Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

1

REPORT DOCUMENTATION PAGE

AD-A206 102

1b. RESTRICTIVE MARKINGS	
3. DISTRIBUTION/AVAILABILITY OF REPORT	Approved for public release; distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	P248-FR
5. MONITORING ORGANIZATION REPORT NUMBER(S)	AFOSR-TR- 89-0272
6a. NAME OF PERFORMING ORGANIZATION	Atmospheric and Environmental Research, Inc.
6b. OFFICE SYMBOL (If applicable)	
7a. NAME OF MONITORING ORGANIZATION	Air Force Office of Scientific Research
6c. ADDRESS (City, State, and ZIP Code)	840 Memorial Drive Cambridge, MA 02139
7b. ADDRESS (City, State, and ZIP Code)	Directorate of Chemical and Atmospheric Sciences Bolling AFB, DC 20332-6448
8a. NAME OF FUNDING/SPONSORING ORGANIZATION	AFOSR
8b. OFFICE SYMBOL (If applicable)	NC
9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	F49620-88-C-0105
8c. ADDRESS (City, State, and ZIP Code)	Bolling AFB, DC 20332-6448

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
	[illegible]	[illegible]	

11. TITLE (Include Security Classification)
On the use of multiprocessing computers for global numerical weather prediction

12. PERSONAL AUTHOR(S) Ross N. Hoffman and T. Nehrkorn

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final Report	FROM 88 Aug. TO 89 Jan.	1989 February 15	41

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Numerical weather prediction, global spectral model multiprocessor

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A preliminary exploration is made of the uses of multiprocessing computers for large scale NWP using spectral models. In general if global communication between processors is relatively fast and easy, then implementing spectral models is feasible.

The global spectral model is recast in terms of latitude and wavenumber tasks. This approach has a number of advantages: The entire algorithm is macrotasked. Only a handful of crucial pointers need to be locked. The spectral transform calculations are localized so that arithmetic always follows the same ordering and all results are exactly reproducible.

The latitude wavenumber tasking scheme is implemented and tested on the Sequent Balance, a shared memory multiple instruction multiple data device. It is argued that this scheme could be easily extended and applied to larger machines of this class and provide a good starting point for distributed memory machines.

The potential of single instruction multiple data machines is huge. A proposed algorithm for this class of machine uses a processor for each horizontal grid point.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE (Include Area Code)	22c. OFFICE SYMBOL
J. Stobie	202 767 4963	AFOSR/NC

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

1. Introduction

Historically, progress in numerical weather prediction (NWP) has paralleled progress in computing capabilities, with each discipline often spurring advances in the other (Schuman, 1982). (A list of all acronyms used here appears in the Appendix.) Advances in computing have also permitted the development of detailed general circulation models (GCMs) for climate studies. Operational NWP is particularly sensitive to computing advances because real time weather forecasting imposes efficiency constraints that limit the degree of sophistication of the models used and hence the accuracy attained by NWP when even the most powerful computers are used; improvements in computation speed often herald improvements in forecasting ability. This is demonstrated in Fig. 1.

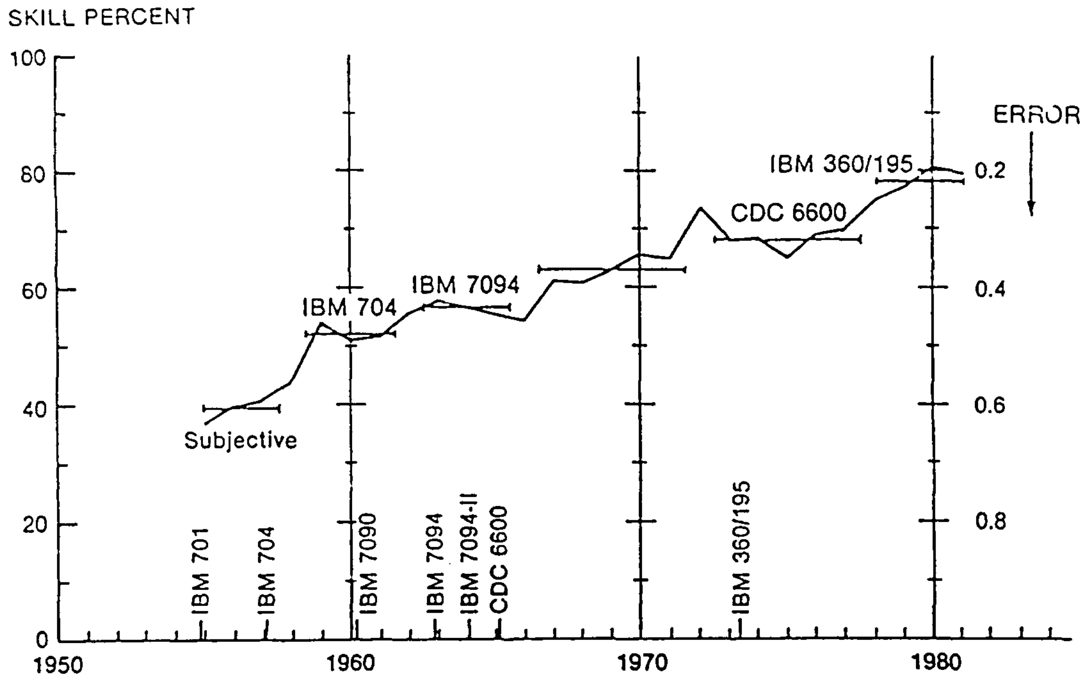


Fig. 1. Increase of NMC prediction skill. Record of skill, averaged annually, of predictions of atmospheric circulation made by the National Meteorological Center. Specifically, the predictions are for 36 hours at 50 kPa (approximately 5.6 km high) over North America. The horizontal bars show averages for the years during which no major changes in models occurred. The tic marks at the bottom show the time of acquisition of each machine. The measure of skill is derived from the so-called S1 score, which is a measure of normalized error in pressure gradients. A chart with an S1 score of 20 is perfect for all practical purposes, and one with 70 is worthless. Skill (percent) is 2 x (70 - S1), which yields 0 for a worthless chart, and 100 for a practically perfect one. Error = 1 - (skill/100) is shown on the right (from Shuman, 1982)

A recent trend in computer technology is the construction of machines composed of a number of processors linked together, allowing multiple simultaneous computations. Such multiprocessing computers contain two or more, in some cases tens of thousands, of individual processing units. These processors may operate in lock step or they may be nearly autonomous. The former case, single instruction multiple data (SIMD), is particularly relevant for image processing applications, while the latter case, multiple instruction multiple data (MIMD), is of more general interest. Multiprocessor computers give a substantial increase in computing speed for large complex numerical problems. Thus, they provide an opportunity to reduce significantly the computing time for GCM studies and NWP. However, a special effort is required to exploit this opportunity. According to the recommendations of GARP Special Report No. 43 (WMO, 1985), "[I]ncreasing computer power and changes in computer architecture...[have] always played a major part in determining integration techniques and will continue to do so if advances in computer technology are to be fully exploited. Therefore, study to determine the most efficient or appropriate algorithms will continue to be necessary, taking into account, for example, the future use of multiprocessors with a very large storage. Development costs of such optimized computer codes will be high, and they should be designed from the outset to be widely usable..."

The design of a computer model intended for a multiprocessor must satisfy a number of special concerns which are not encountered when using a conventional machine. Kung (1980), Fox and Otto (1984), Kuck et al. (1986), Ranka et al. (1988), White and Wiley (1988), Snelling (1988) and Hoffmann and Snelling (1988) describe some of them. An efficient model design minimizes the amount of processor idle time. This is accomplished in part by ensuring that each processor performs roughly the same amount of work (by ensuring load balance). Adaptive algorithms such as the one described in Section 5.1 dynamically balance the load during processing. To accomplish this, a special monitor data structure which may be accessed by only a single process at a time maintains a list of tasks to be performed. Processor idle time may also be minimized by synchronizing communication among processors so that all communicate simultaneously whenever com-

A-1

munication occurs (communication balance). In addition, communication must be organized to preserve the integrity of the data. Strict maintenance of separate load and communication balance is not necessary, of course, since it may be desirable to have some processors communicate while others compute; the two balances are, however, useful guides. In addition, cost tradeoffs between communication and processing must be considered.

The best way to handle these concerns depends partly on hardware features of the machine used, such as the number of processors, the cost of communication, and whether or not there is shared memory among these processors. The manner in which these concerns are satisfied will also depend on how much programming flexibility is allowed by the computer's operating system. System services and the amount of effort to develop and debug parallel applications varies widely (Karp and Babb, 1988). Gustafson et al. (1988) discuss different ways of measuring efficiency. They point out, that for most scientific problems, the most reasonable method of benchmarking scales the problem up as the number of processors increase.

Separate but concurrent with the development of multiprocessor machines has been the development of efficient methods for spectral modeling of the atmosphere. Global spectral models (GSMs) are now widely used for both operational NWP (e.g., at AFGWC, NMC and ECMWF) and GCM climate research (e.g., at NCAR and GFDL). Although most major forecasting and modeling centers currently use spectral models, gridpoint models can also be used, and this approach is favored by some (NEPRF, GLA, UCLA). Before the discovery of the fast fourier transform (FFT), gridpoint models had the advantage of speed over spectral models. With the FFT, spectral models are faster than gridpoint models of comparable resolution and have several other advantages, chiefly: (1) accurate horizontal differentiation, (2) no aliasing, (3) no polar singularities and (4) easy implementation of the semi-implicit time scheme (Bourke et al., 1977). Spectral models are, therefore, the preferred choice, and we expect that they will remain so for the immediate future, especially for operational large-scale forecasting models.

