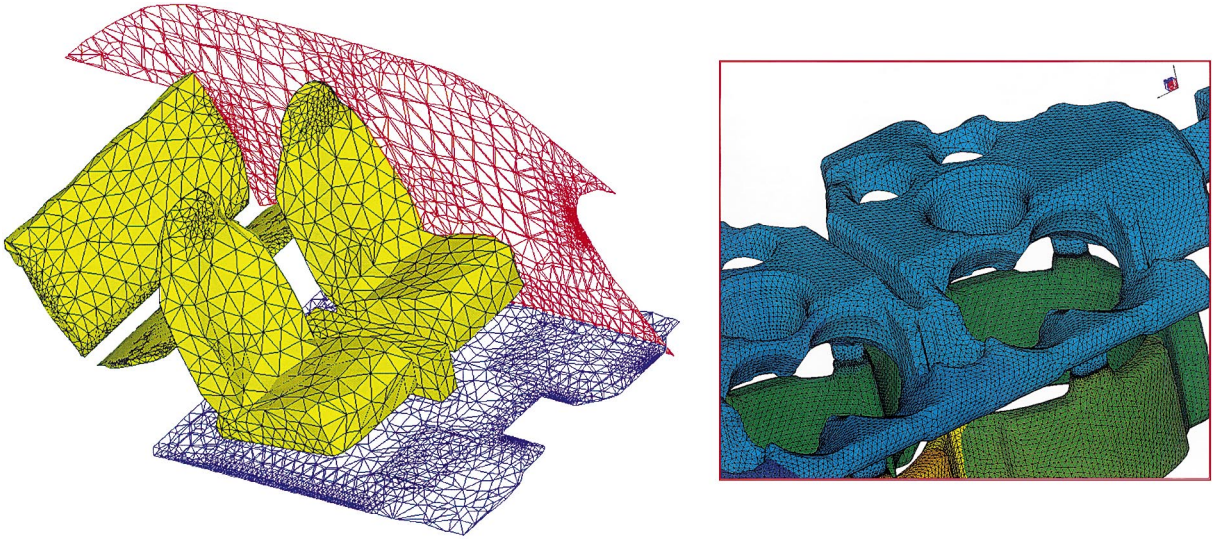

Fig. 3. Illustration of the complex geometries bounding computational flow domains in automotive applications – automatic mesh generation is a necessity.

2. Fluid flow modeling

The finite element treatment of fluid regions within our framework is based on the Galerkin-Least-Squares (GLS) method with discontinuity capturing operators [5–7]. Reynolds-averaged and large-eddy-simulation models are utilized for turbulence simulation [8]. In this treatment, the compressible flow formulation makes use of the *physical entropy variables*

$$\boldsymbol{V}^{\mathrm{c}} = \frac{1}{T^{\mathrm{c}}} \begin{pmatrix} \tilde{\mu} - |u|^2/2 \\ u_1 \\ u_2 \\ u_3 \\ -1 \end{pmatrix}^{\mathrm{c}}, \tag{1}$$

where the superscript c denotes compressible flow variables, T is the fluid temperature and u is the fluid velocity vector with components u_1, u_2, and u_3. The chemical potential $\tilde{\mu}$ is defined as

$$\tilde{\mu} = e + p/\rho - Ts, \tag{2}$$

with e, s, ρ and p being the specific internal energy, specific entropy, density and pressure, respectively.

With these variables, the fluid conservation laws are expressed in symmetric form which intrinsically expresses the mathematical and physical stability provided by the second law of thermodynamics. In turn, the finite element techniques employed herein inherit this fundamental stability and convergence proofs are available [9].

With the definitions above, the Navier–Stokes equation system can be expressed as follows: A solution $\boldsymbol{V}$ is sought for the symmetric convective–diffusive system

$$\tilde{\boldsymbol{A}}_0 \boldsymbol{V}_{,t} + \tilde{\boldsymbol{A}}_i \boldsymbol{V}_{,i} = (\tilde{\boldsymbol{K}}_{ij} \boldsymbol{V}_{,j})_{,i} + \tilde{\boldsymbol{F}}^{\mathrm{src}}, \tag{3}$$

where

$$\tilde{\boldsymbol{A}}_0 = \boldsymbol{U}_{,\boldsymbol{V}}, \tag{4}$$

$$\tilde{\boldsymbol{A}}_i = \boldsymbol{F}_{i,\boldsymbol{V}}^{\mathrm{conv}}, \tag{5}$$

$$\tilde{\boldsymbol{K}}_{ij} \boldsymbol{V}_{,j} = \boldsymbol{F}_i^{\mathrm{diff}}, \tag{6}$$

and

$$\tilde{\boldsymbol{F}}^{\mathrm{src}} = \boldsymbol{F}^{\mathrm{src}}(\boldsymbol{U}(\boldsymbol{V})). \tag{7}$$

Here $\boldsymbol{U}$ is the vector of conservation variables, $\boldsymbol{V}$ is the vector of entropy variables, and $\boldsymbol{U} = \boldsymbol{U}(\boldsymbol{V})$ is the appropriate transformation of variables leading to the symmetric system. The matrices $\tilde{\boldsymbol{A}}_0$, $\tilde{\boldsymbol{A}}_i$, and $\boldsymbol{K}_{ij}$ and the vector. $\tilde{\boldsymbol{F}}^{\mathrm{src}}$ are nonlinear functions of $\boldsymbol{V}$. For convenience, the matrix $\tilde{\boldsymbol{C}}$ is introduced such that the source vector may be written as $\tilde{\boldsymbol{F}}^{\mathrm{src}} \equiv -\tilde{\boldsymbol{C}}\boldsymbol{V}$.

Remark. *Entropy variables do not yield the most efficient form of the incompressible equations. In practice, the advantages of the entropy variable formulation are not significant without the presence of the shocks and discontinuities of compressible flows* [7]. *Hence, the state variables used in the incompressible flow formulation are defined as*

$$\boldsymbol{V}^{i} = \begin{pmatrix} p/\rho_0 \\ u_1 \\ u_2 \\ u_3 \\ T \end{pmatrix}^{i}, \tag{8}$$

where the superscript i denotes incompressible flow variables.

2.1. The Arbitrary-Lagrangian-Eulerian formulation

The analysis of fluid flow with moving boundaries is an increasingly important feature in automotive applications. For example, multiphysics problems often require the movement of the computational fluid domain in response to the deformation of the common solid region boundaries. Furthermore, the flow within many automotive components (e.g., engine cylinders, fuel injectors, etc.) is strongly dependent on movements of nearly rigid boundaries. The Arbitrary-Lagrangian–Eulerian (ALE) method is utilized to account for the deformations in fluid domains [10–12].

ALE boundary conditions ensure that the deforming mesh conforms to both the stationary and moving boundaries. Within the interior of the fluid domain, mesh movement is modeled by the equations of large deformation elasticity. In effect, this model computes the position of the interior nodes as if all nearest nodal neighbors were coupled by an elastic medium and sets the positions and velocities of the boundary nodes to exactly match the boundary motions. Note that this model is a purely mathematical construct; it is a method for updating the mesh in a way that has a reasonable chance of maintaining mesh integrity. Thus, terms such as Cauchy stresses, elastic moduli, and so forth do not have the usual physical interpretation within this context.

