ARMY RESEARCH LABORATORY

Three-Dimensional (3D) Large Fluid Flow Computations for U.S. Army Applications on KSR-1, CM-200, CM-5, and Cray C-90

Nisheeth Patel
Harris Edge
U.S. ARMY RESEARCH LABORATORY

Jerry Clarke
COMPUTER SCIENCES CORPORATION

ARL-TR-712 February 1995

NOTICES

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Road, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute endorsement of any commercial product.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	February 1995	Final, Jul 92–Jul 94

4. TITLE AND SUBTITLE

Three-Dimensional (3-D) Large Fluid Flow Computations for U.S. Army Applications on KSR-1, CM-200, CM-5, and Cray C-90

5. FUNDING NUMBERS

62783-A094

6. AUTHOR(S)

Nisheeth Patel, Harris Edge, and Jerry Clarke*

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Research Laboratory
ATTN: AMSRL-CI-CA
Aberdeen Proving Ground, MD 21005-5067

8. PERFORMING ORGANIZATION REPORT NUMBER

ARL-TR-712

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

Jerry Clarke is employed by Computer Sciences Corporation

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The thrust of this research is to investigate the use of emerging massively parallel computer architectures for production type fluid flow applications. Typical flow computations in a ballistic simulation have reached a stage where a grid set involving a couple of million points are needed for adequate understanding of the flow behavior and computing the resulting forces acting on a projectile with a complex geometrical shape. For acceptable turnaround time, the simulation requires a system with large memory and high performance. Because of the hardware scalability and availability of large in-core memory, the emerging massively parallel processors (MPP) architectures such as Connection Machine's CM-5 and Kendall Square Research KSR-1 have become attractive for ballistic computations. Thus, this research is focused on implementation of complex real-world applications (not simple scaled down problems) on MPPs with the goal of sustainable performance. Because performance measurements obtained using the kernel of the full application and/or scaled down application size or 2-D cases could lead to considerably higher performance estimates, these factors are discussed in light of results from full 3-D applications obtained on MPPs.

14. SUBJECT TERMS

computational fluid dynamics, projectiles, massively parallel computers, missiles, high performance computing, wraparound fin projectile

15. NUMBER OF PAGES

25

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

INTENTIONALLY LEFT BLANK.

TABLE OF CONTENTS

INTENTIONALLY LEFT BLANK.

LIST OF FIGURES

LIST OF TABLES

INTENTIONALLY LEFT BLANK.

1. INTRODUCTION

The thrust of this research was to explore massively parallel processing (MPP) system architectures for production-type applications. Flow computations for ballistic simulations have reached a stage where a grid set involving several million points may be needed for adequate understanding of the flow behavior and to compute the resulting forces acting on a projectile. On most available Cray Y-MP-type supercomputers, these computations would require excessive CPU time and memory. Thus, the emerging massively parallel computer architectures such as the Thinking Machine Corporation's CM-200 and CM-5 and the Kendall Square Research Corporation's KSR-1 have become attractive alternatives for high-performance ballistic computations.

There were three major goals of this work. The first was to develop new algorithms that involve complex data manipulation for efficient use of a given MPP system interconnect network. The second goal was to explore these MPP systems for production-type Army projectile aerodynamics applications. The third goal was related to the other two, in which the sustainable performance of the MPP was investigated. Real-world three-dimensional (3-D) geometries were selected for this study. The applications involved grid sizes on the order of a million points. Usually, single-zone grids are used for relatively simple geometries, but multiple-zone grids are desired for complex body shapes. Because of the complex body shapes, a multiple-zone grid was considered for this study.

The complex boundary conditions associated with practical applications were also retained. In particular, none of the boundary conditions was simplified or hard-coded inside the code to increase the computational efficiency of the application. This is so because, in practice, boundary conditions are written in modular blocks that make it easy to set up the same code for a variety of applications. Again, emphasis was placed on solving the entire application including external input/output (I/O). Performance of the application is presented as a whole and not based on just a few iterations of the code or performance of the code kernel.

The remote location of the computational facilities has led to the development of tools for transferring enormous amounts of data to a local machine for visualization of the 3-D data. These tools have been used for displaying very large amounts of data generated on the CM-200 (on a real-time basis). By transferring data from the CM-200 using Remote Procedure Calls and External Data Representation (XDR), a visual representation of the data has been sent to an X11 window or other devices using frame

buffers. These tools provide not only efficient visualization capability but also provide an initial user scenario interface for debugging applications involving millions of grid points on the CM-200. Some I/O and networking approaches are discussed for their relative merits and their impact on overall efficiency with regard to the performance of the applications on the CM-200.

2. CM-200, CM-5, AND KSR-1

The CM-200 is a single-instruction multiple-data set (SIMD) parallel computer. In the CM-200, 16 data processors and router circuitry, used for interprocessor communication, are placed on a single chip. Each data processor has an arithmetic-logic unit (ALU). There is one floating-point accelerator for a group of 32 data processors. A "node" consists of 32 data processors and their memory, a floating-point accelerator, and communications interfaces. The CM-200 located at the Army High-Performance Computing Research Center (AHPCRC) has 1,024 nodes or 32,768-bit serial processors. Because of the SIMD nature of the CM-200, it is not possible to overlap computation with communication. Thus, communication, if not implemented properly, can significantly degrade the overall performance. The CM-200 has a regular grid communication within its hypercube topology. This grid or "NEWS" communication is designed in such a way that grid neighbors are assigned to hypercube neighbors in the communication network. This allows every processor, in parallel, to pass data to its neighbor, all in the same direction. The cost of grid communication is typically very competitive with the cost of the basic arithmetic. On the other hand, random access or long-distance interprocessor communication is far more expensive than regular grid communication [1].

