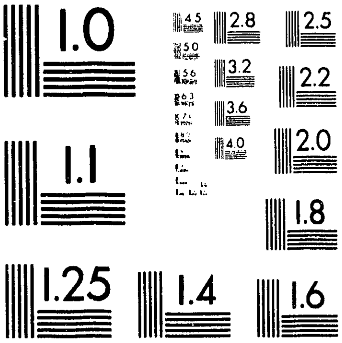

1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

MULTI-CPU PLASMA FLUID TURBULENCE CALCULATIONS ON A CRAY Y-MP C90*

V. E. LYNCH - Presenting Author
Oak Ridge National Laboratory
P. O. Box 2009
Oak Ridge, Tennessee 37831-8058, U. S. A.
Electronic Mail Address: lynch@fedc04.fed.ornl.gov
Phone: (615) 574-0623
FAX: (615) 576-7926

B. A. CARRERAS
Oak Ridge National Laboratory
P. O. Box 2009
Oak Ridge, Tennessee 37831-8070, U. S. A.
Electronic Mail Address: tatedj@fedc04.fed.ornl.gov
Phone: (615) 574-1292
FAX: (615) 576-7926

J. N. LEBOEUF
Oak Ridge National Laboratory
P. O. Box 2009
Oak Ridge, Tennessee 37831-8071, U. S. A.
Electronic Mail Address: leboeuf@fedc04.fed.ornl.gov
Phone: (615) 574-1127
FAX: (615) 576-7926

B. C. CURTIS
National Energy Research Supercomputer Center
P. O. Box 5509, L-560
Livermore, California 94551, U. S. A.
Electronic Mail Address: curtis@nersc.gov
Phone: (510) 423-1330
FAX: (510) 422-0435

R. L. TROUTMAN
National Energy Research Supercomputer Center
P. O. Box 5509, L-560
Livermore, California 94551, U. S. A.
Electronic Mail Address: roy@nersc.gov
Phone: (510) 423-4283
FAX: (510) 422-0435

TECHNICAL PAPER

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

* Research sponsored by the Office of Fusion Energy, U. S. Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

Multi-CPU Plasma Fluid Turbulence Calculations on a CRAY Y-MP C90

V. E. Lynch, B. A. Carreras, and J. N. Leboeuf
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831

B. C. Curtis and R. L. Troutman
National Energy Research Supercomputer Center
Livermore, California 94551

Abstract

Significant improvements in real-time efficiency have been obtained for plasma fluid turbulence calculations by microtasking the nonlinear fluid code KITE in which they are implemented on the CRAY Y-MP C90 at the National Energy Research Supercomputer Center (NERSC). The number of processors accessed concurrently scales linearly with problem size. Close to six concurrent processors have so far been obtained with a three-dimensional nonlinear production calculation at the currently allowed memory size of 80 Mword. With a calculation size corresponding to the maximum allowed memory of 200 Mword in the next system configuration, we expect to be able to access close to nine processors of the C90 concurrently with a commensurate improvement in real-time efficiency. These improvements in performance are comparable to those expected from a massively parallel implementation of the same calculations on the Intel Paragon.

1 Introduction

To predict the size, expense and performance of magnetic fusion reactors, it is critical to know the scaling of particle transport and losses across the magnetic field with machine and plasma parameters. The mechanism for transport across magnetically confined plasmas is not yet known. However, at the plasma edge of the TEXT tokamak, a type of toroidal magnetic confinement device, losses larger than those predicted by classical theory have been shown to be induced by fluctuations.[1] Numerical calculations of plasma edge turbulence are expected to lead to the identification of the transport scaling parameters.[2]

At the plasma edge, the dynamics of the plasma can be modeled by fluidlike equations. A possible model for the turbulence at the edge of tokamaks like TEXT is collisional drift waves destabilized by atomic physics sources such as line radiation cooling caused by light impurities.[3,4] These impurity radiation-driven drift waves are adequately described by five nonlinear fluid equations for plasma density, vorticity, potential, parallel velocity, and temperature, with impurity radiation sources introduced in the latter. These equations are solved as an initial value problem in the nonlinear fluid computer code KITE,[5] which treats the linear terms implicitly and the nonlinear terms explicitly. The implicit treatment of the linear terms requires the inversion of block-tridiagonal matrices. The calculation of the nonlinear terms is usually done by convolutions.

High resolution is needed to resolve all scale lengths in a turbulence problem,[2] requiring the use of large amounts of memory. To study the development of steady-state turbulence, the plasma evolution must be followed for many time steps. Therefore, these calculations also require large amounts of computing time. With present computers, the parameter regime that can be studied is still limited.

To improve the real-time efficiency of plasma turbulence calculations using the KITE family of fluid computer codes, we have explored two main avenues: massively parallel numerical schemes on the Intel iPSC/860 and Touchstone Delta and multiprocessing on the CRAY IIs and the CRAY Y-MP C90. Improvements in performance on the Intels have been reported elsewhere, [6-8] and we concentrate here on the multi-CPU implementation of the KITE code on the CRAY Y-MP C90.

Multiprocessing of the KITE family of codes[9] for the CRAY IIs started upon their arrival at NERSC in the mid 1980s. At that time, use of more than one processor was achieved via software multitasking calls inserted in the Fortran programs. Given the small number of processors (four to eight), the CTSS operating system environment, and the large amount of code modifications required, the experience was less than satisfactory.

The situation has changed substantially with the recent arrival of the CRAY Y-MP C90 at NERSC. Its 16 processors with peak aggregate speed of 16 Gflop, its 268 Mword of shared memory, the UNICOS 7.0 operating system, its efficient CF77 compiling system and associated autotasking and microtasking software make the C90 very attractive for multiprocessing. Autotasking has the advantage of simplicity because it only requires one option of the CF77 compiler to be activated for it to analyze the code and insert compiler directives. It does, however, often parallelize loops that do not account for much of the running time and leaves the important loops as serial code. Fortunately, microtasking is an alternative to autotasking that can produce better performance. It involves analyzing the code by hand or using parallelizing tools and inserting directives where parallelism is found.

Here, we present the results of the multi-CPU implementation of our plasma edge turbulence model on the CRAY Y-MP C90. These results are compared with the corresponding serial implementation on the C90 and with the massively parallel implementation on the Intel parallel computers. In Section 2, the equations used in the present studies are described, and the algorithm is briefly summarized. The multiprocessor implementation of this scheme on the C90 is discussed in Section 3. The timing results for serial, autotasked, and microtasked modes of operation and different problem sizes are presented in Section 4. In Section 5, we give our conclusions.

2 Equations and Numerics

In the study of collisional drift wave-type instabilities driven by impurity radiation at the plasma edge, we assume cylindrical geometry and use fluidlike model equations. The system of equations consists of the continuity equation, coupled to the parallel and perpendicular momentum balance or vorticity equations, and the electron temperature equation, including impurity radiation effects[3]:

$$\frac{d\tilde{n}}{dt} = -V_{*_n}\frac{1}{r}\frac{\partial\tilde{\phi}}{\partial\theta} - c_s\nabla_{\parallel}\tilde{V}_{\parallel} - \chi\nabla_{\parallel}^2(\tilde{\phi} - \tilde{n} - \alpha\tilde{T}) + D\nabla_{\perp}^2\tilde{n} . \tag{1}$$

$$\frac{d\tilde{V}_{\parallel}}{dt} = -c_s\nabla_{\parallel}(\tilde{n} + \tilde{T}) + \mu_{\parallel}\nabla_{\parallel}^2\tilde{V}_{\parallel} + \mu_{\perp}\nabla_{\perp}^2\tilde{V}_{\parallel} . \tag{2}$$

$$\frac{d\tilde{U}}{dt} = -\chi\nabla_{\parallel}^2(\tilde{\phi} - \tilde{n} - \alpha\tilde{T}) + \mu\nabla_{\perp}^2\tilde{U} . \tag{3}$$

$$\begin{aligned}\frac{d\tilde{T}}{dt} &= -V_{*_T}\frac{1}{r}\frac{\partial\tilde{\phi}}{\partial\theta} - \frac{2}{3}c_s\nabla_{\parallel}\tilde{V}_{\parallel} - \frac{2}{3}\alpha\chi\nabla_{\parallel}^2(\tilde{\phi} - \tilde{n} - \alpha\tilde{T}) \\ &\quad + \chi_{\parallel}\nabla_{\parallel}^2\tilde{T} + \chi_{\perp}\nabla_{\perp}^2\tilde{T} + \gamma_T\tilde{T} - \gamma_n\tilde{n} .\end{aligned} \tag{4}$$

