The GCRC Two-Dimensional Zonally Averaged Statistical Dynamical Climate Model: Development, Model Performance, and Climate Sensitivity

Robert Malcolm MacKay
B.A., Physics, California State University Chico, 1978
M.S., Physics, Portland State University, 1983
M.S., Atmospheric Physics, Oregon Graduate Institute, 1991

A dissertation presented to the faculty of the
Oregon Graduate Institute of Science & Technology
Department of Environmental Science and Engineering
in partial fulfillment of the requirements for the degree
Doctor of Philosophy
in
Atmospheric Physics

January 1994

The Dissertation "The GCRC Two-Dimensional Zonally Averaged Statistical Dynamical Climate Model: Development, Model Performance, and Climate Sensitivity Studies" by Robert M. MacKay has been examined and approved by the following Examination Committee:

M. A. K. Khalil
Professor, Oregon Graduate Institute of Scince & Technology,
Thesis Advisor

James A. Coakley, Jr.
Professor, Oregon State University

Reinhold A. Rasmussen
Professor, Oregon Graduate Institute of Scince & Technology

J. Fred Holmes
Professor, Oregon Graduate Institute of Scince & Technology

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my dissertation advisor Dr. Aslam Khalil, for without his many suggestions, words of encouragement, patience, and support this work would not have been possible.

I would also like to thank the other members of my dissertation committee, Dr. James Coakley, Dr. Reinhold Rasmussen, and Dr. Fred Holmes for their time and effort spent as committee members and reading this manuscript.

I thank the other members of the Global Change Research Center Modeling group at OGI: Dr. Yu Lu, Mr. Weining Zhao, Mr. Francis Moraes, Mr. Zhongyi Ye, Ms. Edie Taylor, and Ms. Martha Shearer, for their constructive comments and conversations throughout the duration of this work. Also I would like to express my appreciation to Dr. Douglas Kinneson of Lawrence Livermore National Labs for supplying me with his model's ozone output; Dr. Roy Jenne of NCAR for his help in my acquiring data on clouds, water vapor, surface elevations, and zonal ocean fraction; and Dr. Ronald Stouffer for his helpfull discussions during the Aspen Global Change Institute during the summer of 1992.

I would like to thank all of my colleagues at Clark College and the Clark College administration for their help and support during this work.

I would like to thank all of my family for the patience and encouragement during this work.

Finally, I would like to thank my wife Linda for her careful proof reading of this manuscript, and her never ending patience and love.

This work was partially supported by The U.S. Deparment of Energy grant number DE-FG06-85ER60313, The Andarz Co., and The Oregon Graduate Institute of Science & Technology.

DEDICATION

This dissertation is dedicated to my Uncle, Donald G. MacKay (1917-), who through his own personal success, kindness, and generous spirit has been, without a doubt, the most influential individual in my academic career.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The GCRC Two-Dimensional Zonally Averaged Statistical Dynamical Climate Model: Development, Model Performance, and Climate Sensitivity

Robert M. MacKay, Ph.D.
Supervising Professor: Aslam Khalil

The two-dimensional statistical dynamical climate model that has recently been developed at the Global Change Research Center and the Oregon Graduate Institute of Science & Technology (GCRC 2D climate model) is presented and several new results obtained using the model are discussed. The model solves the 2-D primitive equations in finite difference form (mass continuity, Newton's second law, and the first law of thermodynamics) for the prognostic variables zonal mean density, zonal mean zonal velocity, zonal mean meridional velocity, and zonal mean temperature on a grid that has 18 nodes in latitude and 9 vertical nodes (plus the surface). The equation of state, $p = \rho RT$ and an assumed hydrostatic atmosphere, $\Delta p = -\rho g \Delta z$, are used to diagnostically calculate the zonal mean pressure and vertical velocity for each grid node, and the moisture balance equation is used to estimate the precipitation rate.

The performance of the model at simulating the two-dimensional temperature, zonal winds, and mass stream function is explored. The strengths and weaknesses of the model are highlighted and suggestions for future model improvements are given. The parameterization of the transient eddy fluxes of heat and momentum developed by Stone and Yao (1987 and 1990) are used with small modifications. These modifications are shown to help the performance of the model at simulating the observed climate system as well as increase the model's computational stability.

Following earlier work that analyzed the response of the zonal wind fields predicted by three GCM simulations for a doubling of atmospheric CO_2, the response of the GCRC 2D model's zonal wind fields is also explored for the same experiment. Unlike the GCM simulations, our 2D model results in distinct patterns of change. It is suggested that the observed changes in zonal winds for the $2xCO_2$ experiment are related to the increase in the upper level temperature gradients predicted by our model and most climate models of adequate sophistication and resolution. We thus suggest that the same mechanism controlling the changes in zonal winds for the $2xCO_2$ experiment in our model also contributes to the simulated changes in zonal winds of the more complex GCMs.

CHAPTER 1

INTRODUCTION

Understanding Earth's climate has received much attention in the scientific community over the last 30 years. Today global change scientists are interested in understanding the complex integrated climate system in order to explain the natural climatic fluctuations of the past, as well as the potential changes in climate for the future. Since the mid 1970's the public has become increasingly aware of the possibility that anthropogenic activities have the potential to alter the global and regional climates of the Earth (see for example Rasool and Schneider, 1971 and Ramanathan, 1975). Hence climate change has now grown into a global issue, important not only to scientists but also to policy makers and the public.

Several methods are used by scientists today to investigate the climate system. Observations of the recent climate record and paleoclimatic evidence of past climate can be useful in inferring how the climate system works and how it may work in the future (see Karl et al., 1989). The use of climate models of varying complexities is also an attractive method for studying the possible response of the Earth to perturbations in climate forcing (see Mitchell, 1989). Climate models are especially useful and versatile since one is free to perform essentially any climate system response study and is not constrained merely to perturbations that have already happened. Climate models can also be used to study the intricate interactions and feedbacks between various components of the climate system.

The goal of this dissertation is to describe the development and use of a 2-dimensional (2-D) climate model. To achieve this objective, the work is divided into five distinct topics: 1) An introductory overview of the problem and a review of the state-of-the-science, including a historical prospective. 2) A description of the model, 3) The

model's performance, 4) An analysis of model sensitivity to changes in carbon dioxide concentration, and 5) a summary of the results presented and a discussion of future work.

In this chapter a general overview of the climate system and how scientists have attempted to model it is presented along with the benefits of using a variety of climate models of varying complexity. We conceptually describe the range of climate models in use today and identify where our 2-D model fits into the overall climate modeling hierarchy. A short outline of the historical development of the present science of climate modeling is offered at the end of this chapter for perspective.

1.1 The Climate System and Climate Modeling

As pointed out by Peixoto and Oort (1992) the Earth's climate is a result of complex interactions between the biosphere, $\boldsymbol{B}$ (plants and animals); cryosphere, $\boldsymbol{C}$ (snow, land ice, and sea ice); geosphere, $\boldsymbol{G}$ (mountains, volcanoes, and soils); the atmosphere, $\boldsymbol{A}$ (atmospheric gases, aerosols, and clouds); and the hydrosphere, $\boldsymbol{H}$ (oceans, lakes, atmospheric water vapor, and soil moisture). The primary source of energy driving the climate system is the sun, $\boldsymbol{S}$. It is now generally recognized that the large anthropogenic emissions of trace gases such as carbon dioxide and methane do indeed have the potential to alter the Earth's climate, see Schneider(1989) or Stone (1992). Thus the human species, $\boldsymbol{M}$ (economics, politics, culture, and population dynamics) is also a very important component of the climate system. In the notation of Peixoto and Oort (1992) the climate system is the union of all of these components, i.e.

$$\text{Climate System} = \boldsymbol{S} \cup \boldsymbol{B} \cup \boldsymbol{C} \cup \boldsymbol{G} \cup \boldsymbol{A} \cup \boldsymbol{H} \cup \boldsymbol{M}$$

It is impossible to offer a complete description of the climate system without understanding each component and all interactions between the individual components. This is a formidable task, beyond the present reach of scientists. In an attempt to obtain a more comprehensive understanding of the climate system, research groups often choose to pursue a detailed understanding of a single component of the climate system with the goal of having this understanding ultimately included in a more comprehensive climate model. Alternately there is a need for researchers to take existing knowledge from the

various disciplines and begin the process of creating more integrated models that will include the essential features of each component and the interactions between them.

In this work we concentrate primarily in the formulation and solution of a 2-dimensional climate model that primarily models the dynamics and radiative process of the atmosphere. This dissertation is the completion of a major step in creating a more comprehensive climate model. In the model presented we have included some interactions between the atmosphere, hydrosphere, cryosphere, and human activities and have made the model structure modular for easy modification in the future.

1.2 Climate modeling an overview

The climate of a particular location can be defined as the long term average (typically at least five years and more commonly 30 years) atmospheric and surface conditions of that location. Two important aspects of climate are the length of the time average of a particular meteorological variable Ψ and the extent of the spatial average. Since the natural climate is not stationary (i.e. its average depends on when the average is taken) reference to the specific time of a given period must also be given for a proper description of the climate of interest. For example, it is hard to discuss the climate of California without reference to whether the interest is in the present climate, the climate of the past decade, the past century, or the past millennium. Also, the climate of San Francisco may be much different than the climate of California, North America, or the whole globe. The term climate then, can refer to a variety of time average lengths and spatial average extents as well as to specific eras in time and regions of space. Thus, when studying climates, particular attention must be given to exactly what features of the climate are of interest.

Saltzman (1978) gives an excellent presentation of the various climate models in use today with particular attention to 2-D climate models. Figure 1.1 below is adapted from Saltzman's (1978) Figure 2 giving the averaging hierarchies for the resolution of climate. As noted by Saltzman, the climate models in use can be classified in an identical way and hence Figure 1.1 can also be viewed as the hierarchies of climate models. In the discussion to follow, the symbol Ψ is an arbitrary climate variable of interest

(temperature, pressure, precipitation, soil moisture, etc.), $[\Psi]$ is a time average of Ψ, $\langle\Psi\rangle$ is a vertical average, $\hat{\Psi}$ is latitudinal average, and $\overline{\Psi}$ is a longitudinal average.

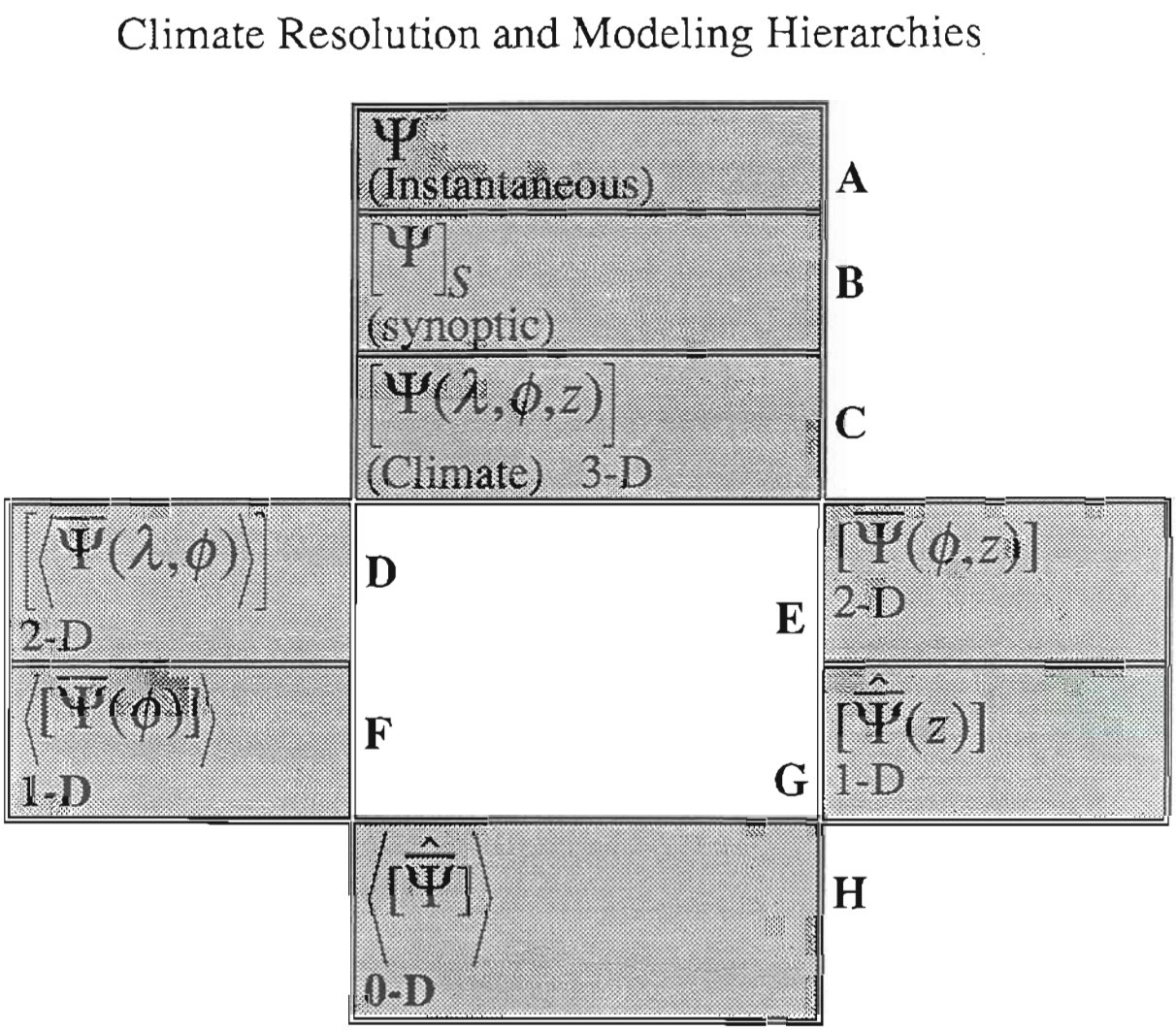

Figure 1.1 The graphical description of climate resolution and climate modeling hierarchies given by Saltzman (1978).

