A00-16188

# AIAA 2000-0275
# A Parallel High-Order Implicit Algorithm for Compressible Navier-Stokes Equations

Haibo Dong and Xiaolin Zhong
University of California, Los Angeles
Los Angeles, CA

**38th Aerospace Sciences Meeting & Exhibit**
January 10–13, 2000 / Reno, NV

# A Parallel High-Order Implicit Algorithm for Compressible Navier-Stokes Equations

Haibo Dong *and Xiaolin Zhong †
University of California, Los Angeles, California 90095

## Abstract

In recent years, direct numerical simulation (DNS) has become a powerful tool to study the stability and transition of compressible boundary layers. The difficulty in using explicit methods for DNS of hypersonic flow is the limitation of the temporal steps since the Navier-Stokes equations are stiff for explicit numerical schemes. Due to the progress in computer techniques in recent years, one way to overcome the limitation of small time steps in explicit method is by using distributed memory computers. However, a short marching step still can not be avoided during the computation due to stability limitation. Implicit method is considered to improve the efficiency of solving Navier-Stokes equations. In this paper, we extend our previous work to develop and validate the new high-order parallelized semi-implicit shock-fitting code solving the compressible Navier-Stokes equations. Divide and conquer method is used as a start point for the solution of the big banded matrix. The new parallel semi-implicit code has been tested in solving a 2-D convection-diffusion model equation first. Efficiency and accuracy are studied. Several test cases, which include the numerical simulations of the stability of 3-D supersonic Couette flow, and hypersonic boundary layer receptivity to freestream acoustic waves over 3-D blunt wedge, are used to evaluated the efficiency and accuracy of the new parallelized code.

## 1 Introduction

The prediction of laminar-turbulent transition in hypersonic boundary layers is a critical part of the aerodynamic design and control of hypersonic vehicles [1]. In recent years, direct numerical simulation has become a powerful tool in the study of fundamental flow physics of the stability and transition of boundary layers. The DNS of compressible boundary layer transition has been carried out by several research groups [2–12]. These studies show that the DNS of high-speed boundary layer transition is feasible on existing computers using efficient and accurate numerical methods. All these simulations are explicit methods, assuming a simple flat surface without the presence of shock waves. Explicit high-order finite-difference methods are used in the non-periodic streamwise and wall-normal directions, and the Fourier spectral collocation methods are used in the periodic spanwise direction [9,12,13]. But it is difficult to apply existing numerical methods for compressible boundary layer DNS to hypersonic boundary layers over blunt bodies, due to the lack of efficient semi-implicit numerical methods for compressible viscous flows [14]. The difficulties are from computing transient hypersonic flow with shock waves, and the stiffness of the equations in reacting hypersonic flows. Moreover, for viscous flow calculations, the extremely small grid sizes in the boundary layers near the wall is used. The stiffness of the governing equations refers to the fact that the time steps required by the stability requirement in the calculations are much smaller than that needed by accuracy consideration so that it is difficult to perform the simulation in reasonable computation time. This requires implicit treatment in numerical computations.

Due to the progress in computer techniques in recent years, another way to overcome the limitation of small time steps in explicit method is by using distributed memory computers. A most common method to distribute the load between processors is by dividing the computational domain into sub-domains. The easiest way to solve a problem divided into sub-domains is by using explicit numerical schemes [15,16]. However, such schemes still suffer from slow rate of convergence for steady state problems, and a short marching step for unsteady problems, due to stability limitation. Many researchers have tried to implement implicit schemes on parallel computers to overcome these limitations. Generally, these works are based on the approximate factorization method employing either an alternating direction implicit(ADI) [17–21] or a lower-upper(LU) factorization [22–25]. For instance, an application of an implicit scheme on MIMD computers was described by Abdallah(1994) [17]. Two or three one-dimensional ADI-like factors solutions in two or three dimensions in a finite difference formulation were used to solve the incompressible Navier-Stokes equations. Povitsky et al.(1997,1998) [20,21] described an asynchronous method for the parallelization of the ADI method. The parallelization efficiency was tested theoretically and experimentally. In Ref. [23], Venkatakrishnan implemented an incomplete lower-upper factorization in their implicit schemes. The preconditioned iterative methods were

*Graduate Student, Mechanical and Aerospace Engineering Department, Student Member AIAA, haibo@seas.ucla.edu
†Associate Professor, Mechanical and Aerospace Engineering Department, Senior Member AIAA, xiaolin@seas.ucla.edu.

used to solve the linear system. Wright(1998) [24] modified Jameson's method [22] and proposed data-parallel lower-upper relaxation method to solve Navier-Stokes equations. Ghizawi and Abdallah(1998) [25] derived a new lower-upper cycle-independent implicit factorization to solve the compressible Navier-Stokes equations. In their method, the factorization has the combined advantages of the stability of LU factorizations as well as the cycle independency that gives it better parallel processing functionality. Generally, in either case, the factors resulting from factoring the multidimensional problem are dependent on each other where the solution from the first factor in needed to be able to solve the second step, and so on. Therefore these steps have to be performed in series, which limits the parallelization features for these types of factorizations. Parallelization of implicit schemes is limited by global spatial data dependencies and the requirement to solve large linear systems in the form of block tridiagonal and/or pentadiagonal matrices. Thus the existing algorithms for sequential computers to distributed memory computers usually requires some algorithmic changes.

There are several methods [26–34] to obtain the solution of tridiagonal or pentadiagonal systems of n linear equations $Tx = d$ on scalar, vector and parallel computers. For a scalar machine the best direct solution method is Gaussian elimination, specialized for tridiagonal systems. For vector computers, cyclic reduction [27,28] is more efficient, since all essential arithmetic operations can be vectorized. One step of cyclic reduction consists of eliminating the odd-numbered unknowns from the even-numbered equations. This reduction can reduce the system to a lower bidiagonal system of half the size. On parallel computers with two CPU's the two-sided Gaussian elimination algorithm, discussed by Babuska(1972) [35], is very useful. The decomposition matrix $T$, as well as the subsequent solution process, can be performed in two parallel parts. Iterative methods [16,19,20,23,24] which include Gaussian-Seidel method, preconditioned conjugate gradient method [34] are also utilized on the MIMD computers.

Divide and conquer [18,29,31,32,32] is one of the most commonly used tools in the construction of algorithms on parallel computers. The basic idea underlying the technique is to solve a given problem of a certain size, by dividing the problem up into similar problems of smaller size, solving them, and then combining those solutions to form the solution of the original problem. Most commonly, the same technique is applied to the smaller problems, making the procedure recursive. Such divide and conquer algorithms are excellently suited for parallel computers. Based on the ways of the interprocessor communication in solving the reduced systems, there are mainly two kinds of methods on massively parallel computers. One way such as all-to-all broadcast [29] is to assemble complete reduced systems on each processor. Then each processor can efficiently solve the systems with a standard serial solver or other solvers. The other way is to leave the reduced systems distributed and to communicate appropriate equations between processors as necessary to consecutively eliminate reduced system coefficients [18]. Each algorithm has more merit in particular computing environments. In our paper, particular DAC algorithm is implemented with message-passing on distributed-memory machines on numerical simulation of hypersonic boundary layer receptivity problem.

Recently, [36–38] has developed and validated a set of fifth and seventh-order shock-fitting schemes for the DNS of practical 2-D and 3-D hypersonic flows over planar or axisymmetric blunt bodies. Recently, we [39] developed a new semi-implicit scheme which treats non-stiff terms in the governing equations explicitly and simultaneously treating the stiff terms implicitly to overcome the stiffness of the Navier-Stokes equations. In Ref. [40], we have already implemented parallel method to our explicit DNS code to study the hypersonic boundary-layer receptivity to freestream disturbances over an elliptic cross-section cone on IBM SP2 computers. However, it is hard to apply the semi-implicit code efficiently to massively distributed memory computers due to lack of effective algorithms in solving the large systems in the form of block multidiagonal matrices.

The objective of this paper is to extend the previous 2-D and 3-D parallel high-order shock-fitting schemes in [36, 38, 39, 41] to solve compressible Navier-Stokes equations implicitly. Divide and conquer algorithm is used as the start point to solved the big banded Jacobian matrices from the semi-implicit methods. A 2-D convection-diffusion model equation has been used to test the accuracy and efficiency of the new parallelized semi-implicit method. Meanwhile, analytical comparisons have been made to study the efficiency of different parallelized algorithms in solving the model equation. DNS of the 3-D supersonic Couette flow, hypersonic boundary layer receptivity over 3-D blunt wedge, are used to investigate the numerical accuracy and efficiency of the new parallelized code with or without shock-fitting method. Accuracy and efficiency are investigated in these test cases.

## 2 Governing Equations and Numerical Methods

### 2.1 Governing Equations

The governing equations are the unsteady three-dimensional Navier-Stokes equations written in a conservation-law form

$$\frac{\partial U}{\partial t} + \frac{\partial F_j}{\partial x_j} + \frac{\partial F_{vj}}{\partial x_j} = 0 \tag{1}$$

where

$$U = \{\rho,\ \rho u_1,\ \rho u_2,\ \rho u_3,\ e\} \tag{2}$$

$$F_j = \begin{Bmatrix} \rho u_j \\ \rho u_1 u_j + p\delta_{1j} \\ \rho u_2 u_j + p\delta_{2j} \\ \rho u_3 u_j + p\delta_{3j} \\ (e+p)\,u_j \end{Bmatrix} \quad (3)$$

$$F_{vj} = \begin{Bmatrix} 0 \\ \tau_{1j} \\ \tau_{2j} \\ \tau_{3j} \\ \tau_{jk}u_k - q_j \end{Bmatrix} \quad (4)$$

$$p = \rho RT \quad (5)$$

$$e = \rho(c_v T + \frac{\rho}{2} u_k u_k) \quad (6)$$

$$\tau_{ij} = -\mu\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) + 2\mu/3 \frac{\partial u_k}{\partial x_k}\delta_{ij} \quad (7)$$

$$q_j = -\kappa \frac{\partial T}{\partial x_j} \quad (8)$$

The details for the expressions above can be found in [36]. The viscosity and heat conductivity coefficients are computed by the Sutherland law and the assumption of a constant Prandtl number. Perfect gas assumption is used in all flows considered in this paper, though the method presented here can be easily extended to nonequilibrium real-gas hypersonic flows.