The CM-5, located at AHPCRC, has 864 processors. The CM-5 is a multiple-instruction multiple-data MIMD) parallel computer. Each cell has a 128 Mflops CPU. The CPUs have four vector units each of 32 Mflops per vector. Each cell is connected to 32 MB of local memory. The cells are connected using a separate control network and high bandwidth data network. The CM-5 also has a high bandwidth parallel data storage system. This disk storage system, called scalable disk array, is integrated into the CM-5 control and data network. Each node of this array contains a microprocessor-based controller with a network interface, a large disk buffer, four controllers, and eight hard disk drives. The data storage nodes have been accommodated within the data network. The parallel I/O operations makes use of these scalable disk array nodes [2].

The Kendall Square Research Corporation's KSR-1 at the U.S. Army Research Laboratory's (ARL) Aberdeen Proving Ground site is a multiprocessor computer with 256 processor and memory cells. Each cell has a 40-Mflops peak CPU and 512-kB cache called "subcache" connected to 32 MB of local memory called "local cache." About half of the subcache can be used for data. A unique feature of the KSR-1 is its hardware connection of all local memory. This makes the local memory look like a globally shared memory. The exchange of data between cells is organized in subpages of 128 B (16 words). The access time from registers to a word in the subcache is 2 cycles, while the access time to a 64-B subblock in the local cache is 20 cycles. Access time to a subpage in the remote cache, at the same level, is 130 cycles. The time to access a subpage from the next level increases to 570 cycles [3]. Because of this memory hierarchy, the data access can have a big impact on the performance when required data are not in the subcache.

3. GOVERNING EQUATIONS AND NUMERICAL METHOD

All flow field solutions were computed with a 3-D, full Navier-Stokes code which can be applied to zonal grid topologies. The compressible Reynolds-averaged Navier-Stokes equations in a 3-D generalized coordinate system are written in the following strong conservation form [4]. The dependent variables, ρ, u, v, w, and e, are mass averaged, where e is specific total energy, T temperature, ρ mean density, p pressure, and t time.

$$\frac{\partial \hat{Q}}{\partial t} + \frac{\partial (\hat{E} - \hat{S})}{\partial \xi} + \frac{\partial (\hat{F} - \hat{T})}{\partial \eta} + \frac{\partial (\hat{G} - \hat{R})}{\partial \zeta} = 0. \qquad (1)$$

The generalized coordinates are as follows:

$\xi = \xi$ (x, y, z) - longitudinal coordinate
$\eta = \eta$ (x, y, z) - near-normal coordinate
$\zeta = \zeta$ (x, y, z) - circumferential coordinate
t = t - time.

The flux vectors are defined as

$$\hat{Q} = \frac{1}{J}\begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{bmatrix} \quad \hat{E} = \frac{1}{J}\begin{bmatrix} \rho U \\ \rho u U + \xi_x p \\ \rho v U + \xi_y p \\ \rho w U + \xi_z p \\ (\rho e + p) U \end{bmatrix} \quad \hat{F} = \frac{1}{J}\begin{bmatrix} \rho V \\ \rho u V + \eta_x p \\ \rho v V + \eta_y p \\ \rho w V + \eta_z p \\ (\rho e + p) V \end{bmatrix}$$

$$\hat{G} = \frac{1}{J}\begin{bmatrix} \rho W \\ \rho u W + \zeta_x p \\ \rho v W + \zeta_y p \\ \rho w W + \zeta_z p \\ (\rho e + p) W \end{bmatrix} \quad \hat{S} = \frac{1}{J}\begin{bmatrix} 0 \\ \xi_x \tau_{xx} + \xi_y \tau_{xy} + \xi_z \tau_{xz} \\ \xi_x \tau_{yx} + \xi_y \tau_{yy} + \xi_z \tau_{yz} \\ \xi_x \tau_{zx} + \xi_y \tau_{zy} + \xi_z \tau_{zz} \\ \xi_x \beta_x + \xi_y \beta_y + \xi_z \beta_z \end{bmatrix}$$

$$\hat{T} = \frac{1}{J}\begin{bmatrix} 0 \\ \eta_x \tau_{xx} + \eta_y \tau_{xy} + \eta_z \tau_{xz} \\ \eta_x \tau_{yx} + \eta_y \tau_{yy} + \eta_z \tau_{yz} \\ \eta_x \tau_{zx} + \eta_y \tau_{zy} + \eta_z \tau_{zz} \\ \eta_x \beta_x + \eta_y \beta_y + \eta_z \beta_z \end{bmatrix} \quad \hat{R} = \frac{1}{J}\begin{bmatrix} 0 \\ \zeta_x \tau_{xx} + \zeta_y \tau_{xy} + \zeta_z \tau_{xz} \\ \zeta_x \tau_{yx} + \zeta_y \tau_{yy} + \zeta_z \tau_{yz} \\ \zeta_x \tau_{zx} + \zeta_y \tau_{zy} + \zeta_z \tau_{zz} \\ \zeta_x \beta_x + \zeta_y \beta_y + \zeta_z \beta_z \end{bmatrix},$$

where

$$\tau_{xx} = -\frac{2}{3}(\mu + \varepsilon)(u_x + v_y + w_z) + 2(\mu + \varepsilon) u_x$$

$$\tau_{xy} = \tau_{yx} = (\mu + \varepsilon)(u_y + v_x)$$

$$\tau_{xz} = \tau_{zx} = (\mu + \varepsilon)(u_z + w_x)$$

$$\tau_{yy} = -\frac{2}{3}(\mu + \varepsilon)(u_x + v_y + w_z) + 2(\mu + \varepsilon)v_y$$

$$\tau_{yz} = \tau_{zy} = (\mu + \varepsilon)(v_z + w_y)$$

$$\tau_{zz} = -\frac{2}{3}(\mu + \varepsilon)(u_x + v_y + w_z) + 2(\mu + \varepsilon)w_z$$

$$\beta_x = C_P\left(\frac{\mu}{P_r} + \frac{\varepsilon}{P_{r_t}}\right)\frac{\partial \hat{T}}{\partial x} + u\tau_{xx} + v\tau_{xy} + w\tau_{xz}$$

$$\beta_y = C_P\left(\frac{\mu}{P_r} + \frac{\varepsilon}{P_{r_t}}\right)\frac{\partial \hat{T}}{\partial y} + u\tau_{yx} + v\tau_{yy} + w\tau_{yz}$$

$$\beta_z = C_P\left(\frac{\mu}{P_r} + \frac{\varepsilon}{P_{r_t}}\right)\frac{\partial \hat{T}}{\partial z} + u\tau_{zx} + v\tau_{zy} + w\tau_{zz}.$$