To take mesh movement into account, the convective or Euler flux in Eqs. (3) and (5) is replaced by the ALE convective flux. The ALE convective flux is related to the Euler flux by the equation

$$\boldsymbol{F}_i^{\mathrm{ALE}} = \boldsymbol{F}_i^{\mathrm{conv}} - u_i^{\mathrm{ALE}}\boldsymbol{U}, \tag{9}$$

where u^{ALE} is the velocity field of the mesh. The Euler Jacobians of Eq. (3) become

$$\tilde{\boldsymbol{A}}_i^{\mathrm{ALE}} = \tilde{\boldsymbol{A}}_i - u_i^{\mathrm{ALE}}\tilde{\boldsymbol{A}}_0, \tag{10}$$

where the $\tilde{\boldsymbol{A}}_0$ and $\tilde{\boldsymbol{A}}_i$ are defined in Eqs. (4) and (5), respectively. This substitution is equivalent to transforming the material derivatives of field variables using

$$\frac{\mathrm{D}w}{\mathrm{D}t} = w_{,t} + \left(u_i - u_i^{\mathrm{ALE}}\right)w_{,i}, \tag{11}$$

where w represents any of the field variables.

Note the following special cases:

$$\text{Eulerian}\quad \begin{aligned} &\tilde{\boldsymbol{A}}_i^{\mathrm{ALE}} = \tilde{\boldsymbol{A}}_i, \qquad \boldsymbol{u}^{\mathrm{ALE}} = \boldsymbol{0} \\ &\boldsymbol{F}_i^{\mathrm{ALE}} = \boldsymbol{F}_i^{\mathrm{conv}} \end{aligned} \tag{12}$$

$$\text{Lagrangian}\quad \begin{aligned} &\tilde{\boldsymbol{A}}_i^{\mathrm{ALE}} = \tilde{\boldsymbol{A}}_i - u_i\tilde{\boldsymbol{A}}_0, \qquad \boldsymbol{u}^{\mathrm{ALE}} = \boldsymbol{u} \\ &\boldsymbol{F}_i^{\mathrm{ALE}} = \boldsymbol{F}_i^{\mathrm{conv}} - u_i\boldsymbol{U} \end{aligned} \tag{13}$$

2.2. The Galerkin-Least-Squares formulation

The finite element weighted residual formulation is as follows: given the definition of the above vectors and matrices, find solution vector V^h such that for all admissible weighting functions, W^h, the vector V^h satisfies the variational equation

$$\begin{aligned} &\int_\Omega \left(\boldsymbol{W}^h\cdot\tilde{\boldsymbol{A}}_0\dot{\boldsymbol{V}}^h - \boldsymbol{W}^h_{,i}\cdot\boldsymbol{F}_i^{\mathrm{conv}}\left(\boldsymbol{V}^h\right) + \boldsymbol{W}^h_{,i}\cdot\tilde{\boldsymbol{K}}_{ij}\boldsymbol{V}^h_{,j} - \boldsymbol{W}^h\cdot\tilde{\boldsymbol{F}}^{\mathrm{src}}\left(\boldsymbol{V}^h\right)\right)\mathrm{d}\Omega \\ &\quad + \sum_{e=1}^{n_{\mathrm{elem}}}\int_\Omega^e \left(L\boldsymbol{W}^h + \tilde{\boldsymbol{C}}\boldsymbol{W}^h\right)\tau\left(L_t\boldsymbol{V}^h - \tilde{\boldsymbol{F}}^{\mathrm{src}}\right)\mathrm{d}\Omega + \sum_{e=1}^{n_{\mathrm{elem}}}\int_{\Omega^e} v\boldsymbol{W}^h_{,\xi_i}\cdot\tilde{\boldsymbol{A}}_0\boldsymbol{V}^h_{,\xi_i}\,\mathrm{d}\Omega \\ &\qquad = \int_\Gamma \boldsymbol{W}^h\cdot\left(-\boldsymbol{F}_i^{\mathrm{conv}}\left(\boldsymbol{V}^h\right) + \boldsymbol{F}_i^{\mathrm{diff}}\left(\boldsymbol{V}^h\right)\right)n_i\,\mathrm{d}\Gamma, \end{aligned} \tag{14}$$

where Ω is the computational domain with boundary Γ; Ω^e; is the domain of element e, $e=1,\ldots,n_{\mathrm{elem}}$; and n_{elem} is the total number of elements. The vectors V^h and W^h are the usual finite element functions; they are

piecewise polynomials within each element domain Ω^e and C^0 continuous over the entire domain Ω. The vector V^h is the time derivative of V^h; ξ is a generalized vector of local coordinates for each element domain Ω^e; and n is the outward normal to the boundary Γ.

In Eq. (14), the first integral on the left-hand side and the integral on the right-hand side constitute the usual Galerkin formulation as applied to the symmetric convective–diffusive system. The convective and diffusive fluxes, $\boldsymbol{F}_i^{\text{conv}}$ and $\boldsymbol{F}_i^{\text{diff}}$, are formulated in an integrated-by-parts form. This results in the conservation of fluxes under all quadrature rules. In addition, the boundary integrals (right-hand side of Eq. (14)) lead to a set of natural (or Neumann) boundary conditions.

In Eq. (14), the second integral on the left-hand side is the least-squares operator. The time dependent quasilinear differential operator for the symmetric convective–diffusive system is

$$L_t \boldsymbol{V}^h \equiv \tilde{\boldsymbol{A}}_0 \dot{\boldsymbol{V}}^h + L\boldsymbol{V}^h, \tag{15}$$

where

$$L\boldsymbol{V}^h \equiv \tilde{\boldsymbol{A}}_i \frac{\partial}{\partial x_i} \boldsymbol{V}^h - \frac{\partial}{\partial x_i} \tilde{\boldsymbol{K}}_{ij} \frac{\partial}{\partial x_j} \boldsymbol{V}^h. \tag{16}$$

The metric τ is the *least-squares matrix* and is a symmetric positive-semidefinite matrix of intrinsic time scales. The design of τ crucially influences the behavior of the numerical solutions; therefore, considerable care is required here.

In Eq. (14), the third integral on the left-hand side is the *discontinuity-capturing* operator. The scalar discontinuity-capturing factor ν has dimension of reciprocal time and is a function of the residual of the differential equation, $L_t \boldsymbol{V}^h - \tilde{\boldsymbol{F}}^{\text{src}}$. The operator satisfies three fundamental properties.

- It acts in the direction of the gradient to control oscillations.
- It is proportional to the residual $L_t \boldsymbol{V}^h - \tilde{\boldsymbol{F}}^{\text{src}}$ for consistency.
- It vanishes quickly in smooth regions of the solution to ensure accuracy.

These properties are achieved through the proper choice of ν.

Grid distortion is a common by-product of automatic mesh generators. The state-of-the-art in automatic mesh generation produces quality discretizations in most regions of computational domains. However, the presence of small and slender features in many applications results in significant element distortion to avoid prohibitively large discretizations. Furthermore, fluid flow applications with moving boundaries (e.g., Fig. 4) often suffer severe distortion of the original mesh.

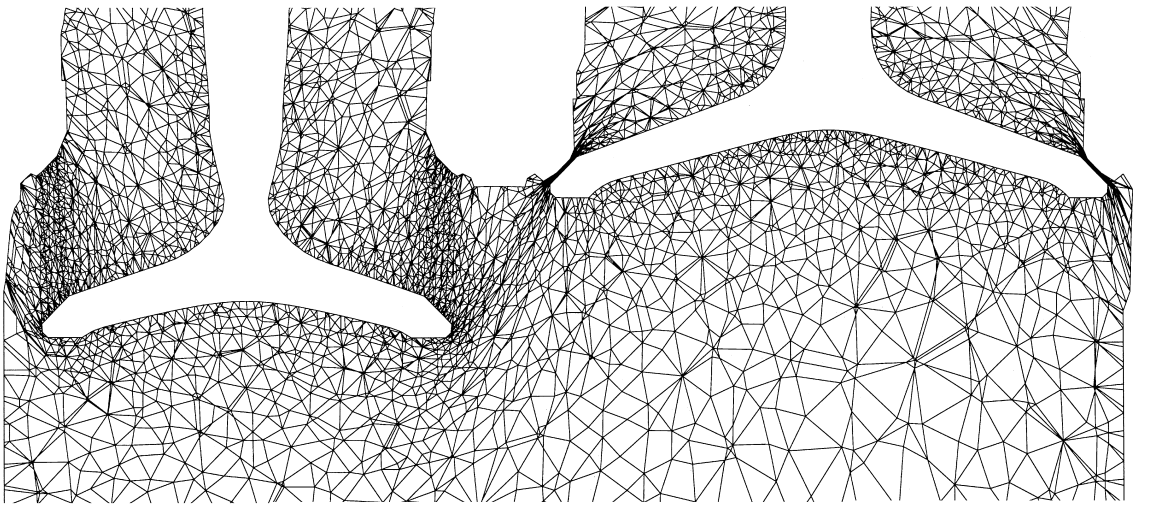

Fig. 4. Illustration of mesh distortion due to moving boundaries.

