NASA Contractor Report 191513
ICASE Report No. 93-53

ICASE

PARALLEL SPATIAL DIRECT NUMERICAL SIMULATIONS ON THE INTEL IPSC/860 HYPERCUBE

Ronald D. Joslin
Mohammad Zubair

NASA Contract No. NAS1-19480
August 1993

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23681-0001

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681-0001

PARALLEL SPATIAL DIRECT NUMERICAL SIMULATIONS ON THE INTEL IPSC/860 HYPERCUBE

Ronald D. Joslin
NASA Langley Research Center, Hampton, VA 23681

and

Mohammad Zubair[1]
Old Dominion University, Norfolk, VA 23529

ABSTRACT

The implementation and performance of a parallel spatial direct numerical simulation (PSDNS) approach on the Intel iPSC/860 hypercube is documented. The direct numerical simulation approach is used to compute spatially evolving disturbances associated with the laminar-to-turbulent transition in boundary-layer flows. The feasibility of using the PSDNS on the hypercube to perform transition studies is examined. The results indicate that the DNS approach can effectively be parallelized on a distributed-memory parallel machine. By increasing the number of processors, nearly ideal linear speedups are achieved with nonoptimized routines; slower than linear speedups are achieved with optimized (machine-dependent library) routines. This slower than linear speedup results because the FFT routine dominates the computational cost and because the FFT routine indicates less than ideal speedups. However, with the machine-dependent routines, the total computational cost decreases by a factor of 4 to 5 compared with standard Fortran routines. The computational cost increases linearly with spanwise, wall-normal, and streamwise grid refinements. The hypercube with 32 processors was estimated to require approximately twice the amount of Cray supercomputer single processor time to complete a comparable simulation; however, it is estimated that a subgrid-scale model, which reduces the required number of grid points and becomes a large-eddy simulation (PSLES), would reduce the computational cost and memory requirements by a factor of 10 over the PSDNS. This PSLES implementation would enable transition simulations on the hypercube at a reasonable computational cost.

[1] This research was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the second author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681.

1. INTRODUCTION

The state of the three-dimensional boundary-layer flow on the wings and fuselage of an aircraft determines the viscous drag portion of the total drag of the aircraft. This viscous drag, which is flow-state dependent, can amount to 40 or 50 percent of the total drag. (See Bushnell et al., 1977.) Any decrease in the viscous drag can lead to reduced fuel expenditures. This fuel savings can translate directly into reduced operating costs each year for the industry in terms of millions (if not billions) of dollars. The flow field on a wing can be in a laminar, turbulent, or transitional (an intermediate state that indicates transition from a laminar to a turbulent flow state) state. Because a laminar flow state yields less viscous drag than a turbulent flow state, laminar flow on the wings is preferable and results in a net fuel savings. Today, turbulent flows engulf most of the wing area of commercial aircraft. Clearly, any aircraft manufacturer that successfully designs an aircraft with "laminar flow wings" (i.e., wings covered primarily by laminar flow) will have an enormous advantage.

As yet, the transition from laminar to turbulent flow is not completely understood. The first reasonably comprehensive method for predicting transition was derived from stability theory, which is the e^N method by Smith and Gamberoni (1956) and Van Ingen (1956). Although the e^N method is widely used to predict transition in a broad class of flows, it does have some limitations: a quasi-parallel boundary layer is assumed; no amplitude information about the ingested disturbance in the boundary layer is taken into account; and the method is semiempirical, which requires some foreknowledge of the flow in transition. The true physical problem involves the ingestion of disturbances that interact in a nonlinear manner in the later stages of transition and are imbedded in a growing boundary layer. Consequently, a method that accounts for nonparallel flow and nonlinear interactions is necessary to predict transition.

Recently, Herbert and Bertolotti (1987) have devised a nonlinear, nonparallel computational method that is based on the parabolized stability equations (PSE). With some success,

Malik and Li (1992) have extended the PSE approach to compute crossflow disturbances in swept Hiemenz flow. Validation of this new approach for a broad class of flows will continue throughout this decade. Before the development of this theory, the only approach to solve the nonparallel, nonlinear boundary-layer transition problem was by direct numerical simulation (DNS). To date, most studies with DNS have been limited to the temporal formulation, in which a spatially periodic computational domain travels with the disturbance and the temporal evolution of the disturbance is computed. This method has enabled the extension of the simulations into the later stages of transition (Zang and Hussaini, 1987, 1990; and Laurien and Kleiser, 1989) and has provided a database of qualitative information that unfortunately lacks the physically realistic spatial representation. Spatial DNS computes spatially evolving disturbances and can provide needed quantitative information about transition. Progress in spatial DNS has been made by, among others, Danabasoglu et al. (1990, 1991) for channel flows; Fasel (1976), Spalart (1989), Fasel et al. (1990), Rai and Moin (1991a, 1991b), Bestek et al. (1992), and Joslin et al. (1992, 1993a) for boundary-layer flows; and Joslin et al. (1993b) for swept-wing flows. For a more complete list of accomplishments in transition prediction with DNS, refer to the reviews by Kleiser and Zang (1991) and Reed (1993). Enormous speed and memory requirements are necessary for spatial DNS because of the large domains and intensive computations that are involved.

Machines that can process large amounts of data at faster speeds are in ever increasing demand. Two possibilities exist for achieving high computational speeds: technological advancements and parallel computations. Technological advancements alone will not provide the desired computational speed because certain intrinsic physical limitations are being reached. An important limitation is the cycle speed, which is governed by the propagation speed of the signal in the given media. For example, the Cray 1 (delivered in 1976) had a cycle time of 12.5 nsec, and the Cray 2 (delivered in 1987) had a cycle time of 4.1 nsec. Although 11 years elapsed, an improvement of only a factor of 3 in processor speed has been

achieved. Parallel computation is a more attractive alternate approach because the cost and size of computer components decrease by an order of magnitude compared with Cray-type supercomputers, with only an incremental decrease in component speed. The real advantage to parallel computing is the increase in computational speed that occurs as the number of processors are increased.

A large body of literature covers the treatment of numerical algorithms for vector and parallel computers. (See Ortega and Voigt, 1988.) In most cases, the numerical treatments have focused on simplified problems. In other cases, algorithms for computationally intensive kernels in isolation from the entire application have been studied. These studies are all necessary; however, an efficient scheme for a kernel does not necessarily result in the efficient implementation of the whole scientific application. Typically, an entire application consists of a number of kernels with different data-distribution requirements, which necessitates data movement between two kernels. This movement can considerably degrade the performance of the entire application on a parallel machine.

The exploitation of parallelism for real-world computations becomes an even greater challenge in the absence of tools that transform sequential codes into parallel codes. A number of issues must be considered in the implementation of an entire application on a parallel computer. Some of these issues include

Data Mapping. The data distribution among the various processors of a parallel machine is a key factor in the efficiency of a parallel implementation. For many scientific applications, the optimal data distribution is not obvious and requires experimentation.

Communication Requirement. The choice of the algorithm and the data distributions determines the communication requirements of an application. Two factors are treated separately: the communication volume and the frequency of communication. Frequent interprocessor communication is not desirable on parallel machines because of the high communication-setup overheads. On such machines, large messages and infrequent commu-

nication are preferable.

Problem Granularity. For a given number of processors, a problem granularity exists below which parallelization is not effective. The problem granularity depends on the hardware and software characteristics of the parallel machine.

Single Node Performance. The efficient implementation of a code on a single node is important because any improvement in the computational performance on a single processor will have a multiplicative effect on the overall parallel performance.

In summary, the whole scientific application needs to be implemented carefully to obtain desirable performance on parallel machines.

Scientific applications can be clearly categorized according to their suitability for parallel implementation; however, many scientific and engineering applications fall into grey areas where the suitability of parallelization for the application is not clear. In most cases, the application must be implemented and tested to determine its suitability for parallel computations. These applications include a variety of diverse numerical approaches. Some examples of these applications are: Fischer et al. (1988) and Henderson and Karniadakis (1991), who discussed the use of spectral element methods to characterize the unsteady Navier-Stokes equations; Otto (1993), who studied chemical reactions in a computational fluid dynamics code; Jackson et al. (1991), who studied incompressible turbulence with a temporal DNS code; and Edison and Erlebacher (1993), who used a fully balanced tridiagonal solver (all three directions) in a temporal DNS code to study compressible, isotropic turbulence.

The goal of the present research is to modify a spatial DNS approach described by Joslin et al. (1993a) to perform boundary-layer transition computations on parallel computers. The suitability of the DNS code for parallelization on a distributed-memory parallel machine, the Intel iPSC/860, is examined.

2. GOVERNING EQUATIONS

To compute the disturbance development, the incompressible Navier-Stokes equations are solved. The streamwise direction is x, the wall-normal direction is y, and the spanwise direction is z. A sketch of the computational domain is shown in Fig. 1. Instantaneous velocities $\underline{\tilde{u}} = (\tilde{u}, \tilde{v}, \tilde{w})$ and pressure $\tilde{p}$ are decomposed into the base components $\underline{U} = (U, V, W)$ and P and the disturbance components $\underline{u} = (u, v, w)$ and p so that

$$\underline{\tilde{u}}(\underline{x}, t) = \underline{U}(\underline{x}) + \underline{u}(\underline{x}, t) \quad \text{and} \quad \tilde{p}(\underline{x}, t) = P(\underline{x}) + p(\underline{x}, t) \tag{1}$$

where $\underline{x} = (x, y, z)$ and t is time.

The base flow is generally the steady-state solution of the Navier-Stokes equations. For simplicity, this study will use the Blasius similarity profiles for the base flow of a flat-plate transition problem.

To determine the disturbance component of the instantaneous velocities and the pressure, substitute equation (1) into the Navier-Stokes equations and subtract the base-flow equations. The resulting unsteady, nonlinear disturbance equations are

$$\frac{\partial \underline{u}}{\partial t} + (\underline{u} \cdot \nabla)\underline{u} + (\underline{U} \cdot \nabla)\underline{u} + (\underline{u} \cdot \nabla)\underline{U} = -\nabla p + \frac{1}{R_{\delta_o^*}} \nabla^2 \underline{u} \tag{2}$$

and the continuity equation is

$$\nabla \cdot \underline{u} = 0 \tag{3}$$

Boundary conditions at the wall and in the far field are

$$\underline{u} = 0 \quad \text{at} \quad y = 0 \quad \text{and} \quad \underline{u} \to 0 \quad \text{as} \quad y \to \infty \tag{4}$$

The equations have been nondimensionalized with respect to the free-stream velocity U_∞, the kinematic viscosity ν, and some length scale at the inflow (e.g., displacement thickness δ_o^*). A Reynolds number can then be defined as $R_{\delta_o^*} = U_\infty \delta_o^* / \nu$.

3. NUMERICAL TECHNIQUES

In the streamwise direction (x-direction), fourth-order central finite differences are used for the pressure equation. At boundary and near-boundary nodes, fourth-order differences are used. For the first and second derivatives in the momentum equations, sixth-order compact differences by Lele (1992) are used. At the boundary and near-boundary nodes, explicit fifth-order finite differences are used. The compact differences lead to tridiagonal systems, and the central finite differences lead to a pentadiagonal system; both of these systems can be solved efficiently by an LU-decomposition with the appropriate backward and forward substitutions.

In the wall-normal direction (y-direction), a Chebyshev series is used to approximate the disturbance at the Gauss-Lobatto collocation points. Because this series and its associated spectral operators are defined on [-1, 1] and the physical problem of interest has either a semi-infinite $[0, \infty)$ or a truncated domain $[0, y_{\max}]$, a transformation is employed. Studies of spectral methods and mapping transformations in unbounded regions have been conducted by Grosch and Orszag (1977) and Boyd (1989). Here, an algebraic mapping is used:

$$y = \frac{y_{\max} s_p (1 + \overline{y})}{2 s_p + y_{\max}(1 - \overline{y})} \quad \text{or} \quad \overline{y} = \frac{(2 s_p + y_{\max}) y - y_{\max} s_p}{y_{\max}(s_p + y)} \tag{5}$$

where $y \in [0, y_{\max})$ and $\overline{y} \in [-1, 1]$; $y_{\max}$ is the wall-normal distance from the wall to the far-field boundary in the truncated domain; and s_p controls the grid stretching in the wall-normal direction. The Chebyshev series operators lead to matrix-matrix multiplies.

In the spanwise direction (z-direction), periodicity is assumed, which allows for Fourier series representations. With the Fourier series, spectral accuracy is obtained in the spanwise direction, and fast Fourier transforms (FFT), or sine and cosine transforms, may be used for fast computation of derivatives. The general Fourier series leads to FFT operations, and the sine and cosine series lead to matrix-matrix multiply operations. For more details on the spectral methods used here, refer to Canuto et al. (1988).

For time marching, a time-splitting procedure was used with implicit second-order Crank-Nicolson differencing for normal diffusion terms; an explicit third-order three-stage Runge-Kutta (RK) method was used for the remaining terms. This time-stepping procedure was used successfully by Streett and Hussaini (1991) for Taylor-Couette flow simulations. The pressure is omitted from the momentum equations (2) for the fractional RK stage, which leads to

$$\frac{\partial \underline{u}^*}{\partial t} + (\underline{u}^* \cdot \nabla)\underline{u}^* + (\underline{U} \cdot \nabla)\underline{u}^* + (\underline{u}^* \cdot \nabla)\underline{U} = \frac{1}{R_{\delta_o^*}} \nabla^2 \underline{u}^* \tag{6a}$$

with boundary conditions

$$\underline{u}^* = \underline{u}_\tau^* \tag{6b}$$

Time is advanced from $\underline{u}^m$ to the intermediate disturbance velocities $\underline{u}^*$, and $\underline{u}_\tau^*$ are intermediate boundary conditions that will be explained later in this section.

