NASA/CR—1999-208688

Shared Memory Parallelization of an Implicit ADI-type CFD Code

Th. Hauser and P.G. Huang
University of Kentucky, Lexington, Kentucky

February 1999

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized data bases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at (301) 621-0134

- Telephone the NASA Access Help Desk at (301) 621-0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/CR—1999-208688

Shared Memory Parallelization of an Implicit ADI-type CFD Code

Th. Hauser and P.G. Huang
University of Kentucky, Lexington, Kentucky

Prepared under Grant NAG3–2099

National Aeronautics and
Space Administration

Lewis Research Center

February 1999

Acknowledgments

The current project is supported by NASA Lewis under grant number NAG3–2099, with Dr. David Ashpis as the technical monitor. NAS and NCSA provide the computer time for the SGI Origin 2000 computers. A special thanks goes to Dr. James Taft at NASA Ames for his advice on refining the LESTool code for serial performance.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076
Price Code: A03

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100
Price Code: A03

Shared Memory Parallelization of an implicit ADI-type CFD code

Th. Hauser and P.G. Huang
Department of Mechanical Engineering,
University of Kentucky,
Lexington, Kentucky 40506, USA

Abstract

A parallelization study designed for ADI-type algorithms is presented using the OpenMP specification for shared-memory multiprocessor programming. Details of optimizations specifically addressed to cache-based computer architectures are described and performance measurements for the single and multiprocessor implementation are summarized. The paper demonstrates that optimization of memory access on a cache-based computer architecture controls the performance of the computational algorithm. A hybrid MPI/OpenMP approach is proposed for clusters of shared memory machines to further enhance the parallel performance. The method is applied to develop a new LES/DNS code, named LESTool. A preliminary DNS calculation of a fully developed channel flow at a Reynolds number of 180, $Re_\tau = 180$, has shown good agreement with existing data.

NASA/CR—1998-208688

1 Introduction

The rapid growth of computer hardware and software has made it possible for CFD to evolve into routine design tools for practical engineering applications in the 21st century. It is expected that LES and DNS will become standard practice in the CFD methodology. However, due to the large computer resources demanded by LES and DNS in practical engineering applications, the speed of paradigm shift not only hinges upon the advancement of new computer hardware and software, but also depends on new CFD algorithms taking advantage of the new computer hardware and software developments.

The recent advancement in cache-based Shared Memory Multiprocessor (SMP) architectures [4], such as Origin 2000 and HP Exemplar, has provided an easy transition of a serial CFD programming style to parallel environments. The shared memory approach using OpenMP [6] simplifies the implementation of ADI type algorithms compared to the distributed programming model using MPI [5]. With OpenMP, the algorithm can be parallelized along lines in the computational domain without specifying data movement, which is necessary in a distributed environment. In addition, the OpenMP model provides an incremental path to a parallel program. This approach is much more efficient than the distributed model, which requires the program's data structures to be explicitly partitioned and hence the entire application must be parallelized according to the partitioned data structures.

The main objective of this paper is to describe the parallel implementation of ADI-type Navier-Stokes solvers on cache-based shared memory parallel computers using OpenMP directives. An overview of the implementation of ADI-type algorithms is presented and performance results based on the SGI Origin 2000 are reported. A number of key features needed to improve the speed of the memory access are highlighted. These improvements in memory access have led to a high floating point performance in our application code (LESTool), which is designed for LES and DNS of turbulent flow using high-order discretization schemes.

2 Governing Equations

The fluid motion is governed by the time-dependent Navier-Stokes equations for an ideal gas which express the conservation of mass, momentum, and energy for a compressible Newtonian fluid. The equations written in curvilinear

coordinates are:

$$\frac{\partial Q}{\partial t} + \frac{\partial F_\xi}{\partial \xi} + \frac{\partial F_\eta}{\partial \eta} + \frac{\partial F_\zeta}{\partial \zeta} = \frac{\partial G_\xi}{\partial \xi} + \frac{\partial G_\eta}{\partial \eta} + \frac{\partial G_\zeta}{\partial \zeta} \tag{1}$$

where Q is the vector of the conservative variables multiplied by the volume of the computational cell, V, and is defined by

$$Q = V\left(\rho,\, \rho u,\, \rho v,\, \rho w,\, \rho E\right)^T. \tag{2}$$

The F's and G's denote the convective and viscous fluxes, respectively, and the subscripts ξ, η and ζ represent the directions of the fluxes. The inviscid fluxes are given by

$$F_\xi = \begin{pmatrix} \rho U_\xi \\ \rho u U_\xi + pV\,\partial\xi/\partial x \\ \rho v U_\xi + pV\,\partial\xi/\partial y \\ \rho w U_\xi + pV\,\partial\xi/\partial z \\ \rho(E + p/\rho)U_\xi - pV\,\partial\xi/\partial t \end{pmatrix}, \tag{3}$$

$$F_\eta = \begin{pmatrix} \rho U_\eta \\ \rho u U_\eta + pV\,\partial\eta/\partial x \\ \rho v U_\eta + pV\,\partial\eta/\partial y \\ \rho w U_\eta + pV\,\partial\eta/\partial z \\ \rho(E + p/\rho)U_\eta - pV\,\partial\eta/\partial t \end{pmatrix}, \tag{4}$$

$$F_\zeta = \begin{pmatrix} \rho U_\zeta \\ \rho u U_\zeta + pV\,\partial\zeta/\partial x \\ \rho v U_\zeta + pV\,\partial\zeta/\partial y \\ \rho w U_\zeta + pV\,\partial\zeta/\partial z \\ \rho(E + p/\rho)U_\zeta - pV\,\partial\zeta/\partial t \end{pmatrix}, \tag{5}$$

where ρ is the density, p is the pressure, u, v, w are the cartesian velocity components, $E = e + K$ is the total energy per unit volume and equal to the sum of the internal energy, e, and the kinetic energy, $K = (u^2 + v^2 + w^2)/2$, and the U's are the contravariant velocity components defined by

$$
\begin{aligned}
U_\xi &= V(\frac{\partial \xi}{\partial t} + \frac{\partial \xi}{\partial x}u + \frac{\partial \xi}{\partial y}v + \frac{\partial \xi}{\partial z}w) \\
U_\eta &= V(\frac{\partial \eta}{\partial t} + \frac{\partial \eta}{\partial x}u + \frac{\partial \eta}{\partial y}v + \frac{\partial \eta}{\partial z}w) \\
U_\zeta &= V(\frac{\partial \zeta}{\partial t} + \frac{\partial \zeta}{\partial x}u + \frac{\partial \zeta}{\partial y}v + \frac{\partial \zeta}{\partial z}w)
\end{aligned} \tag{6}
$$