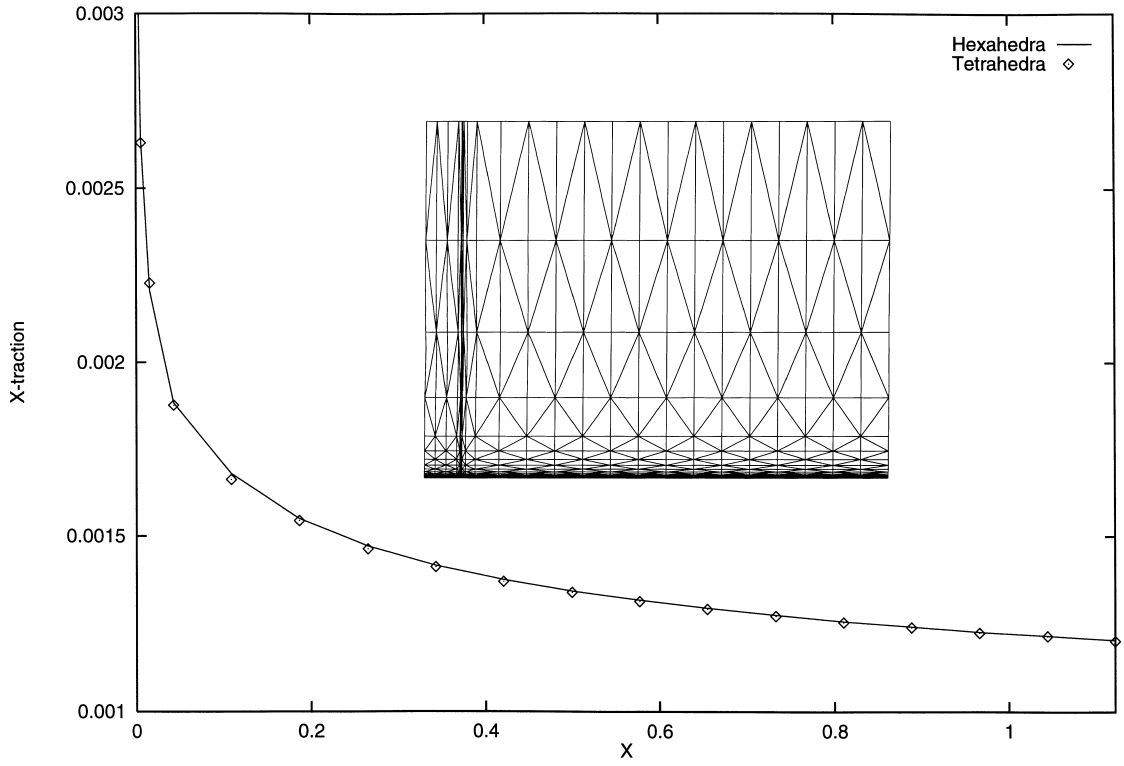

Fig. 5. Traction vs. location on the plate in the streamwise direction for the turbulent boundary layer problem. The mesh depicted in the inset contains highly stretched tetrahedra with aspect ratios near 20,000.

A key feature of the GLS method presented herein is its ability to provide quality solutions with highly distorted unstructured grids. We illustrate this feature with a simple problem: an incompressible turbulent boundary layer on a flat plate is discretized using a tetrahedral mesh (Fig. 5) as well as a hexahedral mesh. To properly resolve the boundary layer, the mesh is heavily stretched near the wall. This results in element aspect ratios near 20,000. The Reynolds number based on the plate length is 12 million. The tetrahedral mesh employs 4500 elements and uses the same nodes as the hexahedral mesh. The traction in the streamwise direction is shown in Fig. 5. Excellent agreement is found between the two mesh discretizations. Both solutions were validated with accepted correlations.

2.3. Turbulence

The Spectrum solver employs several turbulence models including large eddy simulation and Reynolds averaged models such as k–ε. For the simulations presented in this work, a slight variation on the Spalart–Allmaras (S–A) turbulence model is used [13]. This is a one-equation model that governs the kinematic viscosity $\tilde{\nu}$.

Although the k–ε model has acquired a substantial following in the automotive industry, we have found the S–A model to be effective and accurate for a wide range of applications, including internal and external flows. The S–A model is less expensive than the k–ε model because only one partial differential equation is used to model turbulence and coarser discretizations are allowed while still resolving boundary layers. We have also found the S–A equation system to be less stiff than the k–ε counter-part, which makes its iterative solution more effective.

Turbulence modeling requires computation of the distance from a point to the nearest no-slip wall. To obtain this distance, a search algorithm is used to determine the projection of each node on the nearest

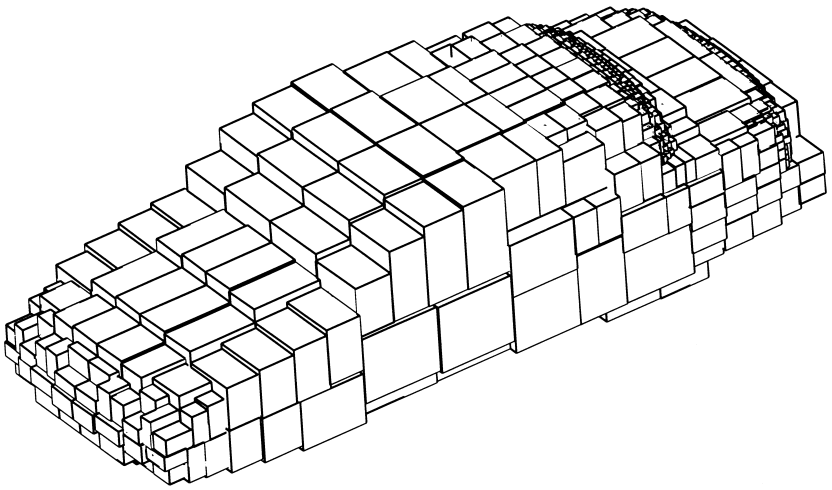

Fig. 6. Bucket sort illustration: each bucket contains approximately an equal number of boundary nodes on a car body model.

no-slip wall. This search employs a bucket sort (illustrated in Fig. 6) for efficiency. Furthermore, this search takes into account the mesh deformation when moving boundaries are used.

3. Parallel processing

The Spectrum solver employs a multiphysics architecture to support simulation of multiple interacting physical phenomena [2,3]. The multiphysics architecture supports different physical interactions through a data model defined as a hierarchical tree of regions and interfaces [4]. Regions of a problem are used to separate the different physics being analyzed over the spatial domain. To support various element types, each region may contain multiple element sets. Each element set is uniform in material model, element topology and element formulation. This decomposition organization is not only useful for user model management, but is also utilized to leverage data-parallel, vector and cache-sensitive hardware architectures.