A full RK stage is completed by advancing the solution in time from $\underline{u}^*$ to $\underline{u}^{m+1}$ by

$$\frac{\partial \underline{u}}{\partial t}^{m+1} = -\nabla p^{m+1} \tag{7a}$$

and

$$\nabla \cdot \underline{u}^{m+1} = 0 \tag{7b}$$

By taking the divergence of equation (7) and imposing zero divergence of the flow field at each RK stage, a pressure equation is obtained

$$\nabla^2 p^{m+1} = \frac{1}{h_t^m} (\nabla \cdot \underline{u}^*) \tag{8}$$

which is subject to homogeneous Neumann boundary conditions; h_t^m are time-step sizes in the RK scheme. This boundary condition is justified in the context of a time-splitting scheme as discussed by Streett and Hussaini (1991).

Because the pressure equation (8) is an inviscid calculation and involves boundary conditions on the normal component of velocity only, a nonzero tangential velocity component

may arise at the computational boundary at the end of each full RK time-step, which is referred to as a "slip velocity." To correct this problem, intermediate boundary conditions as described by Streett and Hussaini (1991) and Joslin et al. (1992) are used, which are given by

$$\underline{u}_\tau^* = \underline{u}_{BC} + h_t^m \left[(1 + \frac{h_t^m}{h_t^{m-1}}) \nabla p_\tau^m - \frac{h_t^m}{h_t^{m-1}} \nabla p_\tau^{m-1} \right] \tag{9}$$

where $\underline{u}_{BC} = 0$ for a rigid wall and $\underline{u}_{BC} = \underline{u}_o$ for an inflow or a wall slot condition, evaluated at the appropriate time in the RK stage.

The solution procedure follows: The intermediate RK velocities $\underline{u}^*$ are determined by solving equation (6). The pressure p^{m+1} is found by solving equation (8). Then, the full RK stage velocities $\underline{u}^{m+1}$ are obtained from equation (7). After the above system is solved three consecutive times, full time-step velocities result. The three-stage RK time steps given by Williamson (1980) are $\{h_t^1, h_t^2, h_t^3\} = \{1/3, 5/12, 1/4\} h_t$, where the sum of the three RK time stages equals the full time step (h_t).

To obtain the pressure p for the two- and three-dimensional boundary-layer problems, solutions of the Poisson equation (8) for each RK stage are required. For three-dimensional simulations with spanwise periodicity assumed, the pressure is determined in transform space, where the Fourier coefficients are evaluated.

For the two-dimensional problem and the zero-wave-number component of the three-dimensional problem, the Poisson equation with Neumann boundary conditions is equivalent to the composite solution of a Poisson problem and a Laplace problem with Dirichlet boundary conditions (Streett and Hussaini, 1991):

$$\nabla^2 p^I = R_o \quad \text{in} \quad \Gamma \qquad p^I = 0 \quad \text{on} \quad \partial\Gamma \tag{10a}$$

$$\nabla^2 p^{II} = 0 \quad \text{in} \quad \Gamma \qquad p^{II} = I_{NF}^{-1} \cdot p_n^I \quad \text{on} \quad \partial\Gamma \tag{10b}$$

where p_n^I are gradients normal to the boundaries, R_o is the known zero-wave-number Fourier coefficient of the right side of the pressure equation (8), and I_{NF} is the influence matrix.

This boundary condition gives the influence of the right side R_o on the boundary. The final solution $p = p^I - p^{II}$ satisfies the original problem and the boundary conditions. The influence-matrix technique is also used for the pressure solver to ensure that the continuity equation is discretely satisfied. Details of the influence-matrix technique are given by Streett and Hussaini (1991), Danabasoglu et al. (1991), and Joslin et al. (1992). To form the influence matrix, a sequence of solutions is first determined for a problem

$$\nabla^2 p^i = 0 \quad \text{in} \quad \Gamma \qquad p^i = \delta_{i,j} \quad \text{on} \quad \partial\Gamma \tag{11}$$

for each discrete boundary point $(\overline{x}_j)$. The $\delta_{i,j}$ is the Dirac-Delta function defined as $\delta_{i,j} = 1$ for $i = j$ and $\delta_{i,j} = 0$ for $i \neq j$. After the vector of normal gradients p_n^i is computed at all of the boundary points, these vectors are stored in columns to yield a matrix referred to as the influence matrix

$$I_{NF} = [p_n^1, p_n^2, \ldots, p_n^{N_B}] \tag{12}$$

where N_B is the number of boundary points minus the corner points. The composed influence matrix gives the residuals of p as a result of the unit boundary-condition influence. The value of one boundary condition is temporarily relaxed so that the problem is not overspecified. This relaxation is accomplished by setting one column of the influence matrix to zero, except for the boundary point of interest, which is set to unity. The corresponding residual is exactly zeroed.

Because the gradient, or boundary condition, at one discrete boundary point was relaxed in the influence-matrix formulation, the desired condition ($p_n = 0$) may not hold at that boundary point. The desired condition may not hold because the discrete compatibility relation may not be true for the pure Neumann problem. To regain this boundary condition, the pressure problem (10) is solved again; this time a nonzero constant (e.g., 0.01) is added to the right side of equation (10a). A pressure correction $\overline{p}$ results. The composite solution satisfies the boundary conditions at all discrete nodes and consists of a linear combination

of p and $\overline{p}$. This combination is found by satisfying the equations

$$a_1 p_n + a_2 \overline{p}_n = 0 \quad \text{on} \quad \partial\Gamma_i \qquad \text{and} \qquad a_1 + a_2 = 1 \tag{13}$$

The final pressure (p^{m+1}) is then given by

$$p^{m+1} = a_1 p + (1 - a_1)\overline{p} \quad \text{with} \quad a_1 = \overline{p}_n / (\overline{p}_n - p_n) \tag{14}$$

In transform space, Poisson equations are solved for the zero-wave-number component; for the remaining wave numbers, Helmholtz equations are solved. To solve the equations efficiently, a fast elliptic solver is required. For this purpose, the tensor-product method described by Lynch et al. (1964) is used. On a nonstaggered grid, this approach was employed by Danabasoglu et al. (1990, 1991) for the channel problem and by Joslin et al. (1992) for the boundary-layer problem.

The discretized equations become

$$(H_n - \beta_n^2) p_n + p_n X^T = R_n \tag{15}$$

where p_n are nonzero Fourier coefficients of the desired pressure solution p, H_n are the wall-normal derivative operators, X^T is the transpose of the streamwise central finite-difference operator, $\beta_n = n\beta$ are spanwise wave-number coefficients of the Fourier series, and R_n are known Fourier coefficients of the right side of the pressure equation (8).

The following matrix operations determine the wall-normal operators H_n:

$$H_o = \hat{D}^2 \qquad \text{and} \qquad H_n = I_{GL}^{G} D \tilde{D} I_{G}^{GL} \tag{16}$$

where D is a spectral wall-normal derivative operator for the stretched grid, $\tilde{D}$ is the derivative matrix with the first and last rows set to zero, and $\hat{D}^2$ is the spectral wall-normal second derivative operator with the boundary conditions contained in the first and last rows. This modification enforces the homogeneous Neumann boundary conditions required for p on the

Gauss-Lobatto grid. The zero-wave-number equation is solved on Gauss-Lobatto points; all other wave-number equations are solved on Gauss points. The interpolation matrix I_{GL}^{G} operates on variables at Gauss-Lobatto points and transforms them to Gauss points; the interpolation matrix I_{G}^{GL} performs the inverse operation. These operators are defined by Joslin et al. (1993a).

The operator H_n is decomposed into

$$H_n = Q\Lambda Q^{-1} \tag{17}$$

where Λ is a diagonal matrix of eigenvalues and Q is the corresponding matrix of eigenvectors. Temporary matrices are introduced and defined

$$\hat{p}_n = Q^{-1} p_n \quad \text{and} \quad G_n = Q^{-1} R_n \tag{18}$$

Equations (17) and (18) are substituted into equation (15) to obtain

$$\Lambda \hat{p}_n + \hat{p}_n X^T = G_n \tag{19}$$

Because X^T is a fourth-order accurate, pentadiagonal matrix, the LU-decomposition method is used and provides an efficient method to solve the pressure equation (19). Equation (19) is used to solve for $\hat{p}_n$, which is then used in equation (18) to solve for p_n.

The operators H_n, the eigenvalue and eigenvector matrices Q, Q^{-1}, Λ, the pentadiagonal operator X^T, and the influence matrix I_{NF} are all mesh-dependent matrices and need to be calculated only once.

Here, disturbances are introduced into the boundary layer by forcing at the inflow boundary; however, the swept-wing problem of Joslin et al. (1993b) used surface suction and blowing to introduce disturbances. At the outflow, the buffer-domain technique of Streett and Macaraeg (1989) is used.

4. PARALLEL IMPLEMENTATION OF THE SPATIAL DNS

In this section, the various data-distribution options available for implementation in the three-dimensional DNS code on a parallel machine are discussed, and the data distribution used in this application is outlined.

4.1 Data Mapping

The DNS code consists of a number of computationally intensive kernels. Dependent upon the data mapping, some of these kernels are executed locally on a single processor, and the rest are executed globally across the processors. The kernels that are executed locally do not require communication between processors; kernels that are executed globally require communication. The major computationally intensive kernels are the matrix-matrix multiplication, the FFT, the tridiagonal solver, and the pentadiagonal solver. The operation counts that correspond to the kernels are illustrated in table 1; these operation counts are for a one-time iteration of the DNS code. Of these major kernels, the matrix-matrix multiplication is the most computationally intensive kernel. Hereafter, n_p denotes the number of processors on a parallel machine, and n_x, n_y, and n_z denote the number of data items (i.e., grid points) in the streamwise, wall-normal, and spanwise directions.

For the three-dimensional problem, three major data mappings exist:

x-mapping. The three-dimensional data are partitioned into n_x two-dimensional planes of $n_y n_z$ data items each. The first n_x/n_p planes are mapped to processor p_0, the next n_x/n_p planes are mapped to processor p_1, and so on.

y-mapping. The three-dimensional data are partitioned into n_y two-dimensional planes of $n_x n_z$ data items each. The first n_y/n_p planes are mapped to processor p_0, the next n_y/n_p planes are mapped to processor p_1, and so on.

z-mapping. The three-dimensional data are partitioned into n_z two-dimensional planes of $n_x n_y$ data items each. The first n_z/n_p planes are mapped to processor p_0, the next n_z/n_p planes are mapped to processor p_1, and so on. An example of this mapping is shown in Fig.

2 for $n_p = 4$.

As stated earlier, the data mapping determines whether a particular kernel is to be executed across all processors or executed locally on a single processor. Table 2 shows the executions of the major kernels for the three data mappings. For the x-mapping, a great deal of communication is clearly required, which is undesirable. Both the y- and z-mapping are more desirable than the x-mapping because most of the kernels are executed locally. Because the operation counts shown in table 1 indicate that the matrix-matrix multiply is higher than FFT's, the z-mapping should be more efficient than the y-mapping. The z-mapping requires that one kernel (FFT) be executed globally. Our implementation of the PSDNS code on the Intel iPSC/860 is based on the z-mapping.

4.2 FFT Implementation

The FFT kernel computes $n_x n_y$ sequences of discrete Fourier transforms of size n_z. The z-mapping distributes sequences across all processors of the machine. One way to compute the discrete Fourier transform of these sequences is as follows: first, the transpose is taken of the three-dimensional data (the resulting distribution is the x-mapping of u), the one-dimensional FFT algorithm is then executed on sequences of length n_z on each processor, the results are multiplied by a coefficient array, the inverse Fourier transform of the data is then computed, and, finally, the results are transposed back to the original data distribution. In this scheme, all global data movement occurs in the transpose step. To keep communication overheads to a minimum, the transpose operation must be implemented efficiently. The transpose operation is a complete exchange operation; every node has an equivalent amount of data to exchange with every other node. The *xor* algorithm is used to implement this exchange procedure because it is the optimal procedure on the Intel iPSC/860. The *xor* algorithm, which is illustrated in table 3, schedules various exchanges at particular nodes to avoid link contention. In this scheme at the ith step, a node j sends data to node iXj.

5. PSDNS VALIDATION

The evolution of a Tollmien-Schlichting wave in the three-dimensional flow is used to validate the PSDNS approach. A disturbance with an initial amplitude $A^o_{1,0} = 1 \times 10^{-6}$ is introduced into the boundary layer by a forcing at the inflow for the PSDNS. The inflow disturbance profiles are obtained with linear stability theory (LST). The parallel-flow assumption is used for comparison with LST. Calculations are made with an inflow Reynolds number $R_{\delta^*_o} = 900$ and frequency $\omega = 0.0774$. The PSDNS was computed on a grid of 200 uniformly spaced streamwise nodes (60 nodes per disturbance wavelength), 61 wall-normal collocation points, and 8 spanwise nodes. The outflow boundary is $121\delta^*_o$ from the inflow boundary, the far-field (or free-stream) boundary is $75\delta^*_o$ from the wall, and the wall-normal grid-stretching parameter s_p is equal to 10. For the time-marching scheme, the disturbance period is divided into 320 time steps.

Figure 3 shows the streamwise evolution of the computed streamwise (u) and wall-normal (v) velocity components of PSDNS compared with LST. Very good agreement in amplitude and phase are found at every spanwise location in the physical domain. (Note, that the buffer-domain region is nonphysical and that the DNS and LST results are not expected to agree.)

6. PERFORMANCE OF PSDNS ON INTEL IPSC/860

Although direct simulations of transition involve large computational grids and many thousands of time steps per simulation, the performance of the PSDNS code can be sufficiently examined for a single time step on smaller grids. The cost and feasibility of a full-scale simulation can be estimated by using scaling information.