At the top (box A of Figure 1.1) is the instantaneous resolution of climate, which at present cannot be modeled. Next (box B) are the explicit-dynamical models with synoptic resolution in time of 15 to 30 minutes. Present day atmospheric general circulation models (AGCMs) fall into this category. The AGCMs are 3-dimensional models (longitude λ, latitude ϕ, and a vertical coordinate z) and come with a variety of spatial resolutions. Washington and Parkinson (1986) give a review of the theory and methods of solution for AGCMs. As with many climate models, the climate parameter of interest, Ψ, is obtained by a numerical solution of an initial value problem. The new value of Ψ after a given time interval (or step) is expressed in terms of its old value at the beginning of the time step Ψ_o plus some average change $\Delta\Psi$ for the time step, estimated using the laws of physics. AGCMs typically use time steps on the order of 15 to 30 minutes, consistent with the synoptic resolution. After calculating the time series for Ψ, hourly, daily, or monthly averages may be taken and then 5 to 10 year averages of these averages can be calculated to obtain the climate output of interest. For example, the average of the northern hemisphere night-time temperatures for January over a ten year period may be of paramount interest.

Farther down the hierarchical ladder are the 2-D climate models of which there are two primary types. Box D of Figure 1 describes the first type of 2-D model which is a horizontal model with longitude and latitude as the coordinates and Ψ averaged in the vertical (and time) $\left[\left\langle\Psi(\lambda,\phi)\right\rangle\right]$ (see for example Charney, 1947 and Philips, 1954). In box E the second type of 2-D model is given with latitude and height (or pressure) as coordinates and Ψ averaged over longitude (zonal average) $[\overline{\Psi}(\phi,z)]$ (see Held and Suarez, 1978 and Schneider, 1984). This is the type of model that we have developed and discuss in this dissertation.

Next down the hierarchy ladder are the 1-dimensional models. There are two types of 1-dimensional models that have been used extensively in the past 20 years. The first, box F, has latitude as a coordinate $\left[\left\langle\overline{\Psi}(\phi)\right\rangle\right]$ (see Budyko, 1969; Sellers, 1973; and North, 1975). The second type of 1-dimensional model, box G, has a vertical coordinate such as pressure or height $[\hat{\overline{\Psi}}(z)]$ (see Manabe and Wetherald, 1967; Ramanathan, 1976; and MacKay and Khalil, 1991). Finally at the bottom of the ladder, is the 0-dimensional model which gives the global average $\left[\left\langle\hat{\overline{\Psi}}\right\rangle\right]$ Goody and Walker (1972).

As noted by Saltzman(1978), the highest resolution models require the least amount of parameterization of sub grid scale physics or sub time step physics, while the lowest resolution models require the most severe parameterization. Saltzman contends that climbing the model hierarchy is one way to get a more realistic representation of the climate system but also points out: "it can be argued that some of the more sophisticated lower resolution statistical-dynamical models may already contain the maximum amount of information that is possible to deduce or verify in a very-long-term integration over geologic eras and hence may be optimum simulation models for this purpose." Another point about modeling complexity is worth making. Most AGCMs in use today use time steps of 15-30 minutes. However the climate variables of interest are often averages over months, years or even centuries. Little work has been done towards the parameterization of sub time step processes in an attempt to increase the computational efficiency of these climate models. As discussed in the historical overview below, the AGCMs of today evolved from synoptic weather prediction models and hence they may be optimally designed for this purpose and not for the prediction of climate change.

The model described in this treatise is a 2-D statistical dynamical model with coordinates of latitude, ϕ, and vertical height, z. This type of model fits in box E of Figure 1.1, where the climate variable of interest is taken as the zonal average value of that variable. There are a variety of models of this type all of varying complexity. The two main types of 2-D $[\overline{\Psi}(\phi, z)]$ models are the energy balance models EBM, Wang et al. (1990b) and Peng et al. (1982); and the momentum models, Held and Suarez (1978), Ohring and Adler (1978), Yao and Stone (1987), and Stone and Yao (1987, 1990). Saltzman (1978) discusses both of these types of models and emphasizes that the momentum models are a much more general type of climate model and hence can be used for a wider range of climate investigations. Both the 2-D EBMs and the 2-D statistical dynamical momentum models calculate the temperature structure of the atmosphere, and sometimes even include implicit or explicit schemes for the calculation of water vapor, surface processes such as ice or snow growth, or vegetation cover. The 2-D momentum models have the advantage over the purely 2-D EBMs in that they solve the fundamental equations of motion and thermodynamics for the zonal, meridional, and vertical components of zonally averaged momentum while the EBMs ignore explicit reference to momentum. The computed values of momentum can then be used in model calculations

of advection of energy , moisture, aerosols, or momentum to more realistically simulate the interactions between the different components of the climate system. In addition, surface momentum is often used in the calculation of surface fluxes of sensible and latent heat, as well as frictional drag. Our 2-D model explicitly calculates the atmospheric momentum.

Previous to this work the OGI data analysis and theory group has developed and implemented a 0-D (box H of Figure 1) climate model for the use as an interactive learning package for high school and first year college students, MacKay and Khalil (1993); and a 1-D EBM (box F of Figure 1) into their learning materials package, MacKay and Khalil (1992). Our group has also created and implemented a research grade 1-D radiative convective model (RCM) of the type shown in box F of Figure 1, MacKay (1990); have used this model in the simulation of the climate sensitivity to changes in the atmospheric concentrations of greenhouse gases and volcanic aerosols, MacKay and Khalil (1991); and have distributed this model world wide for others to use. The development of the 2-D model described in this paper is thus the next step up the Climate modeling ladder shown in Figure 1.1.

The rationale for developing a 2-D climate model at the OGI Global Change Research Center (GCRC) instead of immediately developing a 3-D GCM has several important facets. From a purely local prospective, since the GCRC climate group is relatively new, it seems essential that we develop a broad base of model types from which to grow. This 2-D model is the next component in the climate modeling base at the OGI Global Change Research Center. As noted by Henderson-Sellers and McGuffie (1987) two dimensional models have two distinct advantages over 3-D models 1) they are more computationally efficient and hence can be used for longer time scale integrations than full AGCMs and 2) 2-D models offers insights into understanding particular interactions of the climate system more easily and clearly than is possible with 3-D AGCMs. As pointed out above, Saltzman (1978) suggests that there may be some climate processes that are optimally described with a sophisticated 2-D model, so using a 3-D model to simulate some processes might not be justifiable. Joseph Smagorinsky (1983) one of the pioneers in numerical weather prediction and climate modeling noted, " One must also be prepared to go backward, hierarchically speaking, in order to isolate essential processes responsible for results observed from more comprehensive models". Smagorinsky's

statement suggests that the 2-D model can be used to obtain insights into the mechanisms that are important in governing the response of the climate system to various internal and external climate forcings. We stress the importance of the latter idea more in chapter four when we investigate the response of our 2-D model to a doubling of atmospheric carbon dioxide.

For example, Rosen and Gutowski (1992) recently investigated the change in zonal velocity u and global angular momentum brought about by doubling the atmospheric concentration of carbon dioxide as simulated by three different GCMs. As we point out in Chapter 4 the results from the GCM studies were fairly inconsistent if not altogether confusing. Since the background noise of our model is low, this is an excellent example of an experiment that should and can be repeated with our 2-D model. Analyzing our results and comparing them with those of the GCMs can help add to our understanding of the physical mechanisms driving the different responses of the models and hence possibly the actual climate system. We will explore this further in Chapter 4.

The use of 2-D models, or for that matter a variety of models of varying complexities, truly is an essential ingredient in any complete and comprehensive exploration of the climate system. Before moving on to a description of our 2-D model, we briefly outline the history of meteorology and climate modeling below to provide background for the rest of this dissertation.

1.3 A Historical Prospective

The science of meteorology dates back to the Greeks and Aristotle's book Meteorologica, written about 340 B.C.(Eagleman, 1985). Edmund Halley in the seventeenth century, George Hadley in the eighteenth century, and William Ferrel and G.G. Coriolis both in the nineteenth century all made substantial contributions to our present understanding of the dynamics of the atmosphere,(Forrester, 1981). However, the monumental work of Lewis F. Richardson (1922) can be considered as the birth of present day numerical weather prediction and climate modeling. In Richardson's book Weather Prediction by Numerical Process he made a heroic attempt to forecast the changes of such weather variables as temperature, pressure, precipitation, and wind

velocity over central Europe using the laws of physics and numerical techniques that are much more amenable to modern day digital computers. Even though his model forecast for 6 hours into the future was highly unrealistic, his work paved the way for future scientists in their attempts to model the weather and climate from the fundamental principles of physics. Richardson felt that errors in his initial conditions (initial state of the climate) gave rise to unrealistic results. This is undoubtedly true. However, the fact that he used a six hour time step for grid sizes of approximately 200 km on a side was probably more detrimental to his results. We today can gain an appreciation of Richardson as a true pioneer in the science of weather prediction from his statement, "Perhaps some day in the dim future it will be possible to advance the computations faster than the weather advances and at a cost less than the saving to mankind due to the information gained. But this is a dream."

In 1946 John von Neumann began working at the Institute of Advanced Study (IAS) in Princeton on an electronic computer ENIAC for the purpose of weather prediction (Smagorinsky, 1983); and in 1950 J.G. Charney used a simplified (filtered) version of the basic equations governing atmospheric processes to make the first numerical forecast of geostrophic winds (Holton, 1979). The successful development of the computer by von Neumann and others in the early 1950s opened the doors to the study of numerical weather and climate prediction. Because of the early success of the IAS group, numerical weather prediction had a strong foothold in the scientific community by 1960 and the decade of the 60's saw a variety of atmospheric circulation and climate models produced. Sykuro Manabe, who began working with the Princeton group in 1959, was one of the early pioneers in one dimensional radiative convective models (Manabe and Strickler, 1964), and general circulation models (Manabe and Bryan, 1969). Today there are many scientists working with 3-D general circulation models worldwide.

The roots of the 2-D zonally averaged statistical dynamical model of the type developed in this dissertation, $[\overline{\Psi}(\phi, z)]$, also stem back to the early work of (Charney, 1947; Eady, 1949; and Philips, 1951). Although the original weather prediction models were considered as 2.5 dimensional models having 2 atmospheric layers and horizontal resolution in both latitude and longitude, many of the ideas developed for the turbulent transport of heat and momentum have been adopted for use in the more recent zonally

averaged momentum models. Manabe and Strickler (1964) developed one of the earlier zonally averaged 2-D models in which they calculated the temperature of the surface and atmosphere as a function of latitude and height. Saltzman and Vernekar (1971) were among the first to introduce the calculation of atmospheric dynamics into a 2-D $[\Psi(\phi,z)]$ climate model. Other noteworthy papers on 2-dimensional zonally averaged statistical dynamical models are by Hunt (1973), MacCracken (1972), Temkin and Snell (1976), Schneider and Lindzen (1977), Held and Suarez (1978), Ohring and Adler (1978), Vernekar and Chang (1978), Schoeberl and Strobel (1978), and MacCraken (1987).

