ornl

OAK RIDGE NATIONAL LABORATORY

LOCKHEED MARTIN

ORNL/TM-13271

Parallel Community Climate Model: Description and User's Guide

J. B. Drake
R. E. Flanery
B. D. Semeraro
P. H. Worley
I. T. Foster
J. G. Michalakes
J. J. Hack
D. L. Williamson

MANAGED AND OPERATED BY
LOCKHEED MARTIN ENERGY RESEARCH CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

ORNL-27 (3-96)

ORNL/TM-13271

Computer Science and Mathematics Division

Mathematical Sciences Section

PARALLEL COMMUNITY CLIMATE MODEL: DESCRIPTION AND USER'S GUIDE

J. B. Drake, R. E. Flanery, B. D. Semeraro, P. H. Worley
Oak Ridge National Laboratory

I. T. Foster, J. G. Michalakes
Argonne National Laboratory

J. J. Hack, D. L. Williamson
National Center for Atmospheric Research

Date Published: July 15, 1996

Research sponsored by the U.S. Department of Energy CHAMMP Program of the Office of Health and Environmental Research.

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831
managed by
Lockheed Martin Energy Research Corp.
for the
U.S. DEPARTMENT OF ENERGY
under Contract No. DE-AC05-96OR22464

Contents

Acknowledgments

This report is one of a series of documents describing the use of parallel computers for global climate modeling. The work reported is sponsored by the CHAMMP program of the Department of Energy's Office of Energy Research, Environmental Sciences Division. We gratefully acknowledge the support of CHAMMP program director, David Bader, who encouraged us to stay the course for a production quality implementation on new and experimental massively parallel computing platforms.

We would like to thank several of our collegues and students who have contributed to the project over the past few years: Rick Stevens and Hans Kaper (of ANL) whose enthusiasm early in the project helped define a good working relationship among the collaborators; Paul Swarztrauber and Dick Sato (of NCAR), who along with Steve Hammond and Rich Loft (also of NCAR), who freely shared ideas and apporaches to parallelism for high performance computers; Jim Rosinski (NCAR) who assisted with the validation studies and Tim Sheehan (ORNL), David Walker (ORNL), Brian Toonen, Ravi Nanjundiah and Jace Mogill (of ANL) who performed many of the code optimizations.

PARALLEL COMMUNITY CLIMATE MODEL: DESCRIPTION AND USER'S GUIDE

J. B. Drake, R. E. Flanery, B. D. Semeraro, P. H. Worley
Oak Ridge National Laboratory

I. T. Foster, J. G. Michalakes
Argonne National Laboratory

J. J. Hack, D. L. Williamson
National Center for Atmospheric Research

Abstract

This report gives an overview of a parallel version of the NCAR Community Climate Model, CCM2, implemented for MIMD massively parallel computers using a message-passing programming paradigm. The parallel implementation was developed on an Intel iPSC/860 with 128 processors and on the Intel Delta with 512 processors, and the initial target platform for the production version of the code is the Intel Paragon with 2048 processors. Because the implementation uses a standard, portable message-passing libraries, the code has been easily ported to other multiprocessors supporting a message-passing programming paradigm.

The parallelization strategy used is to decompose the problem domain into geographical patches and assign each processor the computation associated with a distinct subset of the patches. With this decomposition, the physics calculations involve only grid points and data local to a processor and are performed in parallel. Using parallel algorithms developed for the semi-Lagrangian transport, the fast Fourier transform and the Legendre transform, both physics and dynamics are computed in parallel with minimal data movement and modest change to the original CCM2 source code.

Sequential or parallel history tapes are written and input files (in history tape format) are read sequentially by the parallel code to promote compatibility with production use of the model on other computer systems.

A validation exercise has been performed with the parallel code and is detailed along with some performance numbers on the Intel Paragon and the IBM SP2. A discussion of reproducibility of results is included.

A user's guide for the PCCM2 version 2.1 on the various parallel machines completes the report. Procedures for compilation, setup and execution are given. A discussion of code internals is included for those who may wish to modify and use the program in their own research.

1. INTRODUCTION

The Computer Hardware, Advanced Mathematics, and Model Physics (CHAMMP) program [9] seeks to provide climate researchers with an advanced modeling capability for the study of global change issues. As a first goal in the program, current state-of-the-art models have been implemented on massively parallel computers, allowing an increase in spatial resolution. Accomplishment of this task provides the groundwork for the second goal, which is the development of a coupled oceanic and atmospheric model to produce an advanced climate model with improved process representation.

Toward the realization of the program's first objective, a parallel version of the Community Climate Model CCM2 has been developed for scalable parallel MIMD distributed memory computers. Development of the PCCM2 serves two objectives. First, it provides a performance benchmark to indicate how well the current massively parallel computers perform in comparison with machines of a more conventional architecture. Second, it defines the starting point of a development path for future climate models that will couple atmospheric and oceanic models. These models will incorporate more comprehensive physics, different numerical methods and may be written in other parallel programming styles for computers with many thousands of processors.

PCCM2 uses a message-passing, domain decomposition approach, in which each processor is allocated responsibility for computation associated with one part of the computational grid. Messages between processors are generated when data on one processor is needed to complete the computational task of another processor. Much of the research effort associated with development of a parallel code for a distributed computation is concerned with identifying efficient decomposition and communication strategies while at the same time balancing the computational load among the processors. In the PCCM2, this task is complicated by the need to support both semi-Lagrangian transport (for moisture) and spectral transforms (for other fields). Load balancing plays an important role for the physics calculations as well as the spectral dynamics. Parallel input/output are also required for efficient use of a massively parallel processor..

This report gives a brief overview of the parallel algorithms required to implement CCM2 and describes performance issues on distributed memory multiprocessors. The target machine for the parallel code is the Intel Paragon, and a single program, multiple data (SPMD) programming paradigm with message passing was adopted. The code uses the Message Passing Interface (MPI) standard for interprocessor communication providing a degree of portability across platforms. Optionally, the code can be configured so that message passing uses the Parallel Virtual Machine (PVM) constructs for execution across a heterogeneous network of computers, or machine specific (native) communication libraries.

2. HISTORY OF THE NCAR COMMUNITY CLIMATE MODEL

Over the last decade, the NCAR Climate and Global Dynamics (CGD) Division has provided a comprehensive, three-dimensional global atmospheric model to university and NCAR scientists for use in the analysis and understanding of global climate. Because of its widespread use, the model was designated a Community Climate Model (CCM). The original version of the CCM (CCM0A) was based on the Australian spectral model developed by W. Bourke, B. McAvaney, K. Puri, and R. Thurling [6] [26] and was described in [37]. An important broadening of the concept of the NCAR community model occurred in late 1981 with NCAR's decision to utilize the same basic code for global forecast studies (both medium- and long-range) and for climate simulation. Economy and increased efficiency could then be achieved by documenting and maintaining only one set of modular codes. The use of one basic model for both forecasting and climate studies was also seen to have great potential scientific value since a major part

of medium-range (one- to two-week) forecast error is due to the drift toward a model climate which differs from that of the atmosphere. Thus, improvements in the climate aspects of the model should lead to improvements in forecasts. Similarly, many physical parameterizations are deterministic rather than statistical in the sense that they are based on the details of the current model state rather than on some past statistical properties. Thus, performance aspects of parameterized physics can be studied, improved, and verified by examining them in a forecast mode.

Because of the extension of the role of the CCM to include forecast studies as well as climate studies, and because of the expected widespread use for both purposes by university as well as NCAR scientists, a versatile, modular, and well-documented code became essential. The initial version designated CCM0B was developed to meet these requirements. This code grew out of an adiabatic, inviscid version of the spectral model developed at the European Centre for Medium Range Weather Forecasts (ECMWF) by A.P.M. Baede, M. Jarraud, and U. Cubasch [2] to which physical parameterizations and numerical approximations matching those of CCM0A were added. The physical parameterizations included the radiation and cloud routines developed at NCAR [28] and convective adjustment, stable condensation, vertical diffusion, surface fluxes, and surface-energy-balance prescription developed at the Geophysical Fluid Dynamics Laboratory (GFDL) [32] [24] [33] [19]. The vertical and temporal finite differences matched those of the Australian spectral model [6]. The resulting model code, designated CCM0B, was described in a series of technical notes which included a User's Guide [31], a description of model subroutines [39], a detailed description of the continuous algorithms [38], and circulation statistics from long January and July simulations [42].

The advantages of the community model concept, in which many scientists use the same basic model for a variety of scientific studies, were demonstrated in workshops held at NCAR in July 1985 [1], July 1987 [44], and July 1990 [45]. Fundamental strengths and weaknesses of the model have been identified at these workshops through the presentation of a diverse number of applications of the CCM. Much constructive dialogue has taken place between experts in several disciplines at these meetings leading to continued improvements in the CCM with each release.

CCM0B was followed with CCM1 in July of 1987 and included a similar set of detailed technical documentation [40] [5] [4] [43] [16]. Substantial changes were incorporated in the radiation scheme, including a new solar albedo parameterization accounting for the solar zenith-angle dependence of albedo on various surface types, improvements to the absorption of solar radiation by H_2O and O_2, improvements to the long wave absorptance algorithms for H_2O, CO_2 and O_3, changes to account for the liquid water content of stratiform clouds in determining their emissivity, and incorporation of a new finite-difference scheme in the long wave part of the radiation model (see [22]). The vertical finite-difference approximations were modified to conserve energy without adversely affecting the model simulations, and frictional heating was included so that the momentum diffusion produced a corresponding heating term in the thermodynamic equation. The latter two improvements resulted in the energy in the model being conserved to the order of one W m^{-2} and moisture to one-hundredth W m^{-2} energy equivalent over 90-day periods. The horizontal diffusion was modified to a ∇^4 form in the troposphere and included a partial correction for evaluating the operator on pressure surfaces rather than sigma surfaces. The local moisture adjustment was generalized to provide for a global horizontal borrowing [30] in a conserving manner. The vertical diffusion was converted to a nonlinear form for which the eddy-mixing coefficient depended on local shear and stability. The diffusion was applied throughout the atmosphere rather than only below 500 mb as done in CCM0B, which eliminated the need for a dry convective adjustment in the troposphere. The surface drag coefficient was made a function of stability following Deardorff [8] and the equation of state was modified to formally account for moisture in the atmosphere (i.e., virtual temperature was used where appropriate and the variation with moisture of the specific heat at

constant pressure was accounted for). In addition to the above changes to the physics, CCM1 included new capabilities such as a seasonal mode in which the specified surface conditions vary with time, and an optional interactive surface hydrology [7] which followed the formulation presented by Manabe [24]. Since the CCM1 could also be used as a global forecast model, codes to prepare initial data in the CCM history tape format from analyzed observed atmospheric data, such as FGGE Level IIIb analyses [25], and codes to perform nonlinear normal mode initialization [13] [12] were made available.

As a result of the biennial CCM workshops mentioned earlier, the underlying philosophy of the CCM was modified. The original intent was to provide a stable, robust model applicable to a variety of problems. Thus the most recent developments in model physics were deliberately not included in the physical parameterizations in order to provide stable, well known algorithms. This approach leads to more straightforward interpretation of experimental results. The discussions in the workshops highlighted the strengths of this approach, but also pointed out the need for a state-of-the-science model to address many of the very important climate questions being raised today. The decision was made that the next version of the CCM should be brought up to date in all its aspects. Thus the most recent version of the CCM, CCM2, which is expected to be released during the summer of 1991, incorporates the most ambitious set of changes to date.

The bulk of the effort in the NCAR Climate Modeling Section over the last several years has been to improve the physical representation of a wide range of key climate processes in the CCM, including clouds and radiation, moist convection, the planetary boundary layer, and transport. The resulting changes to the model have resulted in a significantly improved simulation and fundamentally better climate model. On the parameterized physics side, changes include the incorporation of a diurnal cycle, along with the required multilayer heat capacity soil model, and major improvements to the radiation scheme, including a δ-Eddington solar scheme (18 spectral bands), a new cloud albedo parameterization, a new cloud emissivity formulation using liquid water path length, a new cloud fraction parameterization, and a Voigt correction to infrared radiative cooling in stratosphere. The moist adiabatic adjustment procedure has been replaced with a stability-dependent mass flux representation of moist convection, and an explicit planetary boundary layer parameterization is now included, along with a modified gravity-wave drag parameterization which introduces changes in the generation and vertical distribution of momentum drag as well as providing the framework for a longer-term non-isotropic formalism.

On the dynamics side, a semi-Lagrangian transport scheme is now the default for water vapor as well as an arbitrary number of other scalar fields (e.g., cloud water variables, chemical constituents, etc.) and the vertical coordinate makes use of a hybrid (or generalized σ) formulation. The model has been developed for a standard horizontal spectral resolution of T42 (2.8 degrees by 2.8 degrees transform grid) with 18 vertical levels and a top at approximately 2.9 mb. The entire model code is also being entirely rewritten with three major objectives: much greater ease of use and modification; conformation to a plug-compatible physics interface; and the incorporation of single-job multitasking capabilities.