The same software architecture that supports multiple physics simulation naturally and cleanly supports independent and parallel computation. For each problem, we maintain the concept of multiple subdomains. A subdomain is a collection of regions which are uniform with respect to linear solution technology (e.g., iterative, direct, etc.). This model provides the support and underlying data structures for efficient parallel processing (as illustrated in Fig. 7). At the higher-level, the coarse-grained subdomains map well to the SPMD programming model [2]. At the lower-level, the fine-grained element sets map well to the

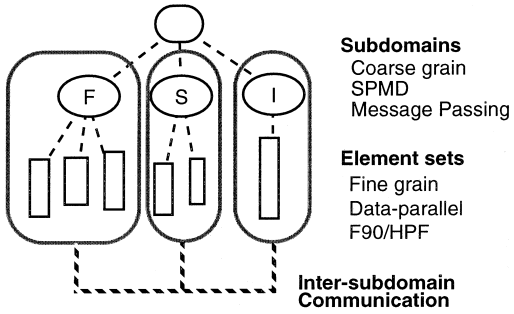

Fig. 7. Parallel processing models under the multiphysics architecture.

data-parallel model [14]. Therefore, the multiphysics architecture maps well to either (or both) parallel programming models.

The data-parallel approach has been investigated within the context of the present method [14–16]. Data-parallel programming requires significant re-coding in a data-parallel language such as Fortran-90 (F90) or High-Performance Fortran (HPF). Along with these dialects, proprietary languages also exist. Thus no data-parallel language standard exists. In addition, the programmer is expected to specify explicit, and often system dependent, data layout. It was also observed that the unified program view simplifies debugging and the investigation showed good speedup in many cases. However, the program development and support costs are significant and the resulting source code is generally not portable.

By comparison, the alternate approach of SPMD is implemented via message-passing. In this case, there is support for a wide range of hardware platforms with either PVM or MPI. This programming model has the advantage of allowing the reuse of large portions of code from the uni-processor version. Excellent performance has been demonstrated on a number of different parallel systems for large, multiphysics simulations. The relative ease of programming and support as well as the portability, scalability and wide availability of systems supporting SPMD make it the preferred model for our applications.

3.1. Coarse-grain parallelism

In the coarse-grain model, each processor of the parallel machine runs a copy of the application to solve one subdomain of the partitioned grid. A schematic description of the coarse-grain parallel architecture is presented in Fig. 8. Conceptually, this approach attempts to parallelize computation at the newly created subdomain level (i.e., at the outermost loop level). For example, the pseudocode below shows the outer subdomain loop implicit in the computation.

```
doacross (subdomains)
  perform local subdomain computations
enddo

communicate non-local data

doacross (subdomains)
  ...
enddo
```

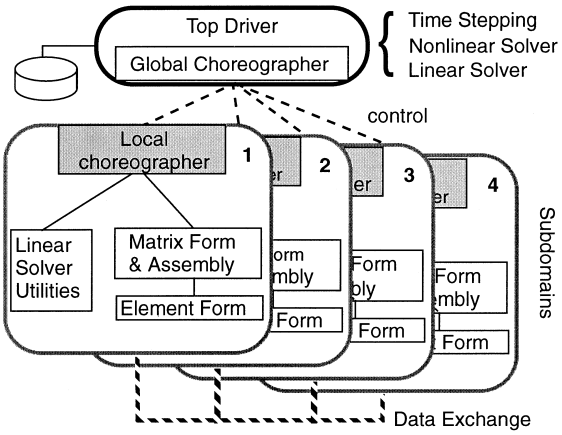

Fig. 8. Multi-subdomain architecture for coarse-grain parallel processing.

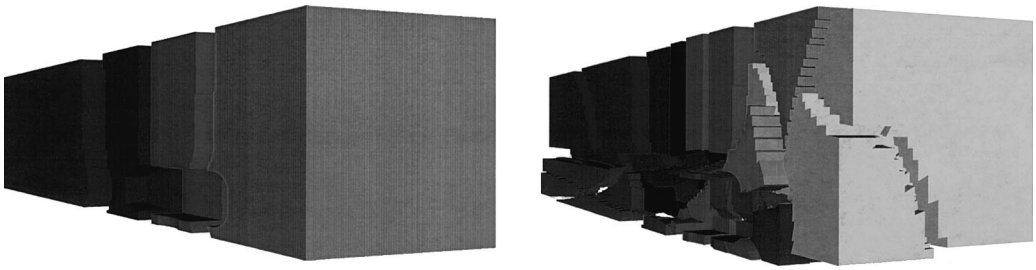

Fig. 9. Domain decomposition illustration.

3.2. Domain decomposition

The decomposition of the computational domain (e.g., Fig. 9) is an active area of research which has produced many algorithms and general purpose tools [18–25]. Many decomposition methods commonly use the recursive spectral bisection (RSB) approach [18]. A significant cost of the RSB method is associated with the computation of eigenvectors of a Laplacian matrix constructed from the adjacency structure of the mesh. Multi-level implementations for the construction of the Laplacian matrix resulted in significant CPU performance improvement [20,21]. Karypis and Kumar present a rigorous analysis of multilevel methods and demonstrate analytically their effectiveness [25].

A common feature of most domain decomposition research is the focus on single homogeneous grid applications. In order to support multiphysics simulations, algorithms which extend the multilevel method of Karypis and Kumar (METIS) to heterogeneous interfaced discretizations are used in the present approach [26].

Associated with domain decomposition, an additional module is used to determine the mappings required to support non-local communication of model data. For example, turbulence models often require at each fluid node the distance from that node to the nearest node on a no-slip wall. With a general domain decomposition a node's nearest wall segment need not exist within the node's own subdomain. The likelihood of this happening increases as a larger number of subdomains is requested. In this case, the module establishes the communication maps that permit the global determination of the distance to the nearest wall.

3.3. Usability

Our experience with parallel processing in the commercial setting uncovered a number of areas where software design must take a usability viewpoint. In particular, many parallel systems required significant additional knowledge by the end users of the application. This knowledge relates to launching parallel jobs, batch queue systems, dedicated vs. time-shared access (throughput issues) and "hidden" system resources such as communication switches, communication protocols in use (i.e., direct, IP), semaphores and shared memory segments.

In this work, system specific mechanisms were designed and implemented to provide as consistent an interface as possible for the user. A standard launch mechanism is now used in parallel Spectrum that constructs necessary batch control files, PVM initialization sequences, etc. and provides a simple, consistent command to start the parallel job running regardless of environment. From a user perspective, a simple "`problem. run`" command is used on each system. A simple command was also designed to cleanup the run-time environment as necessary (e.g., release system resources not freed) which is also provided for each environment.

As shown below, the difference between serial and parallel input files is simply the addition of 3 parameters (highlighted) to the command language:

```
DEFINE_PROBLEM \
  name                = ''frame'' \
  title               = ''external flow'' \
  type                = nonlinear \
  parallel            = on \
  fluid               = on

...
DEFINE_REGION \
  name                = ''flowfield'' \
  type                = fluid \
  state_law_model     = ''air'' \
  num_processes       = 12

...
RUN_SOLVER \
  mode                = batch
```

Another aspect of usability from an industrial application point of view, is the elimination of process non-determinism. It was observed that some computations proceeded along different solution paths when executed in different configurations and some results would vary slightly. The reason for these differences was discovered to be from two sources. First, the partitioning of the work by decomposition changes the sub-problems solved. The amount and range of work will make for local, machine precision rounding differences. These differences are insignificant and similar in quality to those observed between computer systems of different manufacturers. The second case was discovered to be due to non-deterministic operation ordering. Specifically, in the case of global sum operations, each subdomain computes a local sum and sends the result to the global driver process for summing. The order of the intermediate values can lead to different results. In a uniprocessor environment, the order is predetermined since each subdomain sum is executed and summed in the order of the control loop (i.e., `foreach subdomain: 1 to N`). However, in the parallel environment, each subdomain executes asynchronously and returns its results whenever it is finished and can communicate with the global driver.