The viscous fluxes in the ξ, η and ζ directions are given by

$$
G_\xi = V \begin{pmatrix} 0 \\ \tau_{xx}\,\partial\xi/\partial x + \tau_{xy}\,\partial\xi/\partial y + \tau_{xz}\,\partial\xi/\partial z \\ \tau_{xy}\,\partial\xi/\partial x + \tau_{yy}\,\partial\xi/\partial y + \tau_{yz}\,\partial\xi/\partial z \\ \tau_{xz}\,\partial\xi/\partial x + \tau_{yz}\,\partial\xi/\partial y + \tau_{zz}\,\partial\xi/\partial z \\ g_1\,\partial\xi/\partial x + g_2\,\partial\xi/\partial y + g_3\,\partial\xi/\partial z \end{pmatrix}, \tag{7}
$$

$$
G_\eta = V \begin{pmatrix} 0 \\ \tau_{xx}\,\partial\eta/\partial x + \tau_{xy}\,\partial\eta/\partial y + \tau_{xz}\,\partial\eta/\partial z \\ \tau_{xy}\,\partial\eta/\partial x + \tau_{yy}\,\partial\eta/\partial y + \tau_{yz}\,\partial\eta/\partial z \\ \tau_{xz}\,\partial\eta/\partial x + \tau_{yz}\,\partial\eta/\partial y + \tau_{zz}\,\partial\eta/\partial z \\ g_1\,\partial\eta/\partial x + g_2\,\partial\eta/\partial y + g_3\,\partial\eta/\partial z \end{pmatrix}, \tag{8}
$$

$$
G_\zeta = V \begin{pmatrix} 0 \\ \tau_{xx}\,\partial\zeta/\partial x + \tau_{xy}\,\partial\zeta/\partial y + \tau_{xz}\,\partial\zeta/\partial z \\ \tau_{xy}\,\partial\zeta/\partial x + \tau_{yy}\,\partial\zeta/\partial y + \tau_{yz}\,\partial\zeta/\partial z \\ \tau_{xz}\,\partial\zeta/\partial x + \tau_{yz}\,\partial\zeta/\partial y + \tau_{zz}\,\partial\zeta/\partial z \\ g_1\,\partial\zeta/\partial x + g_2\,\partial\zeta/\partial y + g_3\,\partial\zeta/\partial z \end{pmatrix}, \tag{9}
$$

where

$$
\begin{aligned}
g_1 &= u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x\,, \\
g_2 &= u\tau_{xy} + v\tau_{yy} + w\tau_{yz} - q_y\,, \\
g_3 &= u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z\,.
\end{aligned} \tag{10}
$$

The stress tensor τ and the heat flux vector q are defined by

$$\tau_{xx} = \mu\left(2\frac{\partial u}{\partial x} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)\right), \tag{11}$$

$$\tau_{yy} = \mu\left(2\frac{\partial u}{\partial y} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)\right), \tag{12}$$

$$\tau_{zz} = \mu\left(2\frac{\partial u}{\partial z} - \frac{2}{3}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z}\right)\right), \tag{13}$$

$$\tau_{xy} = \mu\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right), \tag{14}$$

$$\tau_{xz} = \mu\left(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}\right), \tag{15}$$

$$\tau_{yz} = \mu\left(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}\right), \tag{16}$$

and

$$q_x = -k\frac{\partial T}{\partial x}, \tag{17}$$

$$q_y = -k\frac{\partial T}{\partial y}, \tag{18}$$

$$q_z = -k\frac{\partial T}{\partial z}, \tag{19}$$

where μ and k are molecular viscosity and thermal conductivity, respectively.

The gradients in the stresses and heat fluxes of any dependent variable, ϕ, where ϕ can be u, v, w or T, are computed as described in appendix A.3.

The pressure is related to the density and temperature, T, according to the equation of state

$$p = (\gamma - 1)(E - \rho K) \tag{20}$$

where γ is the ratio of the specific heats. The coefficient of viscosity, μ, and thermal conductivity, k, are related by the constant Prandl number Pr.

$$k = \frac{\mu c_p}{Pr} \tag{21}$$

3 Numerical method

By defining the advancement of the solution from time level n to $n+1$ as

$$Q^{n+1} = Q^n + \Delta Q^n, \tag{22}$$

the implicit, second-order, three-point-backward approximation in time for the solution of (1) can be expressed as

$$\begin{aligned} \frac{3}{2}\frac{\Delta Q^n}{\Delta t} - \frac{1}{2}\frac{\Delta Q^{n-1}}{\Delta t} &= -\left[\frac{\partial}{\partial \xi}(F_\xi^{n+1} - G_\xi^{n+1}) + \right. \\ &\left. \frac{\partial}{\partial \eta}(F_\eta^{n+1} - G_\eta^{n+1}) + \frac{\partial}{\partial \zeta}(F_\zeta^{n+1} - G_\zeta^{n+1})\right]. \end{aligned} \tag{23}$$

The inviscid and viscous fluxes F and G are linearized by a Taylor series expansion

$$\begin{aligned} F_\chi^{n+1} &= F_\chi^n + A_\chi \Delta Q^n \\ G_\chi^{n+1} &= G_\chi^n + B_\chi \Delta Q^n \end{aligned} \tag{24}$$

where χ can be ξ, η or ζ; A_χ and B_χ are the Jacobian matrices for the inviscid and viscous fluxes and are defined in the appendices A.1 and A.2.

