AD-752 569

GLOBAL WEATHER MODELING WITH VECTOR SPHERICAL HARMONICS

Sven Bjorklund

IBM Federal Systems Division

Prepared for:

Advanced Research Projects Agency

11 October 1972

DISTRIBUTED BY:

NTIS

National Technical Information Service
U. S. DEPARTMENT OF COMMERCE
5285 Port Royal Road, Springfield Va. 22151

BEST
AVAILABLE COPY

Global Weather Modeling with Vector Spherical Harmonics

Report No. 1

AD752569

IBM

Contract No. - DAHC04-72-C-0020

ARPA Order Number - 2089

Program Code Number - 62706D

Name of Contractor - International Business Machines Corporation,
Federal Systems Division

Effective Date of Contract - 15 March 1972

Expiration Date - 9/14/72

Amount of Contract - $48,627

Principal Investigator and Phone Number: Dr. Sven Bjorklund, (301) 840-7253

Project Scientist and Phone Number Dr. Sven Bjorklund, (301) 840-7253

Title of Work: Global Weather Modeling with Vector Spherical Harmonics

Date - 11/10/72

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 2089

D D C
RECEIVED
B

Approved for public release;
distribution unlimited

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by U.S. Army Research Office-Durham under Contract No. DAHC04-72-C-0020.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Advanced Research Projects Agency or the U. S. Government.

I

TABLE OF CONTENTS

Section	
	Summary
1	Review of Vector Spherical Harmonics
2	Weather Equations in Vector Spherical Harmonics
3	Pressure Coordinates
4	The Transform Method
5	The Mintz-Arakawa Model in Vector Spherical Harmonics
6	Angular Momentum
7	Program Design

SUMMARY

This report presents the current status of work under ARPA order 2089, in the weather models, with a view toward climatological application.

Vector Spherical Harmonics are reviewed in the context of solving partial differential equations in spherical coordinates. The basic equations governing global weather models are presented. Vector Spherical Harmonics are applied to these equations in spherical coordinates and it is shown how separation of variables is achieved in the sense that angular dependence is eliminated.

Angular momentum is expressed in terms of Vector Spherical Harmonics. The equation for the conservation of angular momentum takes on a very simple and elegant form in this representation.

The changes of variable necessary to introduce pressure coordinates and σ-coordinates are carried out and the consequences of these changes discussed. The results are used to formulate the Mintz-Arakawa model in terms of Vector Spherical Harmonics in detail.

The transform method which will be used in the numerical solution of the model is reviewed.

The functional specifications for the computer programs to implement the above results are included, as well as a discussion on the proposed implementation methodology.

How this transform method applies to Vector Spherical Harmonics is shown.

Introduction

This report presents the main results of the first six months work supported by ARPA Contract No. DAHC 04-72-C-0020 on global weather modeling using vector spherical harmonics (vsh). The work to date has been mainly mathematical. The primitive equations have been formulated in terms of vsh. The non-linear terms can all be expressed in terms of sums of products of various vector coupling coefficients. These are being developed, but are not reported on here. The transform method to be used for the computation does not require the computation of product terms in vsh. Thus the formulation in this report suffices for the transform approach. However, these explicit expansions are important for checking the transform method results and for analytic work. We expect to have this formulation ready for the next six month report.

The Mintz-Arakawa model has been formulated in terms of σ coordinates and a tentative approach using powers of σ has been developed. Work on correlating actual data with vsh is in progress. The nature of the radial dependence of the spectral coefficients is being investigated in this way. A formulation using vsh with pressure coordinates is presented. A brief discussion of work on angular momentum in progress is also shown.

Although the program design for the computer calculations has been done with other funds and is not necessary for fulfilling the contract commitments, we thought it of sufficient interest to be included in this report. It will enhance the work on the contract and also show where the present and future programming fits into this general schemes.

SECTION 1

Review of Vector Spherical Harmonics

The Vector Spherical Harmonics[1], $T^M_{JL}(\theta,\varphi)$ form a complete orthonormal set of vector functions in θ and φ. They are defined in terms of the scalar spherical harmonics $Y^M_L(\theta,\varphi)$ in the following way

$$T^M_{JL}(\theta,\varphi) \equiv \sum_{\mu=-1}^{+1} \langle LM-\mu\ 1\ \mu \mid JM \rangle\ Y^{M-\mu}_L(\theta,\varphi)\hat{e}_\mu \qquad (1)$$

where the functions $Y^M_L(\theta,\varphi)$ are defined by

$$Y^M_L(\theta\varphi) = \left[\frac{(2L+1)(L-M)!}{4\pi(L+M)!}\right]^{1/2} P^M_L(\cos\theta)\, e^{iM\varphi} \qquad (2)$$

They form a complete set of scalar functions in θ and φ, and satisfy the orthonormality condition

$$\int_0^{2\pi} d\phi \int_0^{\pi} \left[Y^M_L(\theta,\varphi)\right]^* Y^{M'}_{L'}(\theta,\varphi) \sin\theta\, d\theta = \delta_{MM'}\delta_{LL'} \qquad (3)$$

The functions $P^M_L(\cos\theta)$ are associated Lengendre functions defined by

$$P^M_L(x) = \frac{1}{2^L L!}(1-x^2)^{M/2}\frac{d^{L+M}}{dx^{L+M}}(x^2-1)^L, \quad |M| \leq L \qquad (4)$$

The vectors e are defined by

$$\hat{e}_{-1} \equiv \hat{e}_- \equiv \frac{1}{\sqrt{2}}(\hat{i}-i\hat{j});\ \hat{e}_0 \equiv \hat{k};\ \hat{e}_{+1} \equiv \hat{e}_+ \equiv -\frac{1}{\sqrt{2}}(\hat{i}+i\hat{j}) \qquad (5)$$

The symbols $\langle LM\ L'M' \mid L''M'' \rangle$

are Clebsch-Gordan coefficients and satisfy the conditions

$$M+M' = M'' \qquad |L-L'| \leq L'' \leq L+L'$$

The indices J are positive integers or zero with J satisfying

$$|L-1| \leq J \leq L+1$$

and M is a positive or negative integer or zero such that

$$L \geq |M|$$

The explicit form of the Vector Spherical Harmonics is

$$T^{M}_{L,L-1} = \left[\frac{(L-M-1)(L-M)}{2L(2L-1)}\right]^{1/2} Y^{M+1}_{L-1}\,\hat{e}_{-} + \left[\frac{(L-M)(L+M)}{L(2L-1)}\right]^{1/2} Y^{M}_{L-1}\,\hat{e}_{0} + \left[\frac{(L+M-1)(L+M)}{2L(2L-1)}\right]^{1/2} Y^{M-1}_{L-1}\,\hat{e}_{+} \tag{6}$$

$$T^{M}_{LL} = \left[\frac{(L-M)(L+M+1)}{2L(L+1)}\right]^{1/2} Y^{M+1}_{L}\,\hat{e}_{-} + \left[\frac{M}{L(L+1)}\right]^{1/2} Y^{M}_{L}\,\hat{e}_{0} - \left[\frac{(L+M)(L-M+1)}{2L(L+1)}\right]^{1/2} Y^{M-1}_{L}\,\hat{e}_{+} \tag{7}$$

$$T^{M}_{LL+1} = \left[\frac{(L+M+1)(L+M+2)}{2(L+1)(2L+3)}\right]^{1/2} Y^{M+1}_{L+1}\,\hat{e}_{-} - \left[\frac{(L-M+1)(L+M+1)}{(2L+3)(L+1)}\right]^{1/2} Y^{M}_{L+1}\,\hat{e}_{0} + \left[\frac{(L-M+1)(L-M+2)}{(2L+1)(2L+3)}\right]^{1/2} Y^{M-1}_{L+1}\,\hat{e}_{+} \tag{8}$$

The orthonormality of the T^{M}_{JL}'s is expressed in the following equation

$$\int_0^{2\pi}\int_0^{\pi} T^{M*}_{JL}(\theta,\phi)\cdot T^{M'}_{J'L'}(\theta,\phi)\,\sin\theta\,d\theta\,d\varphi = \delta_{MM'}\,\delta_{JJ'}\,\delta_{LL'}\,. \tag{9}$$

An important equation for applications is

$$\hat{e}_\mu Y^{M-\mu}_{L}(\theta,\varphi) = \sum_{J=L-1}^{L+1} \langle L\,M-\mu\,1\,\mu\,|J\,M\rangle\, T^{M}_{JL}(\theta,\varphi) \tag{10}$$

The vector spherical harmonics have particularly convenient differential expressions. One can show[2] that the ordinary differential operations of vector calculus, namely, gradient divergence and curl can be expressed in the following form

$$\nabla\left[f(r,t)Y^{M}_{L}(\theta,\varphi)\right] = -\alpha_L D_{-L} f(r,t)\,T^{M}_{LL+1} + \beta_L D_{L+1} f(r,t)\,T^{M}_{LL-1} \tag{11}$$

$$\nabla\cdot\left[f(r,t)\,T^{M}_{LL-1}(\theta,\varphi)\right] = \beta_L D_{1-L} f(r,t)\,Y^{M}_{L}(\theta,\varphi) \tag{12}$$

$$\nabla\cdot\left[f(r,t)\,T^{M}_{LL}(\theta,\varphi)\right] = 0 \tag{13}$$

$$\nabla\cdot\left[f(r,t)\,T^{M}_{LL+1}(\theta,\varphi)\right] = -\alpha_L D_{L+2} f(r,t)\,Y^{M}_{L}(\theta,\varphi) \tag{14}$$

$$\nabla\times\left[f(r,t)\,T^{M}_{LL-1}(\theta,\varphi)\right] = i\alpha_L D_{1-L} f(r,t)\,T^{M}_{LL}(\theta,\varphi) \tag{15}$$

$$\nabla\times\left[f(r,t)\,T^{M}_{LL}(\theta,\varphi)\right] = i\alpha_L D_{L+1} f(r,t)\,T^{M}_{LL-1} + \beta_L D_{-L} f(r,t)\,T^{M}_{LL+1} \tag{16}$$

$$\nabla\times\left[f(r,t)\,T^{M}_{LL+1}(\theta,\varphi)\right] = i\beta_L D_{L+2} f(r,t)\,T^{M}_{LL}(\theta,\varphi) \tag{17}$$

$$\alpha_L = \left(\frac{L+1}{2L+1}\right)^{1/2} \qquad D_L = \left(\frac{\partial}{\partial r} + \frac{L}{r}\right) \tag{18), (19}$$

$$\beta_L = \left(\frac{L}{2L+1}\right)^{1/2} \tag{20}$$

One can form another set of vector spherical harmonics which we call $A_L^M(\theta,\varphi)$, $B_L^M(\theta,\varphi)$ and $C_L^M(\theta,\varphi)$ defined by

$$A_L^M(\theta,\varphi) = \beta_L T^M_{LL-1}(\theta,\varphi) - \alpha_L T^M_{LL+1}(\theta,\varphi) = \hat{e}_r Y_L^M(\theta,\varphi) \tag{21}$$

$$B_L^M(\theta,\varphi) = \alpha_L T^M_{LL-1}(\theta,\varphi) + \beta_L T^M_{LL+1}(\theta,\varphi) = \frac{r}{\gamma_L} \nabla Y_L^M(\theta,\varphi) \tag{22}$$

$$C_L^M(\theta,\varphi) = T^M_{LL}(\theta,\varphi) = -\frac{i}{\gamma_L}\left(\bar{r} \times \nabla Y_L^M(\theta,\varphi)\right) \tag{23}$$

or

$$B_L^M(\theta,\varphi) = \frac{1}{\gamma_L}\left[\hat{e}_\theta \frac{\partial}{\partial\theta} Y_L^M(\theta,\varphi) + \hat{e}_\varphi \frac{1}{\sin\theta}\frac{\partial}{\partial\varphi} Y_L^M(\theta,\varphi)\right] \tag{24}$$

$$C_L^M(\theta,\varphi) = \frac{1}{\gamma_L}\left[\hat{e}_\varphi \frac{\partial}{\partial\theta} Y_L^M(\theta,\varphi) - \hat{e}_\theta \frac{1}{\sin\theta}\frac{\partial}{\partial\varphi} Y_L^M(\theta,\varphi)\right] \tag{25}$$

where α_L, β_L are as defined earlier, and

$$\gamma_L = \sqrt{L(L+1)} \tag{26}$$

Reproduced from best available copy.

Then,

$$\nabla \cdot f(r,t) A_L^M = D_2 f(r,t) Y_L^M \tag{27}$$

$$\nabla \cdot f(r,t) B_L^M = -\frac{\gamma_L}{r} f(r,t) Y_L^M \tag{28}$$

$$\nabla \cdot f(r,t) C_L^M = 0 \tag{29}$$

$$\nabla \times f(r,t) A_L^M = \frac{i\gamma_L}{r} f(r,t) C_L^M \tag{30}$$

$$\nabla \times f(r,t) B_L^M = i D_1 f(r,t) C_L^M \tag{31}$$

$$\nabla \times f(r,t) C_L^M = \frac{i\gamma_L}{r} f(r,t) A_L^M + i D_1 f(r,t) B_L^M \tag{32}$$

$$\nabla f(r,t) Y_L^M = D_0 f(r,t) A_L^M + \frac{\gamma_L}{r} f(r,t) B_L^M \tag{33}$$

These functions are orthonormal in the function sense and also in the vector sense, i.e. A_L^M points in the radial direction B_L^M and C_L^M are tangential to a sphere and orthogonal to each other in the vector sense for M=M' and L=L'. It is much easier to visualize this set of functions in physical space than the original functions $T_{JL}^M(\theta, \varphi)$.

Non-linear terms can be dealt with through the application of the following equations

$$Y_L^M Y_{L'}^{M'} = \sum_{L''=|L-L'|}^{L+L'} \left[\frac{(2L+1)(2L'+1)}{4\pi(2L''+1)}\right]^{1/2} \langle L0L'0|L''0\rangle \langle LML'M'|L''M+M'\rangle Y_{L''}^{M+M'} \tag{34}$$

and

$$\partial_\mu\left[f(r)Y_L^M(\theta,\varphi)\right] = -(-1)^\mu\sqrt{\frac{L+1}{2L+1}}\langle L+1\,M+\mu\,1-\mu|LM\rangle Y_{L+1}^{M+\mu}(\theta,\varphi)\,D_{-L}f(r)$$
$$+(-1)^\mu\sqrt{\frac{L}{2L+1}}\langle L-1\,M+\mu\,1-\mu|LM\rangle Y_{L-1}^{M+\mu}(\theta,\varphi)\,D_{L+1}f(r) \tag{35}$$

where

$$\partial_{-1} = \frac{1}{\sqrt{2}}\left(\frac{\partial}{\partial x} - i\frac{\partial}{\partial y}\right);\quad \partial_0 = \frac{\partial}{\partial z};\quad \partial_{+1} = -\frac{1}{\sqrt{2}}\left(\frac{\partial}{\partial x} + i\frac{\partial}{\partial y}\right) \tag{36}$$

The details of the treatment of the non-linear terms will be shown later.

1. Edmonds, A. R., Angular Momentum in Quantum Mechanics, Princeton University Press, 1957, p.83 .

2. Loc. Cit. p. 84.

SECTION 2

Weather Equations in Vector Spherical Harmonics

I. Basic Equations for Weather Predictions

The basic equations for a primitive equation model summarized below:

1. Equation of Motion:

$$\frac{\partial \vec{V}}{\partial t} + (\vec{V}\cdot\nabla)\vec{V} + 2\vec{\Omega}\times\vec{V} + \nabla\phi + \frac{1}{\rho}\nabla p = \eta\nabla^2\vec{V} \tag{1}$$

2. Equation of Continuity:

$$\frac{\partial \rho}{\partial t} + \nabla\cdot(\rho\vec{V}) = \frac{\partial \rho}{\partial t} + \nabla\rho\cdot\vec{V} + \rho\nabla\cdot\vec{V} = 0 \tag{2}$$

3. Thermodynamic Equation

$$\frac{1}{c_p T}\frac{dq}{dt} = \frac{1}{T}\frac{dT}{dt} + \frac{k}{p}\frac{dp}{dt} \tag{3}$$

4. Equation of state for an ideal gas:

$$p = \rho R T \tag{4}$$

The variables appearing in these equations are:

$\vec{v}$ = vector velocity of the atmosphere

p = pressure

ρ = density

T = absolute temperature

The following are given functions

φ = gravitational potential

$\frac{dq}{dt} = \dot{q}$ = Heat input to the atmosphere

The following are constants:

r = $\hat{k}\Omega$, Ω = angular velocity of the earth = $7,29 \times 10^{-5}$ /sec

R = universal gas constant

p = specific heat at constant pressure

k = R/Cp

η = dynamic viscosity

The friction term $\eta \nabla^2 \vec{v}$ is one of several possibilities and needs to be modified close to the earth's surface.

II. Method of Solution Using Vector Spherical Harmonics

The method used is to expand the scalar fields as series in scalar spherical harmonics and the vector field $\vec{v}$ in vector spherical harmonics. Either the set T^M_{LL-1}, T^M_{LL}, T^M_{LL-1} or the set

A^M_L, B^M_L, C^M_L may be used. The latter provides a more obvious physical interpretation, so we will expand v in terms of that set.

The know fields $\dot{q}$, φ are also expanded in terms of scalar spherical harmonics. It should be noted that the expansion coefficients are known.

IIa. The Equation of Continuity

We let

$$\rho = \sum_{M,L} \rho^M_L(r,t)\, Y^M_L(\theta,\varphi) \tag{5}$$

$$\vec{v} = \sum_{M,L} a^M_L(r,t)\, A^M_L(\theta,\varphi) + b^M_L(r,t)\, B^M_L(\theta,\varphi) + c^M_L(r,t)\, C^M_L(\theta,\varphi) \tag{6}$$

We write:

$$\nabla\cdot(\rho\vec{v}) = (\nabla\rho)\cdot\vec{v} + (\nabla\cdot\vec{v})\,\rho \tag{10}$$

$$\nabla\rho = \nabla\left(\sum \rho^M_L Y^M_L\right) = \sum D_0 \rho^M_L A^M_L + \frac{\gamma_L}{r}\rho^M_L B^M_L \tag{11}$$

and

$$\nabla\cdot\vec{v} = \nabla\left(\sum a^M_L A^M_L + b^M_L B^M_L + c^M_L C^M_L\right) \tag{12}$$

or

$$\nabla \cdot \vec{V} = \sum_{ML} D_2 a_L^M Y_L^M - \frac{\gamma_L}{r} b_L^M Y_L^M \tag{13}$$

We thus have

$$\nabla \rho \cdot \vec{V} = \left(\sum_{ML} D_0 \rho_L^M A_L^M + \frac{\gamma_L}{r} \rho_L^M B_L^M \right) \left(\sum_{M'L'} a_{L'}^{M'} A_{L'}^{M'} + b_{L'}^{M'} B_{L'}^{M'} + c_{L'}^{M'} C_{L'}^{M'} \right) \tag{14}$$

$$(\nabla \cdot \vec{V}) \rho = \left(\sum_{ML} \rho_L^M Y_L^M \right) \left(\sum_{M'L'} \left(D_2 a_{L'}^{M'} - \frac{\gamma_L}{r} b_{L'}^{M'} \right) Y_{L'}^{M'} \right) \tag{15}$$

These terms are attacked in the following general manner:

The products $A_L^M \cdot A_{L'}^{M'} = Y_L^M \; Y_{L'}^{M'}$, etc., may be written as follows: (The C's are numerical coefficients)

$$A_L^M \cdot A_{L'}^{M'} = Y_L^M \; Y_{L'}^{M'} = \sum_{\substack{L'', M'' \\ \mu = -1, 0, 1}} C_{AA} (L, M, L', M', \mu) \, Y_{L''}^{M''} \tag{16}$$

$$A_L^M \cdot B_{L'}^{M'} = 0 \tag{17}$$

$$B_L^M \cdot B_{L'}^{M'} = - C_L^M \cdot C_{L'}^{M'} = \sum_{\substack{L'',M'',\\ \mu=-1,0,1}} C_{BB}(L,M,L',M',\mu)\, Y_{L''}^{M''} \tag{18}$$

$$B_L^M \cdot C_{L'}^{M'} = - B_L^M C_{L'}^{M'} = \sum_{\substack{L'',M''\\ \mu=-1,0,1}} C_{BC}(L,M,L',M',\mu)\, Y_{L''}^{M''} \tag{19}$$

The nonzero terms in $\nabla\rho\cdot\vec{V}$ are then

$$D_0 \rho_L^M A_L^M \cdot a_{L'}^{M'} A_{L'}^{M'} = \sum_{L'',\ldots} (D_0 \rho_L^M a_{L'}^{M'})\, C_{AA}(L,M,L',M',\mu)\, Y_{L''}^{M''} \tag{20}$$

$$\frac{\gamma_L}{r} \rho_L^M B_L^M \cdot b_{L'}^{M'} B_{L'}^{M'} = \sum_{L'',\ldots} \frac{\gamma_L}{r} \rho_L^M b_{L'}^{M'}\, C_{BB}(L,M,L',M',\mu)\, Y_{L''}^{M''} \tag{21}$$

$$\frac{\gamma_L}{r} \rho_L^M B_L^M \cdot c_L^M C_L^M = \sum \frac{\gamma_L}{r} \rho_L^M c_L^M\, C_{BC}(L,M,L',M',\mu)\, Y_{L''}^{M''} \tag{22}$$

so that

$$\nabla\rho\cdot\vec{V} = \sum_{\substack{L,M,L',M',\\ L'',M'',}} \left[D_0 \rho_L^M a_{L'}^{M'} C_{AA} + \frac{\gamma_L}{r} \rho_L^M \left(C_{BB} b_{L'}^{M'} + C_{BC} c_{L'}^{M'}\right)\right] Y_{L''}^{M''} \tag{23}$$

The term $\rho(\nabla\vec{V})$ is handled by the summation rule for products of Y_L^M. The terms are of the form

$$\rho(\nabla\vec{V}) = \sum_{L'',\ldots} \rho_L^M \left(D_2 a_{L'}^{M'} - \frac{\gamma_L}{r} b_{L'}^{M'}\right) C_{AA}(L,M,L',M',\mu)\, Y_{L''}^{M''} \tag{24}$$

and

$$(\nabla\rho)\cdot\vec{V} = \sum_{\substack{L,M,L',M',\\ L'',M''}} \left(\rho_L^M \left(D_2 a_{L'}^{M'} - \frac{\gamma_L}{r} b_{L'}^{M'}\right) C_{AA}(L',M',L,M,\mu)\, Y_{L''}^{M''}\right. \tag{25}$$

Thus the continuity equation may be written

$$\sum_{ML} \frac{\partial \rho}{\partial t} \rho_L^M Y_L^M = \sum_{\substack{L'', M'', \\ L, M, \\ L', M'}} \left[D_0 \rho_L^M a_{L'}^{M'} C_{AA} + \frac{\gamma_L}{r} \rho_L^M \left(C_{BB} b_{L'}^{M'} + C_{BC} c_{L'}^{M'} \right) \right] Y_{L''}^{M''}$$

$$+ \sum \rho_L^M \left(D_2 a_{L'}^{M'} - \frac{\gamma_L}{r} b_{L'}^{M'} \right) C_{AA} Y_{L''}^{M''} \qquad (26)$$

The coefficients of each Y_L^M are collected and equated on each side of the above equation to yield equations in , t <u>only</u>, of the form

$$\frac{\partial \rho_{L''}^{M''}}{\partial t} = \sum_{\substack{L, L', M, M' \\ \in (\gamma_{L''}^{M''})}} D_0 \rho_L^M a_{L'}^{M'} C_{AA} + \gamma_L \rho_L^M \left(C_{BB} b_{L'}^{M'} + C_{BC} c_{L'}^{M'} \right)$$

$$+ \sum_{L', M', L, M \in (\gamma_{L''}^{M''})} \rho_L^M \left(D_2 a_{L'}^{M'} - \frac{\gamma_L}{r} b_{L'}^{M'} \right) C_{AA} \qquad (27)$$

When this is performed for all values of M and L we are left with a system of partial differential equations involving the expansion coefficients ρ_L^M, $a_{L'}^M$, $b_{L'}^M$ and C_L^M. These must be solved in conjunction with the equation of motion in the next-section, and the thermodynamic equation following that.