CCM2 provides the basis for a large body of experimental and developmental efforts by a large community of university and NCAR climate investigators, many of whom may not be directly involved in the CHAMMP initiative. Because of the community nature of the enterprise, new methods and process modules are continually emerging. The new methods will be incorporated in future releases and versions of the model as seems appropriate for computer efficiency and the requirement for increased capabilities.

3. PARALLEL ALGORITHMS

There are two major dynamics algorithms in the CCM2 code, the spectral transform method [11], [23], [27] and the semi-Lagrangian transport method [41]. The process models for radiation, clouds, surface moisture and temperature share the common feature that they are coupled horizontally only through the dynamics. We lump all these processes under the general term "physics" and note that the physics calculations are independent for each vertical column of grid cells in the model.

The independence of the physics calculations for each horizontal grid point is the primary source of parallelism in the PCCM2. By partitioning the horizontal grid points into blocks and assigning them to processors, a decomposition of the three dimensional, physical space data structures is defined. This decomposition allows the physics calculation for each vertical column of grid points to be performed without the need for interprocessor communication.

The dynamics calculations make use of the spectral transform method for the approximation of all horizontal derivates in the equations except those of the advective term in the moisture transport equation. The spectral transform involves two stages or two separate transforms, the fast Fourier transform (FFT) and the Legendre transform. The Fourier transform integrates information along each east-west grid line in the longitudinal direction. In the spectral transform from grid space to spectral space, this is followed by a Legendre transform integrating the results of the FFT in the north-south, or latitudinal, direction. Thus, the spectral transform operates on data "globally" in that information from each horizontal grid point, and from each processor, contributes to each spectral coefficient.

The FFT can be performed effectively in parallel [10] by exploiting the fact that there are multiple grid lines to be transformed at any one time. Two options for parallel FFT's are currently implemented in PCCM2. One is based on transposing the FFT data so that entire longitude lines are contained in a processor. Since there are multiple lines for each longitude and level, the parallel FFT's can be spread nearly equally among the processors. At the conclusion of the FFT, the data are transposed back to the original processor distribution. The other option is a distributed parallel FFT, where each longitude line is divided across a number of processors. A vector sum algorithm is used to calculate the Legendre transform in PCCM2. The parallelization of the spectral transforms in PCCM2 has driven most of the design decisions adopted for the organization of the data structures.

The advective terms in the moisture equation are approximated using a semi-Lagrangian transport method. The method updates the value of the moisture field at a grid point (the arrival point, A) by first establishing a trajectory through which the particle arriving at A has moved during the current timestep. From this trajectory the departure point, D, is calculated and the moisture field is interpolated at D using shape preserving interpolation. All the calculations involve physical space (grid point) data, and are decomposed over the processors with the same mesh decomposition used to parallelize the physics and the spectral transform. The parallel implementation of this algorithm uses the fact that timestep constraints imposed by the Eulerian dynamics limit the distance between the departure and arrival points in the latitude direction. By extending the arrays in each processor, thus "overlapping" regions assigned to neighboring processors, and updating the overlapped portion of the array prior to each timestep via interprocessor communication, the calculations in the different processors can proceed independently.

The rest of this section describes in more detail the data decomposition and parallel algorithms for the Legendre transform and the semi-Lagrangian transport. A detailed description and comparison of parallel algorithms can be found in the series of papers. [14, 46, 10, 15]. More details on the parallel CCM2, and a description of load balancing techniques used in the physics component of the model, can be found in other papers [21, 47]. A data parallel implementation of PCCM2 is described in [18] and a modestly parallel implementation for shared

memory and distributed memory (1-D decomposition) is described in [17].

3.1. The Spectral Transform Algorithm

The spectral transform method is based on a dual representation of the scalar fields in terms of a truncated series of spherical harmonic functions and in terms of values on a rectangular tensor-product grid whose axes represent longitude and latitude. Representations of the state variables in spectral space are the coefficients of an expansion in terms of complex exponentials and associated Legendre functions,

$$\xi(\lambda,\mu) = \sum_{m=-M}^{M} \sum_{n=|m|}^{N(m)} \xi_n^m P_n^m(\mu) e^{i\cdot m\lambda}, \tag{1}$$

where $P_n^m(\mu)$ is the (normalized) associated Legendre function [34] and $i = \sqrt{-1}$. The spectral coefficients are then determined by the equation

$$\xi_n^m = \int_{-1}^{1} \left[\frac{1}{2\pi} \int_0^{2\pi} \xi(\lambda,\mu) e^{-i\cdot m\lambda} d\lambda \right] P_n^m(\mu) d\mu \equiv \int_{-1}^{1} \xi^m(\mu) P_n^m(\mu) d\mu \tag{2}$$

since the spherical harmonics $P_n^m(\mu)e^{i\cdot m\lambda}$ form an orthonormal basis for square integrable functions on the sphere. In the truncated expansion, M is the highest Fourier mode and $N(m)$ is the highest degree of the associated Legendre function in the north-south representation. Since the physical quantities are real, ξ_n^{-m} is the complex conjugate of ξ_n^m, and only spectral coefficients for nonnegative modes need to be calculated.

To evaluate the spectral coefficients numerically, a fast Fourier transform (FFT) is used to find $\xi^m(\mu)$ for any given μ. The Legendre transform is approximated using a Gaussian quadrature rule. Denoting the Gauss points in $[-1,1]$ by μ_j and the Gauss weights by w_j,

$$\xi_n^m = \sum_{j=1}^{J} \xi^m(\mu_j) P_n^m(\mu_j) w_j. \tag{3}$$

Here J is the number of Gauss points. (For simplicity, we will henceforth refer to (3) as the forward Legendre transform.) The point values are recovered from the spectral coefficients by computing

$$\xi^m(\mu) = \sum_{n=|m|}^{N(m)} \xi_n^m P_n^m(\mu) \tag{4}$$

for each m (which we will refer to as the inverse Legendre transform), followed by FFTs to calculate $\xi(\lambda,\mu)$.

The tensor-product grid in physical space is rectangular with I grid lines evenly spaced along the longitude axis and J grid lines along the latitude axis placed at the Gaussian quadrature points used in the forward Legendre transform. To allow exact, unaliased transforms of quadratic terms the following relations are sufficient: $J \geq (3M+1)/2$, $I = 2J$, and $N(m) = M$ [27]. Using $N(m) = M$ is called a triangular truncation because the (m,n) indices of the spectral coefficients make up a triangular array. The examples in the rest of this section will assume a triangular truncation is used.

3.2. Data Decompositions

In the spectral transform algorithm, computations are performed in both the physical and spherical harmonic (or *spectral*) domains, and transforming from one domain to the other involves passing through the *Fourier domain*, whose coordinates are Fourier wavenumber and latitude coordinates. Thus, we must be concerned with the distribution of data in three domains.

In specifying the domain decompositions, the multiprocessor is viewed as a logical $P \times Q$ two dimensional processor grid. (P and Q are currently compile-time parameters for PCCM2.) For the physical domain, the latitudinal dimension is partitioned into $2Q$ intervals, each containing $J/2Q$ consecutive grid lines along the latitude axis. Each processor row is assigned two of these intervals, one from the northern hemisphere, and the reflected latitudes in the southern hemisphere. This assignment allows symmetry to be exploited in the Legendre transform. The assignment also restricts Q, the number of processor rows, to be no larger than $J/2$.

The longitudinal dimension is partitioned into P equal intervals, with each interval being assigned to a different processor column. The resulting "block" decomposition of the physical domain is illustrated in Fig. 1 for a small example.

The Fourier domain can be regarded as a wavenumber-latitude grid, so, like the physical domain, the Fourier domain is two-dimensional. However, a different decomposition is used. The differences arise because of the way in which the FFT algorithm permutes the ordering of the output Fourier coefficients [36]. But, modulo this reordering, the wavenumber "dimension" is partitioned into P sets of consecutive wavenumbers, with each set being assigned to a different processor column. The partitioning function in the latitude direction is the same as in the physical domain. See Fig. 1 for an example decomposition.

The spectral domain can also be regarded as two dimensional. For example, for a triangular truncation, the domain is a triangular grid whose axes are wavenumber and degree of associated Legendre polynomial (n). The wavenumber "dimension" is partitioned and assigned to processors exactly as for the Fourier domain, i.e. the wavenumbers are reordered, partitioned into consecutive blocks, and assigned to the processor columns. But, unlike the physical and Fourier domains, the remaining dimension in the spectral domain is not partitioned. Instead, all spectral coefficients associated with a given wavenumber are duplicated across all processors in the processor column to which that wavenumber was assigned. It is this duplication that allows the vector sum algorithm described below to be used. Again, see Fig. 1 for an example decomposition.

Note that in a triangular truncation, the number of spectral coefficients associated with a given Fourier wavenumber decreases as the wavenumber increases. Without the reordering of the wavenumbers caused by the FFT, this would cause a noticeable load imbalance, with processor columns associated with larger wavenumbers having very few spectral coefficients. The reordering of the wavenumbers leads to a much better load balance.

3.3. Parallel Legendre Transform

The forward and inverse Legendre transforms are

$$\xi_n^m = \sum_{j=1}^{J} \xi^m(\mu_j) P_n^m(\mu_j) w_j$$

and

$$\xi^m(\mu_j) = \sum_{n=|m|}^{N(m)} \xi_n^m P_n^m(\mu_j)$$

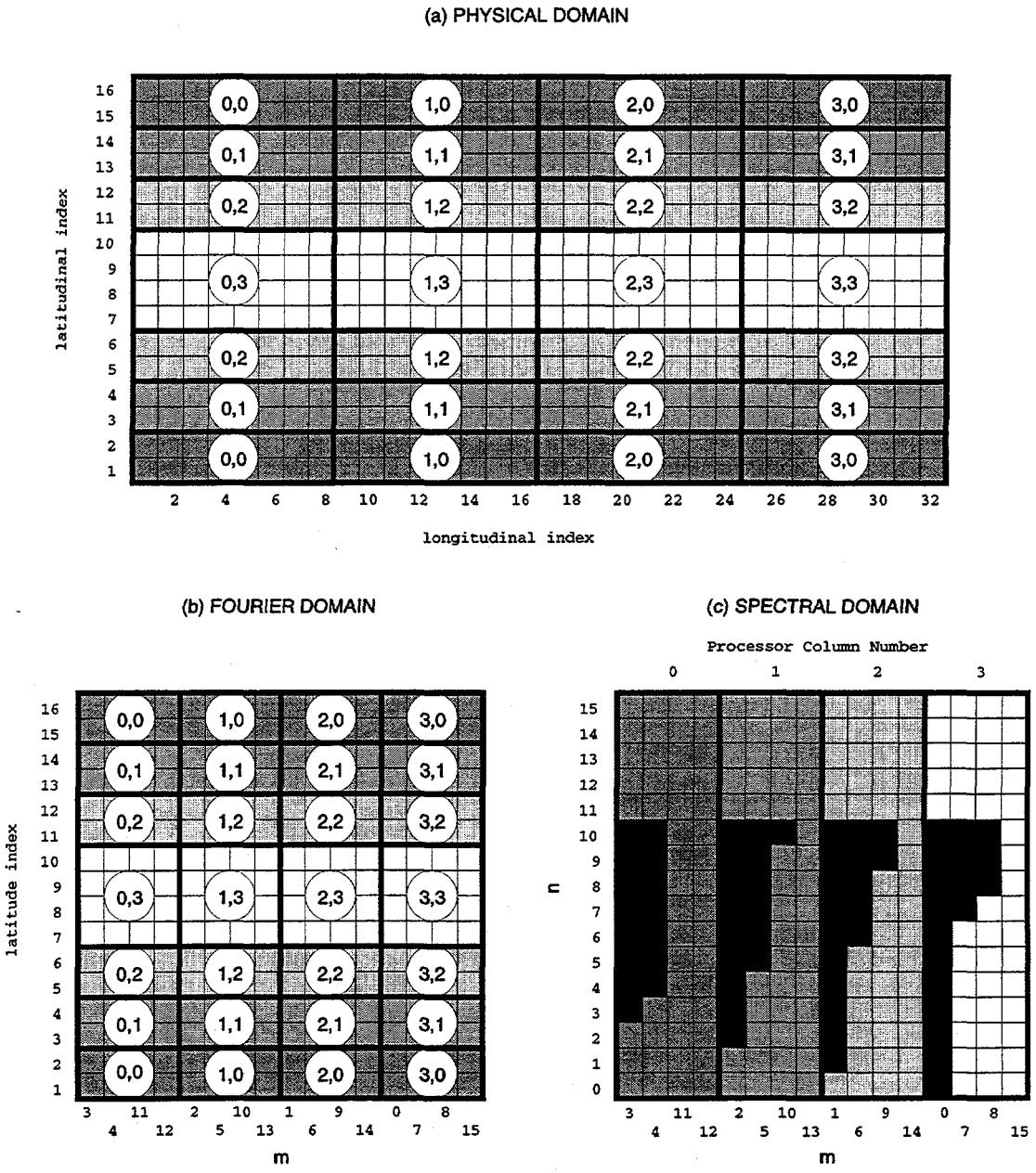

Figure 1: The decomposition of the (a) physical, (b) spectral, and (c) Fourier domains over a 4×4 grid of processors. For figures (a) and (b), each small cell represents a data item. The thicker lines show the boundaries between processors. The circles contain the processor coordinates. The shaded cells in figure (c) represent the spectral coefficients to be included in the spectral transform, and shows how these are decomposed over processor columns

respectively. For the forward Legendre transform, each ξ_n^m depends only on data associated with the same wavenumber m, and so depends only on data assigned to a single processor column. Each processor in that column can calculate independently its contribution to ξ_n^m, using data associated with the latitudes assigned to that processor. To finish the calculation, these P contributions need to be summed, and the result needs to be rebroadcast to all P processors, since spectral coefficients are duplicated within the processor column. To minimize communication costs, local contributions to all spectral coefficients can be calculated first, leaving a P-way vector sum (made up of the local contributions to all of the spectral coefficients assigned to this processor column) and rebroadcast to be calculated. This motivates naming this approach the *vector sum* algorithm.