Remark. *It should be noted that the actual solution results were always the same* (*within appropriate precision*), *but that various solution control values varied from run to run. This lead to differing numbers of solution steps for various runs, for example. It should also be noted that the uniprocessor ordering is not necessarily the best. The issue is primarily that from run to run, a user may observe differences in the path used to arrive at a solution. These user observable differences may lead to a lack of confidence in the final solution.*

The solution to this problem, once it was discovered and understood, was simple. The global driver collects all intermediate scalar results into an array in a fixed order (*i.e., indexed by subdomain*). *Once all results have been gathered, a simple loop through the array yields the desired global sum. This results in a deterministic path to solution.*

3.4. Superlinear scalability

An internal flow computation of a socket instability analysis is used to illustrate the scalability of our work. The simulation involves the computation of steady state flow around a solid cylinder inserted in a cooling socket. The computations were repeated for 250 thousand and 1 million element mesh discretizations using various domain decompositions in the range of 4–64.

The scalability of fluid flow computations on the IBM/SP2 platform is illustrated in Figs. 10 and 11. The 250 K element mesh exhibits near-theoretical scaling between 4 and 32 partitions with superlinearity at 12 and 16 subdomains. In general, this phenomenon is likely due to the subdomains becoming small enough to fit into the individual processor cache memory. Finally, the 1M element mesh shows excellent overall scaling between 24 and 64 partitions with superlinearity at 32 subdomains. Similar results for superlinear

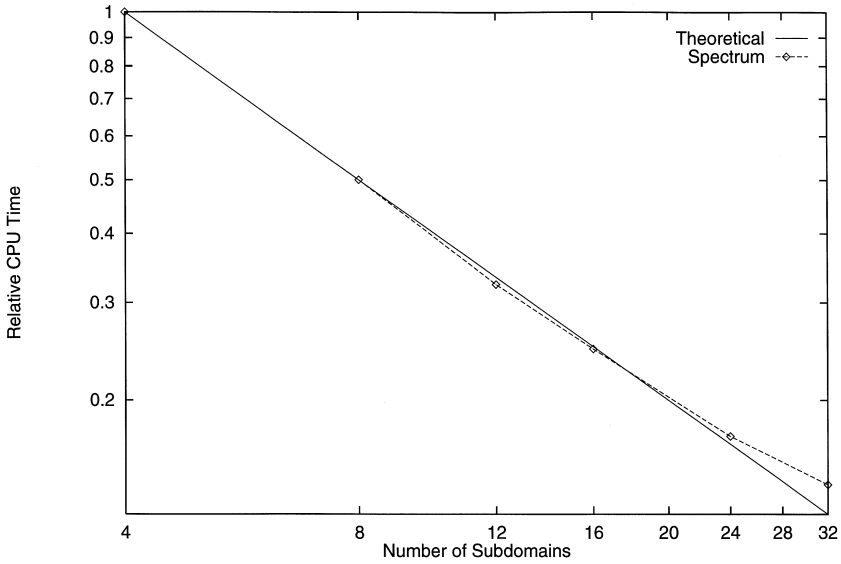

Fig. 10. Scalability of the 250 K element socket instability flow simulation. Near-theoretical scaling is seen in this case with superlinearity at 12 and 16 subdomains.

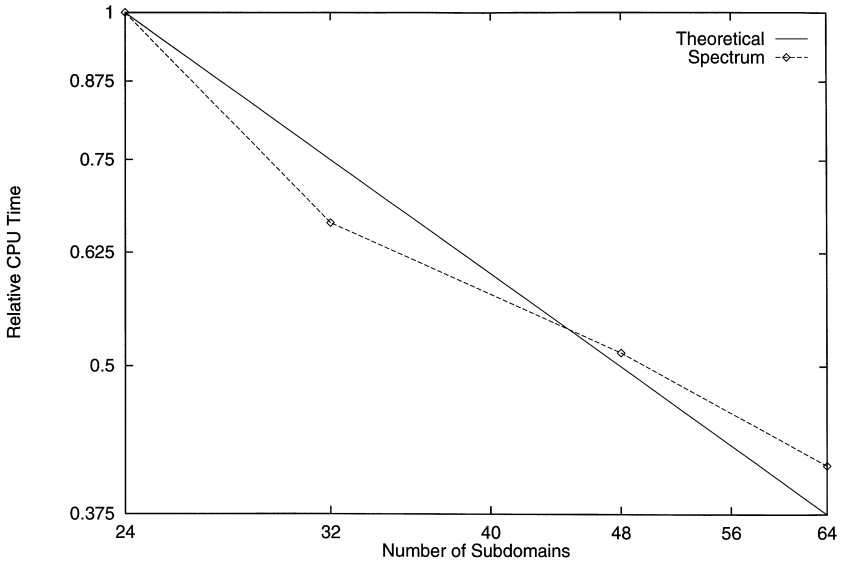

Fig. 11. Scalability of the 1M element socket instability flow simulation. Excellent overall scaling with superlinearity at 32 subdomains is observed.

scalability have also been observed in computations on the Silicon Graphics Origin 2000 and Hewlett–Packard Exemplar systems.

4. Applications

The work described herein has been applied to a variety of problems from the automotive industry. In this section, we illustrate applications of steady low speed exterior (incompressible) and unsteady high speed internal (compressible) flows. The simulations demonstrate the utility of the method in providing quality solutions for a variety of flows using automatically generated tetrahedral grids and accommodating moving boundaries with the resulting element distortion.

4.1. Vehicle exterior aerodynamics

Spectrum was used to simulate a wind tunnel experiment of a half-scale car body mounted on four pylons. Smaller-scale features such as tires, wheel-wells and outside mirrors were not represented in either the experiment or the simulations. The Reynolds number of the flow was 8.4 million, based on the length of the body.

An unstructured mesh consisting of 1.2 million tetrahedral elements was used to model half the vehicle body, assuming lateral symmetry (Fig. 12). The area around the rear window was meshed particularly finely in order to more accurately capture the separation line. Two layers of wedge elements (for a total of 430,000) were added over the vehicle body to resolve the boundary layer. The advantage of this approach is the use of smoothly varying grids with structured quadrilateral cross-sections near the walls for increased accuracy.

The boundary layer mesh generation begins by computing the normals at the wall nodes based on the surface triangulation. The normals are smoothed to avoid abrupt angles in the mesh near the wall. The tetrahedral elements are then moved away from the boundary into the interior and the wedge elements are inserted into the boundary layer.

The results of this steady flow simulation are depicted in Figs. 13–15. The predicted pressure contours on the surface of the vehicle are shown in Fig. 13. Line plots of the coefficient of pressure at the symmetry plane are presented in Fig. 14 for the upper and lower surfaces of the vehicle. Close agreement is noted between the experimental and computational results. In particular, the area behind the rear window exhibits the good prediction of the separation line and pressure on the trunk lid. Fig. 15 shows a set of flow velocity vectors at the symmetry plane depicting the recirculation behind the rear window.