IIb. Equation of Motion

$$\frac{\partial \vec{v}}{\partial t} + (\vec{v}\cdot\nabla)\vec{v} + 2\vec{\Omega}\times\vec{v} + \frac{1}{\rho}\nabla p + \nabla\varphi = \eta\nabla^2\vec{v}$$

As before, let

$$v = \sum_{L,M} a_L^M \; A_L^M + b_L^M \; B_L^M + c_L^M \; C_L^M \tag{28}$$

$$\rho = \sum \rho_L^M Y_L^M \qquad \alpha = \frac{1}{\rho} = \sum \alpha_L^M Y_L^M \tag{29}$$

$$p = \sum p_L^M Y_L^M \tag{30}$$

and given

$$\varphi = \sum \varphi_L^M Y_L^M$$

i.e. the φ_L^M are known

$$\vec{\Omega} = \hat{k}\Omega$$

$\hat{k}$ is the z-direction through earth's axis of rotation.

All coefficients are functions of (r, t) only; the A, B, C, and Y_L^M's are functions of (θ, φ) only. We proceed to calculate each term in the equation of motion. It will be observed that each terms will produce one or more terms of

the form $f_A(r,t)\ A_L^M + f_B(r,t)\ B_L^M + f_C(r,t)\ C_L^M$, when substitution is complete, we can collect terms in A, B, C to obtain equations in the f_A, f_B, f_C, which will contain only the expansion coefficients of the variables and known values.

$$\nabla\varphi = \nabla \sum_{ML} \varphi_L^M Y_L^M = \sum \frac{\partial \varphi_L^M}{\partial r} A_L^M + \frac{\gamma_L}{r} C_L^M \tag{31}$$

$$\nabla p = \nabla \sum p_L^M Y_L^M = \sum \frac{\partial p_L^M}{\partial r} A_L^M + \frac{\gamma_L}{r} C_L^M \tag{32}$$

$$2\vec{\Omega} \times \vec{V} = 2\Omega\, \hat{k} \times \sum a_L^M A_L^M + b_L^M B_L^M + c_L^M C_L^M \tag{33}$$

Reproduced from best available copy.

The expressions for $\hat{K} \times A_L^M$, B_L^M, and C_L^M are

$$\hat{k} \times A_L^M = -i \left\{ \beta_{L-1} \langle LM\,1\,0 | L-1\,M \rangle C_{L-1}^M + \langle LM\,1\,0 | LM \rangle B_L^M + \alpha_{L+1} \langle LM\,1\,0 | L+1\,M \rangle C_{L+1}^M \right\}$$

$$\hat{k} \times B_L^M = -i \left\{ \frac{\alpha_L \beta_{L-1}}{\beta_L} \langle LM\,1\,0 | L-1\,M \rangle C_{L-1}^M + \langle LM\,1\,0 | LM \rangle B_L^M - \frac{\beta_L \alpha_{L+1}}{\alpha_L} \langle LM\,1\,0 | L+1\,M \rangle C_{L+1}^M \right\}$$

$$k \times C_L^M = -i \left\{ \frac{\alpha_L}{\beta_L} \langle LM\,1\,0 | L-1\,M \rangle (-\alpha_{L-1} A_{L-1}^M + \beta_{L-1} B_{L-1}^M) + \frac{1}{r_L} \langle LM\,1\,0 | LM \rangle C_L^M - \frac{\beta_L}{\alpha_L} (\beta_{L+1} A_{L+1}^M + \alpha_{L+1} B_{L+1}^M) \right\}$$

The numerical coefficients are summarized more conveniently as follows:

$$\hat{K} \times A_L^M = \mathcal{b}_A(M,L-1)\, C_{L-1}^M + \mathcal{B}_A(M,L)\, B_L^M + \mathcal{b}_A(M,L+1)\, C_{L+1}^M \tag{34}$$

$$\hat{K} \times B_L^M = \mathcal{b}_B(M,L-1)\, C_{L-1}^M + \mathcal{B}_B(M,L)\, C_L^M + \mathcal{b}_B(M,L+1)\, C_{L+1}^M \tag{35}$$

$$\hat{K} \times C_L^M = \mathcal{A}_c(M,L-1)\, A_{L-1}^M + \mathcal{B}_c(M,L-1)\, B_{L-1}^M + \mathcal{b}_c(M,L)\, C_L^M + \mathcal{A}_c(M,L+1)\, A_{L+1}^M + \mathcal{B}_c(M,L+1)\, B_{L+1}^M \tag{36}$$

Upon multiplying the first of the above equations by a_L^M, the second by b_L^M, the third by c_L^M, and collecting terms, we obtain

$$\begin{aligned}
\hat{K} \times \vec{V} = \sum_{M,L} & \left[a_L^M \mathcal{b}_A(M,L-1) + b_L^M \mathcal{b}_B(M,L-1) \right] C_{L-1}^M + \\
& \left[a_L^M \mathcal{B}_A(M,L) + b_L^M \mathcal{B}_B(M,L) \right] B_L^M + \\
& \left[c_L^M \mathcal{B}_c(M,L-1) \right] B_{L-1}^M + \\
& \left[c_L^M \mathcal{B}_c(M,L+1) \right] B_{L+1}^M + \\
& \left[c_L^M \mathcal{A}_c(M,L-1) \right] A_{L-1}^M +
\end{aligned}$$

$$[c^M_L \, \mathcal{A}_c(M, L+1) \quad] A^M_{L+1} +$$

$$[c^M_L \, b_c(M, L) \quad] C^M_L +$$

$$[a^M_L \, b_A(M, L+1) + b^M_L \, b_B(M, L+1)] \, C^M_{L+1} \tag{37}$$

From this it may be concluded that the Coriolis term $2\vec{\Omega} \times \vec{V}$ produces terms containing

$$A^M_L, \; A^M_{L-1}, \; A^M_{L+1}, \; B^M_L, \; B^M_{L-1}, \; B^M_{L+1}, \; C^M_L, \; C^M_{L-1}, \; C^M_{L+1}$$

out of each A^M_L, B^M_L, C^M_L, term in $\vec{v}$.

Next we consider $(\vec{V} \cdot \nabla)\vec{v}$. This is a very cumbersome expression. In terms of the T^M_{JL} we may write the following expression:

$$(f^M_{JL} \, T^M_{JL} \cdot \nabla) \, f^M_{J'L'} \, T^M_{J'L'} =$$

$$\mathcal{N}_1(L', J') \, f^M_{JL} \sum_{L'', J''} \left\{ \mathcal{N}_2(L', J', L, J, L'', J'') \, D_{L+1} + \mathcal{N}_3(L, J, L', J', L'', J') \, D_{L'} \right\} f^M_{J'L'} \, T^{M+M'}_{J''L''} \tag{38}$$

The $\mathcal{N}_i$ are numerical coefficients. We may now substitute for the T^M_{JL} in terms of A^M_L, B^M_L, and C^M_L to obtain the corresponding result for the A, B, C frame. Since we obtain forms of the type $a^{M'}_{L'} \, D \, b^{M''}_{L''}$ where D is some differential operator in r, we have a combinational notational problem which we circumvent by using the notation:

$$e_{1\,L}^{\,M} \equiv a_L^M \qquad\qquad E_{1L}^M \equiv A_L^M$$

$$e_{2\,L}^{\,M} \equiv b_L^M \qquad\qquad E_{2L}^M \equiv B_L^M$$

$$e_{3\,L}^{\,M} \equiv c_L^M \qquad\qquad E_{3\,L}^{\,M} \equiv C_L^M \tag{39}$$

The term $(\vec{V}\cdot\nabla)\vec{V}$ is then

$$(\vec{V}\cdot\nabla)\vec{V} = \sum_{\substack{L,M,L',M',L'',M'' \\ k',k''=1,2,3}} \mathcal{N}(L',M',\ldots)\, e_{k'L'}^{\,M}\, D(M',L')\, e_{k''L''}^{\,M''}\, E_{k''L''}^{\,M-M'} \tag{40}$$

where $\mathcal{N}$ (L', M', L'', M'', L, M) is a numerical coefficient and D is some differential operator. The principal result is that the nonlinear term has been decomposed into a sum of A_L^M, B_L^M, C_L^M terms, and the coefficients multiplying these terms involve only functions of r and derivatives of functions with respect to r.

The term $\partial\vec{V}/\partial t$ becomes simply

$$\frac{\partial\vec{V}}{\partial t} = \sum \frac{\partial a_L^M}{\partial t} A_L^M + \frac{\partial b_L^M}{\partial t} B_L^M + \frac{\partial c_L^M}{\partial t} C_L^M \tag{41}$$

Finally, we need the $\eta(\nabla^2 \vec{v})$ term. The necessary expressions are:

$$\nabla^2 a_L^M A_L^M = \left[D_0 D_2 + \left(\frac{\gamma_L}{r}\right)^2\right] a_L^M A_L^M + \frac{2\gamma_L}{r} D_1 a_L^M B_L^M \tag{42}$$

$$\nabla^2 b_L^M B_L^M = \frac{2\gamma_L}{r^2} b_L^M A_L^M + \left[D_1^2 + \left(\frac{\gamma_L}{r}\right)^2\right] b_L^M B_L^M \tag{43}$$

$$\nabla^2 c_L^M C_L^M = \left[D_1^2 + \left(\frac{\gamma_L}{r}\right)^2\right] c_L^M C_L^M \tag{44}$$

Thus, $\eta \nabla^2 v =$

$$\eta \left\{ \left[D_0 D_2 + \left(\frac{\gamma_L}{r}\right)^2\right] a_L^M + 2\frac{\gamma_L}{r^2} \right\} A_L^M + \eta \left\{ \frac{2\gamma_L}{r} D_1 a_L^M + \left[D_1^2 + \left(\frac{\gamma_L}{r}\right)^2\right] b_L^M \right\} B_L^M$$

$$+ \eta \left\{ D_1^2 + \left(\frac{\gamma_L}{r}\right)^2 \right\} c_L^M C_L^M \tag{46}$$

All the terms in the equation of motion have been worked out. However, we must evaluate

$$\frac{1}{\rho} \nabla p = \alpha \nabla p$$

which is given by

$$\left(\sum_{M'',L''} \alpha_{L''}^{M''} Y_{L''}^{M''} \right) \left(\sum_{M',L'} D_0 p_{L'}^{M'} A_{L'}^{M'} + \frac{\gamma_{L'}}{r} B_{L'}^{M'} \right) \tag{47}$$

We use the expressions

$$Y^{M''}_{L''} A^{M'}_{L'} = \sum_{\mu,\nu,L,\ell,\ldots} \mathcal{M}_{A1} A^{M}_{J} + \mathcal{M}_{A2} B^{M}_{J} + \mathcal{M}_{A3} C^{M}_{J} \tag{48}$$

where $\mathcal{M}_{i} = \mathcal{M}_{i}(L', M', L'', M'', \mu, \ell, \nu, L)$ is a numerical coefficient

$$Y^{M''}_{L''} B^{M'}_{L'} = \sum_{\mu,L,\nu,\ell} \mathcal{M}_{B1} A^{M}_{J} + \mathcal{M}_{B2} B^{M}_{J} + \mathcal{M}_{B3} C^{M}_{J} \tag{49}$$

$$Y^{M''}_{L''} C^{M'}_{L'} = \sum \mathcal{M}_{C1} A^{M}_{J} + \mathcal{M}_{C2} B^{M}_{J} + \mathcal{M}_{C3} C^{M}_{J} \tag{50}$$

which again decompose the products into sums. The general form of the expressions obtained is therefore

$$\sum_{\substack{L'', M'', \\ L', M', \\ M, J}} \left[\alpha^{M''}_{L''} D_0 p^{M'}_{L'} \mathcal{M}_{A1}(L'', L, M'', L'', M, L) + \alpha^{M''}_{L''} \frac{\gamma_{L'}}{r} \mathcal{M}_{B1} \right] A^{M}_{J}$$

$$+$$

$$\left[\text{Similar expressions in } B^{M}_{J} \text{ and } C^{M}_{J} \right] \tag{51}$$

$$= \alpha \nabla p$$

The final step is to collect coefficients of the A_L^M, B_L^M, and C_L^M and equate to those on the right side of the equation of motion. (If the friction is neglected the collection of terms is equated to zero.) When we do this, we obtain for A_L^M terms the expression following, where the nonlinear terms ($(\vec{V}\cdot\nabla)\vec{V}$, $\alpha\nabla p$) are sketched rather than explicitly enumerated:

From the A_L^M term we obtain, for a particular M and L:

$$\frac{\partial a_L^M}{\partial t} + \sum_{K,M',L',M'',L''} \mathcal{N}\, e_{K'}^{M'} D(M',L')\, e_{L''}^{M''}{}_{L''} + 2\Omega\Big[c_{L+1}^{M}\, \mathcal{A}_c(M,L) + c_{L-1}^{M}\, \mathcal{A}_c(M,L)\Big] + \sum\Big(\alpha_{L''}^{M''} D_0\, p_{L'}^{M'}\, \mathcal{M}_{A1} + \alpha_{L''}^{M''} \frac{\gamma_{L'}}{r}\, \mathcal{M}_{B1}\Big) + \frac{\partial \varphi_L^M}{\partial r} = \eta\left\{\Big[D_0 D_2 + \Big(\frac{\gamma_L}{r^2}\Big)\Big] a_L^M + \frac{2\gamma_L}{r^2}\right\} \tag{52}$$

The first term in the sum is the contribution of $\partial\vec{V}/\partial t$, the second sum results from $(\vec{V}\cdot\nabla)\vec{V}$, the bracket multiplied by 2Ω is the Coriolis term, the following term is from $\alpha\nabla$ p, the last term on the left is from $\nabla\varphi$, while the right side of the equation is the friction term. This yields a nonlinear partial differential equation with the angular part separated out-, invilving the expansion coefficients, a, b, c, p, ρ, α.

A similar performance for the B_L^M and C_L^M coefficients yields additional equations in the expansion coefficients. When the process is repeated for all the values of M and L, we obtain a set of coupled nonlinear differential equations for the expansion coefficients. The angular part is separated, i. e. the differential equations involve r and t only.

IIc. The Thermodynamic Equation

The equation is $$\frac{1}{c_p T}\frac{dq}{dt} = \frac{1}{T}\frac{dT}{dt} + \frac{k}{p}\frac{dp}{dt}$$

or

$$\frac{1}{c_p}\frac{dq}{dt} = \frac{dT}{dt} + k\frac{T}{p}\frac{dp}{dt} \tag{53}$$

we use the expression $$\frac{d\varphi}{dt} = \frac{\partial\varphi}{\partial t} + \vec{v}\cdot(\nabla\varphi) \tag{54}$$

(φ is a scalar function) to obtain

$$\frac{1}{c_p}\frac{dq}{dt} = \frac{\partial T}{\partial t} + \vec{v}\cdot\nabla T + \frac{kT}{p}\left(\frac{\partial p}{\partial t} + \vec{v}\cdot\nabla p\right) \tag{55}$$

To obtain only products, we multiply by p and obtain:

$$p\frac{dq}{dt} = p\left(\frac{\partial T}{\partial t} + \vec{v}\cdot(\nabla T)\right) + kT\left(\frac{\partial p}{\partial t} + \vec{v}\cdot\nabla p\right) \tag{56}$$

As usual, we expand the unknown functions p and T in scalar spherical harmonics and the velocity in terms of vector spherical harmonics. It will be noted that manipulations for p ($\frac{\partial T}{\partial t} + \vec{v}\cdot\nabla T$) and k ($\frac{\partial p}{\partial t} + \vec{v}\cdot\nabla p$) are formally the same, so only one will be indicated. These terms involve $\vec{v}\cdot(\nabla\overline{\ }$) which we write as

$$\vec{v}\cdot(\nabla T) = \left(\sum_{ML} a_L^M A_L^M + b_L^M B_L^M + c_L^M C_L^M\right)\cdot\left(\nabla\left[\sum_{M',L'} T_{L'}^{M'} Y_{L'}^{M'}\right]\right)$$

$$=$$

$$\left(\sum_{M,L} a_L^M A_L^M + b_L^M B_L^M + c_L^M C_L^M\right)\cdot\left(\sum_{M',L'} D_0 T_{L'}^{M'} A_{L'}^{M'} + \frac{\gamma_{L'}}{r} T_{L'}^{M'} B_{L'}^{M'}\right) \tag{57}$$

As in the case of the equation of motion, the nonzero terms that result are $A_L^M \cdot A_{L'}^{M'}$, $B_L^M \cdot B_{L'}^{M'}$, $C_L^{M'} \cdot B_L^{M'}$. Using the decomposition rules for these terms we obtain:

$$\vec{v}\cdot\nabla T =$$

$$\sum_{M',L',M'',L'',M,L} a_L^M D_0 T_{L'}^{M'} C_{AA} Y_{L''}^{M''} + \sum b_L^M \frac{\gamma_{L'}}{r} C_{BB} Y_{L''}^{M''} + \sum c_L^M \frac{\gamma_{L'}}{r} T_{L'}^{M'} C_{BC} Y_{L''}^{M''} \tag{58}$$

This term must now be multiplied by T which can also be handled by summation rules. We have

$$p\cdot(\vec{v}\cdot\nabla T)$$

$$\left(\sum_{L''',M'''} p_{L'''}^{M'''} Y_{L'''}^{M'''}\right)\left(\sum_{L'',M'',L',M',L,M}\left(a_L^M D_0 p_{L'}^{M'} C_{AA} Y_{L''}^{M''}\right) + \sum b_L^M \frac{\gamma_{L'}}{r} C_{BB} Y_{L''}^{M''}\right.$$

$$\left. + \sum c_L^M \frac{\gamma_{L'}}{r} p_L^M Y_{L''}^{M''}\right) \tag{59}$$

which will yield terms involving sums of products of coefficients of T, e.g

$$c_L^M T_{L'}^{M'} p_{L'''}^{M'''} \frac{\gamma_{L'}}{r} Y_{L''''}^{M''''} C_{AA}(M'''', L'''', \ldots) C_{AA}(M'', L'' \tag{60}$$

also, we will encounter terms of the form

$$a_L^M \left(D_0 p_{L'}^{M'}\right) p_{L''''}^{M''''} Y_{L''''}^{M''''} C_{AA}(M'''', \ldots)\, C_{AA}(M'', \ldots) \tag{61}$$

and

$$b_L^M \frac{Y_{L'}}{r} p_{L'''}^{M'''} Y_{L''''}^{M''''} C_{AA}(M'''', \ldots)\, C_{AA}(M'', \ldots) \tag{62}$$

which can be collected for specific values of the indices M'''', L''''. This operation will not be carried out in detail because of the complexity of the algebra.

The term $\rho\, \partial T/\partial t$ is also handled by the decomposition rule; we have

$$\rho \frac{\partial T}{\partial t} = \left(\sum_{M,L} p_L^M Y_L^M\right)\left(\sum_{M',L'} \frac{\partial T_{L'}^{M'}}{\partial t} Y_{L'}^{M'}\right) \tag{63}$$

$$= \sum_{M',L'} \sum_{\substack{M'',L'' \\ M,L}} p_L^M \frac{\partial T_{L'}^{M'}}{\partial t} C_{AA}(M'', L'' \ldots)\, Y_{L''}^{M''} \tag{64}$$

The left side of the equation is also handled by the decomposition rule to obtain

$$\rho \frac{dq}{dt} = \sum_{M',L',\ldots} p_L^M \dot{q}_{L'}^{M'} C_{AA}(M'', L'' \ldots)\, Y_{L''}^{M''} \tag{65}$$

where the $\dot{q}_{L'}^{M'}$ are given coefficients representing heat input.

Finally, the term $kT\left(\frac{\partial \rho}{\partial t} + \bar{v}\cdot\nabla\rho\right)$ is handled analogously to the $p\left(\frac{\partial T}{\partial t} + \vec{v}\cdot\nabla T\right)$ term.

Upon carrying out that operation and collecting all the coefficients of the Y_L^M's, we have a set of coupled, nonlinear partial differential equations involving T_L^M, p_L^M, and $\underset{L}{M}$, and as usual the angular part does not appear; so that the equations are of first order in r and t.

SECTION 3

Pressure Coordinates

If one wants to use pressure coordinates in the spherical representation r is replaced as an independent variable by p. The following treatment is incidentally valid independently of the specific meaning of p, it can be any scalar expressed in spherical coordinates. Let us apply equation (35) in the section 1. to $f(p)Y_L^M(\theta,\varphi)$, where $p=p(r,\theta,\varphi)$. Then $\frac{\partial f(p)}{\partial x} = \frac{\partial f}{\partial p}\frac{\partial p}{\partial x}$ and similarly for y, and z.

Thus,

$$\partial_\mu f(p) = \frac{\partial f}{\partial p}\partial_\mu p$$

and (35) becomes

$$\partial_\mu\left[f(p)Y_L^M(\theta,\varphi)\right] = \left[\frac{\partial f(p)}{\partial p}\partial_\mu p\right]Y_L^M(\theta,\varphi) \tag{1}$$

$$+\frac{L}{r}\sqrt{\frac{L+1}{2L+1}}(-1)^\mu\langle L+1\ M+\mu\ 1\ -\mu|L\ M\rangle Y_{L+1}^{M+\mu}(\theta,\varphi)f(p)$$

$$+\frac{L+1}{r}\sqrt{\frac{L}{2L+1}}(-1)^\mu\langle L-1\ M+\mu\ 1\ -\mu|L\ M\rangle Y_{L-1}^{M+\mu}(\theta,\varphi)f(p)$$

(1) can be used to find the appropriate differential equations in pressure-coordinates, but sometimes there are more direct ways. The gradient, divergence and curl expressions become

$$\nabla f(p)\, Y_L^M(\theta,\varphi) = \frac{\partial f}{\partial p}\, \nabla p\, Y_L^M(\theta,\varphi) + \frac{\gamma_L}{r} f(p)\, B_L^M(\theta,\varphi) \tag{2}$$

$$\nabla \cdot f(p)\, A_L^M(\theta,\varphi) = \frac{\partial f}{\partial p}\, \nabla p \cdot A_L^M(\theta,\varphi) + \frac{2}{r} f(p)\, Y_L^M(\theta,\varphi) \tag{3}$$

$$\nabla \cdot f(p)\, B_L^M(\theta,\varphi) = \frac{\partial f}{\partial p}\, \nabla p \cdot B_L^M(\theta,\varphi) - \frac{\gamma_L}{r} f(p)\, Y_L^M(\theta,\varphi) \tag{4}$$

$$\nabla \cdot f(p)\, C_L^M(\theta,\varphi) = \frac{\partial f}{\partial p}\, \nabla p \cdot C_L^M(\theta,\varphi) \tag{5}$$

$$\nabla \times f(p)\, A_L^M(\theta,\varphi) = \frac{\partial f}{\partial p}\, \nabla p \times A_L^M(\theta,\varphi) + \frac{i\gamma_L}{r} f(p)\, C_L^M(\theta,\varphi) \tag{6}$$

$$\nabla \times f(p)\, B_L^M(\theta,\varphi) = \frac{\partial f}{\partial p}\, \nabla p \times B_L^M(\theta,\varphi) + \frac{i}{r} f(p)\, C_L^M(\theta,\varphi) \tag{7}$$

$$\nabla \times f(p)\, C_L^M(\theta,\varphi) = \frac{\partial f}{\partial p}\, \nabla p \times C_L^M(\theta,\varphi) + \frac{i\gamma_L}{r} f(p)\, A_L^M(\theta,\varphi) + \frac{i}{r} f(p)\, B_L^M(\theta,\varphi) \tag{8}$$

The continuity equation

$$\frac{\partial \rho}{\partial t} + \rho \nabla \cdot \vec{V} + \vec{V} \cdot \nabla \rho = 0$$

can be handled as follows.