By substituting the linearization (24) into equation (23) one yields

$$\begin{aligned} \frac{3}{2}\frac{\Delta Q^n}{\Delta t} + \left[\frac{\partial}{\partial \xi}(A_\xi - B_\xi)\Delta Q^n + \frac{\partial}{\partial \eta}(A_\eta - B_\eta)\Delta Q^n \right. \\ \left. + \frac{\partial}{\partial \zeta}(A_\zeta - B_\zeta)\Delta Q^n\right] &= \Delta R^n \end{aligned} \tag{25}$$

where ΔR^n is evaluated explicitly from the previous time level, n, according to

$$\begin{aligned} \Delta R^n = -\left[\frac{\partial}{\partial \xi}(F_\xi^n - G_\xi^n) + \frac{\partial}{\partial \eta}(F_\eta^n - G_\eta^n) + \frac{\partial}{\partial \zeta}(F_\zeta^n - G_\zeta^n)\right] \\ + \frac{1}{2}\frac{\Delta Q^{n-1}}{\Delta t}. \end{aligned} \tag{26}$$

By further defining a delta-delta variable $\Delta' Q^{n,m}$ such that

$$\Delta Q^{n,m+1} = \Delta Q^{n,m} + \Delta' Q^{n,m} \tag{27}$$
$$Q^{n,m+1} = Q^n + \Delta Q^{n,m+1} \tag{28}$$

where m is the counter for the inner iteration and $Q^{n,m+1}$ and $\Delta Q^{n,m+1}$ become Q^{n+1} and ΔQ^n, respectively when the solution for the inner loop converges, equation (25) can be written in delta-delta form as:

$$\frac{3}{2}\frac{\Delta' Q^{n,m}}{\Delta t} + \frac{\partial}{\partial \xi}(A_\xi - B_\xi)\Delta' Q^{n,m} + \frac{\partial}{\partial \eta}(A_\eta - B_\eta)\Delta' Q^{n,m} + \frac{\partial}{\partial \zeta}(A_\zeta - B_\zeta)\Delta' Q^{n,m} = \Delta R^{n,m} \tag{29}$$

where $\Delta R^{n,m}$ is exactly the net residual of the equation (23) and can be evaluated explicitly by

$$\Delta R^{n,m} = -\frac{3}{2}\frac{\Delta Q^{n,m}}{\Delta t} - \frac{\partial}{\partial \xi}(F_\xi^{n,m} - G_\xi^{n,m}) + \frac{\partial}{\partial \eta}(F_\eta^{n,m} - G_\eta^{n,m}) + \frac{\partial}{\partial \zeta}(F_\zeta^{n,m} - G_\zeta^{n,m}) + \frac{1}{2}\frac{\Delta Q^{n-1}}{\Delta t}. \tag{30}$$

Equation (29) is similar to the iterative method proposed in [8]. Note that, when the inner solution converges, $\Delta R^{n,m} = 0$ and thus equation (23) is satisfied. The exact form of the LHS of (29) is thus not crucial to the solution of (23). Hence, high-order approximations are applied to evaluate ΔR while stable low-order differencing schemes are used to approximate the LHS of (29).

In the RHS (30), a fifth-order upwinding scheme is applied to evaluate the interfacial inviscid fluxes, F, while the sixth-order central differencing scheme is applied to approximate the interfacial viscous fluxes, G. Other differencing schemes, such as high-order Pade [3], ENO [9] and B-splining [2], were also proposed and are currently under test. A first-order upwind differencing approximation and a second-order central differencing scheme were applied in the LHS of (29) to ensure numerical stability.

The resulting equation in finite volume form is given by:

$$[D + \delta_\xi(A_\xi - B_\xi) + \delta_\eta(A_\eta - B_\eta) + \delta_\zeta(A_\zeta - B_\zeta)]\,\Delta' Q = \Delta R \qquad (31)$$

where δ is the finite volume operator and $D = \frac{3}{2} I/\Delta t$.
The diagonal matrix D can be extracted from equation (31)

$$\begin{aligned} D\left[I + D^{-1}\delta_\xi(A_\xi - B_\xi) + D^{-1}\delta_\eta(A_\eta - B_\eta)\right. & \qquad (32) \\ \left. + D^{-1}\delta_\zeta(A_\zeta - B_\zeta)\right]\Delta' Q &= \Delta R. \end{aligned}$$

After factoring and reexpressing one obtains

$$\begin{aligned} D\left[I + D^{-1}\delta_\xi(A_\xi - B_\xi)\right] D^{-1} D\left[I + D^{-1}\delta_\eta(A_\eta - B_\eta)\right] D^{-1} & \qquad (33) \\ D\left[I + D^{-1}\delta_\zeta(A_\zeta - B_\zeta)\right]\Delta' Q = \Delta R & \end{aligned}$$

Equation (33) can be written as an ADI algorithm

$$\begin{aligned} [D + \delta_\xi(A_\xi - B_\xi)]\,\Delta' Q^1 &= \Delta R \\ [D + \delta_\eta(A_\eta - B_\eta)]\,\Delta' Q^2 &= D\Delta' Q^1 \qquad (34) \\ [D + \delta_\zeta(A_\zeta - B_\zeta)]\,\Delta' Q^3 &= D\Delta' Q^2 \end{aligned}$$

The solution of equation (34) involves the inversion of 5×5 block tridiagonal matrices for all three directions.

To reduce computational costs, equation (34) can be cast into the diagonal form of approximate factorization following the approach of Pulliam and Chaussee [7].

$$\begin{aligned} R_\xi^{-1}\,[D + \delta_\xi(\Lambda_\xi - r(B_\xi))]\,R_\xi \Delta' Q^1 &= \Delta R \\ R_\eta^{-1}\,[D + \delta_\eta(\Lambda_\eta - r(B_\eta))]\,R_\eta \Delta' Q^2 &= D\Delta' Q^1 \qquad (35) \\ R_\zeta^{-1}\,[D + \delta_\zeta(\Lambda_\zeta - r(B_\zeta))]\,R_\zeta \Delta' Q^3 &= D\Delta' Q^2 \end{aligned}$$

or

$$\begin{aligned} [D + \delta_\xi(\Lambda_\xi - r(B_\xi))]\,R_\xi \Delta' Q^1 &= R_\xi \Delta R \\ [D + \delta_\eta(\Lambda_\eta - r(B_\eta))]\,R_\eta \Delta' Q^2 &= R_\eta D\Delta' Q^1 \qquad (36) \\ [D + \delta_\zeta(\Lambda_\zeta - r(B_\zeta))]\,R_\zeta \Delta' Q^3 &= R_\zeta D\Delta' Q^2 \end{aligned}$$

where the Λ's and R's are the eigenvalue and eigenvector matrices of the convective Jacobian, respectively, defined in the appendix by equation (38) and (42). The viscous Jacobian matrices are not diagonalized with the same eigenvector matrices used for the inviscid Jacobian. Therefore, they are approximated by their spectral radii, function $r(B) = \max(diagonal(B))$, where B is defined in the appendix A.2. The advantage of (35) is that it involves only the inversion of three scalar tridiagonal matrices, compared to the original form, (34), which requires the inversion of three block tridiagonal matrices.