A multiprocessing GSM model is therefore a particularly relevant and important application of multiprocessing computers since NWP is time constrained and limited by available computer power. A multiprocessing spectral model is also relevant to climate research (e.g. what is the effect of doubling and quadrupling CO_2). For very large models, memory becomes an important concern. As resolution increases to take advantage of cpu speed, memory becomes the most critical resource. In this case there is a tradeoff between efficient usage of the processors and the memory. To conserve the amount of memory required it may be preferable to make model runs in sequence using a multitasking model. Even if full speedup is not achieved this may be more efficient than running copies of the model in parallel, if these can not all be contained in fast memory.

Most hydrodynamical applications of multiprocessors have dealt with grid point models. In contrast to grid point models, interactions in a spectral model are global, that is they involve all modes. Furthermore spectral models are usually implemented with a transform technique in which both a spectral and a grid point representation are used. To date, multiprocessing GSMs have been quite successful but rather specialized and limited in usefulness to a handful of processors and a specific machine. At NCAR, R. Chervin has developed a multitasking version of the earliest Community Climate Model (CCM) for the X-MP. In Chervin's version multiple latitudes and multiple wavenumbers are processed simultaneously but otherwise the original program structure is retained. A completely new multitasking version of the CCM is under study (B. Boville, pers. comm.). At ECMWF, investigators have adapted their global forecast model to run using multiple processors of their CRAY X-MP computers (Gibson, 1985; Dent, 1988). Satisfactory speedups for up to four processors were achieved. The ECMWF design is, however, highly constrained by memory considerations. As a result, the ECMWF GSM is rather complicated and somewhat machine specific.

The ultimate objective of our research is to obtain improved predictive capability of an extremely complex system, the global atmosphere, by improving existing or devising new mathematical models and algorithms for use in multitasking global spectral NWP models. We expect that our effort will produce a multiprocessor GSM that will be a useful and efficient tool for operational and research purposes. The algorithms will be welldocumented and should be easily portable, enabling modelers elsewhere to use them to advantage. Since operational NWP is usually limited in part by existing computer technology, our work should facilitate the transfer of currently developing technology to operational use in an area that has strong economic and societal impact. Also, the more efficient tool made available by our research should permit more detailed simulation studies with GCMs of important climate change problems such as nuclear winter and the effects of changing CO_2 concentration.

The current study demonstrates that spectral models may be adapted to a variety of multiprocessing environment. Our objectives in Phase I were to determine the requirements of an efficient GSM running on a multiprocessor computer and to design algorithms that satisfy these requirements. There are three classes of machine architectures we consider:

1) Shared memory multiple instruction multiple data (sMIMD);

2) Distributed memory multiple instruction multiple data (dMIMD); and

3) Single instruction multiple data (SIMD).

Examples of sMIMD machines are those of Sequent, Alliant, Cray and ETA. In these machines, shared memory may be closely coupled (e.g. Alliant) or loosely coupled (e.g. ETA) to the processors. Examples of dMIMD machines are the Intel and Ncube hypercube machines and examples of sMIMD machines are the Goodyear MPP and Thinking Machines Connection Machine. These classes of machines are further discussed and contrasted with reference to NWP in Section 6. We note that as we progress through this list, the level of risk, level of effort and development time are expected to increase. Because of the limited resources available to Phase I, we have concentrated so far on sMIMD algorithms.

In addition we coded and tested two algorithms on the AFWL/SCP Sequent Balance computer, a sMIMD machine, thereby demonstrating the feasibility of these designs. These experiments have also helped to delineate sources of inefficiency and cor esponding i[illegible]ements. For the most part our Phase I algorithms are independent of specifics of the Sequent system and would be relatively easy to transport to other sMIMD machines such as the Alliant, CRAY 2, ETA 10, etc. Our codes are in fact 99% standard fortran (ANSI X3.9-1978).

Our first sMIMD algorithm, denoted latitude tasking, is the simplest, most straightforward application of multitasking to the existing GSM. Our second sMIMD algorithm, denoted latitude wavenumber tasking, highlights the fact that spectral as well as grid point models have a natural decomposition into tasks. In grid models of course one can map a physical domain onto the computational domain, thereby dividing the problem into tasks. As a result communication between tasks is required at the subdomain boundaries. In spectral models there is also a mapping of physical and spectral domains onto the computational domain. Now communication is required not at boundaries but at transforms between representations.

Access to the Sequent computer was provided to our project free of charge. Without this access and support from AFWL personnel, we would not have made as much progress during Phase I as we did. We take this opportunity to thank AFWL/SC.

The plan of this report is the following: Section 2 describes our contract reporting requirements and fulfills most of the same. Section 3 then schematically describes the functions of a general GSM. This description is greatly simplified, but contains all the essentials. Then Sections 4 and 5 describe our sMIMD algorithms and results for latitude tasking and latitude wavenumber tasking approaches respectively. Section 6 discusses the extension of our work to dMIMD and SIMD architectures, which we anticipate will be the focus of our Phase II work. Section 7 contains a summary and our concluding remarks.

2. Contract reporting requirements

This report provides a comprehensive, cumulative and substantive summary of the progress and significant accomplishments achieved during the total period of the research effort. Since the results of the research effort have not previously been reported in scientific or technical publications, this report provides sufficient substantive detailed discussions of the findings and accomplishments obtained in the pursuit of the planned research objectives in Sections 4, 5 and 6.

Further, in fulfillment of our reporting requirements we include in the body of this report in the following paragraphs a summary overview of the entire contract effort. This overview follows the instructions of paragraph 3.b.(2) of Exhibit A, Reporting Requirements Under Contracts Issued by AFOSR, of the subject contract.

a) Statement of work. The contract, Part 1, Section B, Item 0001AA, states that research will:

   1. Recast global spectral Numerical Weather Prediction (NWP) models in terms of tasks and subtasks most advantageous to a multi-processing computer environment.

   2. Design and analyze multitasking implementations of such models.

   3. Develop and test a simplified baseline, hemispheric spectral model for simulation testing, based on best design concepts.

   4. Code as much of the model above as possible in FORTRAN 77.

   5. Test the code on a simulated or real multiprocessor as time and resources permit.

b) Status. We have attained all our Phase I research goals and made significant progress towards achieving our overall research objectives. Specific accomplishments include the development and testing on the 16 processor Sequent computer at AFWL/SCP of two distinct algorithms for numerical weather prediction. These algorithms, as well as the results of our tests are reported in Sections 4 and 5.

c) Journal publications. None have been prepared. We plan to submit a paper based on the findings reported here to the SIAM J. Sci. Stat. Comput., entitled "Algorithms for multiprocessing spectral numerical weather prediction models" by R. N. Hoffman and T. Nehrkorn.

d) Personnel involved in the contract effort were R. N. Hoffman, T. Nehrkorn, M. Mickelson and J.-F. Louis. No degrees were awarded to these persons during the contract period.

e) i) Papers presented. None.
ii) Consultative and advisory functions to other laboratories and agencies. None.

f) New discoveries, inventions, patents or specific applications stemming from the research effort. None.

g) Other statements which can provide additional insight and information for assessing and evaluating the progress and accomplishments achieved in the research effort. See Section 7, the conclusions and summary section.

3. Schematic GSM algorithm

The GSM actually does most calculations in gridpoint space. Only linear terms are calculated in spectral space. Other terms, such as advection, diabatic physical processes such as moist convection, etc. are localized in grid point space and much easier to calculate there. If we consider, for the sake of explanation, an adiabatic spectral model, then a simple graphical description of the functioning of the model is given in Fig. 2. Beginning with spectral values of the variables (upper left corner), the model transforms these to values of the grid point variables and their spatial derivatives. These are used to calculate the (nonlinear part of the) tendencies which are then transformed back to spectral space. (Some linear terms are added to the tendencies in spectral space, including higher order horizontal diffusion.) The tendencies in spectral space and variables at previous time levels in spectral space are then used by the time stepping procedure to obtain spectral values at the new time. The cycle is continued until the desired forecast length is achieved.