An important consideration in modeling climate with a 2-D model is a realistic parameterization of the turbulent eddy flux of heat and momentum at mid-latitudes due to baroclinic instabilities. Stone (1972,1973, 1978) using ideas originally introduce by Charney and Eady, made substantial progress towards developing realistic parameterizations of these turbulent transport processes. Yao and Stone (1987) and Stone and Yao (1987,1990) have included parameterizations of eddy fluxes of heat and momentum into one of best 2-D statistical dynamical models in existence today. Many of the parameterizations they have used stem from Stone's earlier work and from the work of Branscome (1983). It can be argued that the work of Stone and Yao over the last several years has inspired new interest in two-dimensional modeling because their parameterizations of eddy fluxes of heat and momentum have greatly improved our ability to simulate these sub-gridscale processes .

The 2-D $[\Psi(\phi,z)]$ model developed and presented in this dissertation includes some model physics and parameterizations of physical processes that have been previously used successfully by many other climate modeling researchers as well as some innovative methods for simulating physical processes of the climate system. The intent of this work is to develop a two dimensional climate model (base model) that can be used for a variety of climate studies. The major efforts so far have been spent on implementing model physics for the transfer of radiation through the atmosphere and the calculation of the zonally averaged atmospheric motions. Other physical processes have also been included in the model but as yet are fairly basic. The design of our model is modular to facilitate future step-wise improvements so that changes of model performance due to each enhancement in the model physics can be easily understood. Our ultimate goal is to develop this model into an highly integrated model which will be

able to investigate many of the interactions between the various components of the climate system. In the next chapter we will offer a complete description of the present version of the 2-D climate model.

CHAPTER 2

MODEL DESCRIPTION

2.1 Model Overview

2.1.1 Introduction

We begin with a brief outline of the basic structure of the GCRC 2D statistical dynamical climate model. The basic equations central to the model's operation are then presented, followed by a brief discussion of the general numerical procedures used to solve them. The physics behind each of the equations and the specific details of their solution are described in enough detail to enable the reader to reproduce the results given in this and subsequent chapters. Since the 1D radiative convective model developed and described by MacKay (1990) is one of the key components of the 2D model, we will also briefly describe it and the modifications that have been made. Also, in appendix C, we have included the source code of the model for those interested in the specific details of the numerical scheme used for each segment of the model.

2.1.2 Model Grid

Our 2-D statistical-dynamical model is essentially a series of vertical 1D radiative convective models aligned next to each other. Fluxes of energy, mass, momentum, and moisture can be transferred from one neighboring model grid point to the next; or from the surface to the boundary layer. We solve the model equations for the zonally averaged values of temperature, pressure, mass density, and wind velocities.

The horizontal resolution of each grid may be varied, but for this work is taken to be approximately 9.44 degrees in latitude (170°/18). There are nine atmospheric layers and one surface layer. The atmospheric layers are at fixed heights; with a 200 meter boundary layer and the top of the atmosphere being at 40.0 km. The vertical boundaries (in km) of the grid boxes are 40.0, 20.0, 14.0, 10.0, 8.0, 6.0, 4.0, 2.0, 0.2, and 0.0; while their horizontal boundaries (in degrees latitude) are 85 (85 N), 75.5 , 66, ...-85 (85 S). The polar region between 85 and 90 degrees latitude have been intentional excluded to enhance computational stability. The center of each grid box is taken to be at its center of mass. There are thus 9x18=162 atmospheric grid boxes each of which is in the shape of a thin toroid. For clarity, in figure 2.1, we show the model structure for a two layer four zone model which has the same structure as our 2-D model but with much less resolution.

2.1.3 Clouds

Clouds are prescribed in the model using seasonal averages of observed cloudiness obtained from Roy Jenny at NCAR; see Hahn et al. (1988). The data set is based on surface observations and includes ocean and land values. Low (Cumulus, Stratus, and Cumulonimbus), middle (Altostratus and Nimbostratus), and high (Cirrus) clouds are averaged for each season over a ten year time period (1971-1981) for land based observations and a twenty-nine year period (1952-1981) for ocean based observations. An area weighted average of the land and ocean values is used to obtain the average zonal cloudiness. Appendix B shows the reduced data for the 9.44 degree latitude zones of our model. Low clouds are assumed to be located in the layer between 200 m and 2 km (layer 8), middle clouds between 4 and 6 km (layer 6), and high clouds between 10 and 14 km (layer 3) for latitudes less than 30°(tropics) and between 8 and 10 km (layer 4) for latitudes greater than 30°. The cloud radiation properties are calculated as described by MacKay (1990) and follow the scheme of Lacis and Hansen (1974). The cloud optical depths used due to scattering are, high clouds $\delta = 1.0$, middle cloud $\delta = 1.2 - 0.6|\phi|/85°$, and low clouds $\delta = 4.0 - 2.0|\phi|/85°$. The latitude (ϕ) dependence is included to help simulate the lower optical depths associated with clouds that have a lower water vapor content. The absorption optical depths are calculated as in Lacis and

Hansen (1974). It should be noted that we have used the scattering optical depths of our model clouds as adjustable parameters and the values given above were selected to give a globally averaged absorbed solar energy for the model that is in agreement with observations. As discussed below, we have not taken the same liberty with surface albedos and cloud cover.

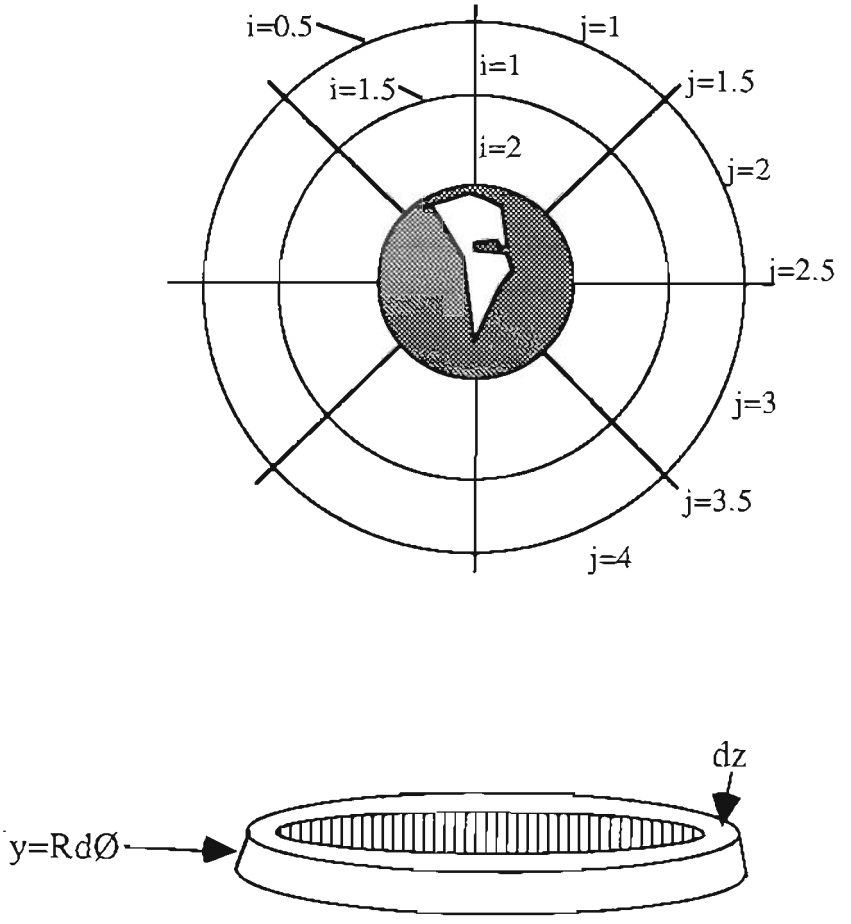

Figure 2.1. The model structure of a 2 layer-4 zone model and geometry of a typical model grid box.

2.1.4 Solar Energy

The daily average flux of incident solar radiation per unit surface area, Q_S, for each zone is calculated following Liou (1980) as

$$Q_s = S(t)\left[\left(\frac{H\pi}{12}\right)\sin\phi\sin\delta + \cos\phi\cos\delta\sin\left(\frac{H\pi}{12}\right)\right] \tag{2.1}$$

where δ is the solar declination (+23.5° degrees on July 21 to -23.5° on December 22), ϕ is the latitude position, H is the half-day length in hours, $S(t)= 1360\ (r(t)/r_m)^2\ W/m^2$, $r(t)$ is the earth-sun distance, and r_m is the mean distance between the Earth and sun (1.49×10^{11} m). The earth-sun distance is calculated following chapter 3 of Goldstien (1980) assuming that the orbital period of the Earth is 365.25 days, the earth is at perihelion on January 4, and its eccentricity is 0.0167. The average solar intensity per unit surface area is assumed to be equal to the product of the solar intensity at the top of the atmosphere, the length of day in hours 2H divided by 24, and the average solar zenith angle ($\cos^{-1}\mu$), i.e.

$$\left[\overline{S}\right] = S_o(t)\frac{H}{12}\mu \tag{2.2}$$

The length of the solar day 2H at latitude ϕ can be calculated following Liou (1980) (pp. 46-47) as

$$\begin{aligned} H &= 12.0 - \cos^{-1}\left[\tan\delta\tan\phi\right]\frac{12.0}{\pi} && -1.0 < \left[\tan\delta\tan\phi\right] < 1.0 \\ H &= 0.0 && -1.0 > \left[\tan\delta\tan\phi\right] \\ H &= 12.0 && +1.0 < \left[\tan\delta\tan\phi\right] \end{aligned} \tag{2.3}$$

Equations (2.1)-(2.3) are then used to calculate the daily average value of μ which is essential in the solar radiative calculations, see MacKay and Khalil (1991).

2.1.5 Surface Features

For most of the simulations described in this thesis, the Earth's surface is assumed to consist of land, a mixed layer ocean of depth 50.0 m, and possibly sea ice. The zonal fraction of ocean and land is taken from the Data Support Section, Scientific Computing Division, of NCAR, DS750.1 Rand elevation Data. The snow-free land surface albedo α_L for each zone is taken from Hansen et al. (1983) and is assumed to be independent of time. Table 2.1 gives these values along with the assumed ocean fraction for each model zone. The two values of 0.7 noted in parentheses for the two southern most grid points were used by us instead of 0.5 in an attempt to get better agreement between model and observations.in the Antarctic region

When the land surface temperature T_L cools to be below 273 K, the albedo increases because of assumed snow and ice accumulation, according to

$$a_L^* = 0.6 + (a_L - 0.6)\exp\left(-\frac{(273 - T_L)}{12}\right) \tag{2.4}$$

The ocean surface albedo α_O is taken from Hansen et al. (1983) and is a function of solar zenith angle μ and surface wind speed V_S,

$$\alpha_O = 0.021 + 0.0421x^2 + 0.1283x^3 - 0.04x^4 + \frac{3.12x^5}{5.68 + V_s} + \frac{0.074x^6}{1 + 3V_s} \tag{2.5}$$

where x=1-μ. The portion of the ocean in a particular model zone, that is covered by sea ice, is assumed to have an albedo of $0.7+(\alpha_O - 0.7)\exp(-2X_I)$ where X_I is the ice thickness in meters. The dependence of the albedo on ice thickness is intended to simulate the penetration of solar energy through the ice into the ocean for thin ice.

All earth based surfaces are assumed to have an infrared emissivity of 1.0. The land area is assumed to have a heat capacity equivalent to 0.5 meters of water and its temperature is calculated from energy considerations according to the net downward flux of IR and solar radiation and the net upward flux of IR and solar radiation at the surface,

$$0.5C_w\rho_w \frac{\partial T_L}{\partial t} = (F\downarrow - \sigma T_L^4 + S\downarrow - S\uparrow)$$

Table 2.1 Horizontal center of model grids with snow free land albedo α_L and percent Ocean fraction ξ for each zone. Negative latitudes are for southern hemisphere.