4.2. Flow within an I.C. engine cylinder

The Spectrum solver has been used to simulate the unsteady flow without combustion inside an operating internal combustion engine. The compressible flow formulation with moving boundaries is used to model the motion of the piston and valves inside the cylinder. The geometric model used in this simulation (Fig. 16) is a 2 valve engine cylinder with its intake and exhaust ports [27]. An automatically generated

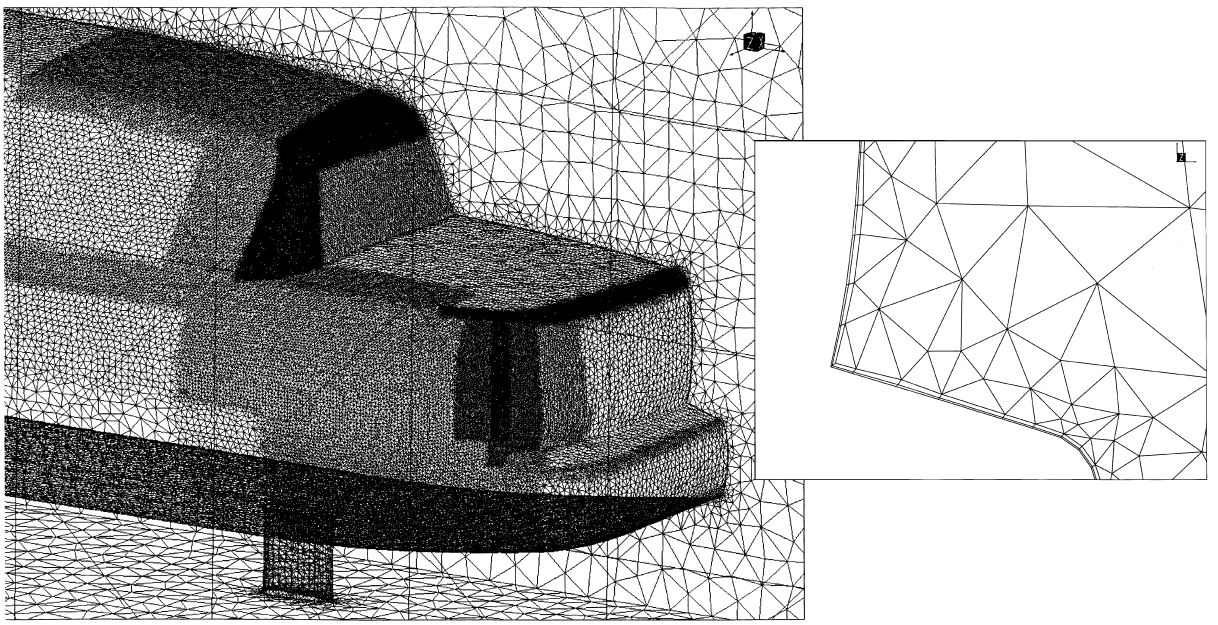

Fig. 12. Finite element mesh used for the exterior flow analysis. This automatically generated grid employs tetrahedral elements in most of the flow domain and two layers of wedge elements near the vehicle surface (inset).

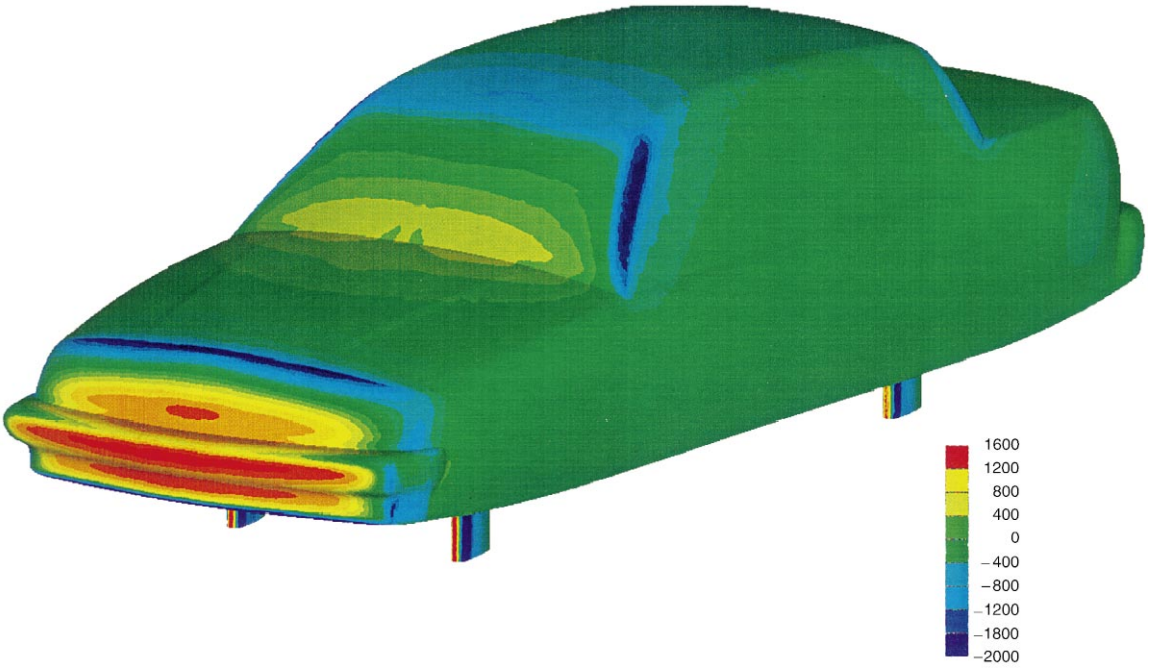

Fig. 13. Predicted pressure contours on the vehicle surface.

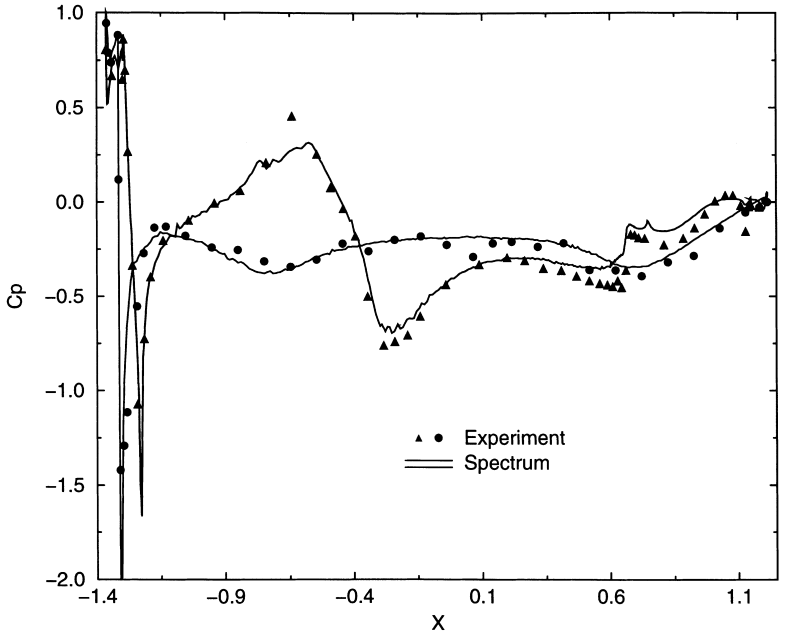

Fig. 14. Coefficient of pressure profile for the lower and upper vehicle surfaces at the symmetry plane. Close agreement is observed between computational and experimental results.

unstructured grid consisting of 593,731 tetrahedral elements and 121,907 nodes is employed. The engine is assumed to be running at 3000 RPM and the simulation is done for both the exhaust and intake strokes. Valve lift and piston position curves as functions of crank angle are used to prescribe the motion of the boundaries of the fluid domain.