For numerical simulations of flow fields over a curved body surface, structured body fitted grids are used to transform the governing equations (1) in the Cartesian coordinates into a set of curvilinear three-dimensional coordinates ($\xi$, $\eta$, $\zeta$, $\tau$) along the body fitted grid lines. The transformation relations for the two set of coordinates are

$$\begin{cases} \xi = \xi(x,y,z) \\ \eta = \eta(x,y,z,t) \\ \zeta = \zeta(x,y,z) \\ \tau = t \end{cases} \Longleftrightarrow \begin{cases} x = x(\xi,\eta,\zeta,\tau) \\ y = y(\xi,\eta,\zeta,\tau) \\ y = y(\xi,\eta,\zeta,\tau) \\ t = \tau \end{cases} \quad (9)$$

The governing equations (1) are transformed into the computational domain ($\xi$, $\eta$, $\zeta$, $\tau$) as follows

$$\frac{1}{J}\frac{\partial U}{\partial \tau} + \frac{\partial E'}{\partial \xi} + \frac{\partial F'}{\partial \eta} + \frac{\partial G'}{\partial \zeta} + \frac{\partial E'_v}{\partial \xi} + \frac{\partial F'_v}{\partial \eta} + \frac{\partial G'_v}{\partial \zeta} + U\frac{\partial(\frac{1}{J})}{\partial \tau} = 0 \quad (10)$$

where

$$E' = \frac{F_1\xi_x + F_2\xi_y + F_3\xi_z}{J} \quad (11)$$

$$F' = \frac{F_1\eta_x + F_2\eta_y + F_3\eta_z + U\eta_t}{J} \quad (12)$$

$$G' = \frac{F_1\zeta_x + F_2\zeta_y + F_3\zeta_z}{J} \quad (13)$$

$$E'_v = \frac{F_{v1}\xi_x + F_{v2}\xi_y + F_{v3}\xi_z}{J} \quad (14)$$

$$F'_v = \frac{F_{v1}\eta_x + F_{v2}\eta_y + F_{v3}\eta_z}{J} \quad (15)$$

$$G'_v = \frac{F_{v1}\zeta_x + F_{v2}\zeta_y + F_{v3}\zeta_z}{J} \quad (16)$$

where $J$ is the Jacobian of the coordinate transformation, and $\xi_x$, $\xi_y$, $\xi_z$, $\eta_x$, $\eta_y$, $\eta_z$, $\eta_t$, $\zeta_x$, $\zeta_y$, and $\zeta_z$ are the grid transformation matrices. In the equations, the transformed inviscid fluxes $E'$, $F'$, and $G'$ are standard flux terms with known eigenvalues and eigenvectors. The transport flux terms $E'_v$, $F'_v$, and $G'_v$ contain both first-order and second-order spatial derivatives of velocity and temperature. These derivatives in the Cartesian coordinates $(x, y, z)$ are transformed into the computational coordinates $(\xi, \eta, \zeta)$ using a chain rule for spatial discretization.

## 2.2 High-Order Semi-Implicit method

Equation (10) for a three-dimensional flow in ($\xi$, $\eta$, $\zeta$, $\tau$) is additively split into relatively nonstiff part $\mathbf{f}(U_{ijk})$ and stiff part $\mathbf{g}(U_{ijk})$ as follows

$$\frac{1}{J}\frac{\partial U_{ijk}}{\partial t} = \mathbf{f}(U_{ijk}) + \mathbf{g}(U_{ijk}) \quad (17)$$

where

$$\mathbf{f}(U_{ijk}) = -\left(\frac{\partial E'}{\partial \xi} + \frac{\partial E'_v}{\partial \xi} + \frac{\partial F'_{v1}}{\partial \eta} + \frac{\partial G'}{\partial \zeta} + \frac{\partial G'_v}{\partial \zeta} + U\frac{\partial(\frac{1}{J})}{\partial \tau}\right)_{ijk} \quad (18)$$

$$\mathbf{g}(U_{ijk}) = -\left(\frac{\partial F'}{\partial \eta} + \frac{\partial F'_{v2}}{\partial \eta}\right)_{ijk} \quad (19)$$

where $i$ , $j$, and $k$ are the grid indices in the $\xi$, $\eta$ and $\zeta$ direction respectively. The transport flux vector in $\eta$ direction, $F'_v$, is split into $F'_{v2}$, the part of the viscous flux terms only involving normal derivatives, and $F'_{v1}$, the part of the viscous flux terms except $F'_{v2}$, i.e.,

$$F'_v = F'_{v1} + F'_{v2} \quad (20)$$

In Eq. (17), $\mathbf{g}(U_{ijk})$ is much stiffer than $\mathbf{f}(U_{ijk})$ since grid spacing in the wall-normal direction is much smaller than that used in streamwise direction for most viscous flow simulations. Therefore, high-order semi-implicit method can be used to overcome the stiffness of $\mathbf{g}(U_{ijk})$ while maintaining high-order temporal accuracy.

A simple local Lax-Friedrichs scheme is used to split the inviscid flux vectors into positive and negative wave fields. For example, the flux term $F'$ in Eq. (19) can be split into two terms of pure positive and negative eigenvalues as follows

$$F' = F'_+ + F'_- \quad (21)$$

where

$$F'_+ = \frac{1}{2}(F' + \lambda U) \quad (22)$$

$$F'_- = \frac{1}{2}(F' - \lambda U) \quad (23)$$

where $\lambda$ is chosen to be larger than the local maximum eigenvalues of $F'$

$$\lambda = \frac{|\nabla\eta|}{J}\left(\sqrt{(\epsilon c)^2 + u'^2} + c\right) \tag{24}$$

where

$$u' = \frac{\eta_x u + \eta_y v + \eta_z w + \eta_t}{|\nabla\eta|} \tag{25}$$

and the parameter $\epsilon$ is a small positive constant added for the smoothness of the splitting. The fluxes $F'_+$ and $F'_-$ contain only positive and negative eigenvalues respectively. Therefore, in the spatial discretization of Eq. (18), the flux derivatives are split into two terms

$$\frac{\partial F'}{\partial \eta} = \frac{\partial F'_+}{\partial \eta} + \frac{\partial F'_-}{\partial \eta} \tag{26}$$

where the first term on the right hand side is discretized by an upwind high-order finite difference method and the second term is discretized by a downwind high-order finite difference method. The fifth-order upwind explicit scheme [36] for the derivative of a variable $\phi$ is

$$\left(\frac{\partial \phi}{\partial \eta}\right)_j = \frac{1}{60\,h}\sum_{k=-3}^{3} a_{j+k}\,\phi_{j+k} \tag{27}$$

where $a_{j\pm3} = \pm 1 + \frac{1}{12}\alpha$, $a_{j\pm2} = \mp 9 - \frac{1}{2}\alpha$, $a_{j\pm1} = \pm 45 + \frac{5}{4}\alpha$, and $a_j = -\frac{5}{3}\alpha$. This scheme is fifth-order upwind scheme when $\alpha < 0$ ($\alpha = -2$). This scheme is fifth-order upwind scheme when $\alpha < 0$ ($\alpha = -2$). The scheme reduces to a sixth-order central scheme when $\alpha = 0$. Meanwhile, the corresponding sixth-order central explicit inner scheme for the second-order spatial derivative in the viscous term is

$$u''_i = \frac{1}{90h^2}\left(u_{i-3} - \frac{17}{2}u_{i-2} + 135u_{i-1} - 245u_i + 135u_{i+1} - \frac{17}{2}u_{i+2} + u_{i+3}\right) \tag{28}$$

The system of ordinary differential equations of Eq. (17) can be integrated in time using semi-implicit temporal schemes, where $\mathbf{f}$ is treated explicitly and $\mathbf{g}$ is treated implicitly. It was shown by Zhong [42] that in order to have a third or higher order temporal accuracy, the semi-implicit method need to be derived in a way that the effects of coupling between the implicit and explicit terms on the accuracy need be considered. Zhong [42] subsequently derived three kinds of third-order semi-implicit Runge-Kutta schemes for high-order temporal integration of the governing equations for reacting flow simulations. For example, third-order Rosenbrock Semi-Implicit Runge-Kutta (ASIRK-2C) Method is

$$\begin{cases} [\mathbf{I} - ha_1\mathbf{J}(\mathbf{u}^n)]\,\mathbf{k}_1 = h\{\mathbf{f}(\mathbf{u}^n) + \mathbf{g}(\mathbf{u}^n)\} \\ [\mathbf{I} - ha_2\mathbf{J}(\mathbf{u}^n + c_{21}\mathbf{k}_1)]\,\mathbf{k}_2 \\ \quad = h\{\mathbf{f}(\mathbf{u}^n + b_{21}\mathbf{k}_1) + \mathbf{g}(\mathbf{u}^n + c_{21}\mathbf{k}_1)\} \\ \mathbf{u}^{n+1} = \mathbf{u}^n + \omega_1\mathbf{k}_1 + \omega_2\mathbf{k}_2 \end{cases} \tag{29}$$

where $\omega_1 = \frac{1}{2}$, $\omega_2 = \frac{1}{2}$, $b_{21} = 1$, $a_1 = \frac{1}{4}$, $a_2 = \frac{1}{3}$, and $c_{21} = \frac{5}{12}$, The parameters of the semi-implicit Runge-Kutta methods are chosen by both stability and accuracy requirements with the simultaneous coupling between the explicit and implicit terms.

In Eqs. (29), $\mathbf{J}$ is the Jacobian matrices of the stiff flux terms is defined by $\mathbf{J}(\mathbf{u}) = \partial\mathbf{g}/\partial\mathbf{u}$. The components of the Jacobian $\mathbf{J}(\mathbf{u})$ are derived by considering the variation of $\mathbf{g}(U_{ijk})$ in Eq. (17):

$$\delta\mathbf{g}(U_{ijk}) = \delta\left(-\frac{DF'_+}{D\eta} - \frac{DF'_-}{D\eta} - \frac{DF'_{v2}}{D\eta}\right)_{ijk} \tag{30}$$

where $D/D\eta$ is the fifth-order finite difference approximation of the derivatives in wall-normal direction, and $F'_+$, $F'_-$ are inviscid fluxes given by Eq. (26). The variations for these inviscid fluxes are

$$\delta F'_\pm(U) = \frac{\partial\left(\frac{1}{2}(F' \pm \lambda U)\right)}{\partial U}\delta U \tag{31}$$

and those for the viscous flux are

$$\delta F'_{v2} = \mathbf{A_v}\frac{\partial^2}{\partial\eta^2}(\mathbf{M}\delta U) + \mathbf{B_v}\frac{\partial}{\partial\eta}(\mathbf{M}\delta U) + \mathbf{C_v}(\mathbf{M}\delta U)$$

where the matrices, $\mathbf{M}$, $\mathbf{A_v}$, $\mathbf{B_v}$, and $\mathbf{C_v}$, can be found in Ref. [39]. Substitiuting above equations into Eq. (30), where the derivatives are approximated by the fifth-order upwind scheme and sixth-order central scheme described in Eqs. (27) and (28), leads to

$$\begin{aligned}\delta\mathbf{g}(U_{ijk}) = {} & \mathbf{A}_{ijk}\delta U_{ij-3k} + \mathbf{B}_{ijk}\delta U_{ij-2k} + \mathbf{C}_{ijk}\delta U_{ij-1k} \\ & + \mathbf{D}_{ijk}\delta U_{ijk} + \mathbf{E}_{ijk}\delta U_{ij+1k} + \mathbf{F}_{ijk}\delta U_{ij+2k} \\ & + \mathbf{G}_{ijk}\delta U_{ij+3k}\end{aligned} \tag{32}$$

where the coefficient matrices can also be found in Ref. [39].