Substitute $\rho = -\frac{1}{g}\frac{\partial p}{\partial r}$ into the continuity equation to obtain

$$\frac{\partial}{\partial t}\frac{\partial p}{\partial r} + \frac{\partial p}{\partial r}\nabla \cdot \vec{V} + \vec{V} \cdot \nabla \frac{\partial p}{\partial r} = 0$$

$$\frac{\partial}{\partial r}\left[\frac{\partial p}{\partial t} + \vec{V} \cdot \nabla p\right] + \frac{\partial p}{\partial r}\nabla \cdot \vec{V} - \frac{\partial \vec{V}}{\partial r} \cdot \nabla p = 0$$

Write

$$\frac{\partial p}{\partial t} + \vec{V} \cdot \nabla p = \omega$$ and using Eqs 3, 4, 5

$$\frac{\partial \omega}{\partial r} + \frac{\partial p}{\partial r}\Big[\sum \frac{\partial a_L^M}{\partial p}\nabla p \cdot A_L^M + \sum a_L^M(p)\frac{2}{r}Y_L^M$$

$$+ \sum \frac{\partial b_L^M}{\partial p}\nabla p \cdot B_L^M - \sum b_L^M(p)\frac{\gamma_L}{r}Y_L^M$$

$$+ \sum \frac{\partial c_L^M}{\partial p}\nabla p \cdot C_L^M\Big] - \frac{\partial \vec{V}}{\partial r} \cdot \nabla p = 0$$

but

$$\frac{\partial p}{\partial r}\left[\sum \frac{\partial a_L^M}{\partial p}\nabla p \cdot A_L^M + \sum \frac{\partial b_L^M}{\partial p}\nabla p \cdot B_L^M + \sum \frac{\partial c_L^M}{\partial p}\nabla p \cdot C_L^M\right] \equiv \frac{\partial \vec{V}}{\partial r} \cdot \nabla p$$

Therefore,

$$\frac{\partial \omega}{\partial r} + \frac{\partial p}{\partial r}\sum\left[\frac{2}{r}a_L^M(p) - \frac{\gamma_L}{r}b_L^M(p)\right]Y_L^M = 0$$

22

and if $\omega = \sum \omega_L^M(p) Y_L^M(\theta,\varphi)$ then

$$\frac{\partial \omega_L^M(p)}{\partial p} + \frac{2}{r} a_L^M(p) - \frac{\gamma_L}{r} b_L^M(p) = 0 \qquad (9)$$

where the $\frac{\partial}{\partial r} = -\rho g \frac{\partial}{\partial p}$ substitution has been used.

Note that we have kept the vertical velocity-component in the derivation. If we leave it out the continuity equation reduces to

$$\frac{\partial \omega_L^M}{\partial p} = \frac{\gamma_L}{r} b_L^M(p) \qquad (10)$$

Thus one can see that only part of the horizontal wind contributes to $\partial\omega/\partial p$ namely that part in the expansion which depends on functions $B_L^M(\theta,\varphi)$. The zonal wind which depends on terms of the form $C_L^0(\theta,\varphi)$ and Rossby waves which depend on terms of the form $C_L^M(\theta,\varphi)$ do not contribute to $\partial\omega/\partial p$. This can perhaps be an alternative explanation to the smallness of ω, namely, the large features of the atmospheric circulation do not contribute to ω.

N ote that equation (1) shows that there are no derivatives with respect to r left when p, θ and φ are the independent variables so in this formulation it is not necessary to use the hydrostatic balance condition to make the substitution

$$\frac{\partial}{\partial r} = -\rho g \frac{\partial}{\partial p}$$

23

Note also that r still remains in the equations and it does not seem possible to eliminate this variable from the equations which means that one cannot really cast the equations in pressure coordinates at all. This does not, however, appear to be too serious since putting r equal to the radius of the earth is a good approximation.

SECTION 4

The Transform Method

In recent years a transform method (1, 2) has been developed which appears to make spectral methods competitive with the finite difference method as far as speed is concerned. We give a brief discussion of the basic ideas and show that they also work for vector spherical harmonics.

Consider a differential equation for a scalar field $\psi(t, \theta, \phi)$ expressed in terms of the independent variables t, θ and φ. Let ψ satisfy a partial differential equation of the form

$$\frac{\partial \psi}{\partial t} + L\psi = N(\psi) \tag{1}$$

When L is a linear operator and N a non-linear term. L and N may or may not contain differential operations. Let us ignore the linear term since it is easy to deal with and just write

$$\frac{\partial \psi}{\partial t} = N(\psi) \tag{2}$$

The conventional spectral method approach to solving (2) is to expand ψ in terms of a complete orthonormal set of functions in θ and φ ie. scalar spherical harmonics $Y_L^M(\theta, \varphi)$

$$\psi(t, \theta, \varphi) = \sum_{M,L} a_L^M(t)\, Y_L^M(\theta, \varphi) \tag{3}$$

To be specific let N (ψ) be equal to $a\psi^2$ where a is a constant used for dimensional reasons. Thus (2) becomes

$$\frac{\partial \psi}{\partial t} = a\psi^2 \tag{4}$$

and substituting (3) into (4) we get

$$\sum_{M,L} \frac{d a_L^M(t)}{dt} Y_L^M(\theta,\varphi) = a\left(\sum_{M',L'} a_{L'}^{M'}(t)\, Y_{L'}^{M'}(\theta,\varphi)\right)\left(\sum_{M'',L''} a_{L''}^{M''}(t)\, Y_{L''}^{M''}(\theta,\varphi)\right) \tag{5}$$

Now product term of the form $Y_{L'}^{M'} Y_{L''}^{M''}$ can be expressed in terms of sums over individual terms $Y_{L'''}^{M'+M''}$ by the use of the formula

$$Y_{L'}^{M'} Y_{L''}^{M''} = \sum_{L'''} \left[\frac{(2L'+1)(2L''+1)}{4\pi(2L'''+1)}\right]^{1/2} \langle L'0L''0|L'''0\rangle \langle L'M'L''M''|L'''M'+M''\rangle\, Y_{L'''}^{M'+M''} \tag{6}$$

Let us abbreviate the result of substituting (6) into (5) by expressing (5) as

$$\sum_{M,L} \frac{d a_L^M(t)}{dt} Y_L^M(\theta,\varphi) = a \sum_{M',L'} \sum_{M'',L''} a_{L'}^{M'}(t)\, a_{L''}^{M''}(t) \sum_{L'''=|L'-L''|}^{L'+L''} I(L',L'',L''',M',M'')\, Y_{L'''}^{M'+M''}(\theta,\varphi) \tag{7}$$

Now using the orthonormatity of the Y_L^M 's we multiply both sides by

$\left[Y_q^p(\theta,\varphi)\right]^* \sin\theta \, d\theta \, d\varphi$ and integrate Then:

$$\sum \frac{da_L^M}{dt} \delta_{Mp} \delta_{Lq} = a \sum_{M',L'} \sum_{M'',L''} a_{L'}^{M'} a_{L''} \sum_{L'''} I(L',L'',L''',M',M'') \delta_{M'+M'',p} \, \delta_{L''',q} \tag{8}$$

or

$$\frac{da_q^p}{dt} = a \sum_{M',L',L''} a_{L'}^{M'} a_{L''}^{p-M'} I(L',L'',q,M',p-M') \tag{9}$$

The basic idea is to replace the time-consuming calculation of the right-hand side of (9) by a faster method.

Let us rewrite (7) in the form

$$\sum \frac{da_L^M}{dt}(t) Y_L^M(\theta,\varphi) = \sum d_{M'+M'',L'''} Y_{L'''}^{M'+M''}(\theta,\varphi) \tag{10}$$

Multiply both sides by $\left[Y^{p}_{q}(\theta,\varphi)\right]^{*}\sin\theta\, d\theta\, d\varphi$ and integrate (10) to obtain

$$\frac{da^{p}_{q}}{dt} = d_{pq} \tag{11}$$

where the explicit form, which one does not want to use, for the right-hand side is that of the right-hand side of (9). Instead, there is a less time consuming way of finding d_{pq} and to attack (10) in the following way. Suppose we truncate L, L', L'' and L''' at N. Abbreviate

$$\sum_{L'''=0}^{2N}\sum_{M} d_{M'+M'',L'''}\, Y^{M'+M''}_{L'''}(\theta,\varphi) \equiv J_N(\theta,\varphi) \tag{12}$$

Write

$$Y^{M}_{L}(\theta,\varphi) = X^{M}_{L}(\theta)\, e^{iM\varphi} \tag{13}$$

then form (5)

$$\begin{aligned} J_N(\theta,\varphi) &= a\left(\sum_{L',M'} a^{M'}_{L'} Y^{M'}_{L'}(\theta,\varphi)\right)\left(\sum_{L'',M''} a^{M''}_{L''} Y^{M''}_{L''}(\theta,\varphi)\right) \\ &= \sum d_{M'+M'',L'''}\, X^{M'+M''}_{L'''}(\theta)\, e^{i(M'+M'')\varphi} \end{aligned} \tag{14}$$

Now let θ and φ take on discrete values θ_j, φ_k and insert them into (14)

Let

$$\varphi_k = \frac{2\pi k}{4N}$$

Multiply each side of (14) by

$$\frac{1}{4N} e^{-2\pi i m k/4N}$$

and sum over k

$$\frac{1}{4N}\sum_k J_N(\theta_j, \varphi_k)\, e^{-2\pi i m k/4N}$$

$$= \sum\sum d_{M'+M'', L'''} X^{M'+M''}_{L'''}(\theta_j)\, e^{i\left[(M'+M'')\frac{2\pi k}{4N} - \frac{2\pi m k}{4N}\right]} \tag{15}$$

Now

$$\frac{1}{4N}\sum_{k=1}^{4N} e^{2\pi i (M'+M''-m)k} = \delta_{M'+M'', m} \tag{16}$$

$$\frac{1}{4N}\sum_k J_N(\theta_j, \varphi_k)\, e^{2\pi i m k/4N} = \sum_m \sum_{L'''} \delta_{m, L'''} X^m_{L'''}(\theta_j) \equiv g_m(\theta_j) \tag{17}$$

m is fixed so the sum over m vanishes and we have a set of equations, one for each m. (17) then become

$$\sum_{L'''} d_{mL'''} X^{m}_{L'''}(\theta_j) = g_m(\theta_j) \tag{18}$$

$X^{m}_{L'''}(\theta_j)$ can then be considered a set of matrices labeled by m with L''' and j as the indices. Finding the inverses of the set of matrices X we can solve (18) for $d_{mL'''}$ which is what the problem to be solved was.

We will now indicate how the transform method is applied to vector spherical harmonics. Since we have an analytic expression for the term $(\vec{v}\cdot\nabla)\vec{v}$ in the vector spherical harmonics formulation and will be able to use it to check the accuracy of the numerical computation, we use $(\vec{v}\cdot\nabla)\vec{v}$ as an example.

The key to putting the transform technique to practice is to make sure that: all ∇ operations are performed in coefficient space, all non-linear operations are performed in physical space. Of course, one must make sure that the density of physical space grid is consistent with the number of coefficients and the particular non-linear operations being performed.

Consider, thus, the nonlinearity:

$$(\vec{v}\cdot\nabla)\vec{v}$$

It can be expressed as

$$(\vec{v}\cdot\nabla)\vec{v} = \frac{1}{2}\nabla(\vec{v}\cdot\vec{v}) - \vec{v}\times(\nabla\times\vec{v})$$

Remembering that we will be computing the vector spherical harmonic coefficients, v_{jl}^{m} , of the variable v and that in coefficient space, the curl of an expanded variable is easy to form, $\vec{v} \times (\nabla \times \vec{v})$ can easily be formed. It requires:

It requires:

- o forming curl in coefficient space and transforming to physical space. $(\nabla \times \vec{v})$.

- o transforming the v_{jl}^{m} to physical space. $(\vec{v})$.

- o forming the cross product at each point on the physical grid $(\vec{v} \times (\nabla \times \vec{v}))$.

The gradient of $\vec{v} \cdot \vec{v}$ is a little more difficult. In the space defined by the $\hat{e}_{\mu}$ unit vectors, $\vec{v}$ can be expressed as:

$$\vec{v} = v_{-}\hat{e}_{-} + v_{o}\hat{e}_{o} + v_{+}\hat{e}_{+}$$

and using the dot product rules in the $\hat{e}_{\mu}$ space

$$\vec{v} \cdot \vec{v} = -2 v_{-} v_{+} + v_{o}^{2}$$

and then applying the gradient operator produces

$$\frac{1}{2}\nabla(\vec{v} \cdot \vec{v}) = v_{o}\nabla v_{o} - v_{o}\nabla v_{+} - v_{+}\nabla v_{-}$$

It can be shown that for a real vector $\vec{v}$

$$v_{-}^{*} = -v_{+}^{*}$$

leading to the relationship

$$v_+ \nabla v_- = (v_- \nabla v_+)^*$$

The remaining steps in computing the advection term accomplish an evaluation of

$$\frac{1}{2} \nabla(\vec{v} \cdot \vec{v}) = v_o \nabla v_o - 2 \, \mathrm{Re}(v_- \nabla v_+)$$

followed by a VSH expansion of the non-linearity.

- o forming gradient of the $\hat{e}_o$ component of $\vec{v}$ and transforming to physical space. (∇v_o).

- o forming gradient of the $\hat{e}_+$ component of $\vec{v}$ and transforming to physical space. (∇v_+).

- o forming the product of the above gradients and the appropriate components of $\vec{v}$ over the physical grid. ($v_o \nabla v_o$ and $v_- \nabla v_+$).

- o Summing the above results over the physical grid forming $((\vec{v} \cdot \nabla) \vec{v})$.

- o Transforming the advection term to coefficient space producing the VSH coefficients of $(\vec{v} \cdot \nabla) \vec{v}$.

To illustrate the advantage of using the transform technique, consider the vector example

$$\frac{\partial \vec{v}}{\partial t} = (\vec{v} \cdot \nabla)\vec{v} + \ldots$$

where

$$\vec{v} = \sum_{M,J,L} v_{JL}^{M} T_{JL}^{M}$$

Formal substitution produces,

$$\frac{\partial v_{JL}^{M}}{\partial t} = \sum_{M'J'L'J''L''} C(M,J,L,M',J',L',J'',L'') v_{J'L'}^{M'} v_{J''L''}^{M-M'}$$

Assumming that the C's are precomputed and available, a rough operation count (where L_m is the truncation value of L) is:

$+ C.v.v$ = 6 real mpy + 4 real add (v complex, C real)

$\simeq$ 5 operations (an operation is mpy + add)

the number of v_{JL}^{M} = the number of $v_{J'L'}^{M'} = \frac{3L_m^2}{2}$

the number of $v_{J''L''}$ (superscript determined by M&M' above) = 3 L_m

$\therefore$ total number of accumulations = total number of C's to be stored

$$\simeq \left(\frac{3L_m^2}{2}\right)\left(\frac{3L_m^2}{2}\right) 3L_m = \frac{27}{4} L_m^5$$

and the total number of operations $\simeq \frac{135}{4} L_m^5$

Approximately 80% of the C's included in the computation above are equal to zero. Assumming that these could be bypassed at no cost, the totals become

number of C's to store $\simeq 1.35\, L_m^5$

and number of operations $\simeq 6.75\, L_m^5$

Use of the transform method involves the following steps:

1) Compute $\nabla \times \vec{V}$
2) Compute ∇V_0
3) Compute $\nabla V_\perp$
4) Compute $\vec{V}$

Over the physical grid. Essentially four vector transformations. $\simeq 5L_m^3$ operations each

5) Combine over physical grid to form $(\vec{V}\cdot\nabla)\vec{V} \simeq 200L_m^2$ operations

6) Transform $(\vec{V}\cdot\nabla)\vec{V}$ to coefficient space (one vector transformation) $\simeq 5L_m^3$ operations.

The total number of operations by this technique is approximately $25L_m^3$. This represents a significant savings for large L_m (see Table below) There is also a large savings in storage locations.

Operation Count for Computing the

$(\vec{V}\cdot\nabla)\vec{V}$ Portion of $\frac{\partial}{\partial t} V_{JL}^M$.

Straight Forward Method . . . . $6L_m^5 + 12L_m^4 + 6L_m^3$

(assumes existence of selection rules that eliminate 80% of the terms at no cost)

Transform Method . . . . $25L_m^3 + (415 + 200 \log_2 L_m) L_m^2 + 25L_m$

L_m	4	8	16	32	64	128
$\frac{SFM}{QM}$	.65	3.2	17	94	500	2500

REFERENCES

1. Orszag, S. A. , "Transform Method for the Calculation of Vector-Coupled Sums: Application to the Spectral Form of the Vorticity Equation," J. of the Atmospheric Sciences, Sept. 1970, pp. 890 - 895.

2. Eliasen, E., Machenhauer, B., Rasmussen, E., "On a Numerical Method for Integration of the Hydrodynamical Equations with a Spectral Representation of the Horizontal Fields," University of Copenhagen, Institute for Theoretical Meteorology, Report No. 2, 1970.

SECTION 5

The Mintz-Arakawa Model in Vector Spherical Harmonics

A simplified version of the Mintz-Arakawa model has been written in terms of the coefficients of the vector spherical harmonic series, representing the dependent variable $\vec{v}$. Also used are the scalar series of the scalar spherical harmonics, used for temperature, ground pressure, etc.

Left out in this preliminary version in agreement with the statement of work in the contract are moisture and radiation.

In order to accommodate the continuous nature of the model, the friction term at the bottom has been modified. Furthermore, it is assumed that the earth's surface temperature is given as a function of position and time. The surface temperature can lateron be calculated from zero heat balance on land, if desired.

The Mintz-Arakawa Model

The Mintz-Arakawa equations, as given in [3] are the equation of horizontal motion (coordinates)

$$\frac{\partial}{\partial t}(\pi \bar{v}) + (\nabla . \pi \bar{v})\bar{v} + \frac{\partial}{\partial \sigma}(\pi \bar{v} \dot{\sigma}) + f \bar{k} \times \pi \vec{v} + \pi \nabla \phi + \sigma \pi \alpha (\nabla \pi) = \pi \bar{F}$$

The thermodynamic equation

$$\frac{\partial}{\partial t}(\pi c_p T) + \nabla . (\pi c_p T \bar{v}) + \frac{\partial}{\partial \sigma}(\pi c_p T \dot{\sigma}) - \pi \alpha \left(\sigma \frac{\partial \pi}{\partial t} + \sigma \bar{v} . \nabla \pi + \pi \dot{\sigma}\right) = \pi \dot{H}$$

36

The continuity equation

$$\frac{\partial \pi}{\partial t} + \nabla \cdot (\pi \bar{v}) + \frac{\partial}{\partial \sigma}(\pi \dot{\sigma}) = 0$$

The moisture continuity equation

$$\frac{\partial}{\partial t}(\pi q) + \nabla \cdot (\pi q \bar{v}) + \frac{\partial}{\partial \sigma}(\pi q \dot{\sigma}) = \pi \dot{Q}$$

The equation of state

$$p = \rho R T$$

and the hydrostatic balance equation

$$\frac{\partial \phi}{\partial \sigma} + \pi \alpha = 0$$

In these equations

π = $p_s - p_T$ = surface pressure--top pressure

$\bar{v}$ = horizontal velocity (in σ coordinates)

σ = $\frac{p - p_T}{p_s - p_T}$ where p is the pressure

$\dot{\sigma}$ = $\frac{d\sigma}{dt}$

$\bar{k}$ = vertical unit vector

f = $2\Omega \sin \varphi$ where

Ω = rate of rotation of the earth and

φ = latitude, positive worthward from the equation

ϕ = geopotential

α = specific volume = $\frac{1}{\rho}$

$\bar{F}$ = horizontal vector frictional force per unit mass.

37

c_p = specific heat (for dry air) at constant pressure

T = temperature

$\dot{H}$ = diabatic heating rate per unit mass

q = mixing ratio

λ = longitude

$\dot{\varphi}$ = rate of moisture addition

The boundary conditions are:

At $\sigma = 1$ over land

$$\dot{\sigma} = 0$$

$$\phi = \phi_4(\lambda, \varphi) = \phi_G =$$ geopotential of the earth's surface

and $F_H = 0$ where F_H is the vertical heat flux at the surface.

At $\sigma = 1$ over ocean

$$\dot{\sigma} = 0$$

$$\phi = 0$$

$$T = T_S(\lambda, \varphi)$$

and at $p = p_T$:

$$\frac{dp}{dt} = 0$$

which is the fr surface condition.

These equations are written in momentum form, that is, the variables $\bar{V}$, T and q are multiplied by π, so instead of $\bar{V}$, T and q, the variables are $\pi\bar{V}$, $\pi c_p T$, πq.

This can be accomplished by multiplying the equation of motion with π, the continuity equation by $\bar{v}$ and adding the two resulting equations. In the next section a short derivation of these equations is given and the major approximations made, are listed. However, the equations will be written in the original variables $\bar{v}$, T, etc. rather than $\pi\bar{v}$ etc. The reason for this is that besides the variable ($\pi\bar{v}$) also the quantity $\bar{v}$ is used (see for instance the equation of motion where the second term on the left is $\nabla\cdot(\pi\bar{v})\,\bar{v}$. This quantity $\bar{v}$ is obtained from $\frac{(\pi\bar{v})}{\pi}$, which requires a division. It was thought to be desirable to avoid this at this point. Also the reasons for using etc. are mainly that in the finite difference approximation, it is necessary that momentum is conserved by the mathematical approximation techniques used. This simplies, in that case, to use ($\pi\bar{V}$) instead of $\vec{v}$.

The following section reviews the derivation and lists the major approximation used. In that section the equation for the moisture has not been reviewed as moisture has been deleted from the Vector Spherical Harmonics Model at the present time.

The following approximations are made to obtain the quasi-static approximation. [1, p 20-22].