Grid Center (°North)	Land Albedo, α_L	% Ocean Fraction, ξ
80.3	0.45	89
70.8	0.25	70
61.4	0.17	29
51.9	0.17	42
42.5	0.21	46
33.1	0.22	56
23.6	0.26	62
14.2	0.18	74
4.7	0.14	77
-4.7	0.14	76
-14.2	0.17	77
-23.6	0.22	76
-33.1	0.24	88
-42.5	0.22	96
-51.9	0.23	99
-61.4	0.2	91
-70.8	0.5 (0.7)	28
-80.3	0.5 (0.7)	0.0

The temperature of the surface layer of sea ice is calculated assuming that the top 0.2 m of the ice absorbs and emits radiation,

$$0.2C_I\rho_I\frac{\partial T_I}{\partial t} = (F\downarrow - \sigma T_I^4 + S\downarrow - S\uparrow)$$

If the sea ice is covered by snow then the complete snow layer is assumed to absorb and emit radiation and the surface temperature of the top layer of ice is calculated according to,

$$H_s C_I \frac{\rho_I}{4.0} \frac{\partial T_I}{\partial t} = (F\downarrow - \sigma T_I^4 + S\downarrow - S\uparrow)$$

where we have assumed that the snow density is one-fourth the density of ice. Precipitation falls as snow on sea ice if the air temperature above the snow surface falls below 272 K. If the surface temperature of the snow rises above 272 K, enough snow melts to drop the surface temperature back down to 272 K until all of the snow has melted.

The Ocean part of a zone has a thermal heat capacity which is directly proportional to its depth. Thus, the Temperature of the ocean changes gradually with time and is calculated according to the relation,

$$C_w D \rho_w \frac{\partial T_O}{\partial t} = (F\downarrow - \sigma T_O^4 + S\downarrow - S\uparrow - SH - LH) * \frac{(A_O - A_I)}{A_O}$$

$$+ \frac{A_I}{A_O} \frac{(T_I - T_O)}{\left[\frac{X_I}{K_I} + \frac{X_s}{K_s}\right]} + \frac{K_h}{a^2 \cos\phi} \frac{\partial}{\partial\phi}\left(\cos\phi \frac{\partial T_O}{\partial\phi}\right)$$

where C_w and C_I the specific heat of water and ice (4.2 J/(gK) and 2.1 J/(gK) respectively), D is the depth of the mixed layer and H_s is the depth of snow over ice, ρ_w and ρ_I are the density of water and ice (1.0 g/cm^3 and 0.91 g/cm^3), T_O is the ocean temperature, $F\downarrow$ is the downward flux of IR radiation from the atmosphere, σ= 5.67x10^{-8} W/m^2/K^4, $S\downarrow$ and $S\uparrow$ are the downward and upward fluxes of solar radiation, SH and LH are the fluxes of sensible and latent heat leaving the oceans surface, A_O is the surface area of the ocean, A_I is the area of sea ice, K_I and K_s are the thermal conductivity of ice and snow (2.2 J/K/m and 0.3 J/K/m), T_I is the temperature of the top of the sea ice, X_I and X_S are the average ice and snow thicknesses, a is the radius of the earth, and K_h is a horizontal turbulent diffusivity constant for the mixed layer ocean (we use 1.2x10^{12} m2/yr). Since the Antarctic continent completely occupies zone 18, we specify the southern ocean boundary at zone 17.

The effective radiating temperature of the earth's surface, which is used for calculating the upward flux of IR radiation received by the atmosphere, is taken as an

area weighted average of the temperatures of the land, ocean, and sea ice (T_L, T_O, and T_I). Taking the zonal ocean fraction to be ξ and the fraction of it covered by sea ice to be χ, the effective surface radiating temperature is calculated from

$$T_S = \left[\xi\left((1-\chi)T_O^4 + \chi T_I^4\right) + (1-\xi)T_L^4\right]^{1/4} \tag{2.6}$$

Although the simulations described in this thesis assume a 50 meter deep ocean mixed layer with no transport to the deep ocean, we have done some preliminary work with a deep layer ocean model that can be coupled to our model's atmosphere. We describe this model in a later section of this chapter since the existing treatment of the ocean is just a simplified version of the more complete ocean model.

We have not explicitly considered orography and we have neglected vegetation type and soil type in this work.

2.2 Model Equations

The GCRC 2D statistical dynamical climate model uses the primitive equations, in zonally averaged form, to solve for the climate state of the model planet (i.e. the temperature, pressure, three dimensional wind velocity, radiative fluxes, and atmospheric moisture/precipitation). There is extensive discussion of these equations in the literature; e.g. Lorentz (1967), Saltzman (1978), and Holton (1979). The form of the equations most closely mimic that used by Yao and Stone (1987) for their statistical-dynamical model, with the difference being that their equations are written for a sigma vertical coordinate system instead of a z=ln(p) coordinate system as in our model. The equations used are:

the equation of continuity,

$$\frac{\partial \rho}{\partial t} = -\frac{1}{a\cos\phi}\frac{\partial(\rho v\cos\phi)}{\partial \phi} - \frac{\partial \rho w}{\partial z} \tag{2.7}$$

the horizontal equations of motion,

$$\frac{\partial \rho u}{\partial t} = -\frac{1}{a\cos\phi}\frac{\partial[\rho(uv + \overline{u'v'})\cos\phi]}{\partial\phi} - \frac{\partial[\rho(uw + \overline{u'w'})]}{\partial z}$$
$$+\rho fv + \rho\frac{\tan\phi}{a}(uv + \overline{u'v'}) + \rho F_\lambda \tag{2.8}$$

$$\frac{\partial \rho v}{\partial t} = -\frac{1}{a}\frac{\partial p}{\partial\phi} - \frac{1}{a\cos\phi}\frac{\partial[\rho v^2\cos\phi]}{\partial\phi} - \frac{\partial[\rho(vw + \overline{v'w'})]}{\partial z}$$
$$-\rho fu - \rho\frac{u^2\tan\phi}{a} + \rho F_\phi \tag{2.9}$$

the first law of thermodynamics

$$\frac{\partial[\rho C_p T]}{\partial t} = -\frac{1}{a\cos\phi}\frac{\partial[\rho C_p\cos\phi(vT + \overline{v'T'})]}{\partial\phi} + \rho(Q + Q_L) + 1.4RT\left[\frac{\partial\rho}{\partial t} + \frac{v}{a}\frac{\partial\rho}{\partial\phi}\right]$$
$$+1.4RTw\frac{\partial\rho}{\partial z} - \frac{\partial[\rho C_p(wT + \overline{w'T'})]}{\partial z} \tag{2.10}$$

the equation of state,

$$P = \rho RT \tag{2.11}$$

the assumed condition of hydrostatic equilibrium

$$dP = -\rho g dz \tag{2.12}$$

and the moisture balance equation

$$\frac{\partial s\rho}{\partial t} = -\frac{1}{a\cos\phi}\frac{\partial s\rho v\cos\phi}{\partial\phi} - \frac{\partial s\rho w}{\partial z} + C^* - E^* \tag{2.13}$$

In the above equations ρ is the air density (in g/m^3), t is time, ϕ is the latitude coordinate, u, v, and w are the zonal, meridional, and vertical velocities respectively, f is the Coriolis parameter, F_λ and F_ϕ and are the zonal and meridional friction terms, C_p is

the heat capacity of dry air at constant pressure, R is the ideal gas constant for air, Q is the diabatic heating rate per unit mass of air (J/g) due to the fluxes of solar and terrestrial radiation, Q_L is the latent heat release (or absorption) per unit mass, g is the acceleration due to gravity, s is the specific humidity of water vapor, and C* and E* are the rate of evaporation and condensation respectively (in $gH_2O/m^3/s$). The symbols in the above equations are standard and are also defined in Appendix A along with all other major variable symbols used throughout this dissertation in order of their appearance. All variables are assumed to be zonal averages. The prime indicates deviation from the zonal average and the overbar is included when a zonal average is emphasized. The above six equations (2.7 through 2.12) are solved as an initial valued problem using the spatial grid of 9 vertical atmospheric layers and 18 latitude zones, described previously. The pressure, density, and temperature (P, ρ, and T) of a zone are assumed to be the values of the respective variables at the center of the zone; while the vertical, meridional, and zonal velocities (w,v, and u) are calculated at the boundary between zones.

We use a grid system that is common in models of this type and in general circulation models; see Ghan et al. (1982) and Yao and Stone (1987). That is, the center of a grid point is indicated by an integer, while the boundary is designated as an integer -1/2 (see Figure 2.2 below). A grid point is designated by two numbers (i,j), where i is the vertical grid position and j is the horizontal grid position. For example, grid point 1,1 is at the center of the top atmospheric layer directly above 80.3° (85°-4.7°) north latitude, grid 1, 9.5 is at the center of the top atmospheric layer directly above the equator, and grid point 8.5, 18 is at the top of the lowest atmospheric layer directly above 80.3° south latitude. It should be noted that northward moving meridional velocities(south winds) are considered positive, as are eastward moving (westerlies) zonal velocities, and upward moving vertical velocities.

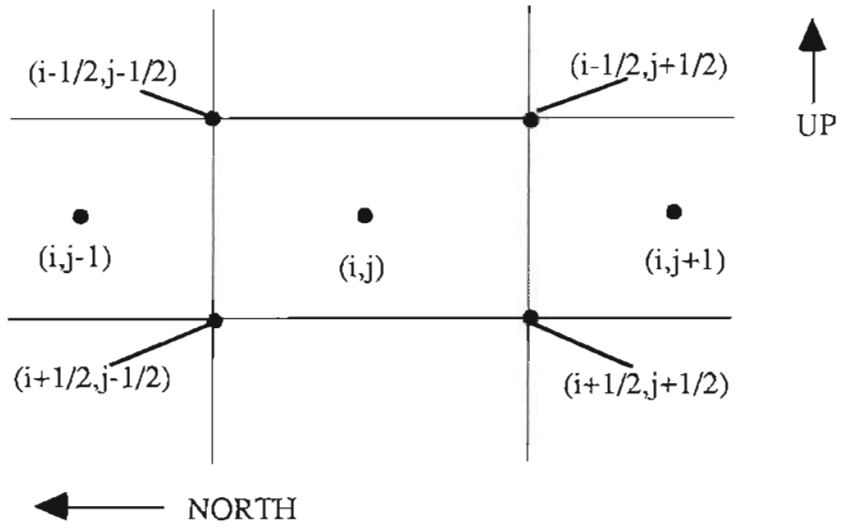

Figure 2.2. The model grid system used for numerical calculations in the GCRC 2-D statistical dynamical climate model.

2.2.1 Numerical Overview

The above equations are solved using a box type numerical procedure similar to those used by Khalil and Rasmussen (1985) for the calculation of the transport of trace gases in the atmosphere, and also similar to that used by Yao and Stone (1987) for their two-dimensional statistical-dynamical model. For example, the first term on the right hand side of equation 2.8 represents the time rate of change of zonal momentum due to the meridional flux of zonal momentum by advection and turbulence. The numerical form of the horizontal advective part of this first term is written as,

$$\Delta[\rho u]_{j+1/2} = \frac{\Delta t}{a\cos\phi_j \Delta\phi} \times$$

$$\left\{\left[\left(\frac{g_{j+1/2}+g_{j+3/2}}{2}\right)\left(\frac{u_{j+1/2}+u_{j+3/2}}{2}\right)\right]-\left[\left(\frac{g_{j-1/2}+g_{j+1/2}}{2}\right)\left(\frac{u_{j-1/2}+u_{j+1/2}}{2}\right)\right]\right\} \qquad (2.14)$$

where

$$g_{j+1/2} = \left(\frac{\rho_j + \rho_{j+1}}{2}\right) v_{j+1/2} \cos\phi_{j+1/2}$$

In the above; the first index i, corresponding to the vertical level, has been omitted since it is the same for each term when we consider horizontal transport only. As another example we consider vertical advection. The second term on the right hand side of equation 2.7 is written as,

$$\Delta\rho_{i,j} = \frac{\Delta t}{\Delta z_i}\left\{\left(\frac{\rho_{i,j}+\rho_{i-1,j}}{2}\right) w_{i-1/2,j} - \left(\frac{\rho_{i,j}+\rho_{i+1,j}}{2}\right) w_{i+1/2,j}\right\} \qquad (2.15)$$

This type of numerical procedure illustrated by the above examples for horizontal and vertical advection, is described rigorously by Kasahara (1977). It has the advantage of being intuitive and it also guarantees the conservation of the quantity being transported whether it is mass, momentum, or energy.