The range of parameters is limited to the capability of the machine. The Intel iPSC/860 hypercube at NASA Langley Research Center has 32 processors, each with 8 megabytes of memory. Because the single precision is limited to 32-bit words and because simulations of transition require the computations of small-scale phenomena, all performance test cases are

double-precision (64-bit words) computations.

The first sequence of performance simulations is computed on a grid of 64 streamwise points and 41 wall-normal points. Figure 4 shows both the computational cost (total cost minus communication) and the communication cost for each processor in CPU sec for a variation in the spanwise grid and the number of processors. For a given computational grid, a decrease in both the computational and communication cost is achieved by increasing the number of processors. For this parallel implementation, the communication cost does not exceed 6 percent of the total cost for all grids and variations in the number of processors that were considered.

Figure 5 shows the relative cost of the major numerical techniques and shows the speedup of each technique with the number of processors. As expected, the computational cost breakdown indicates that the majority of the time is spent on matrix-matrix multiply operations (table 1). The results suggest that a negligible change occurs in the cost contributions from each numerical technique with an increase in the number of processors. The speedup of each numerical technique as the number of processors increases indicates a nearly ideal linear speedup for the matrix-matrix multiply and the tridiagonal solver. Because matrix-matrix multiplies account for nearly 80 percent of the total computational cost and because the speedup for the matrix-matrix multiplies are nearly ideal, the total speedup approaches a nearly linear rate. For example, the theoretically ideal speedup rate is 4 with an increase from 8 to 32 processors and the speedup of 3.4 was realized in the total computational cost.

Figure 6 shows the computational cost and the slowdown for a spanwise grid refinement with 8 processors. Matrix-matrix multiplies dominate the PSDNS cost; matrix-matrix multiplies and tridiagonal and pentadiagonal solvers slow down at a faster rate than other major numerical techniques. For example, the matrix-matrix multiply has a theoretically ideal rate of slowdown of 8 when n_z is increased from 8 to 64. The FFT adds the least additional cost as the spanwise grid is refined from 8 to 64; the FFT results in only an increase of a factor of

3.3 in the computational cost. (The ideal slowdown rate is 8, which is undesirable because of the cost increase.)

The relative balance in work load between respective processors is an important element in documenting the PSDNS performance. Figure 7 shows the computational and communication cost for each stage of the three-stage Runge-Kutta time step for each processor of an 8-processor simulation on a spanwise grid of $n_z = 8$. In this figure, the AIMS performance software shows the work load for each processor in a separate display area. The lines that connect the processor areas indicate global communication; the shaded areas indicate the computational work; the blank spaces between shaded areas indicate idle times. The results show that all processors are load balanced, with the exception of the first node (node = 0). For the present combination of numerical techniques and parallel implementation, this is the best load balance that can be expected. Because of the influence-matrix pressure solver that is used on the first node only, additional work is always required on this node. The idle time amounts to about 20 percent of the total cost for a single time-step advancement.

In the second series of simulations, the matrix-matrix multiply is optimized because it requires over 80 percent of the total computational cost of the PSDNS on the hypercube (see Fig. 5). For a more efficient code, the matrix-matrix multiply routine would have to be improved (if possible). Instead of a standard Fortran routine, the library routine DGEMM is used here to attempt optimization. The performance simulations are repeated.

Figure 8 shows the computational and communication costs with a spanwise grid refinement and an increase in the number of processors. In comparison with Fig. 4, the trend of reduced cost as the number of processors is increased remains the same; however, the relative communication cost has become significant. Although the quantitative cost of communication is the same as before the new matrix-matrix multiply routine was introduced, the communication now equals 20 to 30 percent of the total cost because the new matrix-matrix multiply routine has reduced the total computational cost by a factor of 4 to 5 in comparison

with the original code implementation.

Figure 9 shows the relative cost of the numerical techniques and the speedup of each technique with the number of processors. With the new matrix-matrix multiply routine, the major numerical techniques are more balanced in terms of relative work load. The communication and FFT have comparable relative cost and now dominate over the other major numerical techniques. The matrix-matrix multiply and the tridiagonal solver both have nearly ideal speedups as before; however, the total speedup is only about half of the ideal rate. This decrease in efficiency occurs because the FFT routine is now the dominant numerical technique and the FFT rate of speedup is not ideal.

Figure 10 shows the numerical cost and the slowdown for a spanwise grid refinement with eight processors. The numerical techniques and communication cost are balanced; however, communication becomes dominant as the spanwise grid is refined. The slowdown rate as spanwise grid is increased are the same as those for the initial comparison (Fig. 6); however, the total slowdown rate has decreased significantly. Because the FFT routine is the dominant numerical technique and because the FFT slowdown rate is small, the total rate of slowdown closely follows the FFT rate. This result is advantageous; it indicates that little additional cost will result from refinement of the grid in the spanwise direction.

The present multiprocessor implementation of the PSDNS can be evaluated by varying the number of $x - y$ planes on each processor. For example, the use of eight spanwise grid points on an eight-processor simulation indicates that each processor performs computations on a single $x - y$ plane; eight spanwise grid points on a four processor simulation indicates that each processor performs computations on two $x - y$ planes. Figure 11 shows the cost for a single $x - y$ plane on each processor compared with the cost of four $x - y$ planes on each processor. Because FFT and global communication costs increase, cost increases are incurred in both cases with increased number of processors. Although the FFT technique and the global communication are both effected by changes in the number of processors for a

fixed number of $x-y$ planes on each processor, the costs are constant for all other techniques (Fig. 11). For four $x-y$ planes on each processor, note the initial drop in the computational cost from two to four processors.

In the remainder of this section, the streamwise and wall-normal grids will be refined to determine the cost scalings. The relative cost of the major numerical techniques and the speedup of each technique with the number of processors are shown in Fig. 12, with a streamwise grid refinement from $n_x = 64$ to $n_x = 128$. Decreases in both the computational and communication costs are achieved by increasing the number of processors. The relative communication cost accounts for 10 to 30 percent of the total computational cost.

Figure 13 shows the relative numerical cost breakdown for the dominant kernels. The numerical techniques and the communication are balanced; the FFT routine and communication dominate the cost. Similar to the speedup rates from the earlier case, the total rate of speedup (as the number of processors increases) is slower than the ideal speedup rate because the dominant FFT routine and the communication have speedup rates that are less than ideal.

Figure 14 shows the computational and communication costs with a spanwise grid variation. The results show trends similar to the $n_x = 64$ grid shown in Fig. 9. If the spanwise grid is refined by a factor of four, then a factor-of-three increase in the total computational cost results.

As in Fig. 11, Fig. 15 shows the cost of the PSDNS with a single $x-y$ plane on each processor and with four $x-y$ planes per processor. For the single $x-y$ plane on each processor, the cost increases with the number of processors in use; however, the case with four $x-y$ planes on each processor results in a decrease in the total computational cost. This result is encouraging for the performance of large-scale simulations with a large number of processors.

Figures 6, 10, and 14 show the effect of spanwise grid refinements on the computational

and communication costs. Figure 16 shows that streamwise (x) refinements lead to nearly linear theoretical increases in cost; Fig. 17 shows that wall-normal (y) refinements also lead to nearly ideal linear increases in computational cost. Although the results in Fig. 16 show that the matrix-matrix multiplies scale like n_y^2, the FFT and communication dominate the cost leading to the total cost scaling like the FFT rate.

7. DISCUSSION

The present performance data for the PSDNS suggest that insufficient core memory is a limitation of the hypercube. The largest grid that fits on a single node has 128 streamwise, 41 wall-normal, and 4 spanwise points (21,000 total grid points). An attempt to perform computations with 8 spanwise planes of this same $x - y$ grid failed because of insufficient memory. Because the code requires about 160 bytes per grid point, the 21,000-point grid used about 3.4 megabytes of memory; the failed grid required 6.8 megabytes (plus operating system). The largest grid that could potentially be used for the present PSDNS code has less than 42,000 grid points on each processor. With the code optimized for the matrix-matrix multiplies, the total cost for a single time step varied significantly with the grid and the number of processors in use. To determine if simulations of transition can be undertaken, the grid requirements must be specified to estimate the computational cost requirements. The grid resolution is highly dependent on the problem and the numerical techniques. To estimate the feasibility of using PSDNS on the hypercube, a sample transition problem computed on a single processor of a Cray-2 supercomputer is used for comparison.

In a recent study, Joslin and Streett (1993b) computed the nonlinear evolution of a crossflow vortex packet on a swept wing with spatial DNS on the Cray-2. The cost of this computation amounted to approximately 125 CPU hr with a single processor. The grid contained 901 chordwise, 61 wall-normal, and 32 spanwise points (1.76 million total points) which required 36.6 megabytes of core memory. The unsteady computation required 9500 time steps in the time-marching scheme to reach the nonlinear inflectional velocity profile

stage, which occurs just prior to the laminar-to-turbulent transition.

Each $x-y$ plane of the Joslin and Streett (1993b) study contained 55,000 grid points (8.8 megabytes of memory), which is beyond the capability of the present hypercube (8 megabytes per processor). In this case, the feasibility of using PSDNS has been easily determined by examining the memory limitation alone. However, if 16 or 32 megabytes of memory per processor were available transition studies could potentially be conducted on the hypercube. Then the feasibility of using this parallel computer would rest on the computational cost of such a simulation.

The temporal cost can be determined based on the previous performance results of a single time step. From data in Figs. 16 and 17, the rates at which computational cost increases with streamwise and wall-normal grid refinements can be determined at 1.95 and 2.15, respectively. The performance results indicate that a single time step on a grid of 64 streamwise, 41 wall-normal, and 32 spanwise points distributed on 32 processors will cost 3.7 sec for each processor. With the scaling rates, the computation by Joslin and Streett (1993b) performed on the hypercube is estimated to cost 77 sec for each processor per time step. This results in a total cost of 206 hr for each processor to achieve the nonlinear inflectional velocity profile state described by Joslin and Streett (1993b). With the dedicated use of a 32-processor hypercube with 16 megabytes of memory per processor, a simulation could be completed in approximately 9 days, which is nearly twice the cost of using a supercomputer. This comparison is a rough estimate of the total computational cost required for the simulations because only small grids can be used. On a grid with 64 streamwise, 41 wall-normal, and 32 spanwise points, the computational cost of 3.0 sec resulted on a single processor of a Cray-Y/MP; the performance was 189 megaflops. For the same grid, the computational cost on the hypercube resulted in 3.7 sec for each processor and roughly 153 megaflops.

From the estimate, simulations can apparently be performed on the hypercube, provided that each processor has at least 16 megabytes of memory. Similar to using supercomputers,

the hypercube would require a number of days to complete a single simulation. To decrease the memory and computer-cost requirements, two basic alternatives can be explored. The first alternative is an increase in the number of processors in use for a given grid; the second alternative is the reduction of the computational grid size for a given simulation. However, by decreasing the size of the grid, the PSDNS will not resolve the smaller scales (subgrid), which will degrade the results. To capture these small scales with an appropriate model, the PSDNS approach becomes a large-eddy simulation (LES) code, or PSLES. As discussed and demonstrated by Piomelli et al. (1990), LES can reduce the computational grid and cost by an order of magnitude in comparison with DNS. If a subgrid-scale mode could accurately capture the physics in the boundary-layer flow, then a PSLES on a computational grid of 256 streamwise, 41 wall-normal, and 32 spanwise points, distributed on 32 processors, could potentially be computed. The cost for a single time step would be 14 sec for each processor. For the swept-wing problem described by Joslin and Streett (1993b), the total computational cost on this LES grid would amount to 37.5 hr for each processor, or 1.5 days. Although the PSLES performance scalings are slightly underestimated because additional costs are involved with this model, PSLES seems plausible, and its use on parallel computers will be explored in the near future.

8. CONCLUDING REMARKS

The performance of a recently implemented parallel spatial direct numerical simulation (PSDNS) approach on the Intel iPSC/860 is documented. The PSDNS results are in good agreement with linear stability theory for a small-amplitude test case, which serves as the initial validation of the code on the parallel computer. The performance results show nearly ideal linear speedups, which are achieved by increasing the number of processors. The computational cost shows nearly theoretical linear increases with streamwise, wall-normal, and spanwise grid refinements. The results show that the work is well balanced between the processors (except the first node, which will have approximately 15 to 20 percent larger work

load because of the numerical techniques employed). Furthermore, a speedup with a factor of 4 to 5 was obtained by using machine-dependent libraries rather than standard Fortran routines.

The feasibility of using the PSDNS on the hypercube to compute transitional flows is assessed. A comparative study with the Cray supercomputer demonstrates that PSDNS could be used for transition studies on the hypercube, provided that each processor had 16 megabytes of memory. Furthermore, the use of a subgrid-scale model to compute large-eddy simulations (PSLES) would reduce the computational cost by an order of magnitude compared with PSDNS. Large-eddy simulations could readily be used to study transition on the hypercube at a reasonable computational cost.

ACKNOWLEDGMENTS

The authors wish to express their gratitude to Dr. Bart A. Singer, High Technology Corporation, for reviewing this manuscript. Also, thanks goes to Ms. Jonay A. Campbell, Mason and Hanger Services Incorporated, for her editorial assistance.

REFERENCES

Bestek, H., Thumm, A., and Fasel, H. F. (1992). Numerical investigation of later stages of transition in transonic boundary layers. In *First European Forum on Laminar Flow Technology*, March 16-18, 1992. Hamburg, Germany.

Boyd, J. P. (1989). Chebyshev-Fourier Spectral Methods. *Lecture Notes in Physics*, **49**, Springer-Verlag, New York.

Bushnell, D. M., Hefner, J. N., and Ash, R. L. (1977). Effect of compliant wall motion on turbulent boundary layers, *Phys. Fluids*, **20**(10), s31-48.