Subscript h indicates horizontal component

1° Vertical acceleration is neglected

2° Horizontal component of $\bar{\Omega}$ is ignored in the expression for the Coriolis force

3° $\left(\frac{D\bar{V}}{Dt}\right)_h$ is replaced by $\left(\frac{D\bar{V}_h}{Dt}\right)_h$

4° $\mathrm{div}_h\,(\rho\bar{V})$ is replaced by $\mathrm{div}_h\,(\rho V_h)$

The equation of motion can be written then in two parts. If W is the vertical velocity and the subscript h indicate the horizontal component:

$$\left(\frac{\partial \bar{v}_h}{\partial t} + \bar{v}_h \cdot \nabla \bar{v}\right)_h + W \cdot \frac{\partial \bar{v}_h}{\partial z} + \alpha \nabla_h p + \left(2\Omega \times \bar{V}_h\right)_h = \text{Friction.} \qquad (1)$$

which is the horizontal part. The vertical part becomes

$$\frac{\partial p}{\partial z} = -\rho g \qquad (2)$$

The friction term of the horizontal part will be specified later; in the vertical part it has been neglected [1, p 33-41].

Furthermore, the continuity equation is written without approximation:

$$\frac{\partial \rho}{\partial t} = -\nabla_h (\rho \bar{V}_h) - \frac{\partial (\rho w)}{\partial z} \qquad (3)$$

while the thermodynamic equation becomes:

$$\frac{\partial \theta_T}{\partial t} + \bar{V}_h \cdot \nabla \theta_T + W \frac{\partial \theta_T}{\partial z} = \frac{1}{c_p} \left(\frac{p_0}{p}\right)^{\kappa} Q \qquad (4)$$

where θ_T is the potential temperature $\left(\frac{p_0}{p}\right)^{\kappa} T$.

$\kappa = \frac{c_p - c_v}{c_p}$

Pressure Coordinates

In these coordinates pressure instead of r is used as the vertical coordinate. This simplifies the equations, though it complicates the boundary condition. However, it also agrees with customary meteorological terminology - Any function φ' (x, y, z) will be changed to a function φ (x, y, p(x, y, z).

If $$\omega = \frac{dp}{dt} = \frac{\partial p}{\partial t} + \bar{V}_h \cdot \nabla p + w \frac{\partial p}{\partial z}$$

the continuity equation can be given in the new coordinates as

$$\frac{\partial \omega(p,\theta,\varphi)}{\partial p} + \nabla_h \cdot V_h(p,\theta,\varphi) = 0$$

This is the continuity equation in pressure coordinates.

Equations (1) and (4) become

$$\frac{\partial \bar{V}_h}{\partial t} + \bar{V}_h \cdot \nabla_h \bar{V}_h + \omega \frac{\partial \bar{V}_h}{\partial p} + (2\bar{\Omega} \times \bar{V}_h)_h + g \nabla_h z = \qquad (5)$$

Friction

$$\frac{\partial \theta}{\partial t} + \bar{V}_h \nabla_h \theta_T + \omega \frac{\partial \theta}{\partial p} = \frac{1}{c_p}\left(\frac{p_0}{p}\right)^{\kappa} Q \quad = \text{heating} \qquad (6)$$

Furthermore:

$$\frac{\partial z}{\partial p} = -\frac{\alpha}{g} = -\frac{RT}{gp} \qquad (7)$$

where

$$\alpha = \frac{RT}{p} \qquad (8)$$

σ Coordinates [3]

In order to accommodate Orographic features and, in general, make the earth's surface a surface on which only (θ, φ) vary a new variable is defined:

$$\sigma = \frac{p}{\pi} \tag{9}$$

where $\pi(\theta, \varphi, t)$ is the surface pressure. Also from here on, $\overline{V}$ indicates the horizontal component of the velocity, unless otherwise indicated. The following are the transformation rules:

$$\frac{\partial}{\partial p} = \frac{\partial}{\partial \sigma}\frac{\partial \sigma}{\partial p} = \frac{1}{\pi}\frac{\partial}{\partial \sigma} \tag{10}$$

$$\left(\frac{\partial}{\partial x}\right)_p = \left(\frac{\partial}{\partial x}\right)_\sigma + \frac{\partial}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \pi}\right)_p \frac{\partial \pi}{\partial x}$$

etc.

Now $\left(\frac{\partial \sigma}{\partial \pi}\right)_p = -\frac{\sigma}{\pi}$ which follows from eq. (9)

Then in vector form:

$$\nabla_p = \nabla_\sigma - \frac{\sigma}{\pi}\cdot(\nabla \pi)\frac{\partial}{\partial \sigma} \tag{11}$$

Also

$$\left(\frac{\partial}{\partial t}\right)_p = \left(\frac{\partial}{\partial t}\right)_\sigma + \frac{\partial}{\partial \sigma}\left(\frac{\partial \sigma}{\partial \pi}\right)_p \frac{\partial \pi}{\partial t} - \frac{\sigma}{\pi}\frac{\partial \pi}{\partial t}\frac{\partial}{\partial \sigma} \tag{12}$$

First the continuity equation will be changed to the new coordinates.

We had the continuity equation:

$$\frac{\partial \omega}{\partial p} + \nabla_h \cdot \bar{V}_h = 0 \qquad (13)$$

Now $\omega = \frac{dp}{dt} = \frac{d(\sigma\pi)}{dt} = \pi\frac{d\sigma}{dt} + \sigma\frac{d\pi}{dt} =$

$$\pi\omega_\sigma + \sigma\frac{\partial \pi}{\partial t} + \sigma\bar{V}\cdot\nabla_h\pi$$

Reproduced from best available copy.

where $\omega_\sigma = \frac{d\sigma}{dt}$ is the new vertical vertical σ - velocity.

The subscript σ will be left out, from now on so that ω_σ becomes ω

Using eqs (10), (11), (12), equation (13) becomes:

$$\frac{\partial(\pi\omega)}{\partial\sigma} + \frac{\partial\pi}{\partial t} + \sigma\frac{\partial}{\partial t}\left(\frac{\partial\pi}{\partial\sigma}\right) + \sigma\bar{V}\frac{\partial}{\partial\sigma}\nabla\pi + \sigma\nabla\pi\cdot\frac{\partial\bar{V}}{\partial\sigma}$$

$$+ \bar{V}\cdot\nabla\pi + \pi\nabla\cdot\bar{V} - \sigma\nabla\pi\cdot\frac{\partial\bar{V}}{\partial\sigma} = \frac{\partial\pi}{\partial t} + \nabla\cdot(\pi\bar{V}) + \frac{\partial(\pi\omega)}{\partial\sigma} = 0$$

as $\frac{\partial\pi}{\partial\sigma} = 0$ because π is not a function of σ. (Earth's surface is $\sigma = 1$ surface).

So the new continuity equation is:

$$\frac{\partial\pi}{\partial t} + \nabla\cdot(\pi\bar{V}) + \frac{\partial(\pi\omega)}{\partial\sigma} = 0 \qquad (14)$$

Using eq. (10) the equation of motion becomes:

$$\frac{\partial\bar{V}}{\partial t} + \bar{V}\cdot\nabla\bar{V} + \omega\frac{\partial\bar{V}}{\partial\sigma} + (2\bar{\Omega}\times\bar{V})_h + \nabla\phi - \frac{\sigma}{\pi}\cdot\nabla\pi\frac{\partial\phi}{\partial\sigma}$$

$$= \text{FRICTION} \qquad (15)$$

with the hydrostatic equation:

$$\frac{\partial \phi}{\partial \sigma} = -\frac{RT}{\sigma} \tag{16}$$

Integrating eq (14) from $\sigma = 0$ to $\sigma = 1$ gives:

$$\frac{\partial \pi}{\partial t} = -\nabla \cdot \int_0^1 (\pi \bar{V})\, d\sigma \tag{17}$$

where use is made of $\omega = 0$ at $\sigma = 0$ and $\sigma = 1$

Eliminating $\frac{\partial \pi}{\partial t}$ in eqs (14) and (17) results in:

$$\frac{\partial \omega}{\partial \sigma} = \frac{1}{\pi}\left\{ -\nabla \cdot (\pi \bar{V}) + \int_0^1 \nabla \cdot (\pi \bar{V})\, d\sigma \right\} \tag{18}$$

Noting that in equation (14) $\frac{\partial \pi}{\partial \sigma} = 0$ gives

$$\frac{\partial \omega}{\partial \sigma} = -\frac{1}{\pi}\left[\frac{\partial \pi}{\partial t} + \nabla \cdot (\pi V) \right] \tag{19}$$

which is equivalent to (14)

Eq. (18) can be used to calculate ω.

The thermodynamic equation becomes:

$$\left(\frac{\partial \theta_T}{\partial t}\right)_\sigma + \bar{V} \cdot \nabla \theta_T + \omega \frac{\partial \theta_T}{\partial \sigma} = \frac{1}{c_p}\left(\frac{p_0}{\sigma \pi}\right)^\kappa Q \tag{20}$$

Equations (15), (17), (18), (20) are to be solved as follows [2, p 341-344]

The initial data to be given are:

$$\pi(\theta, \varphi, t=0)$$
$$T(\theta, \varphi, t=0)$$
$$V_\sigma(\theta, \varphi, t=0)$$

First from eq. (16) calculate ϕ . To this purpose $\phi(\sigma=1)$ must be given. This is orographic data. Next ω is calculated from eq (18). This is the diagnostic part of the computation.

Next with the help of eq (15), $\frac{\partial \bar{V}}{\partial t}$ is computed. Eq. (17) is used to calculate $\frac{\partial \pi}{\partial t}$. Eq. (20) gives $\frac{\partial \theta}{\partial t}$. The results are used to calculate the $\bar{V}$, π, T at time t=t + Δ t

Friction Term

For a tentative description of the vertical eddy transport the expression, given by Robert [5] was used:

$$\text{Friction} = \alpha \frac{\partial}{\partial p}\left(p^2 \frac{\partial \bar{V}}{\partial p}\right)$$

α =. 0002 hour^{-1}.

Boundary Conditions

If eddy viscosity and bottom friction are neglected, sufficient boundary conditions are

$$\omega = 0 \quad \text{at} \quad \sigma = 1$$
$$\omega = 0 \quad \text{at} \quad \sigma = 0$$

The first condition specifies a zero mass flux into the earth. The second condition is interpreted as the free surface condition $\frac{dp}{dt} = 0$. Though the condition at the top and bottom are the same, they mean different things, namely, at the top $\frac{dp}{dt} = 0$. Now $p = \sigma\pi$ so $\frac{dp}{dt} = \sigma\frac{d\pi}{dt} + \pi\frac{d\sigma}{dt}$. As $\sigma = 0$ at the top and $\frac{d\sigma}{dt} = \omega$, it follows that $\omega = 0$ at the top.

These two boundary conditions are still sufficient if only the eddy viscosity is neglected. This is because of $\omega=0$ at the two boundaries and the occurance of only $\omega\frac{\partial\bar{v}}{\partial\sigma}$ in the equation of motion, e.g. (15). [2, p. 96, 97]

In those cases $\bar{V}_S$ the surface velocities, could be calculated from the differential equation (15) by putting $\omega\frac{\partial\bar{v}}{\partial\sigma} = 0$ and using a suitable expression for the surface friction. In many cases this frictional force is assumed to be very large so that $\bar{V}_s$ is effectively equal to zero. If this is not the case $\bar{V}_s$ must be calculated. To do this with the differential equation might be too inaccurate. The forces that are acting on the air particles are dominated by frictional, viscous forces which are not included in eq. (15). These forces work in a layer of about 1 mm thick [1, p 38]. Within this layer the wind velocity changes rapidly to become zero on the ground. At the top of this layer $\bar{V}$ is no longer zero. It is therefore better to give some boundary condition on v in that case. The same is true if the viscosity is no longer neglected. The vertical transport of eddies can be taken into account with a friction term $K\frac{\partial^2 v_h}{\partial z^2}$ [1, p 38, 39] in which case the equation becomes 2nd order in z and two boundary conditions have to be given [2, p 96] Rather than

putting the horizontal velocity equal to zero at the surface, a drag coefficient is introduced [5].

$$\left(\frac{\partial \overline{V}}{\partial \sigma}\right)_s = -\varepsilon' \overline{V}_s$$

where ε' is an experimental constant. Robert gives $\varepsilon' = 40$ which corresponds to a surface layer of about 800 meter

At the top of the layer $\frac{\partial \overline{V}}{\partial \sigma} = 0$ [2, p 96]. These are the conditions used here. However more work on this problem will have to be done.

Vector Spherical Harmonics

If vertical accelleration is neglected, the equation of motion falls into 2 parts, one part becomes the equation of motion (15) for the horizontal motion, the other part becomes the hydrostatic equation (16). Consequently $\overline{v}$ can be expanded into

$$\overline{V} = \sum_{L,M} \left\{ v_{bL}^{M}(\sigma, t)\, B_L^M + v_{cL}^{M}(\sigma, t)\, C_L^M \right\} \qquad (21)$$

where the B_L^M and C_L^M are the vector spherical harmonics as defined in [4]. They are functions of the coordinates (θ, φ) only. The v_{bL}^{M} and v_{cL}^{M} are functions of (σ, t). They are scalars and, in general are complex. Similarly the vertical σ -velocity is written as:

$$\omega = \sum_{L,M} \omega_L^M(\sigma, t)\, Y_L^M \qquad (22)$$

the ground pressure as

$$\pi = \sum_{L,M} \pi_L^M(\sigma,t)\, Y_L^M \tag{23}$$

the geopotential

$$\phi = \sum_{L,M} \phi_L^M(\sigma,t)\, Y_L^M \tag{24}$$

the temperature

$$T = \sum_{L,M} T_L^M(\sigma,t)\, Y_L^M \tag{25}$$

the potential temperature

$$\theta_T = \sum_{L,M} \theta_{TL}^M(\sigma,t)\, Y_L^M \tag{26}$$

Before substituting these expressions into equations (15), (16), (17), (18) and (19) it should be pointed out that the use of transform methods avoid the necessity of calculation coupling coefficients. Vector spherical harmonics expansions of non-linear expressions such as $\pi\bar{V}$ and $(\bar{v}.\nabla)\,\bar{v}$ can readily be obtained without any further approximation except truncational at maximum value of L in equations (21) to (26). The coefficients of the VSH expansions of $(\bar{V}.\nabla)\,\bar{v}$ for instance are indicated by

$$\left[(\bar{v}\cdot\nabla)\bar{v}\right]_{B_L^M} \quad ; \quad \left[(\bar{v}\cdot\nabla)\bar{v}\right]_{C_L^M}$$

In other words

$$(\bar{v}\cdot\nabla)\bar{v} = \sum_{LM}\left[\bar{v}\cdot\nabla v\right]_{B_L^M} B_L^M + \sum_{LM}\left[\bar{v}\cdot\nabla\bar{v}\right]_{C_L^M} C_L^M$$

An exception was made for $\left[\pi\bar{v}\right]_{B_L^M,\, C_L^M}$ which was given its own symbol

$$\chi_{bL}^M = \left[\pi\bar{v}\right]_{B_L^M}$$

$$\chi_{cL}^M = \left[\pi\bar{v}\right]_{C_L^M}$$

To change equations (15), (16), (17), (18) and (19) into equations for the spherical harmonics coefficients, use is made of the expressions where the vertical velocity has been ignored.

$$\nabla_\sigma f Y_L^M = \frac{\sqrt{L(L+1)}}{a} f B_L^M \tag{27}$$

$$\nabla_\sigma \cdot f B_L^M = -\frac{\sqrt{L(L+1)}}{a} f Y_L^M \tag{28}$$

$$\nabla_\sigma \cdot f C_L^M = 0 \tag{29}$$

Also, in [4] a general expression is given for $2\bar{\Omega}\times\bar{v}$. From this:

$$\left[(2\bar{\Omega}\times\bar{v}_h)_h\right]_{B_L^M} = 2i\Omega\lambda_L^M v_{cL-1}^M - \frac{2i\Omega M}{L(L+1)} v_{bL}^M + 2i\Omega\lambda_{L+1}^M v_{cL+1}^M \tag{30}$$

$$\left[(2\bar{\Omega}\times\bar{v}_h)_h\right]_{C_L^M} = 2i\Omega\lambda_L^M v_{bL-1}^M - \frac{2i\Omega M}{L(L+1)} v_{cL}^M + 2i\Omega\lambda_{L+1}^M v_{bL+1}^M \tag{31}$$

with

$$\lambda_L^M = \frac{1}{L}\sqrt{\frac{(L-1)(L+1)(L-M)(L+M)}{(2L-1)(2L+1)}} \qquad (32)$$

Substituting eqs (21) to (26) into eq (15) and making use of eqs (27) to (32) and collecting terms multiplying the same B_L^M and C_L^M the following equations replace eq. (15):

$$\frac{\partial v_{bL}^M}{\partial t} + \left[(\bar{v}\cdot\nabla)\bar{v}\right]_{B_L^M} + \left[\omega\cdot\frac{\partial \bar{v}}{\partial \sigma}\right]_{B_L^M} + \left[\frac{RT}{\pi}\cdot(\nabla\pi)\right]_{B_L^M}$$
$$+ 2i\Omega\lambda_L^M v_{cL-1}^M - \frac{2i\Omega M}{L(L+1)} v_{bL}^M + 2i\Omega\lambda_{L+1}^M v_{cL+1}^M +$$
$$\frac{\sqrt{L(L+1)}}{a}\phi_L^M = \alpha\frac{\partial}{\partial\sigma}\left(\sigma^2\frac{\partial v_{bL}^M}{\partial\sigma}\right) \qquad (15a)$$

and

$$\frac{\partial v_{cL}^M}{\partial t} + \left[(\bar{v}\cdot\nabla)\bar{v}\right]_{C_L^M} + \left[\omega\cdot\frac{\partial \bar{v}}{\partial \sigma}\right]_{C_L^M} + \left[\frac{RT}{\pi}(\nabla\pi)\right]_{C_L^M}$$
$$+ 2i\Omega\lambda_L^M v_{bL-1}^M - \frac{2i\Omega M}{L(L+1)} v_{cL}^M + 2i\Omega\lambda_{L+1}^M v_{bL+1}^M =$$
$$\alpha\frac{\partial}{\partial\sigma}\left(\sigma^2\frac{\partial v_{cL}^M}{\partial\sigma}\right) \qquad (15b)$$

The hydrostatic equation becomes:

$$\frac{\partial \phi_L^M}{\partial\sigma} = -\frac{R}{\sigma}T_L^M \qquad (16a)$$

50

Equation (17) becomes:

$$\frac{\partial \pi_L^M}{\partial t} = \frac{\sqrt{L(L+1)}}{a} \int_0^1 \chi_{6L}^M \, d\sigma \tag{17a}$$

while the equation for the vertical σ -velocity becomes:

$$\frac{\partial \omega_L^M}{\partial \sigma} = \left[\frac{1}{\pi} \sum_{LM} \left\{ \frac{\sqrt{L(L+1)}}{a} \left(\chi_{6L}^M - \int_0^1 \chi_{6L}^M \, d\sigma \right\} Y_L^M \right]_{Y_L^M} \tag{18a}$$

A division by has to be made here, however, this is done in an expression for a small quantity.

The thermodynamic equation is changed to:

$$\frac{\partial \theta_{TL}^M}{\partial t} + \left[\bar{V} . \sum_{LM} \frac{\sqrt{L(L+1)}}{a} \theta_{TL}^M B_L^M \right]_{Y_L^M} + \left[\omega \frac{\partial \theta_T}{\partial \sigma} \right]_{Y_L^M} = \frac{1}{c_p} \left(\frac{p_0}{\sigma \pi} \right)^{\kappa} Q_L^M Y_L^M \tag{20a}$$

Radial Dependence

The equations (15a) to (20a) still contain the derivitive $\frac{\partial}{\partial \sigma}$ and the integration $\int F d\sigma$. As the B_L^M, C_L^M etc are not eigenfunctions of the Navier Stokes equation, no natural radial functions have yet been decided on. Also it should be kept in mind that the amplitude of these radial functions are functions of time to be evaluated by solving the differential equations. The result is a set of coupled ordinary differential equations with these amplitudes as the dependent variables and time as the independent variable. The simplest

51

functions are the powers of σ . They have the important property that products of two powers is again a power of σ . This will simplify the treatment of the non-linear terms considerably. Therefore power series will be used as a first try.

Now the spacial boundary conditions impose certain conditions on these coefficients to be maintained at all times.

<u>Power Series Expansion in σ</u>

Suppose we consider here one possible approach. Let

$$v_{bL}^{M}(\sigma,t)=\sum_{k=0} v_{bL}^{M(k)}(t)\,\sigma^{k} \quad ; \quad v_{cL}^{M}(\sigma,t)=\sum v_{cL}^{M}(t)\,\sigma^{k}$$

A typical boundary condition would be $\bar{v} = 0$ at the surface, where $\sigma = 1$. So the boundary condition becomes:

$$\sum_{k=0} v_{bL}^{M(k)} = \sum_{k=0} v_{cL}^{M(k)} = 0 \quad ; \quad \sigma = 1$$

and the coupled differential equations have to be solved with these extra conditions. The method adopted will be described using power series, however, it is not necessarily limited to power series.

Suppose emperical evidence suggests an Nth degree curve for, say, the temperature. So there are N + 1 coefficients. If there is one boundary condition, for instance T = o at $\sigma = 1$, it means that an Nth degree curve must be fitted with the condition that T = 0 at $\sigma = 1$. This will give N degrees of freedom. If, on the other hand, the temperature is given in N points, (besides T = 0 at $\sigma = 1$), the Nth degree curve can be reconstructed at any time without losing any

52

information. Instead of using the (N+1) coefficients as variables it is therefore equally possible to use the value of T at N levels as variables; the value of T at the boundary can be obtained from the boundary condition and the N values of T. Furthermore, there is another advantage. Writing out the equations for both methods, the equations for the coefficients get extremely complicated while the equations for level values remain the same, no matter how many levels are used.

The following method has been adopted to work with the layer values of the variables, and still retain the advantage of continuous σ dependence. Suppose that, as an example,

$$v_{bL}^{M} = \sum_{k} v_{bL}^{M(k)} \sigma^{k} \qquad 0 \leq \sigma \leq 1 \tag{33}$$

Initially the V_{CL}^{M} are calculated from the initial conditions for $\bar{V}$. Suppose $\bar{V}$ is given in N layers. In that case V_{CL}^{M} is known at N different σ's. If V_{CL}^{M}, in a particular model, is to be represented by a Nth degree curve, V_{bL}^{M} in N + 1 layers have to be given initially to compute the coefficients $V_{bL}^{M(k)}$. If a boundary condition is given, say $V_{bL}^{M} = 0$ at $\sigma = 1$, then $\sum_{k} v_{bL}^{M(k)} = 0$ and N layer values are sufficient to calculate the constants. The boundary condition provides one more equation. From eq (33) the N equations are

$$v_{bLj}^{M} = \sum_{k} v_{bL}^{M(k)} \sigma_{j}^{k} \tag{34}$$

where j indicates the layer. The coefficients σ_j^k are the known pressures. The $v_{\ell L}^{M(k)}$'s can be calculated with the condition (in the example) that. $\sum v_{\ell L}^{M(k)} = 0$. The solution can be written as:

$$v_{\ell L}^{M(k)} = \sum_n S_{kn}^{(v)} v_{\ell Ln}^{M} \tag{35}$$

which is a linear combination of the N given layer values $v_{\ell Ln}^{M}$. The S has been given the superscript (v) because the boundary condition on $\bar{v}$ has been incorporated in the solution.