The time integration for the dynamics portion of the model follows a leap-frog scheme similar to that described by Hansen et al. (1983). The source terms: radiative heating, hydrostatic adjustment, and friction are updated every two hours. The dynamical terms: velocities, pressures, densities, and advective heating are calculated using a time step of 15 minutes. The flow of calculations is: radiative heating, ocean energy transport due to turbulent diffusion and ice and snow accumulation, eight iterations of dynamic calculations (eight fifteen minute intervals), frictional drag calculation, and hydrostatic adjustment. The process then repeats in a cyclic manner as shown in figure 2.3.

It is important to note that we have also used an eighth order Shiparo filter on the temperature, meridional velocity, zonal velocity, and surface density fields once each calculation cycle to eliminate instabilities associated with two grid point noise in the solution. This filtering technique which was developed by Shapiro (1970) has the

advantage of maintaining the original general shape of the function being filtered after many filtering iterations and hence it is non-dissipating.

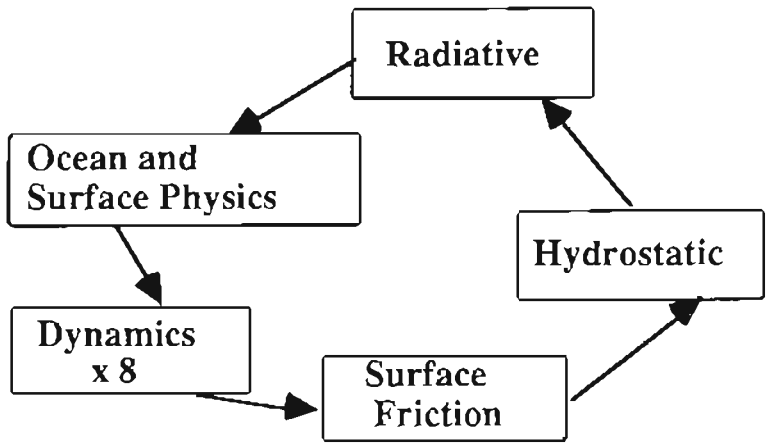

Figure 2.3. The basic flow of the model calculations.

2.3 Solution of the Basic Equations

Below we describe the specifics of the numerical solution of the basic equations for our 2D model. In particular, we investigate how the equations are used to obtain a numerical solution of the whole system. Also in this section, we intend to highlight the assumptions and simplifications used to obtain a solution to the basic equations.

2.3.1 Initial Conditions.

The initial conditions used to start the model calculations are extremely important since, if the initial conditions are too extreme, large amplitude oscillations will grow rapidly and the model outputs will diverge. As discussed in chapter 1, lack of realistic and consistent initial conditions was one of the reasons Richardson's initial attempt at a numerical forecast was so unsuccessful. Our model can run in a two-dimensional energy balance model (EBM) mode quite easily by omitting the calls to the dynamics portion of the program. In the EBM mode, the temperature structure of the model is calculated

using the radiative convective calculations described below coupled with the transfer of horizontal turbulent energy (driven purely from temperature gradients) and vertical convective energy. The steady state solution of the EBM model provides a convenient set of initial conditions for the temperature, density (or pressure), and velocities (which have been set to zero). Using the steady state EBM atmosphere as an initial state, the dynamics portion of the program can then be added as a perturbation to this and a new steady state solution can be obtained which is closer to the real observed atmosphere. Additional refinements can be made until the mean annual output of the model agrees closely with the observed mean annual climate. This solution, obtained by bootstrapping, can then be used as the initial conditions for model climate sensitivity studies due to perturbations in the model climate system. Our assumption is that if the model's initial conditions are similar to the observed state of the real Earth, then model perturbation studies, such as the response of the model to a doubling of CO_2, will help us estimate and understand possible changes and interactions of the real climate system due to similar perturbations of it.

2.3.2 The equation of continuity

The equation of continuity is broken into two parts; horizontal and vertical. The horizontal transport of mass is calculated by the first term on the right hand side of equation (2.7). The spatial differencing scheme used is similar to that of equation (2.14) and is written as,

$$\frac{\Delta \rho_{i,j}}{\Delta t} = \frac{\left[g_{i,j+\frac{1}{2}} - g_{i,j-\frac{1}{2}} \right]}{a \cos \phi_{i,j} \left[\phi_{i,j-\frac{1}{2}} - \phi_{i,j+\frac{1}{2}} \right]}$$

The vertical transport of mass is performed during the hydrostatic adjustment. That is, the total mass in a vertical column is redistributed to satisfy the hydrostatic equation, equation (2.12), and the equation of state, equation (2.11). The hydrostatic

adjustment process transfers mass, latent and sensible heat, and zonal and meridional momentum. The second term on the right hand side of equation (2.7) is used diagnostically to calculate the vertical velocity, w, from the calculated transfer of mass during the hydrostatic adjustment process, i.e.

$$w_{i-\frac{1}{2},j} = \frac{\Delta m_{i-\frac{1}{2},j}}{\Delta t_r (\rho_{i-1,j} + \rho_{i,j})/2.0}$$

where Δm is the mass flux in g/m^2 across the i-1/2 boundary required to reestablish hydrostatic equilibrium in one radiative time step Δt_r=2.0 hr. The process of hydrostatic adjustment is discussed in more detail in section 2.3.5.

2.3.3 Zonal Velocity

The finite difference form of the first term on the right hand side of equation (2.8) is written as in equation (2.14). The turbulent transport of zonal momentum, the $\overline{u'v'}$ part of this term, is parameterized following Stone and Yao (1987 and 1990). Since this section is a description of the numerical technique used, we will postpone a discussion of this parameterization until a section 2.4.1. The first part of the second term in equation (2.8) is calculated as a flux of zonal momentum during the hydrostatic adjustment process,

$$\left[\frac{\Delta \rho u_{i,j-\frac{1}{2}}}{\Delta t_r}\right]_{i+\frac{1}{2},j} = \frac{\Delta m_{i+\frac{1}{2},j}\left(u_{i,j-\frac{1}{2}} + u_{i+1,j-\frac{1}{2}}\right) + \Delta m_{i+\frac{1}{2},j-1}\left(u_{i,j-\frac{1}{2}} + u_{i+1,j-\frac{1}{2}}\right)}{2.0 \times \Delta t_r (z_{i-\frac{1}{2},j} - z_{i+\frac{1}{2},j})}$$

The second part of this term, $\overline{u'w'}$ is assumed to depend on the vertical shear of zonal velocity u according to

$$\frac{\partial \rho \overline{u'w'}}{\partial z} = \frac{\partial}{\partial z} \rho K_z \frac{\partial u}{\partial z} \qquad (2.16)$$

The values of K_z were originally estimated from those given by Liu et al. (1984) but it was found that we could not obtain the Ferrel circulations cells in either hemisphere with these values of K_z which were based on Radon measurements. We have thus used Kz=10 m^2/s everywhere except between layers 1 and 2 and layer 2 and 3 at the top of the model where we have used values of K_z=20 m^2/s and 4 m^2/s respectively to inhibit high zonal winds developing in the stratosphere. Stone and Yao (1987) completely eliminated the vertical turbulent flux of zonal momentum from their calculations noting that the output of their model was not significantly affected by whether they included a parameterization for K_z or not. As noted above, we did find that our output was significantly affected by the choice in K_z profile so in this regard our model behaves differently from the model of Stone and Yao. It is also noteworthy that when we experimented with setting all values of Kz to zero we obtained tropical easterlies that were much greater than observed. This result is not surprising since, as will be shown in chapter 3, our model has a propensity for strong tropical easterlies.

The numerical form of the $\overline{u'w'}$ term in equation (2.5) used in the model is given by,

$$\frac{\partial \rho u' w'}{\partial z} =$$

$$\left\{ K_{z_{i-\frac{1}{2},j-\frac{1}{2}}} \rho^{*}_{i-\frac{1}{2},j-\frac{1}{2}} \frac{\left(u_{i-1,j-\frac{1}{2}} - u_{i,j-\frac{1}{2}} \right)}{z_{i-1,j-\frac{1}{2}} - z_{i,j-\frac{1}{2}}} - K_{z_{i+\frac{1}{2},j-\frac{1}{2}}} \rho^{*}_{i+\frac{1}{2},j-\frac{1}{2}} \frac{\left(u_{i,j-\frac{1}{2}} - u_{i+1,j-\frac{1}{2}} \right)}{z_{i,j-\frac{1}{2}} - z_{i+1,j-\frac{1}{2}}} \right\}$$

$$\div \left(z_{i-\frac{1}{2},j-\frac{1}{2}} - z_{i+\frac{1}{2},j-\frac{1}{2}} \right)$$

where,

$$\rho^{*}_{i-\frac{1}{2},j-\frac{1}{2}} = \frac{1}{4}\left(\rho_{i,j} + \rho_{i-1,j} + \rho_{i,j-1} + \rho_{i-1,j-1} \right)$$

The third and fourth terms in equation (2.8) can be written as

$$\left[\frac{\Delta(\rho u)_{i,j-\frac{1}{2}}}{\Delta t}\right]_{Terms\_3,4} = \left(\frac{\rho_{i,j}+\rho_{i,j-1}}{2.0}\right) v_{i,j-\frac{1}{2}} \left[f_{i,j-\frac{1}{2}} + \frac{u_{i,j-\frac{1}{2}} \tan\phi_{j-\frac{1}{2}}}{a}\right]$$

The final term in equation (2.8), the frictional drag force, is assumed to be non-zero only at the Earth's surface. As is standard practice (see for example Washington and Parkinson, 1986) we assume that the surface shear stress in the meridional and longitudinal direction τ_ϕ and τ_λ are given by

$$\tau_\phi = -\rho C_D v\sqrt{u^2+v^2} \quad and \quad \tau_\lambda = -\rho C_D u\sqrt{u^2+v^2}$$

where the surface drag coefficient C_D is of the order of 0.001 to 0.003 (we use 0.002). The surface horizontal frictional forces are given by

$$F_\phi = \frac{1}{\rho}\frac{\partial \tau_\phi}{\partial z} \quad and \quad F_\lambda = \frac{1}{\rho}\frac{\partial \tau_\lambda}{\partial z}$$

The finite difference form for the frictional term in equation (2.8) is thus

$$\left[F_\lambda\right]_{9,j-\frac{1}{2}} = \left[\frac{1}{\rho}\frac{\partial \tau_\lambda}{\partial z}\right]_{9,j-\frac{1}{2}} =$$

$$-C_D u_{9,j-\frac{1}{2}} \sqrt{u_{9,j-\frac{1}{2}}^{\,2} + v_{9,j-\frac{1}{2}}^{\,2}} \div \left(z_{8\frac{1}{2},j-\frac{1}{2}} - z_{9\frac{1}{2},j-\frac{1}{2}}\right)$$

2.3.4 Meridional Velocity

Except for the first term in equation (2.9), the finite difference form of each term in equation (2.9) is essentially identical to that used for equation (2.8) and hence will not be repeated here. The first term of equation (2.6) is calculated as

$$\left[\frac{\partial \rho v_{i,j-\frac{1}{2}}}{\partial t_d}\right]_{term\_1} = \frac{1}{a}\frac{p_{i,j+\frac{1}{2}} - p_{i,j-\frac{1}{2}}}{\phi_{j-\frac{1}{2}} - \phi_{j-\frac{1}{2}}}$$

and the $\overline{v'w'}$ is estimated by,

$$\frac{\partial \rho \overline{v'w'}}{\partial z} = \frac{\partial}{\partial z} \rho K_z \frac{\partial v}{\partial z} \tag{2.17}$$