From Eq. (32), we can obtain the global Jacobian matrix for the system of ordinary differential equations, Eq. (17). For fifth and sixth-order schemes used in this paper, the semi-implicit scheme involves implicit equations of banded block matrix of nonzero seven diagonal elements along the $j$ grid direction. This block septidiagonal system of equations can be solved efficiently by a banded matrix solver.

### 2.3 Parallelization Approach

The standard (portable) message passing interface(MPI) is the parallel library we used to parallelize our code. A cluster of workstations are used to run our parallel codes. The present configuration has 24 RISC/6000 processors. The main memory capacity of these nodes is 256 megabytes. All of these nodes are connected via ethernet.

The first step of parallelization approaches is to initialize the MPI environment and to establish communicators that describe the communication contexts and the associated groups of processors. The

MPI_COMM_WORLD default communication that defines one communication context and uses the set of all processors as its group is the one we used in this version. There are several modes of communication in MPI. In this work the data exchange at the block boundaries is implemented using the MPI_SENDRECV routine. This routine is a locally blocking one which means that a send or a receive would not return until it is complete, and therefore a tight synchronization is achieved among the processors. Here, we are using the concept of volley which during a volley each processor sends and/or receives a maximum of one message, all messages being passed simultaneously.

### 2.4 Divide and Conquer Method

Basically, current banded matrix solver proceed in three steps by using divide and conquer method. First, the subsystems in each processor are reduced to N-form equations which called reduced systems. Second, this reduced systems are solved by the interprocessor communication of all necessary reduced system equations. Third, the solution is completed by backsubstitution of reduced system unknowns into the remaining N-form equations of each subdomain. Though the current divide and conquer method is developed for solving the septidiagonal, pentadiagonal or tridiagonal matrix on MIMD parallel computers, only tridiagonal matrix solver using $p$ nodes are presented here. The extension to septidiagonal or pentadiagonal matrix solver using $p$ nodes is straightforward.

Consider the tridiagonal linear system $Tx = d$ with $n = pk$ equations. The divide and conquer method divides the given tridiagonal system into $p$ parallel tasks of size $k = n/p$. $T$ is partitioned into block tridiagonal form with each diagonal block a $k \times k$ tridiagonal matrix and each subdiagonal block a $k \times k$ null matrix, except for one single nonzero element on its upper right (lower left) corner. The tridiagonal system can be written in block form as

$$\begin{pmatrix} T_1 & C_1 & & \\ B_1 & T_2 & \ddots & \\ & \ddots & \ddots & C_{p-1} \\ & & B_p & T_p \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_p \end{pmatrix}$$

where $T_j$ are tridiagonal matrices,

$$T_j = \begin{pmatrix} a_{(j-1)k+1} & c_{(j-1)k+1} & & \\ b_{(j-1)k+2} & a_{(j-1)k+2} & \ddots & \\ & \ddots & \ddots & c_{jk-1} \\ & & b_{jk} & a_{jk} \end{pmatrix}$$

and

$$B_j = \begin{pmatrix} 0 & \cdots & b_{(j-1)k+1} \\ & \ddots & \\ & \ddots & \\ 0 & \cdots & 0 \end{pmatrix}$$

$$C_j = \begin{pmatrix} 0 & \cdots & 0 \\ & \ddots & \\ & \ddots & \\ c_{(j-1)k} & \cdots & 0 \end{pmatrix}$$

where $j = 1, 2, ..., p$. $\mathbf{x}$ and $\mathbf{d}$ are partitioned to conform with the blocks of $T$:

$$x_j = \begin{pmatrix} x_{(j-1)k+1} \\ x_{(j-1)k+1} \\ \vdots \\ x_{(j-1)k+p} \end{pmatrix} \quad d_j = \begin{pmatrix} d_{(j-1)k+1} \\ d_{(j-1)k+2} \\ \vdots \\ d_{(j-1)k+p} \end{pmatrix}$$

For $j = 1, 2, ..., p$, by defining

$$\begin{aligned} y_j &= T_j^{-1} d_j \\ z_1 &= T_1^{-1} e_k \\ z_m &= T_j^{-1} e_1 \qquad z_{m+1} = T_j^{-1} e_k \\ z_{2p-2} &= T_p^{-1} e_1 \end{aligned}$$

where $m = 2i - 2$, $e_1$ and $e_k$ are the first and last unit vector. We have the following implicit relations

$$\begin{aligned} x_1 &= y_1 + \alpha_1 z_1 \\ x_j &= y_j + \alpha_{2j-2} z_{2j-2} + \alpha_{2j-1} z_{2j-1} \\ &\qquad 2 \le j \le p-1 \\ x_p &= y_p + \alpha_{2p-2} z_{2p-2} \end{aligned}$$

where $\alpha_1$, $\alpha_2$,..., $\alpha_{2p-2}$, $\alpha_4$ satisfy the relations as following reduced systems:

$$\begin{pmatrix} {e_k}^T z_1 & 1/b_{k+1} & & \\ 1/c_k & {e_1}^T z_2 & \ddots & \\ & \ddots & \ddots & 1/b_{k(p-1)+1} \\ & & 1/c_{k(p-1)} & {e_1}^T z_{2p-2} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{2p-2} \end{pmatrix} = \begin{pmatrix} {e_k}^T y_1 \\ {e_1}^T y_2 \\ \vdots \\ {e_1}^T y_{2p-2} \end{pmatrix}$$

From the solution of above equations, we then obtain the desired solution $x_1$, $x_2$, $x_3$. In this paper, we are using two approaches, all-to-all broadcast [29] and folded skip-decoupling method [18], to solve above reduced systems. All-to-all broadcast assembles complete reduced systems on each processor. Then each processor can solve the systems independently. For $p$ processors, the method needs minimal $\mathbf{int}(log_2(p-1))+1$ number of

volleys, each volley requiring passage of $p$ messages. The folded skip-decoupling method leaves the reduced systems distributed and to communicate appropriate equations between processors as necessary to consecutively eliminate reduced system coefficients. For $p$ processors, $p-1$ volleys are necessary to complete solution of the reduced system, two messages being passed per volley.

### 2.5 Boundary Conditions

The physical boundary conditions for viscous flows are non-slip condition for velocity and isothermal or adiabatic condition for temperature. The freestream flow conditions are specified by a given flow. For the flow disturbed by disturbances, the disturbances are specified according to the particular physical nature of the disturbances.

For numerical simulations, it is necessary to set numerical boundary conditions for some flow variables in addition to the physical boundary conditions. This is especially the case at the computational boundary of the exit and inlet. There have been many investigations on the issues of numerical boundary conditions [43–49] for the direct numerical simulations of compressible as well as incompressible boundary layers. Examples of the work include: Israeli and Orszag (1981) presented a sponge layer with absorbing boundary conditions in the study of problems involving wave propagation, Streett and Macaraeg (1989/90) proposed a buffer domain in the outflow boundary for unsteady transition-to-turbulence simulations, Poinsot and Lele (1992) discussed characteristic-based boundary conditions for direct simulation of compressible viscous flows, and Guo, Kleiser, and Adams (1994) compared the results obtained by using above different boundary conditions in the simulation of compressible boundary layer transition. Recently, Pruett et al. (1995) used outflow buffer domain to do DNS of high-speed boundary-layer flows, and Collis and Lele (1996) studied the problem of compressible swept leading-edge receptivity by using inflow sponge and outflow sponge boundary conditions.

Since the emphasis of current paper is the parallelized semi-implicit method for efficient and accurate time integration of the governing equations, we will mainly consider flows with a supersonic exit where the reflection of disturbances are negligible.

## 3 Numerical Results

A more flexible three-dimensional solver has been written by using parallelized explicit and semi-implicit high-order upwind schemes for the spatial discretization with and without a high-order shock fitting algorithm. Fourier collocation method are used in computing the azimuthal direction. Investigations have been done as followings:

### Performance Studies on Parallelized Explicit and Implicit Algorithms

The execution time of an algorithm for a problem size denoted by $n$, on a parallel system with $P$ processors can be written as:

$$T(n,P) = T_{calculation} + T_{communication} \tag{33}$$

Ideally, we can assume that no other overhead occurs except communication of the overlap regions, the sequential execution time is:

$$T(n,1) = P \times T_{calculation} \tag{34}$$

Hence the speedup and parallel efficiency are given by:

$$\begin{aligned} S(n,P) &= P \times \frac{T_{calculation}}{T_{calculation} + T_{communication}} \\ &= \frac{P}{1 + T_{communication}/T_{calculation}} \qquad (35) \\ E(n,P) &= \frac{1}{1 + T_{communication}/T_{calculation}} \qquad (36) \end{aligned}$$

For the best algorithm, $S(n,P) = P$, $E(n,P) = 1$, which means the ratio of $\frac{T_{communication}}{T_{calculation}}$ is close to zero. The amount of data sent and received per processor is proportional to the number of boundary cells, while the amount of computations performed by each processor is proportional to the number of interior cells. For the model problem, we have:

$$T_{calculation} = c_1 \times nx \times ny \times t_{calc} \tag{37}$$

$$T_{communication} = c_2 \times 2 \times nx \times t_{comm} \tag{38}$$

where $t_{calc}$ represents the time required to perform a floating point operation, $t_{comm}$ denotes the time needed to communicate one floating point number, $c_1$ and $c_2$ are constant. The cost of sending a message between neighbouring processors can be written as:

$$T(n) = t_{delay} + nt_{send} \tag{39}$$

where $n$ is the size of message, $t_{delay}$ is the delay time caused by hardware and software delays, and $t_{send}$ is the marginal communication time per word. Usually, $t_{delay}$ is much larger than $t_{send}$. To avoid too much $t_{delay}$ during the parallelized computation, we prefer sending a big message instead of many short messages. So $t_{comm}$ depends very much on the average size of the message that is sent: for small messages $t_{comm} \simeq t_{delay}$ while for very large messages $t_{comm} \simeq t_{send}$. In our study, we are considering ideal case of $t_{comm} \simeq t_{send}$.