An auxiliary time-independent equation is also solved to determine the electrostatic potential, ϕ from the vorticity U:

$$\tilde{U} = \rho_s^2\nabla_{\perp}^2\tilde{\phi} . \tag{5}$$

The total time derivative in Eqs. (1–4) is such that

$$\frac{d}{dt} = \frac{\partial}{\partial t} - c_s\rho_s(\vec{\nabla}\tilde{\phi}\, x\, \zeta)\cdot\vec{\nabla} , \tag{6}$$

where the second term in Eq. (6) is the convective nonlinearity.

Here, the electrostatic potential, density n, parallel velocity $V_{\parallel}$, and temperature T, fluctuations have been normalized to

$$\tilde{\phi} = \frac{|e|\phi}{T_{e0}}, \quad \tilde{n} = \frac{n}{n_0}, \quad \tilde{V}_{\parallel} = \frac{V_{\parallel}}{c_s}, \quad \tilde{T} = \frac{T}{T_{e0}}, \tag{7}$$

where T_{e0} is the equilibrium electron temperature, n_0 is the equilibrium density, and $c_s = \sqrt{T_{e0}/M_i}$ is the sound speed. In Eqs. (1) to (6), $V_{*n} = c_s\rho_s/L_n$ and $V_{*T} = c_s\,\rho_s/L_T$ are the diamagnetic drift velocities foi the density and the temperature with gradient scalelengths L_n and L_T, respectively; $\rho_s = c_s/\omega_{ci}$ is the sound Larmor radius; $\chi_{\parallel} = 2\,\kappa_{\parallel}/3n_0$ is the normalized parallel thermal conductivity; $\chi = \chi_{\parallel}/1.07$ is the normalized resistivity; $\chi_{\perp}$ is the normalized perpendicular thermal conductivity; D is the particle diffusivity; μ is the classical fluid viscosity; $\mu_{\parallel}$ and $\mu_{\perp}$ are the parallel and perpendicular ion viscosities; and the parameter α is such that α= 1.71. The symbol $\parallel$ means parallel to the total magnetic field, which is expressed as

$$\vec{B} = B_0(\vec{\zeta} - \frac{r}{R}\frac{1}{q}\vec{\theta}), \tag{8}$$

where the safety factor $q=rB_0/(RB_p)$, with B_0 the toroidal magnetic field in the ζ direction, B_p the poloidal magnetic field in the θ direction, the major radius R, and radial variable r. The symbol $\perp$ indicates the direction perpendicular to the toroidal magnetic field. For periodic perturbations proportional to cos $(m\theta+n\zeta)$ or sin $(m\theta+n\zeta)$, with m the poloidal mode number and n the toroidal mode number, $\nabla_{\parallel}= 0$ at the radial position for which q(r)= m/n. The radial position is called the resonant or singular surface for this perturbation.

Impurity radiation manifests itself in the electron temperature equation, Eq. (4), through the terms $\gamma_T= -2/3n_z\, dI_z/dT_{e0}$ and $\gamma_n = n_zI_z/T_{e0}$ multiplying temperature and density fluctuations, respectively, with n_Z the impurity density and I_Z the low-z impurity cooling rate. Both can act as sources of fluctuations at the plasma edge of magnetic confinement devices such as tokamaks. The impurity radiation strength causes the fluctuations with low-to-moderate m numbers to grow exponentially in time in the linear phase. A turbulent, saturated steady state is obtained through coupling to damped modes with higher m numbers and generation of a mean poloidal sheared flow velocity via the convective nonlinearity. The impurity radiation strength and profile, as well as the equilibrium temperature and density profiles, are held fixed throughout the calculation; we are, therefore, in a driven turbulence situation.

The edge turbulence model represented by Eqs. (1–5) has been implemented in the computer code KITE.[5] To maximize resolution, Eqs. (1–5) are solved within a cylindrical annulus that extends from r_{min} to the minor radius a of the cylinder. The fluctuating fields are written as expansions in sines and cosines:

$$\tilde{f}(r,\theta,\zeta) = \sum_{m,n}\left[f^c_{mn}(r)\cos(m\theta + n\zeta) + f^s_{mn}(r)\sin(m\theta + n\zeta)\right], \tag{9}$$

in the poloidal (θ) and toroidal (ζ) angles.

The boundary conditions are such that the radial components of the velocity is zero at conducting walls placed at $r=r_{min}$ and r=a and so are density, temperature and parallel velocity perturbations. That is, for all fields,

$$f(r_{min},\theta,\zeta)= f(a,\theta,\zeta) . \tag{10}$$

To complete the numerical representation of the fields, we use finite differences in r. First and second radial derivatives are calculated with three-point, finite-difference formulas. Derivatives in θ and ζ are performed analytically. All quantities are stored in spectral form and are never transformed to a finite-difference grid in θ and ζ.

These nonlinear equations are solved as an initial value problem. The numerical scheme presently used in the computer code KITE treats all linear terms in the perturbation implicitly. This requires the inversion of a block-tridiagonal matrix for each Fourier component, with the size of the blocks given by the number of equations (ten here, five each for the sine and cosine components) and the number of blocks set by the number of grid points. The matrix inversion problem is solved with the block-tridiagonal linear system solver of Hindmarsh.[10] The blocks of the matrix and the corresponding rows of the right-hand side vector are stored in memory. The solver uses block Gauss elimination. Partial pivoting is done within block rows only. These routines (SOL and SOLBT) are vectorized over the number of equations only, which leads to short vector lengths, because pivoting makes it difficult to vectorize the matrix inversion over the number of modes, which would lead to longer vector lengths. The nonlinear terms are explicit and accurate to first order in time. They lead to convolutions over poloidal and toroidal modes that are performed analytically rather than using fast Fourier transforms because only modes within a narrow helicity band are of interest. The subroutine (MULT) performing the mode convolutions is fully vectorized with do-loop unfolding.[11]

3 Multi-CPU Implementation

For the KITE family of codes, the matrix operations and mode convolutions are the most time-consuming parts of the calculations. This is so not only in serial implementation on the CRAYs but also in massively parallel implementation on the Intel iPSC/860 and Touchstone Delta.[8] Most of the effort has, therefore, been put into achieving efficient multi-processing of the matrix operations and convolutions on the C90.

Compared to the algorithm development and extensive code modifications necessary to obtain efficient massively parallel code for the Intel parallel computers, few changes are required for the code to multiprocess effectively on the C90. We did restructure the code so that the matrix and right-hand side were stored in memory for all the modes. Previous codes read the matrix from disk one mode at a time each time step and calculated the right-hand side with each mode overwriting the right-hand side for the last one.This I/O bound version was preferred on the CRAY IIs to circumvent the scheduling and charging scheme in effect at NERSC, at the expense of slower turnaround, more than to save memory storage.