2.3.5 The first Law of thermodynamics

Since the form of the first law of thermodynamics used by us in the numerical calculation of heat transfer is somewhat unique, we derive it below for clarity and then explain the numerical procedure used to calculate the change in temperature of a grid point in the 2-D model atmosphere. The basic form of the first law of thermodynamics (see Lorentz, 1967) is,

$$C_p dT = (Q + Q_L)dt + \alpha dP \tag{2.18}$$

where Cp is the heat capacity (J/g/K) of the atmosphere at constant pressure, dT is the infinitesimal temperature change in time interval dt, Q_T the diabatic heating rate per unit mass (which includes radiative heating and release or absorption of latent heat), α is the specific volume which is the reciprocal of density ρ, and dP is the infinitesimal pressure change. We assume the temperature change dT occurs in a two step process; first an adiabatic temperature change due to the pressure change dP and then a diabatic temperature change due to the diabatic heating Q. For the adiabatic process it is easy to show that

$$\frac{dT}{T} = \kappa \frac{dP}{P} \quad where \quad \kappa = \frac{R}{C_p} \tag{2.19}$$

Taking the differential of the equation of state $P = \rho RT$ and multiplying both sides by α yields

$$\begin{aligned} \alpha dP &= \alpha d\rho RT + \alpha \rho R dT \\ &= \alpha d\rho RT + R dT \\ &= \alpha d\rho RT + \frac{R\kappa T}{P} dP \\ &= \alpha d\rho RT + \alpha \kappa dP \end{aligned}$$

or

$$dP = \frac{RTd\rho}{(1-\kappa)} = \gamma RTd\rho \approx 1.4RTd\rho$$

since

$$\frac{1}{(1-\kappa)} = \frac{C_p}{C_p - R} = \frac{C_p}{C_v} = \gamma \approx 1.4$$

Thus we write the first law as,

$$C_p \frac{dT}{dt} = (Q + Q_L) + 1.4\alpha RT \frac{d\rho}{dt} \tag{2.20}$$

Since

$$\frac{d}{dt} = \frac{\partial}{\partial t} + \langle u, v, w \rangle \cdot \nabla = \frac{\partial}{\partial t} + \frac{v}{a}\frac{\partial}{\partial \phi} + w\frac{\partial}{\partial z}$$

we can rewrite (2.20) as

$$\rho C_p \frac{\partial T}{\partial t} = -\rho C_p \frac{v}{a}\frac{\partial T}{\partial \phi} - \rho C_p w \frac{\partial T}{\partial z} + \rho(Q + Q_L) + \frac{\rho 1.4RT}{\rho}\left[\frac{\partial \rho}{\partial t} + \frac{v}{a}\frac{\partial \rho}{\partial \phi} + w\frac{\partial \rho}{\partial z}\right] \tag{2.21}$$

At this point we could derive equation (2.10) without the turbulent terms, by multiplying the equation of continuity by C_pT and adding it to (2.21); instead we proceed in an alternate direction to derive our working numerical equations.

We break (2.21) into three parts corresponding to horizontal transport, vertical transport, and heating due to radiant heat energy and latent heat (Q),

$$\rho C_p \frac{\partial T}{\partial t} = -\rho C_p \frac{v}{a}\frac{\partial T}{\partial \phi} + 1.4RT\left[\frac{\partial \rho}{\partial t} + \frac{v}{a}\frac{\partial \rho}{\partial \phi}\right], \tag{2.22}$$

$$\rho C_p \frac{\partial T}{\partial t} = -\rho C_p w \frac{\partial T}{\partial z} + 1.4RT\left[\frac{\partial \rho}{\partial t} + w\frac{\partial \rho}{\partial z}\right], \tag{2.23}$$

and

$$C_p \frac{\partial T}{\partial t} = (Q + Q_L) \tag{2.24}$$

where $\frac{\partial T}{\partial t}$ and $\frac{\partial \rho}{\partial t}$ in each of equations (2.22) and (2.23) correspond to the changes that occur during horizontal or vertical transport separately and $\frac{\partial T}{\partial t}$ in (2.24) corresponds to the temperature change due to radiant heating or heat by latent heat energy ($Q+Q_L$). The radiative heating and energy transport associated with the convective adjustment, is calculated in the radiative part of the model and will be discussed below in the overview of the radiative convective model physics. The parameterization of the heating associated with the release of latent heat is discussed in section 2.4.4.

Using the continuity equation (2.7) the

$$\frac{\partial \rho}{\partial t}+\frac{v}{a}\frac{\partial \rho}{\partial \phi}$$

part of equation (2.22) can be written as ,

$$\frac{\partial \rho}{\partial t}+\frac{v}{a}\frac{\partial \rho}{\partial \phi}=-\frac{1}{a\cos\phi}\frac{\partial \rho v\cos\phi}{\partial \phi}+\frac{v\cos\phi}{a\cos\phi}\frac{\partial \rho}{\partial \phi}=-\frac{\rho}{a\cos\phi}\frac{\partial v\cos\phi}{\partial \phi}$$

we thus rewrite (2.22) as,

$$\frac{\partial T}{\partial t}=-\frac{v}{a}\frac{\partial T}{\partial \phi}-\frac{1.4RT}{C_p}\left[\frac{1}{a\cos\phi}\frac{\partial v\cos\phi}{\partial \phi}\right] \tag{2.25}$$

The numerical form of equation (2.25) used is

$$\frac{\Delta T}{\Delta t_d}=\left(\frac{1.4R}{C_p}+1\right)\frac{\left[g_{j+\frac{1}{2}}\left(\frac{T_{i,j}+T_{i,j+1}}{2.0}\right)-g_{j-\frac{1}{2}}\left(\frac{T_{i,j}+T_{i,j-1}}{2.0}\right)\right]}{a\rho_{i,j}\cos\phi_j\left(\phi_{j-\frac{1}{2}}-\phi_{j+\frac{1}{2}}\right)}$$

and the $\overline{T'v'}$ term of equation (2.10) is written as

$$\frac{\Delta T}{\Delta t_d}=\frac{\left[T'v'_{i,j+\frac{1}{2}}\left(\frac{\rho_{i,j}+\rho_{i,j+1}}{2.0}\right)\cos(\phi_{j+\frac{1}{2}})-T'v'_{i,j-\frac{1}{2}}\left(\frac{\rho_{i,j}+\rho_{i,j-1}}{2.0}\right)\cos(\phi_{j-\frac{1}{2}})\right]}{a\rho_{i,j}\cos\phi_j\left(\phi_{j-\frac{1}{2}}-\phi_{j+\frac{1}{2}}\right)}$$

where $\overline{T'v'}$ is parameterized following Stone and Yao (1990) as disscussed below.

Equation (2.21) can be rewritten as,

$$\frac{\partial T}{\partial t}=-w\left[\frac{\partial T}{\partial z}+\frac{g}{C_p}\right]+\frac{1.4RT}{\rho C_p}\frac{\partial \rho}{\partial t} \tag{2.26}$$

if we note that

$$1.4RTw\frac{\partial \rho}{\partial z}=1.4RTw\frac{1}{1.4RT}\frac{\partial P}{\partial z}=-\rho g w$$

and make use of the assumption of hydrostatic equilibrium. Note that the first term on the right hand side of (2.26) is the transport of potential temperature and the second term is due to the adiabatic heating (cooling) due to increase (decrease) of the density of the atmospheric layer as a result of advection.

2.3.5 Hydrostatic Adjustment

The method we use to restore the atmosphere to hydrostatic equilibrium relies on the transport of mass between adjacent vertical layers until both the equation of state,

$$P = \rho RT$$

and the condition of hydrostatic equilibrium

$$dP = -\rho g dz$$

are satisfied. For example, if the pressure difference between the two vertical layers i and i+1 is too large, then mass is transported from i+1 to i according to,

$$\left(P_{i+1} - dP_{i+1}\right) - \left(P_i + dP_i\right) = (\rho_{i+\frac{1}{2}} + \Delta\rho_{i+\frac{1}{2}})g\left(z_i - z_{i+1}\right) \tag{2.27}$$

where dP_{i+1} (dP_i) is pressure change of layer i+1 (i) due to mass flow out of (into) that layer and

$$\rho_{i+\frac{1}{2}} = \frac{\rho_i + \rho_{i+1}}{2.}$$

Rewriting (2.27) we have

$$\left(P_{i+1} - P_i\right) - \rho_{i+\frac{1}{2}} g\left(z_i - z_{i+1}\right) = dP_{i+1} + dP_i + \Delta\rho_{i+\frac{1}{2}} g\left(z_i - z_{i+1}\right) \tag{2.28}$$

As a finite amount of mass Δm flows out of (or into) a layer the layer's density, pressure, and temperature can change. Taking the differential of both sides of the equation of state we find,

$$dP = d\rho RT + \rho R dT \tag{2.29}$$

We now note that for vertical transport equation (2.26) predicts that the change in temperature dT of the layer in a time increment dt is,

$$dT = -wdt\left[\frac{\partial T}{\partial z} + \frac{g}{C_p}\right] + \frac{1.4RT}{\rho C_p} d\rho$$

Combining this expression with (2.29) and noting that 1.4R/Cp=0.4 gives

$$dP = -\rho R w dt\left[\frac{\partial T}{\partial z} + \frac{g}{C_p}\right] + 1.4RTd\rho \tag{2.30}$$

When a finite amount of mass Δm is transported from layer i+1 to layer i across the boundary between them (i+1/2) we have,

$$\rho_{i+\frac{1}{2}} w_{i+\frac{1}{2}} dt = \Delta m_{i+\frac{1}{2}} \quad and \quad \Delta\rho_i = \frac{\Delta m_{i+\frac{1}{2}}}{\left(z_{i-\frac{1}{2}} - z_{i+\frac{1}{2}}\right)}$$

We can thus rewrite (2.30) numerically as,

$$\Delta P_i = \frac{\left(\Delta m_{i+\frac{1}{2}} - \Delta m_{i-\frac{1}{2}}\right) 1.4RT_i}{\left(z_{i-\frac{1}{2}} - z_{i+\frac{1}{2}}\right)} +$$

$$\frac{\Delta m_{i+\frac{1}{2}}}{\left(z_{i-\frac{1}{2}} - z_{i+\frac{1}{2}}\right)} R\left(\frac{(T_{i+1} + T_i)}{2.} - \frac{g}{C_p}\left(z_i - z_{i+\frac{1}{2}}\right)\right) - \frac{\Delta m_{i-\frac{1}{2}}}{\left(z_{i-\frac{1}{2}} - z_{i+\frac{1}{2}}\right)} R\left(\frac{(T_{i-1} + T_i)}{2.} + \frac{g}{C_p}\left(z_{i-\frac{1}{2}} - z_i\right)\right)$$

or

$$\Delta P_i = \xi_{i,i+\frac{1}{2}} \Delta m_{i+\frac{1}{2}} - \xi_{i,i-\frac{1}{2}} \Delta m_{i-\frac{1}{2}} \tag{2.31}$$

where

$$\xi_{i,i+\frac{1}{2}} = \frac{R}{\left(z_{i-\frac{1}{2}} - z_{i+\frac{1}{2}}\right)}\left(1.4T_i + \frac{(T_{i+1} + T_i)}{2.} - \frac{g}{C_p}\left(z_i - z_{i+\frac{1}{2}}\right)\right)$$

and

$$\xi_{i,i-\frac{1}{2}} = \frac{R}{\left(z_{i-\frac{1}{2}} - z_{i+\frac{1}{2}}\right)}\left(1.4T_i + \frac{(T_{i-1} + T_i)}{2.} - \frac{g}{C_p}\left(z_{i-\frac{1}{2}} - z_i\right)\right)$$

The parts of ΔP_i associated with mass transport across the i+1 or i boundary can thus easily be identified. We can rewrite (2.28) as

$$B_{i+\frac{1}{2}} = dP_{i+1} + dP_i + \Delta\rho_{i+\frac{1}{2}} g\left(z_i - z_{i+1}\right)$$

where we have defined $B_{i+1/2}$ to be identical to the left hand side of (2.28). Combining this with equation (2.31) and noting that

$$\Delta\rho_{i+\frac{1}{2}} = \frac{\Delta\rho_i + \Delta\rho_{i+1}}{2.}$$

$$= \left(\frac{\left(\Delta m_{i+\frac{1}{2}} - \Delta m_{i-\frac{1}{2}} \right) \Big/ \left(z_{i-\frac{1}{2}} - z_{i+\frac{1}{2}} \right) + \left(\Delta m_{i+\frac{3}{2}} - \Delta m_{i+\frac{1}{2}} \right) \Big/ \left(z_{i+\frac{1}{2}} - z_{i+\frac{3}{2}} \right)}{2.0} \right)$$

gives

$$B_{i+\frac{1}{2}} = -\left(\xi_{i,i-\frac{1}{2}} + \frac{0.5g\left(z_i - z_{i+1}\right)}{\left(z_{i-\frac{1}{2}} - z_{i+\frac{1}{2}} \right)} \right) \Delta m_{i-\frac{1}{2}}$$

$$+ \left(\xi_{i,i+\frac{1}{2}} - \xi_{i+1,i+\frac{1}{2}} + \frac{0.5g\left(z_i - z_{i+1}\right)}{\left(z_{i-\frac{1}{2}} - z_{i+\frac{1}{2}} \right) - \left(z_{i+\frac{1}{2}} - z_{i+\frac{3}{2}} \right)} \right) \Delta m_{i+\frac{1}{2}}$$

$$+ \left(\xi_{i+1,i+\frac{3}{2}} + \frac{0.5g\left(z_i - z_{i+1}\right)}{\left(z_{i+\frac{1}{2}} - z_{i+\frac{3}{2}} \right)} \right) \Delta m_{i+\frac{3}{2}} \qquad (2.32)$$

We have eight such equations for the eight boundaries (1 1/2 to 8 1/2) that we allow vertical mass flux (i.e. no mass flux at the top of the atmosphere and no mass flux at the Earth's surface). We can thus express (2.32) as a matrix equation and solve it for the mass fluxes needed across a boundary to restore a vertical column to hydrostatic equilibrium.

$$\vec{B} = \underline{\xi}\vec{m} \quad or \quad \vec{m} = \underline{\xi}^{-1}\vec{B}$$

where an arrow above represents a column vector and the underbar represents a matrix.