Canuto, C., Hussaini, M. Y., Quarteroni, A., and Zang, T. A. (1988). *Spectral Methods in Fluid Dynamics*. Springer-Verlag, New York.

Danabasoglu, G., Biringen, S., and Streett, C. L. (1990). Numerical simulation of spatially-evolving instability control in plane channel flow. *AIAA Paper No. 90-1530.*

Danabasoglu, G., Biringen, S., and Streett, C. L. (1991). Spatial simulation of instability control by periodic suction blowing. *Phys. Fluids A*, **3**(9), 2138-2147.

Eidson, T. M., and Erlebacher, G. (1993). Implementation of a fully-balanced periodic tridiagonal solver on a parallel distributed memory architecture, (to be submitted for publication in *Concurrency: Practice and Experience*).).

Fasel, H. F. (1976). Investigation of the stability of boundary layers by a finite-difference model of the Navier-Stokes equations. *J. Fluid Mech.*, **78**, 355-383.

Fasel, H. F., Rist, U., and Konzelmann, U. (1990). Numerical investigation of the three-dimensional development in boundary-layer transition. *AIAA J.*, **28**(1), 29-37.

Fischer, P. F., Ho, L.-W., Karniadakis, G. E., Ronquist, E. M., and Patera, A. T. (1988). Recent advances in parallel spectral element simulation of unsteady incompressible flows. *Computers and Structures*, **30**(1-2), 217-231.

Grosch, C. E., and Orszag, S. A. (1977). Numerical solution of problems in unbounded regions: coordinate transforms. *J. Comput. Phys.*, **25**, 273-296.

Henderson, R., and Karniadakis, G. E. (1991). Hybrid spectral-element-low-order methods for incompressible flows, *J. Sci. Comp.*, **6**(2), 79-99.

Herbert, Th., and Bertolotti, F. P. (1987). Stability analysis of nonparallel boundary layers. *Bull. Am. Phys. Soc.*, **32**, 2079.

Jackson, E., She, Z.-S., and Orszag, S. A. (1991). A case study in parallel computing: I. Homogeneous turbulence on a hypercube, *J. Sci. Comp.*, **6**(1), 27-45.

Joslin, R. D., Streett, C. L., and Chang, C.-L. (1992). Validation of three-dimensional incompressible spatial direct numerical simulation code–a comparison with linear stability

and parabolic stability equations theories for boundary-layer transition on a flat plate. *NASA TP-3205*, 1992.

Joslin, R. D., Streett, C. L., and Chang, C.-L. (1993a). Spatial DNS of boundary-layer transition mechanisms: Validation of PSE theory. (accepted for publication in *Theor. and Comp. Fluid Dyn.*).

Joslin, R. D. and Streett, C. L. (1993b). The role of stationary crossflow vortices in boundary-layer transition on swept wings. (submitted for publication in *Phys. Fluids A*).

Kleiser, L., and Zang, T. A. (1991). Numerical simulation of transition in wall-bounded shear flows. *Ann. Rev. Fluid Mech.*, **23**, 495-537.

Laurien, E., and Kleiser, L. (1989). Numerical simulation of boundary-layer transition and transition control. *J. Fluid Mech.*, **199**, 403-440.

Lele, S. K. (1992). Compact finite difference schemes with spectral-like resolution. *J. Comput. Phys.*, **103**, 16-42.

Lynch, R. E., Rice, J. R., and Thomas, D. H. (1964). Direct solution of partial difference equations by tensor product methods. *Num. Math.*, **6**, 185-199.

Malik, M. R. and Li, F., Three-dimensional boundary layer stability and transition. *Aerotech '92*, Paper No. 921991, 1992.

Ortega, J. M. and Voigt, R. G. (1988). A bibliography on parallel and vector numerical algorithms. *ICASE Interim Report 6.*

Otto, J. C. (1993). Parallel execution of a three-dimensional, chemically reacting, Navier-Stokes code on distributed-memory machines. *AIAA Paper 93-3307-CP.*

Piomelli, U., Zang, T. A., Speziale, C. G., and Hussaini, M. Y. (1990), On the large-eddy simulation of transitional wall-bounded flows. *Phys. Fluids A*, **2**(2), 257-265.

Rai, M. M., and Moin, P. (1991a). Direct numerical simulation of transition and turbulence in a spatially-evolving boundary layer. *AIAA Paper No. 91-1607.*

Rai, M. M., and Moin, P. (1991b). Direct numerical simulation of turbulent flow using finite-difference schemes. *J. Comput. Phys.*, **96**, 15-53.

Reed, H. L. (1993). Progress in transition modelling: Spatial direct numerical simulations. *AGARD-R-793.*

Smith, A. M. O., and Gamberoni, N. (1956). Transition, pressure gradients, and stability theory. *Douglas Aircraft Company Report No. ES-26388.*

Spalart, P. R. (1989). Direct numerical study of leading-edge contamination. In *Fluid Dynamics of Three-Dimensional Turbulent Shear Flows and Transition*, AGARD-CP-438, 5.1-5.13.

Streett, C. L., and Macaraeg, M. G. (1989). Spectral multi-domain for large-scale fluid dynamic simulations. *Int. J. Appl. Numer. Math.*, **6**, 123-140.

Streett, C. L., and Hussaini, M. Y. (1991). A numerical simulation of the appearance of chaos in finite-length Taylor-Couette flow. *Appl. Numer. Math.*, **7**, 41-71.

Van Ingen, J. L. (1956). A suggested semi-empirical method for the calculation of the boundary-layer transition region. *University of Delft Report VTH-74*, Department of Aerospace Engineering, Delft, The Netherlands.

Williamson, J. H. (1980). Low-storage Runge-Kutta schemes. *J. Comput. Phys.*, **35**(1), 48-56.

Zang, T. A., and Hussaini, M. Y. (1987). Numerical simulation of nonlinear interactions in channel and boundary-layer transition. *Nonlinear Wave Interactions in Fluids*, **87**, 131-145.

Zang, T. A., and Hussaini, M. Y. (1990). Multiple paths to subharmonic laminar breakdown in a boundary layer. *Phys. Rev. Lett.*, **64**, 641-644.

Table 1. Operation Counts for the Major Kernels

Kernel	Operation count (oc)	Normalization count $= \text{oc}/n_x n_y n_z$
MAT-MAT	$O(n_x n_y^3 n_z)$	$O(n_y^2)$
FFT	$O(n_x n_y n_z log_2 n_z)$	$O(\log_2 n_z)$
TRIDIAG	$O(n_x n_y n_z)$	$O(1)$
PENTADIAG	$O(n_x n_y n_z)$	$O(1)$

Table 2. Major Kernel Executions With Different Data Mappings

Kernel	x-mapping	y-mapping	z-mapping
MAT-MAT	global	global	local
FFT	local	local	global
TRIDIAG	global	local	local
PENTADIAG	global	local	local

Table 3. Illustration of the *xor* Scheme for Complete Exchanges

node steps	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

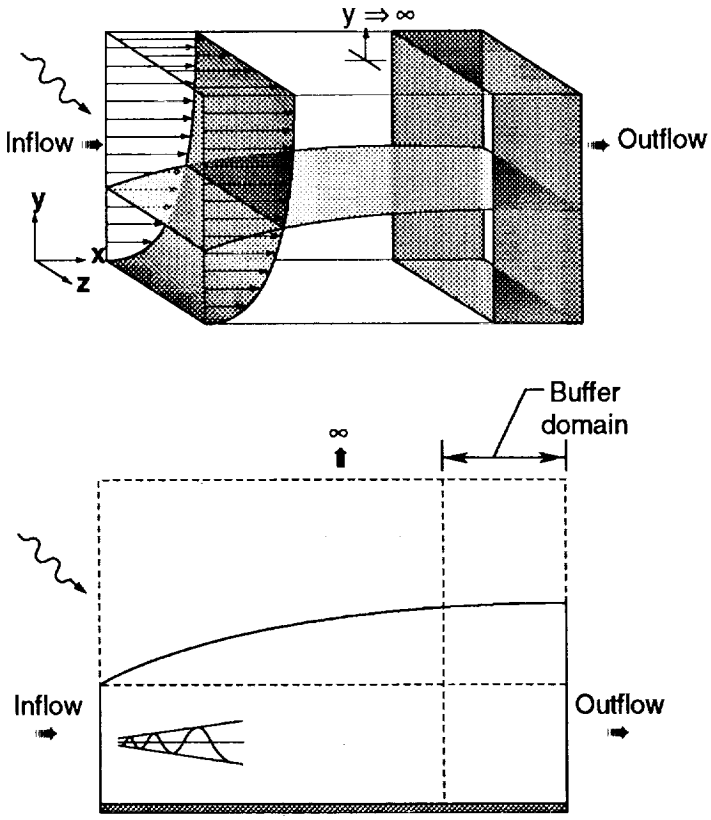

Fig. 1. Computational domain of boundary-layer transition problem.

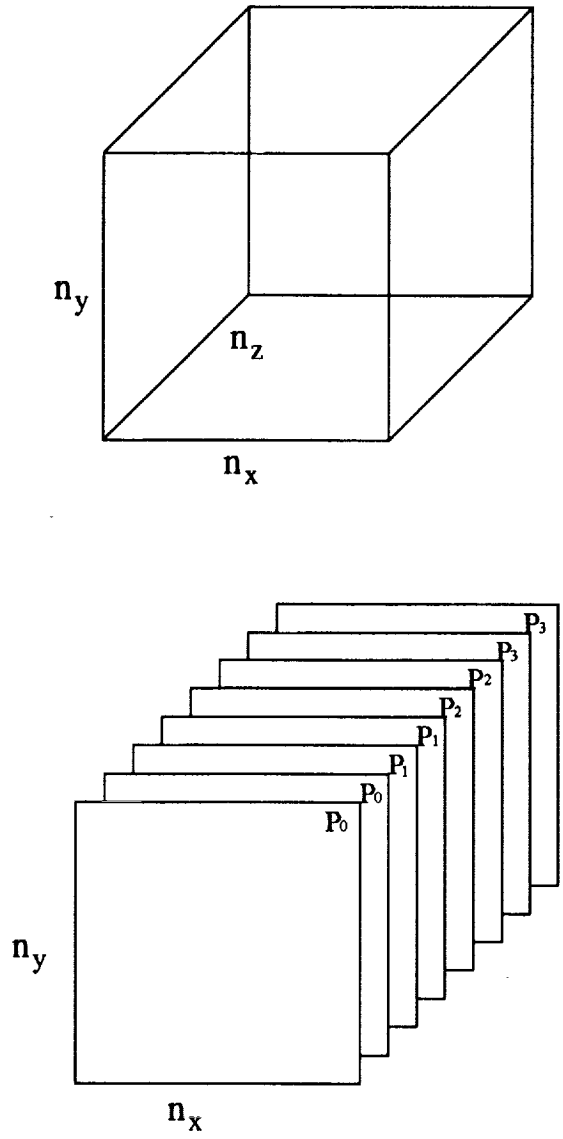

Fig. 2. The z-mapping of $n_z = 8$ spanwise grid onto four-processor machine.

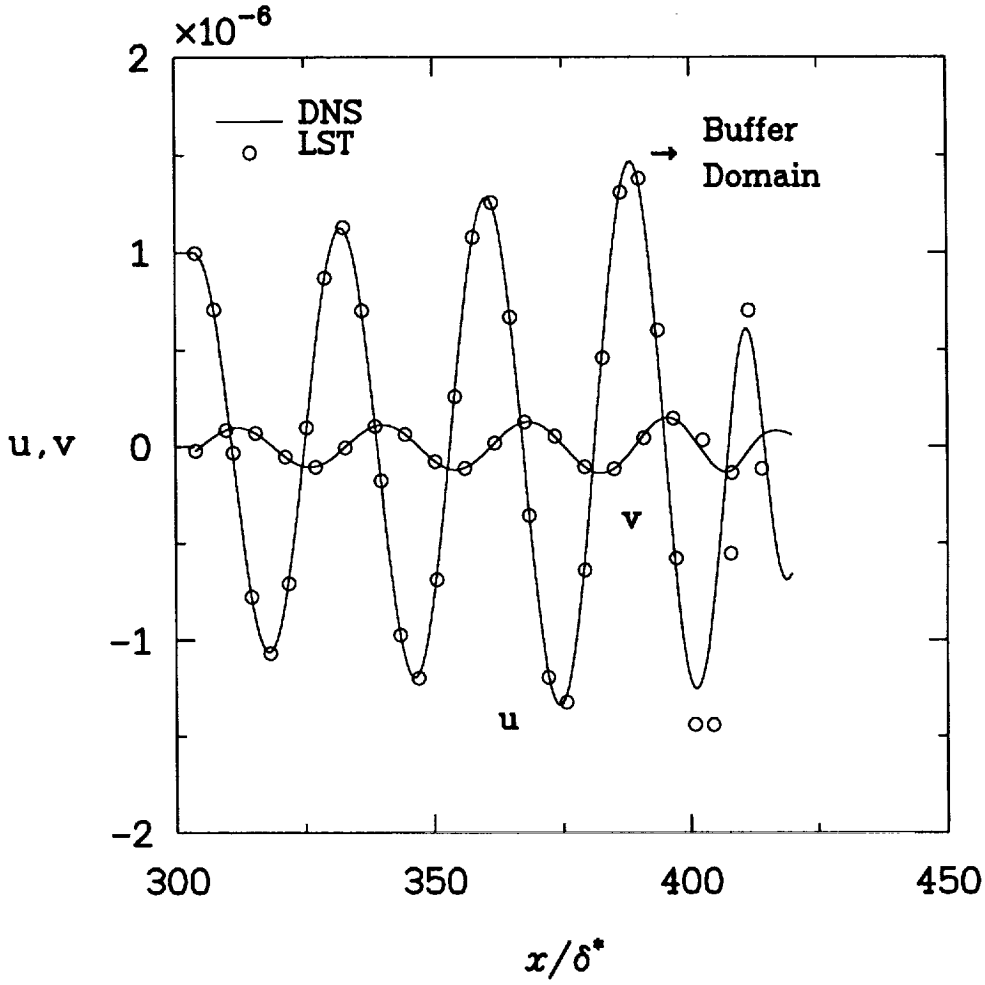

Fig. 3. Amplitude growth with downstream distance for a Tollmien-Schlichting wave with initial amplitude $A^o = 1 \times 10^{-6}$, Reynolds number $R_{\delta_o^*} = 900$, and frequency $\omega = 0.0774$.