### Convection-Diffusion Model Equation

A two-dimensional linearized model convection-diffusion equation bounded by two parallel walls is

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} = \frac{1}{R}\frac{\partial^2 u}{\partial y^2} \tag{40}$$

where $R$ is the "Reynolds number". The boundary conditions are $u(x,0) = u(x,1) = 0$. When $R$ is large, there is a thin viscous boundary layer on the wall with large gradients in $y$ direction. This model problem is not a practical flow problem, but it is used to test the accuracy and efficiency of the parallelized semi-implicit method. ee are looking for the temporal development of the solution in the following form:

$$u(x,y,t) = Y(y)e^{ikx}e^{-i\omega t} \tag{41}$$

where $k$ is a real number. The complex parameter $\omega$ and $Y(y)$ are an eigenvalue and eigenfunction of the characteristic equation. Substituting Eq. (41) into Eq. (40) leads to the following solution:

$$u_n(x,y,t) = Ce^{\frac{Ry}{2}}\sin n\pi y e^{ik(x-t)}e^{-\alpha_n t} \tag{42}$$

where $n = 1, 2, \cdots$. The solution represents and exponential decay of the oscillation energy. There, if we use $u_n(x,y,0)$ given by Eq. (42) as an initial condition, the exact solution of the model equation is also given by the same equation.

Several algorithms which include explicit method, semi-implicit method using divide and conquer and using iterative method, fully implicit method using iterative method, have been studied on the performance of the algorithms by solving the above model equations. Figure 2 shows the comparisons of the efficiency between different algorithms by only partitioning the computational domain in y direction. Among all the algorithms, explicit method has the highest efficiency on communication, however, its computation time step is very small due to the very small grid sizes in y direction. Full implicit method has the lowest efficiency compared with explicit method and semi-implicit, but it can approach highest computation time step. However, it requires large memory to convert full implicit equations and takes a large amount of CPU time. Semi-implicit method is the a compromise between computational efficiency and numerical accuracy for the DNS studies, where only the derivatives in the wall-normal direction are treated implicitly. Between different parallelized semi-implicit methods, divide and conquer method has better efficiency than iterative method since the divide and conquer method spends less time on communication when more processors participate in the computation.

Then, we chose parallelized semi-implicit method with DAC method to solve the big banded Jacobian matrix as the start algorithm in direct numerical simulation. Similar to semi-implicit method of Navier-Stokes equations presented in the previous section, the finite difference discretization of Equation (40) leads to a system of semi-discrete ordinary different equations, i.e.

$$\frac{du_{ij}}{dt} = \mathbf{f}(u_{ij}) + \mathbf{g}(u_{ij}) \tag{43}$$

where

$$\mathbf{f}(u_{ij}) = -\frac{\partial u}{\partial x} \tag{44}$$

$$\mathbf{g}(u_{ij}) = \left\{-\frac{\partial u}{\partial y} + \frac{1}{R}\left(\frac{\partial^2 u}{\partial y^2}\right)\right\}_{ij} \tag{45}$$

where explicit third-order upwind approximation is used for $u_x$, and fourth-order central difference approximation is used for $u_y$ and $u_{yy}$ terms. ASIRK-1C has been used to do the temporal discretization. The divide and conquer method with all-to-all broadcast solver and folded skip-decoupling solver on reduced systems are used to solve the big banded matrix from the semi-implicit method.

A periodic boundary condition is used in the $x$ direction. An antisymmetric boundary condition is used at the walls to calculate $u$ located at one grid-point outside of the walls. Simple uniform grids are used. The conditions for the calculation are: $R = 10$, $k = 0.01$, $C = 1$, $n = 3$. To test the accuracy of the new algorithm, the computation uses a set of $51 \times 45$ grids to discretize the computation domain, $(0, \frac{2\pi}{k}) \times (0, 1)$ and three processors are used during computation.

Figure 3 and Figure 4 show the numerical solution distribution along the $x$ direction at $y = 0.18$ and the distribution along the $y$ direction at $x = 0$ and at $t = 0.5335$. Figure 5 shows the comparison of contours of numerical solution and exact solution at $t = 0.5335$. The agreements are well.

To investigate the efficiency of the parallelized semi-implicit code, no domain partitionings is in $x$ direction. Only the number of subdomains in $y$ direction are varied. CPU time is recorded and compared for several cases. Table 2 and Table 3 show the comparisons of execution time, speedup and parallel efficiency for different partitions in $y$ direction using the two kinds of reduced system solvers in the new parallelized semi-implicit codes. Both all-to-all broadcast solver and folded skip-decoupling method solve the problem efficiently. Along with the increase of the processor, the folded skip-decoupling method approaches higher parallel efficiency and speedup compared to all-to-all broadcast solver since there are less messages passed during interprocessor communication in solving the reduced systems.

### 3-D Supersonic Couette Flow Stability

Compressible Couette flow is a wall-bounded parallel shear flow which is a simple example of hypersonic shear flows. Because the mean flow is parallel, the linear stability analysis based on the full Navier-Stokes equations does not involve the parallel approximation of a developing boundary layer. The LST results for compressible Couette flow are taken from [50]. Both steady and unsteady three-dimensional computations are tested.

#### Steady Flow Solutions

We first used the high-order parallelized semi-implicit Navier-Stokes codes to compute the steady solutions

Table 1: Numerical errors for computations of supersonic Couette flow using parallelized semi-implicit scheme. ($e_1 = ||e||_1$ and $e_2 = ||e||_2$)

| Grids | $e_1 \times 10^{-8}$ | ratio | $e_2 \times 10^{-8}$ | ratio |
|---|---|---|---|---|
| 91 | 5.4772 | – | 1.0418 | – |
| 181 | 0.1776 | 30.84 | 0.0294 | 35.47 |

of the supersonic Couette flow. Divide and conquer method is used to solve the big Jacobian matrix parallelizedly. The results are compared with the "exact" solutions obtained by a shooting method with several order of magnitudes smaller errors. Several cases by using different grids and different processors have been tested. The results shown in this paper are using 3 partitionings in $x$ direction and 3 partitionings in $y$ direction. There is no partitioning in $z$ direction. Each subdomain has grids of $61 \times 61 \times 4$. Figure 6 shows the grids of $x - y$ cross-section of the computation domain. Different colors represent the different fields computed by different processors. The flow conditions are: $M_\infty = 2$, the upper wall is an isothermal wall with $T_\infty = 220.66667K$ while the lower wall is an adiabatic wall. The gas is assumed perfect gas with $\gamma = 1.4$ and $Pr = 0.72$. The viscosity coefficient is calculated by the Sutherland's law

$$\mu = T^{3/2}\left(\frac{1+C}{T+C}\right) \tag{46}$$

where $C$ is taken to be 0.5.

Figure 7 and Figure 8 shows the comparisons f the steady temperature and velocity profile obtained by using a parallelized semi-implicit fifth-order upwind scheme with the "exact" solution. 180 uniform grid points are used in y direction. From the figure, numerical results agree well with the exact solutions. To evaluate the accuracy of the algorithm, the numerical simulations are conducted using two sets of uniform grids. The quantitative numerical errors of the simulations using two kinds of uniform grids are listed in Table 1. The table shows that the numerical errors for this parallelized fifth-order semi-implicit scheme in spatial discretization are of the order of $10^{-7}$ using 91 grid points and $1.776 \times 10^{-8}$ using 181 grid points. The theoretical ratio of the errors between the coarse grids and the fine grids are 32 for a fifth-order scheme. The results in the table show that the numerical algorithms are able to maintain high order of accuracy.

Unsteady Flow Solutions

We conduct numerical simulations for the temporal stability of the compressible Couette flow by simulating the development of given initial disturbances in the 3-D flow field. The initial conditions are the steady flow solutions plus disturbances given by a set of eigenfunctions obtained by linear stability analysis. For small initial disturbances, the growth or decay of the disturbances is given by the eigenvalue of the eigen-mode. The same stretched grids are used in $y$ direction as those used in the LST calculation. The computational domain in the simulation is one period in length in the $x$ direction and periodic boundary conditions are used.

The flow conditions of the first case are: $M_\infty = 2$ and $Re_\infty = 1000$. For this case, the initial disturbance wave has a dimensionless wave number $\alpha = 3$, and the eigenvalue obtained from the temporal linear stability analysis is

$$\begin{aligned} \omega &= \omega_r + \omega_i i \\ &= 5.52034015848 - 0.132786378788i \end{aligned} \tag{47}$$

where a negative $\omega_i$ means that the disturbances will decay in time with a dimensionless frequency, $\omega_r$. 9 processors are used for the computation and $60 \times 60 \times 4$ stretching grids are used for each subdomain.

Figure 9 shows the comparison of the DNS results using parallelized semi-implicit method and the LST prediction for the time history of velocity perturbations at a fixed point in the 3-D supersonic Couette flow field. The disturbance wave decays along with time increases. Figure 10 shows the distribution of instantaneous flow perturbations in the y direction after about six wave periods. The figures show that the instantaneous perturbations of all flow variables for the 3-D numerical simulations using the new parallelized semi-implicit method agree well with the LST results.

In order to study the efficiency of parallelized big Jacobian matrix solver, we keep no partitionings in both $x$ and $z$ directions. Only different number of processors are used in solving the domain partitionings in $y$ direction. Table 4 shows the comparison of records of real CPU time consumed between the parallelized semi-implicit method by using different processors in $y$ direction and the full explicit method after running the codes to the end of one time wave periods. Compared with parallelized explicit method, the computational efficiency can be improved by using parallelized semi-implicit method. But the efficiency decreases along with the number of subdomain increasing since more communication time is used to solve the banded Jacobian matrix and part of them are unnecessary waiting.