The molecular viscosity is μ, and the turbulent viscosity is ε. The Jacobian of the generalized coordinates is

$$J^{-1} = x_\xi y_\eta z_\zeta + x_\eta y_\zeta z_\xi + x_\zeta y_\xi z_\eta - x_\xi y_\zeta z_\eta - x_\eta y_\xi z_\zeta - x_\zeta y_\eta z_\xi.$$

The velocities in the ξ, η, and ζ coordinates are

$$\begin{aligned} U &= \xi_x u + \xi_y v + \xi_z w \\ V &= \eta_x u + \eta_y v + \eta_z w \\ W &= \zeta_x u + \zeta_y v + \zeta_z w, \end{aligned} \tag{2}$$

which represent the contravariant velocity components.

The air was assumed to be a perfect gas, satisfying the equation-of-state

$$p = \rho RT, \tag{3}$$

where R is the gas constant (287 N-m/kg-K for air). For the dependence of laminar viscosity on the temperature, Sutherland's law was used:

$$\mu = 1.458 \frac{T^{3/2}}{T + 109.333} \times 10^{-6} \frac{kg}{m\text{-}s}. \tag{4}$$

The laminar and turbulent Prandtl numbers, P_r and P_{r_t}, were assumed constant with values of 0.72 and 0.9, respectively. The ratio of specific heats was also assumed constant and equal to 1.4. The specific heat capacities at constant volume and constant pressure are C_v and C_p, respectively.

(C_v = 717 J/kg-K and C_p = 1,004 J/kg-K for air).

The total energy per unit mass, e, is given by

$$e = C_v T + 0.5 (u^2 + v^2 + w^2). \tag{5}$$

The local pressure is determined using the following relation

$$p = \rho (\gamma - 1)\left[e - 0.5\left(u^2 + v^2 + w^2\right)\right]. \tag{6}$$

4. NAVIER-STOKES ALGORITHM IMPLEMENTATION

Since the scalable architectures of the KSR-1, CM-200, and CM-5 are well-suited for an explicit-type method, an explicit multiple-stage finite-difference method has been adopted. The code solves the Navier-Stokes equations in generalized coordinates with no thin layer assumption. The code is capable of performing both steady and unsteady computations. For steady flows, a local time-stepping convergence acceleration scheme is used [4].

Adaptive artificial dissipation terms have been added to the equations. The dissipation terms are required to dampen high-frequency oscillations associated with the odd-even decoupling in central-difference schemes. Also, the artificial dissipative terms are required to capture shock and contact

discontinuities without undesirable oscillations. An adaptive blend of second- and fourth-order differences has been used for artificial dissipation. The dissipation terms have been improved by a directional eigenvalue scaling, which has been found effective for highly stretched grids in both inviscid and boundary layer or wake regions [4].

For a time-accurate solution, the explicit scheme requires a limited time-step size that must be determined by the numerical stability criterion of the scheme. Thus, a time-step size dictated by Courant-Friedrichs-Lewy (CFL) condition is the maximum over all grid cells. However, convergence to the steady-state solution can be accelerated by sacrificing the time accuracy of the scheme and advancing the solution at each cell in time by the maximum possible local time-step in that cell. The local time-stepping is also well-suited for parallel processing because it allows concurrent computation of the time-step size in all cells and avoids the long distance communication involved in computing the maximum time-step size.

The solution process requires the computation of a flux value that includes artificial dissipation from the dependent variables. This can be done concurrently for all interior grid points in a given zone. In the case of a multiple-zone grid, the aforementioned process is performed for each zone until all zones are finished. Then the solution array is obtained by solving the first stage in parallel for all interior points of a given zone. Again, the process repeats for any additional zones. Next, appropriate boundary conditions are updated, each in parallel. The major concern when applying the complex boundary conditions, which take longer, is that processors representing interior grid points remain inactive or do not participate in any useful work. Thus, boundary conditions must be carefully considered. The interzone data exchange is done next for overlaid zones. Each grid zone is separately mapped to memory. Since each grid zone can have different sizes and boundary conditions, a substantial amount of long distance communication may be involved in the exchange process. Although no computation is involved in this exchange process, there is a large communications overhead. This cycle repeats for the next stage. Variants of this cycle can be explored by applying boundary conditions and/or interzone exchanges less frequently in each stage, to achieve better performance.

The code was rewritten using the CM FORTRAN language for the CM-200 and CM-5. CM FORTRAN is based on the array syntax of FORTRAN 90 and has additional directives that allow users to guide the compiler to distribute data to memory in a desired pattern. The CM compiler and data network handle both regular and irregular communication in codes based on assigned data layout. Efficient data layout increases the possibility of data locality, thus increasing the efficiency of the regular

communication. Once data are laid out, their address on the CM-5 remains physically static in the memory [5].

The KSR-1 memory architecture is substantially different from the previously mentioned CM-5. On the KSR-1, which is called an "allcache memory" machine, there is no fixed physical location for a data address in the memory. When needed, the data move to the node where the code called for them [6]. Initially for the KSR-1, the original CRAY version of the code was modified for use on the KSR-1 with KSR parallel directives with very little effort. However, in an attempt to obtain increased performance on the KSR-1, the code underwent substantial restructuring, which required considerable effort.

5. VISUALIZATION

The visualization of flow data demands rendering of large quantities of 3-D data. The vast quantities of data generated on massively parallel computers require improvement to the traditional visualization techniques. While machines like the CM-200 have attached frame buffers, these machines are typically located at a remote site. It is usually desirable to transfer the data to a local machine to do the visualization. The massive amounts of data, however, make it difficult to transfer files to another architecture and usually are too large for many workstations' modest disk capacity.