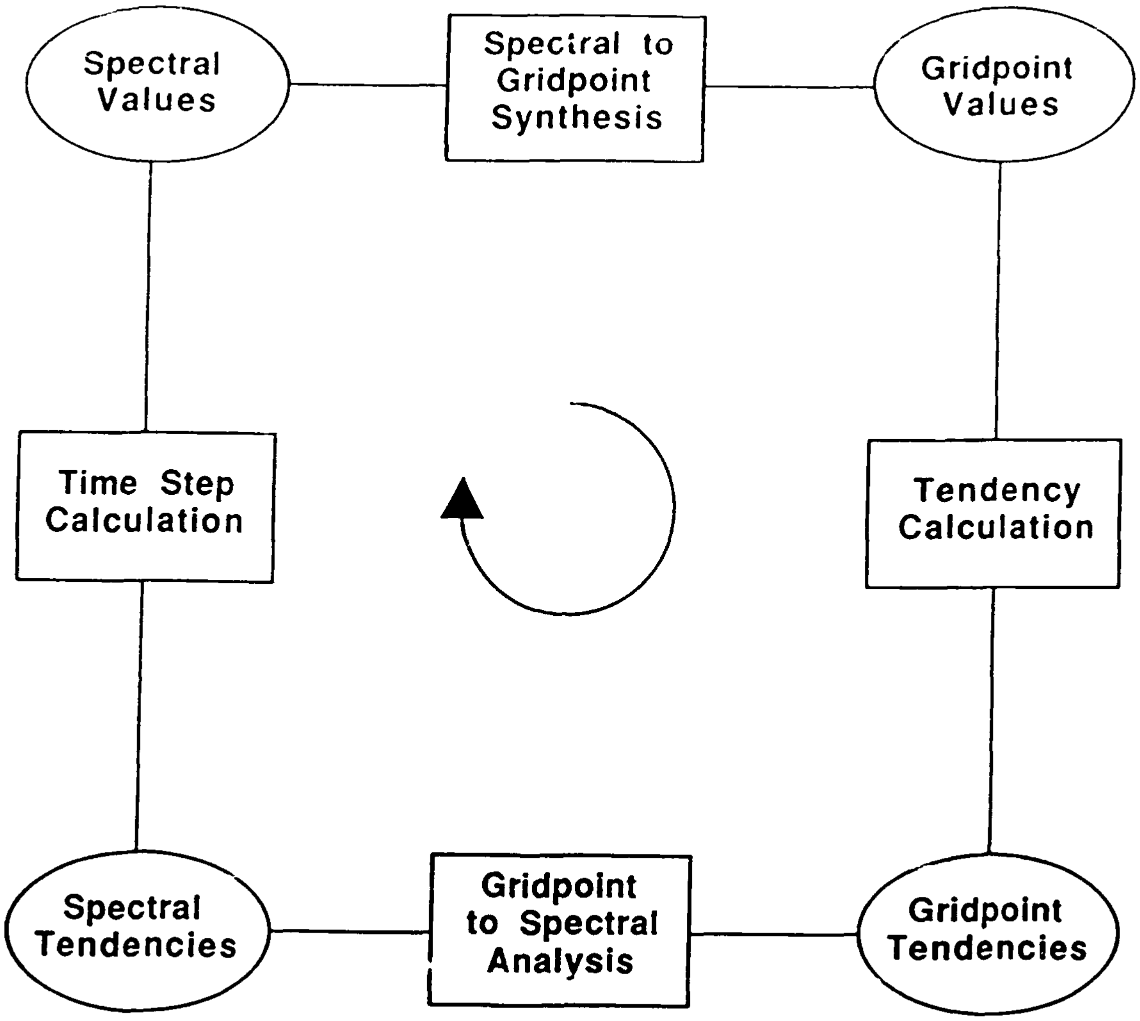


Fig. 2. Functional overview of GSM algorithm

As the basis for describing our multiprocessing algorithms, we first consider a representative but schematic GSM algorithm. Let $\underset{\sim}{X}(\lambda, \mu)$ be all gridpoint variables in all layers at longitude λ and sine of latitude μ. The fourier transform of $\underset{\sim}{X}$ is $\underset{\sim}{X}_m$ and the Legendre transform of $\underset{\sim}{X}_m$ is $\underset{\sim}{X}_m^n$. where m is the longitudinal wavenumber and n is the meridional mode number. For clarity of notation, μ, λ and normalization factors will be omitted in most of the equations that follow.

A representative time step is composed of the following steps:

(1) Legendre synthesis at each μ for each m:

$$\underset{\sim}{X}_m = \sum_n \underset{\sim}{X}_m^n P_m^n,$$

where P_m^n are the associated Legendre polynomials. Meridional derivatives of the fields are also calculated at this time by replacing P_m^n by its derivative in the above expression.

(2) Fourier synthesis at each μ for each λ:

$$\underset{\sim}{X} = \sum_m \underset{\sim}{X}_m e^{im\lambda}.$$

Longitudinal derivatives are also calculated at this time by replacing $e^{im\lambda}$ by $ime^{im\lambda}$ in the above expression.

(3) Gridpoint operations at each (λ, μ) coordinate:

$$\underset{\sim}{f} = \underset{\sim}{F}(\underset{\sim}{X}),$$

where $\underset{\sim}{f}(\lambda, \mu)$ is a vector of gridpoint tendency and flux terms in all layers, and where $\underset{\sim}{F}$ is a nonlinear function of $\underset{\sim}{X}$ and its derivatives at a single latitude-longitude point. This step involves vertical coupling.

(4) Fourier analysis at each μ for each m:

$$\underset{\sim}{f}_m = \sum_\lambda \underset{\sim}{f}(\lambda)\ e^{-im\lambda}$$

(5) Gauss-Legendre integration for each m and n:

$$f_{-m}^{n} = \sum_{\mu} w(\mu)\, P_{m}^{n}(\mu)\, f_{-m}(\mu)$$

where $w(\mu)$ are the Gauss-Legendre weights. Flux divergence terms are integrated by parts. A more complicated, but for present purposes equivalent, expression is used for these terms.

(6) Time integration for each m and n. Once the time tendencies are assembled, the model state is advanced in time by solving the semi-implicit equation, which is schematically given by

$$\left[X_{-m}^{n}\big|_{t+\Delta t} - X_{-m}^{n}\big|_{t-\Delta t}\right]/(2\Delta t) = \left(f_{-m}^{n} - L_{n} X_{-m}^{n}\right)\Big|_{t} + L_{n}\left[X_{-m}^{n}\big|_{t+\Delta t} + X_{-m}^{n}\big|_{t-\Delta t}\right]/2$$

here L_n is the linear operator governing gravity wave evolution and all quantities are evaluated at the time levels indicated. The operator L_n couples vertical layers. At the completion of step (6), we are ready to begin step (1) again.

The conventional implementation of this algorithm is to initialize the spectral tendencies f_{-m}^{n} to zero or to their linear components and then perform steps (1) through (5) with a loop on latitude, accumulating the tendencies in f_{-m}^{n}. Except for the accumulation in step (5), the computation for each latitude is independent. This implementation makes efficient use of storage, since only the gridpoint variables for a single latitude are needed at any one time. (Actually, two latitudes at a time are usually processed, one in each hemisphere; this allows further efficiencies by making use of symmetry properties.) In the next two sections, we present results using this basic algorithm, macrotasking different subsets of steps (1) through (6)

The following comments are relevant.

1. The tendency calculation need not include linear terms. These might be handled more efficiently in spectral space during the time step

process. In particular the term involving the lapacian of geopotential in the time tendency calculation is best calculated in spectral space. This term requires knowledge of surface elevation in spectral space. The time step process also includes two (linear) computational devices: horizontal (del-fourth) diffusion to limit the build up of small scale energy and the Robert time filter to prevent time splitting of the solution due to the leap frog nature of the time scheme.

2. Gridpoint values needed for tendency calculation for boundary layer flux calculations include surface characteristics such as SST and drag coefficient. In many cases the surface characteristics will be fixed, but some may vary, such as soil moisture. In addition some time lagged values are used to insure numerical stability of certain diffusive processes in the boundary layer model.

3. All physical processes might be included in the tendency calculation. However, and this has generally been the case, if some are adjustment processes, such as a dry adiabatic adjustment or a moist convective adjustment, then a second set of transforms is required. This second loop is like the first (see Fig. 2) except instead of calculating tendencies we calculate adjusted values and instead of time stepping we replace values. The Robert time filter should be performed after (or in conjunction with) this replacement process. In this diabatic model the two loops alternate.

4. To accommodate the Robert time filter we must time march one step beyond the actual final time.

5. The semi-implicit time scheme uses constant matrices which depend on wave number n, the time increment Δ and a parameter α describing the time averaging operator. These matrices should be precalculated during initialization. Since these matrices depend only on the combination of $n_t \alpha \Delta$, where n_t is one for the first time step and two thereafter, they might have to be recalculated after the first time

step. However, if $\alpha = 1$ for first step (backwards implicit) and $\alpha = 1/2$ otherwise (semi-implicit), then $n_t \ \alpha \ \Delta = \Delta$ always and the matrices in question are in fact constant.

6. There are two favored truncations in spectral space, rhomboidal and triangular. The names rhomboidal and triangular describe the shape of the region in the m,n plane retained in the truncation. Since the meteorological fields are real valued only non negative values of m are retained. For both truncations m varies from 0 to N_m. For rhomboidal truncation n varies from m to m + N_n, while for triangular truncation n varies from m to N_n. In our Phase I tests we have used rhomboidal truncation exclusively.

4. Latitude macrotasking scheme

Our first approach, denoted latitude macrotasking, is the most straightforward adaptation of the GSM to a multiprocessing environment. In the latitude macrotasking scheme, the computations of step (1) through (5) of Section 3 are considered one task, which is handled by different processors for different latitudes. The time integration, step (6), is considered a task to be handled by different processors for different spherical harmonics or modes. We began with our portable, configurable version of the GSM which is similar to the NMC and AFGL GSMs (Sela, 1980; Brenner et al., 1982). This code is entirely written in standard Fortran and permits easy changes in resolution, truncation, domain and included physics. In the present work we used 9 layers, a rhomboidal 15 truncation and no diabatic physical processes, i.e. no radiation, precipitation, etc. In addition we decoupled the humidity variable so that it is simply a tracer used for diagnostic purposes.

The transform grid had 20 latitudes (per hemisphere) and 48 longitudes. A time step of 1 hour was used. To test the proper implementation of this code on the Sequent, a 12-hour forecast on the Sequent with only boundary layer physics was compared with the results of a 12-hour forecast previously performed on the AER Harris H800 computer. The forecast on the

Harris used the full model physics (i.e. including dry and moist convection, and large scale precipitation). Global root mean square (rms) differences between the two forecasts were consistent with the different model physics: 5-10% (25%) of the forecast increment, i.e. the forecast minus the initial conditions, for the horizontal wind (temperature).

Modifications to the code to enable multiprocessing involved some rearrangement of storage, so that variables shared between the processors were stored separately from those that are local to each processor. In the latitude macrotasking scheme, all variables are shared among the processors except for those that are defined at a single latitude. Thus only the values of the Legendre polynomials, the Fourier coefficients and gridpoint values of the variables, and the gridpoint values of the nonlinear terms are local to the processors. The Fourier coefficients and the grid point values of the variables use the same storage space.

We implemented the latitude loop with macrotasking, i.e. tasks were assigned to the processors at the subroutine level. For the simple version of the GSM used in these tests, we used static task allocation, i.e. each processor was assigned a predetermined number of latitudes, since the computational load is well balanced between latitudes. In models with complicated physics, there may be large enough differences in the computational load between latitudes to warrant dynamic task allocation.