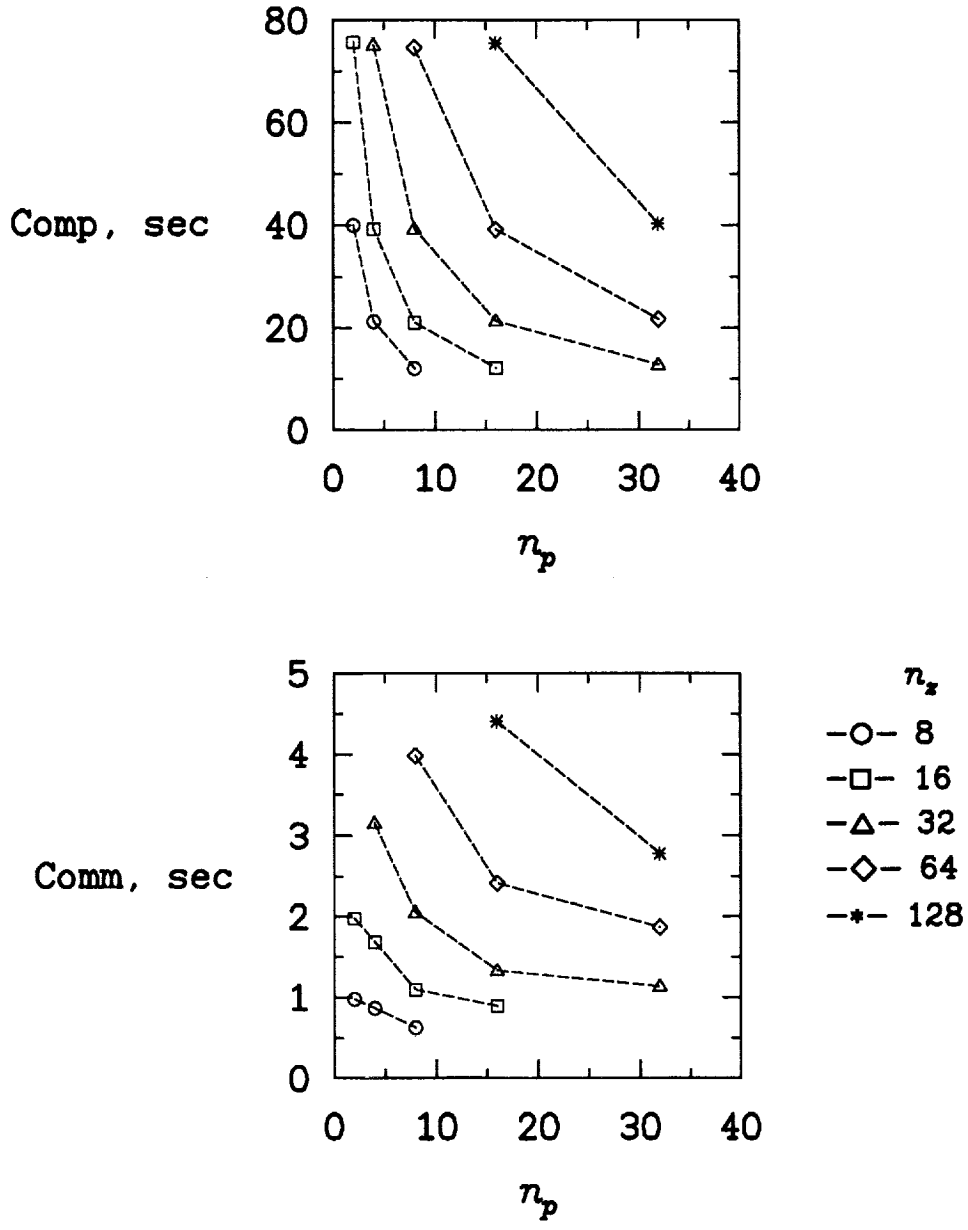

Fig. 4. Computational and communication cost with number of processors (n_p) for initial implementation of PSDNS, where $n_x = 64$ and $n_y = 41$.

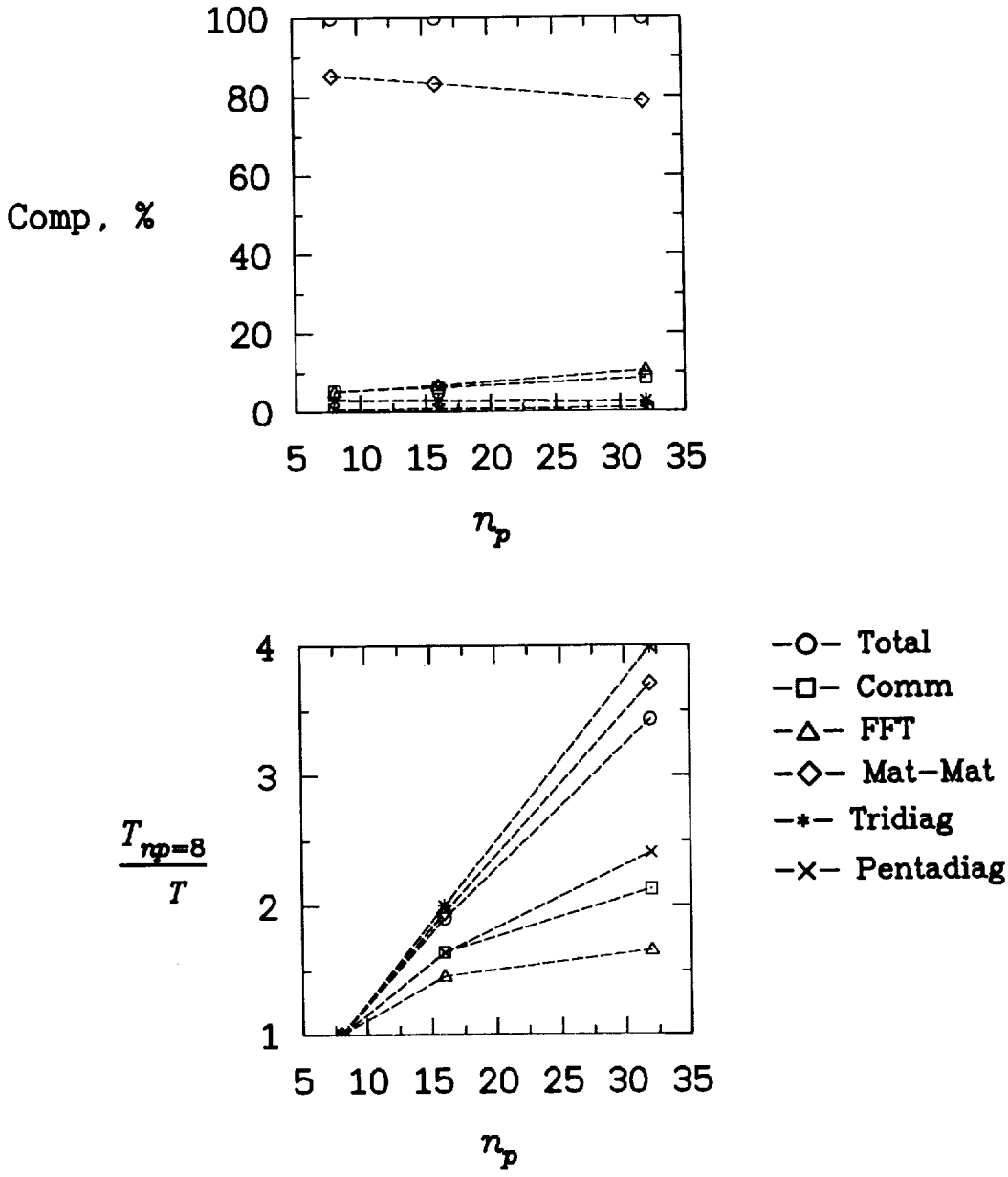

Fig. 5. Computational-cost breakdown and speedup with number of processors (n_p) for initial implementation of PSDNS, where $n_x = 64$, $n_y = 41$, and $n_z = 64$.

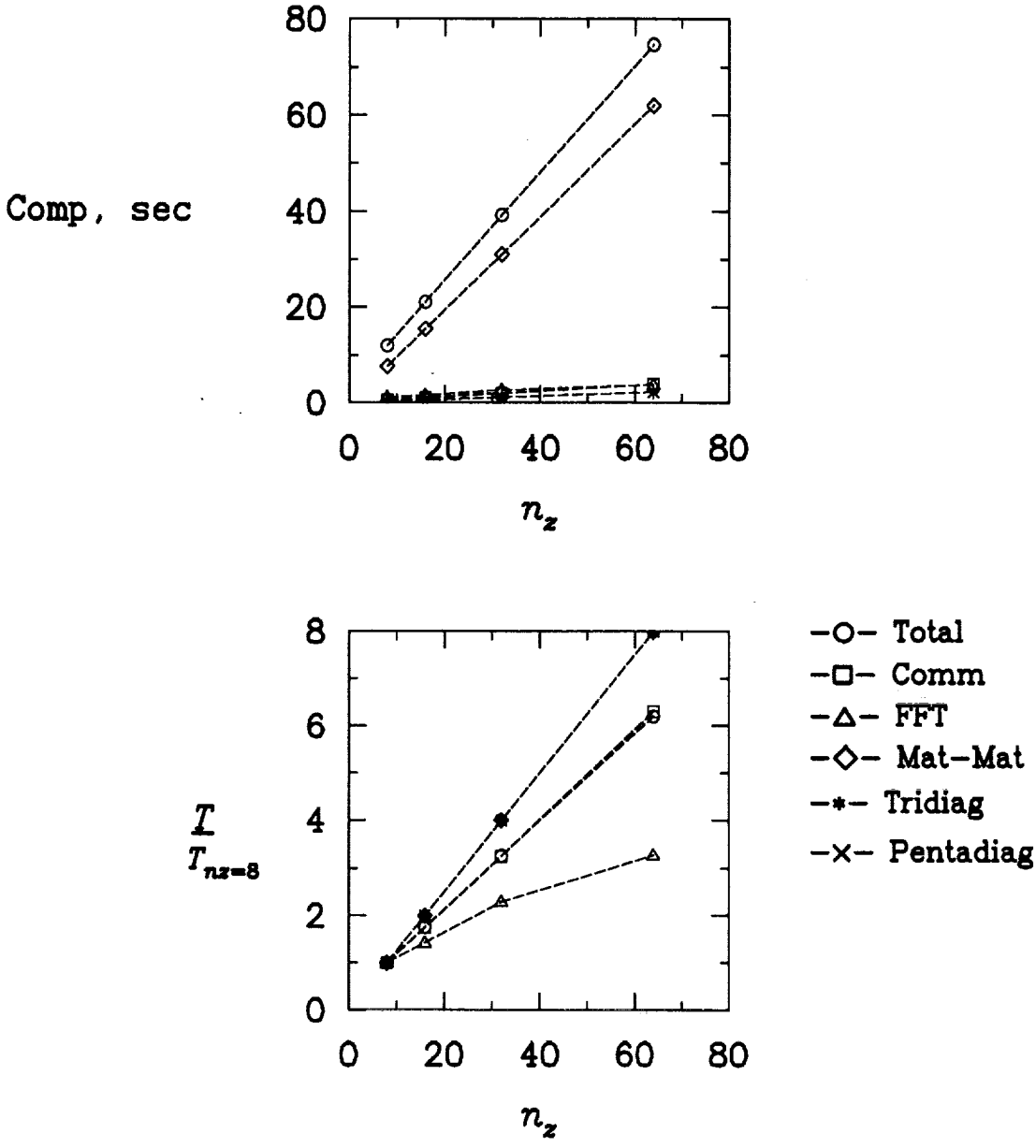

Fig. 6. Computational-cost breakdown and slowdown with spanwise grid (n_z) for initial implementation of PSDNS, where $n_x = 64$, $n_y = 41$, and $n_p = 8$.

Fig. 7. Computational-cost breakdown for each processor for initial implementation of PSDNS, where $n_x = 64$, $n_y = 41$, and $n_z = 8$.