One approach, described here, is to efficiently transfer selected portions of data to a local machine without a large disk capacity and accomplish the visualization almost entirely in software. This allows the visualization to be accomplished on machines that do not have special graphics hardware. In this way, massive amounts of data can be immediately visualized and stored for future visualization while not impeding the original computation. If the data transfer and visualization steps are separated, the computation, visualization, and storage can happen concurrently. In addition, the process can be maximized for a particular application by performing each step on the most appropriate architecture.

For these reasons, the visualization of flow data from the CM-200 has been broken into two steps: (1) efficiently transferring the data to a local computer, and (2) displaying and manipulating images of the data. In an effort to allow flexibility in choosing platforms for the data transfer and visualization, both steps are portable, relying on no special hardware or architecture-specific software.

Each time-step of the Navier-Stokes calculation produces several scalar values at each grid point (density, X, Y, Z momentum and energy). Every tenth time-step is dumped to a file on the Data Vault. The filename, along with the size, shape, and normalizing values are written to a shared memory segment queue on the front-end processor. The Zonal code is then free to resume processing. This results in a minimal impact on the code performance since the transfers to the Data Vault and shared memory are fast operations.

A separate process reads the information from the shared memory queue, opens the Data Vault file, reads it into the CM-200, and normalizes the data. Then using RdT, a data transfer library, selected planes are transferred to a local processor. The data are transferred in XDR format, developed by Sun Microsystems, and stored on the local machine in host binary format. Once on the local machine, the data are visualized in an X11 window using ShAYD, a collection of hardware-independent routines that are used to visualize curvilinear, multiple-zone grids and their scalar values. Since all of the rendering is done in software, massive grids can be handled, one small section at a time, to produce a final image of the entire time-step. Each scalar value is mapped to a hue, and a light source can be optionally applied. The hue is interpolated across each polygon and used, along with the light value, to produce either a value in the X11 color map or a 24-b RGB value. The 24-b output is used to produce videos of many time-steps.

6. FLOW APPLICATIONS

The following applications are flow field computations of projectile shapes which are representative of those used by the U.S. Army. The M549 application is an example of a shell at angle of attack. The wraparound fin projectile application is an example of a finned projectile or missile. Both applications require large amounts of computer memory and time.

6.1 M549 Projectile. The projectile shape consists of a flat nose, a forebody section composed of an ogive-cylinder, and an aftbody section which includes the boattail and flat base. The zonal grid preserves the actual corners in the nose and base area. The zonal grids are overlapped by single cell with the adjacent neighboring grids without any mismatch. Thus, interzone boundary conditions do not require any interpolation or computation. Only data transfer is involved at zonal interfaces. A nonreflecting boundary condition was imposed on the outer boundary [4]. This approach allows setting of the outer boundary close to the body. From the computational point of view, it allows the removal of a relatively large number of grid points from the outer region for supersonic flow computations. This decreases

memory requirements because the memory used for the computation is proportional to the total number of grid points. The eliminated grid points result in decreased computation time for a given application. However, the computation involved in a nonreflecting-type boundary condition is relatively more involved than a simple free-stream-type boundary condition. The overhead imposed by complex boundary conditions is relatively low on Crays compared to the CM-200 and CM-5.

For Mach = 1.5 and an angle of attack of 3^{o}, the pressure contours in Figure 1 clearly show the asymmetry in the shock wave at the nose area and the region of overexpansion and recompression. Notice that the use of a nonreflecting boundary condition allows the shock to pass through the outer boundary. A sequence of frames from the visualization of the computed flow fields reveals unsteady vortex shedding with vortices being formed and shed in the wake. As the Mach number is increased, the recirculation region becomes entrapped in the area behind the base area. The visualization may raise questions about dislocation of events. Real-time visualization opens up an area that needs further investigation and may prove to be a key element in this type of study.

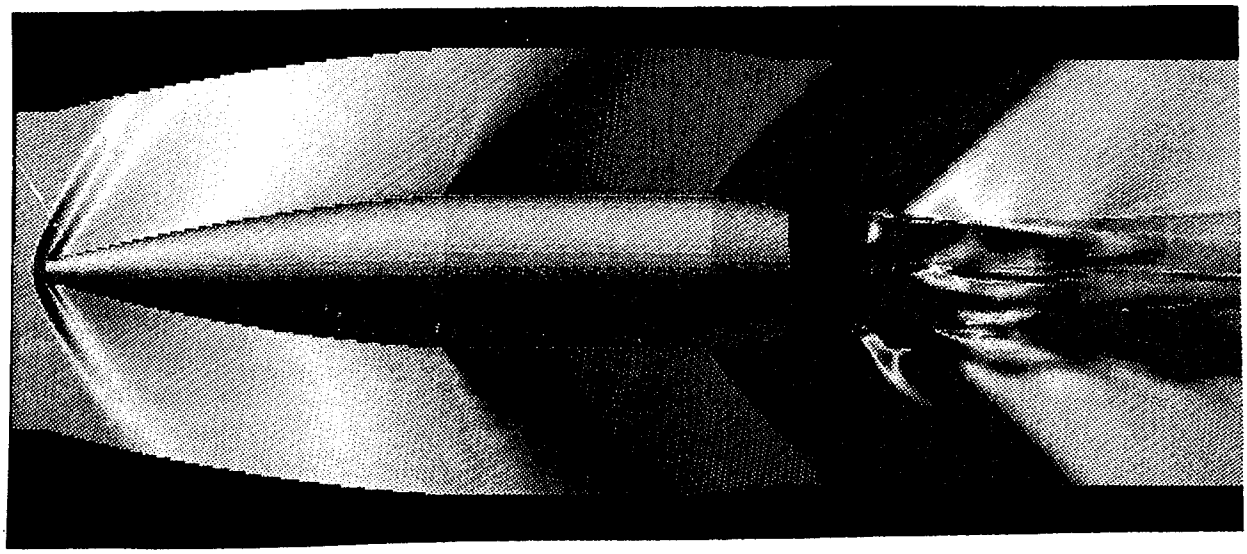

Figure 1. Pressure contours for M549 projectile.