On the C90, the autotasked version of the code is produced automatically by the CF77 compiling system when the -zp option is used on the compiler line with no further changes made to the serial version. The microtasked version was produced by first profiling the code to assess which modules account for most of the computational time. The automatic parallelizer tool FORGE was then applied to these modules to detect parallelism, and compiler directives were inserted where parallelism was found. To microtask the matrix inversion, only the following three lines were required:

```
CMIC\$ DOALL
CMIC\$*SHARED (index.nmatx.mjml.ampls.bmpls,cmpls,y,impls)
CMIC\$*PRIVATE (l)
      do 21 l=2,index+1
       call solbt(nmatx,mjml.ampls(1,1,1,l),bmpls(1,1,1,l),
      &  cmpls(1,1,1,l),y(1,1,l),impls(1,1,l))
   21 continue
```

The parallelization is performed over the number of modes l. Each variable in the do-loop over modes must be declared to be shared with all the processors or private to each processor. To microtask the convolutions over modes, the following four lines of compiler directives were inserted in the subroutine MULT:

```
CMIC\$ DOALL
CMIC\$*SHARED (mj,lmax,l0,k1max,l1max,l1h,k1h,l1g,k1g,h,g,f)
CMIC\$*PRIVATE (kp,lp1,lp,j,fs,l)
CMIC\$*SAVELAST

      do 1999 l=1,lmax
      do 1897 j=0,mj
 1897 fs(j)=0.
      do 1898 lp=l1max(l-1)+1,l1max(l),4
      if(l1max(l)-lp.lt.4) go to 100
      do 98 j=0,mj
```

```
     fs(j)=((((fs(j)+g(j,llg(lp)*h(j,l1h(lp)))
    1            +g(j,llg(lp+1)*h(j,l1h(lp+1)))
    1            +g(j,llg(lp+2)*h(j,l1h(lp+2)))
    1            +g(j,llg(lp+3)*h(j,l1h(lp+3)))
  98 continue
     go to 1898
 100 do 150 lp1=lp,llmax(l)
     do 150 j=0,mj
 150 fs(j)=fs(j)+g(j,l1g(lp1))*h(j,l1h(lp1))
1898 continue
      do 1998 kp=k1max(l-1)+1,k1max(l)
      do 1998 j=0,mj
      fs(j)=fs(j)-g(j,k1g(kp)*h(j,k1h(kp))
1998 continue
      do 1999 j=0,mj
1999 f(j,l)=0.5*fs(j)
```

The convolution routine is highly optimized. Each inner loop is vectorized over the number of radial grid points and has do-loop unfolding with four terms written out explicitly. With unfolding the array fs needs to be accessed and stored only one-fourth as frequently. The outer loop is performed in parallel over the number of modes. One other subroutine was microtasked, with parallelization over the number of modes. It is a simple tridiagonal solver over radial grid points (TRDG) that is called for all modes when the potential is calculated from the vorticity [Eq. (5)].

With these modifications, the KITE profile displayed as a pie chart in Fig. 1 is obtained for our plasma edge turbulence model run for 100 time steps with 385 grid points and 539 modes. The matrix operations (SOL/SOLBT) account for 47% of total CPU time, the convolutions (MULT) for 41%, and the tridiagonal solver (TRDG) for 3%, for a total of 91% . It is clear from Fig.1 that all parts of the calculations that matter are the ones that have been microtasked save for 9% of the computational burden, which is made up of the startup serial routines. This startup overhead would be much lower if a calculation over thousands of steps had been profiled.

4 Results

The arbitrary memory limit currently in effect on the C90 at NERSC is set at 80 Mword. For the KITE implementation of the plasma edge turbulence model given by Eqs. (1–5), the memory size, Msize, is dominated by the matrices and by varying the number of modes, Nmodes, and grid points, MJ, and we have obtained the following scaling:

$$\text{Msize (Mword)} = 0.66 + 3.76 \times 10^{-4} \text{ (MJ x Nmodes)}. \qquad (11)$$

The present studies have, therefore, been limited to grid sizes and numbers of modes that result in a memory size ≤ 80 Mword, with the largest size calculations attempted having MJ= 385 and Nmodes= 539 for Msize= 79 Mword. We point out that the calculations reported here were not carried out in single-user or dedicated mode but in the usual, competitive, multiuser batch environment that production calculations must be performed in on the C90 at NERSC. These are, therefore, not so much a reflection of the best the machine can do but more an assessment of how well calculations can be performed under the day-to-day constraints imposed by the system (swapping, scheduling) and varying machine workload (time of day, hundreds of competing tasks). They do, therefore, give a true measure of real-time efficiency.

Comparisons of the performance of serial, autotasked, and microtasked implementations of the KITE code on the C90 have been made based on test calculations covering 100 time steps, and with the number of grid points and modes held fixed at MJ=385 and Nmodes=539. The results of these comparisons are shown in Fig. 2, where connect seconds and wallclock seconds are displayed as bar charts for all three modes of operation. Connect seconds are used on the C90 as a measure of the average time spent in concurrent CPUs. Figure 2 shows that good overlap is obtained with autotasking, and even better overlap is achieved by microtasking, with 4.12 processors accessed concurrently in microtasked mode compared to 2.55 processors in autotasked mode. The wallclock time is within 7% of the connect time for all three modes of operation. The improvement of performance with microtasking and autotasking over serial operation is directly proportional to the number of CPUs accessed concurrently, with the best real-time efficiency or lowest wallclock time for fastest turnaround obtained in microtasked mode. From now on, we, therefore, concentrate on results obtained with the microtasked version of KITE.

Tests of the KITE code in microtasking mode have been carried out in which the number of CPUs requested on the C90 was varied from 2 to 16 by setting the environment variable NCPUS to the appropriate value and the average number of concurrent CPUs achieved was recorded. As shown in Fig. 3, after 8 CPUs requested up to the maximum of 16, the average number of CPUs accessed is almost constant and hovers around 4.3. These tests were carried out for 100 time steps, MJ=385, and Nmodes=539. For production calculations with 5000 time steps or more, we average 5.9 processors when we request 16 processors. In either case, the number of CPUs accessed concurrently varies slightly depending on the machine workload at the time the calculations are performed.

Calculations have been performed in microtasked mode for 100 time steps with varying numbers of grid points and modes. All 16 processors were requested for these tests. Results of these studies are shown in Fig. 4 where the average number of concurrent CPUs obtained, Ncpus, is displayed as a function of the size of the calculation, MJ x Nmodes. The upper and lower set of data are for Nmodes= 539 and 201, respectively, and different MJs. The middle set of data is for MJ=385 and different Nmodes up to a maximum of 539. The number of CPUs accessed concurrently depends linearly on both the number of modes and the number of grid points, with a much stronger dependence on the number of modes. A global fit to the data yields the following scaling for the number of CPUs accessed concurrently

$$\text{Ncpus} = 0.835 + 5.3 \times 10^{-3}\ \text{Nmodes} + 1.4 \times 10^{-3}\ \text{MJ}\ . \qquad (12)$$

The stronger dependence of the number of concurrent CPUs on the number of modes just reflects the fact that parallelization is done over modes in the C90 version of KITE.

For these same calculations with varying numbers of grid points and modes, we have also plotted in Fig. 5 the average time spent in CPU or connect seconds, T_{Connect}, and the wallclock seconds, $T_{\text{wallclock}}$, as a function of the size of the calculation, MJ x Nmodes. In marked contrast to the three distinct dependencies observed in Fig. 4 for the number of concurrent CPUs as a function of calculation size, the data points for the connect seconds and wallclock seconds lie on a straight line; a very good fit to the data is

$$T_{\text{connect}}\ (\text{seconds}) = 0.38 + 7.21 \times 10^{-6}\ (\text{MJ x Nmodes})\ , \qquad (13)$$

and

$$T_{\text{wallclock}}\ (\text{seconds}) = 0.43 + 8.61 \times 10^{-6}\ (\text{MJ x Nmodes})\ . \qquad (14)$$

The linear dependence on MJ is expected because the number of operations in KITE scales as the number of grid points. However, while the number of operations is proportional to MJ x Nmodes for the matrix operations, it does scale as MJ x (Nmodes)^2 for the convolutions. This is indeed what is obtained in serial mode for problem sizes, such as the ones considered here, with Nmodes > MJ. However, as shown in Fig. 4, the concurrent CPUs scale as Nmodes so that more processors work simultaneously on the calculation as the number of modes increases, with the net effect that the connect seconds and wallclock seconds scale linearly with Nmode as well as MJ.