In contrast to the diagonal ADI method, (35), the block ADI method treats the viscous Jacobian matrices in a more exact manner and the method allows the use of implicit boundary conditions. As a result, the solution of (29) based on the block ADI method, (34), is more stable and the method allows a larger time step to be used in the calculations. On the other hand, the diagonal ADI method, (35), reduces the computational effort per timestep by a factor of 5. Therefore, the current code allows a combination of the diagonal and block ADI in different directions. For example, if the wall is normal to the η-direction, one can apply the block ADI only in the η-direction while the diagonal ADI is used in the other two directions.

4 Parallel Implementation

4.1 Algorithmic Design and Memory Management

Since the operators δ_ξ, δ_η, and δ_ζ are one-dimensional operators and only dependent on variables in their corresponding directions, the solution algorithm for the right-hand side, as well as the ADI-type left-hand side, is designed in such a way that data along the i, j and k-directions (corresponding to the ξ, η and ζ-directions, respectively) is first copied into one-dimensional scratch arrays. Then a directional independent module is called to perform the operations using the information provided by these one-dimensional arrays. Once the operations are completed, the resulting data are copied back into the three-dimensional arrays. Figure 1 shows the copying of the data into scratch arrays in the i and j-directions. This implementation not only provides a simplification in the programming style but also ensures a continuous data flow when evaluating the LHS and RHS operators. This results in primary- and secondary-cache hit rates of 98% and 97%, respectively, on

a single-processor Origin 2000 using 120^3 grid points. The array assignment from 3-D to 1-D arrays and vice-versa can be easily implemented using the new Fortran 90 array-section feature to be discussed in the next section.

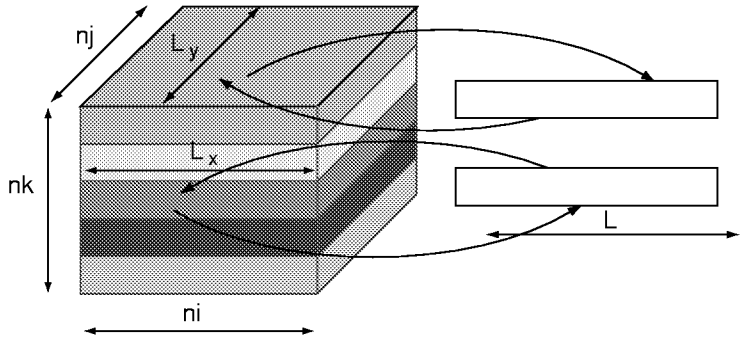

Figure 1: Schematic data transfer of 3D arrays into 1D scratch arrays

The data management illustrated in figure 1 ensures high cache-hit rates but the serial performance on the Origin 2000 for a 64^3 test case showed only 60 MFLOP/s (peak 380 MFLOP/s). Even though the average performance of applications on NAS computers is reported to be 40-50 MFLOP/s, we considered this number unsatisfactory for performing large-scale DNS and LES simulations. To increase the memory bandwidth, all arithmetic operations involving the 5x5 block matrices, used in block tridiagonal and periodic block tridiagonal solvers, were unrolled. This resulted in a much longer and less desirable code, but the performance results outweigh this disadvantage. As shown in table 1, the unrolling of all the 5x5 matrix operations gives rise to a nearly tripled memory bandwidth and a doubled bandwidth for the L1 and L2 caches, respectively. As a result, the MFLOP rate on a single processor increased to 120, a number we consider a satisfactory performance on the Origin 2000.

4.2 Benefits of Using Fortran 90

Fortran 90 was chosen as the programming language for the LESTool code. A number of new language features compared to Fortran 77 are found to be very useful for the development of a efficient, portable and maintainable program. Some of these features are highlighted below.

Table 1: Optimization on a single processor (64^3 grid points)

	no unrolling	unrolling
MFLOP/S	60	119
CPU-time (s)	703	314
primary cache hit rate	0.985	0.961
secondary cache hit rate	0.967	0.947
memory bandwidth used (MB/s)	9.9	28.49
L1 - L2 bandwidth used (MB/s)	80.07	157.84

1. Control of precision: Because the current code is intended for 3-D multi-block general coordinates using high-order schemes, the numerical precision of program variables is important to reduce the main memory usage of the code. For example, high precision is needed for field quantities while lower precision can be used for the geometric quantities, such as surface area vectors. Moreover, for the weighting of the high-order interpolation functions, the use of a short integer, which provides a precision up to 4 significant digits, may be sufficient. Fortran 90 provides a convenient and portable way for such a precision control, as shown below.

```
 MODULE kind_spec_module
  IMPLICIT NONE
  INTEGER, PARAMETER :: high = SELECTED_REAL_KIND(15,307)
  INTEGER, PARAMETER :: low  = SELECTED_REAL_KIND(6,32)
  INTEGER, PARAMETER :: short = SELECTED_INT_KIND(4)
  INTEGER, PARAMETER :: long  = SELECTED_INT_KIND(9)
END MODULE kind_spec_module
```

2. Modules: As already shown in the previous example, the module syntax provides a mean to package global parameters, derived types and associated operations, and to access them wherever needed. This enables a more maintainable programming style.