### Receptivity of A Hypersonic Boundary Layer

The numerical simulation for the receptivity of hypersonic flows over a blunt wedge is carried out using the new 3-D fifth-order shock fitting scheme where the outer grid line is the bow shock. Figure 1 shows a schematic of the general 3-D shock fitted grids. The grids are stretched in both streamwise and wall-normal directions. The unsteady bow shock shape and shock oscillations are solved as part of the computation so-

lutions. In our test case the body surface of the 3-D parabolic leading edge is given by

$$x^* = b^* y^{*2} - d^* \tag{48}$$

where $b^*$ is a given constant and $d^*$ is taken as the reference length.

In the simulations, steady flow solutions are first obtained by advancing the unsteady flow computations to convergence using the new parallelized semi-implicit computer code. No disturbances are imposed in the freestream. Subsequently, freestream disturbances are superimposed on the steady mean flow to investigate the development of T-S waves in the boundary layer with the effects of the bow shock interaction. The freestream disturbances are assumed to be weak monochromatic planar acoustic waves with wave front oblique to the center line of the body in the $x-z$ plane at an angle of $\psi$. The perturbations of flow variables in the freestream introduced by the freestream acoustic wave before reaching the bow shock can be written in the following form:

$$q'_\infty = |q'|_\infty e^{i(k\cos\psi\, x + k\sin\psi\, z - \omega t)} \tag{49}$$

where $|q'|$ represents one of the flow variables, $|u'|$, $|v'|$, $|w'|$, $|p'|$, and $|\rho'|$. The freestream perturbation amplitudes satisfy the following relations:

$$|u'|_\infty = \epsilon\cos\psi\,, \qquad |v'|_\infty = 0\,,$$
$$|p'|_\infty = \gamma M_\infty \epsilon\,, \qquad |\rho'|_\infty = M_\infty \epsilon$$
$$|w'|_\infty = \epsilon\sin\psi$$

where $\epsilon$ represents the freestream wave magnitude. The angle $\psi$ is the angle of freestream wave with respect to the $x$ axis in the $x$-$z$ plane, where $\psi = 0$ corresponds to 2-D planar waves. The parameter $k$ is the dimensionless freestream wave number which is related to the dimensionless circular frequency $\omega$ by:

$$\omega = k\,(\cos\psi + M_\infty^{-1}) \tag{50}$$

The dimensionless frequency $F$ is defined as:

$$F = \frac{\omega^* \nu^*}{U_\infty^{*2}}\,. \tag{51}$$

Steady Flow Solutions

The specific flow conditions are

$M_\infty = 15$ $\qquad$ $\epsilon = 5\times 10^{-4}$
$T^*_\infty = 192.989\,K$ $\qquad$ $p^*_\infty = 10.3\,Pa$
$T^*_w = 1000\,K$ $\qquad$ $\gamma = 1.4$
$R^* = 286.94\,Nm/kgK$ $\qquad$ $Pr = 0.72$
$b^* = 40\,m^{-1}$ $\qquad$ $d^* = 0.1\,m$
$T^*_r = 288\,K$ $\qquad$ $T^*_s = 110.33\,K$
$\mu^* = 0.17894\times 10^{-4} kg/ms$
Nose Radius of Curvature $= r^* = 0.0125\,m$
$Re_\infty = \rho^*_\infty U^*_\infty d^*/\mu^*_\infty = 6026.55$

The body surface is assumed to be a non-slip wall with an isothermal wall temperature $T^*_w$.

To show the efficiency and accuracy of the code using high-order semi-implicit method, we compare the results of the steady flow solutions of the Navier-Stokes equations for the viscous hypersonic flow over the 3-D blunt wedge obtained by using this new method with the results obtained by using the parallelized explicit fifth-order unsteady computer code in Ref. [40]. We use ASIRK-1C method in time steps and advance the solutions to a steady state without freestream perturbations.

The numerical accuracy of the parallelized semi-implicit method is evaluated by comparing the solutions with those of parallelized explicit method. Figure 11 shows two kinds of different domain partitionings which correspond to different processors in computing in $y$ direction. In the simulation, 321 grid points are used in the streamwise direction, 241 points are used in the wall-normal direction, and 4 Fourier collocation points are used in computing the azimuthal direction. Figure 12 shows steady flow solutions for temperature and Mach number contours obtained by using 12 processors simultaneously. There are two subdomains in $x$ direction and 6 subdomains in $y$ direction. Figure 13 compares the pressure on the body surface between these two numerical methods. Figure 14 shows the comparison of Mach number distribution along the stagnation line. All these steady solutions show that the results of parallelized semi-implicit method agree well with the results of the parallelized explicit method. Accurate steady solutions can be obtained by running parallelized high-order semi-implicit method. Table 5 shows the comparison of the computational time consuming between parallelized semi-implicit method and the parallelized explicit method. The parallelized semi-implicit method can reach steady state quickly without losing accuracy.

Unsteady Flow Solutions

In this section, we choose the generation of boundary-layer T-S and inviscid instability waves by freestream acoustic disturbances for hypersonic flow over a parabolic leading edge with freestream disturbance frequency $F = 1770$, and $\epsilon = 5\times 10^{-3}$ as the test case for comparing the new parallelized semi-implicit method using divide and conquer method solving banded Jacobian matrix with the parallelized explicit method. We change the freestream disturbance wave angle $\psi$ to $30^0$, $45^0$, $60^0$. Efficiency and accuracy of the new method are studied.

Figure 15 shows the comparison contours of the instantaneous perturbation $u'$ the velocity in x direction after the flow field has reached a periodic state for $\psi = 30^0, 45^0$, and $60^0$. The numerical solutions are obtained by using the new parallelized high-order semi-

implicit method with a set of $320 \times 240 \times 16$ grids. The instantaneous contours of $u'$ show the development of three-dimensional first-mode waves in the boundary layer on the surface. For $\psi = 60^0$ case, the first-mode waves in the boundary layer keeps developing. When $\psi$ changes to $45^0$, the break region between first-mode waves and second-mode waves in the boundary layer appears. The length of the region of first-mode wave domination is shorter when $\psi$ given to $45^0$. Parallelized high-order semi-implicit method made us feasible to study more on this complicated hypersonic receptivity problem. Further studies and comparisons will be continued. Figures 16 and 17 compare the distribution of instantaneous presure perturbations along the wedge surface and behind the bow shock surface respectively. The results agree well between the parallelized semi-implicit method and parallelized explicit method.

## Summary

We are developing new algorithms to efficiently apply implicit method on massively distributed memory computers. Parallelized semi-implicit treatment is used to overcome stiffness in viscous wall-normal derivative terms, while the streamwise terms are computed by explicit methods parallelizedly for efficient unsteady flow calculations. As a startup, divide and conquer method is used to solve the big banded Jacobian matrix from the parallelized semi-implicit method. All-to-all broadcast solver and folded skip-decoupling solver are applied to solved the reduced systems from parallel computers. From the results of studying convection-diffusion model equation and the instability of supersonic Couette flow, the computation efficiency can be improved by using parallelized semi-implicit methods while maintaining the same accuracy as that of the parallelized explicit methods. We can also get the satisfied results by using the new parallelized semi-implicit methods for the direct simulations of hypersonic boundary layers over blunt body to freestream acoustic disturbance. Meanwhile, the computational time consuming is decreased.

## Acknowledgments

This research was supported by the Air Force Office of Scientific Research under grant numbers F49620-97-1-0030 and F49620-00-1-0101 monitored by Dr. Len Sakell.

## References

[1] Board, D. S., "Final Report of the Second Defense Science Board Task Force on the National Aero-Space Plane (NASP)," *AD-A274530, 94-00052, November*, 1992.

[2] Erlebacher, G. and Hussaini, M. Y., "Numerical experiments in supersonic boundary-layer stability," *Physics of Fluids: A*, Vol. 2, 1990, pp. 94–104.

[3] Ng, L. L. and Erlebacher, G., "Secondary Instability in Compressible Boundary Layers," *Physics of Fluids: A*, Vol. 4, 1992, pp. 710–717.

[4] Thumm, A., Wolz, W., and Fasel, H., "Numerical Simulation of Spatially Growing Three-Dimensional Disturbance Waves in Compressible Boundary Layers," *Laminar-Turbulent Transition, IUTAM Symposium*, Toulouse, France, 1989, edited by D. Arnal, R. Michel, Springer-Verlag Berlin, 1990.

[5] Fasel, H., Thumm, A., and Bestek, H., "Direct Numerical Simulation of Transition in Supersonic Boundary Layer: Oblique Breakdown," *Transitional and Turbulent Compressible Flows*, ASME FED-Vol. 151, 1993, pp. 77–92.

[6] Eibler, W. and Bestek, H., "Spatial Numerical Simulations of Linear and Weakly Nonlinear Instabilities in Supersonic Boundary Layers," *Theoretical and Computational Fluid Dynamics*, Vol. 8, 1996, pp. 219-235.

[7] Adams, N. A. and Kleiser, L., "Numerical Simulation of Transition in a Compressible Flat Plate Boundary Layer," *Transitional and Turbulent Compressible Flows*, ASME FED-Vol. 151, 1993, pp. 101–110.

[8] Adams, N. A., "Subharmonic Transition to Turbulence in a Flat-Plate Boundary Layer at Mach Number 4.5," *Journal of Fluid Mechanics*, Vol. 317, 1996, pp. 301–335.

[9] Guo, Y., Adams, N. A., Sandham, N. D., and Kleiser, L., "Numerical Simulation of Supersonic Boundary Layer Transition," *Application of Direct and Large Eddy Simulation to Transition and Turbulence*, AGARD-CP-551, 1994.

[10] Reed, H. L., "Direct Numerical Simulation of Transition: the Spatial Approach," *Progress in Transition Modeling*, AGARD-Report-793 1994.

[11] Pruett, C. D. and Chang, C. L., "A Comparison of PSE and DNS for High-Speed Boundary-Layer Flows," *Transitional and Turbulent Compressible Flows*, edited by L. D. Kral and T. A. Zang, ASME FED-Vol. 151, 1993, pp. 57-67.

[12] Pruett, C. D., Zang, T. A., Chang, C.-L., and Carpenter, M. H., "Spatial Direct Numerical Simulation of High-Speed Boundary-Layer Flows, Part I: Algorithmic Considerations and Validation," *Theoretical and Computational Fluid Dynamics*, Vol. 7, 1995, pp. 49–76.