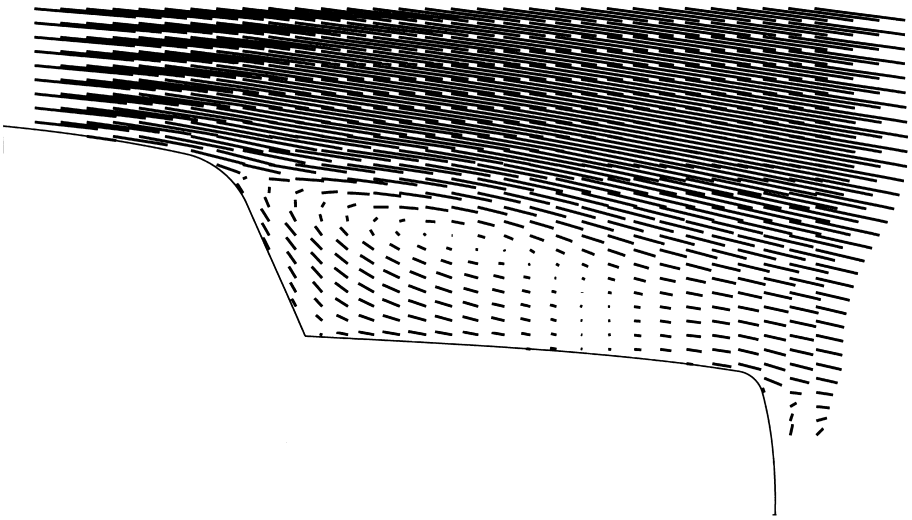

Fig. 15. Flow velocity vectors at the symmetry plane depicting the recirculation behind the rear window. The predicted separation point on the roof is in close agreement with the experimental observation.

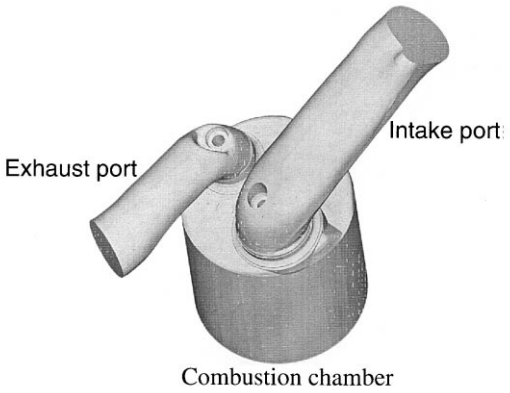

Fig. 16. Geometry of the fluid domain with the engine cylinder.

The flow problem is initialized with zero velocity and the pressures, temperatures and turbulent kinematic viscosities listed in Table 1 are used. Boundary conditions for the flow are specified at the intake and exhaust planes, fixed walls and moving walls. At the intake plane, pressure and temperature are specified as a function of crank angle; the turbulent kinematic viscosity is given as a function of velocity magnitude and the velocity direction is set parallel to the pipe wall. At the exhaust plane, pressure and temperature are

Table 1
Initial conditions for the in-cylinder flow problem[a]

Location	Pressure (kPa)	Temperature (K)	Kinematic viscosity (m^2/s)
Inside the intake port	90	296	10^{-6}
Inside the exhaust port	106	941	10^{-6}
Inside the cylinder	652	1860	10^{-3}

[a] Remark: No velocity profile is specified at the intake or exhaust ports. The simulation is therefore solved with pressure-driven boundary conditions. This setup allows for potential flow reversal at the intake and/or exhaust port without the need for changing the boundary conditions based on the flow direction.

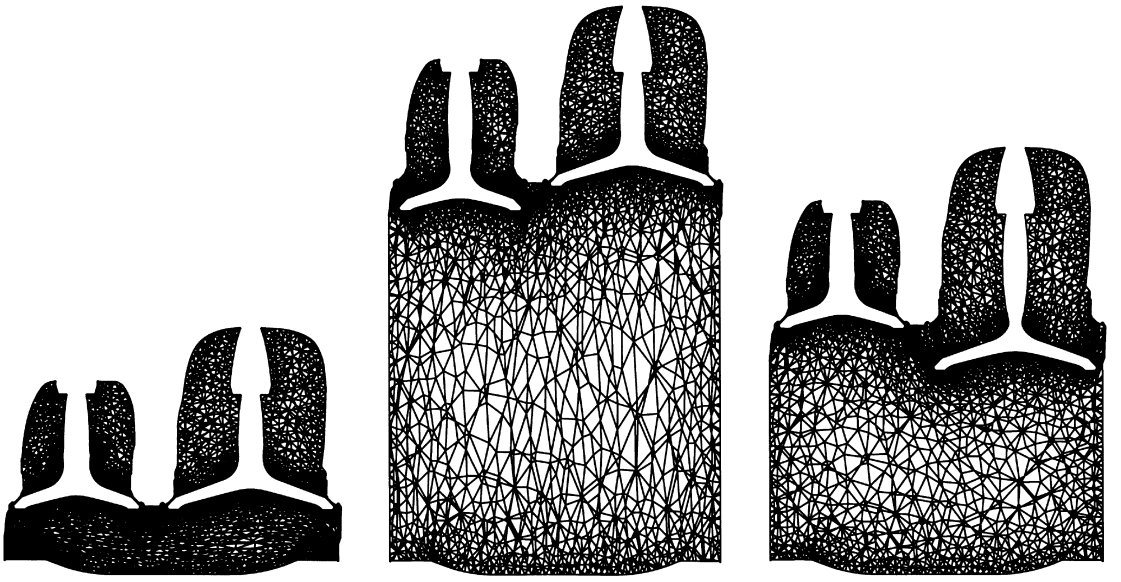

Fig. 17. Views of mesh deformation using one slicing plane through the cylinder at different crank angles.

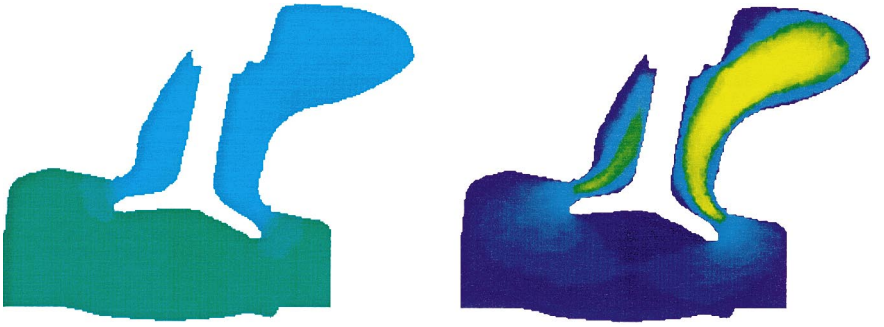

Fig. 18. Views of pressure and velocity magnitude using a slicing plane through the cylinder late in the exhaust stroke.

specified as a function of crank angle, and the velocity direction is set parallel to the pipe wall. On the fixed walls of the pipes and cylinder, the velocity and turbulent kinematic viscosity are set to zero. On the moving walls of the valves and piston, the velocity is specified as a function of crank angle and the turbulent kinematic viscosity is set to zero. An adiabatic boundary condition is used on the cylinder, piston, valve and pipe walls.

Figs. 17–19 depict the mesh deformation, pressure and velocity magnitude contours as several crank angles. Note that this simulation is performed with one single mesh. No remeshing was used in the entire 720° range of crank angle rotation. This illustrates the ability of the ALE method to admit large and complex boundary deformations and the ability of the GLS method to obtain quality solutions on highly distorted mesh configurations. This is an essential feature for the utility of fluid flow simulation tools within the context of simulation based design in the automotive industry.