In performing production calculations,we have observed that better real-time efficiency can in fact be obtained. More concurrent CPUs can be accessed, and connect seconds per time step go down by a commensurate amount. Results of tests over 5000 time steps are displayed in Fig. 6(a) for concurrent CPUs and Fig. 6(b) for connect seconds as a function of number of modes. For reasons of affordability, we have kept the number of grid points fixed at 385. The results of Fig. 6(a) show that the average number of CPUs accessed concurrently is larger for these calculations than for those with 100 steps, with 5.9 concurrent processors obtained for the largest calculations with MJ=385 and Nmodes=539. A reasonable fit to the data of Fig. 6(a) is Ncpus (5000 steps) = Ncpus (100 steps) + 1.6. It is also clear from Fig. 6(b) that the connect seconds per time step are lower for the 5000 time step calculations than for those with 100 steps. A good fit to the data of Fig. 6(b) is $T_{connect}$ (5000 steps) = $T_{connect}$ (100 steps) - (0.35 + 5 x 10^{-4} Nmodes). These improvements are partly due to a better averaging of system performance with more steps, but also because the serial initialization is reduced over 5000 steps to much less than the 9% of the computational burden it accounted for over 100 steps as shown in Fig. 1. In fact, as shown in Fig. 6(b), when the data for 100 time steps are corrected for the serial setup, the connect seconds per step reduce to the values obtained from the calculations with 5000 time steps.

Microtasked calculations on all 16 processors of the C90 have been compared to identical calculations performed with the massively parallel version of KITE on all 128 processors of the ORNL Intel iPSC/860.[8] The results of these calculations with 385 grid points and a varying number of modes are shown in Fig. 7, where connect seconds per time step are displayed as a function of number of modes. Timings from tests over 5000 time steps were used for the C90 data. Real-time efficiency as measured by connect time is better by a factor of 10 on the C90 as compared to the iPSC/860. These comparisons are for the best version of the code running on each machine. Because of the memory limit of 8Mbyte per node on the iPSC/860, its version of the code must recalculate the matrix blocks every time step as they are needed. The C90 version stores all the matrices in its 268-Mword memory and only calculates them once. Furthermore, the number of processors is kept fixed on the iPSC/860, whereas the number of accessible processors on the C90 increases with the number of modes. Earlier extrapolations[8] of results obtained with KITE on the Intel iPSC/860 and Delta to the Intel Paragon, with its faster communications between processors, its better compiler optimization, and its 2048 processors show that it will be possible to obtain a gain in real-time efficiency equivalent to the C90 on the Paragon.

Production calculations of impurity radiation-driven drift waves as a model of turbulence at the edge of the TEXT tokamak have been performed on the C90 with the microtasked version of the computer code KITE. These calculations had 385 grid points and 539 modes. The time evolution of Eqs. (1–5) has been followed for 40,000 steps deep into the nonlinear saturated steady state. Even though these calculations were performed in the usual multiuser batch operation of the C90, we have been able to complete 20,000 contiguous steps over one weekend and have routinely completed 5000 steps overnight. As a measure of comparison, we were able to complete only 1000 steps every 2 or 3 days, depending on machine load, for the same size calculations in serial mode on the CRAY IIs and C90. This improvement in turnaround in microtasked mode on the C90 is due to the fact that these production calculations have been able to access an average of 5.9 processors concurrently. Because of this improvement in real-time efficiency, we are now able to achieve a nonlinear saturated steady state within a few days instead of a few weeks. Results of these production calculations are illustrated in the color plate displayed in Fig. 8. Color contours of the plasma density as a function of radius (vertical axis in each frame) and poloidal angle (horizontal axis) are shown as the nonlinear calculation advances in time.

We have concentrated for this paper on results from the KITE code because it is one of our most mature and computationally intensive production codes when applied to plasma edge turbulence calculations. However, the best microtasking performance that we have obtained to date on the C90 is with the DTEMFAR code, which solves a paradigmic one-equation model of dissipative trapped electron mode (DTEM) turbulence in slab geometry.[12] It uses an algorithm identical to KITE as far as matrix operations and mode convolutions are concerned, with the matrix solver accounting for 70% of the computational burden. In multiuser mode, we were able to access 7.42 concurrent CPUs, as compared to 1.65 processors for the autotasking version, for a calculation over 100 time steps with 534 grid points and 439 modes, corresponding to a memory size of 11 Mword. In dedicated, single-user mode, with the environment variable MP_DEDICATED activated, which does not work in multiuser mode, we have accessed up to 15.52 concurrent CPUs out of an absolute maximum of 16 processors.

5 Conclusions

In the next system configuration on the C90 at NERSC, the memory size available to users will be set to a maximum of 200 Mword. This will enable us to perform experimentally relevant calculations of plasma edge turbulence with higher radial resolution and including more modes to increase mode couplings and radial mixing. Using Eq. (11), we will be able to fit 500 grid points and 1000 modes in memory, for a memory size of Msize= 190 Mword, only slightly below the maximum allowed. Using the scalings of Eqs. (12) and (13) obtained from test calculations of 100 time steps, we could expect accessing 6.8 CPUs concurrently out a maximum possible of 16 and respectable timings of 4 connect seconds per time step. With the additional improvements observed over 5000 time steps in average number of concurrent CPUs and connect seconds per step, we can expect to access 8.5 processors concurrently and timings of 3.1 connect seconds per step for the largest calculations. These anticipated improvements in real-time efficiency are due to the 16 fast processors, large memory of the C90, and the number of processors accessed concurrently scaling linearly with problem size for our microtasking implementation. Comparable performance for the same size problem is also anticipated on the Intel Paragon.

Acknowledgments

We thank Drs. J. Sheffield and R. A. Dory of ORNL and Dr. C. W. McCurdy of NERSC for their support and encouragement in carrying out this project.

This research was sponsored by the Office of Fusion Energy, U. S. Department of Energy, under contract DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

References

1. Ch. P. Ritz, R. V. Bravenec, P. M. Schoch, R. D. Bengtson, J. A. Boedo, J. C. Forster, K. W. Gentle, Y. He, R. L. Hickcok. Y. J. Kim, H. Lin, P. E. Phillips, T. L. Rhodes, W. L. Rowan, P. M. Valanju, and A. J. Wootton, Phys. Rev. Lett. **62**, 1844 (1989).
2. B. A. Carreras, N. Dominguez, J. B. Drake, J. N. Leboeuf, L.A. Charlton, J. A. Holmes, D. K. Lee, V. E. Lynch and L. Garcia, Int. J. Supercomput. Appl. **4**, 97 (1990).
3. A. S. Ware, P. H. Diamond, B. A. Carreras, J. N. Leboeuf and D. K. Lee, Phys. Fluids B **4**, 102 (1992).
4. J. N. Leboeuf, D. K. Lee, B. A. Carreras, N. Dominguez, J. H. Harris, C. L. Hedrick, C. Hidalgo, J. A. Holmes, J. R. Ruiter, P. H. Diamond, A. S. Ware, Ch. P. Ritz, A. J. Wootton, W. L. Rowan, and R. V. Bravenec, Phys. Fluids B **3**, 2291 (1991).
5. L. Garcia, H. R. Hicks, B. A. Carreras, L. A. Charlton, and J. A. Holmes, J. Comput. Phys. **65**, 253 (1986).
6. J. B. Drake, B. F. Lawkins, B. A. Carreras, H. R. Hicks, and V. E. Lynch, "Implementation of a 3-D Nonlinear MHD Calculation on the Intel Hypercube," ORNL-6335, Martin Marietta Energy Systems, Inc., Oak Ridge National Laboratory, Oak Ridge, Tennessee, 1987.
7. V. E. Lynch, B. A. Carreras, J. B. Drake, J. N. Leboeuf, and J. R. Ruiter, Comput. Sys. Eng. **2**, 299 (1991).