2.3.6 Atmospheric Moisture

For moisture, we assume the constant relative humidity profile for each latitude zone introduced by Manabe and Wetherald (1967) and used by MacKay and Khalil (1991). The relative humidity r is expressed in terms of the surface relative humidity h' and the pressure p (in atmospheres) by

$$r = r_o(P - 0.02) / 0.98$$

We use data from the Data Support Section, Scientific Computing Division, of NCAR, DS205.0, N Hem. Climatological Grid Data (N.H. and S.H.) for the surface relative humidity of each zone for each season. These data are presented in tabular form in table 2.2 and graphically in figure 2.4. We use a linear interpolation of the seasonal averages to estimate the surface relative humidity at any give time.

Table 2.2. Surface relative humidity r_0 (in percent) used for the calculation of water vapor mixing ratio in the GCRC 2D Model. From NCAR, Data support section, Scientific Computing Division.

Latitude (Deg)	MAM	JJA	SON	DJF
-80.3	58	58	64	72
-70.8	63	66	66	71
-61.4	79	78	77	76
-51.9	82	79	80	82
-42.5	77	76	77	79
-33.1	73	74	73	73
-23.6	69	68	66	69
-14.2	74	70	70	75
-4.7	78	77	77	79
4.7	77	79	79	76
14.2	65	73	72	67
23.6	61	65	66	66
33.1	64	64	66	70
42.5	68	69	72	76
51.9	74	76	80	80
61.4	76	79	83	79
70.8	80	85	80	73
80.3	79	87	77	75

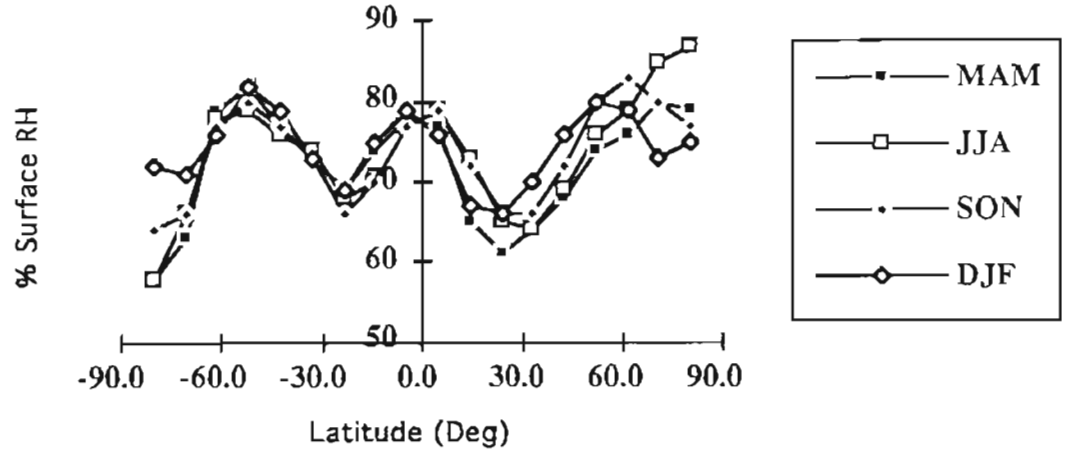

Figure 2.4. Surface relative humidity (%) used for the calculation of water vapor mixing ratio in the GCRC 2D Model. From table 2.2.

The specific humidity, s, is related to the temperature T and the relative humidity r by

$$s = rs^*(T) \qquad s > 3.0\times10^{-6}\,\frac{gH_2O}{gAir}$$

$$s = 3.0\times10^{-6} \qquad rs^*(T) < 3.0\times10^{-6}\,\frac{gH_2O}{gAir}$$

where s^* is the saturation vapor pressure. The saturation vapor pressure is assumed to depend on the temperature according to the Clausius Clapeyron equation,

$$s^*(T) = s^*(273K)\exp\left[\frac{0.622L}{R}\left(\frac{1}{273K}-\frac{1}{T}\right)\right]$$

see for example Washington and Parkinson (1986). L is the Latent Heat of vaporization (2510-2.38[T-273]J/g) from Stone and Carlson 1979, R the ideal gas constant (J/[kgK]), and $s^*(273)=3.75\text{x}10^{-3}$ $gH_2O/gair$. Thus as the temperature of a model grid point changes, the water vapor content also changes.

We use a modified specific heat capacity Cp* identical to that used by Manabe and Wetherald (1967) to account for the greater thermal inertia of the atmosphere due to an assumed fixed relative humidity,

$$C_p^* = C_p\left[1 + \frac{L_v}{C_p}\frac{\partial s}{\partial T}\right]$$

We have replaced Cp by Cp* on the left side of equation (2.10), the first law of thermodynamics. That is, the thermal inertia of the moist layer is assumed to be larger than a dry layer but the dry sensible heat capacity of the layer is the same for both a dry and moist layer.

We can use equation 2.13 to diagnostically calculate the difference between the condensation and evaporation for a model grid point. To estimate the precipitation for a given grid point we rely on the fact that its actual moisture content is in a quasi-steady state, since it only changes as the temperature changes. We assume that all cooling processes during a time step results in precipitation. We thus estimate the rate of precipitation $\dot{P}$ in mm/day by,

$$\dot{P} = \frac{\partial s}{\partial T}\frac{\Delta T_R^C + \Delta T_D^C}{\Delta t_R}\frac{\rho_{air}}{\rho_w}\Delta z$$

where ΔT_R^C and ΔT_D^C are the magnitudes of the cooling term associated with energy loss during the radiative (including convection) and dynamical (including hydrostatic adjustment) calculations respectively. This estimate of precipitation works fairly well considering our present simplification of the moisture budget, see section 3.4 of the next chapter. Improvements in precipitation estimates are expected following a more comprehensive treatment of our model hydrodynamics. Future model improvements will include a prognostic calculation of the moisture budget as well as a prognostic determination of clouds.

2.4 Other Parameterizations

2.4.1 Parameterization of Eddy Momentum Flux $\overline{u'v'}$.

For the flux of eddy momentum $\overline{u'v'}$,we use the parameterization scheme given by Yao and Stone (1987) with the slight modifications described by Stone and Yao (1990) As they describe their parameterization in great detail, we will only outline the

essentials of their scheme below. We will attempt to give enough information so the reader can reproduce our calculations from this dissertation as well as understand the underlying theory of the work of Yao and Stone. Starting from the equations of motion (2.7)-(2.9) and the continuity equation, Pedlosky (1979) in chapter 6 of his classic text on geophysical fluid dynamics, derives the equation for the conservation of potential vorticity (q) on a beta plane using the quasigeostrophic approximation (his equations 6.5.32 or 6.5.19). In dimensional form his equation takes the form of,

$$\left[\varsigma + \beta_o y + \frac{f_o}{\rho_s}\frac{\partial}{\partial z}\left(\frac{\rho_s}{N_s^2}\frac{\partial}{\partial z}\frac{\delta p}{\rho_s}\right)\right] = 0$$

where (2.33)

$$\varsigma = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$$

is the relative vorticity , β_o is the rate of change of the Coriolis parameter f (in the vicinity of some reference position corresponding to f_0) with respect to meridional coordinate y, ρ_s is the density of the stationary state, x and z are the longitudinal and vertical coordinates (both in meters), δp is the first order perturbation from the stationary state of the pressure , and N_S is the *Brunt – Väisälä* frequency of the stationary state,

$$N_s^2 = \frac{g}{\theta_s}\frac{d\theta_s}{dz}$$

with θ_s being the potential temperature of the stationary state. Note that (2.33) is a perturbation equation about some mid latitude position y_0, usually taken to be 45 degrees.

If we note that for geostrophic flow,

$$u = -\frac{\partial \Psi}{\partial y} \quad and \quad v = \frac{\partial \Psi}{\partial x}$$

where

$$\Psi = \frac{\delta p}{f_o \rho_s}$$

and using this definition for Ψ, we can rewrite (2.33) as,

$$\frac{\partial q}{\partial t} = -J(\Psi, q)$$

where

$$q = \left[\varsigma + \beta_o y + \frac{f_o}{\rho_s} \frac{\partial}{\partial z} \left(\frac{\rho_s}{N_s^2} \frac{\partial}{\partial z} \frac{\delta p}{\rho_s} \right) \right]$$

$$= \nabla^2 \Psi + \beta_o y + \frac{f_o^2}{\rho_s} \frac{\partial}{\partial z} \left(\frac{\rho_s}{N_s^2} \frac{\partial}{\partial z} \Psi \right) \qquad (2.34)$$

and

$$J(A,B) = \frac{\partial A}{\partial x} \frac{\partial B}{\partial y} - \frac{\partial A}{\partial y} \frac{\partial B}{\partial x}$$

is the Jacobian of the functions A and B. Held (1978) starts from (2.34) to show that the height parameter

$$h = f_o^2 \frac{\partial \overline{u}}{\partial z} \frac{1}{\beta N^2}$$

appearing often in the model of Charney (1947), is related to the vertical structure of the baroclinic disturbance. Held notes that when h<<H "the amplitude of the most unstable wave decays more or less exponentially above the ground with an e-folding proportional to h " (H is the atmospheric scale height). If h>>H then H takes the role of the vertical scale for the most unstable wave. Held shows that the parameter

$$d = H / (1 + \gamma)$$
$$\text{where} \quad \gamma = H / h \qquad (2.35)$$

is a good measure of the vertical structure of the baroclinic disturbance.

It is worth noting that in Chapter 7 of his book, Pedlosky (1979) discusses instability theory in some detail. In sections 7.6-7.8, he discusses the basic mechanism of baroclinic instability, the simple baroclinic model of Eady (1949), and Charney's model. Other texts contain a review of this material, but in my opinion, Pedlosky's text is by far the most authoritative on baroclinic instability as it pertains to the parameterization of Yao and Stone (1987, 1990).