5. Conclusions

We have presented a finite element method for the simulation of steady and unsteady flow problems. The formulation is based on the GLS method with discontinuity capturing operators. The ALE method is

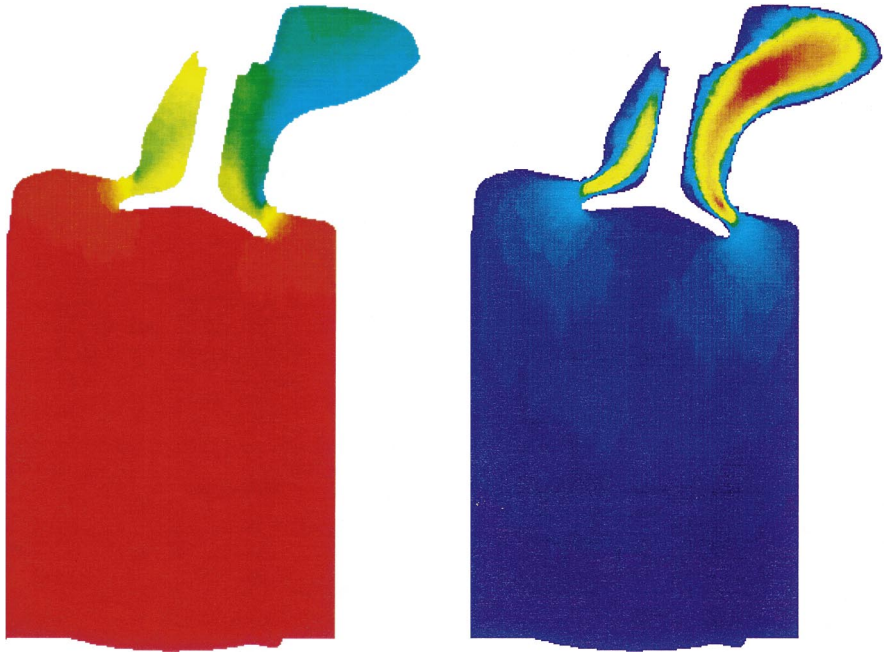

Fig. 19. Views of pressure and velocity magnitude using a slicing plane through the cylinder early in the exhaust stroke.

utilized to account for deformable fluid domains. The method is particularly forgiving to element distortion. By taking advantage of automatic mesh generation and parallel computing, dramatic reduction in turn-around time for flow analysis is achieved. Several applications are presented which demonstrate the utility and accuracy of finite element solutions in automotive engineering problems and high-light the scalability of the software.

Acknowledgements

Portions of this work were supported by the Defense Advanced Research Projects Agency under Federal Cooperative Agreement F30602-95-2-0007. Additional support was provided by the State of California under Defense Conversion Matching grants C95-0248 and C96-0078. Centric gratefully acknowledges this support as well as that provided by our commercial customers.

References

[1] T.P. Gielda, B.E. Webster, M.E. Hesse, D.W. Halt, The Impact of Computational Fluid Dynamics on Automotive Interior Comfort Engineering, AIAA Paper 96-0794, 34th Aerospace Sciences Meeting and Exhibit, January 1996.
[2] M. Eldredge, T.J.R. Hughes, R. Raefsky, R. Ferencz, S. Rifai, B. Herndon, High-performance parallel computing in industry, in: T. Tezduyar, T.J.R. Hughes (Eds.), Parallel Computing Special issue on High Performance Computing in Flow Simulations, vol. 23, 1997, pp. 1217–1233.
[3] S.M. Rifai, Multiphysics simulation for coupled fluids thermal and structural analysis on high-performance computing platforms, in: Proceedings of the Eighth Annual Thermal and Fluids Analysis Workshop (TFAWS 97), NASA Johnson Space Center, Houston, Texas, September 1997.
[4] S.M. Rifai, Z. Johan, W-P. Wang, J.-P. Grisval, T.J.R. Hughes, R.M. Ferencz, Multi-physics simulation of flow-induced vibrations and aeroelasticity on parallel computing platforms, Comput. Methods Appl. Mech. Engrg. 174 (1999) 393–417.
[5] T.J.R. Hughes, Recent progress in the development and understanding of supg methods with special reference to the compressible euler and Navier–Stokes equations, Int. J. Num. Meth. Fluids 7 (1987) 1261–1275.
[6] F. Shakib, T.J.R. Hughes, Z. Johan, A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier–Stokes equations, Comp. Meth. Appl. Mech. Eng. 89 (1991) 141–219.

[7] G. Hauke, T.J.R. Hughes, A unified approach to compressible and incompressible flows, Comp. Meth. Appl. Mech. Eng. 113 (1994) 3849–3896.
[8] K. Jansen, Z. Johan, T.J.R. Hughes, Implementation of a one-equation turbulence model within a stabilized finite element formulation of a symmetric advective–diffusive system, Comp. Meth. Appl. Mech. Eng. 105 (1993) 405–433.
[9] C. Johnson, A. Szepessy, P. Hansbo, On the convergence of shock-capturing stream- line diffusion finite element methods for hyperbolic conservation laws, Math. Comput. 54 (1990) 107–129.
[10] T.J.R. Hughes, W.K. Liu, T.K. Zimmerman, Lagrangian–Eulerian finite element formulation for incompressible viscous flows, Comp. Meth. Appl. Mech. Eng. 29 (1981) 329–349.
[11] T. Tezduyar, M. Behr, 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 numerical tests, Comp. Meth. Appl. Mech. Eng. 94 (1992) 339–351.
[12] C. Farhat, M. Lesoinne, N. Maman, Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation geometric conservation and distributed solution, Int. J. Numer. Meth. in Fluids 21 (1995) 807–835.
[13] P.R. Spalart, S.R. Allmaras, A one-equation turbulence model for aerodynamic flows, AIAA Paper 92-0439.
[14] Z. Johan, Data-parallel finite element techniques for large-scale computational fluid dynamics, Ph.D. Thesis, Stanford University, Stanford, 1992.
[15] Z. Johan, T.J.R. Hughes, K.K. Mathur, S.L. Johnsson, A data parallel finite element method for computational fluid dynamics on the connection machine system, Comp. Meth. Appl. Mech. Eng. 99 (1992) 113–134.
[16] Z. Johan, K.K. Mathur, S.L. Johnsson, T.J.R. Hughes, An efficient communications strategy for finite element methods on the connection machine CM5 system, Comp. Meth. Appl. Mech. Eng. 113 (1994) 363–387.
[17] M.S. Shephard, M.K. Georges, Automatic three-dimensional mesh generation by the finite octree technique, Int. J. Num. Meth. Eng. 32 (1991) 709–7449.
[18] H.D. Simon, Partitioning of unstructured problems for parallel processing, Comp. Sys. Eng. 2 (1991) 135–148.
[19] C. Farhat, M. Lesoinne, Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics, Int. J. Num. Meth. Eng. 36 (1993) 745–764.
[20] B. Hendrickson, R. Leland, A Multi-level Algorithm for Partitioning Graphs, Sandia National Labs Rep., Albuquerque, NM 87185-1110.
[21] D. Vanderstraeten, C. Farhat, P.S. Chen, R. Keunings, O. Zone, A retrofit and contraction based methodology for the fast generation and optimization of mesh partitions: beyond the minimum interface size criterion, Comp. Meth. Appl. Mech. Eng. 133 (1996) 25–45.
[22] J.G. Malone, Automated mesh decomposition and concurrent finite element analysis for hypercube computers, Comp. Meth. Appl. Mech. Eng. 70 (1998) 27–58.
[23] A. George, J. Lui, Evolution of the minimum degree ordering algorithm, SIAM Review 31 (1989) 1–19.
[24] M.T. Heath, P. Raghavan, A Cartesian Nested Dissection Algorithm, Technical Report, UIUCDCS-R-92-1772 Department of Computer Science, University of Illinois, Urbana, IL, 1992.
[25] G. Karypis, V. Kumar, Analysis of multilevel graph partitioning, Report 95-037, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455, 1995.
[26] G. Karypis, V. Kumar, METIS: Unstructured Graph Partitioning and Sparse Matrix Ordering System, Department of Computer Science, University of Minnesota, Minneapolis, MN 1995.
[27] Z. Johan, A. Moraes, J. Buell, R.M. Ferencz, In-Cylinder Cold Flow Simulation using a Finite Element Method, to appear.