8. V. E. Lynch, B. A. Carreras, J. B. Drake, J. N. Leboeuf, and P. Liewer, "Performance of a Plasma Fluid Code on the Intel Parallel Computers," Proceedings of Supercomputing '92, Minneapolis, Minnesota, November 16-20, 1992, pp. 286-293, IEEE Computer Society, Los Alamitos, California, 1992.

9. H. R. Hicks, and V. E. Lynch, J. Comput. Phys. **63**, 140 (1986).

10. A. C. Hindmarsh, "Solution of Block-Tridiagonal Systems of Algebraic Equations," UCID-30150, Lawrence Livermore National Laboratory, Livermore, California, 1977.

11. H. R. Hicks, B. A. Carreras, J. A. Holmes, D. K. Lee, and B. V. Waddell, J. Comput. Phys. **44**, 46 (1981).

12. B. A. Carreras, K. L. Sidikman, P. H. Diamond, P. W. Terry, L. Garcia, Phys. Fluids B **4**, 3115 (1992).

Figure Captions

Figure 1: Profile of the plasma fluid turbulence calculations with the micortasked computer code KITE on the CR! Y-MP C90. The pie chart indicates the breakdown of the computational burden for 100 time steps, MJ= 385 grid points and Nmodes= 539 modes between matrix operations (SOLBT/SOL), the mode convolutions (MULT), the tridiagonal solver (TRDG) and the serial startup routines.

Figure 2: Timings of KITE for 100 time steps, 385 grid points and 539 modes o:: the C90 in serial, autotasked and multitasked modes. Wallclock seconds in light shading and connect seconds in dark shading (average time spent in CPU) are displayed in a bar chart for all three modes of operation. Microtasking provides the best real-time efficiency.

Figure 3: Average number of CPUs accessed in microtasking mode as a function of number of CPU's requested for runs of KITE with 385 grid points and 539 modes on the C90 for 100 time steps.

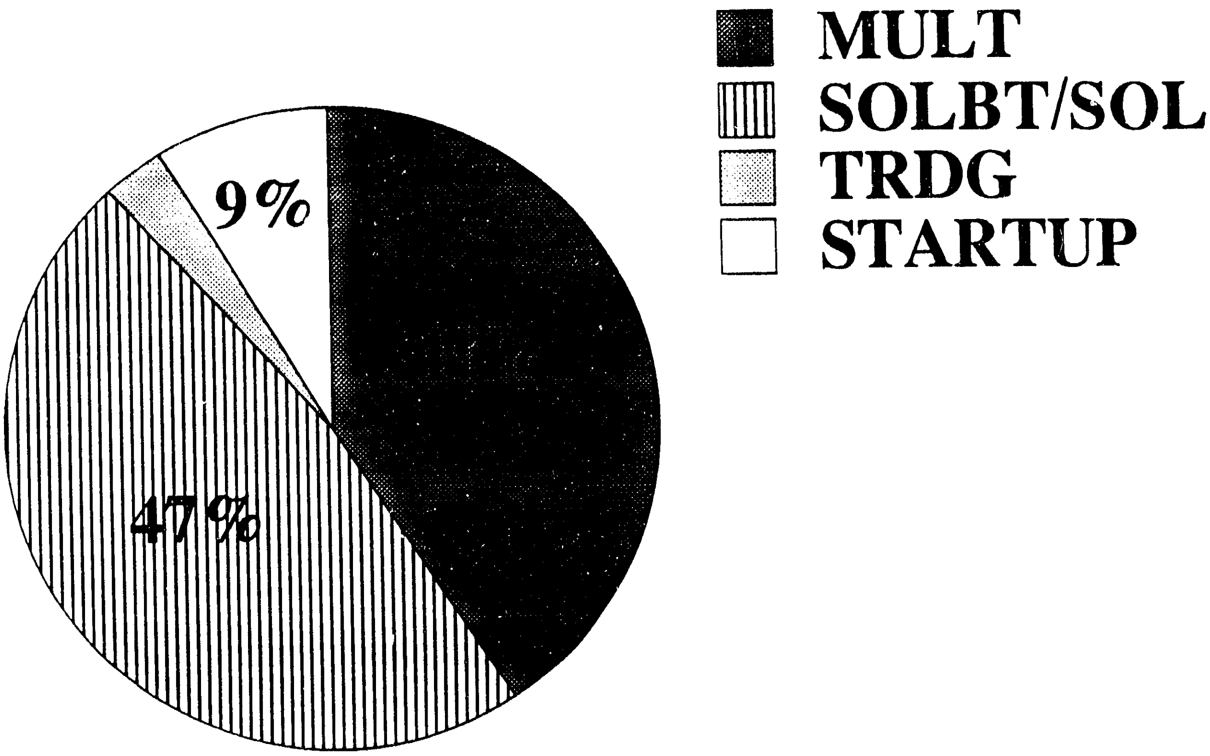
Figure 4: Average number of concurrent CPUs in microtasking mode as a function of calculation size MJ x Nmodes for calculations over 100 time steps with KITE on the C90. The upper and lower data points are for Nmodes= 539 and 201 and varying MJ. The middle data points are for MJ=385 and varying Nmodes.

Figure 5: Connect seconds (solid circles) and wallclock seconds (open circles) as a function of calculation size MJ x Nmodes for calculations over 100 time steps with KITE in microtasking mode on the C90.

Figure 6: Comparison between microtasked C90 KITE calculations for 100 (solid circles) and 5000 (open circles) time steps: a) average number of concurrent CPU's as a function of number of modes and b) connect seconds per time step as a function of number of modes. The connect seconds for 100 steps with startup substracted are plotted as solid squares.

Figure 7: Connect seconds per time step for identical calculations with the microtasked version of KITE on the sixteen processors of the C90 and with the massively parallel version of KITE on all 128 processors of the Intel iPSC/860.

Figure 8: Color contours of the plasma density as a function of radius (vertical axis in each frame) and poloidal angle (horizontal axis) are shown as the nonlinear production calculation advances in time.