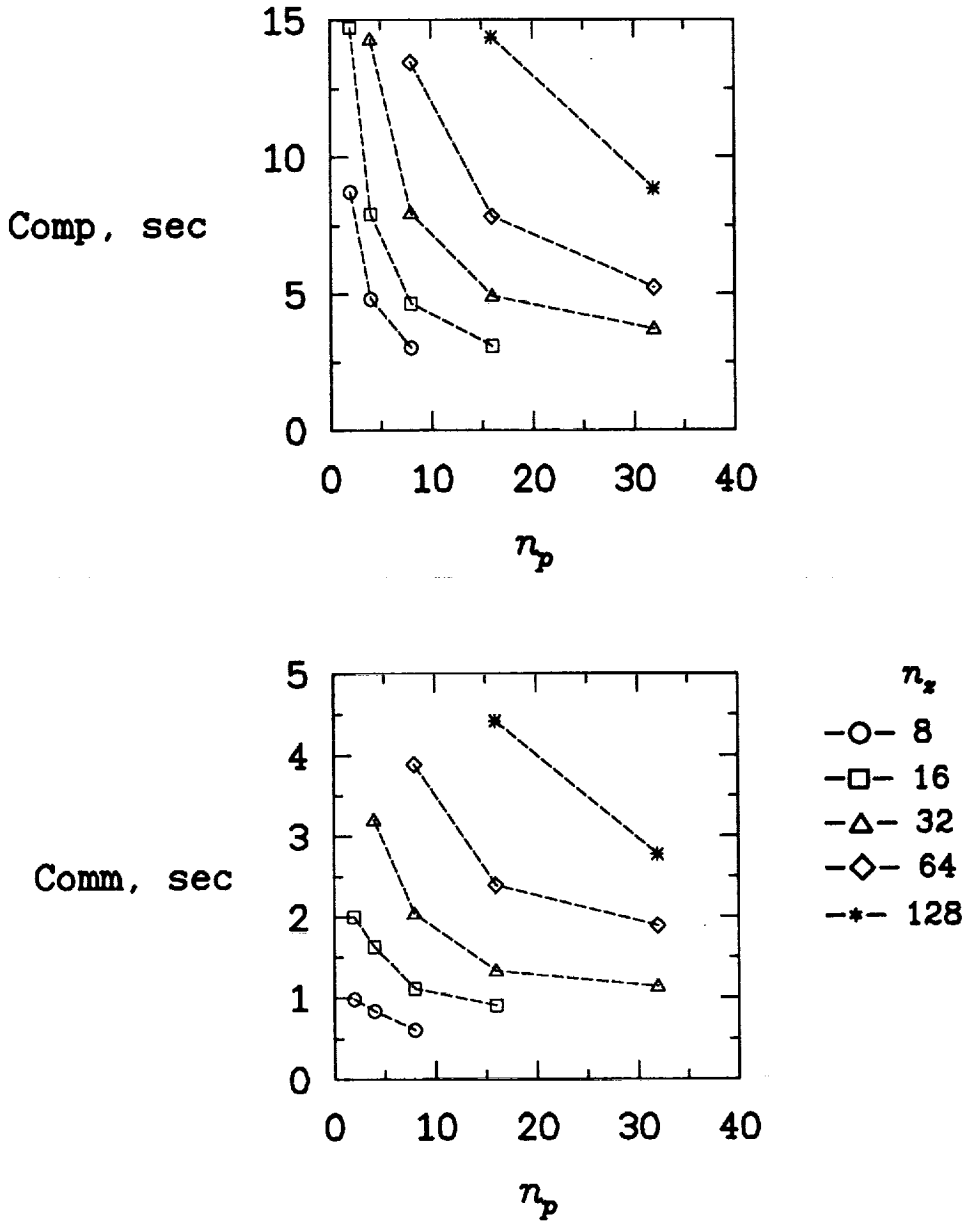

Fig. 8. Computational and communication cost with number of processors (n_p) for optimized PSDNS, where $n_x = 64$ and $n_y = 41$.

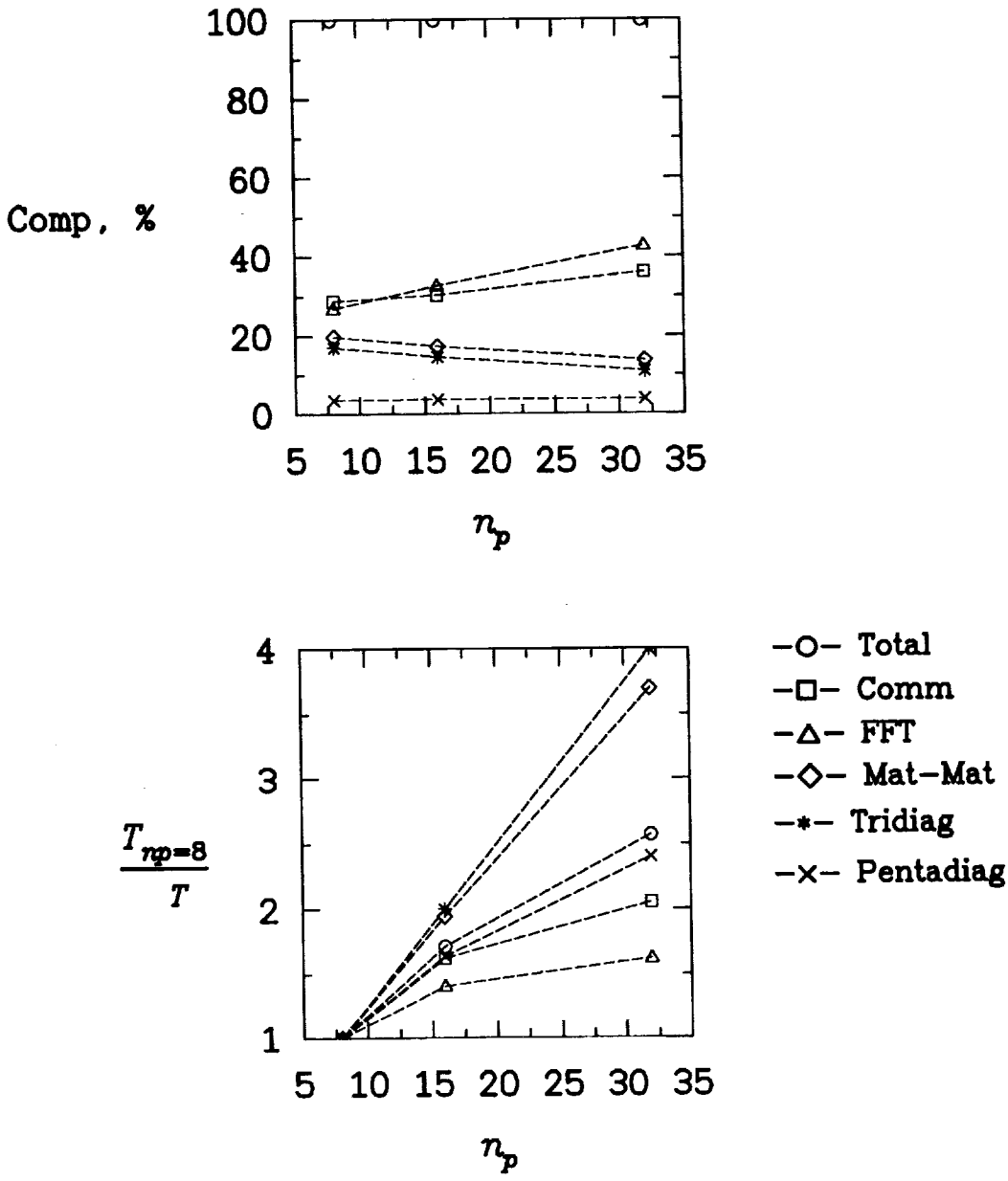

Fig. 9. Computational-cost breakdown and speedup with number of processors (n_p) for optimized PSDNS, where $n_x = 64$ and $n_y = 41$, and $n_z = 64$.

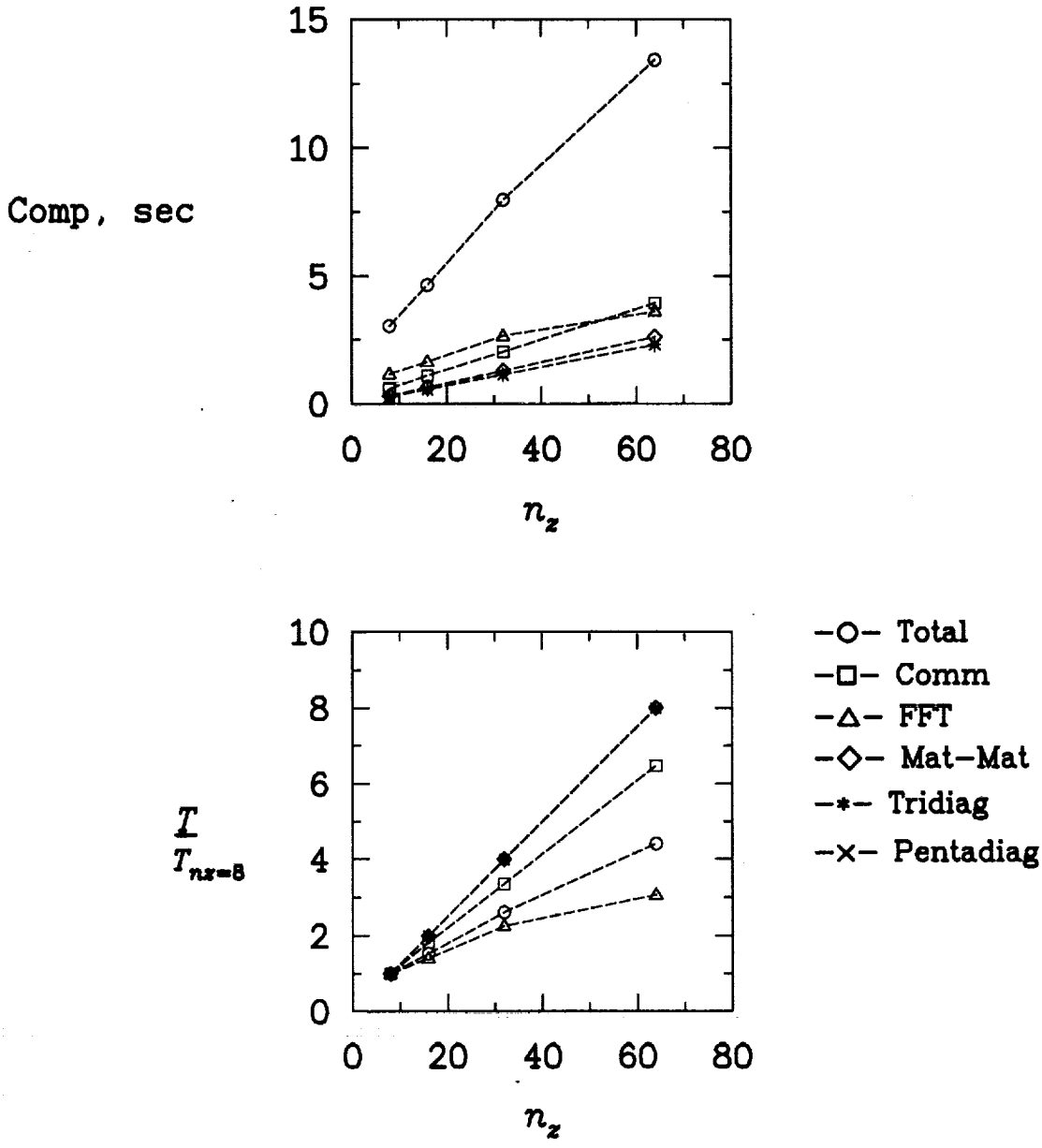

Fig. 10. Computational-cost breakdown and slowdown with spanwise grid (n_z) for optimized PSDNS, where $n_x = 64$, $n_y = 41$, and $n_p = 8$.

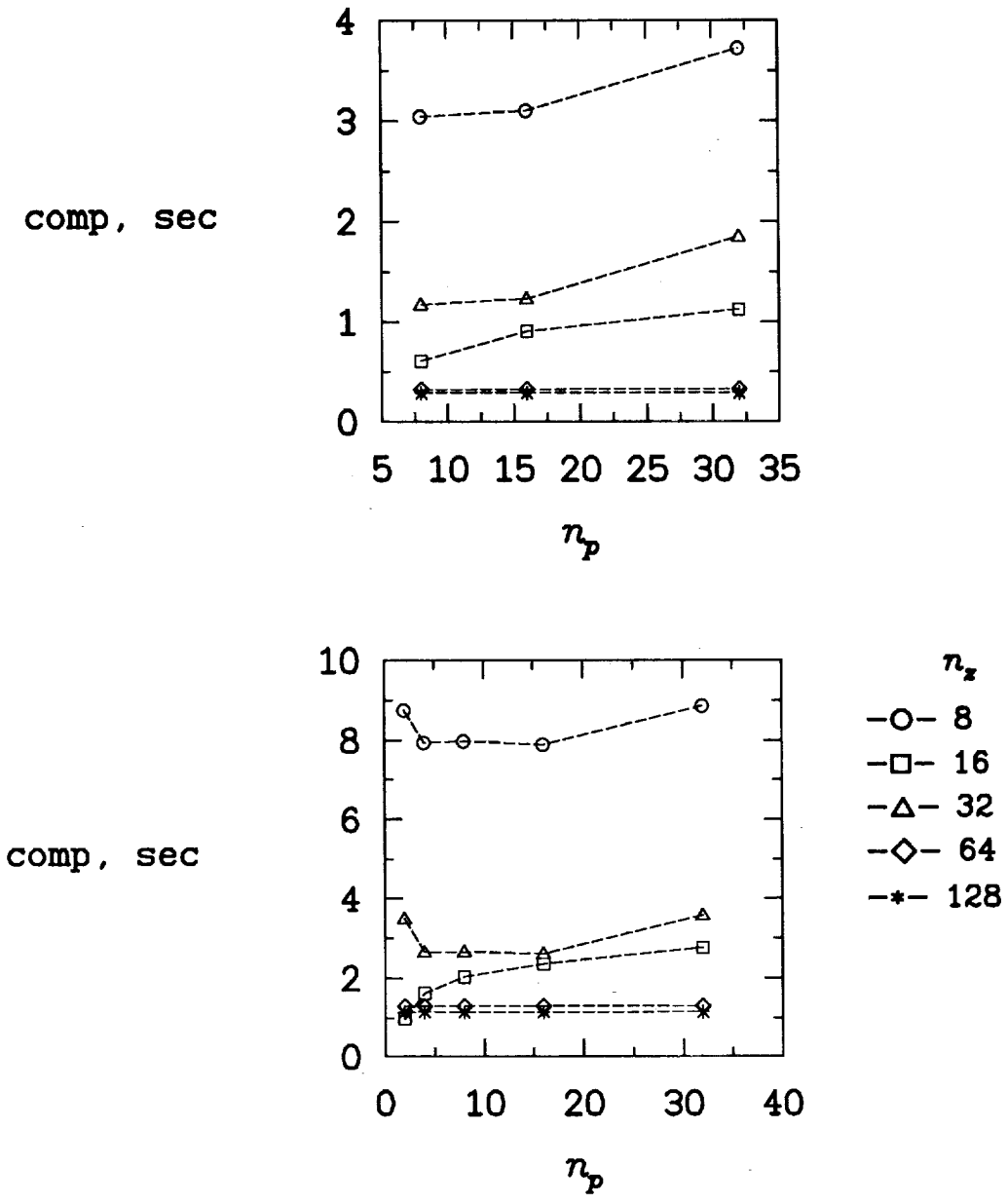

Fig. 11. Computational-cost breakdown for one (top) and four (bottom) $x - y$ planes per processor and number of processors (n_p) for optimized PSDNS, where $n_x = 64$ and $n_y = 41$.