The M549 was modeled using a three-zone grid which had dimensions in powers of two. The grid dimensions of the three zones were (32×128×64), (256×128×64), and (128×64×64). The performance of the entire code for the aforementioned grid on the CM-200 using 32K processors or 1,048 nodes is approximately 1,400 Mflops. In comparison, the performance of the same application on a Cray-2 rated

at 90 Mflops using a single processor and 340 Mflops using four processors. The Cray-2 is used for comparison because it has a large main (core) memory required for 3-D applications.

Performance can vary significantly on the CM-200 based on the grid dimensions. A three-zone grid with dimensions of (30×94×39), (249×65×39), and (120×46×39) and containing approximately 1 million points, was also used for the M549 application. Code performance was significantly worse for this grid than the previously mentioned power-of-two grid which has approximately 2.9 million points. With the power-of-two grid, the code completes one iteration every 12 s. Using the 1 million point grid, the code needed 24 s to complete a single iteration. Despite its larger size, the code ran faster using the power-of-two grid on the CM-200.

As stated previously, the entire M549 application, using the power-of-two grid, including various boundary conditions and interzone communications takes approximately 12 s/iteration on a 32K processor CM-200. A detailed breakdown of the timing reveals that the code spends only about 4 s/iteration solving the kernel (or working on the interior grid points) and 8 s/iteration applying the boundary conditions which includes zonal interface data transfer (or working on the boundary points). As discussed previously, the data transfer and computations on the zonal boundaries can impose impressive overhead, which can cause a huge gap between the performance of the kernel and the performance of the entire code for this type of application. In this case, interzone communication has turned out to be one of the most expensive operations on the CM-200 because of the long-distance communication involved. It would not be difficult to find applications involving a single-block grid and simple boundary conditions that can be executed concurrently with the interior points and obtain much better performance than obtained for the M549 application. The main emphasis here are the issues that were encountered while keeping the code flexible and practical enough for a variety of applications.

6.2 Wraparound Fin Projectile. Wraparound fins have an advantage over most fin designs. The shape of the fin allows it to be folded next to the projectile body, which provides benefits in storing, packaging, and launching. Tube-launched projectiles are an example of the benefits wraparound fins can provide. The wraparound fin conforms to the cylindrical shape of the projectile while in the launch tube, providing an efficient use of space. However, the cylindrical shape of wraparound fins does have a disadvantage. The aerodynamics of the fin may cause undesirable flight performance. For instance, the roll moment coefficient of projectiles employing wraparound fins varies in sign and magnitude with the Mach number. The effect during flight could mean undesirable changes in spin rate and spin direction,

leading to the instability of the projectile. It is hoped that computational fluid dynamics (CFD) can be used to predict the aerodynamics of wraparound fin projectiles and provide a valuable tool for their design [7].

The flow field of a wraparound fin projectile is intricate. Factors that can affect the fin's performance include shock interaction, fin-body juncture effects, and boundary layer separation. To capture these effects, a suitable computational model must be provided. Some of these flow field characteristics are evident in Figure 2, which shows normalized pressure contours of the wraparound fin projectile at a Mach number of 1.5. The geometry of the wraparound fin projectile would be difficult to model without using a zonal grid topology. A cutaway view of the computational grid can be seen in Figure 3. The zonal approach facilitated the building of the computational model. Like the previously mentioned M549 grid, overlapped zones share at least one grid cell in a given direction with an adjacent zone. The shared grid cells have identical coordinates in both of the overlapped zones. The communication between zones for the wraparound fin is more complex than for the M549. The wraparound fin requires more zones to model the geometry than the M549. In addition, the zones for the wraparound fin has more adjacent neighboring zones with which to communicate. For example, the zone defining the ogive-cylinder portion of the wraparound fin body must exchange information with the other four zones of the grid, while the corresponding zone of the M549 exchanges information with only two zones. Like the M549, a multi-zone solution is obtained by performing the integration of the governing equations in all zones and then exchanging information between overlapping zones before advancing to the next iteration. The use of a wraparound grid would have resulted in large and rapid variations of the metric terms, which could have severely degraded the quality of the solution. The zonal approach allowed the accurate modeling of the wraparound fin projectile's geometry, while retaining a smooth continuous computational mesh.

The computationally modeled surface was one-fourth of the entire projectile surface. The symmetry of the projectile allowed the modeling of one fin and approximately 45° of body surface on either side of the fin. A periodic boundary condition was used to take advantage of this symmetry. This helped to reduce the number of grid cells needed for the computation, but it restricted the angle of attack to 0°. The calculation of the roll moment coefficient was the primary goal in this initial effort. For this reason, the flow field was computed for the body and fins only, since the influence of the recirculating flow at the base on the roll moment coefficient is expected to be small. Future computations which include flow field solutions at the projectile base will hopefully verify the accuracy of this assumption. A zero gradient boundary condition was used at the trailing edges of the fins for the downstream boundary condition. A nonreflecting boundary condition was applied to the outermost grid plane from the body surface. The

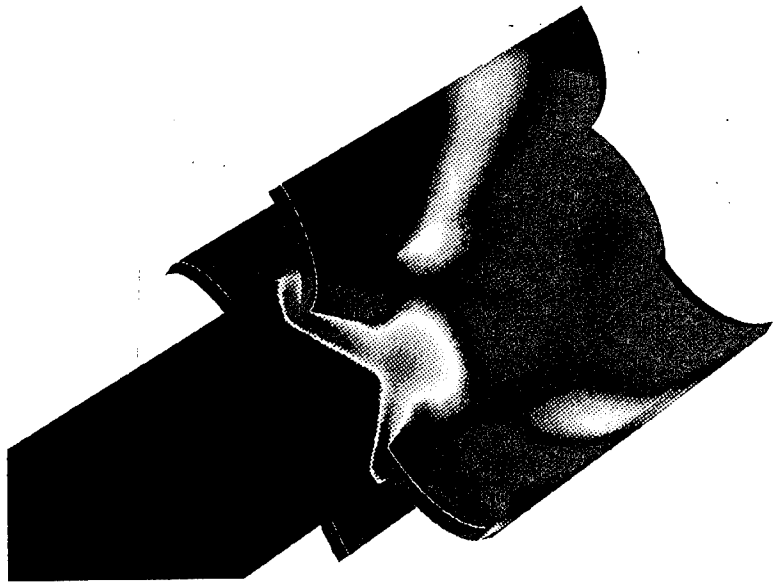

Figure 2. Pressure contours for wraparound fin projectile.