If $v_{\ell L}^{M}$ has to be differentiated, $\frac{\partial v_{\ell L}^{M}}{\partial \sigma}$ has to be expressed in the layer values. This can be done as follows: From eq. (34).

$$\left(\frac{\partial v_{\ell L}^{M}}{\partial \sigma}\right)_j = \sum_k k\, v_{\ell L}^{M(k)} \sigma_j^{k-1} \tag{36}$$

Substituting eq. (35) into eq (36) and interchanging the $\sum$ signs, gives:

$$\left(\frac{\partial v_{\ell L}^{M}}{\partial \sigma}\right)_j = \sum_n v_{\ell Ln}^{M} \left(\sum_k k\, S_{kn}^{(v)} \sigma_j^{k-1}\right) = \sum_n d_{jn} v_{\ell Ln}^{M} \tag{37}$$

where

$$d_{jn} = \sum_k k\, S_{kn}^{(v)} \sigma_j^{k-1} \tag{37a}$$

So the derivative of the dependent variable can again be expressed as a linear combination of its values in the N layers. As before all boundary conditions have been incorporated in the coefficients.

Integration gives no difficulty either.

Equation (34 gives:

$$\int_0^{\sigma} v_{bL}^{M} \, d\sigma = \sum_k v_{bL}^{M(k)} \frac{\sigma_j^{k+1}}{k+1} \tag{38}$$

Substituting eq (35) into eq (38) gives:

$$\int_0^{\sigma} v_{bL}^{M} \, d\sigma = \sum_i \mathcal{J}_{ji} \, v_{bLi}^{M} \tag{39}$$

where

$$\mathcal{J}_{ji} = \sum_k s_{ki}^{(v)} \frac{\sigma_j^{k+1}}{k+1} \tag{39a}$$

It should be pointed out that these coefficients are calculated once and need not be calculated again unless the model is changed to a different number of layers.

The method described above, has been applied to remove differentiation and integration in the σ coordinate in the equations describing the Mintz-Arakawa model. Though the latter is specifically a two-layer model (2 velocities), the vector spherical harmonics model will not be limited to a linearly or quadratically varying velocity.

For instance the hydrostatic equation

$$\frac{\partial \phi_L^M}{\partial \sigma} = -\frac{R T_L^M}{\sigma} \tag{16a}$$

Now

$$T_L^M = \sum_{k=0} T_L^{M(k)} \sigma^k \tag{40}$$

and

$$\sum_k T_L^{M(k)} = T_{LG}^M \tag{41}$$

is the boundary condition where T_{LG}^M can be calculated from the given ground temperature

$$T_G = \sum_{LM} T_{LG}^M Y_L^M \tag{42}$$

As before:

$$T_L^{M(k)} = \sum_i S_{ki}^{(T)} T_{Li}^M \tag{43}$$

The $S_{ki}^{(T)}$ are calculated by inversion of the matrix $\left\{\sigma_j^k\right\}$ of equation (40) with boundary condition eq (41) .

Substituting eq. (40) into eq (16a) results in

$$\int_1^\sigma \frac{\partial \phi_L^M}{\partial \sigma} d\sigma = \phi_L^M(\sigma) - \phi_{LG}^M = -R \sum_k T_L^{M(k)} \left(\frac{\sigma^k - 1}{k}\right) ; \tag{44}$$

$$\phi_L^M(\sigma) = \phi_{LG}^M + \sum_i \beta_{ji}^{(T)} T_{Li}^M \tag{45}$$

56

with

$$\beta_{ji}^{(T)} = R \sum_{k=1} \frac{s_{ik}^{(T)}(1-\sigma_j^k)}{k} \tag{45a}$$

There are N independent coefficients $T_L^{M(k)}$ so there are N layers values necessary.

Suppose that originally K values of T_L^M are given as the initial data. $K > N$. Now the ground temperature is given for all time. So equation (40) gives rise to K equations with the N unknowns $T_L^{M(k)}$ (one unknown is eliminated with the help of eq. (41)) Rather than integrating K layer equations, the K equations

$$T_{Lj}^M = \sum_{k=0} T_L^{M(k)} \sigma_j^k \tag{40a}$$

are solved with a least square method, giving

$$T_L^{M(k)} = \sum_i s_{ki}^{(T)} T_{Li}^M \tag{46}$$

N new "smoothed" values of the temperature are then used as initial values:

$$\tilde{T}_{Lj}^M = \sum_i T_{Li}^M \left(\sum_k s_{ki}^M \sigma_j^k\right) \tag{47}$$

$\tilde{T}_{Lj}^M$ will take the place of T_L^M in the previous discussion. Also the σ's at which the initial data are given need not be the same as the σ's at which the integration is performed.

Eq. (45) with eq (45a) takes the place of the hydrostatic equation. Next the equation (17a) for the ground pressure will be transformed.

$$\frac{\partial \pi_L^M}{\partial t} = \frac{\sqrt{L(L+1)}}{a} \int_0^1 \chi_{bL}^M \, d\sigma \tag{17a}$$

Again χ_{bL}^M is written as:

$$\chi_{bL}^M = \sum_k \chi_{bL}^{M(k)} \sigma^k \tag{48}$$

As with temperature, $\overline{V}$ is subjected to boundary conditions. The coefficients $x_L^{M(k)}$ must be calculated by inverting eq (48), adding the boundary conditions as extra equations. An example of this will be given later. For now suppose the solution is

$$\chi_{bL}^{M(k)} = \sum_i s_{ki}^{(\chi)} \chi_{bLi}^M \tag{49}$$

where the index i runs over the independent layers

Integrating eq (48) gives

$$\sum_k \int_0^1 \chi_{bL}^{M(k)} \sigma^k \, d\sigma = \sum_k \chi_{bL}^{M(k)} \frac{1}{k+1} \tag{50}$$

Substitution of eq. (48) into eq (50) results in:

$$\int_0^1 \chi_{bL}^M \, d\sigma = \sum_i \chi_{bLi}^M f_i \tag{51}$$

where

$$f_i = \sum_k \frac{s_{ki}^{(x)}}{k+1} \tag{51a}$$

The equation (17a) for the ground pressure becomes:

$$\frac{\partial \pi_L^M}{\partial t} = \frac{\sqrt{L(L+1)}}{a} \sum_i f_i \chi_{bLi}^M \tag{52}$$

$$= \frac{\sqrt{L(L+1)}}{a} \sum_i f_i \left[\left(\sum_{LM} \pi_L^M Y_L^M \right) \cdot (\bar{v}) \right]_{B_L^M} \tag{52a}$$

Equation (49) resulted from adding the boundary conditions to eq. (48) and solving the set of resulting equations. This is possible because eq (48) is a system of N-1 equations with N+1 unknowns. The boundary conditions add two more equation. For instance in the case of a given drag coefficient $\varepsilon' = \frac{1}{\varepsilon}$ the boundary condition on the surface is:

$$\varepsilon \left(\frac{\partial \bar{v}}{\partial \sigma} \right)_{S'} = - \bar{v}_S \tag{53}$$

At the earth's surface $\sigma = 1$, so π is not a function of σ. Therefore

$$\varepsilon \left(\frac{\partial (\pi \bar{v})}{\partial \sigma} \right)_{S'} = - \pi \bar{v}_S$$

$$\varepsilon \left(\frac{\partial \chi}{\partial \sigma} \right)_{S'} = - \chi_S$$

or in spherical harmonics coefficients:

$$\varepsilon\left(\frac{\partial \chi^{M}_{bL}}{\partial \sigma}\right)_S = -\left(\chi^{M}_{bL}\right)_S \qquad (54)$$

With the help of eq (48) this can be written as:

$$\varepsilon \sum_k k\, \chi^{M(k)}_{bL} = -\sum_k \chi^{M(k)}_{bL} \qquad (54a)$$

The extra equation is:

$$\sum_k (\varepsilon k + 1)\, \chi^{M(k)}_{bL} = 0 \qquad (55)$$

Together with $\frac{\partial \chi^{M}_{bL}}{\partial \sigma} = 0$ at $\sigma = 0$, which gives $X^{M(1)}_{b\,L} = 0$

This forms a system of (N+1) equations in (N+1) unknowns. The matrix $\left\{ S^{(\chi)}_{ki} \right\}$ is the inverse of the coefficient matrix of this system.

In this case there are (N-1) layers where $\bar{v}$ or $\bar{X}$ is calculated.

Example: (See Figure I)

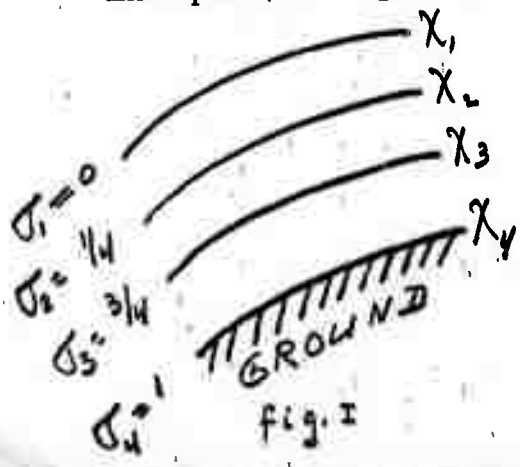

fig. I

Suppose that $X^{M}_{b\,L}$ is represented by a cubic equation

$$\chi^{M}_{bL} = \sum_{k=0}^{3} \chi^{M(k)}_{bL}\, \sigma^{k}$$

Let $\varepsilon = \frac{1}{40} = .025$. The four equations to determine the $X_{CL}^{M(k)}$ are:

$$\sum_{k=0}^{3} \chi_{CL}^{M(k)} \left(\tfrac{1}{4}\right)^k = \chi_{CL2}^{M} \quad ; \quad \sum_{k} \chi_{CL}^{M(k)} (1 + .025k) = 0$$

$$\sum_{k=0} \chi_{CL}^{M(k)} \left(\tfrac{3}{4}\right)^k = \chi_{CL3}^{M} \quad ; \quad \chi_{CL}^{M(1)} = 0$$

where χ_{CL2}^{M} and χ_{CL3}^{M} are measured values.

or in Matrix form:

$$\begin{pmatrix} 1 & \frac{1}{4} & \frac{1}{16} & \frac{1}{64} \\ 1 & \frac{3}{4} & \frac{9}{16} & \frac{27}{64} \\ 1 & 1 & 1.05 & 1.075 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \chi_{CL}^{M(0)} \\ \chi_{CL}^{M(1)} \\ \chi_{CL}^{M(2)} \\ \chi_{CL}^{M(3)} \end{pmatrix} = \begin{pmatrix} \chi_{CL2}^{M} \\ \chi_{CL3}^{M} \\ 0 \\ 0 \end{pmatrix} \qquad (56)$$

Inversion of the matrix in this equation gives the coefficients S_{k2} and S_{k3} where k = 0, 1, 2, 3. (Also, of course those of S_{ki}, and S_{k4}). During the computation the values of the χ's on the ground and top of the atmosphere need not be used. If it is necessary to obtain those values, they can be calculated from eq. (54) and (54a)

$$\left(\chi_{CL}^{M}\right)_S = -\varepsilon \sum_{k} \chi_{CL}^{M(k)} \qquad (57)$$

where the $\chi^{M(k)}_{\ell L}$ are found from eq. (56)

Furthermore:

$$\left(\chi^{M}_{\ell L}\right)_{TOP} = \chi^{M(0)}_{\ell L} \qquad (58)$$

Initially four or more $\chi^{M}_{\ell Li}$ might have been given

This gives rise to the least square problem

$$\sum_{k=0}^{3} \chi^{M(k)}_{\ell L} \sigma_i^{\,k} = \chi^{M}_{\ell Li} \qquad 1 \geqslant i \geqslant 4 \qquad (59)$$

to be solved under the conditions (54) and (55)

If the solution is $\chi^{M(k)}_{\ell L} = \sum s^{(\chi)}_{ki} \chi^{M}_{\ell Li}$, then the smoothed values $\tilde{\chi}^{M}_{\ell Li}$ to be used in eq. (56) are

$$\tilde{\chi}^{M}_{\ell Li} = \sum_{i} \chi^{M}_{\ell Li} \left(\sum_{k} s^{(\chi)}_{ki} \sigma_i^{\,k}\right) \qquad i = 2, 3$$

Integration of eq. (18a) goes as follows:

From eq. (51)

$$\int_0^{\sigma} d\sigma \int_0^{\sigma} \chi^{M}_{\ell L}\, d\sigma = \sum \chi^{M}_{\ell Li} f_i \sigma \qquad (60)$$

where the f_i are given by eq. (51a)

Further more

$$\int_0^{\sigma} \chi^{M}_{\ell L}\, d\sigma = \sum_{i} \chi^{M}_{\ell Li} \sum_{k} \frac{s^{(\chi)}_{ik}}{k+1} \sigma_j^{\,k+1} \qquad (61)$$

It follows that eq (18a) in spherical harmonics coefficients becomes:

$$\omega_j = \frac{1}{\pi} \sum_{LM} Y_L^M \left\{ \frac{\sqrt{L(L+1)}}{a} \left(\sum_i \chi_{\sigma Li}^M \vartheta_{ji} \right) \right\} \tag{62}$$

where:

$$\vartheta_{ji} = \sum_k \frac{s_{ik}^{(\chi)}}{k+1} \left(\sigma_j^{k+1} - \sigma_j \right) \tag{63}$$

Use is made that ω=0 for σ = 0

Also it should be kept in mind that π is still a function of (θ, φ). However, as before, if π is given in a sufficient number of points, the spherical harmonics coefficients ω_{Lj}^M can be obtained from eq. (62) without finite differencing.

To obtain the equation of motion in terms of spherical harmonics coefficients, three terms remain to be evaluated, $\frac{\partial v_{\sigma L}^M}{\partial \sigma}$, $\frac{\partial v_{\sigma L}^M}{\partial \sigma}$ and the friction term. First let

$$v_{\sigma L}^M = \sum_k v_{\sigma L}^{M(k)} \sigma^k$$

Then

$$\frac{\partial v_{\sigma L}^M}{\partial \sigma} = \sum_k k \, v_{\sigma L}^{M(k)} \sigma^{k-1} \tag{64}$$

As $\bar{v}$ and χ have the same kind of boundary conditions

$$v_{\sigma L}^{M(k)} = \sum_i s_{ki}^{(\chi)} v_{\sigma Li}^M \tag{65}$$

63

Substituting eq. (65) into eq (64) gives:

$$\frac{\partial v_{bLj}^{M}}{\partial \sigma} = \sum_i v_{bLi}^{M} \left(\sum_k k \sigma_j^{k} s_{ki}^{(X)} \right) \tag{66}$$

So the derivitives are again a linear combination of the (N-1) independently calculated layer values. In the same way

$$\frac{\partial v_{cLj}^{M}}{\partial \sigma} = \sum_i v_{cLi}^{M} \left(\sum_k k s_{ki}^{(X)} \sigma_j^{k} \right) \tag{67}$$

The friction term $\alpha \frac{\partial}{\partial \sigma}\left(\sigma^2 \frac{\partial \bar{v}}{\partial \sigma}\right)$ has the two components

$$\left[\alpha \frac{\partial}{\partial \sigma} \left(\sigma^2 \frac{\partial \bar{v}}{\partial \sigma} \right)_j \right]_{B_L^M} = \alpha \sum_i h_{ji} v_{bLi}^{M} \tag{68}$$

where

$$h_{ji} = \sum_k k(k+1) \sigma_j^{k} s_{ki}^{(X)} \tag{69}$$

and

$$\left[\alpha \frac{\partial}{\partial \sigma} \left(\sigma^2 \frac{\partial \bar{v}}{\partial \sigma} \right)_j \right]_{C_L^M} = \alpha \sum_i h_{ji} v_{cLi}^{M} \tag{70}$$

Using eqs (66), (67). (68), and (69) with eqs (15a) and (15c)

gives

$$\frac{\partial v^M_{bLj}}{\partial t} + \left[\bar{v}_j \cdot \nabla \bar{v}_j\right]_{B_L^M} + \frac{1}{a}\left[\left\{\frac{1}{\pi}\cdot\sum\sqrt{L(L+1)}\,Y_L^M\cdot\left(\sum_i \vartheta_{ji}\chi^M_{bLi}\right)\right\}\cdot\right.$$

$$\left.\left\{\sum_i d_{ji}\bar{v}_i\right\}\right]_{B_L^M} + \frac{1}{a}\sqrt{L(L+1)}\,\phi^M_{bL} + \frac{\sqrt{L(L+1)}}{a}\sum_i \beta_{ji}\,T^M_{Li} +$$

$$\frac{R}{a}\left[\frac{T_j}{\pi}\sum_{LM}\sqrt{L(L+1)}\,\pi^M_L B^M_L\right]_{B_L^M} - \frac{2i\Omega M}{L(L+1)}\,v^M_{bLj} +$$

$$2i\Omega\lambda^M_{L+1}\,v^M_{c,L+1,j} + 2i\Omega\lambda^M_L\,v^M_{c,L-1,j} = \alpha\sum_i h_{ji}\,v^M_{bLi} \qquad (71)$$

Reproduced from best available copy.

and:

$$\frac{\partial v^M_{cLj}}{\partial t} + \left[\bar{v}_j \cdot \nabla \bar{v}_j\right]_{C_L^M} + \frac{1}{a}\left[\left\{\frac{1}{\pi}\cdot\sum_{LM}\sqrt{L(L+1)}\,Y_L^M\left(\sum \vartheta_{ji}\chi^M_{bLi}\right)\right\}\cdot\right.$$

$$\left.\left\{\sum_i d_{ji}\bar{v}_i\right\}\right]_{C_L^M} + \frac{R}{a}\left[\frac{T_j}{\pi}\cdot\sum_{LM}\sqrt{L(L+1)}\,\pi^M_L B^M_L\right]_{C_L^M} - \frac{2i\Omega M}{L(L+1)}\,v^M_{cLj}$$

$$+ 2i\Omega\lambda^M_{L+1}\,v^M_{bL+1,j} + 2i\Omega\lambda^M_L\,v^M_{bL-1,j} = \alpha\sum_i h_{ji}\,v^M_{cLi} \qquad (72)$$

In a similar way the thermodynamic equation can be treated. This gives

$$\frac{\partial \theta^M_{TLj}}{\partial t} + \frac{1}{a}\left[\bar{v}_j\cdot\left(\sum_{LM}\sqrt{L(L+1)}\,\theta^M_{TLj}B^M_L\right)\right]_{Y_L^M} + \left[\frac{1}{\pi}\sum_{LM}\left\{\frac{\sqrt{L(L+1)}}{a}Y_L^M\cdot\right.\right.$$

$$\left.\left.\left(\sum_i \chi^M_{bLi}\vartheta_{ji}\right)\right\}\cdot\left\{\sum_{LM} Y_L^M\cdot\left(d_{ji}\,\theta^M_{TLi}\right)\right\}\right]_{Y_L^M} = \frac{1}{c_p}\left[\left(\frac{p_0}{\sigma\pi}\right)^{\kappa}\cdot\left\{\sum_{LM}\varrho_L^M Y_L^M\right\}\right]_{Y_L^M} \qquad (73)$$

65

with $\alpha_{ji} = \sum_k k \sigma_j^{k-1} S_{ki}^{(\theta)}$ where $S_{ki}^{(\theta)}$ is calculated, using the proper boundary condition on θ_T

In these equations terms such as ($V \cdot \nabla V$) and other non-linear terms are to be calculated with the transform method.

The equation contain only variables in the (N-1) layers, Eqs (71), (72), (73) together with eqs (52) have to be solved. They are ordinary coupled first order non-linear differential equation. The boundary conditions imposed on the variables θ and $\bar{V}$ are already incorporated in these equations and they have to be solved with the proper, given, initial conditions.

(1) Handbuch der Physik Band 48, Springis berlag, 1957

(2) Lectures on Numerical Short Range Weather Prediction, World Meteorological Organization, Hydrometeosdat, Leningrad 1969.

(3) A documentation of the Mintz-Arakawa Two-Level Atmospheric General Circulation model

W. L Gates, ES Batten, A. B. Kahle, and AB Nelson

(4) The Integration of a Low Order Spectral Form of the Primitive Meteorological Equations, A. J. Robert, Journal of the Meteorological Soc. Vol. 44, No. 5.

Angular Momentum

The angular momentum vector of a moving fluid of density and velocity V is given by

$$\vec{L} = \int \vec{r} \times \rho \vec{v} \, d\tau \tag{1}$$

where $d\tau$ is the volume element and $\vec{r}$ is the radius vector. To begin with consider the case of a sphere surrounded by a fictitions atmosphere of spherically symmetric density •

$\vec{V}$ can be expanded in the form

$$\vec{V}(r,t) = \sum_{M,L} \left[a_L^M(r,t) A_L^M(\theta,\varphi) + b_L^M(r,t) B_L^M(\theta,\varphi) + c_L^M(r,t) C_L^M(\theta,\varphi) \right] \tag{2}$$

Remembering that

$$A_L^M(\theta,\varphi) = \hat{e}_r Y_L^M(\theta,\varphi) \tag{3}$$

$$B_L^M(\theta,\varphi) = \frac{r}{\sqrt{L(L+1)}} \nabla Y_L^M(\theta,\varphi) \tag{4}$$

$$C_L^M(\theta,\varphi) = \frac{i\bar{r}}{\sqrt{L(L+1)}} \times \nabla Y_L^M(\theta,\varphi) \tag{5}$$

The contributions for these three kind of terms are then

$$\vec{L}_A = \int \rho\, a_L^M(r,t)\, \vec{r} \times \hat{e}_r A_L^M(\theta,\varphi)\, d\tau = 0 \tag{6}$$

$$\vec{L}_B = \frac{1}{\sqrt{L(L+1)}} \int \rho\, b_L^M(r,t)\, \vec{r} \times \nabla Y_L^M(\theta,\varphi)\, d\tau$$

$$= \int_{r_0}^{r_{max}} -\frac{i\rho(r)}{\sqrt{L(L+1)}}\, b_L^M(r,t)\, r^2 dr \int_0^{2\pi}\int_0^{\pi} C_L^M(\theta,\varphi) \sin\theta\, d\theta\, d\varphi \tag{7}$$

where

$$C_L^M = T_{LL}^M = \left[\frac{(L-M)(L+M+1)}{2L(L+1)}\right]^{1/2} Y_L^{M+1}\hat{e}_- + \frac{M}{[L(L+1)]^{1/2}} Y_L^M \hat{e}_0$$

$$-\left[\frac{(L+M)(L-M+1)}{2L(L+1)}\right]^{1/2} Y_L^{M-1}\hat{e}_+ \tag{8}$$

Consider the integral

$$\int_0^{2\pi}\int_0^{\pi} C_L^M(\theta,\varphi) \sin\theta\, d\theta\, d\varphi = \sqrt{4\pi} \int_0^{2\pi}\int_0^{\pi} Y_0^0(\theta,\varphi)\, C_L^M(\theta,\varphi) \sin\theta\, d\theta\, d\varphi$$

Using the orthonormality of the Y_L^M's the only non-zero terms for this integral must be those originating in C_L^M's which contain Y_0^0 terms. A look at equation (8) shows that there are no such terms.