Within the loop over latitude, all the shared variables are read-only, with the exception of the spectral coefficients of the tendencies. Thus, the integrity of the data is readily preserved within the latitude loop, as long as the updating of the tendencies in step (5) is safeguarded. We implemented step (5) by first computing the partial sums for each latitude, for all the variables in one vertical layer, in local work arrays in each processor; the shared data region holding the tendencies for the particular layer was then locked, the local sums added to the global values, and the lock was released. To minimize contention for the locked data, each processor started the computations at a different layer.

The time-stepping was implemented with microtasking, i.e. tasks were assigned to the processors at the DO-LOOP level. We note that the computations in step (6) for one mode do not affect those of any other wavenumber, thus eliminating any need for safeguarding shared data before it is updated. The microtasking was implemented with compiler directives which were expanded into parallel code by a preprocessor provided as part of the Sequent FORTRAN compiler. The parallel code thus produced uses static scheduling, which is appropriate for this application since the computational load is well balanced among the different modes.

There are several possible sources of inefficiency inherent in the design and implementation of this multiprocessing scheme. The most important of these is a mismatch of the number of processors and the number of latitudes: if the number of latitudes is not a multiple of the number of processors, different processors will be assigned different numbers of latitudes, resulting in load imbalance in the latitude loop. A similar load imbalance can arise in the microtasking if the number of modes is not a multiple of the number of processors; because the number of modes is much larger than the number of latitudes, this load imbalance is less serious. In our GSM, even and odd modes (those with n-m=even or odd) are treated separately, so the relevant numbers are the number of even and odd modes. For the R15 truncation, there are 128 even and 128 odd modes. Finally, if the number of processors exceeds the number of layers, different processors may attempt to lock the same region of shared memory in the last step of the latitude loop, resulting in unproductive idle time. Because the updating of the shared memory takes up only a small amount of the total time spent in the latitude tasks (less than 10% in our case), this is a less important source of inefficiency.

We tested the macrotasking separately from the microtasking, with varying numbers of processors. The results for the macrotasking are shown in Table 1. Each row in the table represents a 12-hour forecast; all the forecast results agreed with the single-processor version of the GSM to within round-off error. The execution times, which are elapsed wall-clock times, are averaged over 12 time steps; the times given for the latitude task exclude the time used for initiating the child processes. The second column shows the minimum and maximum number of latitudes handled by each of the processors; for optimum load balance, the two should be identical. The

Table 1. **Execution times for macro-tasking of latitudes, without microtasking. See the text for details**

		TOTAL TIME STEP				LATITUDE TASK				
						Execution Time			Maximum Wait	
Number of processors	Min/Max No. of Latitudes per processor	Exec. Time (sec)	Speedup	Efficiency	Sec	Fraction of Time Step	Sec	Fraction of Lat. Step	Speedup	Efficiency
1	20/20	147.54	1.0	100.0%	134.90	91.4%	-	-	1.0	100%
4	5/5	47.03	3.13	78.4%	33.88	72.0%	0.19	0.6%	3.98	99.5%
8	2/3	34.20	4.31	53.9%	20.40	59.6%	6.85	33.6%	6.61	82.7%
10	2/2	27.56	5.35	53.5%	13.69	49.7%	0.15	1.1%	9.85	98.5%
12	1/2	28.12	5.25	43.7%	13.70	48.7%	6.85	50.0%	9.85	82.1%

speedup is defined as the ratio of single processor over multi-processor execution time, and the efficiency is the speedup divided by the number of processors. The maximum wait is the time between the first and last processor to finish the latitude tasks. The effects of load imbalancing can be seen by comparing the efficiency of the latitude task for 4 and 10 processors (well balanced) with those for 8 and 12 processors (poorly balanced). Since the maximum number of latitudes per processor does not decrease when 12 instead of 10 processors are used, no speedup is gained. In the case with 12 processors, some conflicts in accessing shared memory occurred because the number of processors exceeded the number of layers; however, this did not result in any additional inefficiency, because (fortuitously) the processors idled by a memory lockout were also the ones which only had one latitude assigned to them. The speedups shown over the entire time step illustrate the decrease in overall efficiency, if macrotasking alone is implemented, as more processors are added, due to the decreasing portion of parallel code. It should be noted, however, that the overall speedups of Table 1 represent lower bounds, since the fraction of time spent in the latitude tasks would be higher for a model with sophisticated physics.

Table 2. Execution times for macro-tasking of latitudes, combined with microtasking of the time stepping

No. of Processors	Minimum/Maximum No. of Modes per Processor	Execution Time	Speedup	Efficiency
1	128/128	147.20	1.0	100.%
4	32/32	37.54	3.92	98.0%
8	16/16	22.12	6.65	83.2%
10	12/13	15.11	9.74	97.4%
12	10/11	14.90	9.88	82.3%

Test runs using both macrotasking and microtasking are shown in Table 2. The second column shows the minimum and maximum numbers of modes assigned to each processor; again, for perfect load balance the two should be the same. It is apparent that the microtasking is very efficient, since the overall efficiencies are very close to those for the latitude tasks alone (viz. Table 2). A comparison of the case with 4 and 10 processors illustrates that the load imbalance in the microtasking (in the case with 10 processors) does not have an appreciable effect on the overall efficiency.

5. Latitude wavenumber macrotasking scheme

The success of the latitude macrotasking scheme requires a tightly coupled shared memory architecture. This approach cannot be implemented on a distributed memory machine nor is it expected to be efficient in loosely coupled systems. In this section we pursue a different tack. By breaking the basic algorithm at the Fourier representation we can attain macro-tasking for both latitude and wavenumber tasks. Here the latitude tasks are somewhat smaller than those used in the latitude only macrotasking scheme. The advantages of latitude wavenumber tasking are that the entire algorithm is macrotasked, the Legendre calculations are completely ac-complished within a single task so that reproducibility is assured and only a single memory lock is required.

For latitude wavenumber macrotasking, we consider the calculations in steps (2), (3) and (4) (of Section 3) for each latitude j to be a task handled by one of the processors. Similarly, we consider the calculations in steps (5), (6) and (1) for each wavenumber m to be a task. This implies that steps (1)-(5) are no longer performed in a loop on latitude as in the conventional single processor implementation.

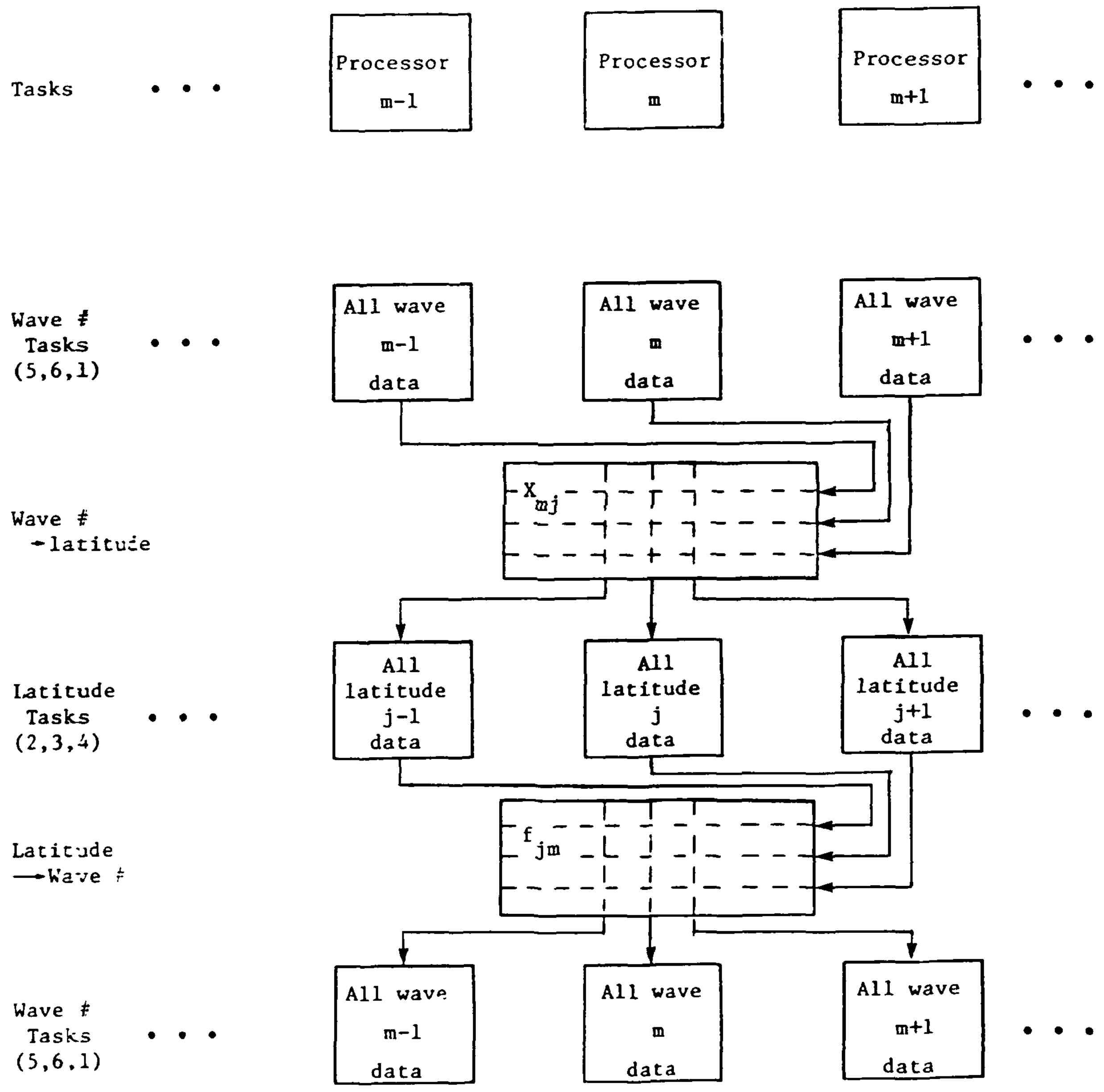
We can achieve load balance either by properly apportioning the number of tasks handled by each processor or by dynamic task allocation. We chose the former approach in Section 4 and the latter approach here (Section 5.1). For example, with static allocation, excluding diabatic processes, all latitude tasks are roughly the same size, and all wavenumber tasks are roughly the same size. This suggests evenly dividing all latitude tasks and all wavenumber tasks among available processors. Including diabatic processes upsets this balance since the amount of computations required by the parameterized diabatic processes (e.g. convection) varies with latitude. We may compensate for this lack of balance by pairing small and large tasks of the same type. With dynamic allocation imbalances are less of a problem since the most time consuming task of each type could be assigned first.