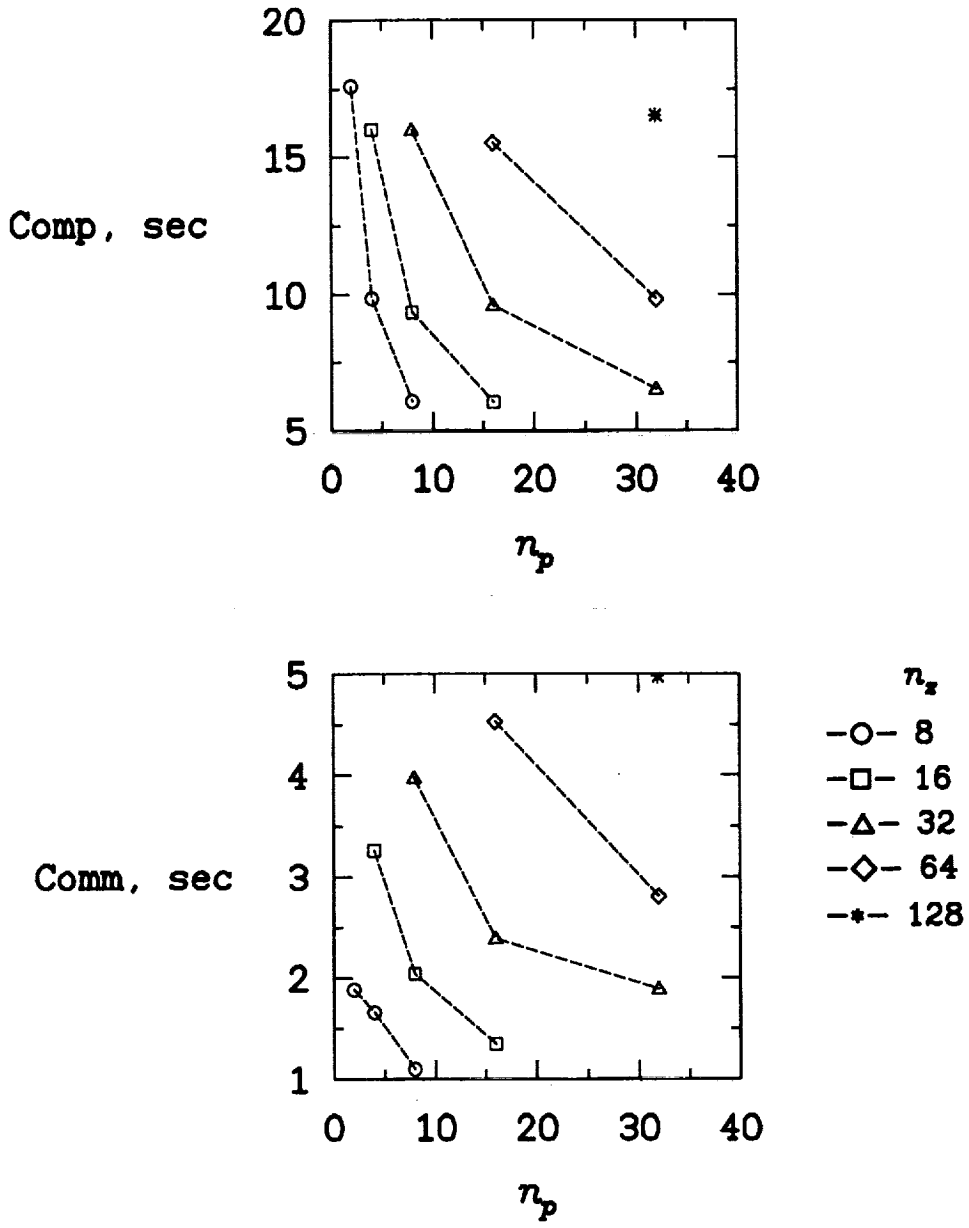

Fig. 12. Computational and communication cost with number of processors (n_p) for optimized PSDNS, where $n_x = 128$ and $n_y = 41$.

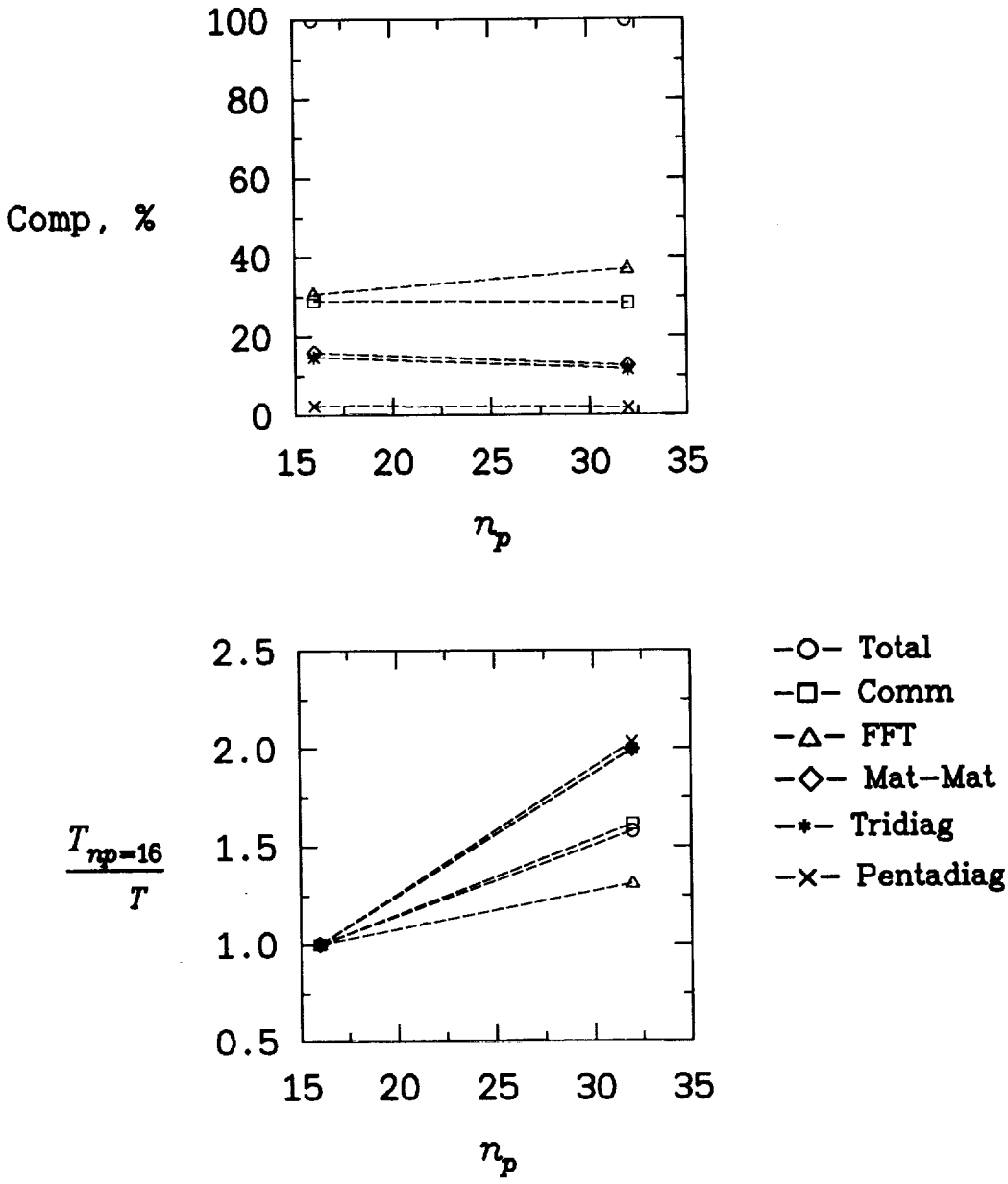

Fig. 13. Computational-cost breakdown and speedup with number of processors (n_p) for optimized PSDNS, where $n_x = 128$, $n_y = 41$, and $n_z = 64$.

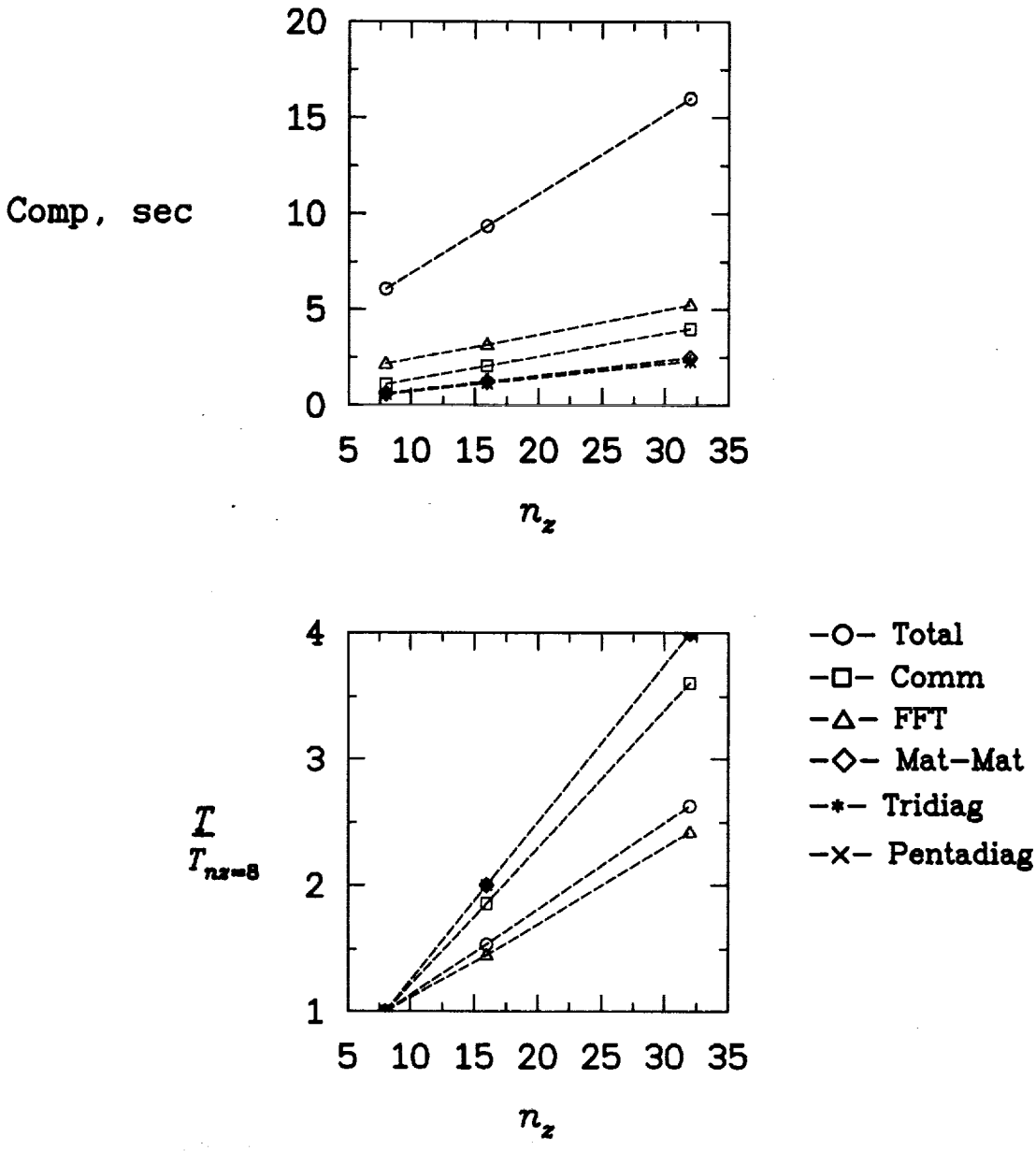

Fig. 14. Computational-cost breakdown and speedup with spanwise grid (n_z) for optimized PSDNS, where $n_x = 128$, $n_y = 41$, and $n_p = 8$.