Consequently $L_B = 0$.

$$\vec{L}_c = \int \vec{r} \times \rho(r) \sum_{M,L} c_L^M(r,t)\, C_L^M(\theta,\varphi)\, d\tau \tag{9}$$

$$= \sum_{M,L} \int \rho(r)\, c_L^M(r,t)\, \vec{r} \times \frac{i}{\sqrt{L(L+1)}} \left(\vec{r} \times \nabla Y_L^M(\theta,\varphi)\right) d\tau$$

where

$$\vec{r} \times \left(\vec{r} \times \nabla Y_L^M(\theta,\varphi)\right) = \vec{r}\left(\vec{r} \cdot \nabla Y_L^M\right) - \nabla Y_L^M r^2$$

$$= -r^2 \nabla Y_L^M(\theta,\varphi)$$

$$= -r\sqrt{L(L+1)}\, B_L^M(\theta,\varphi) \tag{10}$$

Thus

$$\vec{L}_c = -\sum i \int_{r_0}^{r_{max}} \rho(r)\, c_L^M(r,t)\, r^3 dr \int_0^{2\pi}\int_0^{\pi} B_L^M(\theta,\varphi) \sin\theta\, d\theta\, d\varphi \tag{11}$$

$$B_L^M(\theta,\varphi) = \left(\frac{L+1}{2L+1}\right)^{1/2} T_{LL-1}^M(\theta,\varphi) + \left(\frac{L}{2L+1}\right)^{1/2} T_{LL+1}^M(\theta,\varphi) \tag{12}$$

Again only terms in B_L^M containing Y_0^0 will contribute. Equations (6) and (8) in Section 1 show that these terms are

70

$$T_{10}^{-1} = Y_0^0 \hat{e}_- \; ; \; T_{10}^{0} = Y_0^0 \hat{e}_0 \, , \; T_{10}^{1} = Y_0^0 \hat{e}_+ \tag{13}$$

or

$$B_1^{-1} = \sqrt{\frac{2}{3}}\, T_{10}^{-1} \; , \; B_1^{0} = \sqrt{\frac{2}{3}}\, T_{10}^{0} \, . \; B_1^{1} = \sqrt{\frac{2}{3}}\, T_{10}^{1} \tag{14}$$

Thus we have reached the result that to the approximation that $\rho = \rho(r)$ the only terms capable of having a total angular momentum different from zero are:

$$c_1^{-1}(r,t)\, C_1^{-1}(\theta,\varphi) \, , \; c_1^{0}(r,t)\, C_1^{0}(\theta,\varphi) \, , \text{ and } \; c_1^{1}(r,t)\, C_1^{1}(\theta,\varphi)$$

Now, of course, two important approximations have been made: the earth was assumed spherical and the density was assumed to depend only on the distance from the center of the earth but not on the angles θ and φ. These are, however, pretty good approximations and to a considerable degree, then, the angular momentum of the atmosphere resides in the three terms found above. In particular the term $c_1^0(r,t)\, C_1^0(\theta,\varphi)$ is the main component of the zonal circulation.

$$C_1^0 = \frac{i\,\hat{e}_\varphi}{\sqrt{2}} \frac{\partial Y_1^0}{\partial \theta} = i\sqrt{\frac{3}{8\pi}}\, \hat{e}_\varphi \frac{\partial \cos\theta}{\partial \theta} = -i\sqrt{\frac{3}{8\pi}}\, \hat{e}_\varphi \sin\theta \tag{15}$$

71

The imaginary value does not matter, it simply means that $c_1^0(r,t)$ also has to be imaginary to make the velocity and the angular momentum real. The result implies that c_1^0 must be very stable since if we assume that at one instant of time the angular momentum only has a z-component, i.e. $c_1^1(r,t) = c_1^{-1}(r,t) = 0$, and the other velocity terms cannot absorb any total angular momentum, the law of conservation angular momentum will keep C_1^0 going. The next step is to remove the approximations $\rho = \rho(r)$ and to write

$$\rho = \rho(r,\theta,\varphi,t) = \sum_{M,L} \rho_L^M(r,t)\, Y_L^M(\theta,\varphi) \tag{16}$$

Now

$$\vec{L} = \int \vec{r} \times \rho \vec{V}\, d\tau$$

has terms of the form

$$\vec{L}_A = \sum_{L',M'} \int \vec{r} \times \sum_{L,M} \rho_L^M(r,t)\, Y_L^M \hat{e}_r\, Y_{L'}^{M'}(\theta,\varphi)\, d\tau$$

Again we get since $\vec{r} \times \vec{e}_r = 0$ that $\vec{L}_A = 0$

The terms $\vec{L}_B$, i.e.

$$\vec{L}_B = \sum_{L',M'} \vec{r} \times \left(\sum_{L,M} \rho_L^M(r,t)\, Y_L^M(\theta,\varphi) \frac{r}{[L'(L'+1)]^{1/2}} \nabla Y_{L'}^{M'}(\theta,\varphi) \right) d\tau$$

$$= -i \sum_{L',M'} \int r \sum_{L,M} \rho_L^M(r,t)\, Y_L^M(\theta,\varphi)\, C_{L'}^{M'}(\theta,\varphi)\, d\tau \qquad 72$$

We now see that unless $L=L'$ we get no contribution to L_B also unless $M=M'-1$, M' or $M'+1$ we get no contribution. Consequently the calculation becomes relatively simple. For L_c one gets the same kind of result. This concludes the discussion on angular momentum. Further work is in progress.

SECTION 7

PROGRAM DESIGN

CONTENTS

		Page
1.0	INTRODUCTION	1-1
2.0	PROJECT REQUIREMENTS TO BE SATISFIED USING PROGRAM SYSTEMS	2-1
2.1	Contractual Requirements	2-1
2.2	Research Requirements	2-1
2.2.1	Develop Models	2-1
2.2.2	Develop Initialization Techniques	2-1
2.2.3	Eliminate Radial Dependence	2-2
2.2.4	Develop Computational Efficiency	2-2
2.2.5	Determine Accuracy	2-2
3.0	PROGRAM SYSTEM FUNCTIONAL REQUIREMENTS	3-1
3.1	Integration	3-1
3.2	Initialization	3-2
3.3	Output	3-2
3.4	Table Generation	3-2
3.5	Flexibility	3-2
3.5.1	Model Changes	3-3
3.5.2	Algorithm Changes	3-3
3.5.3	Experimental Changes	3-3
3.5.4	Initialization Changes	3-4
3.5.5	Output Formats	3-4
4.0	PROGRAM SYSTEM OBJECTIVES	4-1
4.1	Baseline and Particular Systems	4-1
4.1.1	Baseline System	4-1
4.1.2	Particular Systems	4-1
4.2	Orderly Growth	4-2
4.3	Computer Requirements	4-3
4.4	Parameterization	4-3
5.0	BASELINE SYSTEM DEVELOPMENT PLAN	5-1
5.1	Structured Programming	5-1
5.2	Subsystem Structure	5-2
5.2.1	The GO Subsystem	5-3
5.2.2	The BUILD Subsystem	5-3
5.2.3	The RESTORE Subsystem	5-4

		Page
5.3	Subsystem Execution Control	5-4
5.4	Baseline System Development Procedure	5-6
6.0	BASELINE SYSTEM DESIGN	6-1
6.1	The GO Subsystem	6-1
6.1.1	GO Data Sets	6-2
6.1.2	GO Subsystem Components	6-5
6.2	The BUILD Subsystem	6-7
6.2.1	BUILD Data Sets	6-10
6.2.2	BUILD Subsystem Components	6-10
6.3	The RESTORE Subsystem	6-13
6.3.1	RESTORE Data Sets	6-13
6.3.2	RESTORE Subsystem Components	6-15
6.4	Data Set Summary	6-15
Appendix A	PRECOMPILATION CAPABILITIES USING A PRECOMPILER IN A PROGRAM PRODUCTION ENVIRONMENT	A-1

1.0 Introduction

IBM has developed analytic methods for applying Vector Spherical Harmonics (VSH) to the global weather prediction problem. This technique shows promises of advantages in computational speed and accuracy over current methods.

The Weather Systems Programming department is currently engaged in developing a program system to investigate and demonstrate these advantages. This work is supported by IBM internal funding (IRAD) and a contract with the Advanced Research Projects Agency (ARPA).

The IRAD funds cover general investigation into the advantages of VSH methods, while the ARPA contract requires implementation in VSH of a specific weather model - the Mintz-Arakawa model. A general program system has been designed to address both of these objectives. The ARPA contract is being charged for development of those portions of the system necessary to implement the Mintz-Arakawa model. IRAD funds are being used for development of the general system. The ARPA Contract could be performed even if the IRAD task were terminated. However, the performance of the ARPA contract will be significantly enhanced as a result of the IRAD task.

It may be of interest to review the evolution of the program system. The major requirement is to numerically solve models expressed in VSH. This involved a "predictor" that integrates differential equations over time. Accordingly, the contents of the predictor were investigated in detail. These included:

- Transformations between physical and spectral space including evaluation of differential operators in physical space.
- Application of numerical integration techniques to solve the differential equations in spectral space.
- Preparation of an initial state from raw data
- Output capabilities
- Calculation of vector coupled sums via analytic forms

VSH provides a method for solving an arbitrary set of differential equations in spherical coordinates. It was desired to retain this generality in the predictor.

Accordingly, an extremely general and flexible program for solving an arbitrary set of partial differential equations in spherical coordinates (e.g. a weather model) was designed to the level of detail needed for coding. This program is highly parameterized. Choices of code modules to be

included, sizes of working storage arrays, computational loop limits, and data set format specifications in the job control language can all be specified via parametric expressions.

This report discusses the following topics:

- Project requirements that influence the program system design
- Program system functional requirements & design objectives.
- A development plan for implementing and modifying the program system.
- The current state of the program system design

2.0 PROJECT REQUIREMENTS TO BE SATISFIED USING PROGRAM SYSTEMS

2.1 Contractual Requirements

The current ARPA contract requires a test computation using Vector Spherical Harmonics (VSH) of a simplified version of the Mintz-Arakawa Model.

2.2 Research Requirements

The program system will be used to support the continuing research into VSH methods, properties and advantages - both theoretical and computational. The following sections discuss particular research areas where programming will be used.

2.2.1 Develop Models

VSH permits inclusion of expressions in the differential equations (models) hitherto neglected or simplified because of computational complexity. Solutions of models incorporating more exact and complete descriptions of physical phenomena are thus possible. More accurate models for weather prediction and climatology can be developed without being unduly constrained by computational difficulties.

2.2.2 Develop Initialization Techniques

Weather data collected on a global scale is not of consistent accuracy and completeness. This is particularly true of wind data. This inconsistency is further compounded by the nonsynchronous nature of satellite collected data. Initialization of a global predictor requires resolution of this data into a complete sample of greatest possible accuracy. VSH techniques will be developed to produce this initial condition.

2.2.3 Eliminate Radial Dependence

Application of VSH to partial differential equations (PDE's) in spherical coordinates produces PDE's with radius and time as the independent variables and VSH expansion coefficients as the dependent variables. Methodology will be developed to transform these PDE's into time dependent ordinary differential equations. The transformations will be done by expanding each VSH coefficient in suitable radial functions.

2.2.4 Develop Computational Efficiency

Application of VSH to a complex predictive model produces a tractable and computationally efficient formulation. Algorithms employed in areas such as transformations and integration as well as data management techniques will be developed to further enhance the computational efficiency of the predictions.

2.2.5 Determine Accuracy

Prediction accuracy and computational speed depend on a combination of:

a. The model used

b. The quality of the initial conditions

c. The truncation limits of VSH expansions

d. The form and order of radial expansions

e. The computational algorithms employed

The sensitivity of predictive accuracy to choices in the above areas, as well as the effects of coupling among these areas, will be determined.

3.0 PROGRAM SYSTEM FUNCTIONAL REQUIREMENTS

3.1 Integration

The major function of the program system is to numerically solve the model equations using VSH methods. This involves a prediction that generally

a. Expands physical variables in vector and scalar spherical harmonic coefficients and transforms harmonic coefficients to physical variables

b. Computes linear and nonlinear terms in differential equations to form derivatives

c. Integrates over time in VSH coefficients to produce predictive results.

The exact method of integration to use is not known at this time. One of the research objectives is to find the best applicable integration method. Numerical integration methods are generally divided into two classes:

a. Explicit, or open, methods in which the state at a given time is computed as a linear combination of state and derivative values at previous time points.

b. Implicit, or closed, methods which are the same as explicit methods with the addition of some multiple of an estimate of the derivative evaluated at the same time point at which the state is being computed.

The integration method may be a single step, where only the state and derivative values from one past time are needed, or a multi-step, where state and derivative values from several previous times are required. A requirement for integration is the computation of derivatives in spectral space.

The elements of spectral derivative computation are:

a. Computation of radial derivatives of spectral variables, which is itself a research area.

b. Formation of differential terms in physical space from spectral variables and differential terms.

c. Transformation of the differential equations in physical space to corresponding differential equations in spectral space.

3.2 Initialization

The predictor requires an initial state of known data as a starting point for its prognostic computations. Derivation of this initial state from raw data is also a functional requirement of the programming system. Current techniques are generally regarded as unsatisfactory in producing a consistent, accurate initial sample. One of the project requirements is to investigate the application of VSH methods to this problem. This in itself is a major research area which will require the ability to investigate and implement alternative spectral transforms, computational algorithms and mixes of raw data. This may be done in several stages:

a. Artificially generate an initial state

b. Apply VSH to existing initialization methods

c. Implement currently unknown VSH methods.

3.3 Output

Data from the predictor must be output to inform the research. Likewise, information from the initialization, in addition to the initial state for the predictor, must be output. Further, benchmark comparisons and contractual requirements may necessitate particular content and format of output.

3.4 Table Generation

Data used in the prediction, initialization and output will, whenever possible, be precomputed to enhance computational speed.

There will be a large amount of data in the tables (possibly several mission quantities) and computing them is a major task. The tables generally do not change with every predictor run. In fact, once an operational model configuration were chosen, they would not change at all. They may thus logically be computed and written in a program separate from the predictor.

3.5 Flexibility

Overall flexibility is a functional requirement since the major research areas require the ability to make changes in every segment of the program system, from the highest level to the lowest.

3.5.1 Model Changes

Models are sets of PDE's in spherical coordinates and their boundary conditions. The program system must be able to accept the following types of model changes which are enumerated in order of complexity.

a. Execution time parameter changes (e.g., assign values to the symbolic coefficients that appear in the model equations)

b. Reformulation of an existing equation (e.g., addition of terms to more closely approximate the physical system)

c. Modification of boundary conditions (e.g., change a constant boundary condition to a function of time)

d. Add equations (e.g., explicitly model the dynamics of a physical variable hitherto assumed constant)

e. Change physical system (e.g., model seismological phenomena).

3.5.2 Algorithm Changes

The computational algorithms used to implement the various transformations, differential equation component calculations and integration are themselves the subject of investigation as to accuracy and efficiency.

Possibly several candidate algorithms will be tried in a given application and modifications to specific algorithms will be made. This implies the need for ease of replacement and modification of algorithmic components of the programming system.

3.5.3 Experimental Changes

The VSH method expands physical variables in spectral functions of latitude and longitude and currently unknown functions of radius. The program system must support experimental changes of the following forms in these expansions:

a. Number of terms in spectral expansion

b. Functions to be used in radial expansion

c. Number of terms in radial expansion.

3.5.4 Initialization Changes

There are three main initialization objectives

a. Apply VSH to efficiently produce an initial sample of greatest possible consistency and accuracy. This sample will contain the variables necessary to initialize our model.

b. Accept a variety of input data and formats and transform them to the form necessary to initialize our model. This is particularly true when performing benchmark comparisons.

c. Experiment with the quantity of initialization data necessary for accurate prediction.

These objectives imply continuing changes and additions in the initialization section of the program system.

3.5.5 Output Formats

There are two types of output:

a. General predictive results describing the weather state. The form and content is generally well defined.

b. Data to illuminate particular points of interest in research. The form and content is usually defined - and redefined - as experimentation proceeds.

Also, the content and format of output will be determined by

a. Internal Use

b. Contract Requirements

c. Benchmark Comparisons.

Varieties of data must be aggregated and output in desired formats; plots of various types must be produced. The data output may be raw or processed input data, tables of coefficients used in the computations, the results of intermediate steps in the predictions, or the predictions. In many cases the exact nature of the output will be defined after the baseline system structure has been built (changes and additions will be required to satisfy previously undefined or unanticipated requirements). The output section must be able to satisfy these data needs, especially those generated by specific research questions, contract requirements and benchmarks.

4.0 PROGRAM SYSTEM OBJECTIVES

The program system must be developed to meet two major objectives

a. To provide a general and flexible testbed for investigation in the major research areas.

b. To provide, when necessary, the most efficient overall program for the solution of a given problem.

4.1 Baseline and Particular Systems

4.1.1 Baseline System

The first objective leads to a more elaborate and perhaps less efficient overall structure than might be attainable if certain segments of the system could be made less general. However, a general structure is necessary since the major areas for research impact every aspect of the program system and require maximum flexibility for satisfactory support and response. The format of the model equations, the boundary conditions, the radial fitting functions and the initializing process are all certain to be changed several times in the search for maximum accuracy and fidelity to physical reality. Algorithms and program parameters for such things as transformations, derivative calculations and integration, are bound to be altered and replaced to enhance accuracy and computational efficiency. Therefore a system must be designed to accept changes, both trivial (integration time step) and fundamental (integration method) without undue effort and delay. We will call this a "Baseline System," in that it provides the framework in which various alternatives can be easily implemented and investigated.

The Baseline System must provide a flexible and easily changeable testbed for experimentation in the major research areas. It must serve as the development ground and checkpoint for particular systems.

4.1.2 Particular Systems

Generality serves the overall research objectives, but there may be occasions--baseline comparisons, contract requirements, promising research configurations when the maximum overall program efficiency is desirable.

The information gained from research using the baseline system will allow a given configuration of model equations and computational algorithms

to be modified from the baseline framework. It would be made computationally efficient and concise as an overall system in its own right. This would be a "Particular System," and could exist independent of the baseline.

The Baseline System is meant to provide the framework for experimentation and investigation. Within the constraints of the necessary generality this implies, the elements of the Baseline System are to be implemented as efficiently as possible. There may, however, be instances such as benchmark comparisons or implementation contracts when the overall system efficiency must be as great as possible. In these cases a Particular System will be built, using and modifying elements of the Baseline System, in order to eliminate the inefficiencies imposed by generality.

4.2 Orderly Growth

In the course of the research many alternatives will be investigated in all of the major research areas, such as models, initialization and algorithms. The best choice in a given instance may not be the first or the last candidate tried. The exact implementation configuration will vary from run to run, and each will have its advantages and disadvantages. This implies several growth objectives for the Baseline System.

a. Documentation - The status of the system should be recorded in design documents, reports, program listings, run results, and program libraries in a coherent and consistent fashion to make the systems both visible and accessible.

b. Maintenance of the system - Additions and modifications must be made in an orderly and consistent fashion, and records maintained.

c. Change by expansion - All changes to algorithms or additional algorithms will be additive. Compile time or execution time switches will allow choice of desired alternatives. In this way, the Baseline System retains all its past capabilities, and they are easily accessible.

d. Performance evaluation - The merits of a particular configuration should be measurable, and consistent comparisons between particular configurations should be possible, within the framework of the baseline system. The need to implement particular systems as entities separate from the baseline should be minimized.

4.3 Computer Requirements

Global Weather Prediction requires enormous amounts of data, both initial and intermediate. Further, the data requirements are dynamic depending, for example, on spectral expansion truncation values. It is infeasible to design an experimental system with the assumption that all requisite data will be core resident throughout the prediction computations. It may be that there are large machines with sufficient core capacity for such a system, but ready and heavy access to such a machine may be difficult. A better objective is to design the Baseline System with minimum core requirements. This implies heavy use of direct access peripheral storage devices and intermediate input/output (I/O) processing. However, the range of machines that can be used is considerably expanded. Also, the emphasis on efficiency at this time is in the computational aspects of prediction and not on input/output. If a particular system looks promising and can be configured to be core resident on some existing machine, this can easily be done. Another objective is to design the Intermediate I/O and attendant data set formats of the Baseline System so that intermediate I/O in a particular system can be made as efficient as possible.

4.4 Parameterization

The program system development must account for implications of the program system functional requirements and objectives. Flexibility of the system implies that there will be a large number of parameters which must be assigned values for system execution. These include spectral expansion truncation values, integration time step and time limit, and output time interval.

Minimum core use implies that the amount of core required for a given execution will depend on execution time parameters such as expansion truncation values. Moreover, sizes of working regions in various algorithms will also depend on these execution time parameters.

Flexibility in Models and algorithms and generality of input and output mean source code modifications and additions. Orderly growth dictates that these changes be additive whenever possible. Compilation and link editing of a system configuration thus involves a choice of source code modules and paths within modules. This, along with the core configuration is essentially a parameterization of the source code structure.

Minimum core also dictates tables of precomputed coefficients and intermediate data storage on direct access data sets. For space and execution efficiency, details of these data set formats such as record length and blocksize are dependent on certain execution parameters, such as spectral expansion truncation limits. This is a parameterization of the data control block (DCB) information of the job control language (JCL) specifications of these data sets.

In summary, there are a large number of parameters necessary to define a given system configuration. They are:

a. Execution time parameters, such as spectral expansion truncation limits, some of which determine compile time parameters

b. Compile time parameters, such as program segments and core segmentation, some of which are functions of execution time parameters.

c. Data set definition parameters, such as record length, some of which are functions of execution time parameters.

It is desirable to describe a given system configuration by specifying as few of these parameters as possible. This can be done by determining those parameters that are independent and used in specifying other parameters. It should be necessary to specify only those parameters. Algorithms should be built into the system to determine values for the other dependent parameters, and to construct the code segments and JCL automatically.

This can be done through the use of precompilation facilities described in the Appendix. Briefly, this involves the PL/1 precompilation facility operating on FORTRAN and JCL - as well as PL/1-code. Parametric expressions in the code are evaluated to determine:

a. Segments of code to be included in the compilation.

b. Storage allocations.

c. DCB specifications in the JCL.

This implies that the system must be recompiled when execution time parameters which affect precompilation parameters change.

Also, the iterative nature of the research and the number of parameters in the system add an additional constraint. A particular version of the system, with its execution and precompilation parameters set, must be archived and capable of easy restoration.

5.0 BASELINE SYSTEM DEVELOPMENT PLAN

Because of its generality and size, a coherent plan for the development of the Baseline System is necessary. The plan and the implementation procedure should allow the system structure to be visible and easily modifiable during its construction. The system should be built so that changes at one stage of the implementation do not cause extensive modification of previously written code.

5.1 Structured Programming

Baseline System design and development will use a technique called structured programming in order to maintain visibility, consistency and efficiency--both in implementation and execution. The basic tenet of structured programming is that computer programs can be designed and written with a high degree of structure, which permits them to be more easily understood for testing, maintenance, and modification.

To further enhance the readability, visibility and coherence of the code, a repetitive process of code segmentation is undertaken. The first step in this process is to formulate a one-page skeleton program which represents the entire program. This is done by selecting some of the most important lines of code in the program and then filling in what lies between those lines by names. Each new name will refer to a new segment to be stored in a library and called by a macro facility. In this way, we produce a program segment with something under 50 lines, so that it will fit on one page.

Now, the segments at the next level can be written in the same way, referring as appropriate to segments to be later written (also setting up dummy segments as they are named in the library). As each dummy segment becomes filled in with its code in the library, the recompilation of the segment that includes it will automatically produce new updated, expanded versions of the developing program.

This leads to a top down system implementation. The system is created in execution sequence. Only instructions which can be executed as they are created are coded, all prior instructions required have already been coded. In the top-down approach, the highest level code is written first, then the next lower level code, etc., down to the lowest levels of code.

Under OS/360, for example job control, linkage editor, "supervisory" and "data management" code is written in that order, and only then the source modules which typically give a system its functional capability.