3. Dynamic memory control: Memory may be allocated and deallocated on demand in Fortran 90. This results in a more flexible implementation that will respond to changes of grid size, as shown in the example below.

```
REAL(HIGH), DIMENSION(:, :, :), ALLOCATABLE :: x, y, z
READ(1) ni, nj, nk
ALLOCATE(x(ni, nj, nk), y(ni, nj, nk), z(ni, nj, nk))
DEALLOCATE(x, y, z)
```

4. Pointer variables: Pointers will enable the definition of aliases to different memory regions. As can be seen from the example below, all dependent variables are grouped together with the leading dimension being the number of variables. This arrangement is preferred by the cache-memory architecture while pointer variables provide precise names to access a single variable inside this memory block.

```
ALLOCATE(variables(5, ni, nj, nk))
rho  => variables(1, :, :, :)
rhou => variables(2, :, :, :)
```

5. Array syntax: As discussed in section 2.1, this feature simplifies the programming of the ADI-algorithm, and results in a source code which is short and easy to comprehend. For example, the copying operation for the k-direction of the three-dimensional variables to the one-dimensional scratch array is depicted below.

```
rho_1d(:) = rho_3d(i, j, :)
```

6. Derived types: This feature provides a mean to group related data together, as shown in the next example. Here we reduce the storage needed for the cell-face vectors by just storing the magnitude in a low-precision floating point number and the directional cosines for each vector component in three, short integers.

```
TYPE, public :: storage
   REAL(low) :: magnitude
   INTEGER(short), DIMENSION(3) :: vector
END TYPE storage
!
TYPE(storage), DIMENSION(ni, nj, nk) :: normal_vector
```

7. Rich set of intrinsic functions: In addition to the powerful array syntax, a number of new intrinsic functions, such as TRANSPOSE, MATMUL and DOT_PRODUCT, provide a convenient way for matrix and vector operations. For example, interpolating a variable ϕ_f to any order involves a dot product of the weighting and variable vectors:

$$\phi_f = \sum_{i=1}^{order} w_i \phi_i \tag{37}$$

In Fortran 90, the interpolation can be expressed very conveniently in an order-independent manner.

```
 rho_interface(i)  = DOT_PRODUCT(rho_1d(is:ie), &
weighting(:, i))
```

For the multiblock implementation, the new features of Fortran 90, such as abstract data types and generic programming, have been tested. This programming style enabled us to reveal the weaknesses of the different vendor compilers. It should be mentioned that the current version (7.2.1) of the SGI compiler has solved all our compiler-related difficulties.

4.3 Parallel Implementation using OpenMP

For the parallelization of the ADI-type solver, the recently developed OpenMP specification for programming shared-memory multiprocessors is used. SGI adopted the OpenMP standard for the ORIGIN series in the version 7.2.1 compiler and HP has promised that their implementation of the OpenMP specification will be released in the upcoming compiler. Because OpenMP is a portable and scalable shared-memory multiprocessing application program interface (API) that gives programmers a simple and flexible interface for developing parallel applications, it is our belief that it will become the equivalent of MPI, the standard for distributed memory programming.

The LESTool code was parallelized by placing OpenMP directives in the outer loop within the LHS and RHS operations. This involved decomposing the 3D problem into groups of 1D lines, with each group assigned to a dedicated processor. The efficiency of the parallel decomposition was enhanced by the use of the "first touch" policy, which is specific to the SGI Origin 2000. This implies that the memory allocated is physically placed in the processor

node that touches the memory location first. All large three dimensional blocks of memory, initialized in planes of constant k, were distributed into different nodes. This allows an easy parallelization for the i- and j-directions.

After finishing the computation in i- and j-directions, the solution in the k-direction is performed. On typical distributed memory-computers this presents a problem because the memory has been distributed in ij-planes and therefore no processor can access data along k-lines. In contrast, the solution in the k-direction poses no difficulty on a shared-memory computer. In the current approach the outer loop was chosen to be in the j-direction, and the 1-D partition of the ik-planes was parallelized.

5 Parallel Performance Results

The mesh used for this test problem (a direct numerical simulation of a fully developed turbulent channel) was 120^3 grid points resulting in a memory usage of 1.5 Gbytes. The large arrays from this problem size do not fit into the aggregate of the cache memories and the data is scattered across local memories in different processor nodes. The performance on a varying number of processors is illustrated by measuring the speedup and MFLOP rate for a computation of four time-steps, with 5 inner ADI iterations being performed at each time step.

The results in Fig. 2 show that the speedup scales reasonably well for up to 16 processors (approximately 10 times speed up) while a gradual flattening of the speedup is observed when using 32 processors (approximately 13 times speedup). The MFLOP rate per processors drops from 112 for a single process to 85 for 16 processors and to 75 for 32 processors. Although the MFLOP rate is still sufficiently high for 32 processors, the drop of the performance is related to longer memory access times. Although the memory access time can be further reduced by additional tuning of the code, we anticipate no major breakthrough. Instead, we propose to tackle this problem by developing a new hybrid MPI/OpenMP approach, which will be described in section 7.

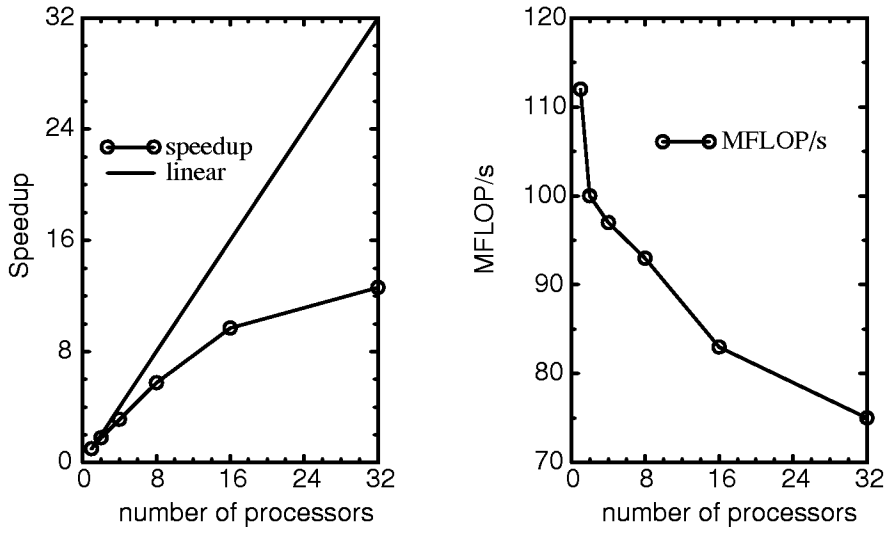

Figure 2: Speedup and MFLOP/s on the SGI ORIGIN 2000 (120^3 grid points)

6 DNS of fully developed turbulence in a channel

The results presented in this paper were based on the direct numerical simulation of a fully developed channel flow reported in [1]. The flow has two periodic (x and z) directions and the solid wall condition was imposed in the y direction. The Reynolds number based on the wall shear velocity, $u_\tau = \sqrt{\tau_w/\rho}$, is $Re_\tau = \rho_w u_\tau H/\mu_w = 180$, where H is half the channel width. The streamwise and spanwise dimensions of the channel are $4\pi H$ and $2\pi H$, respectively. The computation is carried out on a grid using 200x121x200 grid points in x, y, and z, respectively. The flow field is initialized with a laminar solution and random fluctuations are superimposed on the pressure field. The governing equations are then integrated in time until a statistical equilibrium is reached ($t\, u_\tau/H > 30$).