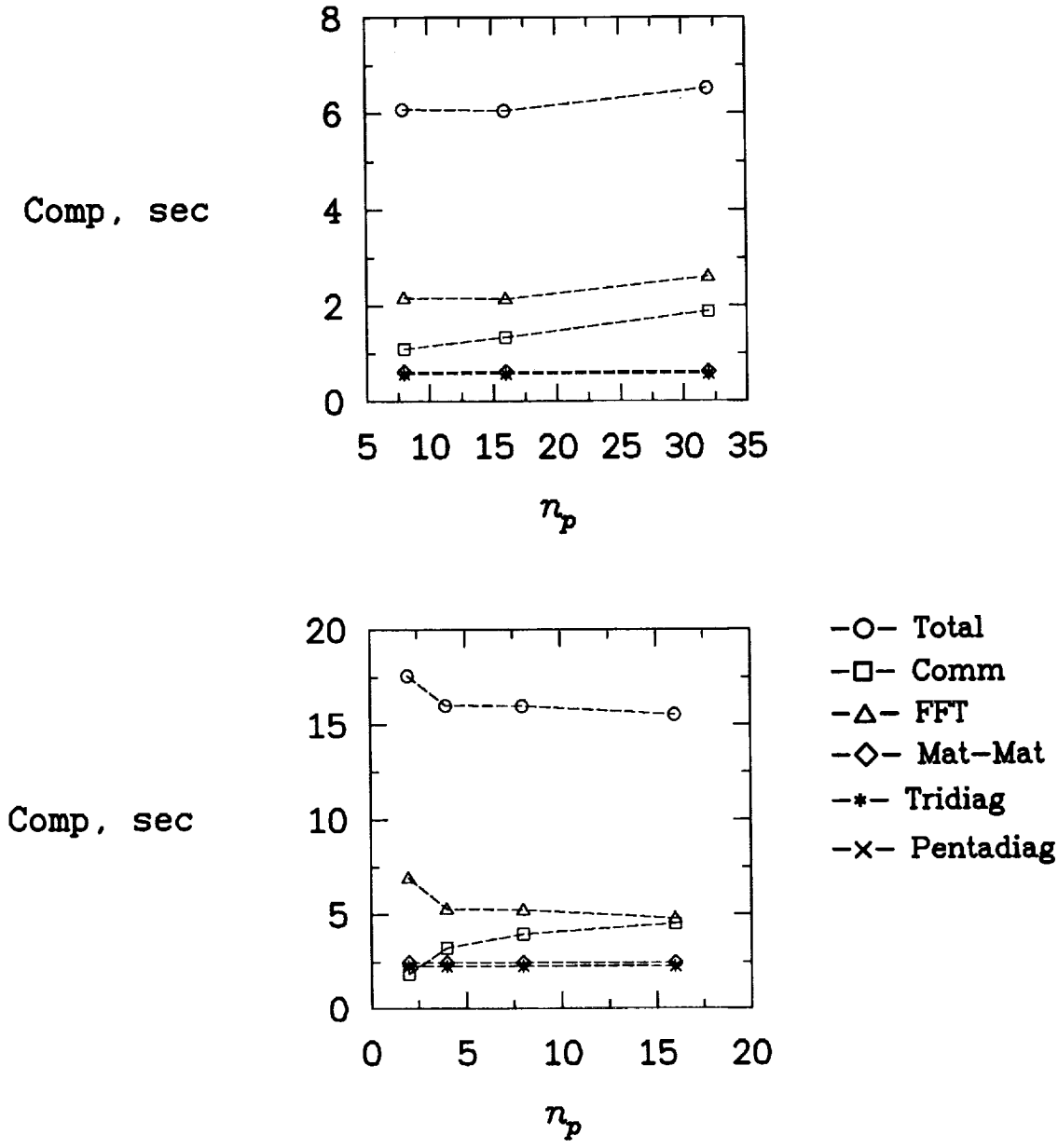

Fig. 15. Computational-cost breakdown for one (top) and four (bottom) $x-y$ plains per processor and number of processors for optimized PSDNS, where $n_x = 128$ and $n_y = 41$.

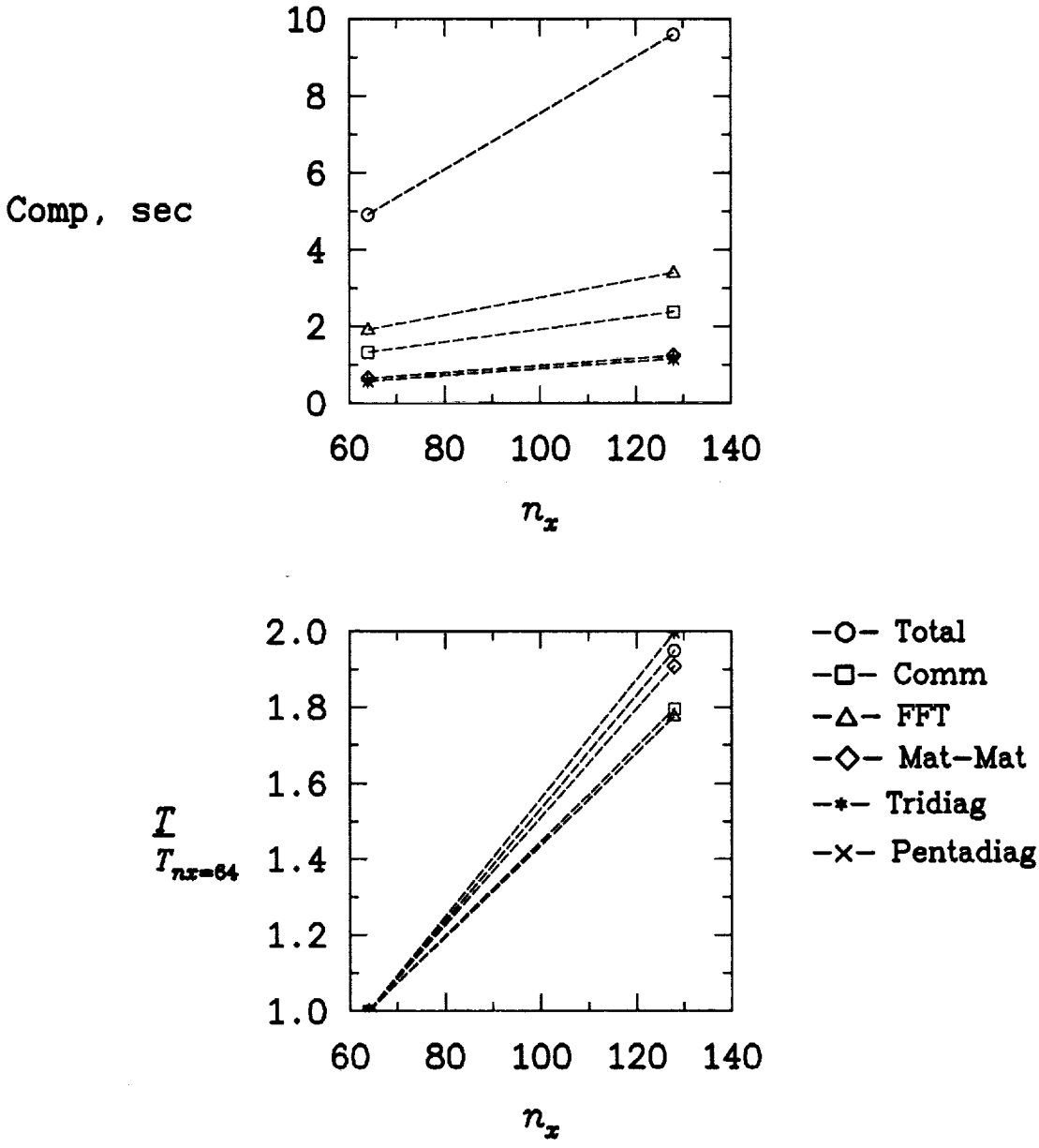

Fig. 16. Computational-cost breakdown and slowdown with streamwise grid refinement for optimized PSDNS, where $n_y = 41$, $n_z = 32$, and $n_p = 16$.

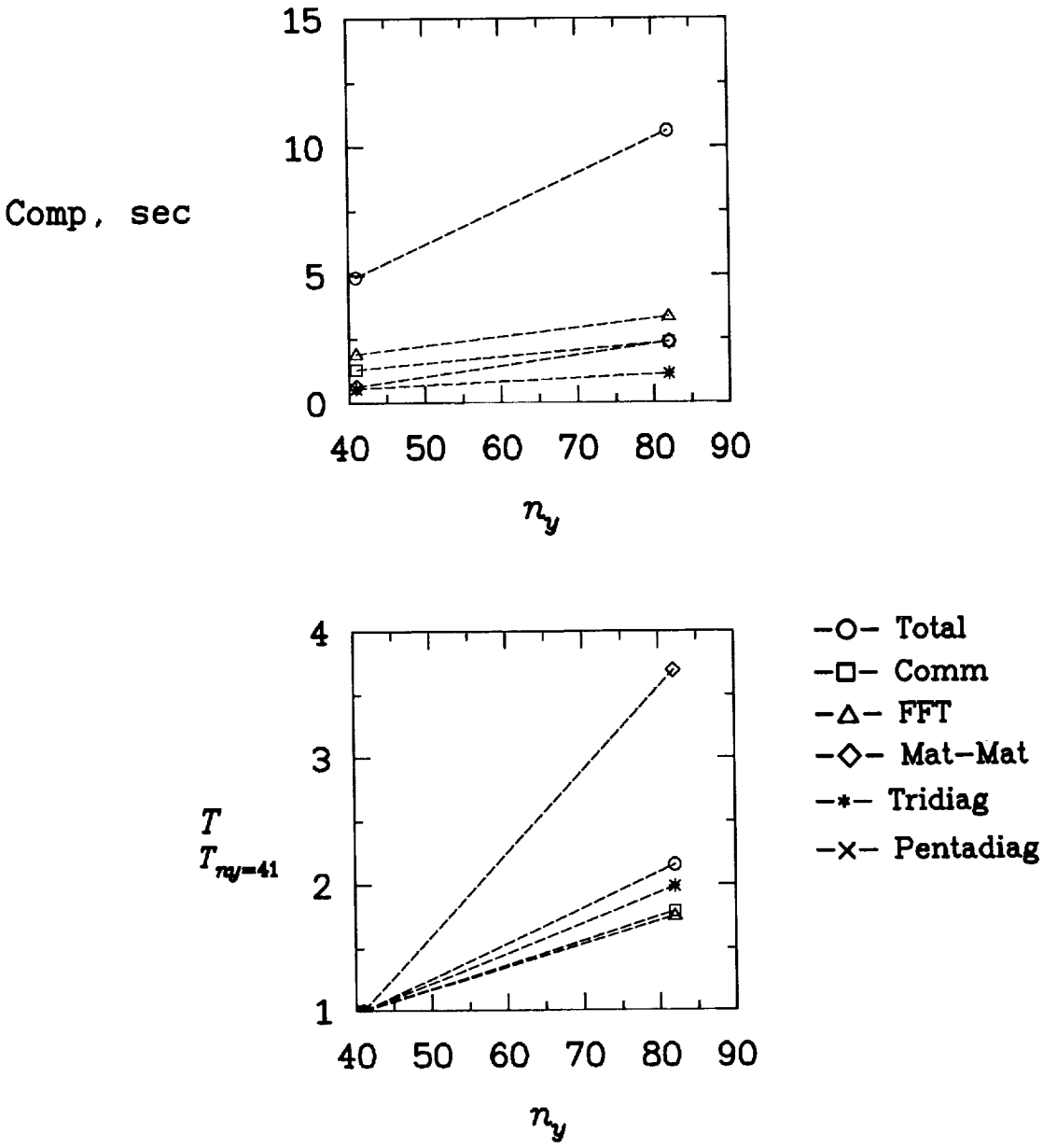

Fig. 17. Computational-cost breakdown and slowdown with streamwise grid refinement for optimized PSDNS, where $n_x = 64$, $n_z = 32$, and $n_p = 16$.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1993	Contractor Report

4. TITLE AND SUBTITLE
PARALLEL SPATIAL DIRECT NUMERICAL SIMULATIONS ON THE INTEL IPSC/860 HYPERCUBE

5. FUNDING NUMBERS
C NAS1-19480
WU 505-90-52-01

6. AUTHOR(S)
Ronald D. Joslin
Mohammad Zubair

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Computer Applications in Science and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER
ICASE Report No. 93-53

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA CR-191513
ICASE Report No. 93-53

11. SUPPLEMENTARY NOTES
Langley Technical Monitor: Michael F. Card
Final Report
Submitted to Journal of Scientific Computing

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited

Subject Category 61

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

The implementation and performance of a parallel spatial direct numerical simulation (PSDNS) approach on the Intel iPSC/860 hypercube is documented. The direct numerical simulation approach is used to compute spatially evolving disturbances associated with the laminar-to-turbulent transition in boundary-layer flows. The feasibility of using the PSDNS on the hypercube to perform transition studies is examined. The results indicate that the direct numerical simulation approach can effectively be parallelized on a distributed-memory parallel machine. By increasing the number of processors, nearly ideal linear speedups are achieved with nonoptimized routines; slower than linear speedups are achieved with optimized (machine-dependent library) routines. This slower than linear speedup results because the FFT routine dominates the computational cost and because the routine indicates less than ideal speedups. However, with the machine-dependent routines, the total computational cost decreases by a factor of 4 to 5 compared with standard Fortran routines. The computational cost increases linearly with spanwise, wall-normal, and streamwise grid refinements. The hypercube with 32 processors was estimated to require approximately twice the amount of Cray supercomputer single processor time to complete a comparable simulation; however, it is estimated that a subgrid-scale model, which reduces the required number of grid points and becomes a large-eddy simulation (PSLES), would reduce the computational cost and memory requirements by a factor of 10 over the PSDNS. This PSLES implementation would enable transition simulations on the hypercube at a reasonable computational cost.

14. SUBJECT TERMS	15. NUMBER OF PAGES
spatial direct numerical simulations; parallel computing	45
	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		

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

☆U.S. GOVERNMENT PRINTING OFFICE: 1993 - 728-064/86058

National Aeronautics and
Space Administration
Code JTT
Washington, D.C.
20546-0001

Official Business

Penalty for Private Use, $300

BULK RATE
POSTAGE & FEES PAID
NASA
Permit No. G-27

POSTMASTER: If Undeliverable (Section 158 Postal Manual) Do Not Return