Percent of CPU time

FIGURE 1

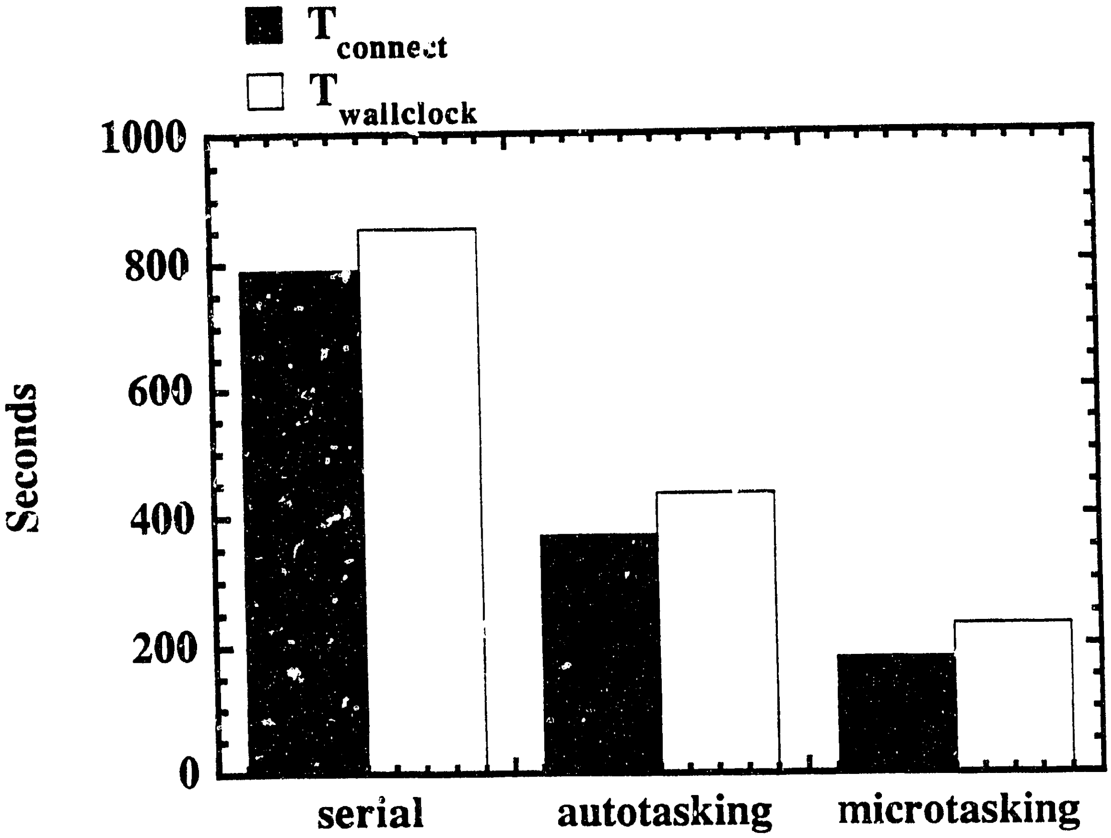

■ T connect
□ T wallclock
1000
800
600
400
200
0
Seconds
serial
autotasking
microtasking

FIGURE 2

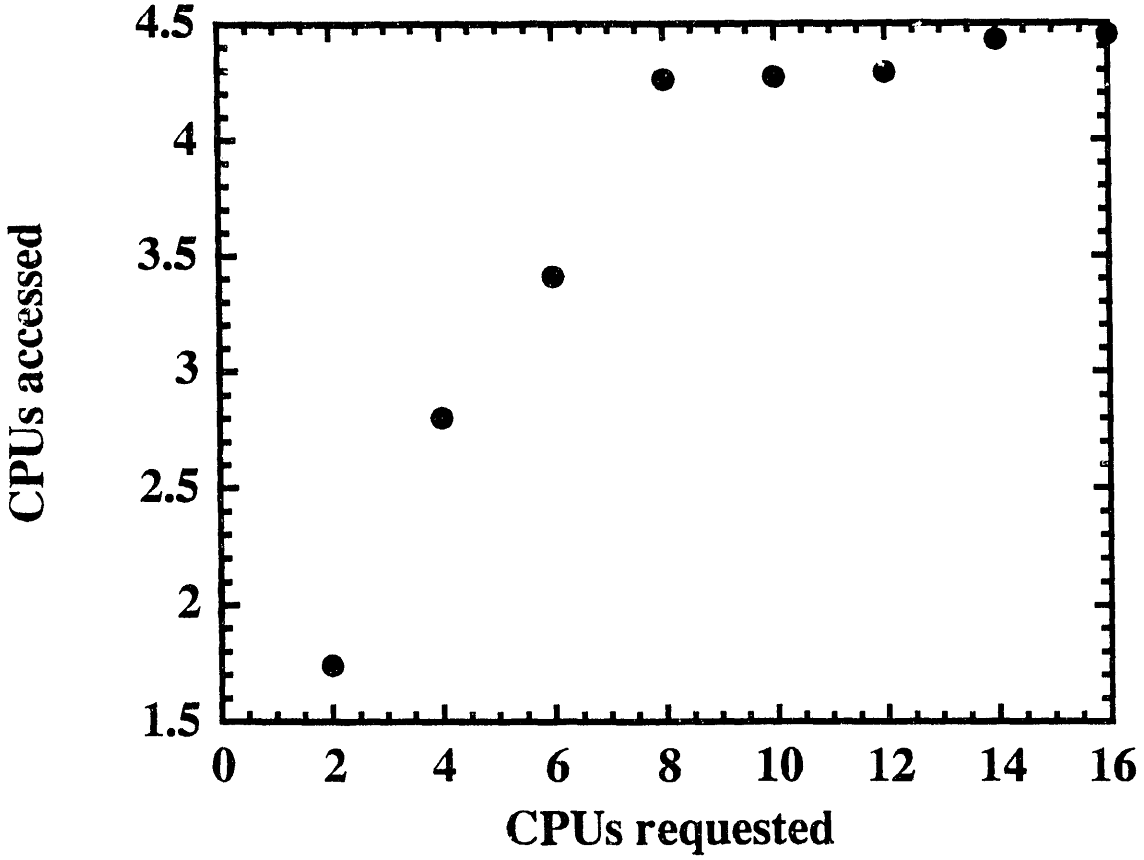


FIGURE 3

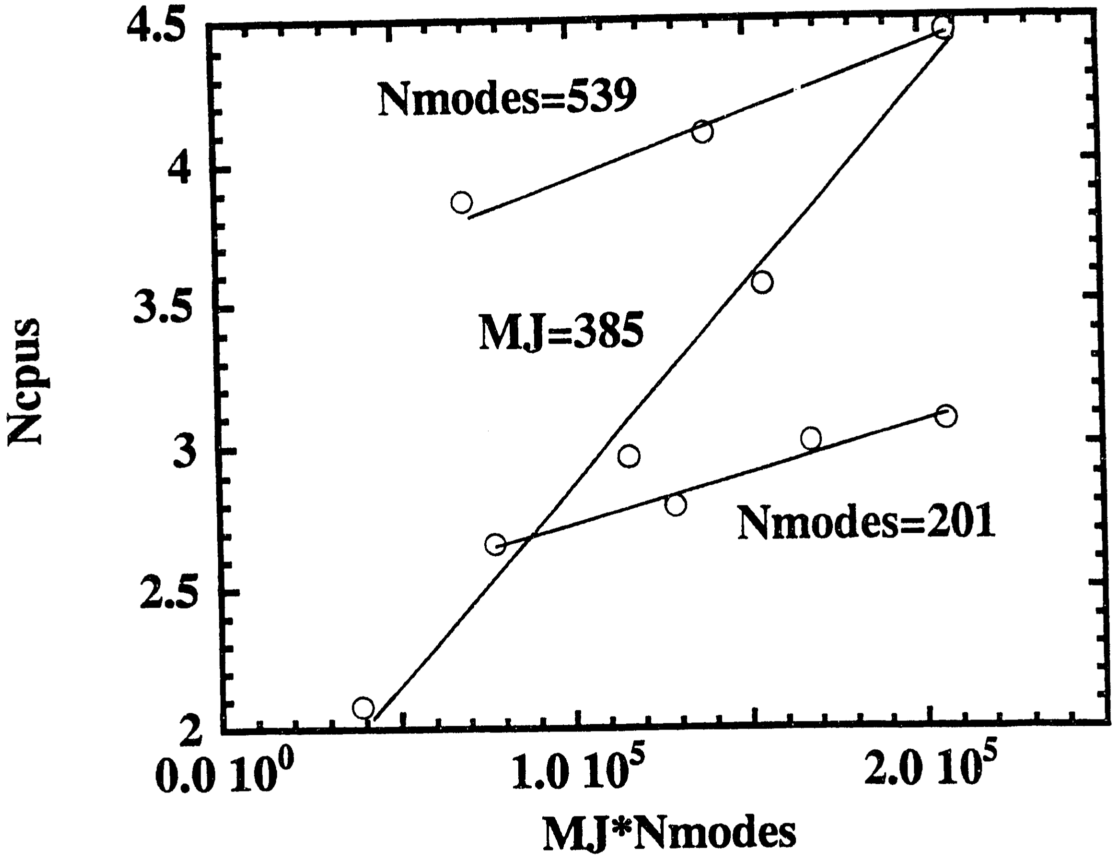

4.5
4
3.5
3
2.5
2
Ncpus
0.0 10^0
1.0 10^5
2.0 10^5
MJ*Nmodes
Nmodes=539
MJ=385
Nmodes=201