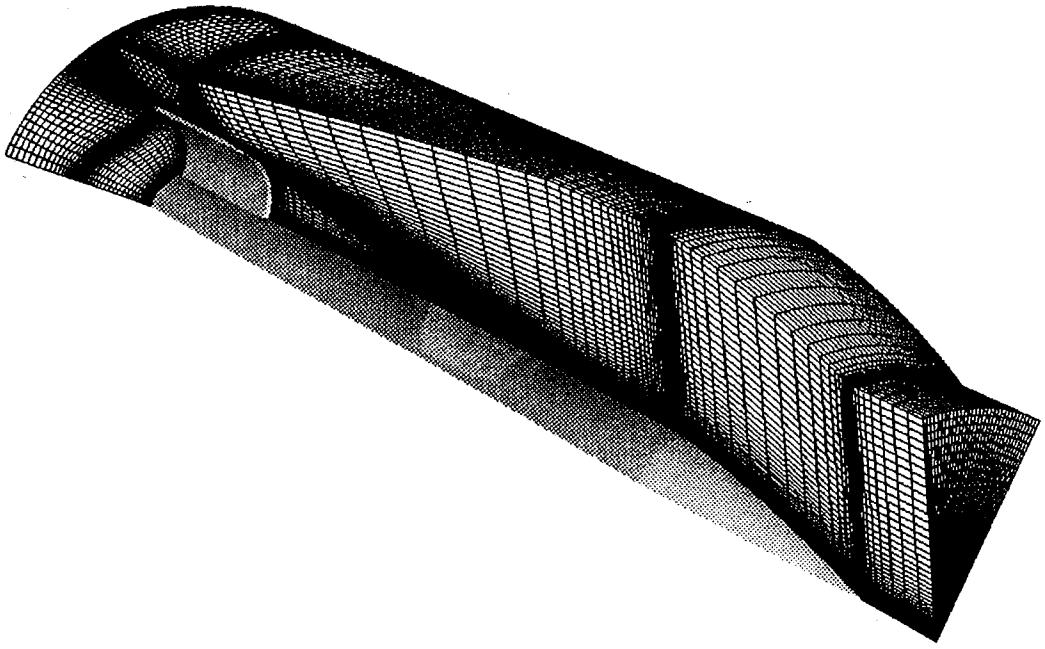

Figure 3. Computational grid.

nonreflecting boundary condition allowed the outermost grid plane to be placed relatively close to the body, which reduced the number of computational points needed for a solution. In order to obtain a viscous solution, the fin and body surfaces were modeled with a no-slip boundary condition.

Figure 4 shows the computational model of the wraparound fin projectile. One-fourth of the projectile surface is shown, along with the outline of the five grid zones comprising the computational model. It is easy to see how the zonal grid blocks allow the accurate modeling of the fin. As mentioned previously, information is passed between the individual zones to advance the solution to convergence. The zones in Figure 4 differ not only in physical size but also in the number of grid points. The dimensions of each grid zone are (24×96×80), (64×64×32), (64×64×32), (64×34×80), and (96×96×80). Computational models often require the use of zones in which the number of points vary. In total, approximately 1.4 million grid points were used for the computation. The same grid was used for the computations on each computer mentioned.

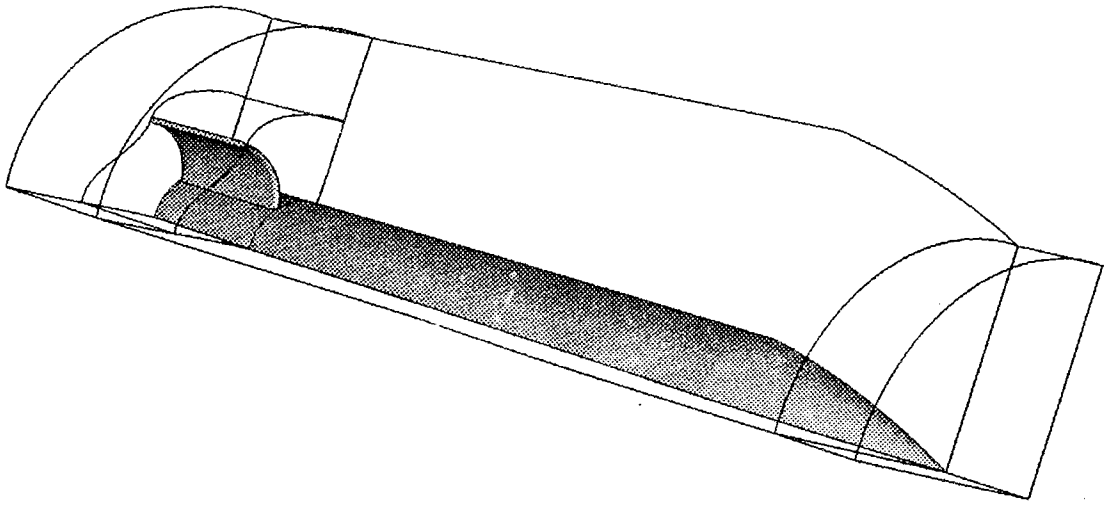

Figure 4. Computational model.