The column-wise vector sum is a separate step in the algorithm, and the communication is not overlapped with computation. But there are sophisticated techniques for calculating the vector sum that effectively minimize both the communication cost and the associated parallel computation cost. Currently we use a variant of the recursive halving algorithm [35].

For the inverse transform, calculation of $\xi^m(\mu_j)$ requires only spectral coefficients associated with wavenumber m, all of which are local to every processor in the corresponding processor column. Thus, no interprocessor communication is required in the inverse transform.

In summary, using the vector sum algorithm to compute the Legendre transforms incurs no additional computational cost, is perfectly parallel with good load balance within a processor column, and requires interprocessor communication in only the forward transform. Moreover, this communication can be implemented very efficiently. Furthermore, few modifications to CCM2 were required to implement this algorithm in PCCM2.

The disadvantages of the vector sum algorithm are that all computations *within* the spectral domain must be calculated redundantly (in the processor column), the communication in the forward Legendre transform can not be overlapped with communication, and additional storage is required to hold the duplicated spectral coefficients. Since relatively little work is done in the spectral domain in CCM2, this redundant work has not proved to be an issue, and the vector sum has proved to be a viable parallel algorithm for PCCM2. For a more detailed discussion of these issues see [14, 46, 10, 15, 21].

3.4. Semi-Lagrangian Transport

The advection of moisture in CCM2 uses a semi-Lagrangian transport (SLT) method in conjunction with shape preserving interpolation [41]. The method updates the value of the moisture field at a grid point (the arrival point, A) by first establishing a trajectory through which the particle arriving at A has moved during the current timestep ($2\Delta t$). This trajectory is found iteratively using the interpolated velocity field at the mid-point, M, of the trajectory. From this mid-point the departure point, D, is calculated and the moisture field is interpolated at D using shape preserving interpolation. All the calculations involve physical space (grid point) data and are decomposed over the processors with the same mesh decomposition described above.

The modifications made for the parallel implementation involved a redefinition of the extended grid arrays already implemented for the SLT. Extended grids are necessary since cubic interpolation requires two additional points outside the region being interpolated. Extending the grids even further leads to regions of overlap among the processors, but it can be guaranteed that with enough extension the departure point and subsequent interpolation of the moisture field will use only the data on the extended grid, and thus local to the processor. The amount of the extension is controlled by separate parameters for the latitudinal and longitudinal directions.

The overlap regions on each processor must be updated each timestep. Communication is blocked in such a way to allow the possibility of overlap with more than one processor. This

can occur, for example, when a large number of processors are used and each processor has only two latitudes. The setting of the extended grid at the poles also requires communication between processors. In particular, the pole point, which occupies an entire latitude line in the extended grid, is assigned a value based on the zonal average of nearby latitude lines. A sum across the pole processors is required for this to be calculated. Since the pole processors lie on the first row of the processor mesh, a separate procedure is used for these processors.

4. PARALLEL MODEL VALIDATION STUDIES

Validation of the CCM2 implementation has been done on several levels. Code internals and algorithm equivalency must be checked for any code port. But as a climate model, special steps are required for CCM2. The model has already been validated by NCAR as a viable representation of the earth's atmosphere and as a climate model yielding present day earth climate statistics when forced by present climatological boundary conditions. The purpose of the validation studies here, is not to return to validation against observational data but to verify that the ported model yields the same climate statistics as any other implementation. Specifically, output is compared with a set of Cray YMP runs. Since the underlying weather phenomena modeled by a climate model is sensitive to initial conditions, so is the CCM2. Due to differences in machine arithmetic, the particular path taken by the computation will not be the same on two different machines. But the speed of separation of the paths follows known rates and the time averaged statistics should be the same within bounds of climate variability. So the comparison with another machine implementation is not arbitrary. It is recommended that any port of the PCCM2 to another platform perform the same validation exercises.

As a first order check on accuracy and the parallel implementation a check is performed of the transformation of initial conditions. Input data are read and transformed to spectral space in the initialization phase of the code. After spectral truncation and transformation back to physical space the data should match very closely (machine precision) with the CRAY result. This is the first check and validates the parallel spectral transform. The second check verifies that the growth of error between the parallel (or ported sequential) results and the NCAR CRAY results is within expected bounds. A third validation compares the monthly averages for all prognostic fields at the end of a three month perpetual January simulation. Finally, climate statistics are examined for seasonal averages of multi-year runs.

The standard working precision of the code is 64-bit, although a single precision option is available at compile time. The single precision (32bit) calculation was found to be inaccurate unless the Gauss points and weights used in the spectral method were calculated in double precision. So these parts of the calculation are always performed in 64 bit precision.

4.1. Error Growth

If the code is run on a different computer with even slight differences in machine arithmetic, the resulting output from the model will be different. The model is sensitive to initial conditions, and the particular path or trajectory taken by the model state represents the natural variability of the climate system. However, the rate of departure of the model state when started from slightly perturbed initial conditions is well known. A study by Rosinski and Williamson [29] indicates the expected departure in the temperature field, for example. A key component of the validation of the model implementation is the comparison of model output with CRAY model output. The CRAY model has been extensively compared with observational data and the characteristics (and shortcomings) of the model climate are documented in [20].

The following graphs give a comparison of diagnostic output from a CRAY YMP 1 day run and parallel runs on an 8x8 mesh on the Intel Paragon and an IBM SP2. The same T42 simulation was performed on each machine. The time step size is 20 minutes for a total of

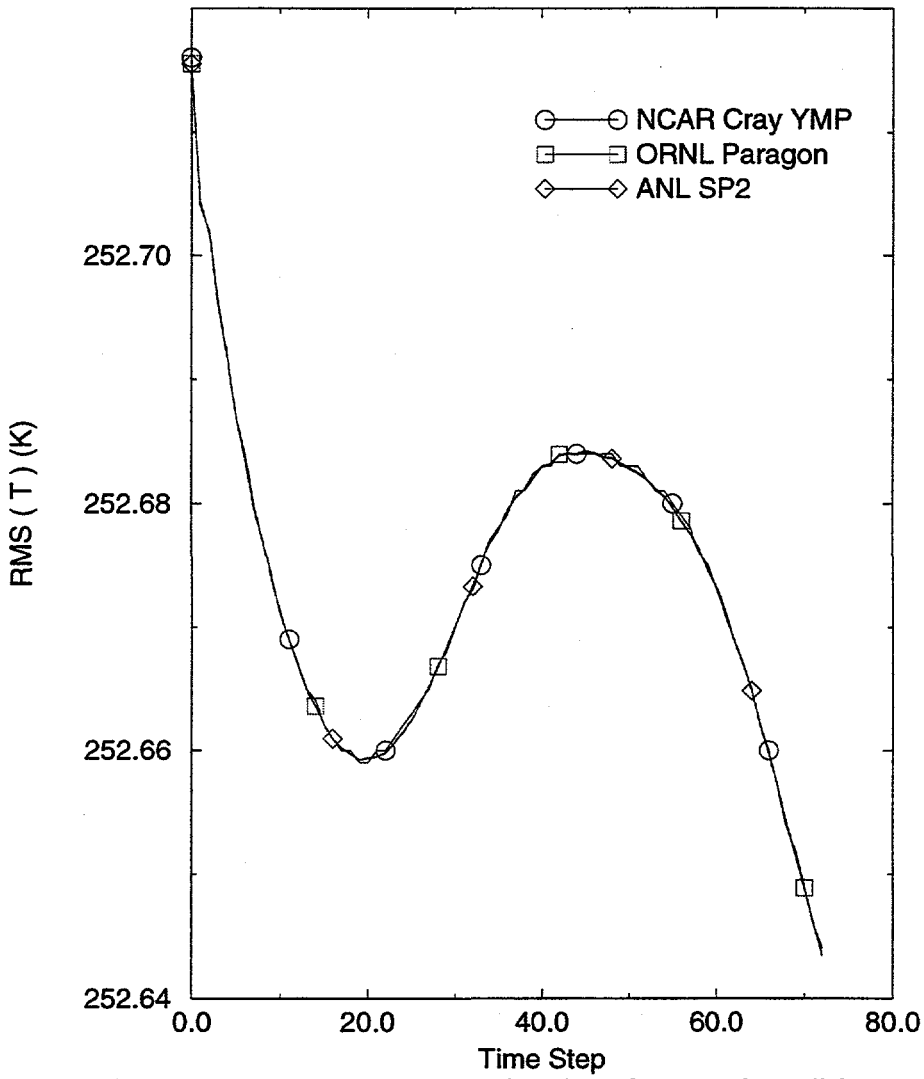

Figure 2: Comparison of RMST values from CRAY and parallel codes.

72 timesteps per day. Figure 2 shows the root mean square of the temperature for the CRAY version of CCM2 and PCCM2 on the Paragon and SP2. The curves are indistinguishable graphically indicating several digits of agreement. Figure 3 shows the difference between the results for the parallel versions on the Paragon and the SP2.

A closer look at the other summary output from the model indicates the level of agreement of the implementations on different platforms. All these indicate very close agreement and are a substantial component of the validation of the implementations. Figure 4 shows the difference of the root mean square of the vorticity between the CRAY results and either the Paragon or SP2 results. Figure 5 shows the differences in the root mean square divergence and Figure 6 shows differences for the global moisture integral.

A comparison of history tapes at the end of a 1 day run was performed using the CCM post processor to analyze all the fields. The same level of agreement was found.

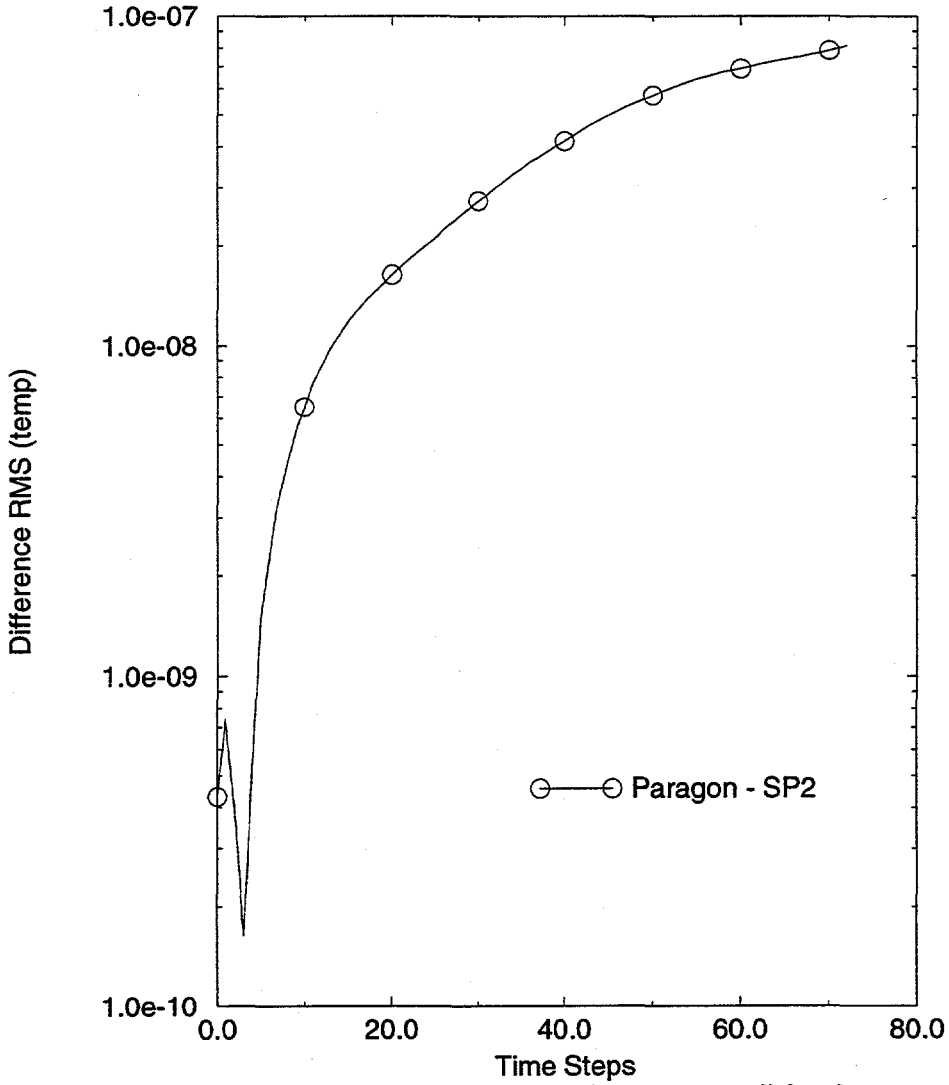

Figure 3: Difference of RMST values from parallel codes.