Within each task, a processor can proceed independently of the others: no communication is necessary and, hence, the integrity of the data is readily preserved. Since no partial sums are used, the ordering of tasks is immaterial. Within tasks arithmetic will follow prescheduled ordering so that results are exactly reproducible. However ordering of data movement should be immaterial. The transitions (1)-(2) and (4)-(5) require processors to share the results of their computations, and care is required for them to do this efficiently and without compromising the data. Essentially a processor's results from step (1) are placed in memory for all processors to read when performing step (2). The same procedure is followed for steps (4)-(5). A method of insuring data integrity is to maintain a critical set of indicators of the valid time of each segment of shared memory (Section 5.2). In a loosely coupled system, shared memory might be arranged so that each processor reads data which range over the entire set of Fourier coefficients but writes to only to a small contiguous part of memory (Fig. 3). However for a tightly coupled machine like the Sequent storing the Fourier variables and tendencies in the same order is advantageous (Section 5.3)

As this scheme is novel, our Phase I strategy concentrates on the transforms in our software implementation and tests (Section 5.4). The reasoning is as follows: The transforms are the part of the algorithm which require sharing of data and communication between processes while calculations at a gridpoint or at a spectral mode are localized at a single processor. Therefore, we consider only a single variable at a single vertical level and use null versions of the tendency and time step calculations. That is the tendency calculation simply sets the gridpoint

Fig. 3

Shared Storage Configuration



Note: "m data" or "j data" may be a single wave #/latitude or a group of wave #s/latitudes.

tendencies equal to the gridpoint values and the time step calculation simply sets the spectral values equal to spectral tendencies. (To be sure that any potential errors would not be masked by other errors we overwrite the old gridpoint values in the tendency calculation and the old spectral tendencies in the time step calculation with a special value, -777.) With these null versions of the tendency and time step calculations the model variables simply cycle between grid point and spectral representations.

In this algorithm subscripts and subscript ranges listed in Table 3 are employed. Generally, $N_n = N_m$. The j index of 0 is never used. We reserve k for future use as a vertical index.

Table 3. Subscript usage.

Subscript	Description	Range
i	longitude	$i = 1, N_i$
j	latitude	$j = -N_j, N_j$
k	level/layer	$k = 1, N_k$
m	zonal wave index	$m = 0, N_m$
n	total wave index	$n = m, m + N_n$
$\hat{n}$	$(n - m)$	$\hat{n} = 0, N_n$

5.1. Task allocation

We designed a dynamic task allocation algorithm for the latitude wavenumber tasking scheme. The task allocation is based on guarding three critical pointers by a software lock. These pointers define the Next_task_TYPE (NTYPE), the Next_TASK_index (NTASK) and the Next_task_TIME (NTIME). NTYPE is 1 if the next task to be performed is a latitude task, -1 if it is a wavenumber task and 0 if there are no further tasks to be performed. NTASK is the latitude or wavenumber index of the next task and NTIME is the valid model time at the start of the next task.

Whenever a processor is available to begin a new task it executes the task allocation algorithm. This algorithm locks the critical task pointers, uses them to set the local pointers describing the current task for the current processor and updates the critical task pointers. If the critical task pointers are locked, indicating that another processor is executing the task allocation algorithm, the current processor waits until the lock is released.

5.2. Data Integrity

To maintain data integrity we track the valid time of each segment of shared nonconstant storage. A segment is the output of a single latitude or wave number task. In other words for each latitude and wave number we store a time in a shared locked variable, either

VTLAT(j),	the Valid_Time_of_LATitude j or
VTWAVE(m),	the Valid_Time_of_WAVEnumber m.

While a task is updating shared nonconstant storage it locks its VTLAT or VTWAVE variable. Before a task reads a segment of shared nonconstant storage it checks that the VTLAT or VTWAVE value is correct. At the end of reading, another check may be performed, but this is optional (once debugging is complete) because of the structure of the algorithm, i.e. no latitude task can complete until all preceeding wavenumber tasks are complete and vice versa. (See the following discussion of equivalencing the Fourier arrays.) If a segment to be read is locked or not yet current the task goes on to other work or waits.

The VTLAT and VTWAVE are critical data. However an actual lock is not necessary: only one process will write to one of these locations at any one time, although many processes may be trying to read from it. An effective lock is simply to store a special value (-777) in the valid time variable.

5.3. Storage requirements

Storage requirements for this scheme are not much greater than other schemes. In the latitude tasking scheme, we maintain three complete sets of spectral coefficients, for the previous two time levels and to accumulate the spectral tendencies. In the current scheme, the spectral tendencies are accumulated locally, for each wavenumber, so we need to keep only two complete time levels of spectral data. However we also need to store the Fourier coefficients for the variables and tendencies. If stored as outlined in the following paragraphs, the Fourier data require $2*N_j/N_n$ times the storage of one set of spectral coefficients.

The Fourier variables and tendency data, XVF and XTF, must include storage for all wavenumbers included in the truncation but not for all wavenumbers required by the FFTs. Recall that higher wavenumbers generated by the quadratic terms are truncated by the FFTs. For rhomboidal truncations, we must have $N_i \geq 3*N_m + 1$. Since the Fouriers are complex the grid point storage is 50% greater than the Fourier storage. In order to store only the truncated Fourier coefficients in shared storage we must move them to local arrays before performing the FFTs.

Using local storage for the grid point values and FFTs also allows us to use the same storage for XVF and XTF. This approach requires that latitude tasks move the XVF data to local storage before performing the FFT synthesis and then move the XTF data from local storage after performing the FFT analysis. There is no possibility of loss of data integrity if the XTF and XVF arrays are congruent. By congruent we mean that the storage locations for a single latitude are equivalenced and the storage locations for a single wave number are equivalenced. This keeps the storage used by the individual tasks distinct. Data integrity is assured in this case since the algorithm structure guarantees that a wavenumber task cannot complete without data from each latitude and vice versa. That is, step (5), the Gauss-Legendre integration for any m requires Fourier data at all j. Thus none of step (5) can complete unless all of step (4) are done. Similarly step (2) the Fourier synthesis at any j requires Fourier data at all m. Again none of step (2) can complete until all of step (1) are done. Since all the Fourier arrays will have the same number of latitudes and wavenumbers they will be congruent if j and m are the first two indices and all these arrays are stored together in one common block with no intervening variables.

The calculation of the P_m^n had to be revised since they are used in the wavenumber tasks, not the latitude tasks. We considered three possibilities: Option A is to precalculate all P_m^n. This option requires storage of $N_m*N_n*N_j$ real numbers. Option B is to precalculate all P_m^m. Option C is to continue with the present scheme, but this requires a fair bit of extra work or of extra synchronization. In any case the ϵ_m^n used in the P_m^n calculation are precalculated. Option B was implemented as a reasonable compromise.

5.4. Test results

The algorithm described above was completely specified and coded. Several short test experiments were run using a rhomboidal 15 truncation, 20 latitudes in each hemisphere and 48 longitudes. The correctness of the results was verified by monitoring the global rms value of the variable in spectral and gridpoint space. In most of the runs we performed three complete sets of transforms or time steps. Because of the Robert time filter, the forecast time or last filtered time is at the end of the second time step. Differences between initial and final conditions should be zero since only transforms were calculated. Differences were of the size to be expected due to roundoff errors and appear to grow linearly with the number of time steps. For example, at the end of 11 time steps, the rms difference had grown to be about 11/3 times larger than the rms difference at the end of three time steps. Round off errors for all runs of the same forecast length were identical, independent of the number of processors used, demonstrating that our algorithm maintains a set order of arithmetic operations.

Timing results are shown in the table 4. These numbers are the wall clock time used to perform the multitasked portion of the code. Activity on the Sequent, when these runs were performed was not significant. We ignored the time due to the initial and final segments because in any realistic application their contribution would be insignificant since many time steps would be used. (Perhaps 100 per 24 hours of forecast.) Similarly, the time used to fork the child processes was not included.