The solutions of the flow field were used to compute the aerodynamic forces of the projectile. Figure 5 shows the roll moment coefficient computed from flow field solutions obtained on the CM-5. The comparison with the experimental data is quite good. This is an indication that CFD can be used to predict the aerodynamics of complex fin shapes.

The aforementioned wraparound fin projectile application was implemented on four computers—the Cray C-90, CM-200, CM-5, and KSR-1. Being a shared-memory machine, performance on the C-90 is

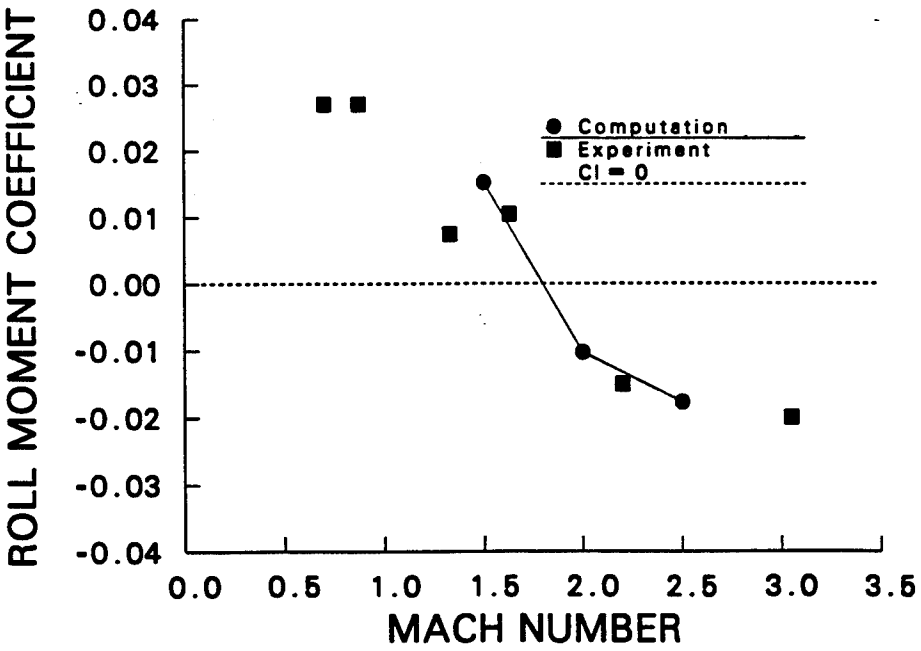

Figure 5. Roll moment coefficient vs. Mach number.

a good basis for comparison. The results of the application are based on the same grid size for the four machines.

The C-90 performance was measured for a single processor. Since the C-90 is a shared-memory machine, no communication overhead is present. The code is highly vectorized, and its performance on a shared-memory machine has been documented [4, 8]. The CM-5 code was rewritten using CM FORTRAN as mentioned previously. The implementation retained the original boundary conditions in the modular blocks. None of the boundary conditions was simplified to obtain favorable performance. The boundary conditions, including interzone data transfer, represent irregular communication. While the kernel, which represents computation at all interior field points, involves regular communication because of the structured grid topology. Of the total time on the CM-5, more than 50% was spent on boundary conditions, which incudes zonal data exchange at the boundaries. This is not surprising. Even though boundary conditions involve very few grid points, and substantially less computation compared to interior field points, they represent a large portion of the communication overhead. Thus, the communication overhead is a critical element for real-world applications. This is quite different from applications with boundary effects favorable for benchmarking, which indicate superlinear speedup which could be misleading. On the CM-5, the application was run to completion. The CM-5 is being used for routine computations using the code. The high-speed parallel I/O, based on a scalable disk array, was found to be adequate for running an application in which solution files and restart files were stored.

The performance on the KSR-1 represents performance of the code kernel only for the wraparound fin projectile. It does not include boundary conditions and interzone data transfer for this application. The code was restructured considerably. This included coalescing many 3-D arrays into fewer four-dimensional (4-D) arrays and restructuring the calls to subroutines. All this was done to keep data local in the subcache and cache [9]. A superlinear speedup was reported for a National Aeronautics and Space Administration (NASA) benchmark kernel using similar techniques [10]. The difference is that this application is larger in size, has many more arrays, and includes multiple zones. Out of 100 s of CPU time, the CPU was starving for data about 50 s. In other words, the CPU was simply waiting for data to be available to that cell. This is likely to occur when a given code section has many large arrays to work with, arrays have a stride in indices in more than one dimension, a cache is not big enough, or data requirements are small relative to the 16-word subpage size. Often, many subpage swaps are required before all needed data become available in the local cache. Part of the cache thrashing problem can be addressed by padding array dimensions, but this should be done on a case-by-case basis. This is undesirable for parametric studies of the same application in which grid size and array dimensions in the code may change. The I/O of the KSR-1 was found to be much slower than the parallel I/O of the CM-5 and CM-200.

Neither of the code versions was fine-tuned for the aforementioned application. Thus, the results presented do not necessarily represent the best possible performance for the given grid size. Nevertheless, they represent realistic expected performance in the application field. At the time these results were acquired, both Kendall Square Research Corporation and Thinking Machine Corporation were developing software for their respective systems. Also, as a requirement for use of the CM-200 and CM-5, the following disclaimer must be issued when presenting results of their performance:

> *"These results are based upon a beta version of the software and, consequently, are not necessarily representative of the performance of the full version of this software."*

Table 1 lists timings for the C-90, CM-200, CM-5, and KSR-1 computers for the wraparound fin application discussed previously. In general, the performance of the CM-5 was found to be on par with the C-90. The performance of the CM-200 was comparable to the performance of the CM-5 and C-90, while the performance of the KSR-1 was disappointing for this application. Table 1 shows that for the full application, meaning full use of boundary conditions and zonal data exchange, the performance of the CM-5 using 256 processing nodes was nearly identical to the performance of the C-90 using 1 processing

node. The performance is measured in iterations per minute, where an iteration is defined as a computation for all the zones. The KSR-1 was a kernel-only computation. The poor timing for the KSR-1 reflects the problems discussed previously.