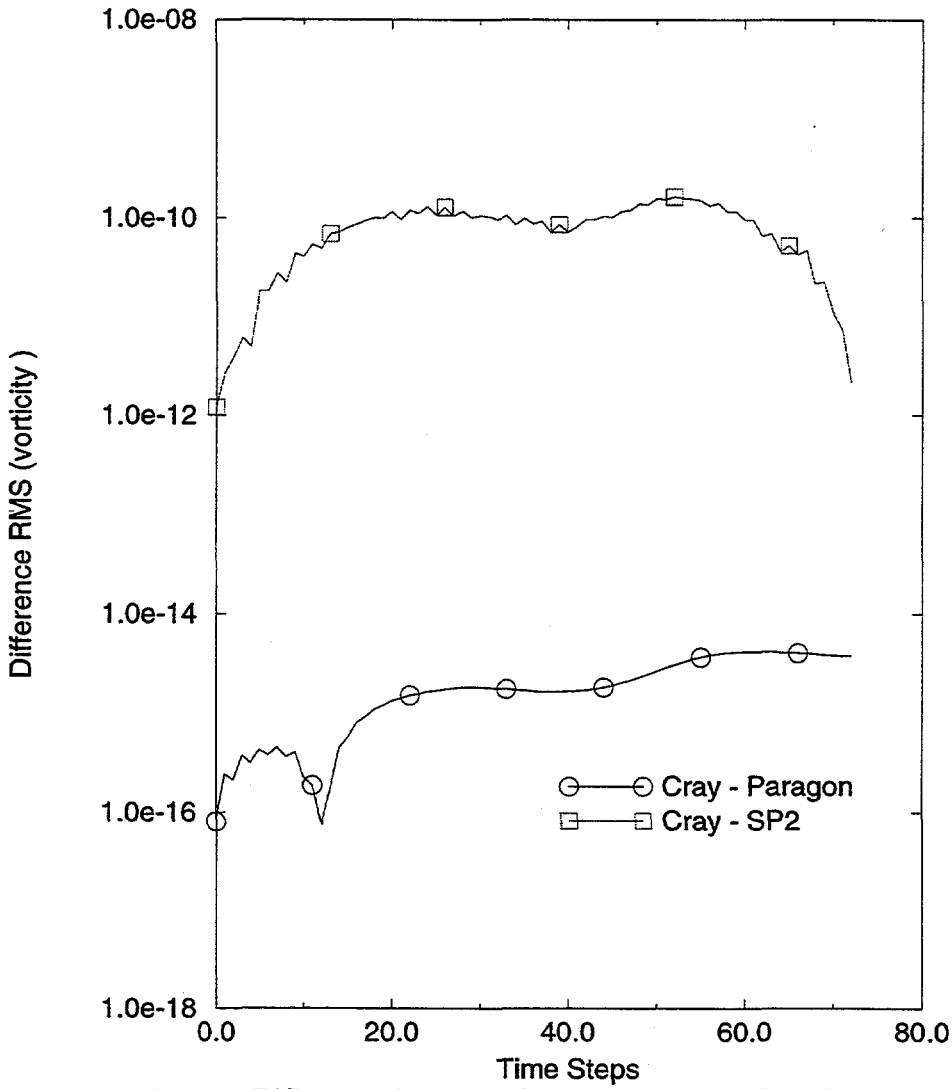

Figure 4: Difference of RMSZ values from CRAY and parallel codes.

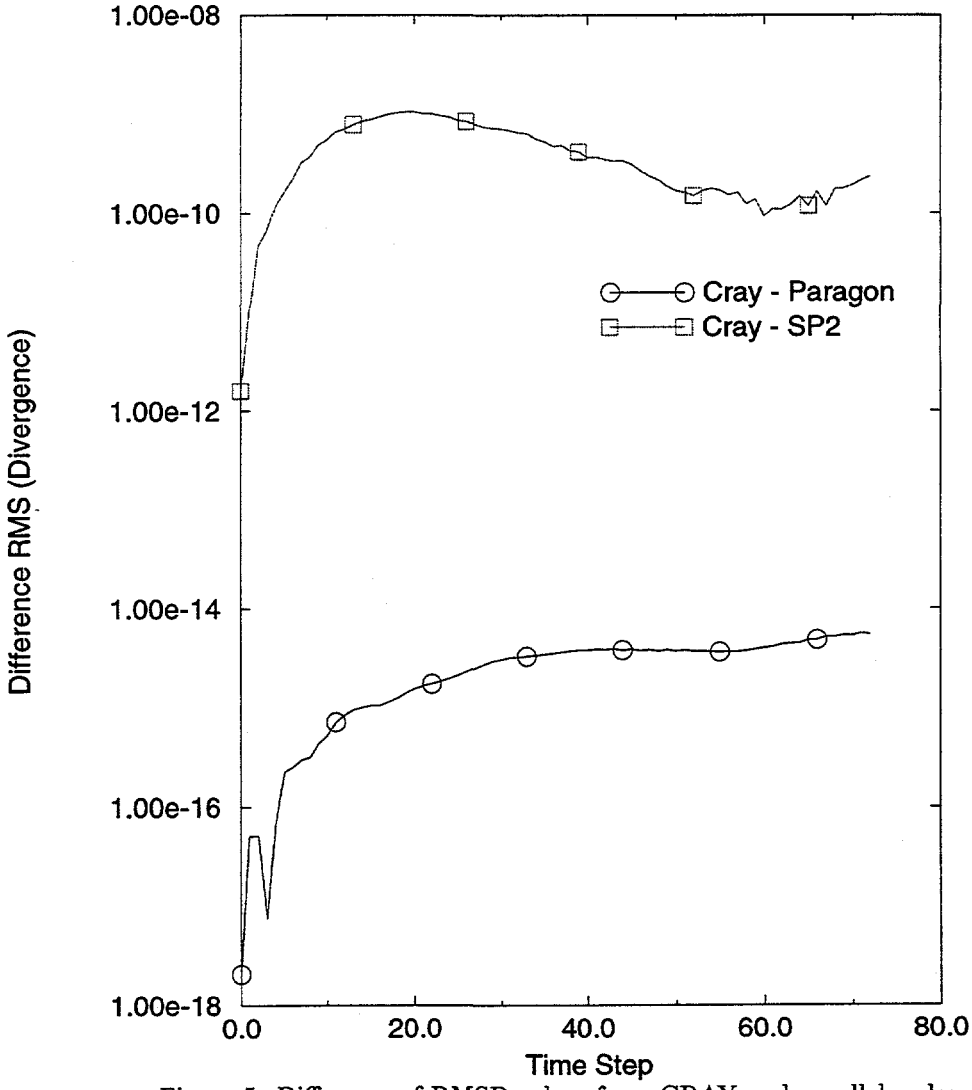

Figure 5: Difference of RMSD values from CRAY and parallel codes.

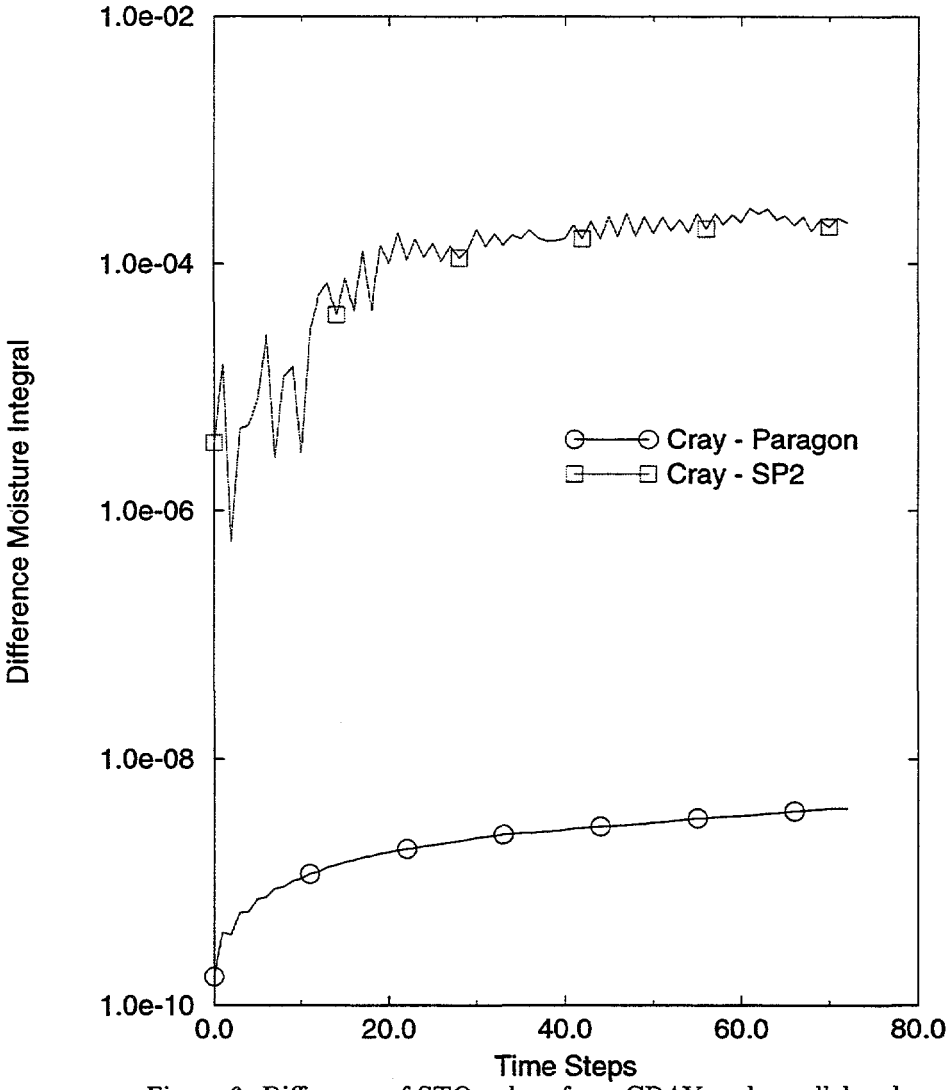

Figure 6: Difference of STQ values from CRAY and parallel codes.

4.2. Reproducibility

Changing the number of processors for the simulation is like changing the machine on which the code is run. Consistency of the numerical results is therefore an issue for production runs where the number of processors may change over the course of the simulation. This issue is referred to as reproducibility and can be paraphrased by asking if the results from one machine configuration exactly (bit for bit) reproduce the results from another machine configuration on the same machine.

Non-reproducibility can arise in otherwise "correct" parallel implementations due to the nonassociativity of floating point addition. In the parallel spectral algorithm the order of the sum in the Legendre transform is different depending on the number of processors. The computation of global sums for diagnostics is also sensitive to the order in which the sum is taken. But it is possible to impose order by carefully structuring the parallel algorithms. This reordering results in little loss of performance.

The PCCM2.1 is fully reproducible for power of two horizontal resolutions: T21, T42, T85, T170, etc. If on changing numbers of processors on a given machine, the results differ by any amount, then there is an implementation error (bug) or hardware problem. This feature has helped tremendously in trouble shooting problems with the implementation on new architectures.