[13] Lele, S. K., "Compact Finite Difference Schemes with Spectral-like Resolution," *Journal of Computational Physics*, Vol. 103, 1992, pp. 16–42.

[14] Kleiser, L. and Zang, T. A., "Numerical Simulation of Transition in Wall-Bounded Shear Flows," *Ann. Rev. Fluid Mech.*, Vol. 23, 1991, pp. 495–535.

[15] Yadlin, Y. and Caughey, D. A., "Parallel Computing Strategies for Block Multigrid Implicit Soultion of the Euler Equations," *AIAA Journal*, Vol. 30, No. 8, pp. 2032–2038, 1992.

[16] Tidriri, M. D., "Domain Decomposition for Compressible Navier-Stokes Equations with Different Discretizations and Formulations," *Journal of Computational Physics*, Vol. 119, 1995, pp. 271-282.

[17] Abdallah, S., "Parallel Processing for Implicit Solutions of the Compressible Navier-Stokes Equations," *AIAA Journal*, Vol. 32, No. 12, pp. 2469–2471, 1994.

[18] M. A. Lambert, G. H. R. and Hewett, D. W., "A Parallel DSDADI Method for Solution of the Steady State Diffusion Equation," *Parallel Computing*, Vol. 23, pp. 2041–2065, 1997.

[19] A. Averbuch, L. Ioffe, M. I. L. V., "Two-dimensional parallel solver for the solution of Navier-Stokes Equations with Constant and Variable Coefficients Using ADI on Cells," *Parallel Computing*, Vol. 24, pp.673–699, 1998.

[20] Povitsky, A. and Wolfshtein, M., "Parallelization Efficiency of CFD Problems on a MIMD Computer," *Computers and Fluids*, Vol. 26, No. 4, 1997, pp. 359-371.

[21] Povitsky, A., "Efficient Parallelization of a Parabolized Flow Solver," *Computers and Fluids*, Vol. 27, No. 8, 1998, pp. 985-1000.

[22] Jameson, A. and Yoon, S., "Lower-Upper Implicit Schemes with Multiple Grids for the Euler Equations," *AIAA Journal*, Vol. 25, No. 7, pp. 929–935, 1987.

[23] Venkatakrishnan, V., "Parallel Implicit Unstructured Grid Euler Solvers," *AIAA Journal*, Vol. 32, No. 10, pp. 1985–1991, 1994.

[24] M. Wright, G. Candler, D. B., "Data-Parallel Line Relaxation Method for the Navier-Stokes Equations," *AIAA Journal*, Vol. 36, No. 11, pp. 1603–1609, 1998.

[25] Ghizawi, N. and Abdallah, S., "Parallel Processing Scheme for the Navier-Stokes Equations, Part 1: Scheme Development," *AIAA Journal*, Vol. 36, No. 11, pp. 2013–2019, 1998.

[26] vand der Vorst, H., "Large Tridiagonal and block Tridigonal Linear Systems on Vector and Parallel Computers," *Parallel Computing*, Vol. 5, pp. 45–54, 1987.

[27] Gallopoulos, E. and Saad, Y., "A Parallel Block Cyclic Reduction Algorithm for the Last Solution of Elliptic Equations," *Parallel Computing*, Vol. 10, pp. 143–159, 1989.

[28] A. Krechel, H. P. and Stuben, K., "Parallelization and Vectorization Aspects of the Solution of Tridiagonal Linear Systems," *Parallel Computing*, Vol. 14, pp. 31–49, 1990.

[29] Brugnano, L., "A Parallel Solver for Tridiagonal Linear Systems for Distributed Memeory Parallel Computers," *Parallel Computing*, Vol. 17, pp. 1017–1023, 1991.

[30] M. Heath, E. N. and Peyton, B., "Parallel Algorithms for Sparse Linear Systems," *SIAM Review*, Vol. 33, pp. 420–460, 1991.

[31] Bondeli, S., "Divide and Conquer: a Parallel Algorithm for the Solution of a Tridiagonal Linear System of Equations," *Parallel Computing*, Vol. 17, pp. 419–434, 1991.

[32] Mehrmann, V., "Divide and Conquer Methods for Block Tridiagonal Systems," *Parallel Computing*, Vol. 19, pp. 257–279, 1993.

[33] N. Mattor, T. J. Williams, D. W. H., "Algorithm for Solving Tridigonal Matrix Problems in Parallel," *Parallel Computing*, Vol. 21, pp. 1769–1782, 1995.

[34] A. Basermann, B. R. and Schelthoff, C., "Preconditioned CG Methods for Sparse Matrices on Massively Parallel Machines," *Parallel Computing*, Vol. 23, pp. 381–398, 1997.

[35] Babuska, I., "Numerical Stability in Problems of Linear Algebra," *SIAM J. Numer. Anal.*, Vol. 9, pp. 53–77, 1972.

[36] Zhong, X., "Direct Numerical Simulation of Hypersonic Boundary-Layer Transition Over Blunt Leading Edges, Part I: New Numerical Methods and Validation ," AIAA paper 97-0755, Jan. 1997.

[37] Zhong, X., "Direct Numerical Simulation of 3-D Hypersonic Boundary Layer Receptivity to Freestream Disturbances," *AIAA paper 98-0533, 36th AIAA Aerospace Sciences Meeting and Exhibit, January 12-15, Reno, Nevada*, 1998.

[38] Zhong, X., "High-Order Finite-Difference Schemes for Numerical Simulation of Hypersonic Boundary Layer Transiton," *Journal of Computational Physics*, Vol. 144, August 1998, pp. 662-709.

[39] Dong, H. and Zhong, X., "High-Order Semi-Implicit Simulation of Hypersonic Boundary Layer Stability and Transition," *AIAA paper 98-0127, 36th AIAA Aerospace Sciences Meeting and Exhibit, January 12-15, Reno, Nevada*, 1998.

[40] Zhong, X. and Dong, H., "Hypersonic Boundary-Layer Receptivity to Freestream Disturbances over an Elliptical Cross-Section Cone," *AIAA paper 99-0409*, January 1999.

[41] Zhong, X., "Direct Numerical Simulation of Hypersonic Boundary-Layer Transition Over Blunt Leading Edges, Part II: Receptivity to Sound ," AIAA paper 97-0756, Jan. 1997.

[42] Zhong, X., "Additive Semi-Implicit Runge-Kutta Schemes for Computing High-Speed Nonequilibrium Reactive Flows," *Journal of Computational Physics*, Vol. 128, 1996, pp. 19-31.

[43] Collis, S. S. and Lele, S. K., "A Computational Approach to Swept Leading-Edge Receptivity," *AIAA paper 96-0180*, 1996.

[44] Guo, Y., Kleiser, L., and Adams, N. A., "A Comparision Study of an Improved Temporal DNS and Spatial DNS of Compressible Boundary Layer Transition," *AIAA paper 94-2371*, June 1994.

[45] Isreali, M. and Orszag, S. A., "Approximation of Radiation Boundary Condition," *Journal of Computational Physics*, Vol. 41, 1981, pp. 115–135.

[46] Orszag, S. A., Israeli, M., and Deville, M. O., "Bounary Conditions for Incompressible Flows," *Journal of Scientific Computing*, Vol. 1, No. 1, 1986, pp. 75–111.

[47] Givoli, D., "Non-reflecting boundary Conditions," *Journal of Computational Physics*, Vol. 94, 1991, pp. 1–29.

[48] Streett, C. L. and Macaraeg, M. G., "Spectral Multi-Domain for Large-Scale Fluid Dynamic Simulations," *Applied Numerical Mathematics*, Vol. 6, 1989/90, pp. 123–139.

[49] Poinsot, T. J. and Lele, S. K., "Boundary Conditions for Direct Simulations of Compressible Viscous Flows," *Journal of Computational Physics*, Vol. 101, 1992, pp. 104–129.

[50] Hu, S. and Zhong, X., "Linear stability of viscous supersonic plane Couette flow," *Physics of Fluids*, Vol. 10, No. 3, March 1998.

Table 2: Comparisons of execution time, speedup and parallel efficiency for different partitioning in solving model equations by using all-to-all broadcast solver.

| Partitioning | Excution Time | Speedup | Efficiency(%) |
|---|---|---|---|
| 1 | 1768.4 | 1 | 100 |
| $1 \times 2$ | 902.2 | 1.96 | 98 |
| $1 \times 4$ | 515.6 | 3.43 | 85.83 |
| $1 \times 8$ | 306.5 | 5.77 | 72.12 |
| $1 \times 16$ | 166.1 | 10.65 | 66.56 |
| $1 \times 20$ | 171.1 | 10.34 | 51.7 |

Table 3: Comparisons of execution time, speedup and parallel efficiency for different partitioning in solving model equations by using folded skip-decoupling solver.

| Partitioning | Excution Time | Speedup | Efficiency(%) |
|---|---|---|---|
| 1 | 1771.6 | 1 | 100 |
| $1 \times 2$ | 903.9 | 1.96 | 98 |
| $1 \times 4$ | 499.0 | 3.55 | 88.74 |
| $1 \times 8$ | 271.3 | 6.53 | 81.63 |
| $1 \times 16$ | 151.3 | 11.71 | 73.17 |
| $1 \times 20$ | 127.5 | 13.90 | 69.48 |

Table 4: Efficiency comparisons between parallelized explicit method and parallelized semi-implicit method for the simulations of unsteady supersonic Couette flow. The grids of each subdomain is $60 \times 60 \times 4$.

| | Explicit Method | Semi-Implicit | Ratio |
|---|---|---|---|
| CFL | 0.2 | 14.5 | - |
| time(sec) | $2.1076 \times 10^{-5}$ | $2.1076 \times 10^{-5}$ | - |
| # of nodes | CPU time(sec) | CPU time(sec) | - |
| 1 | 6,716.13 | 797.64 | 8.42 |
| 6 | 6,947.25 | 1,057.42 | 6.57 |
| 12 | 7,364.53 | 1,490.80 | 4.94 |
| 18 | 7,688.47 | 2,248.09 | 3.42 |
| 20 | 7,823.96 | 2,399.86 | 3.26 |

Table 5: Efficiency comparisons between parallelized explicit method and parallelized semi-implicit method for the simulations of receptivity of hypersonic boundary layer over blunt wedge. The total grids of computational domain is $320 \times 240 \times 4$.

| | Explicit Method | Semi-Implicit | Ratio |
|---|---|---|---|
| CFL | 0.085 | 1.46 | - |
| time(sec) | $1.7268 \times 10^{-7}$ | $1.7268 \times 10^{-7}$ | - |
| Partitioning | CPU time(sec) | CPU time(sec) | - |
| $2 \times 1$ | 18,734.3 | 2,779.6 | 6.74 |
| $2 \times 3$ | 6,416.6 | 1,217.5 | 5.27 |
| $2 \times 6$ | 3,373.4 | 728.6 | 4.63 |
| $2 \times 8$ | 2,612.7 | 628.1 | 4.16 |
| $2 \times 10$ | 2,109.1 | 529.9 | 3.98 |

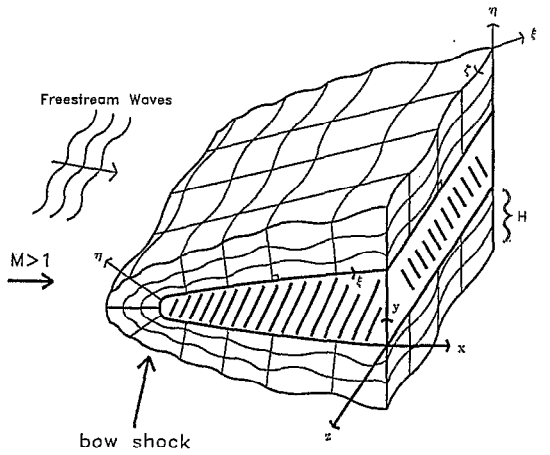


Figure 1: A schematic of 3-D shock fitted grids for the direct numerical simulation of hypersonic boundary-layer receptivity to freestream disturbances over a blunt leading edge.