Table 1. Computer Timings for the Wraparound Fin Application

Computer	Time (iterations/minute)	Processing Units Used	Notes
C-90	6.3	1	full application
CM-5	6.3	256	full application
CM-200	2.0	16k	full application
KSR-1	0.55	64	kernel only

7. SUMMARY

The focus of this paper was to investigate issues related to the implementation of a large-scale real-world application on distributed memory systems—the CM-200, CM-5, and KSR-1. Results of the computation of 3-D flow about the M549 projectile employing three zones and a wraparound fin projectile using 1.4 million grid points and five zones were presented. The performance of the entire wraparound fin application on the CM-200 and CM-5 along with the performance of the code kernel on the KSR-1 were compared with the performance of the full application on the Cray C-90. The CM-5 turned out to be the MPP system of choice and is being used routinely for similar applications and parametric studies at ARL. The performance of the CM-200 varied greatly with changes in grid dimensions. CM-5 performance may also vary with changes in grid dimensions. However, large differences in performance on the CM-5 due to grid dimension changes have not been found for the applications described previously. On the KSR-1, differences in performance can be obtained by padding dimensions. When emphasis is placed on sustainable performance for large-scale applications, implementation is much more involved on the CM-200, CM-5, and KSR-1. The allcache memory architecture of the KSR-1 did not perform as well for this large-scale application as reported for similar small-scale problems. The KSR-1 performance is tied to the issue of how often a given processor must fetch from the cache to subcache and from the local cache of other processors to obtain required data. Thus, the communication issue is not necessarily addressed effectively for applications using dynamic address binding and dynamic data movement of the allcache memory.

The present implementation underscores the important issues related to scalability. When a parallel kernel of code is scaled-up with favorable boundary effects, its performance seems to scale-up almost linearly. This could be misleading for a practitioner in the application field because it masks the substantial communication and synchronization overhead associated with realistic applications, such as multiple-zonal computations. Finally, the speed of I/O is also an important element for practical usefulness of distributed memory systems for large-scale applications.

8. REFERENCES

[1] Thinking Machines Corporation. "CM-200 Technical Summary." Cambridge, MA, June 1991.

[2] Thinking Machines Corporation. "The Connection Machine System." Cambridge, MA, September 1992.

[3] Kendall Square Research Corporation. "KSR Parallel Programming." Waltham, MA, February 1992.

[4] Patel, N. R., W. B. Sturek, and G. A. Smith. "Parallel Computation of Supersonic Flows Using a Three-Dimensional, Zonal, Navier-Stokes Code." BRL-TR-3049, U.S. Army Ballistic Research Laboratory, Aberdeen Proving Ground, MD, November 1989.

[5] Thinking Machines Corporation. "CM FORTRAN Programming." Cambridge, MA, November 1991.

[6] Kendall Square Research Corporation. "KSR FORTRAN Programming." Waltham, MA, February 1992.

[7] Edge, H. L. "Computation of the Roll Moment Coefficient for a Projectile With Wrap-Around Fins." ARL-TR-23, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, December 1992.

[8] Patel, N. R., W. B. Sturek, and R. E. Hiromoto. "A Parallel Compressible Flow Algorithm for Multiprocessors." Application of Parallel Processing in Fluid Mechanics, Fluids Engineering Division, vol. 47, ASME, June 1987.

[9] Patel, N. "Implementation and Performance Experience With a 3-D Multi-Block Flow Solver on KSR 1." U.S. Army Research Laboratory Symposium on Supercomputing Proceedings, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, September 1993.

[10] Breit, S., and D. Zirl. "Technical Applications on the KSR-1: High Performance and Ease of Use." U.S. Army Research Laboratory Symposium on Supercomputing Proceedings, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, September 1993.

INTENTIONALLY LEFT BLANK.

NO. OF COPIES	ORGANIZATION
2	ADMINISTRATOR ATTN DTIC DDA DEFENSE TECHNICAL INFO CTR CAMERON STATION ALEXANDRIA VA 22304-6145
1	DIRECTOR ATTN AMSRL OP SD TA US ARMY RESEARCH LAB 2800 POWDER MILL RD ADELPHI MD 20783-1145
3	DIRECTOR ATTN AMSRL OP SD TL US ARMY RESEARCH LAB 2800 POWDER MILL RD ADELPHI MD 20783-1145
1	DIRECTOR ATTN AMSRL OP SD TP US ARMY RESEARCH LAB 2800 POWDER MILL RD ADELPHI MD 20783-1145
	ABERDEEN PROVING GROUND
5	DIR USARL ATTN AMSRL OP AP L (305)

NO. OF
COPIES ORGANIZATION

ABERDEEN PROVING GROUND

23 DIR USARL
ATTN AMSRL WT
DR I MAY
DR A BARROWS
AMSRL WT PA
DR T MINOR
AMSRL WT PB
DR E SCHMIDT
DR A MIKHAIL
DR J SAHU
MR B GUIDOS
MR P WEINACHT
MR H EDGE
MR E FERRY
DR P PLOSTINS
MR C NIETUBICZ
AMSRL WT PD
DR B BURNS
AMSRL WT PE
MR A HORST
AMSRL WT W
DR C MURPHY
AMSRL WT WB
DR W D'AMICO
AMSRL CI
DR W MERMAGEN
AMSRL CI AD
MS D HISLEY
AMSRL CI CA
DR N PATEL
AMSRL CI C
DR W STUREK
AMSRL CI CC
MS B BROOME
AMSRL CI S
DR A MARK
AMSRL SF IB
MR J CLARKE

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number ARL-TR-712 Date of Report February 1995

2. Date Report Received _____

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) _____

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) _____

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. _____

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) _____

CURRENT ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

_____
Organization

_____
Name

_____
Street or P.O. Box No.

_____
City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)

(DO NOT STAPLE)