5. PARALLEL CCM2 USER'S GUIDE

This User's Guide is an extension to the NCAR CCM2 User's Guide [3] for execution in a distributed memory parallel environment. Many of the procedures described are installation and machine specific to the Intel Paragon at Oak Ridge National Laboratory or the IBM SP2 at Argonne National Laboratory. The implementations of the PCCM2 have left as much as possible of the original NCAR coding and procedures in place so that anyone familiar with the use of the CCM2 code can make the transition to use of the parallel code with minimal effort. The "physics" routines are essentially untouched allowing easy modification of PCCM2 by users in pursuit of their scientific research with little regard to parallelism.

Following the NCAR CCM2 User's Guide [3] we first present the details necessary for running the code on various parallel platforms. Next we describe PCCM2 internals and material that might be useful for making modifications to the code and setting it up on a particular system. The first subsection presents execution scripts for use on the Intel Paragon running OSF/1 Version 1.3 or later. Environment variables that must be set for the execution are described and pertinent libraries are named for the standard control simulation.

Section 5.2 describes the setup of the history tape output in the parallel environment. To efficiently execute in a parallel environment the large volume of code output must be processed in parallel, striping the output across several RAID's and using multiple I/O nodes. High bandwidth is achieved in this way. Accessing the history tape outputs and converting them to a format readable by other machines is accomplished using the filters described in section 5.3.5.

The final section discusses post processing tools for the PCCM2 model output.

5.1. Running the Model

The following *namelist* file can be used for an initial run.

```
&CCMEXP
CTITLE = 'T42 Control Run',
NCDATA = 'ICdat',
BNDTI = 'tibds',
BNDTVS = 'tvbds',
```

```
 BNDTVO = 'ozn',
 IRT=0,
 NSREST=0,
 NSWRPS='passwd',
 NSVSN='rstrt',
 NDENS  =     1,
 NNBDAT = 000901,
 NNBSEC = 0,
 NNDBAS = 0,
 NNSBAS = 0,
 MFILT  =    10,
 DTIME  =  1200.,
 NESTEP =   14400,
 NHTFRQ =   18,
 IRAD   =   -1,
 IRADAE =   -12,
 SSTCYC = .T.,
 OZNCYC = .T.,
 DIF4=1.E16,
 PARHIST = .F.,
 ACCRST = .T.,
 &END
```

The meaning of the input parameters is documented clearly in the User's Guide to the NCAR CCM2 and so is not discussed here. The exception is PARHIST and ACCRST which are described in section 5.2.2.

A **restart run** starting with dataset number r0024, is given in the following *namelist.*

```
&CCMEXP
 CTITLE = 'PCCM2 Control Run B',
 NCDATA = 'ICdat',
 BNDTI = 'tibds',
 BNDTVS = 'tvbds',
 BNDTVO = 'ozn',
 IRT=0,
 NSREST=1,
 NSWRPS='passwd',
 NSVSN='rstrt',
 NREVSN='r0024*',
 NDENS  =     1,
 NNBDAT = 000901,
 NNBSEC = 0,
 NNDBAS = 0,
 NNSBAS = 0,
 MFILT  =    10,
 DTIME  =  1200.,
 NESTEP =   262800,
 NHTFRQ =   72,
 IRAD   =   -1,
 IRADAE =   -12,
 SSTCYC = .T.,
 OZNCYC = .T.,
```

```
DIF4=1.E16,
PARHIST = .F.,
ACCRST = .T.,
&END
```

5.1.1. Intel Paragon

The Paragon has two logical classes of processors, compute nodes and service nodes. When you log in to the Paragon, a service node responds to UNIX commands. Compilation and linking are also handled by the service nodes. To run a parallel program a partition of compute nodes must be assigned to the user. This is done using the *mkpart, cmkpart, pexec* commands. Partitions are removed using *rmpart, crmpart* commands. Loading the program on the compute nodes is done under OSF/1 by executing the code with a partition name. For example, *pccm2 -pn chammp001.* The *pexec* command both creates a partition and loads a code. The form of the command is *pexec -sz 64 pccm2.* (See the *man* entry for *application* on the Paragon systems for further information.)

An environment variable is set by the user to identify the directory in which the code is to find input files and in which to create the output files. The input datasets can also be specified by giving the full path names in the *namelist* input. In the example below, the execution path environment variable is set to the users PFS directory. The following commands run the pccm2 code on 256 processors (16x16 mesh) in a parallel file system directory named chammp/t42.

```
%setenv CCM_EXEC_PATH /pfs/chammp/t42
%pexec -sz 256 pccm2 &
```

In the PFS directory the code expects to find the initial condition and boundary input datasets as well as the *namelist* input in a file named *fort.050.* The standard error and output files from node zero of the parallel execution will be placed in files named *pccm.out.0000* and *pccm.error.0000.*

The PCCM2.1 code on the Intel Paragon can be run using the native NX message passing libraries or, alternatively, using MPI, PVM or PICL libraries when OSF/1 is the operating system. An implementation for the SUNMOS operating system is also available but is limited by lack of parallel I/O.

5.1.2. IBM SP2

PCCM2.1 may be run on the IBM SP2 with the native IBM message-passing passing library (MPL) or with IBM's implementation of the standard Message Passing Interface (MPI) software. Although the message-passing performance of MPL and MPI is virtually identical, you may prefer one message-passing layer over the other, depending on the libraries available at your installation or your wish to use additional facilities (such as MPE for performance data gathering under MPI).

Whether your program uses MPL or MPI depends on how it was linked at compile time. MPL programs are linked by using **mpxlf**, which automatically includes the necessary libraries. MPI programs are linked by using **mpixlf**. Since IBM's MPI runs on top of MPL, the details for running the code are identical in either case.

The following is a sample invocation for running the model on four processors. It shows operational parameters being set using environment variables, some of which are accessed by the IBM Parallel Operating Environment (POE) software. See **man poe** for additional information.

```
% setenv MP_HOSTFILE <file>
% setenv MP_PROCS <n>
% setenv MP_EUILIB us
```

```
% setenv XLFRTEOPTS namelist=old
% setenv CCM_EXEC_PATH .
% pccm2
```

These commands are executed on the control workstation for the IBM SP2 or on one of the computational nodes. On installations with batch processing, the commands may be incorporated within the batch script. The nodes that are used to run the program are determined by MP_HOSTFILE.

MP_HOSTFILE. This is set to the name of a file that contains a list of the nodes on which the program is to run, one node per line. On some systems, such as the IBM SP at Argonne, the file is created automatically by the scheduling program when the partition is allocated. On the Argonne machine, use the following command to set the variable to the correct value:

```
% setenv MP_HOSTFILE $HOME/SPnodes.'getjid'
```

The character delimiting the string "getjid" is a backquote character (‘). Other installations may provide similar mechanisms for determining the setting of the MP_HOSTFILE variable. Consult your local system support staff. Additional information on the IBM SP at Argonne is available on-line:

```
http://www.mcs.anl.gov/Projects/sp/
```

MP_PROCS. This is set to the number of processors on which to run the model. There must be at least this many node name entries in the file specified by MP_HOSTFILE.

MP_EUILIB. This informs POE which low-level communication mechanism to use: "us" specifies the high-performance switch feature if it is available on your installation. If "us" is not available, set this variable to "ip." This is, however, considerably slower.

XLFRTEOPTS. This is not a POE environment variable but it is required when using PCCM2.1 compiled with most current installations of IBM XLF Fortran. Setting the environment variable XLFRTEOPTS to the string "namelist=old" causes the XLF Fortran run-time system to interpret *namelist* input properly by using pre-Fortran90 syntax. The default is to use Fortran90 *namelist* syntax.

CCM_EXEC_PATH. In the directory specified by CCM_EXEC_PATH, the model expects to find the *namelist* input in a file named *fort.50* as well as the files containing initial condition and boundary input data sets; the names of these are specified in the *namelist* file. Output data sets will be written to this directory as well. The default is the current working directory (‘.’).

5.1.3. Network of Workstations with PVM

To run the code on a network of workstations using the Parallel Virtual Machine (PVM) software, the executable *pccm2* must be located in the users directory *pvm3/bin/$ARCH,* where $ARCH specifies the particular machine, eg. RS6K. Since passing environment variables through PVM is a bit tricky, the user should either specify the location of the binary files directly in the *namelist* file or provide links to the appropriate files in the users home directory. Output will be written to the users home directory.

To start a PVM session, it is assumed that PVM has been installed on a cluster of workstations (or the MPP) and that appropriate links and modifications to path statements have been

made. The user starts the PVM console by typing *pvm* in some window. This responds with a prompt and allows the user to add machines, i.e. other workstations, defining the virtual machine. In another window, the user executes the *pccm2* code which will spawn the *pccm2* on the other processors of the virtual machine. Logically, the program is still defined on a mesh of processors though, in fact, the virtual machine may consist of a heterogeneous collection of processors linked with a local area network.

Further information on PVM may be found by browsing

`http://www.epm.ornl.gov/pvm/` .

PCCM2.1 may also be used under MPI on networks of IBM or other vendor's workstations. Consult

`http://www.mcs.anl.gov/mpi/`

for information on obtaining and installing MPICH, the publicly available version of MPI, and for running parallel programs under this version.

5.2. Output from Parallel Model

Three types of output are to be expected from a production run of the PCCM2.1: model history output (binary), restart data (binary) and printed diagnostic information. There are a number of differences between the standard CCM2 output files and what can be expected in a MPP environment. These differences are largely due to the lack of standards (*defacto* or otherwise) in parallel output operations. Each vendor provides a machine specific solution to the problem of efficient output and in the hope of providing a production quality implementation we have endeavored to take advantage of these solutions.

5.2.1. Printed Output

The printed output of the code is found in the file *pccm.out.0000*. The suffix indicates that the file is written from processor (0,0) of the logical mesh. It consists of version information, the *namelist* input summary which controls the simulation, a listing for each step the program executed giving

RMSZ global RMS vorticity,

RMSD global RMS divergence,

RMST global RMS temperature,

STPS global mass integral,

STQ global moisture integral,

COUR maximum Courant number for horizontal velocity field.

From these values one can determine the progress of the run and whether something is amiss.

Also included in the printed output is information about the contents of history tapes written at each history tape output. Finally, the "END OF MODEL RUN" message is printed with summary information.

The other print file generated by the PCCM2 code is *pccm.error.0000*. Error messages to unit 0, the standard error unit, are included in this file. Also timing information for each time step is included if the INTERVAL_TIMER has been enabled.

Both the *pccm.out.0000* and *pccm.error.0000* files are output from processor (0,0) of the logical processor mesh. Output from other processors is sent to */dev/null* and is therefore

unavailable. It has been found that opening output files on every processor will crash or hang many MPP systems. For debugging purposes, this option is still included as a compile time option, but is not supported in the standard configuration.

5.2.2. History Output

> *Note:* The CCM2 Users Documentation refers to "history volumes" (or sometimes "history tapes") and "history files." A CCM "history volume" ("history tape") refers to a file created by the model as it runs; such a file consists of output from one or more history periods (in other words, it is a file in ordinary computing terminology). A CCM "history file" is the data in a volume for one period; it is thus a logical subset of the data in a volume. The following discussion uses the CCM2 terminology unless specifically indicated.

PCCM2 history output volumes are *identical* in structure to those written by the NCAR CCM2. The model history volumes may be written by the parallel model in one of two modes, either *volume-complete* (a single file containing data from all processors) or *volume-decomposed* (multiple files containing spatially decomposed partial data that, when combined, contain the complete set of data for the volume).

Volume-complete mode exactly duplicates CCM2 history output format and requires no additional postprocessing to recompose history output. One processor (or a subset of processors) collects data from the other processors and then writes the history output by using the standard Fortran write mechanism for unformatted (binary) data. The parallel I/O implementation on the Intel Paragon uses the parallel file system (PFS) facilities. The write, though actually in parallel from a number of processors, is to a single binary file. The PFS software manages the file pointers in an appropriate fashion. History volumes are named h*dddd*, in accordance with the CCM2 User's Guide, where *dddd* is the number of the history volume. In volume-complete mode, the PCCM2 *namelist* settings to specify history frequency and the number of history writes per volume are the same as CCM2.

Volume-decomposed mode is specified by setting the *namelist* variable PARHIST to `.TRUE.`. In this mode, each processor row writes its set of local latitude records in a separate partial history volume. Later, the partial volumes may be combined to generate a complete history volume, or they may be left in partial history volume form (if postprocessing software can handle this). This mode provides a simple method of exploiting parallel I/O on systems where processors can write to different devices. For example, on the IBM SP2, processors may write to their local disks or to separate I/O nodes if available. In volume-decomposed mode, data for each partial history volume is collected and written by the "western-most" processor in each row of the mesh. In other words, the model generates PP_NPROC_LT partial volumes, where PP_NPROC_LT is the number of processors decomposing the latitudinal dimension.