Figure 3 shows the dimensionless velocity, $u^+ = u/u_\tau$, plotted against dimensionless wall distance, $y^+ = y\, u_\tau/\nu$. Also shown in this figure is the comparison of the predicted mean velocity profile against the data cited in [1] for the same Reynolds number. The dotted line represents the desirable profile in the viscous sublayer, $u^+ = y^+$, and the dashed line denotes the

log-law line, $u^+ = \ln(y^+)/0.41 + 5.2$. The figure shows that the current mean velocity profile matches the data of Kim et al. [1] very well. A detailed comparison of the DNS statistics of turbulent quantities will be presented elsewhere.

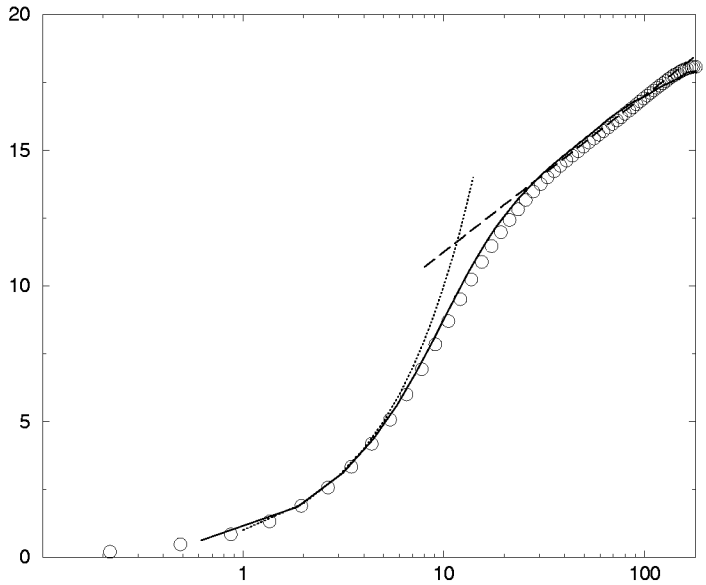

Figure 3: Mean velocity profiles: ——, LESTool; ∘, DNS of [1]; ·····, viscous sublayer; – – – – –, log layer.

7 Outlook – Combining MPI/OpenMP

The future of high-performance computer hardware will continue to evolve in the direction of clusters of SMP computers. In this model, SMP computing nodes are interconnected by fast, high-speed data links. While OpenMP provides a convenient way for parallel programming, MPI is the natural approach for distributed computing. In further development of the LESTool code, we therefore propose to use a hybrid MPI/OpenMP approach in an attempt to combine the best features of the two approaches. The hybrid approach provides the flexibility to choose between shared and distributed memory computing, or a combination of the two.

Our ultimate goal is to use the CFD code for the simulation of complex geometries by applying the multi-block concept. This strategy is designed to

take advantage of the hybrid MPI/OpenMP practice. From the overall pool of processors, different groups of processors will be clustered together. These clusters of processors will contain one or more grid blocks and communicate using MPI. Within each cluster the solver is parallelized using OpenMP. A schematic overview of this concept is depicted in figure 4. Testing of the code using this hybrid practice is currently underway.

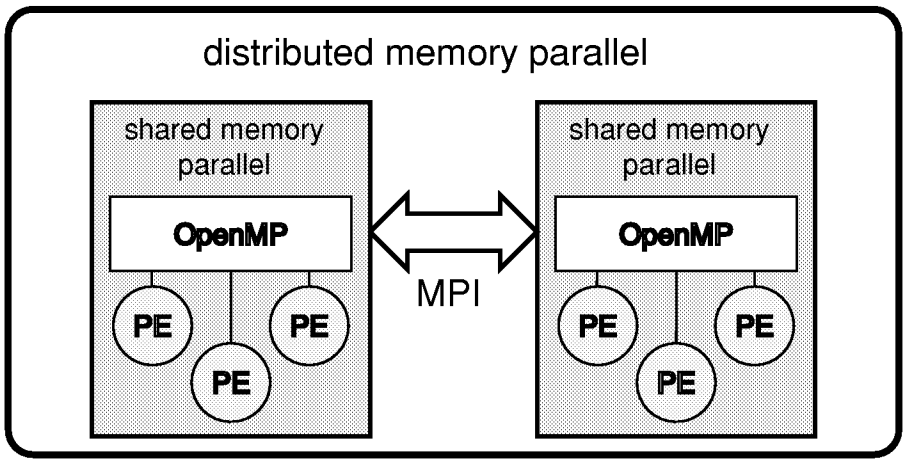

Figure 4: A concept for a hybrid MPI/OpenMP parallelization

8 Concluding Remarks

In this paper the numerical method for a high-order CFD code in curvilinear coordinates is described and two variants of the iterative implicit time integration scheme are discussed in detail. The DNS results obtained with this code show good agreement with results reported in the literature.

Experience on the shared-memory parallelization of an ADI-type CFD code is reported. Data management by loading 3-D arrays into 1-D arrays has been found to be very successful for the cache-based architectures. One surprising finding is the need to unroll all 5-by-5 matrix operations to achieve high MFLOP rates.

The advantage of Fortran 90 in implementing the current code is clear. The excellent new features of Fortran 90 in comparison to Fortran 77 offer the possibility to create an efficient, portable and maintainable program. Although compiler errors from different vendors have been encountered, we

found the overall advantages of using this new language outweigh the disadvantages. The performance of the current 7.2.1 version of the SGI compiler is very satisfactory.