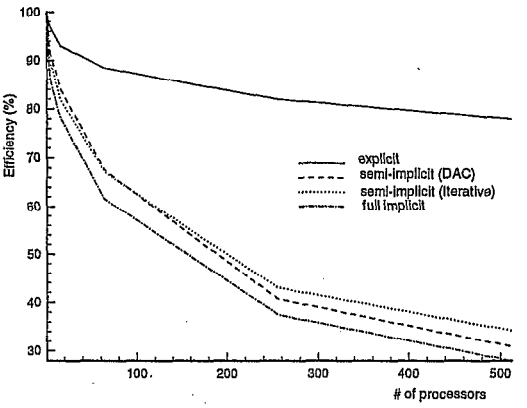


Figure 2: Comparison of efficiency of different parallelized algorithms by using different number of processors on model equation study. (Subdomain grids $61 \times 61$).

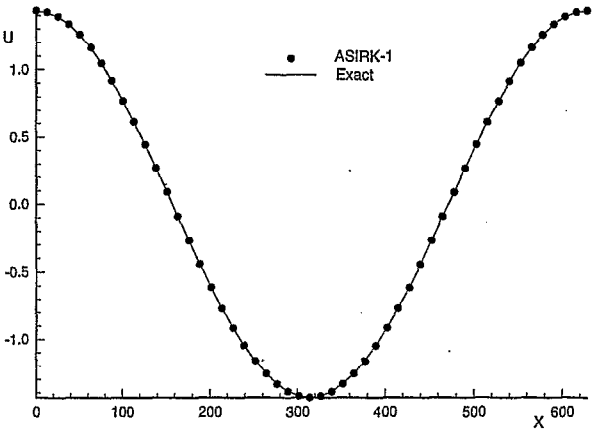


Figure 3: Comparisons of the distribution of transient solution in x direction between numerical result and exact solution ($y = 0.16, t = 0.5335$).

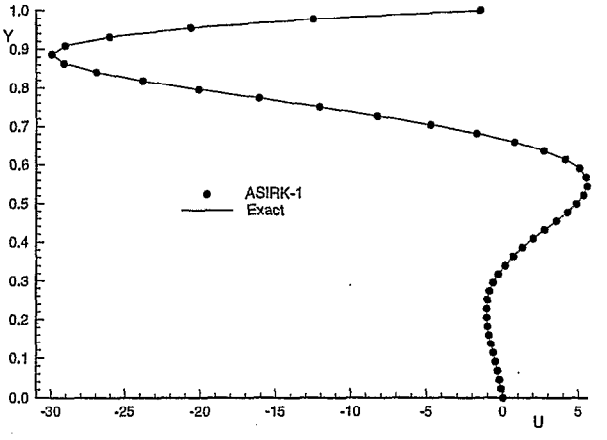


Figure 4: Comparisons of the distribution of transient solution in y direction between numerical result and exact solution ($x = 0, t = 0.5335$).

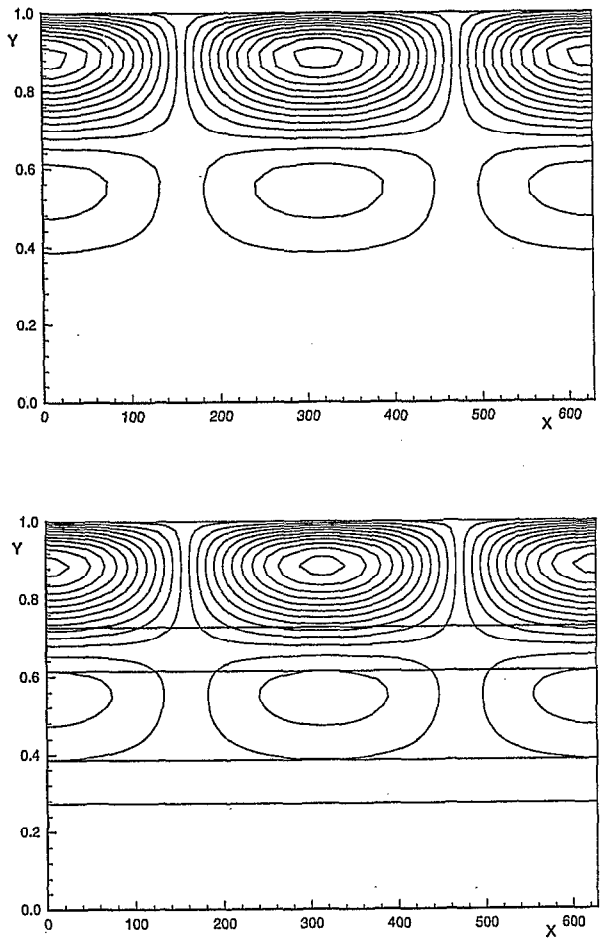


Figure 5: Contours of instantaneous solution at $t = 0.5335$, exaction solution (upper figure), and parallelized semi-implicit solution (lower figure) by using 3 processors.

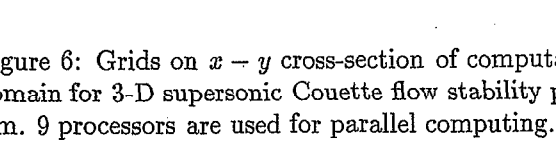

Figure 6: Grids on $x - y$ cross-section of computation domain for 3-D supersonic Couette flow stability problem. 9 processors are used for parallel computing.

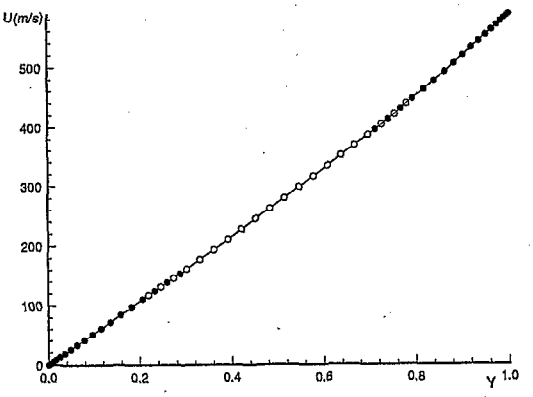


Figure 7: Comparison of the temperature distribution in $y$ direction between parallelized semi-implicit result and exact solution (solid line: exact solution, circle: numerical result, 3 processors in $y$ direction)

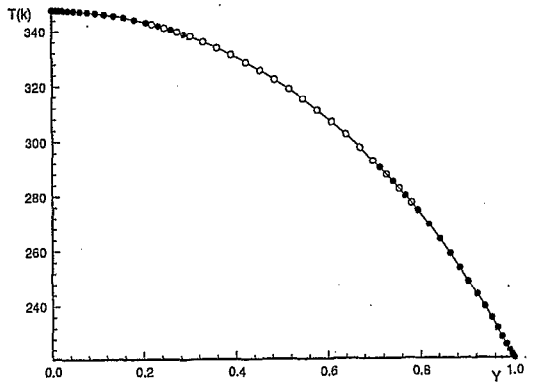


Figure 8: Comparison of the temperature distribution in $y$ direction between numerical result and exact solution (solid line: exact solution, circle: numerical result, 3 processors in $y$ direction)

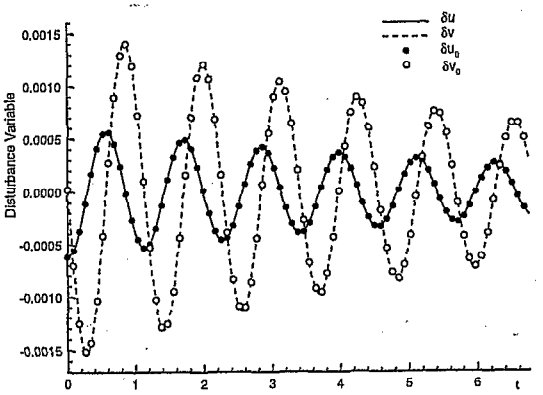


Figure 9: Time history of velocity perturbations at a fixed point in the 3-D supersonic Couette flow field (LST: $\delta u_0$ and $\delta v_0$, DNS: $\delta u$ and $\delta v$). $M = 2.0$, $Re = 1,000$.

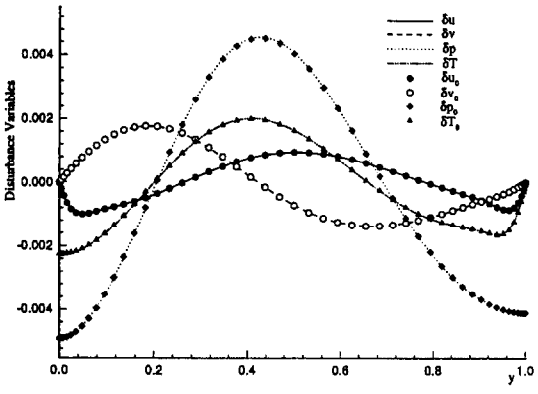


Figure 10: Distribution of instantaneous flow perturbations in y direction. (LST: $\delta u_0$, $\delta v_0$, $\delta p_0$ and $\delta T_0$ DNS: $\delta u$, $\delta v$, $\delta p$ and $\delta T$).