Thus, development proceeds through the controlled addition of new modules to an always checked-out program. That is, supervisory programs run early in the development phase, first calling on dummy modules for which functional modules are substituted later.

To augment the ordered implementation of the top-down approach, record keeping and program module library procedures must be formatted and followed. In the structured programming approach, this is done via the Programming Production Library (PPL).

The PPL is used to maintain the current status and past history of all code generated in a project in a human readable form, except as specifically noted to the contrary. The current status of the PPL will include, at least, separate Library Sections which represent the following six classes of code and data:

a. Job Control Source Code

b. Linkage Editor Source Code

c. Program Source Code

d. Object Module Code

e. Load Module Code

f. Test and Control Data

These libraries generally reside in partitioned data sets on direct access storage devices.

Standard PPL procedures exist to perform the following functions:

a. Add/Delete/Modify any Source Member

b. Assemble/Compile any Object Member

c. Linkedit any Load Member

d. Execute Debug/Test Program.

5.2 Subsystem Structure

In order to best satisfy the research requirements and program system functional requirements and objectives, the baseline system will be segmented into three subsystems.

5.2.1 The GO Subsystem

This subsystem will include

a. Integration of the model (prediction)

b. Initialization for prediction

c. Output

d. Table generation for prediction, initialization, and output.

Each of the above functions - initialization, table generation, prediction, output - addresses various project requirements. Each function contains many options and the functions may be combined in various ways depending on what is desired. The GO subsystem will therefore consist of each function implemented as a separate program module. The execution of each function in turn, and the options chosen in each function, will be controlled by parameters specified by the user, or constructed from user specified parameters.

5.2.2 The BUILD Subsystem

The purpose of this subsystem is to construct an executing version of the GO subsystem. It automates to the greatest extent possible the high degree of parameterization present in the GO subsystem.

The BUILD subsystem will do the following:

a. Use precompilation facilities to include code segments and arrange working storage.

b. Construct JCL for executing the GO subsystem and insert it in Procedure Library.

c. Compile and Link edit Input, Table Generation, Predictor, Output load modules.

d. Construct control parameter data records to select options within the modules and choose the modules to be executed.

e. Archive the GO subsystem including source and object code, load modules, JCL and control parameter data records.

5.2.3 The RESTORE Subsystem

The purpose of this subsystem is to restore a GO subsystem configuration, complete with JCL and control parameter records, that was archived by the BUILD subsystem. Only that portion of the GO subsystem necessary for execution will be restored.

The following will be done:

a. Restore the Input, Table Generator, Predictor and Output load modules to the direct access load library.

b. Restore the GO JCL procedure to the procedure library for subsequent execution.

c. Restore the control parameter data records to the control parameter direct access data set.

5.3 Subsystem Execution Control

Execution control within a subsystem and communication between the subsystems are implemented by the Control Parameter Preparation (CPP) program and an associated direct access data set - the CPP data set.

The CPP data set consists of various types of records:

a. System Master Record - Major record for subsystem communication, contains:

   1. A code identifying the system configuration for archiving purposes.

   2. Indicators for GO subsystem module execution inclusion

   3. Pointers to compile time parameter records to be moved into the subsystem master records at build time

b. GO Subsystem Master Records - One each for Input, Predict, and Output modules. Identifies all requisite parameters for each function.

   1. Compile time parameters

   2. Pointers to data records containing execution time parameters.

c. Parameter Records - Contain either execution or compile time parameters for the GO subsystem modules.

d. Working Records - Used by CPP program for CPP file maintenance.

The major function of the CPP program is to maintain the CPP file. Data records are created or updated depending on input parameters. Master record pointers are updated to indicate desired data records. These functions are performed in the BUILD and GO subsystems.

In addition, the tasks performed by the CPP program in each of the subsystems are:

a. GO

1. Checks CPP master and data records for completeness and validity

2. Sets an operating system condition code to control execution of desired GO Subsystem modules.

3. Marks CPP master record to indicate that tables have been generated or copied.

b. BUILD

1. Checks input parameters for validity and completeness

2. Updates the CPP Master record according to input parameters

3. Marks the system master record "No Go" until BUILD is complete

4. Marks the CPP master record for table generation in the GO Subsystem

5. Constructs an input data set of utility control statements for the archiving step.

c. RESTORE

1. Marks the system master record as "No Go" until restoration is complete

2. Constructs an input data set of utility control statements for the step that restores from the archive data set

3. Marks the system master record for the GO Subsystem to copy tables from the archive data set.

4. Updates the CPP master record from the archive tape.

5.4 Baseline System Development Procedure

Figure 5-1 presents the procedure by which the Baseline System will be developed. Mechanisms are provided for system generation and subsequent revisions.

There is a three level classification for Baseline Systems:

a. Version

Recompilation with different compile time parameters. No new source code, but possible corrections to existing code

b. Extension

Modification addition of new options (additive growth). The key is a CPP program addition to recognize and manage the new options.

c. System

A new system has been generated when the following cannot be done

1. Allow any restored version to execute

2. Rebuild any earlier version with suitable input parameters

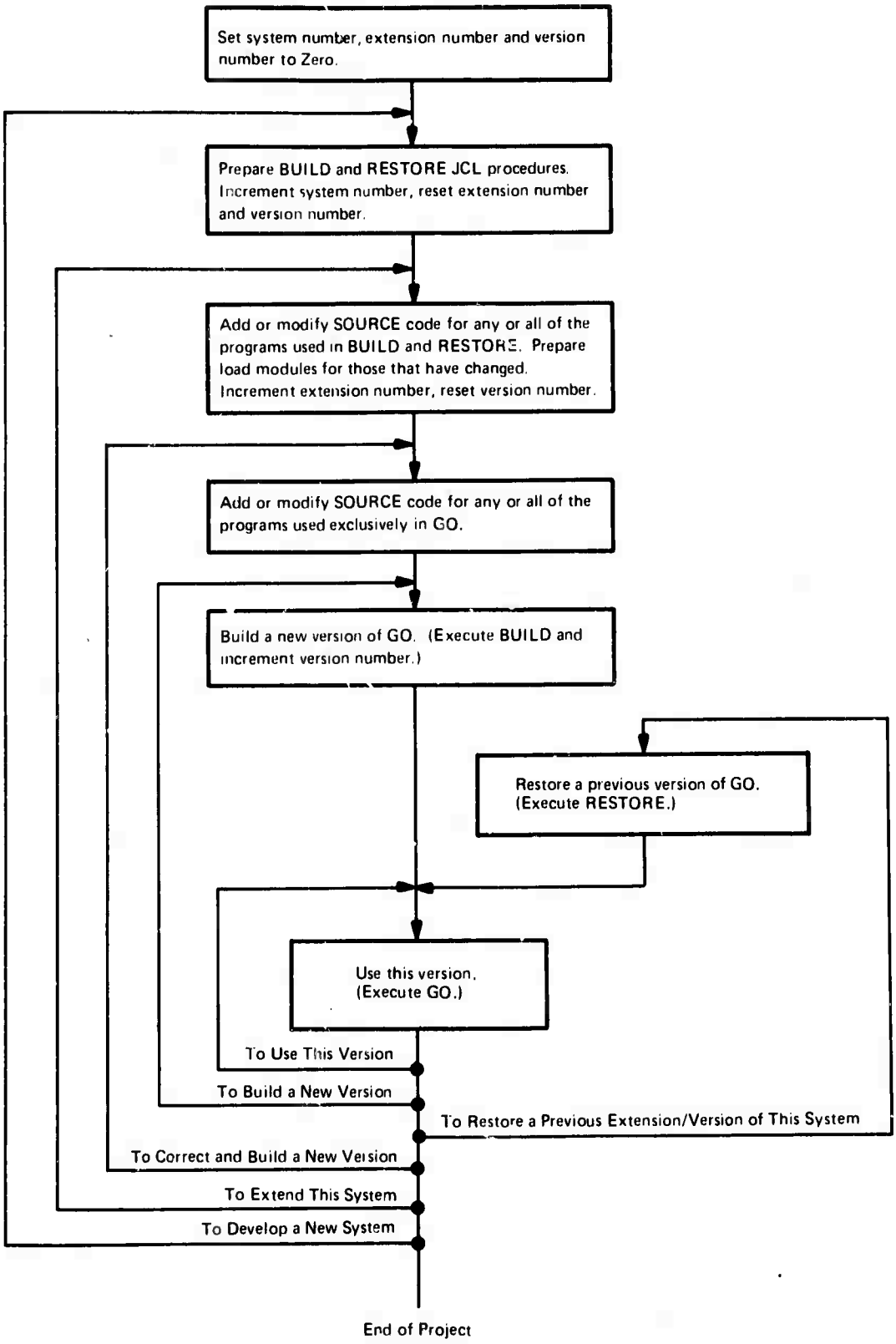

Figure 5-1. System Development Procedure

6.0 BASELINE SYSTEM DESIGN

This section describes in detail the three subsystems of the Baseline System. The CPP program as described in Section 5.4 is referenced in each subsystem discussion. The data sets and their use in each subsystem is summarized.

6.1 The GO Subsystem

The major function of the GO Subsystem is to produce predictive estimates of the weather. This involves four functions:

a. Table Generation - Compute and write data sets of constants used in the GO Subsystem. These data sets may optionally be copied from archived data. This function is included in the GO Subsystem, rather than the BUILD (or RESTORE) subsystem, because the JCL describing the table data sets depends on precompilation parameters. This JCL is built as part of the GO procedure.

b. Input - Produce an initial sample for prediction from raw data. This may be done in several ways:

   1. Artificially generate an initial state

   2. Apply VSH to existing initialization methods

   3. Implement currently unknown VSH methods

c. Prediction - Integrate the model equations. This involves numerical integration of the model equations expressed in VSH. The integration method is the subject of research. Derivatives in VSH must also be computed for the integration.

d. Output - Produce output concerning any of the above functions. Tailor output to meet benchmark specifications and contract requirements.

The GO Subsystem will consist of load modules for each of the above functions as separate job steps. These load modules will be produced by the BUILD Subsystem discussed in Section 6.2.

There are many execution options possible among the four functions. For example, the Input module might process some raw data and pass the results to the Output module for plotting. The Table Generator and Predictor modules are not executed.

The execution of a given function will be controlled by operating system condition codes. The CPP program will set the condition code indicating execution of any combination of the modules. The modules desired will be indicated via input. CPP will also perform a validity and completeness check on the CPP data set after it has processed any input parameters. If validation fails, CPP sets a job condition code of 99, which terminates execution. The functional flow of the GO Subsystem with attendant data sets is shown in Figure 6-1. The data sets are discussed in the following section, and the functions are discussed in detail in subsequent sections.

6.1.1 GO Data Sets

6.1.1.1 TABLES

Precomputed constants are used in the Input, Predictor and Output modules. They reside on a number of TABLES data sets constructed by the Table Generation Module. Unless otherwise noted they are direct data sets.

a. TABLE CONTROL - Sequential data set containing copies of the CPP Master records at table construction time.

b. PREDINIT - Sequential data set containing various constants that will reside in core during Predictor Module execution. They are tables defining radial fitting functions, frequently used Clebsh-Gordan coefficients and constants used in the Fourier step of the transformation from spectral to physical space.

c. RADIAL ONE - Constants used in constructing first order radial differential operators of spectral coefficients.

d. RADIAL TWO - Constants used in constructing second order radial differential operators of spectral coefficients.

e. PS LEGENDRE - Constants used in the Legendre step of the transformation from physical to spectral space.

f. PS FOURIER - Constants used in the Fourier step of the transformation from physical to spectral space.

g. VS REAL - Constants used in transforming between VSH and Scalar Spherical Harmonic (SSH) real components.

h. VS COMPLEX - Constants used in transforming between VSH and SSH complex components

i. SP LEGENDRE - Constants used in the Legendre step of the transformation from Spectral to physical space.

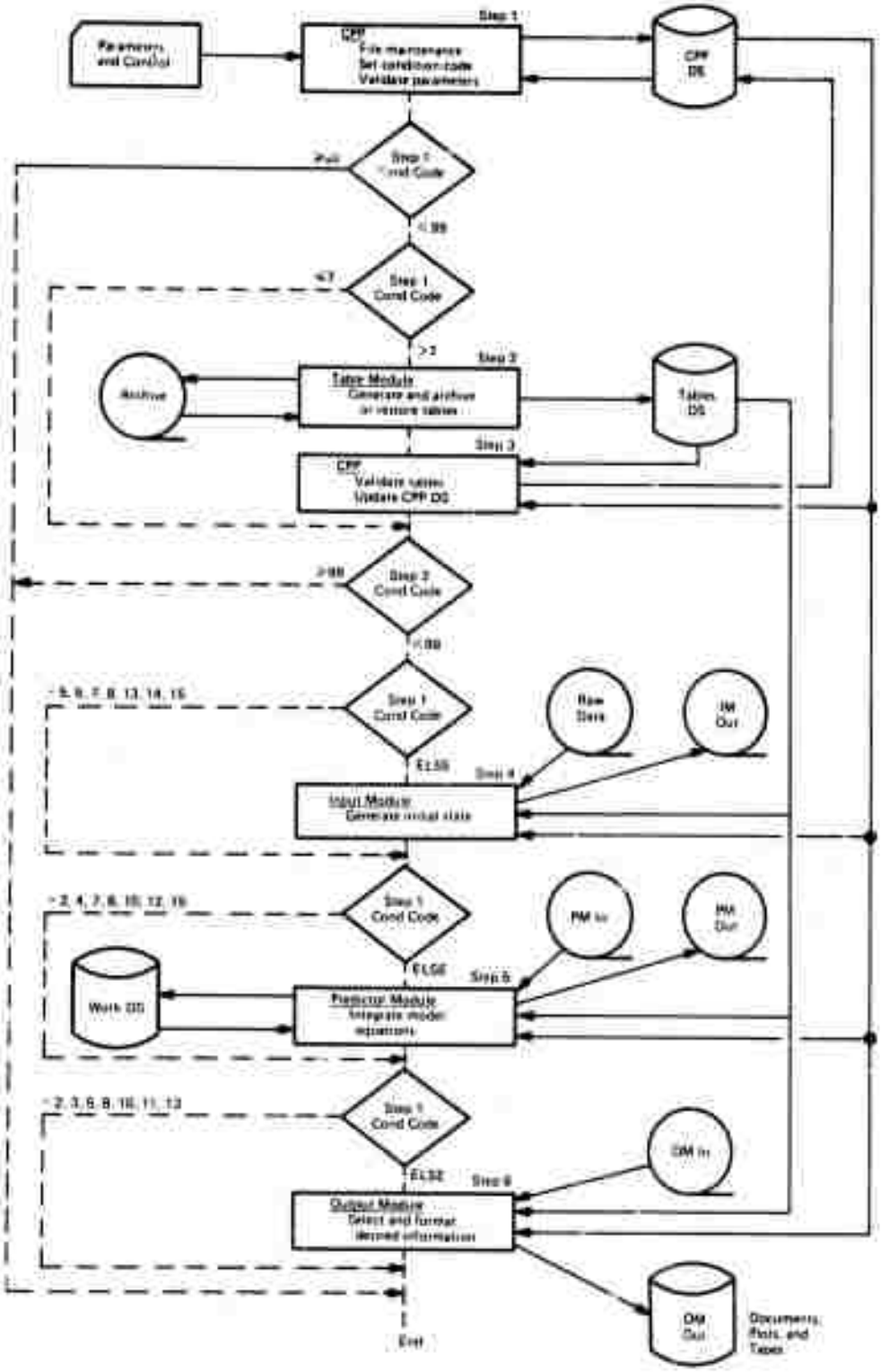

Figure 6-1. GO Subsystem

6.1.1.2 ARCHIVE3

Sequential data set which contains a copy of the TABLE data sets. The Table Generation Module writes this file when it builds tables, and reconstructs them from this file when a system has been restored.

6.1.1.3 RAW DATA

One or more data sets containing weather observations. The form and content of these data sets is not currently known. This will be the primary input to the Input Module.

6.1.1.4 IM OUT

Sequential data set, containing the initial conditions needed by the Predictor Module. These conditions will be expressed in physical space and in spectral space. The format of this data set is the same as that of the PM OUT data set.

6.1.1.5 PM IN

Sequential data set, containing initial conditions needed by the Predictor Module. This is either a previously computed IM OUT or PM OUT data set.

6.1.1.6 PM OUT

Sequential data set containing Predictor Module output. It consists of predicted values of the physical and spectral variables at selected times. The format of this data set is the same as that of the IM OUT data set.

6.1.1.7 OM IN

Sequential data set, containing data to be processed by the Output Module. This is either a previously computed IM OUT or PM OUT data set.

6.1.1.8 OM OUT

One or more data sets containing GO Subsystem results in user specified format (e.g., tables, plots, tapes, etc.). The form and content of these data sets is not currently known. This will be the primary output of the GO Subsystem.

6.1.1.9 Work Data Sets

Used to extend Predictor Module storage without undue use of core. They are all direct, temporary data sets and exist only during Predictor execution.

There are five data sets associated with each physical prognostic variable and its spectral coefficients. They are:

a. P VALUE - The value of the variable in physical space.

b. P DIFF - The value of spatial differential operators applied to the physical variable.

c. S VALUE - The value of the variable in spectral space at one or more time instants.

d. SDR - The value of the first and second radial partial derivatives of the spectral coefficients for this variable.

e. SDT - The value of the temporal partial derivatives of the spectral coefficients for this variable at one or more time instants.

There is also one data set for all the diagnostic variables indicated in the model.

PD VAR - The value of the diagnostic variables in physical space.

6.1.2 GO Subsystem Components

6.1.2.1 Table Generation Module

This module performs one of two functions depending on the status of the table flag in the System Master Record of the CPP data set:

a. Build and write the TABLE data sets if table build is indicated. This also involves archiving the Tables or the ARCHIVE3 data set.

b. Restore the TABLE data sets from the ARCHIVE3 data set if table restoration is indicated.

A detailed design of this module has not yet been done. However, the format and contents of the tables are known, as are the formulae necessary to produce them. Implementation of this module will be in the most straightforward fashion, since use and not production of the tables is of prime interest.

6.1.2.2 Input Module

This module produces an initial sample for the predictor from raw data using VSH methods. Input to the module is the IM IN data set and module output is the IN OUT data set. Of the three major functions of the input module - input, processing, output - only the format and contents of the output data set are currently known in any detail.

The raw data may come in several forms and there may be varying mixes of data. This is particularly true in the case of benchmarks and implementation contracts, where the form and content of the raw data will be specified as a part of the contract.

Implementation of input processing is currently seen as a three stage process.

a. Artificially generate an initial state to begin exercising VSH predictive formulations.

b. Apply VSH methods to existing initialization technology.

c. Research and implement currently unknown VSH methods.

As in Table Generation, implementation will be straightforward. The principal merit of VSH methods in this area is a gain in the consistency and accuracy of the initial sample. The proportion of time spent in initialization relative to prediction obviates undue concern for coding efficiency here.

6.1.2.3 Predictor Module

The prime function of the Predictor module is to integrate the model equations expressed in VSH coefficients. This involves an initialization consisting of

a. Reading and setting up incore tables

b. Building first copies of applicable work data sets from initial data. This involves data sets such as P Value and S Value.

c. Writing the initial state as the first record of the PM OUT data set.

After initialization, an iterative integration process is entered. Starting with the input initial state, the state is advanced in specified time steps up to some maximum specified time.

The Predictor module and its attendant data sets have been designed with sufficient generality to allow implementation of any desired integration method, either explicit or implicit, single or multistep.

Any of these techniques involve forming linear combinations of past states and derivative values, and computing new derivative values. The formation of a given linear combination is relatively straightforward. One requirement is to have available state and derivative values at the required number of previous times. This can be done by creating, writing, and accessing appropriate work data sets. The exact mechanization of the data set creation will be done in a consistent fashion in terms of Job Control Language and is discussed in the BUILD Subsystem description. Writing and accessing the data sets is easily done in the code implementing a particular integration technique.

Another requirement for integration is the computation of derivatives in spectral space. This is the largest part of the integration code. The formation of the linear combination of states and derivatives for integration can be coded and changed separately from the derivative computations. The derivative computations are an input to the integration. The actual integration is, in a topdown sense, above the derivative computation. Figure 6-2 shows the flow of derivative computation.

6.1.2.4 Output Module

This module produces desired output describing any of the other GO Subsystem functions. The output may be printout, plots, or data sets. The primary input to the module is the OM IN data set and the primary output is the OM OUT data set. The formats of both of these data sets are known. The requirements and formats of other output have not yet been defined. The initial version of the Output module will contain only code necessary to get the GO Subsystem "on the air." The Output module contents will grow as research, benchmark and contract requirements dictate.

6.2 The BUILD Subsystem

The main purpose of the BUILD Subsystem is to prepare an executable version of the GO Subsystem. Figure 6-3 shows the functional flow of the BUILD Subsystem.