The new OpenMP directives are used to parallelize the code on the SGI Origin 2000 computer. OpenMP offers a portable and simple way to use directive-based parallelization for the current LESTool code. We believe that for shared-memory parallelization OpenMP will be as widely accepted as its counterpart MPI for distributed parallel processing.

The scaling of the current code is fairly satisfactory up to 16 processors. However, the parallel efficiency for more than 32 processors is only marginal due to excess memory-access time.

A hybrid MPI/OpenMP concept that combines the best features of both MPI and OpenMP standards is proposed. This practice can be coupled with the multi-block strategy of the current CFD code and is believed to offer the additional mileage needed to speed up the code for multi-node, multi-processor computer architectures.

Acknowledgement

The current project is supported by NASA-Lewis under grant number NAG3-2099, with Dr. David Ashpis as the technical monitor. NAS and NCSA provide the computer time for the SGI Origin 2000 computers. A special thanks goes to Dr. James Taft at NASA-Ames for his advice on refining the LESTool code for serial performance.

References

[1] J. Kim, P. Moin, and R. Moser. Turbulence statistics in fully developed channel flow at low Reynolds number. *Journal of Fluid Mechanics*, 177:133–166, 1987.

[2] A.G. Kravchenko and P. Moin. B-spline methods and zonal grids for numerical simulation of turbulent flows. Technical Report TF-73, Flow Physics and Computation Division, Department of Mechanical Engineering, Stanford University, Stanford, California, 94305, 1998.

[3] S.K. Lele. Compact finite difference schemes with spectral-like resolution. *Journal of Computational Physics*, 103:16–42, 1992.

[4] D. E. Lenoski and W. D. Weber. *Scalable Shared-Memory Multiprocessing.* Morgan Kaufmann Publishers Inc., San Francisco, 1995.

[5] Message Passing Interface Forum, http://www.mpi-forum.org/docs/mpi-11-html/mpi-report.html. *MPI: A Message-Passing-Interface Standard*, May 1994.

[6] OpenMP Architecture Review Board, http://www.openmp.org/. *OpenMP: A Proposed Standard API for Shared Memory Programming.*, October 1997.

[7] T.H. Pulliam and D.S. Chaussee. A diagonal form of an implicit approximate factorization algorithm. *Journal of Computational Physics*, 39:347–363, 1981.

[8] Thomas H. Pulliam. Time accuracy and the use of implicit methods. AIAA-93-3360, 1993.

[9] Chi-Wang Shu. Essentially Non-Oscillatory and Weighted Essentially Non-Oscillatory schemes for hyperbolic conservation laws. NASA CR-97-206253 ICASE, Report No. 97-65, 1997.

A Appendix

A.1 Euler Flux Jacobians

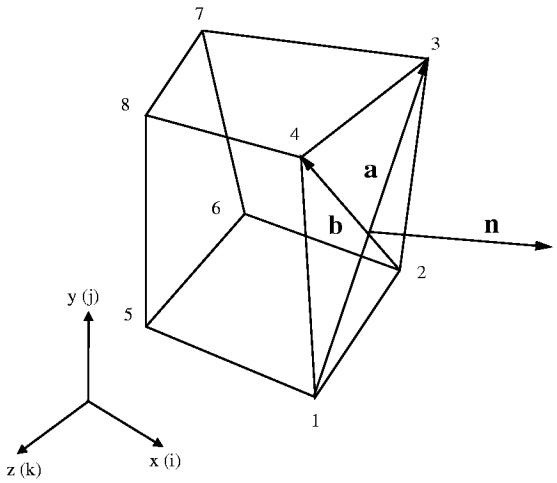

Figure 5: Discrete control volume

The Jacobian Matrix A_χ, where χ denotes ξ, η or ζ direction, at a control volume face shown in figure 5 can be defined as

$$A_\chi = \frac{\partial F_\chi}{\partial Q} = R^{-1}\Lambda_\chi R$$

where Λ_χ and R are the eigenvalue and eigenvector matrices of the Jacobian, respectively. The diagonal matrix Λ_χ can be written as

$$\Lambda = \begin{pmatrix} \lambda_{\chi,1} & 0 & 0 & 0 & 0 \\ 0 & \lambda_{\chi,2} & 0 & 0 & 0 \\ 0 & 0 & \lambda_{\chi,3} & 0 & 0 \\ 0 & 0 & 0 & \lambda_{\chi,4} & 0 \\ 0 & 0 & 0 & 0 & \lambda_{\chi,5} \end{pmatrix} \tag{38}$$

where

$$\lambda_{\chi,1} = \lambda_{\chi,2} = \lambda_{\chi,3} = U_\chi \quad (39)$$

$$\lambda_{\chi,4} = U_\chi + cV\sqrt{(\partial\chi/\partial x)^2 + (\partial\chi/\partial y)^2 + (\partial\chi/\partial z)^2} \quad (40)$$

$$\lambda_{\chi\,5} = U_\chi - cV\sqrt{(\partial\chi/\partial x)^2 + (\partial\chi/\partial y)^2 + (\partial\chi/\partial z)^2}. \quad (41)$$

The eigenvectors R can be expressed by the product of the three matrices

$$R = CNM \quad (42)$$

where

$$C = \begin{pmatrix} -c^2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & c & 0 & 0 & 1 \\ 0 & -c & 0 & 0 & 1 \end{pmatrix} \quad (43)$$

$$N = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & n_{n1} & n_{n2} & n_{n3} & 0 \\ 0 & n_{b1} & n_{b2} & n_{b3} & 0 \\ 0 & n_{c1} & n_{c2} & c_{c3} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (44)$$

and

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -u & 1 & 0 & 0 & 0 \\ -v & 0 & 1 & 0 & 0 \\ -w & 0 & 0 & 1 & 0 \\ (\gamma-1)K & -(\gamma-1)u & -(\gamma-1)v & -(\gamma-1)w & \gamma-1 \end{pmatrix} \quad (45)$$

In equation (44), vectors (n_{n1}, n_{n2}, n_{n3}), (n_{b1}, n_{b2}, n_{b3}) and (n_{c1}, n_{c2}, n_{c3}) corresponds to the unit vectors normal to the control volume face, ($\overrightarrow{13} \times \overrightarrow{24}$), and the units vectors in the 1-3 and 2-4 directions, respectively, as shown in figure 5.