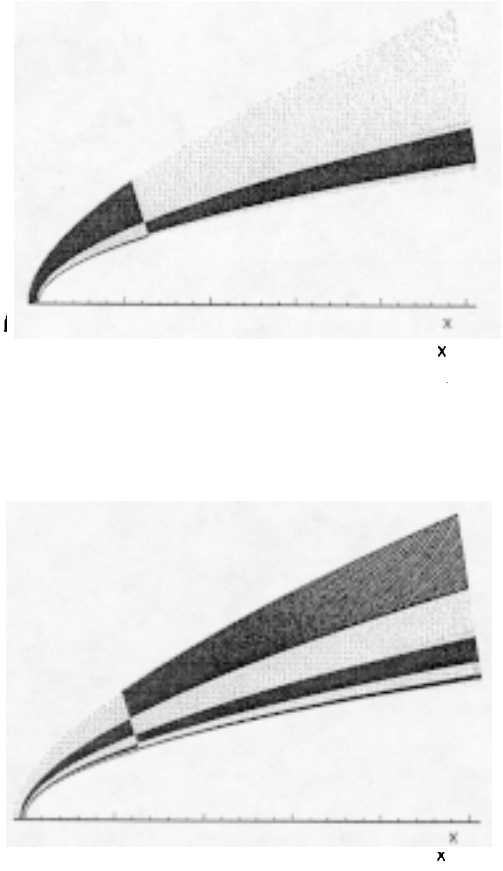


Figure 11: Computation grids for simulation of hypersonic flow over a blunt wedge using 3 processors in $y$ direction (upper figure) and 6 processors in $y$ direction (lower figure). Different colors represent the different subdomains.

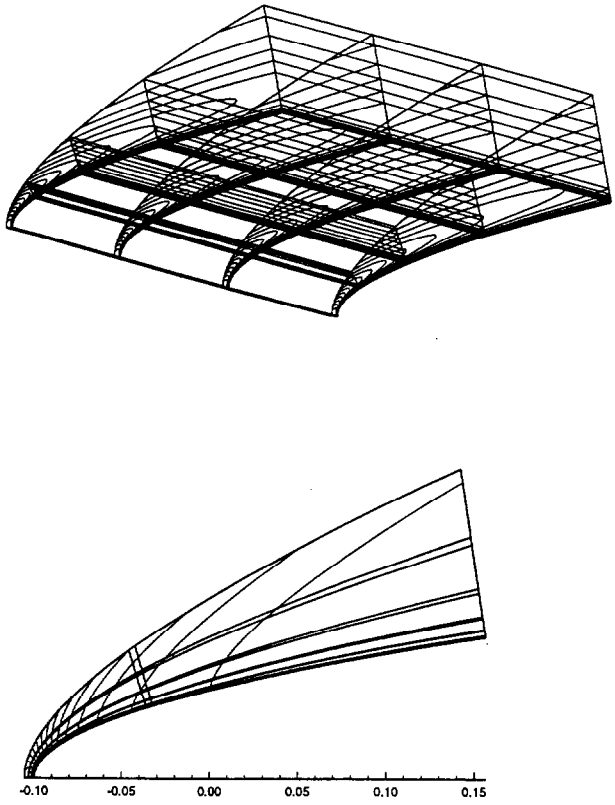


Figure 12: 3-D temperature contours (upper figure) and pressure contours on one cross-section (lower figure) of steady flow solutions for hypersonic flow over a blunt wedge by using high-order parallelized code.

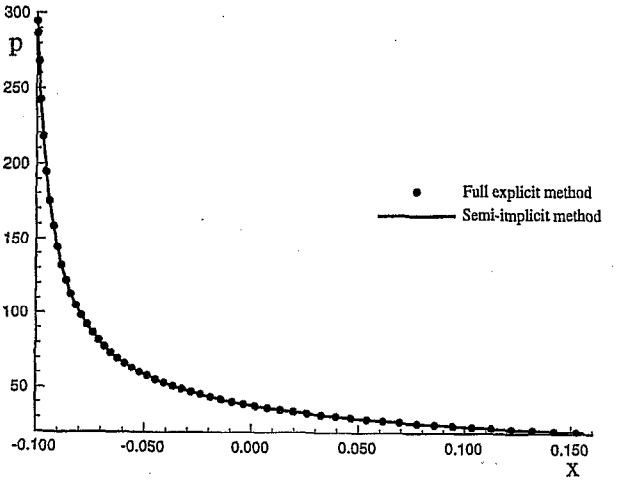


Figure 13: Comparison of steady solution of the pressure profile along the body surface between parallelized explicit method and parallelized semi-implicit method.

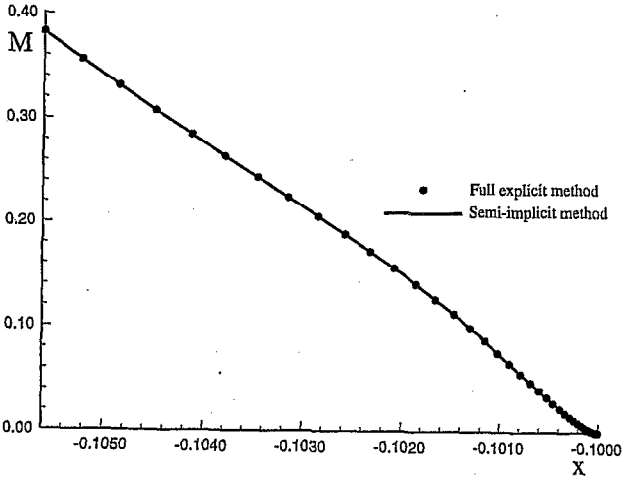


Figure 14: Comparison of steady solution of the Mach number along the stagnation line between parallelized explicit method and parallelized semi-implicit method.

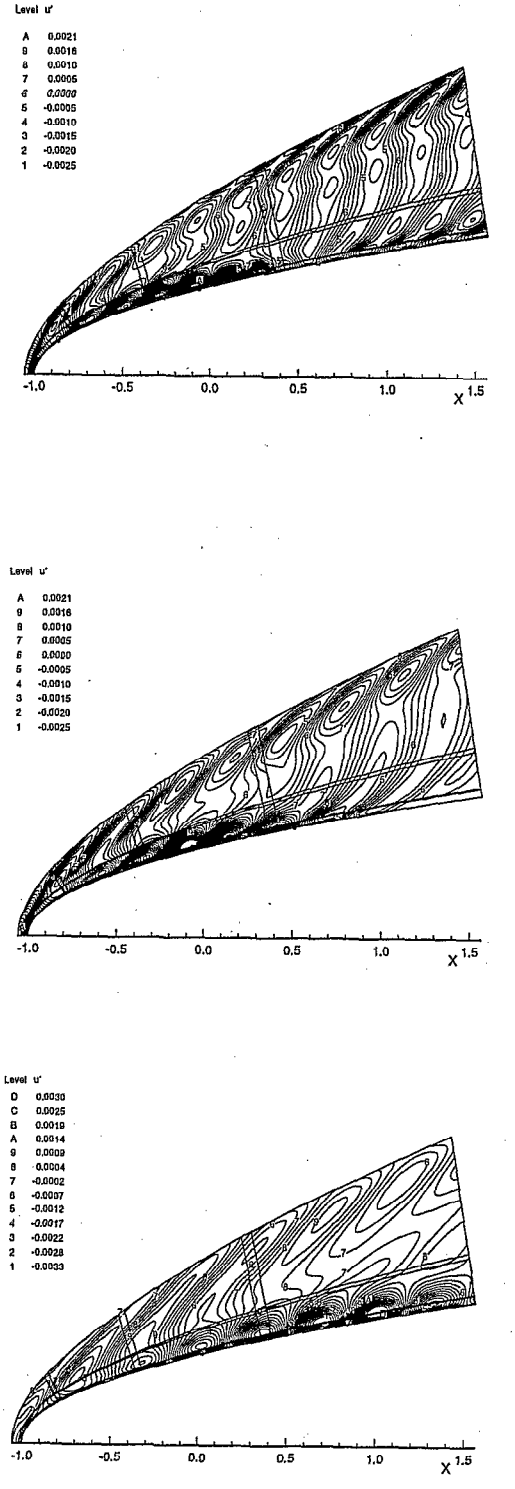


Figure 15: Instantaneous $u'$ contours for the receptivity to freestream disturbances for 3-D hypersonic boundary-layer over a parabolic leading edge. $\psi = 30^0$ (upper figure), $\psi = 45^0$ (middle figure), $\psi = 60^0$ (lower figure).

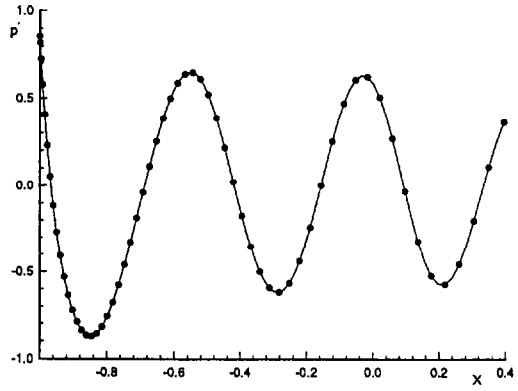


Figure 16: Comparison of distribution of instantaneous pressure perturbations along the parabola surface between the parallelized explicit method and parallelized semi-implicit method. (solid line: semi-implicit results, dot: explicit results)

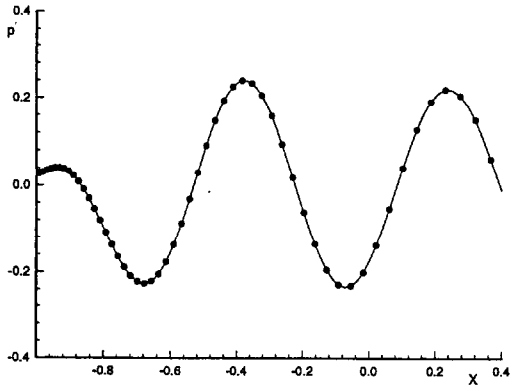


Figure 17: Comparison of distribution of instantaneous pressure perturbations behind the bow shock between the parallelized explicit method and parallelized semi-implicit method. (solid line: semi-implicit results, dot: explicit results)