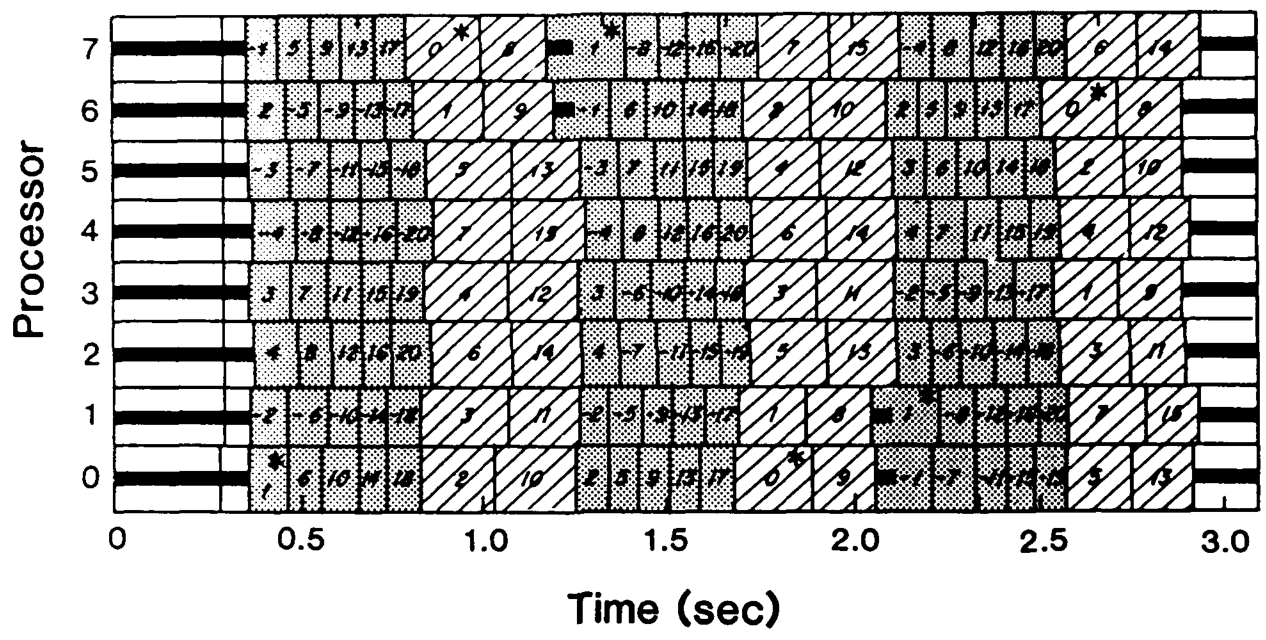
Table 4. Latitude wavenumber timings

Number of Processor	Time (sec)	Speed Up	Efficiency (%)
1	18.93	1.0	100
8	2.79	6.8	85
9	2.72	7.0	77

Here efficiency is defined as the speed up factor divided by the number of processors. Because there are 40 latitude tasks and 16 wavenumber tasks per cycle the 8 processor configuration is well balanced. Adding the ninth processor does not improve the overall time very much since additional time is wasted waiting for synchronization. These results seen in Fig. 4 which shows the activity for each processor for the 8 and 9 processor runs as a function of time. In this figure, latitude tasks are dotted and wavenumber tasks are hatched. Nonproductive time is marked by the solid horizontal bars. Nonproductive time may be characterized as start up time, synchronization time, and completion time. Wavenumber task 0 and latitude task 1 are marked by an asterisk in the figure. These tasks also compute the rms values used to monitor the correctness of the calculation. Clearly different processors perform different tasks for different time steps. This demonstrates the dynamic nature of the task allocation algorithm.

6. Discussion and extensions to other architectures

The approach of Section 5 could be extended in several ways for both shared and distributed MIMD architectures. For the Sequent and other tightly coupled machines where local and shared memory are similar in speed and ease of use, the amount of communication with shared memory is of no



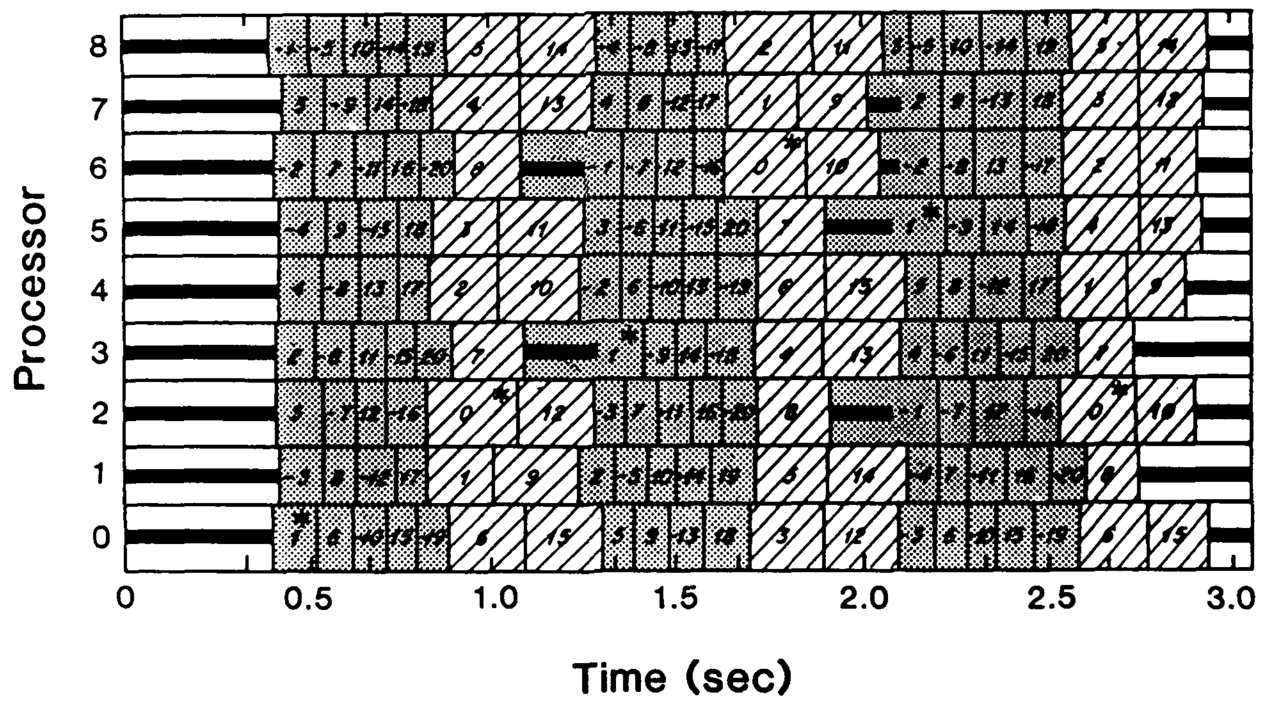


Fig. 4. Processor activity for runs of the latitude wavenumber macrotasking algorithm using eight (top) and nine (bottom) processors. See text for details.

concern. In loosely coupled systems, this traffic should be minimized. Similarly in dMIMD systems, the volume of communication between processors should be minimized. The latitude wavenumber approach greatly limits communication. If load balance can be maintained, static task allocation is desireable, since it also limits communication. Static task allocation also greatly simplifies the design of dMIMD algorithms. The ability of the latitude wavenumber algorithm developed in Phase I for dynamic task allocation would be useful in environments not dedicated solely to running the GSM. Finally, for SIMD, new approaches are needed. One approach is to assign each processor to one or a small number of latitude longitude grid points. These issues are described in turn in the following sections.

6.1. General extensions to the latitude wavenumber approach

More detailed breakdowns within the latitude and wavenumber tasks are possible and desirable to allow better load balancing and to make use of more processors for a given model resolution. With dynamic task allocation and a finer division of the algorithm it becomes possible to allow better load balancing simply because there are more tasks. With static task allocation, it may be desireable to pair latitudes and wavenumbers. This reduction in granularity may be offset by creating subtasks. A finer breakdown is possible since (1), (2), (4) and (5) may be decoupled vertically, i.e. each level may be handled separately, while (3) and (6) may be decoupled horizontally with each grid point or mode handled separately.

There is a special concern for dynamically allocating tasks for NWP models and other systems which are sensitively dependent on initial conditions; that is, if the order of arithmetic operations varies from run to run, round off error will also vary and the results will be irreproducible. This may be of little consequence to operational NWP, but may be important for some research applications. However for the latitude wavenumber tasking scheme and the finer breakdowns listed above, the arithmetic operations are self contained within single tasks and subtasks. This assures reproducibility and almost entirely eliminates the need to lock storage. Data integrity is maintained by synchronization mechanisms which insure that all preceeding tasks are complete before their results are used.

Microtasking within tasks and subtasks might be used to achieve load balance in cases for which load balance cannot be achieved at the macrotasking level. Many compilers will provide microtasking at the loop level either automatically or according to directives.

In the latitude wavenumber algorithm we make no special treatment for the FFTs. The breakdown we have outlined is rather coarse. Each of the processors in the above discussion might actually be a group of processors and multiprocessor FFT methods as described by Briggs et al. (1987) might be implemented within each group. We suggest using this approach within an SIMD algorithm below. Briggs et al. however did not consider communication costs in their analysis. Such costs are significant for many architectures and are a major concern for loosely coupled architectures (Saltz et al., 1987). For global spectral models all FFTs are the same relatively modest length (100 - 1000). Therefore, for operational NWP on a coarse grained machine, it may be economically efficient to implement the FFTs in hardware within each processor.

Within the basic approach of Section 5 the transforms between grid point and spectral space are recognizable as transpose splitting operations (McBryan and Van de Velde, 1987). According to McBryan and Van de Velde algorithms for problems in this class may be relatively portable if appropriate library routines are used.

6.2. Analysis of communication load in GSM algorithms

For many types of multiprocessors communication through shared memory or communication channels may easily become a bottleneck. In this regard the latitude wavenumber tasking approach is superior to the latitude only tasking approach. The analysis of the communication load in these two approaches follows. To simplify the analysis we treat the null model of simple spectral transformations introduced in Section 5.

The principal shared data for latitude tasking are the spectral coefficients at three time levels. For each latitude, step (1) fetches all spectral coefficients at the central time and step (5) fetches and stores all spectral coefficients for the tendencies. Step (6), the time stepping, requires all previous, central and tendency spectral coefficients to be fetched and all new time level spectral coefficients to be stored. In all $(3{*}N_j + 4)N_n{*}N_m$ memory accesses are required per time step. This could be reduced considerably if several latitudes were done by each processor and if each processor had enough storage to keep a copy of the central and tendency spectral coefficients.

If communication restrictions were especially severe, then memory accesses could be reduced further by reproducing all three time levels of spectral coefficients and the calculation of step (6) across all processors (J. Sela, pers. comm., 1988). In this approach each of P processors calculates the contribution to the spectral tendencies for $P{*}N_j$ latitudes. These contributions are then added up in shared memory and fetched by each processor. The total memory access is only $3{*}P{*}N_n{*}N_m$.