The transformation R converts the conservative variables into Riemann variables

$$R\delta Q = (\delta p - c^2\delta\rho,\ \rho\delta U_{T1},\ \rho\delta U_{T2},\ \delta p + \rho c\delta U_n,\ \delta p - \rho c\delta U_n)^T \tag{46}$$

where U_n is the contravariant velocity normal to the control volume face $(\overrightarrow{13} \times \overrightarrow{24})$ and and U_{T2} are two tangential velocities in the $\overrightarrow{13}$ and $\overrightarrow{24}$ directions, respectively.

A.2 Viscous Flux Jacobians

Similarly, the Jacobian matrices B_χ can be computed as

$$B_\chi = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ -m_{22}u - m_{23}v - m_{24}w & m_{22} & m_{23} & m_{24} & 0 \\ -m_{23}u - m_{33}v - m_{34}w & m_{23} & m_{33} & m_{34} & 0 \\ -m_{24}u - m_{34}v - m_{44}w & m_{24} & m_{34} & m_{44} & 0 \\ b_1 & b_2 & b_3 & b_4 & m_{55} \end{pmatrix} \tag{47}$$

where

$$\begin{aligned} b_1 &= -m_{22}u^2 - m_{33}v^2 - m_{44}w^2 + m_{55}(K - c_vT) \\ &\quad -2m_{23}uv - 2m_{24}uw - 2m_{34}vw \\ b_2 &= (m_{22} - m_{55})u + m_{23}v + m_{24}w \\ b_3 &= m_{23}u + (m_{33} - m_{55})v + m_{34}w \\ b_4 &= m_{24}u + m_{34}v + (m_{44} - m_{55})w \end{aligned} \tag{48}$$

and

$$\begin{aligned} m_{22} &= \nu V[4(\partial\chi/\partial x)^2/3 + (\partial\chi/\partial y)^2 + (\partial\chi/\partial z)^2] \\ m_{33} &= \nu V[(\partial\chi/\partial x)^2 + 4(\partial\chi/\partial y)^2/3 + (\partial\chi/\partial z)^2] \\ m_{44} &= \nu V[(\partial\chi/\partial x)^2 + (\partial\chi/\partial y)^2 + 4(\partial\chi/\partial z)^2/3] \\ m_{55} &= \frac{k}{\rho}V[(\partial\chi/\partial x)^2 + (\partial\chi/\partial y)^2 + (\partial\chi/\partial z)^2] \\ m_{23} &= \nu V(\partial\chi/\partial x)(\partial\chi/\partial y)/3 \\ m_{24} &= \nu V(\partial\chi/\partial x)(\partial\chi/\partial z)/3 \\ m_{34} &= \nu V(\partial\chi/\partial y)(\partial\chi/\partial z)/3 \end{aligned} \tag{49}$$

A.3 Coordinate transformation

Gradients of any variable ϕ are evaluated using the following transformation:

$$\begin{pmatrix} \partial x/\partial a & \partial x/\partial b & \partial x/\partial c \\ \partial y/\partial a & \partial y/\partial b & \partial y/\partial c \\ \partial z/\partial a & \partial z/\partial b & \partial z/\partial c \end{pmatrix} \begin{pmatrix} \partial \phi/\partial x \\ \partial \phi/\partial y \\ \partial \phi/\partial z \end{pmatrix} = \begin{pmatrix} \partial \phi/\partial a \\ \partial \phi/\partial b \\ \partial \phi/\partial c \end{pmatrix} \tag{50}$$

where a, b and c are components of the transformed coordinates defined by the P-E, 1-3 and 2-4 directions, respectively, as shown in figure 5.

A.4 Metric vectors

The metric vectors $V(\partial\xi/\partial x,\ \partial\xi/\partial y,\ \partial\xi/\partial z)$, $V(\partial\eta/\partial x,\ \partial\eta/\partial y,\ \partial\eta/\partial z)$ and $V(\partial\zeta/\partial x,\ \partial\zeta/\partial y,\ \partial\zeta/\partial z)$ are the the surface vectors normal to the control volume faces in the ξ, η and ζ directions, respectively and can be written in a general form as $(S_1,\ S_2,\ S_3)$, which can be defined by the cross product of the vector $\overrightarrow{13}$ and $\overrightarrow{24}$ for any control volume face as shown in figure 5

$$\vec{S} = \frac{1}{2V}(\overrightarrow{13} \times \overrightarrow{24}) \tag{51}$$

Note that $\vec{S}$ and $\overrightarrow{PE}$ are not necessarily in the same direction in general curvilinear coordinates.

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 1999	Final Contractor Report

4. TITLE AND SUBTITLE

Shared Memory Parallelization of an Implicit ADI-type CFD Code

5. FUNDING NUMBERS

WU–523–90–13–00
NAG3–2099

6. AUTHOR(S)

Th. Hauser and P.G. Huang

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

University of Kentucky
Department of Mechanical Engineering
Lexington, Kentucky 40506

8. PERFORMING ORGANIZATION REPORT NUMBER

E–11472

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

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135–3191

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA CR—1999-208688

11. SUPPLEMENTARY NOTES

Project Manager, David Ashpis, Turbomachinery and Propulsion Systems Division, NASA Lewis Research Center, organization code 5820, (216) 433–8317.

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Unclassified - Unlimited
Subject Categories: 02, 07, and 34 Distribution: Nonstandard

This publication is available from the NASA Center for AeroSpace Information, (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***

A parallelization study designed for ADI-type algorithms is presented using the OpenMP specification for shared-memory multiprocessor programming. Details of optimizations specifically addressed to cache-based computer architectures are described and performance measurements for the single and multiprocessor implementation are summarized. The paper demonstrates that optimization of memory access on a cache-based computer architecture controls the performance of the computational algorithm. A hybrid MPI/OpenMP approach is proposed for clusters of shared memory machines to further enhance the parallel performance. The method is applied to develop a new LES/DNS code, named LESTool. A preliminary DNS calculation of a fully developed channel flow at a Reynolds number of 180, $Re_\tau = 180$, has shown good agreement with existing data.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Parallel computing; OpenMP; Shared memory; Optimization	29
	16. PRICE CODE: A03

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