The name of a partial history volume is appended with a four-digit processor row-identifier. In volume-decomposed mode, the volume with row identifier "0000" contains the CCM history header and records containing data from the latitudes processed on the first row of processors; the other partial volumes contain no header but only latitude records. In volume-decomposed mode, only one history write is made per volume (equivalent to `MFILT = 1` in the *namelist*). This allows recombination of the output by simple concatenation:

```
cat h0001.* > h0001
```

When you are using volume-decomposed mode, the *namelist* variable NREFRQ should be set higher than the default of 1. Otherwise, the model will generate restart data each time it writes a history file to a volume. A suitable value for NREFRQ when PARHIST is set to `.TRUE.` is 5. That is, the model will generate restart data after every five history writes. By

contrast, in volume-complete mode (`PARHIST = .false.`), the default number of history writes per file is five (i.e., `MFILT = 5`), so it is not necessary to modify NREFRQ.

5.2.3. Post Processing History Tape Output

History tape output produced by NCAR's CCM2 and the PCCM2 codes have the same format. Files are stored and transferred between machines as binary files due to their large size and written in machine dependent binary formats. A code to translate these binary files between different formats has been written for each architecture which will access these files. Filters which do this binary translation using a FORTRAN-C interface can be compiled on any machine with the RPC- library (RFC1050, "Remote Procedure Calls: Protocol Specification", in Network Programming Guide, part no. 800-3850-10, Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, CA 94043.) These filters are based on the External Data Representation (XDR) standard (RFC1014, "External Data Representation:Protocol Specification", in Network Programming Guide, part no 800-3850-10, Sun Microsystems, Inc., 2550 Garcia Avenue, Mountain View, CA 94043.) Using these filters, the binary file passed between machines is in XDR format and easily decoded to machine dependent binary format on the desired architecture.

XDR Conversion Filters

Included with the FORTRAN-C source code is a pair of makefiles, "makefile" and 'make.mach_ind', which have been designed to allow the user to create the executables by simply specifying the architecture (e.g. 'make cray'.)

encode: takes as input a machine dependent binary history file and produces an XDR-formatted machine independent binary history file.

decode: takes as input an XDR-formatted machine independent binary history file and produces a machine dependent binary history file.

For example, to convert a history file, *h0001*, to the xdr format, use

```
%encode h0001
```

This will produce the file *h0001.xdr*. To produce a machine binary from an XDR file, eg. *h0001.xdr*, use

```
%decode h0001
```

5.3. Restarts

PCCM2, like CCM2, can periodically save restart data. The restart data can be used to restart the model and resume execution. This feature is useful both for safeguarding against crashes and for extending beyond what was previously simulated. The NESTEP parameter in the *namelist* file can be increased and the simulation continued beyond what had been originally set as the last time step.

Restart data consists of a save of much of the model's state data, considerably more than just history (output) data. This is because intermediate values such as tendencies must be restored so that the model can continue on a restart as if the run had not been interrupted. If the restart does not occur at the same frequency as the calculation of radiation absorptivities and emissivities (RADABS), the data set is considerably larger because this information must be saved in the restart files as well.

5.3.1. Generating Restart Data

Restart data is generated automatically as the simulation progresses, at an interval determined by the setting of *namelist* variables.

- NHTFRQ — This specifies the number of simulation hours (or time steps) between each history write.
- MFILT — This specifies the number of history writes that make up a history volume.
- NREFRQ — This specifies the number of completed history volumes between each restart write.
- NINAVG — This allows the user to specify monthly averaging of history output, which also changes restart behavior. Restarts with monthly averaging are described separately in Section 5.3.3.

The number of restarts generated for a given length of simulation is determined by the number of complete history volumes written, which is in turn determined by the number of history writes per history volume, which is in turn determined by the number of history writes per hour of simulation. The default settings for these variables are as follows.

```
NHTFRQ = -24,
MFILT = 5,
NREFRQ = 1,
NINAVG = 'A',
```

These default settings result in the output of restart data every five model days. History is output every 24 model hours (NHTFRQ = -24), a history "volume" is finished every 5 outputs (MFILT = 5), and restart data is generated after every 1 completed volume (NREFRQ = 1). If, instead, one wishes to keep the same restart generation frequency but have only one history write per volume, the settings are as follows.

```
NHTFRQ = -24,
MFILT = 1,
NREFRQ = 5,
NINAVG = 'A',
```

Restart files. Each time the model generates a restart data, it generates a so-called Master Regeneration Data Set: a file whose name is r*dddd*, where *dddd* is the number of the restart (starting at 1 for an initial run). The Master Regeneration Data Set contains model state data that is duplicated (as opposed to decomposed) over processors in the parallel model. It is written by processor 0 (zero). In addition to the Master Regeneration Data Set, the model writes a Primary Restart Data Set, which is contained in one file (on a Paragon) or multiple files, one per processor (other platforms). The name of the Primary Restart Data file in the single file case is r*dddd*.A. In the case of multiple restart files, the name is r*dddd*.A.*pppp* where *pppp* is the processor identifier. The multifile Primary Restart Data may be recombined into a single file by using simple concatenation.

```
%cat r0001.A.* > r0001.A
```

It is necessary to combine the files into a single Primary Restart Data file only if the model is to be restarted on a different number of processors.

Alternating restart files. By default, PCCM2 generates restart data sets in sequence: r0001, r0002, r0003, and so forth as the model runs. However, if only the most recent restarts are needed, PCCM2 can generate them in alternating mode to save space. Setting the *namelist* variable ACCRST to `.FALSE.` causes the model to write and overwrite only two restart data sets, switching back and forth between them.

Namelist	Sequence of File names Generated
`ACCRST=.TRUE.`	r0001, r0002, r0003,...
`ACCRST=.FALSE.`	r0000, r0001, r0000,...

History regeneration data sets. Unlike CCM2, PCCM2 acquires all information for the restarted run from the *namelist* and from the restart data sets; it does not read previously generated history files, nor does it write or read history regeneration restart files. History regeneration runs and restarts within averaged periods are not supported.

5.3.2. Restarting the Model

PCCM2 supports restart and branch runs. History regeneration runs are not supported. The difference between a restart and a branch is that a restart will number subsequent history and restart files as if the run had simply continued from the point of the restart. A branch, on the other hand, starts numbering of the history and restart files at the beginning. Because all model settings come from the restart files and from the *namelist* files (not from previously written history files or regeneration files), there is no other substantive difference between a restart run and a branch run with the parallel model.

A restart run is specified by setting the *namelist* variable NSREST to 1 (one). A branch run specified with a value of 3 (three). (For an initial run, the setting of NSREST is 0 (zero)). If a restart or branch is specified, the model opens the Master and Primary restart data sets whose base name is specified by the string-valued *namelist* variable NREVSN. For example, to restart from the files r0009 and r0009.A, specify

```
NSREST = 1,
NREVSN = 'r0009',
```

The model can be restarted directly from the per-processor restart files (r0009.A.0000, r0009.A.0001, and so on) provided the restart run is on the same number of processors as the original run. Adding '*' to the value of NREVSN

```
NREVSN = 'r0009*',
```

indicates that you are restarting from the file r0009. The Intel PFS restart file is also specified with a * even though it is a single file. This is because it is written from all nodes simultaneously. When restarting on a number of processors different from the original run, you must combine the Primary Restart Data in a single file (see above) and leave the '*' off the NREVSN string. If NSREST is 0 (zero) for an initial run, the setting of NREVSN is ignored.

Restart files generated by CCM2 cannot be used to restart PCCM2 or vice versa. The structure of the records in PCCM2 is vertical column oriented, with all fields associated with a column of grid points grouped together on output. Since restarting on a different number of processors will involve a different allocation of columns to processors this allows efficient, parallel input of the checkpoint/restart data.

5.3.3. Restarts and Monthly Averaging

PCCM2 supports monthly averaging on the primary history tape. This is specified in the *namelist* by setting the value of NINAVG to the string value 'Q' for the primary history tape:

```
NINAVG = 'Q',
```

In monthly average mode, PCCM2 writes the primary history data set only once each month. Restart data sets are written at this time as well. Naming of the monthly averaged history volumes and restart data sets differs from normal history output mode. Each volume is named as described in the following on-line documentation in the NCAR release of CCM2.1.

> The file naming convention of the monthly average history tape will be of the form mm-yy (mm=month, yy=year). All monthly average restart files will also contain the month and year as part of their names. For example, a monthly average history tape for the month of December during model year 0 will have the file name 12-00. The primary regeneration file that goes with the December average will be r12-00.A.

In addition, if `PARHIST = .TRUE.` in the *namelist*, the monthly restart history volumes will be written in volume-decomposed mode: processor row numbers will be appended to the file names. Midmonthly restarts are not supported in the parallel model; monthly average restart files are written at the same time monthly average history is written. As with restart files in normal history mode, the monthly average restart files are written one file per processor.

5.4. PCCM2 Internals

This section describes building the code: the compilation and specification of compile time options for the parallel code. Described are how to specify the resolution, the number of processors and dimensions of the logical processor mesh for running the code, the execution and algorithmic options specifying different parallel algorithms in the code.

5.4.1. Building the Model

The basic model parameters controlling the setup of the code (not the physical parameters of the model) are specified in the file *params.h*. This file is included in the source of every routine and must be modified before compilation for a given machine. Current machine options for PCCM2 are INTEL, RS6000, SUN and CRAY.

The source code in .F files must be run through a preprocessing step with */lib/cpp* to pull out those sections of code appropriate to the build configuration. The preprocessing step produces .f files which are then compiled and linked to produce the executable. A makefile is provided which contains particular machine compilation options. It is necessary to edit the makefile only once for each computing platform. The location of message passing libraries and optimization options are set in the makefile.

PP_PCCM: The parallel model is obtained by defining PP_PCCM. All modifications to the sequential code for parallelism have been surrounded by #ifdef PP_PCCM ...#endif .

PP_TRIANG_TRUNC: A triangular spectral truncation specifies the horizontal resolution of the model. The default set in *params.h* is T42, but a variety of other resolutions are also supported. From the triangular truncation the number of grid points in the longitudinal direction, PP_GLON, and the number of grid points in the latitudinal direction, PP_GLAT, are defined. The number of levels, PP_PLEV, is always set at 18 to match the physics parameterizations.

PP_NPROC_LT: The number of processors to be used in the latitude direction. The number of processors must evenly divide the number of latitudes, PP_GLAT. We also require that at

least 2 latitudes per processor be configured. This allows an efficient Legendre transform taking advantage of the symmetry of the spherical harmonics.

PP_NPROC_LN: This must be a power of two and for the distributed FFT algorithms, the PP_GLAT/PP_NPROC_LT must be greater than or equal to four.

PP_NPROC: The total number of processors used in the execution of the model will be PP_NPROC_LT*PP_NPROC_LN.

INTEL: Machine target is an Intel RX, DELTA or PARAGON.

RS6000: Machine target is an IBM SP2, or cluster of RS6000 workstations.

SUN: Machine target is cluster of SUN workstations.

CRAY: Machine target is a cluster of CRAY's or a T3D.

PP_RADLB: To load balance the short-wave radiation calculation.

PP_FFTX: To use the transpose algorithm for parallel computation of FFT's.

BLOCK_FFT: Block the computation of FFT's for efficiency. Adds storage but generally improves execution time significantly.

PP_GLAT: Global number of latitudes for truncation.

PP_GLON: Global number of longitudes for truncation.

PP_PLEV: Number of vertical levels in the model. Climate parameterizations assume 18 levels.

WORD4: Computational precision. For 32-bit architectures (4 bytes per floating point number) this value defined will generate a single precision (real*4) code. Undefined (the default) will generate double precision (real*8) code. Note that intrinsic functions are renamed in *params.h* depending on WORD4. For the real*8 implementations the promotion of constants MUST be done on the compile line. (See the example makefiles for appropriate compiler flags.)

I_SIZE: Number of bytes in an integer.

R_SIZE: Number of bytes in a real.

D_SIZE: Number of bytes in a double precision real.

PP_INTEL_PFS: For INTEL only. Do output and input using the Intel Parallel File System.

NX_FORCETYPE: For INTEL only. Message passing protocol.

MPI: For MPI Standard message passing.

MPL: For MPL message passing.

Special debugging flags useful for new installations.

PP_ERROR: To enable error and debug messages to stderr on each node. The routine *error_dup.c* determines whether each node opens a file pccm.error.???? or only node zero.

NOOUT: Disable history output.

PP_NORESTART: Disable restart output.

In addition to these flags there are several other definitions in the code that are only for the use of the developers.

5.4.2. Code Structure

The code structure is designed to permit easy portability to other parallel platforms as well as ease of modification for climate researchers. Both the computer science / parallel computing community and the climate research community are served by the design. In addition, we have put a great deal of effort into the optimization of code for good parallel performance.