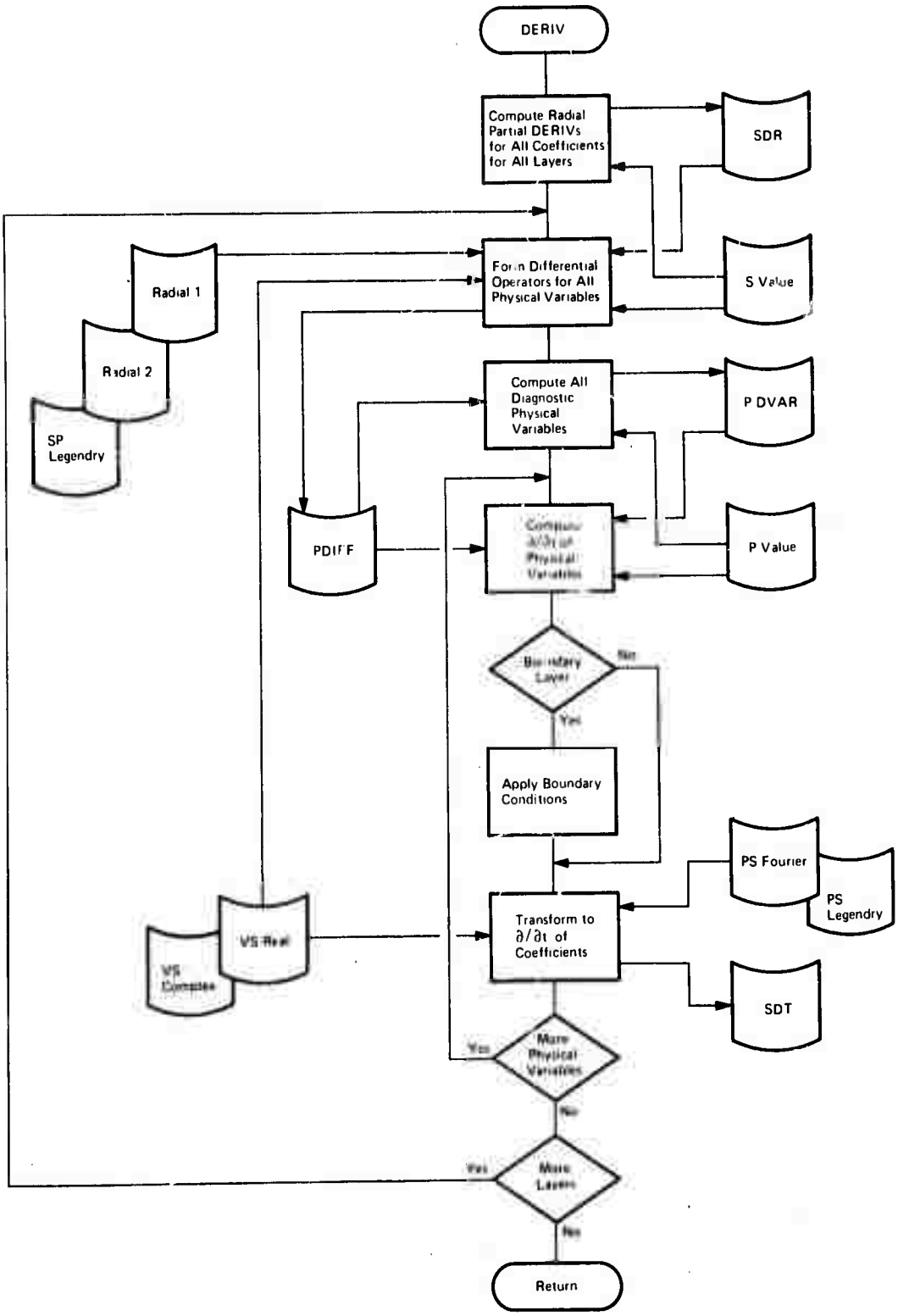

Figure 6-2. Derivative Computations

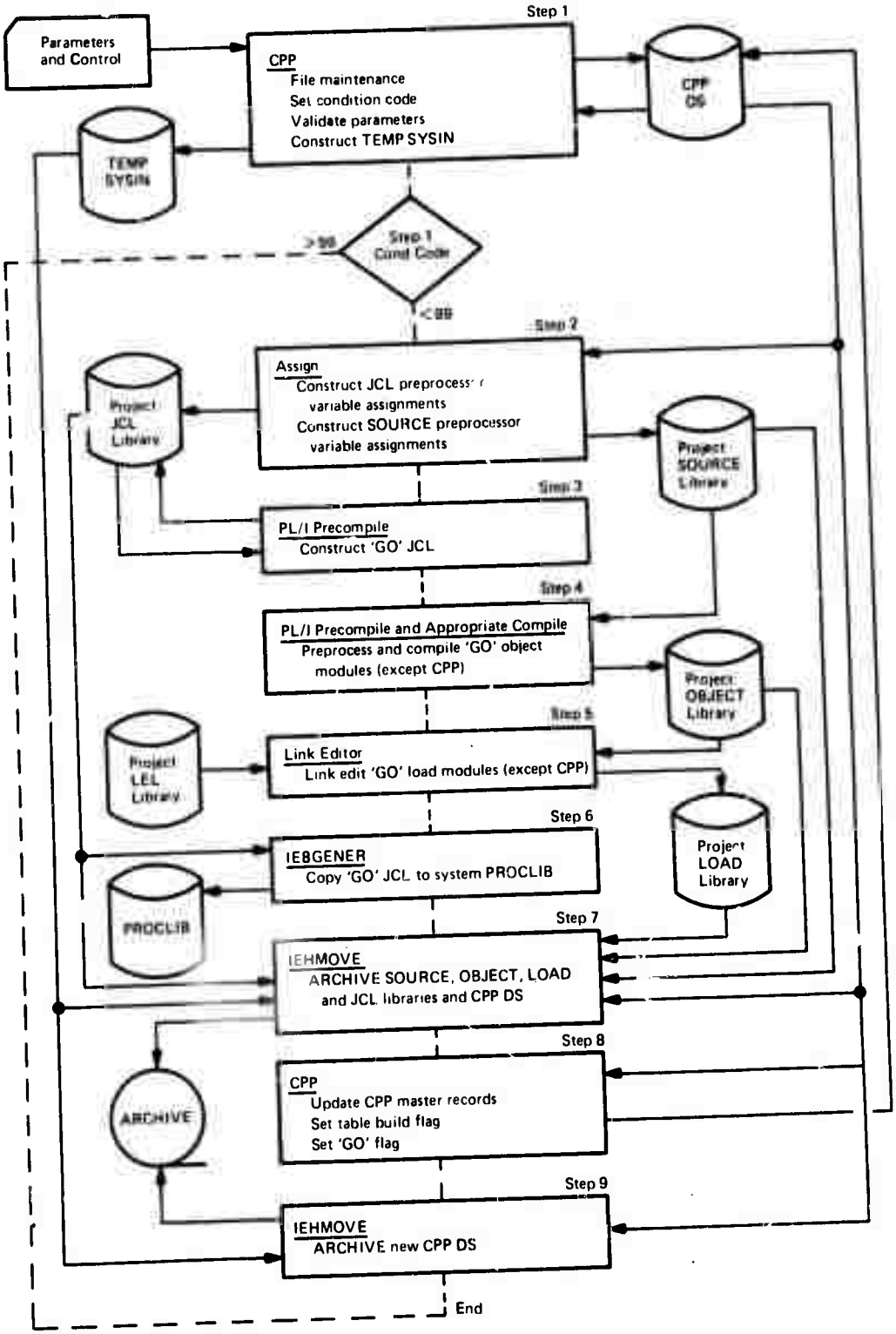

Figure 6-3. Build Subsystem

104

6.2.1 BUILD Data Sets

The BUILD Subsystem accesses and updates various program libraries. These are maintained as partitioned data sets on direct access devices. It also prepares archive data sets from the program libraries and CPP data set. These data sets are sequential, generally a tape. Each of the catalogued data sets listed below except the system procedure library will have the prefix qualifying name FGATWA6A, e.g., FGATWA6A. SOURCE. All data sets are permanent. All except ARCHIVE are partitioned data sets.

a. SOURCE - Contains source code of all programs except OS/360 utilities used in the baseline system.

b. OBJECT - Contains object modules of source code compiled from SOURCE members.

c. LOAD - Contains load modules of OBJECT members

d. LEL - Contains linkage editor code for forming load modules from OBJECT modules and, possibly, other load modules.

e. JCL - Contains Job Control Language (JCL) used to execute the three baseline subsystems. At a minimum, copies of the procedures for executing the BUILD, GO, and RESTORE Subsystems will be members of this data set.

f. ARCHIVE1 - Contains a copy of the project libraries resulting from a pass through the BUILD Subsystem, and a copy of the CPP data set before the master records are updated to reflect the just completed BUILD step.

g. ARCHIVE2 - Contains a copy of the CPP data set after the master records have been updated to reflect the BUILD step.

h. PROCLIB - The operating system procedure library. JCL procedures used to execute jobs must be copied from project libraries, such as FGATWA6A.JCL, to PROCLIB before they can be executed.

6.2.2 BUILD Subsystem Components

The first step is to read and process input parameters. These parameters are of two types, execution and precompile parameters. Execution parameters are used when executing the GO Subsystem. Integration time step is an example of an execution parameter.

Precompile parameters can be subdivied into JCL and SOURCE code parameters. They are used to build the JCL procedure and the source programs that will comprise the GO Subsystem. They may be thought of as execution parameters for the BUILD Subsystem. JCL parameters are used to compute such things as data set record length and blocksize. Source code parameters concern work storage sizes and segments of code to be included in program compilation. Precompile parameters may be functions of other precompile parameters.

The CPP program uses input parameters to update the CPP master and data records, and checks for a valid and complete set of input. If the data is in error, CPP sets a job condition code of 99, and the subsequent job steps are not executed. The CPP program also prepares a temporary data set containing utility control statements to be used for archiving the GO Subsystem once it is built.

At the completion of this step, CPP data records containing precompile parameters have been created, and the CPP master record pointers have been modified to index appropriate data records.

The next step is to assign values to all precompile parameters used to build the GO Subsystem JCL procedure and source programs. This is done in a PL/I program that accesses appropriate CPP data records, computes values for parameters that are functions of other parameters, and builds two data sets:

a. JCLPARM - A member of the project JCL library. It contains statements assigning values to all JCL precompile parameters.

b. SOURCEPARM - A member of the project SOURCE library. It contains assignments for all source code precompile parameters used in the Input, Table Generator, Predictor, and Output modules.

After JCLPARM and SOURCE PARM are written, the JCL procedure for executing the GO Subsystem must be built. This is done using the PL/I precompiler facility. A program executed by the precompiler has the following features:

a. %INCLUDE JCL Parameter declarations.

   This builds in line code identifying all JCL parameters as certain precompiler variable types

b. %INCLUDE JCLPARM.

   This makes the contents of the JCLPARM data set built in the previous step inline code. Values are thus assigned to all JCL parameters declared via the previous %INCLUDE.

c. Code using the JCL parameters to build JCL statements.

The output of this program is card images of the JCL procedure to be used for executing the GO Subsystem. The output is written on the project JCL library as member WA6GO. This process produces a JCL procedure with data set definitions consistent with the execution and precompile parameters input to the BUILD Subsystem.

Each of the GO Subsystem program modules are prepared in a similar fashion. The precompiler is used for a first pass through the source code to resolve references to precompiler variables. A source code control segment executed by the precompiler contains %INCLUDE's to:

a. Declare source code parameters

b. Put SOURCEPARM contents in line

c. Define work storage areas as functions of precompile parameters

d. Define I/O data sets as functions of precompile parameters

e. Include code segments.

The code segments may be included or excluded depending on values of precompile parameters. Further, they may contain expressions using precompile variables. The output of the precompiler is source code resulting from resolution of the precompile parameters and ready for compilation. This code is compiled and object modules for Input, Table Generator, Predictor and Output are written on the project OBJECT library.

The object modules are link edited and executable load modules written on the project LOAD library. The JCL procedure previously built is copied to the operating system procedure library for subsequent execution.

The version of the GO subsystem just constructed is archived for possible future restoration. This involves copying the following information onto a sequential data set:

a. The project SOURCE library

b. The project OBJECT library

c. The project LOAD library

d. The project JCL library

e. The CPP data set.

After archiving, the BUILD is complete. The CPP program marks and updates the master records accordingly, and the updated CPP records are also archived.

6.3 The RESTORE Subsystem

The RESTORE Subsystem retrieves the executing portion of an archived GO Subsystem and restores it to appropriate libraries for subsequent execution. The functional flow is shown in Figure 6-4.

6.3.1 RESTORE Data Sets

The RESTORE Subsystem accesses permanent data sets used in the other subsystems, and builds four temporary data sets for its own use.

a. ARCHIVE1 - Sequential data set containing an archived GO Subsystem. RESTORE copies the JCL and LOAD data sets back to temporary partitioned direct access data sets.

b. ARCHIVE2 - Sequential data set containing CPP data set master records. RESTORE copies these back to the CPP data set.

c. TEMPSYSIN - A sequential data set built by the CPP program. It contains utility control statements used in copying from ARCHIVE1 and ARCHIVE2 to TEMPJCL, TEMPLOAD and TEMPCPP.

d. TEMPJCL - A temporary partitioned data set containing the JCL library copied from ARCHIVE1.

e. TEMPLOAD - A temporary partitioned data set containing the LOAD library copied from ARCHIVE1.

f. TEMPCPP - A temporary direct data set containing the CPP data copied from ARCHIVE2.

g. PROCLIB - A partitioned data set containing the operating system procedure library. The GO JCL procedure is copied from TEMPJCL to PROCLIB so that it can be executed.

h. LOAD - A partitioned data set containing the project LOAD library. The Input, Table Generator, Predictor, and Output load modules are copied from TEMPLOAD to LOAD for subsequent execution.

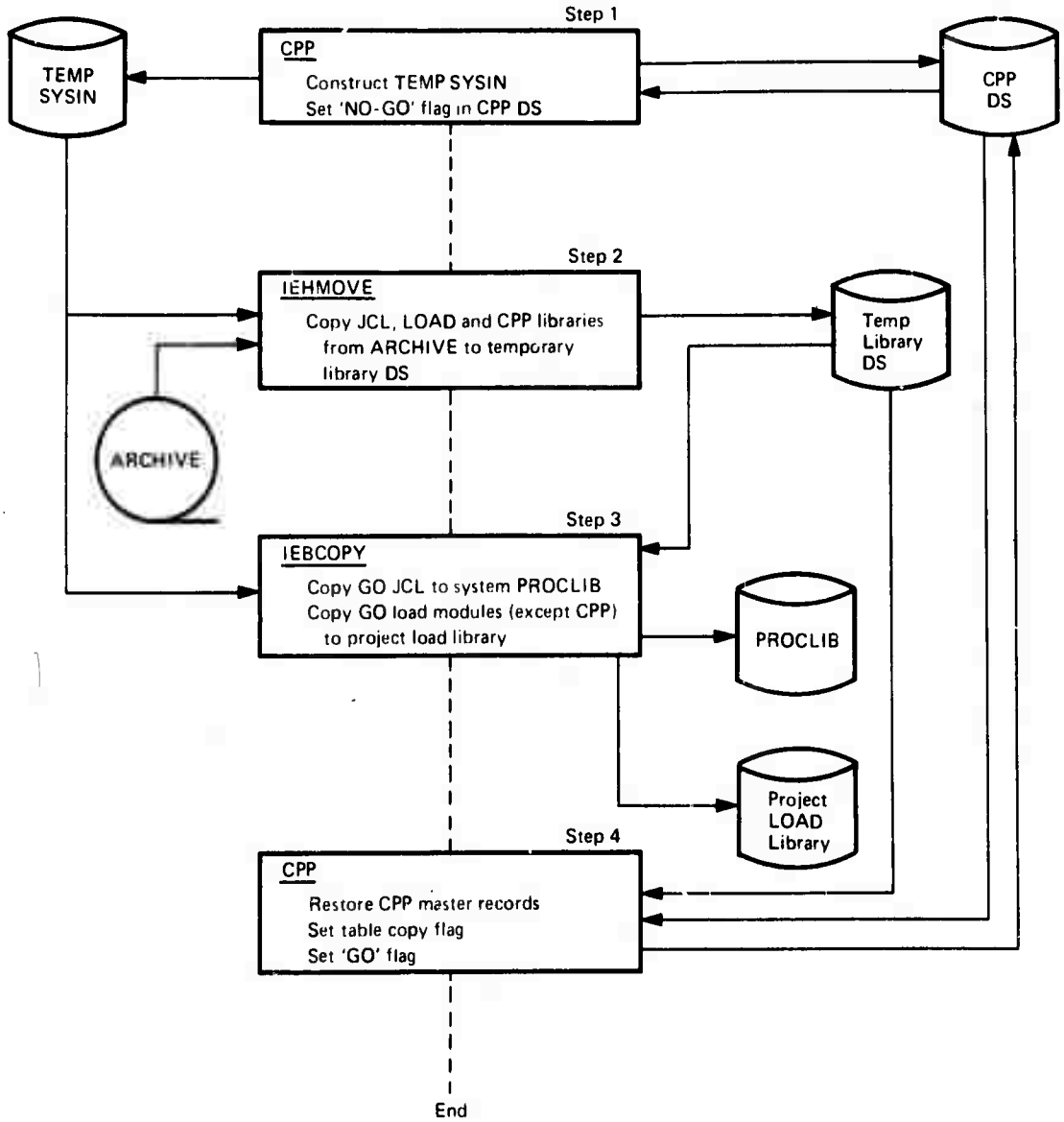

Figure 6-4. Restore Subsystem

6.3.2 RESTORE Subsystem Components

The CPP program marks the CPP system master record "no-go" to indicate that the restore process is not complete and prevent system execution until it is. CPP also builds an input data set of utility control cards for retrieval of the GO Subsystem in the next step.

The next step recreates the LOAD and JCL project libraries from ARCHIVE1 and the CPP data set from ARCHIVE2.

The Input, Table Generation, Predictor, and Output load modules are copied from the temporary, restored LOAD data set to the project LOAD library. The GO JCL procedure is copied from the temporary JCL data set to the system procedure library. The CPP master records are copied from the temporary CPP data set to the CPP data set.

The restored GO Subsystem is ready for execution. The CPP program marks the system master record accordingly, and indicates that archived tables must be restored when the GO Subsystem is executed for the first time.

This table restoration, while logically part of the RESTORE step, must be done in the GO step. The JCL (which varies from version to version) describing the tables is contained in the GO procedure.

6.4 Data Set Summary

Table 6-1 enumerates the data sets used in the baseline system. The following classifications are used:

a. Data Set Organization

   1. Direct

   2. Partitioned

   3. Sequential

b. Data Set Type

   1. Permanent - exists after system executes

   2. Temporary - exists only during subsystem execution

c. GO, BUILD, RESTORE

   1. I - used as input for some step in the subsystem

   2. O - output produced on the data set by some step in the subsystem.

Table 6-]. Data Set Summary

	Name	Organization	Type	Go	Build	Restore
	CPP	DIRECT	PERM	I/O	I/O	I/O
Project Program Library	SOURCE	PARTITIONED	PERM		I/O	
	OBJECT	PARTITIONED	PERM		O	
	LOAD	PARTITIONED	PERM		O	
	LEL	PARTITIONED	PERM		I	
	JCL	PARTITIONED	PERM		I/O	
	PROCLIB	PARTITIONED	PERM		O	
System Archives	ARCHIVE1	SEQUENTIAL	PERM		O	I
	ARCHIVE2	SEQUENTIAL	PERM		O	I
	ARCHIVE3	SEQUENTIAL	PERM	I/O		
	RAW DATA	SEQUENTIAL(?)	PERM	I		
GO Subsystem Module Communications	IN OUT	SEQUENTIAL	PERM	I/O		
	PM IN	SEQUENTIAL	PERM	I/O		
	PM OUT	SEQUENTIAL	PERM	I/O		
	ON IN	SEQUENTIAL	PERM	I/O		
	ON OUT	SEQUENTIAL(?)	PERM	I/O		
	TABLE CONTROL	SEQUENTIAL	PERM	O		
Predictor Module Tables	PREDINIT	SEQUENTIAL	PERM	I/O		
	RADIAL ONE	DIRECT	PERM	I/O		
	RADIAL TWO	DIRECT	PERM	I/O		
	PSLEGENDRE	DIRECT	PERM	I/O		
	PSFOURIER	DIRECT	PERM	I/O		
	VSREAL	DIRECT	PERM	I/O		
	VSCOMPLEX	DIRECT	PERM	I/O		
	SPLEGENDRE	DIRECT	PERM	I/O		
Predictor Work Data Tables	PVALUE	DIRECT	TEMP	I/O		
	PDIFF	DIRECT	TEMP	I/O		
	SVALUE	DIRECT	TEMP	I/O		
	SDR	DIRECT	TEMP	I/O		
	SDT	DIRECT	TEMP	I/O		
	PDVAR	DIRECT	TEMP	I/O		
Restore Work Data Sets	TEMPJCL	PARTITIONED	TEMP			I/O
	TEMPLOAD	PARTITIONED	TEMP			I/O
	TEMPCPP	PARTITIONED	TEMP			I/O

Appendix A. PRECOMPILATION CAPABILITIES USING A PRECOMPILER IN A PROGRAM PRODUCTION ENVIRONMENT

Generally speaking, a precompiler is a program that edits source code prior to compilation. This is similar to the "macro instruction" capability provided by most assemblers. This discussion is limited to capabilities of the PL/1 precompiler[1], which has been integrated with FORTRAN to allow precompiling in that language.

The capabilities provided by this package allow:

a. Including source code members from a library

b. Substitutions (e.g., of array dimensions, the scope of do-loops, and names of variables) from parameters set at compile time.

c. A "macro" capability whereby the user can extend the language syntactically to suit his own needs.

d. Conditional compilation of code.

With these facilities, the user can

a. Eliminate the repetitious task of duplicating common code (e.g., COMMON statements, DIMENSION and EQUIVALENT statements used more than once, by suitable inclusion of precoded members.

b. Aid the structuring process of system development by including dummy code sections which are filled in as the system is developed.

c. Improve running time (while preserving modularity) by including code, rather than calling subroutines with their attendant linkage delay.

d. Compile many versions of a program by setting parameters, without altering Source code.

e. Write "macros" to shorten the process of writing similar code sections.

(1) IBM Publication C20-1689, An Introduction to the Compile-Time Facility of PL/1, Aug. 1968.

In the following discussion, we will illustrate these capabilities by means of examples. In order to avoid an irrelevent discussion of the precompiler's syntax, the preprocessor statements are indicated by underlining english language statements. FORTRAN will be used as the source language, although the facilities are also part of PL/1 and have been extended to cover OS/360 Job Control Language.

Example 1. Including Code from a source library. In the following example, Source library LIBRARY contains a member, COMN, with the "System" COMMON statements. The programmer avoids having to duplicate COMMON statements by including the text of COMN:

SOURCE PROGRAM	MEMBER COMN
SUBROUTINE SUB	COMMON/A/X, Y, Z
Include COMN from LIBRARY	COMMON/B/U, V, W

```
     Additional Code
          .
          .
     RETURN
     END
```

The text produced by the precompiler is

```
SUBROUTINE SUB
COMMON/A/X, Y, Z
COMMON/B/U, V, W
     Additional Code
     RETURN
     END
```

Example 2. Parametizing array dimensions and other variables. The dimension of array A, and the scope of the do-loop, are from a preprocessor variable "DIM"

SOURCE PROGRAM

Set Preprocessor variable DIM = 10

```
DIMENSION A (DIM)
DO 100 I = 1, DIM
   A (I) = B  (I)
100 CONTINUE
```

The text produced by the Precompiler is

```
DIMENSION A (10)
DO 100 I = 1, 10
   A (I) = B  (I)
100 CONTINUE
```

Example 3. Conditional compilation of code. If the preprocessor variable DEBUG is set to PRODUCTION the code to call PDUMP is to be skipped.

SOURCE CODE

Set Preprocessor variable DEBUG = PRODUCTION

```
SUBROUTINE S(B)
A=B
```

If Preprocessor variable DEBUG = PRODUCTION then

```
CALL PDUMP (A,B)
```

Also

```
CALL PRINT (A)
END
```

Precompiler Output

```
SUBROUTINE S(B)
A=B
CALL PRINT (A)
END
```

114

If the Preprocessor variable DEBUG were set to, e.g., "TEST" in the first line above, the code produced would be:

```
SUBROUTINE S(B)

A=B

CALL PDUMP (A,B)

END
```

Because of the "else" statement, the call to PRINT was skipped and the call to PDUMP copied.

Example 4. Alternate formulation of code. In this example, the variable S has alternative expressions. In one case, it is to be computed as C*F*SIN(ALPHA), in another, as B*C/FUNC (P). This is accomplished by setting Preprocessor variable FORMULATION.

SOURCE CODE

```
Set Preprocessor variable FORMULATION = F1

If Preprocessor variable FORMULATION = F1

    S = C*F*SIN (ALPHA)

Else

    S = B*C/FUNC (P)
```

OUTPUT of Precompiler in this case is

```
S = C*F*SIN (ALPHA)
```

If FORMULATION were set to some other value the other expression would be copied into Source code.

Example 5. A macro called "STEP" is to return code to generate the commonly used expression

```
variable :: = variable + 1.
```

SOURCE CODE

Macro Definition STEP(X): Return (X = X + 1)

(This line states that the macro returns the form (argument followed by "=" sign followed by argument followed by "+1").

```
        .
        .
    STEP (ALPHA)

    STEP (BETA)

    STEP (Z)

        .
        .

    END
```

The output of the Precompiler is

```
    ALPHA = ALPHA + 1

    BETA  = BETA  + 1

       Z  = Z + 1

        .
        .

    END
```

The programmer is spared the necessity of coding each statement shown above. In this sense FORTRAN has been extended to include a form "STEP" among its keywords.