FIGURE 4

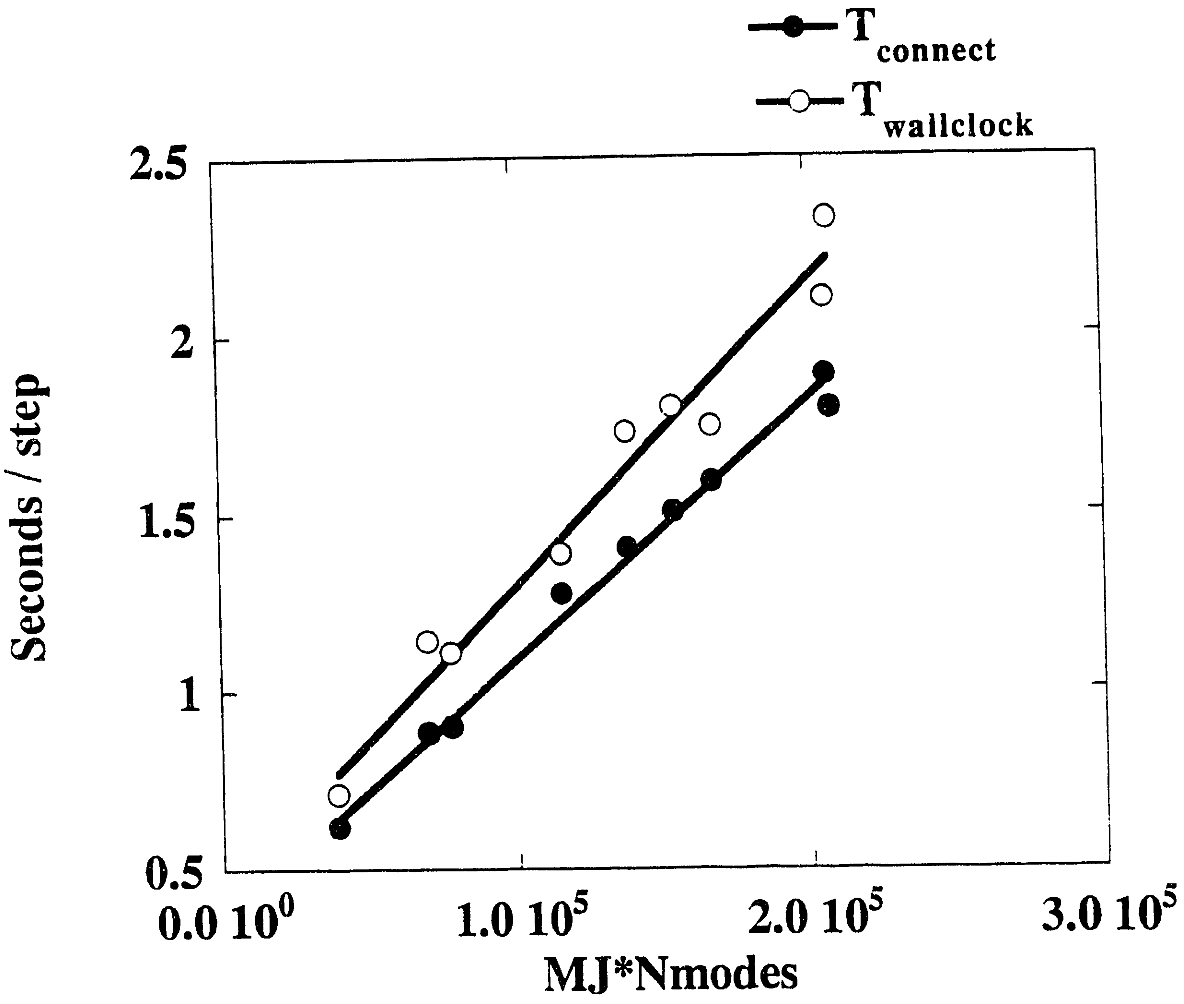

T connect
T wallclock
2.5
2
1.5
1
0.5
Seconds / step
0.0 10^0
1.0 10^5
2.0 10^5
3.0 10^5
MJ*Nmodes

FIGURE 5

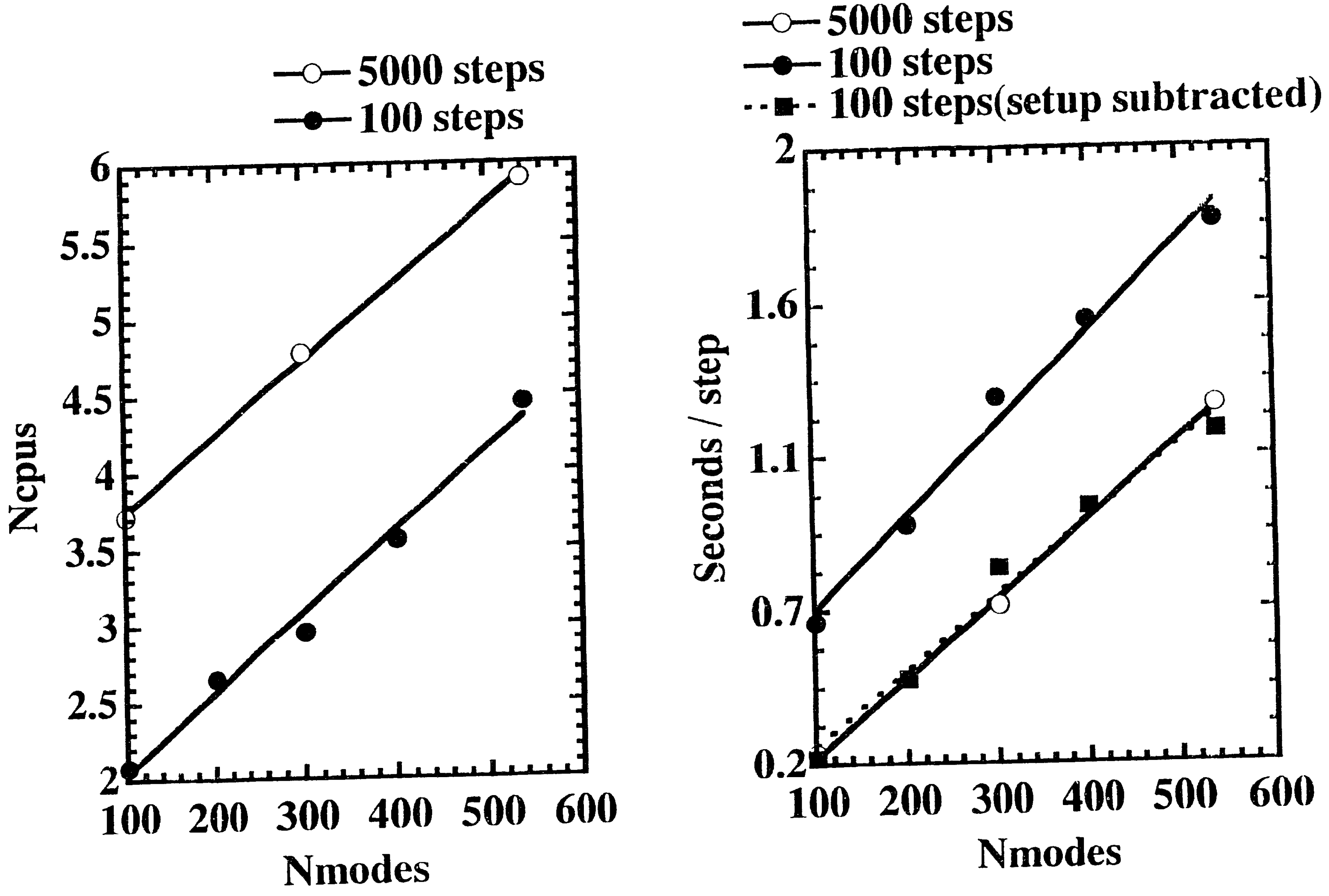

5000 steps
100 steps
Ncpus
6
5.5
5
4.5
4
3.5
3
2.5
2
100 200 300 400 500 600
Nmodes
5000 steps
100 steps
100 steps(setup subtracted)
Seconds / step
2
1.6
1.1
0.7
0.2
100 200 300 400 500 600
Nmodes