For the climate researcher who wants to modify the physics parameterizations, we note that the "physics" routines are almost identical to those in CCM2. To make a change in the physics no regard for parallelism should be required. The radiation and adjustment calculations associated with a column of the atmosphere are performed entirely on processor. This is also true of surface processes associated with any given point.

The parallel programming paradigm used is single program, multiple data (SPMD) with explicit message passing. A generic message passing functionality has been assumed based

on SEND, RECV, SWAP and BCAST, which are then implemented in a machine or message specific library. Message passing implementations are available for MPI (Message Passing Interface), PVM (Parallel Virtual Machine), PICL (Portable, Instrumented Communication Library), MPL (IBM Native), and NX (Intel Native). Porting the code to another platform or message passing system should be as easy as providing the proper interface to the low level routines.

Since input and output continue to be a source of frustration for parallel computer users, we have implemented an option where all reading and writing is done from a single node (processor zero). The single node I/O should work on any parallel computer, but perhaps, not at the performance level necessary for production runs. We also provide for parallel, optimized I/O on the supported platforms. Since there are not I/O standards for distributed memory parallel programming, a new port will necessarily require some effort optimizing the I/O.

The call tree of the PCCM2 is essentially unchanged from CCM2. A significant exception to this is the elimination of the routine LINEMS. This routine has been split into PHYSICS and XFORM to accommodate blocked FFT's and transpose algorithms for the parallel spectral transform. The effect has been to separate the physics computation from the transforms.

Examining the data structures used in the PCCM2, the user will observe that the major 3d arrays in *common /com3d/* have been modified. A new subscript has been added to account for the hemisphere (1=S, 2=N).

```
      real u3(plond,plevd,platd,2,2),
     $     v3(plond,plevd,platd,2,2),
     $     t3(plond,plevd,platd,2,2),
     $     q3(plond,plev,3+pcnst,platd,2,2)
```

The first index now refers to the local (on processor) longitude index. The second index is for the vertical level and is unchanged from CCM2. The third index is the latitudinal index. The fourth is new and refers to the hemisphere. The last index refers to the time level. *plond* and *platd* have also been redefined in *pmgrid.com*.

```
      plond=plon + 1 + 2*nxpt, ! slt extended domain longitude
      platd=p_lato2 + 2*nxpt + 2*jintmx, ! slt extended domain lat.
```

They now define extra points for interpolation in SCAN1A and for processor overlap information used in SCAN1A for the semi-Lagrangian method.

In general, modifications to data structures reflect the local processors data size and not the global problem data size. This effects a decomposition of the data among the processors.

5.4.3. PCCM2 Routines

Many of the CCM2 routines have been drastically altered in the migration to PCCM2. This is particularly true of routines having to do with spectral transforms and the dynamics calculation. This section collects comments on individual routines. It is not an exhaustive list of routines modified but may prove helpful for researchers who wish to explore the code and make modifications to the algorithms.

CCM2: A call to PARLYZ has been added to initialize the parallel versions processor configuration. STEPDRV, called from CCM2, is now the driver routine for STEPON to allow dynamic memory allocation of the history buffer.

INIDAT: The first spectral transform takes place in this routine. The FFT routines are called along with the SPETRU routine which performs the spectral truncation. These sections are completely reworked for parallelism. The accumulation of the global statistics for DRY MASS and MASS OF MOISTURE are computed in INIDAT. This section of code ensures

that these global statistics will be exactly the same (bit-for-bit) regardless of the number of processors used in the computation. See the reproducibility discussion above.

SCAN1: The latitude loop in SCAN1 has been split into two parts. The first calculates physics (calls PHYSDRV) for all latitudes on the processor. Then FFT's are performed on all the latitudes simultaneously allowing efficient block and transpose algorithms to be employed. The second latitude loop transforms to spectral space (calls XFORMDRV).

Since the global integrals of the prognostic fields are computed in SCAN1, there is are constructs to ensure reproducibility of the sum on different numbers of processors. This makes the code look somewhat more complicated than the standard version.

SCAN1A: The SLT calls are in SCAN1A. The major change is the addition of an hemisphere index to the extended field arrays. This provides storage of the overlap regions between processors.

BANDIJ: The output of this routine is now a local latitude index in the extended field array. When something is wrong in the physics it often shows up first with an excessive wind which blows departure points out of the range of the overlap region. An error message is printed from BANDIJ when this occurs.

OVRLAP: The message passing for the semi-Lagrangian update of the overlap regions is done in this routine. OVRLAP is called from SLTINI which is executed once per timestep. The overlap region can be increased or decreased by modifying the parameter *nxpt* in *pmgrid.com*. The amount of data sent for the overlap is dynamically varied depending on the wind conditions at a given latitude up to the maximum specified by *nxpt*.

RFTLON: This is the wrapper for all the FFT routines and parallel algorithms. There is no restriction in the FFT's on power of two points.

SCAN2: The global integrals involved in the moisture calculation are performed in SCAN2. Due to reproducibility considerations this routine contains added parallel constructs for the global sums. The spectral synthesis of SCAN2 is performed in SPEGRD.

6. PCCM2 CODING STANDARDS

The following rules guide the coding style used in CCM2 and the PCCM2 implementation.

Common blocks in include files. Only named common should be used with one common block and associated declarations per include file. Data items in a common block must be ordered by type, with long items first to avoid alignment problems.

No GO TO or computed GOTO statements.

IMPLICIT NONE Variables should be typed using the default FORTRAN 77 typing for readability. Integers should begin with the letters (i-n) and reals should begin with (a-h,o-z). Complex quantities should be clearly identified, variable names that begin with a (c) or (z) are preferred. Parameter variables should begin with (p)

The use of parameter variables is encouraged. Parameter variables should be used directly in the code.

Comments for each subroutine should clearly identify what variables are input to the subroutine and not changed and what variables are output or changed on exit.

6.1. Modularity

The goal of modularity is to obtain code that is easily read and modified. For CHAMMP this is particularly important for the addition of new or modified physics. The need for standardized interfaces has been recognized for some time. In particular see, [Pielke, R.A. and Arrit, R.W., (1984) "A proposal to standardize models" in Bull. Am. Met. Soc. 65: 1082]

6.2. Physics Modules

The parallel model adopts the use of columnar physics modules that are "plug compatible". The CHAMMP development group adopts the guidelines set forth in "Rules for Interchange of Physical Parameterizations", by Kalnay, Kanamitsu, Pfaendtner, Sela, Suarez, Stackpole, Tuccillo, Umscheid and Williamson. These rules are duplicated for reference purposes.

- A package shall refer only to its own subprograms and the ANSI FORTRAN intrinsic functions.
- A package shall provide separate set-up and running procedures, each with a single entry point. All initialization of static data must be done in the set-up procedure, and the running procedure shall not modify the static data.
- All communication with the package shall be through the argument list at the entry points.
- The package shall not use blank COMMON
- Arguments shall be clearly documented. In particular, data items shall be defined in physical terms, and identified as being: (1) needed on input and not changed, (2) needed on input and modified, (3) simply output, or (4) workspace; and EXTERNAL subprograms shall be described in detail. All data shall be in SI units.
- The horizontal index shall be the innermost of FORTRAN arrays. The range of this index processed on each call shall be specifiable through the argument list.
- The number of levels the package uses shall be specifiable through the argument list.
- All dimensions of dummy argument arrays shall appear in the argument list.
- No array index shall exceed its declared dimension.
- The package shall not use the STOP statement.
- I/O form the package shall be limited to diagnostic output written to FORTRAN units specified in the argument list.

As point physics methods are developed these should also be plug-compatible with the appropriate extension and modification of the above rules.

7. References

[1] R. Anthes. 1986 summary of workshop on the NCAR Community Climate/Forecast models. *Bull. Amer. Meteor. Soc.*, 67:194–198, 1986.

[2] A. P. M. Baede, M. Jarraud, and U. Cubasch. Adiabatic formulation and organization of ECMWF's spectral model. ECMWF Tech. Rept. No. 15, European Centre for Medium-Range Weather Forecasts, Reading, England, 1979.

[3] L. J. Bath, J. Rosinski, and J. Olson. User's guide to NCAR CCM2. NCAR Tech. Note NCAR/TN–379+IA, National Center for Atmospheric Research, Boulder, Colo., 1992.

[4] L. M. Bath, M. A. Dias, D. L. Williamson, G. S. Williamson, and R. J. Wolski. Documentation of NCAR CCM1 program modules. NCAR Tech. Note NCAR/TN–287+IA, National Center for Atmospheric Research, Boulder, CO, 1987.

[5] L. M. Bath, M. A. Dias, D. L. Williamson, G. S. Williamson, and R. J. Wolski. User's guide to NCAR CCM1. NCAR Tech. Note NCAR/TN-287+IA, National Center for Atmospheric Research, Boulder, CO, 1987.

[6] W. Bourke, B. McAvaney, K. Puri, and R. Thurling. Global modeling of atmospheric flow by spectral methods. *Meth. Comp. Phys.*, 17:267–324, 1977.

[7] M.I. Budyko. *Heat Balance of the Earth's Surface.* Gidrometeozidat, Leningrad, 1956.

[8] J.W. Deardorff. Parameterization of the planetary boundary layer for use in general circulation models. *Mon. Wea. Rev.*, 100:93–106, 1972.

[9] Department of Energy. Building an advanced climate model: Progress plan for the CHAMMP climate modeling program. DOE Tech. Report DOE/ER-0479T, U.S. Department of Energy, Washington, D.C., December 1990.

[10] P.H. Worley D.W. Walker and J.B. Drake. Parallelizing the spectral transform method - part II. *Concurrency: Practice and Experience*, 4:509–531, 1992.

[11] E.B. Eliasen, B. Machenhauer, and E. Rasmussen. On a numerical method for integration of the hydrodynamical equations with a spectral representation of the horizontal fields. Rep. No. 2, Institut for Teoretisk Meteorologi, Kobenhavns Universitet, Denmark, 1970.

[12] R. M. Errico. A description of software for determination of normal modes of the NCAR community climate model. NCAR Tech. Note NCAR/TN-287+STR, National Center for Atmospheric Research, Boulder, CO, 1986.

[13] R. M. Errico and B. E. Eaton. Nonlinear normal mode initialization of the NCAR CCM. NCAR Tech. Note NCAR/TN-303+IA, National Center for Atmospheric Research, Boulder, CO, 1987.

[14] I. Foster, W. Gropp, and R. Stevens. The parallel scalability of one- and two-dimensional decompositions of the spectral transform method. Technical Report ???, Argonne National Laboratory, Argonne, IL, December 1990.

[15] I. T. Foster and P. H. Worley. Parallel algorithms for the spectral transform method. Technical Report ORNL/TM-12507, Oak Ridge National Laboratory, Oak Ridge, TN, 1994.

[16] J. J. Hack, L. M. Bath, G. S. Williamson, and B. A. Boville. Modifications and enhancements to the NCAR Community Climate Model (CCM1). NCAR Tech. Note NCAR/TN-336+STR, National Center for Atmospheric Research, Boulder, Colo., 1989.

[17] J.J. Hack, J.M. Rosinski, D.L. Williamson, B.A. Boville, and J.E. Truesdale. Computational design of the NCAR Community Climate Model. *Parallel Computing*, 21:1545–1570, 1995.

[18] S.W. Hammond, R.D. Loft, J.M.Dennis, and R.K. Sato. Implementation and performance issues of a massively parallel atmospheric model. *Parallel Computing*, 21:1593–1620, 1995.

[19] J.L. Holloway and S. Manabe. Simulation of climate by a global general circulation model 1. hydrologic cycle and heat balance. *Mon. Wea. Rev.*, 99:335–370, 1971.

[20] David P. Baumhefner James W. Hurrell, James J. Hack. Comparison of NCAR Community Climate Model (CCM) climates. NCAR Tech. Note NCAR/TN-395+STR, National Center for Atmospheric Research, Boulder, Colo., 1993.

[21] J.Drake, I. Foster, J. Michalakes, B. Toonen, and P. Worley. Design and performance of a scalable Parallel Community Climate Model. *Parallel Computing*, 21:1571–1592, 1995.

[22] J.T. Kiehl, R.J. Wolski, B.P. Briegleb, and V. Ramanathan. Documentation of radiation and cloud routines in the NCAR Community Climate Model (CCM1). NCAR Tech. Note NCAR/TN-288+IA, National Center for Atmospheric Research, Boulder, Colo., 1987.

[23] B. Machenhauer. The spectral method. In *Numerical Methods Used in Atmospheric Models*, volume II of *GARP Pub. Ser. No. 17. JOC*, chapter 3, pages 121–275. World Meteorological Organization, Geneva, Switzerland, 1979.