This approach also makes sense for dMIMD architectures in which the processors are arranged in a hypercube geometry. In this case each of the dimensions of the hypercube is collapsed in turn. The final result must then be fanned out. The total number of communication transfers would be only about $2{*}P{*}N_n{*}N_m$. The final fan out can be eliminated if each processor accumulates its own sum but then the total communication transfers would be roughly $d{*}P{*}N_n{*}N_m$, where d is the dimension of the hypercube. In spite of its heavier communication burden, this last approach should be most efficient, since the communication load is evenly balanced and always between nearest neighbors.

For the latitude wavenumber tasking approach the principal shared data are the past and central spectral coefficients and the central and tendency Fourier coefficients. At each of the interfaces between steps (1) and (2) and between steps (4) and (5) a complete set of fourier data is stored and fetched. Additionally, during the step (6), the past and central spectral coefficients are fetched and the new central spectral coefficients are stored. Each value is fetched and/or stored exactly once. As a result the total number of shared memory accesses per time step is only $4*N_m*N_j + 3*N_m*N_n$. With static task allocation, the spectral coefficients would not need to be shared in the latitude wavenumber tasking approach and the total number of accesses would be only $4*N_m*N_j$. The corresponding value for latitude tasking was roughly $(3/4)*N_n$ larger. The latitude wavenumber tasking approach may also be used with dMIMD architecture as described below. With static task allocation, the total number of communication transfers would be only $2*N_m*N_j$. Note that the communications load for the latitude wavenumber approach is independent of the number of processors.

6.3. Static task allocation for MIMD architectures

The latitude wavenumber approach is appropriate for both shared and distributed MIMD architectures. In both dMIMD and loosely coupled sMIMD machines it is important to limit communication.

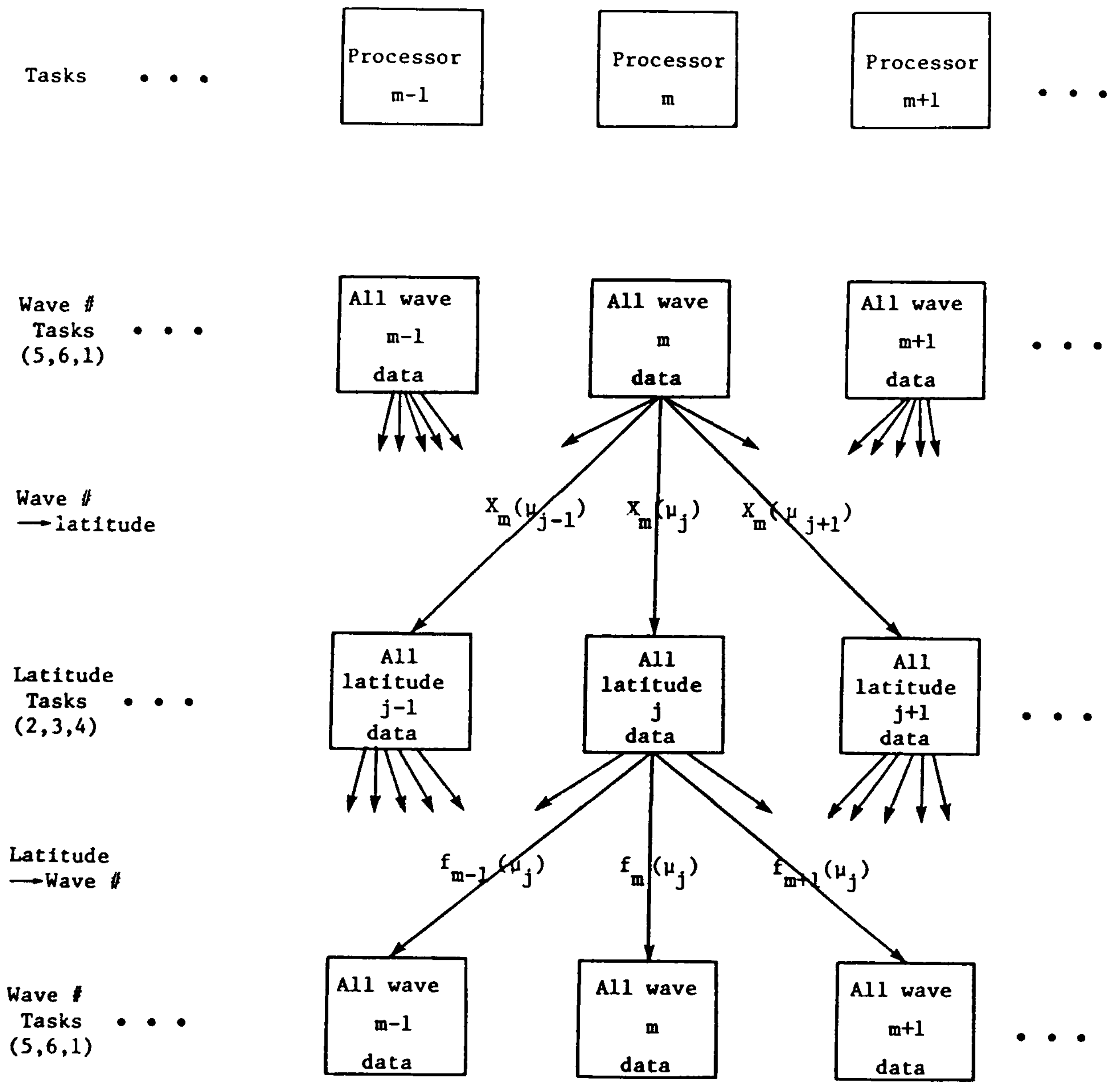
When shared memory is slow relative to local memory it is important to limit communication through shared memory. Static decomposition would be favored in such cases since certain constant data and previous data could be fixed to the appropriate processor. For example if one processor always handles wavenumber task m then only this processor requires the spectral coefficients associated with m. For similar reasons, static decomposition would be desireable for dMIMD systems.

With static task allocation the dMIMD algorithm would look very much like the sMIMD algorithm. Fig. 5 describes the basic dMIMD algorithm. Rather than accessed via shared memory, the Fourier data are actively communicated among the processors. To maintain data integrity each data segment communicated should include a valid time associated with the data.

In addition for dMIMD architecture, static decomposition would fix the communication data paths, that is we would know where to send the data at synchronization points, and eliminate the need for a dynamic task allocation mechanism. For dMIMD architectures, dynamic task allocation would require a means of passing control among the processors. In addition a table

Figure 5

Data Flow Configuration



Note: "m data" or "j data" may be a single wave #/latitude or a group of wave #s/latitudes.

describing the current and past task for each processor would have to be maintained in all processors to describe the changing communication paths for the data. The current calculation in any processor would have to be interruptible when a new process begins on some other processor. Posting new information describing the state of the processor which has just initiated a task in the task table would implicitly request data from many other processors. This request for data from previous tasks should be handled immediately so that the new task is not kept waiting.

For static decomposition pairing up latitudes, one polar and one equatorial might be advantageous. Tropical physics is more complex (mostly due to convection), while at the poles we might limit the number of grid points and Fourier truncation. We might call this approach folding the latitude domain. Similarly, a triangular truncation might be implemented by folding the wavenumber domain. This would pair wavenumbers 0 and N_m, 1 and N_m-1, etc., yielding the same number of spectral modes for each processor.

6.4. Dynamic task allocation for a loosely coupled sMIMD multitasking environment

A number of very powerful loosely coupled sMIMD machines exist or are planned. By loosely coupled, we mean that these machines have local memory which is much faster and/or easier to access than their larger shared memory. In operation, these machines may have to perform many functions at once. Any collection of computers or powerful workstation connected by a network with a large shared memory device might also be considered a loosely coupled sMIMD machine. Algorithms developed on the Sequent during Phase I could easily be improved and ported to such machines.

Static decomposition as described in the previous section does not take advantage of the flexibility of the latitude wavenumber algorithm. Static balancing is bound to fail if the processors are also performing other functions simultaneously. Our dynamic task allocation algorithm adjusts to the external load. With a robust file server/network this algorithm might be implemented in background at low priority on a collection of work stations. The computational load would be automatically distributed to idle workstations. One concern here: What if the slowest workstation takes on the last latitude task. A more robust algorithm would keep track of the wall clock time each processor required for each task and would cause each processor to decline a task if it appears that performing the task would actually slow the computation.

In the case of a loosely coupled architecture where accessing shared memory actually required network I/O to a file, or of an architecture which does not insure data integrity, it might not be desireable to equivalence the Fourier tendencies and variables. In this case we should arrange storage so that writing (for each task) is to contiguous storage, but reading roams all over the storage (see Fig. 3).

6.5. An SIMD algorithm

SIMD architectures require a fresh look at the problem. SIMD machines include the Goodyear MPP, Thinking Machines Connection Machine and the Active Memory Technology DAP. With these machines each grid point (i.e. latitude longitude location on the transform grid) might be allocated each own processor. For example, if the communication channels allow it, we might implement an SIMD GSM using a topology consisting of a ring of hypercubes. To describe this approach we let each x represent a processor. Then in grid point space each processor corresponds to a single latitude longitude point:

```
x x x x x x x   ...   x x

x x x x x x x   ...   x x
                              Lat. |
x x x x x x x   ...   x x          |
                                   V
          .
          .
          .

x x x x x x x   ...   x x

     Long. --->
```

The communication connections between grid points however are not the usual 2D grid. Along each latitude circle the points are connected by a hyper-cube. This allows the grid point to Fourier and Fourier to grid point transforms to be conducted efficiently if the number of longitudes is a power of two. In Fourier space each processor corresponds to either the real or imaginary part of a single Fourier coefficient at a single latitude:

```
x x x x x x x   ...   x x

x x x x x x x   ...   x x
                              Lat. |
x x x x x x x   ...   x x              |
                                   V
          .
          .
          .

x x x x x x x   ...   x x

     Wavenumber --->
```

Truncation demands that the rightmost approximately one third of the Fourier coefficients be ignored or set to zero. That is about a third of the processors will be idled in Fourier space. The processors for each wavenumber communicate in a ring topology. The transformations from Fourier to spherical harmonic space and from spherical harmonic space to Fourier space are then accomplished marching either the Fourier data or the spherical harmonic data around the ring holding the other data in place.