FIGURE 6

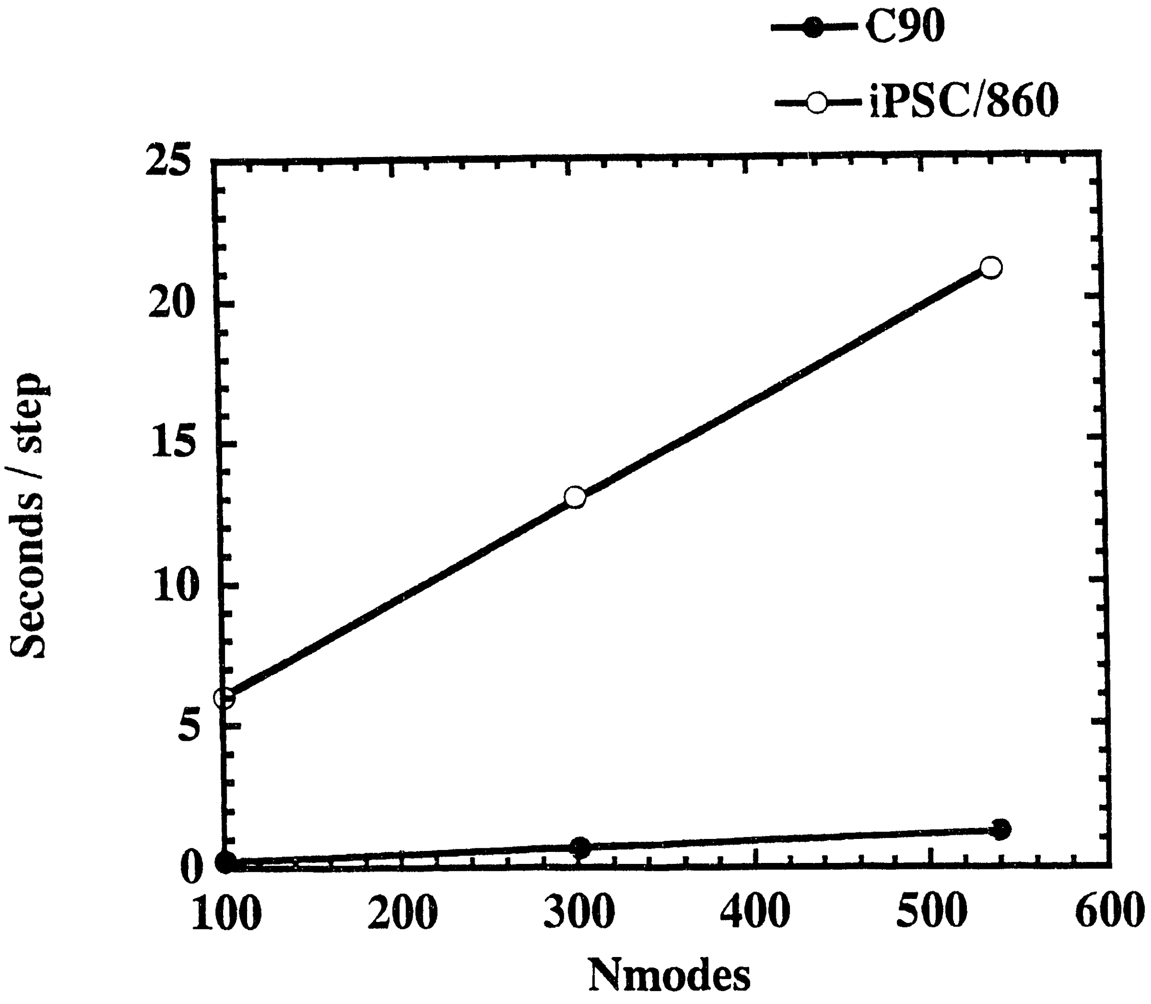
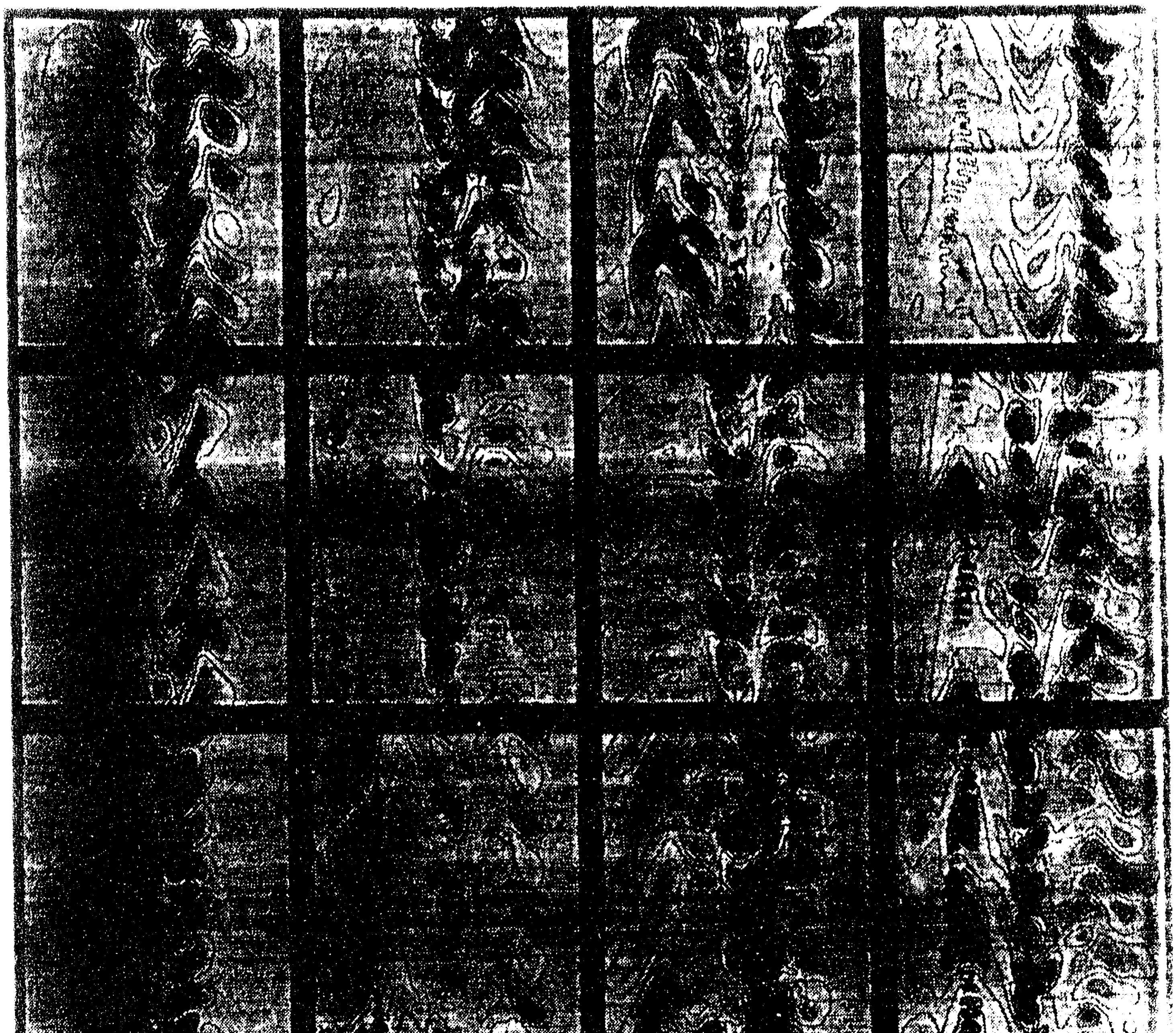

C90
iPSC/860
Seconds / step
0
5
10
15
20
25
100
200
300
400
500
600
Nmodes

FIGURE 7



DATE
FILMED
10/13/93

END