[24] S. Manabe, J. Smagorinsky, and R.F. Strickler. Simulated climatology of a general circulation model with a hydrologic cycle. *Mon. Wea. Rev.*, 93:769–798, 1965.

[25] T.A. Mayer. Generation of CCM format history tapes from analyzed data. NCAR Tech. Note NCAR/TN-322+STR, National Center for Atmospheric Research, Boulder, Colo., 1988.

[26] B. J. McAvaney, W. Bourke, and K. Puri. A global spectral model for simulation of the general circulation. *J. Atmos. Sci.*, 35:1557–1583, 1978.

[27] S. A. Orszag. Transform method for calculation of vector-coupled sums: Application to the spectral form of the vorticity equation. *J. Atmos. Sci.*, 27:890–895, 1970.

[28] V. Ramanathan, E.J. Pitcher, R.C. Malone, and M.L. Blackmon. The response of a spectral general circulation model to refinements in radiative processes. *J. Atmos. Sci.*, 40:605–630, 1983.

[29] J.M. Rosinski and D.L. Williamson. On the accumulation of rounding errors in a global atmospheric model. *SIAM J. Scientific Computing*, in press.

[30] J.-F. Royer. Correction of negative mixing ratios in spectral models by global horizontal borrowing. *Mon. Wea. Rev.*, 114:1406–1410, 1986.

[31] R. K. Sato, L. M. Bath, D. L. Williamson, and G. S. Williamson. User's guide to NCAR CCM0B. NCAR Tech. Note NCAR/TN-211+IA, NTIS PB83 263988, National Center for Atmospheric Research, Boulder, Colo., 1983.

[32] J. Smagorinsky. General circulation experiments with the primitive equations. 1. the basic experiment. *Mon. Wea. Rev.*, 91:98–164, 1963.

[33] J. Smagorinsky, S. Manabe, and J.L. Holloway. Numerical results from a nine-level general circulation model of the atmosphere. *Mon. Wea. Rev.*, 93:727–768, 1965.

[34] I. A. Stegun. Legendre functions. In M. Abramowitz and I. A. Stegun, editors, *Handbook of Mathematical Functions*, chapter 8, pages 332–353. Dover Publications, New York, 1972.

[35] R. A. van de Geijn. On global combine operations. LAPACK Working Note 29, Computer Science Department, University of Tennessee, Knoxville, TN 37996, April 1991.

[36] D. W. Walker, P. H. Worley, and J. B. Drake. Parallelizing the spectral transform method. Part II. *Concurrency: Practice and Experience*, 4(7):509–531, October 1992.

[37] W.M. Washington. Documentation for the Community Climate Model (CCM), version 0. Technical Report NTIS PB82-194192, Climate Section, National Center for Atmospheric Research, Boulder, Colo., 1982.

[38] D. L. Williamson. Description of NCAR Community Climate Model (CCM0B). NCAR Tech. Note NCAR/TN-210+STR, NTIS PB83 231068, National Center for Atmospheric Research, Boulder, Colo., 1983.

[39] D. L. Williamson, L. M. Bath, R. K. Sato, T. A. Mayer, and M. L. Kuhn. Documentation of NCAR CCM0B program modules. NCAR Tech. Note NCAR/TN-212+IA, NTIS PB83 263996, National Center for Atmospheric Research, Boulder, Colo., 1983.

[40] D. L. Williamson, J. T. Kiehl, V. Ramanathan, R. E. Dickinson, and J.J. Hack. Description of NCAR Community Climate Model (CCM1). NCAR Tech. Note NCAR/TN–285+STR, National Center for Atmospheric Research, Boulder, Colo., 1987.

[41] D. L. Williamson and P. J. Rasch. Two-dimensional semi-Lagrangian transport with shape-preserving interpolation. *Mon. Wea. Rev.*, 117:102–129, 1989.

[42] D.L. Williamson and G.S. Williamson. Circulation statistics from January and July simulations with the NCAR Community Climate Model (CCM0B). NCAR Tech. Note NCAR/TN-244+STR, NTIS PB85 165637/AS, National Center for Atmospheric Research, Boulder, Colo., 1984.

[43] G. S. Williamson and D. L. Williamson. Circulation statistics from seasonal and perpetual January and July simulations with the NCAR Community Climate Model (CCM1):r15. NCAR Tech. Note NCAR/TN-302+STR, NTIS PB88–192620/AS, National Center for Atmospheric Research, Boulder, Colo., 1987.

[44] D. L. Williamson, Ed. Report of the second workshop on the community climate model. NCAR Tech Note NCAR/TN-310+PROC, National Center for Atmospheric Research, Boulder, CO 80307, 1988.

[45] D. L. Williamson, Ed. CCM progress report - July 1990. NCAR Tech Note NCAR/TN–351+PPR, National Center for Atmospheric Research, Boulder, CO, July 1990.

[46] P.H. Worley and J.B. Drake. Parallelizing the spectral transform method - part I. *Concurrency: Practice and Experience*, 4:269–291, 1992.

[47] P.H. Worley and I.T. Foster. Parallel spectral transform shallow water model: A runtime-tunable parallel benchmark code. In *Scalable High Performance Computing Conference*, pages 207–214. IEEE Computer Society Press, 1994.

INTERNAL DISTRIBUTION

1. E. F. D'Azevedo
2. T. S. Darland
3–4. J. B. Drake
5–6. R. E. Flanery
7. K. L. Kliewer
8. M. R. Leuze
9. C. E. Oliver
10–14. S. A. Raby
15. B. A. Riley
16–17. B. D. Semeraro
18. T. Sheehan
19. R. F. Sincovec
20–21. P. H. Worley
22. K-25 Applied Tech. Library
23. Y-12 Technical Library
24. Laboratory Records - RC
25–26. Laboratory Records Dept.
27. Central Research Library
28. ORNL Patent Office

EXTERNAL DISTRIBUTION

29. David C. Bader, Atmospheric and Climate Research Division, Office of Health and Environmental Research, Office of Energy Research, ER-76, U.S. Department of Energy, Washington, DC 20585

30. Edward H. Barsis, Computer Science and Mathematics, P. O. Box 5800, Sandia National Laboratory, Albuquerque, NM 87185

31. Dominique Bennett, CERFACS, 42 Avenue Gustave Coriolis, 31057 Toulouse Cedex, FRANCE

32. Bill L. Buzbee, Scientific Computing Division, National Center for Atmospheric Research, P.O. Box 3000, Boulder, CO 80307

33. Captain Edward A. Carmona, Parallel Computing Research Group, U. S. Air Force Weapons Laboratory, Kirtland AFB, NM 87117

34. Peter Campbell, Environmental Science Divsion, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

35. Professor I-Liang Chern, Department of Mathematics, National Taiwan University, Taipei, Taiwan, R.O.C.

36. Ray Cline, Sandia National Laboratories, Center for Computational Engineering Box 969 Livermore, CA 94550

37. Alexandre Chorin, Mathematics Department, Lawrence Berkeley Laboratory, Berkeley, CA 94720

38. James Corones, Ames Laboratory, Iowa State University, Ames, IA 50011

39. Jean Coté, RPN, 2121 Transcanada Highway, Suite 508, Dorval, Quebec H9P 1J3, CANADA

40. William Dannevik, Lawrence Livermore National Laboratory, P. O. Box 808, L-16, Livermore, CA 94550

41. Iain S. Duff, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon OX11 0QX, England

42. John Dukowicz, Los Alamos National Laboratory, Group T-3, Los Alamos, NM 87545

43. Ian Foster, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

44. Geoffrey C. Fox, NPAC, 111 College Place, Syracuse University, Syracuse, NY 13244-4100

45. Dr. Rhys Francis, Div. of Information Technology, CSIRO, 723 Swanston Street, Carlton, Vic. 3053, AUSTRALIA

46. Paul O. Frederickson, ACL, MS B287, Los Alamos National Laboratory, Los Alamos, NM 87545

47. John Gustafson, 236 Wilhelm, Ames Laboratory, Iowa State University, Ames, IA 50011

48. Phil Gresho, Lawrence Livermore National Laboratory, L-262, P. O. Box 808, Livermore, CA 94550

49. William D. Gropp, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

50. James J. Hack, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

51. Michael T. Heath, University of Illinois, Department of Computer Science, 2304 Digital Computer Laboratory, 1304 West Springfield Avenue, Urbana, IL 61801-2987

52. Michael Henderson, Los Alamos National Laboratory, Group T-3, Los Alamos, NM 87545

53. Dr. Fred Howes, Office of Scientific Computing, ER-7, Applied Mathematical Sciences, Office of Energy Research, U. S. Department of Energy, Washington, DC 20585

54. Dr. Gary Johnson, Office of Scientific Computing, ER-7, Applied Mathematical Sciences, Office of Energy Research, U. S. Department of Energy, Washington, DC 20585

55. Lennart Johnsson, Thinking Machines Inc., 245 First Street, Cambridge, MA 02142-1214

56. J.R. Jump, ECE Dept., Rice University, P.O. Box 1892, Houston, TX 77251

57. Malvyn Kalos, Cornell Theory Center, Engineering and Theory Center Bldg., Cornell University, Ithaca, NY 14853-3901

58. Hans Kaper, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

59. Alan H. Karp, IBM Scientific Center, 1530 Page Mill Road, Palo Alto, CA 94304

60. Kenneth Kennedy, Department of Computer Science, Rice University, P. O. Box 1892, Houston, Texas 77001

61. Tom Kitchens, ER-7, Applied Mathematical Sciences, Scientific Computing Staff, Office of Energy Research, Office G-437 Germantown, Washington, DC 20585

62. Rich Loft, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

63. Michael C. MacCracken, Lawrence Livermore National Laboratory, L-262, P. O. Box 808, Livermore, CA 94550

64. Norman D. Malmuth, Science Center, Rockwell International Corporation, 1049 Camino Dos Rios, P.O. Box 1085, Thousand Oaks, CA 91358

65. Robert Malone, C-DO/ACL, MS B287, Los Alamos National Laboratory, Los Alamos, NM 87545

66. Len Margolin, Los Alamos National Laboratory, Los Alamos, NM 87545

67. Hal Marshall Laboratory for Scientific Computation, Rm. 271 Cooley Bld., University of Michigan, Ann Arbor, MI 48109-2104

68. Frank McCabe, Department of Computing, Imperial College of Science and Technology, 180 Queens Gate, London SW7 2BZ, ENGLAND

69. James McGraw, Lawrence Livermore National Laboratory, L-306, P. O. Box 808, Livermore, CA 94550

70. Paul C. Messina, Mail Code 158-79, California Institute of Technology, 1201 E. California Blvd. Pasadena, CA 91125

71. John G. Michalakes, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

72. Dr. David Nelson, Director of Scientific Computing, ER-7, Applied Mathematical Sciences, Office of Energy Research, U. S. Department of Energy, Washington, DC 20585

73. Joseph Oliger, Computer Science Department, Stanford University, Stanford, CA 94305

74. Robert O'Malley, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, Troy, NY 12180-3590

75. James M. Ortega, Department of Applied Mathematics, Thornton Hall, University of Virginia, Charlottesville, VA 22901

76. Ron Peierls, Applied Mathematical Department, Brookhaven National Laboratory, Upton, NY 11973

77. Richard Pelz, Dept. of Mechanical and Aerospace Engineering, Rutgers University, Piscataway, NJ 08855-0909

78. Andrew Priestley, Institute for Computational Fluid Dynamics, Reading University, Reading RG6 2AX, ENGLAND

79. Lee Riedinger, Physics Department 505 Nielsen Physics, Knoxville, TN 37996

80. William C. Skamarock, 3973 Escuela Court, Boulder, CO 80301

81. Richard Smith, Los Alamos National Laboratory, Group T-3, Mail Stop B2316, Los Alamos, NM 87545

82. Peter Smolarkiewicz, National Center for Atmospheric Research, MMM Group, P. O. Box 3000, Boulder, CO 80307

83. Jurgen Steppeler, DWD, Frankfurterstr 135, 6050 Offenbach, WEST GERMANY

84. Rick Stevens, Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439

85. Paul N. Swarztrauber, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

86. Wei Pai Tang, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1

87. Robert G. Voigt, ICASE, MS 132-C, NASA Langley Research Center, Hampton, VA 23665

88. Andrew B. White, Los Alamos National Laboratory, P. O. Box 1663, MS-265, Los Alamos, NM 87545

89. David L. Williamson, National Center for Atmospheric Research, P. O. Box 3000, Boulder, CO 80307

90. Samuel Yee, Air Force Geophysics Lab, Department LYP, Hancom AFB, Bedford, MA 01731

91. Office of Assistant Manager for Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P. O. Box 2001, Oak Ridge, TN 37831-8600

92–93. Office of Scientific & Technical Information, P. O. Box 62, Oak Ridge, TN 37831