(This might be termed a systolic dot product.) If the spherical harmonic data is fixed then each processor has all the P_m^n for that n at all latitudes; otherwise each processor has the P_m^n for that latitude for all n. In spectral space each processor corresponds to a single spectral mode:

```
x x x x x x x   ...   x x

x x x x x x x   ...   x x
                               Total | Wavenumber
x x x x x x x   ...   x x            |
                                     V
      .
      .
      .

x x x x x x x   ...   x x

    Wavenumber ——>
```

Splitting the real and imaginary parts of the Fourier or spectral variables between pairs of processors will not create great problems. All operations in Fourier and spectral space are linear and most involve only arithmetic with real constants. In Fourier space, multiplies by i, the imaginary unit, will require some data interchanges with nearest neighbors. Otherwise all operations on the real and imaginary parts of the data are the same.

The number of active total wavenumbers retained in spectral space is only about 2/5 (or 2/3) of the number of latitudes for rhomboidal (or triangular) truncation. Thus only about 27% (or 44%) of the processors would be active in spectral space. The low percentage of active processors in spectral space may not be objectionable since most calculations are in grid point space.

Alternatively, other approaches which make better use of the available processors in spectral space are possible. For example, if each processor has enough memory one could allocate 2 or more grid points to a single processor. Some particular advantages would be obtained if the latitude domain were folded such that Northern and Southern Hemisphere points were paired. This would save some arithmetic and, for rhomboidal truncation, double the number of active processors in spectral space. If sufficient memory were available, additional folding, of the wavenumber domain in half and the gridpoint domain in thirds would raise the active percentage of processors to nearly 100% in Fourier space and 80% in spectral space. This would require 6 grid points per processor. A similar arrangement without the North South pairing would allow a triangular truncation model in which the active percentage of processors would be nearly 100% in Fourier space and 67% in spectral space.

7. Summary and concluding remarks

We have made a preliminary exploration of the uses of multiprocessing computers for large scale NWP using spectral models. In general if global communication between processors is relatively fast and easy, then implementing spectral models is feasible. We find that:

- Some operational centers are already using 2 or 4 processors.

- The basic GSM in use today can easily be macrotasked over latitude and microtasked within time stepping. Machines with either few processors or with tightly coupled large shared memory could be efficiently used this way.

- An adiabatic version of the GSM using latitude macrotasking was implemented on the Sequent. Efficiencies of 98% were achieved, provided the number of processors allowed an even division of the latitude tasks.

- The GSM can be recast in terms of latitude and wavenumber tasks. This approach has a number of advantages: The entire algorithm is macrotasked. Only a handful of crucial pointers need to be locked with this algorithm. The Legendre calculations are localized so that arithmetic always follows the same ordering and all results (including roundoff errors) are exactly reproducible. The communication load is small and independent of the number of processors.

- The latitude wavenumber tasking scheme was implemented and tested on the Sequent, a sMIMD device. The efficiency of our implementation is sensitive to the ratios of the number of tasks of different types to the number of processors. Microtasking within the tasks might be used to even out the load when the macrotasking division of work is not balanced.

- We believe the latitude wavenumber tasking scheme could be easily extended and applied to large sMIMD machines. It is also a good starting point for dMIMD machines, with hypercube topology.

- The potential of SIMD machines with a hypercube topology is huge. For this class of machine we would use one processor for each horizontal grid point. At least one spectral algorithm (of the Navier-Stokes equations) has been implemented successfully on the Connection Machine (Tomboulian et al., 1988).

Finally, we note that one of the goals of the SBIR program is commercialization. There are many opportunities for commercialization in the field of weather forecasting: there is no doubt that a large market exists for a better forecast. Over the last decade, it has been demonstrated especially by the European Center for Medium-Range Weather Forecasting (ECMWF) that given greater resources for operations and research, it is possible to make better weather predictions. At the same time, requirements for more detailed and accurate forecasts have increased, in part because as forecast skill grows, public expectations also grow. The number and uses of weather predictions will certainly grow as the reliability of the forecasts increase. Because government agencies involved in weather prediction have limited mandates and cannot market their products, the opportunity for commercialization of weather prediction has been growing, and this growth will continue. Further, as the cost of communications and computers decrease, private weather forecasting becomes more feasible. The results of the current project demonstrate the usefulness of multiprocessing computers for lowering the computational cost of NWP.

8. References

Brenner, S., C.-H. Yang, and S.Y.K. Yee, 1982: The AFGL spectral model of the moist global atmosphere: Documentation of the baseline version. AFGL-TR-82-0393, AFGL, Meteorology Division, Hanscom AFB, Bedford, MA 01731, 65 pp.

Briggs, W.L., L.B. Hart, R.A. Sweet and A. O'Gallagher, 1987: Multiprocessor FFT methods, SIAM J. Sci. Stat. Comput., 8, s27-s42.

Bourke, W., B. McAvaney, K. Puri, and R. Thurling, 1977: Global modeling of atmospheric flow by spectral methods. In General Circulation Models of the Atmosphere. Methods in Computational Physics, Vol. 17, edited by J. Chang, Academic Press, New York, pp. 267-324.

Dent, D., 1988: The ECMWF model: Past, present and future. In Multiprocessing in Meteorological Models, edited by G.-R. Hoffmann and D. F. Snelling, Springer-Verlag, pp. 369-381.

Fox, G.C., and S.W. Otto, 1984: Algorithms for concurrent processors. Physics Today, 37, 53-59.

Gibson, J.K., 1985: A production multi-tasking numerical prediction model. Comp. Phys. Communications, 37, 317-327.

Gustafson, J. L., G. R. Montry and R. E. Benner, 1988: Development of parallel methods for a 1024-processor hypercube. SIAM J. Sci. Stat. Compute., 9, 609-638.

Hoffmann, G.-R. and D. F. Snelling, 1988: A comparative study of the ECMWF weather model on several multiprocessor architectures. In Multiprocessing in Meteorological Models, edited by G.-R. Hoffmann and D. F. Snelling, Springer-Verlag, pp. 419-432.

Karp, A. H. and R. B. Babb, 1988: A comparison of 12 parallel Fortran dialects. IEEE Software, September, 52-67.

Kung, H.T., 1980: The structure of parallel algorithms. Advances in Computers, Vol. 19, Academic Press, New York, pp. 65-112.

Kuck, D. J., E. S. Davidson, D. H. Lawrie and A. H. Sameh, 1986: Parallel supercomputing today and the Cedar approach. Sci., 231, 967-974.

McBryan, O.A., and E. F. Van De Velde, 1987Hypercube algorithms an implementations, SIAM J. Sci. Stat. Comput., 8, s227-s287.

Ranka, S., Y. Won and S. Sahni, 1988: Programming a hypercube multicomputer. IEEE Software, September, 69-77.

Saltz, J.H., V.K. Naik and D.M. Nicol, 1987: Reduction of the effects of the communication delays in scientific algorithms on message passing MIMD architectures, SIAM J. Sci. Stat. Comput., 8, s118-s134.

Sela, J.G., 1980: Spectral modeling at the National Meteorological Center. Mon. Weather Rev., 108, 1279-1292.

Shuman, F.G., 1982: Numerical weather prediction. In The Federal Plan for Meteorological Services and Supporting Research, Fiscal Year 1983, Department of Commerce, Washington, D.C., pp. 1-1 - 1-13.

Snelling, D. F., 1988: Tools for assessing multiprocessing. In Multiprocessing in Meteorological Models, edited by G.-R. Hoffmann and D. F. Snelling, Springer-Verlag, pp. 237-253.

Tomboulian, S., C. Streett and M. Macaraeg, 1988: Spectral solution of the incompressible Navier-Stokes equations on the Connection Machine2. To appear in Super Computing 88.

White, P. W. and R. L. Wiley, 1988: U.K. Meteorological Offices's plans for using multiprocessor systems. In Multiprocessing in Meteorological Models, edited by G.-R. Hoffmann and D. F. Snelling, Springer-Verlag, pp. 215-224.

WMO, 1985: Report of the Seminar on Progress in Numerical Modelling and the Understanding of Predictability as a Result of the Global Weather Experiment - Sigtuna, Sweden, October, 1984. WMO/TD - No. 33, WMO, Case postale No. 5, CH-1211 Geneva 20, Switzerland.

Appendix. Acronyms

AFGL	Air Force Geophysics Laboratory [Hanscom AFB, MA 01731]
AFGWC	Air Force Global Weather Central [Offut AFB, NE]
AFWL	Air Force Weapons Laboratory
ANSI	American National Standards Institute
CCM	Community Climate Model
DAP	distributed array processor
ECMWF	European Center for Medium Range Weather Forecasts [Shinfield Park, Reading, Berkshire RG2 9AX, England]
FFT	fast fourier transform
GARP	Global Atmospheric Research Program
GCM	general circulation model
GFDL	Geophysical Fluid Dynamics Laboratory (NOAA) [Princeton University, Princeton, NJ 08540]
GLA	GSFC Laboratory for Atmospheres (NASA) [Greenblet, MD 29771]
GSM	global spectral model
MIMD	Multiple Instruction Multiple Data
MPP	massively parallel processor
NCAR	National Center for Atmospheric Research [Boulder, CO 80307]
NEPRF	Naval Environmental Prediction Research Facility [Monterey, CA]
NMC	National Meteorological Center (NOAA NWS) [Washington, D.C. 20233]
NWP	numerical weather prediction
SBIR	Small Business Innovative Research
SIMD	Single Instruction Multiple Data
SST	Sea Surface Temperature
UCLA	University of California at Los Angeles [Los Angeles, CA]
WMO	World Meteorological Organization [Geneva]