Yao and Stone (1987) begin the derivation of their parameterization by using the relationship between the eddy fluxes of zonal momentum per unit mass u, potential temperature, and quasi-geostrophic potential vorticity q given by Green (1970)

$$\frac{\partial}{\partial y}\overline{u'v'} = f\frac{\partial}{\partial p}\frac{\overline{\theta'v'}}{\sigma} - \overline{q'v'}$$
$$\text{where} \quad \sigma = \frac{\partial \theta_s}{\partial p} \tag{2.36}$$

An alternate derivation of this equation, to that of Green, can be obtained by combining equations 7.2.24 and 7.2.22 of Pedlosky. Green's parameterization also assumes that the two terms on the right hand side of (2.36) can be parameterized by a mixing-length formulation as,

$$\overline{\theta'v'} = -K_{vy}\frac{\partial \overline{\theta}}{\partial y} - K_{vp}\sigma$$

$$\overline{q'v'} = -K_{vy}\frac{\partial \overline{q}}{\partial y} - K_{vp}\frac{\partial \overline{q}}{\partial p}$$

Stone and Yao (1987) then show that combining these assumptions, using the thermal wind relation, and an equation equivalent to our (2.34) they obtain

$$\frac{\partial}{\partial y}\int_0^{P_s}\overline{u'v'}dp$$
$$= \int_0^{P_s}\left\{K_{vy}\left(\beta - \frac{\partial^2 u}{\partial y^2}\right) - \frac{f}{\sigma}\frac{\partial \theta}{\partial y}\frac{\partial K_{vy}}{\partial p}\right\}dp \tag{2.37}$$

for the gradient of the flux of zonal momentum per unit mass. Thus only K_{vy} needs to be parameterized to calculate the vertical mean eddy momentum flux. They estimate the vertical profile of $\overline{u'v'}$ to be the same at each latitude and equal to the average vertical profile obtained by their 3-D control run obtained using the GISS Model II GCM described by Hansen et al. (1983). They then experimented with the vertical structure of $\overline{u'v'}$ and found that the general circulation was not too strongly dependent on it, and hence suggested that a fixed vertical structure based on their 3-D control run was valid. We use the same vertical structure for our 2-D model and have also experimented with the dependence of the assumed vertical structure of $\overline{u'v'}$ on the general circulation of our model.

To complete the parameterization scheme, Stone and Yao (1987) used the parameterization of K_{vy} developed by Branscome (1983) for eddy heat transport parameterization, with several modifications. In the notation of Stone and Yao (1987) Branscome's parameterization for K_{vy} is

$$K_{BC} = \frac{1}{\sqrt{2}} \left\langle \frac{\partial u}{\partial z} \right\rangle \frac{\langle N \rangle}{f} d^2 e^{-1/d}$$

where the triangular brackets indicate a vertical average of any variable x,

$$\langle x \rangle = \frac{\int_0^\infty x e^{-1/d} dz}{\int_0^\infty e^{-1/d} dz}$$

As in Stone and Yao (1990), d , the vertical scale of the baroclinic disturbance, is given by (2.35)

$$d = H / (1 + \gamma) \qquad \text{where} \quad \gamma = H / h \tag{2.35}$$

using

$$\gamma = \frac{\beta \langle H \rangle \langle N^2 \rangle}{f^2 \langle \partial u / \partial z \rangle}$$

Stone and Yao (1987) note that although Branscome's parameterization for K_{vy} works well for eddy heat transport, it falls short, physically, if used directly in (2.37) since it would predict convergence of momentum flux at all latitudes. They thus suggest using a K_{vy} modified slightly from Branscome's

$$K_{vy} = K_{BC} + K_{NL} \tag{2.38}$$

where K_{NL} represents non-linear effects. A separate K_{NL} is calculated for each hemisphere by requiring that momentum is conserved in each hemisphere, i.e. no net convergence of eddy momentum flux. Since the non-linear effects are assumed to penetrate deeply in the vertical direction, K_{NL} is taken to be independent of height. Also the disturbance is assumed to propagate meridionally through regions of westerly winds, but decay exponentially in regions of easterly winds. They thus take K_{NL} to be of the form

$$K_{NL} = \begin{cases} K_o, [\bar{u}] > 0 \\ K_o e^{y/L}, [\bar{u}] < 0 \end{cases} \tag{2.39}$$

where y is the distance into the region of easterlies from $[\bar{u}] = 0$, and L is the horizontal scale depth. Stone and Yao (1990), based on experiments with their model, suggest a

value of L=313 km as an appropriate scale depth. To include the effects of moisture Stone and Yao (1987) modify (2.37) to be

$$\frac{\partial}{\partial y}\int_0^{p_s}\overline{u'v'}dp = \int_0^{P_s}\left\{K_{vy}\left(\beta-\frac{\partial^2 u}{\partial y^2}\right)-\frac{(1+M)\mathrm{f}}{\sigma_e}\frac{\partial\theta}{\partial y}\frac{\partial K_{vy}}{\partial p}\right\}dp \tag{2.40}$$

where

$$M=\frac{L_v}{C_p}r\frac{\partial s^*}{\partial T} \quad and \quad \sigma_e=\sigma+\frac{L_v}{C_p}\left(\frac{p_o}{p}\right)^{\kappa}\frac{\partial s}{\partial p}$$

In the above, r, s*, and s are the relative humidity, saturation specific humidity, and specific humidity of the basic state respectively. To summarize, for the parameterization of the eddy momentum flux, we use (2.40) combined with (2.38) and (2.39). The results of this parameterization will be shown in the next chapter.

In the next chapter we describe a modification to the above scheme that seems to improve our results for the zonal velocity. In the modified version of the Stone and Yao scheme for the calculation of $\overline{u'v'}$ we have simply set

$$K_{BC}=0 \quad if \quad |\phi|>\phi^* \tag{2.41}$$

where ϕ^* is some prescribed high latitude position. In our modified parameterization scheme we have set ϕ^*=65°.

As shown in the next chapter this modification seems to improve the agreement between observed and modeled values of $\overline{u'v'}$ as well as enhance the stability of the model in the polar regions. Our justification for this modification is purely based on empirical evidence at present. However, the realization that the Stone and Yao parameterization scheme disagrees with observations in the polar regions and that our modification seems to improve the parameterization there, is a motivating factor for future theoretical research.

2.4.2 Meridional Eddy Flux of Heat Energy $\overline{v'T'}$

For the parameterization of the meridional flux of heat energy we use the parameterization of Branscome (1983) with the modifications described by Stone and

Yao (1990). In terms of the variables discussed above, the final form of their parameterization is

$$\overline{v'\theta'} = 0.6\frac{gd^2\langle N\rangle}{\langle\bar{\theta}\rangle f^2}\left(\left\langle\frac{\partial\bar{\theta}}{\partial y}\right\rangle\right)^2 \exp\left\{-\frac{z}{\langle H\rangle}(1.48\gamma + 0.48)\right\} \times\left[1-\exp\left\{-\frac{z}{.45km}\right\}\right] \tag{2.42}$$

The physical justification of this parameterization is based on the quasigeostrophic perturbation theory outline above for the $\overline{u'v'}$ parameterization. For use in (2.10) we multiply (2.42) by the appropriate conversion factor $(P/P_s)^\kappa$ to obtain $\overline{v'T'}$. Equation (2.42) alone works well at simulating the general latitudinal dependency of the eddy heat flux, but we find our implementation of it alone underestimates the total turbulent flux of heat energy from the tropics to the high latitude regions and hence results in equator to pole temperature gradients that are too large. We thus add to (2.42) a second term $\overline{v'\theta'}_2$ that depends simply on the square of the latitudinal temperature gradient as,

$$\overline{v'\theta'}_2 = -C_T\left(\frac{\partial\bar{\theta}}{\partial y}\right)\left|\frac{\partial\bar{\theta}}{\partial y}\right|\left[1-\exp\left\{-\frac{z}{.45km}\right\}\right] \tag{2.43}$$

where C_T is an adjustable parameter of the model. We found this additional term was needed in our model primarily for the higher layers of the troposphere where equation (2.42) fails to simulate the secondary maxima observed in both hemispheres in the midlatitudes near the tropopause. We also multiply (2.43) by the appropriate conversion factor $(P/P_s)^\kappa$ to obtain $\overline{v'T'}_2$.

2.4.3 Vertical Eddy Flux of Heat Energy $\overline{w'\theta'}$

We again follow Stone and Yao (1990) for an estimation of the vertical flux of eddy heat energy. We do not go into the details of their parameterization here but only give it in its basic form. The interested reader is encouraged to consult the original work of Stone and Yao (1990) for full details. Their parameterization is given by

$$\overline{w'\theta'} = -0.6\frac{g^2 d^2}{\langle\theta\rangle^2\langle N\rangle f^2}\left(\frac{\partial\overline{\theta}}{\partial y}\right)^2\left|\frac{\partial\overline{\theta}}{\partial y}\right|\chi(r)\left[\frac{z}{D}-\frac{z^2}{4D^2}\right]e^{-z/D}$$

where

$$r = \frac{\langle\Gamma_m\rangle}{\langle\Gamma_d\rangle}\frac{\left\langle\frac{\partial\overline{T}}{\partial z}\right\rangle+\langle\Gamma_m\rangle}{\left\langle\frac{\partial\overline{T}}{\partial z}\right\rangle+\langle\Gamma_d\rangle} \qquad (2.44)$$

$$\chi = \frac{1+r/\lambda}{1+\lambda} \quad \textit{and} \quad \lambda = \frac{.573\sqrt{r}}{1-0.427r^{0.302}}$$

Although the original form of the first law of thermodynamics (2.10) requires a knowledge of $\overline{w'T'}$, we actually use an equation similar to 2.26

$$\frac{\partial T}{\partial t} = -w\left[\frac{\partial T}{\partial z}+\frac{g}{C_p}\right]+\frac{1.4RT}{\rho C_p}\frac{\partial\rho}{\partial t} \qquad (2.26)$$

for the heating due to the vertical turbulent transport of heat given by

$$\frac{\partial T}{\partial t} = -\left[\frac{\partial\overline{w'\theta'}}{\partial z}\right]$$

Note that the $\left[\frac{\partial T}{\partial z}+\frac{g}{C_p}\right]$ term of 2.26 is equivalent to $\frac{\partial\theta}{\partial z}$.

2.4.4 Latent Heating

Although we have not as yet included a prognostic equation for specific humidity we have attempted to estimate some of the fluxes of latent heat energy with moisture flux calculations using our prescribed humidity profiles. Our idea is that whenever water vapor is added to (or removed from) the air of a layer by advection it is immediately removed from (or added to) the air of the layer by condensation (or evaporation). This will keep the absolute humidity of the layer fixed during moisture transport but will result in latent heating or cooling of the layer. This model thus assumes a quasi-steady state moisture budget. The horizontal flux of latent heat Q_L is calculated according to,

$$Q_L = \rho L_v\left(\overline{s}\,\overline{v}+\overline{s'v'}\right)$$

while the vertical flux of latent heat is,

$$Q_L = \rho L_v \left(\overline{sw} + \overline{s' w'} \right)$$

The advection terms are relatively easy to calculate from the dynamics and hydrostatic adjustment portions of the model. We assume the vertical eddy heat flux is taken care of by our convective scheme which is described below. We estimate the horizontal eddy flux of latent energy following the method of Stone and Yao (1990). That is

$$\overline{s' v'} = r \left(\frac{p}{p_o} \right)^{\kappa} \frac{\partial s^*}{\partial T} \left(\overline{v' \theta'} \right)$$

where s is the specific humidity, r the relative humidity, p pressure, p_o=1 atm, and $\left(\overline{v' \theta'} \right)$ is calculated as in equation (2.42).

2.4.5 Convection

To simulate convection we divide up each hemisphere into two regions, the tropics and the mid to high latitudes. We use a lapse rate adjustment scheme, similar to that presented by Stone and Carlson (1979) which is based on the baroclinic stability relation. Below we briefly describe this convective parameterization.

At the surface the fluxes of latent and sensible heat are given by

$$\begin{aligned} LH &= \rho L_v C_D V_s \left[s^*(T^*) - s(T(0)) \right] \\ SH &= \rho C_p C_D V_s \left[T^* - T(0) \right] \end{aligned} \tag{2.45}$$

where $C_D V_S$ is the surface drag coefficient times the surface velocity V_S, T* is the surface temperature, T(0) is the air temperature at the surface, s is the specific humidity and s* is the specific humidity at saturation. These fluxes are used to calculate the heating (or cooling) of the atmospheric boundary layer and surface.

We follow the suggestion of Stone and Carlson (1979) and calculate the critical lapse rate, γ_c, for convective adjustment by requiring that the lapse rate of the atmosphere be less than both the moist adiabatic lapse rate and the critical lapse rate predicted from the baroclinic stability theory of a two level model. The moist adiabatic lapse rate is given by

$$\gamma_m = \gamma_d \frac{1+\frac{\varepsilon L_v e_s}{PRT}}{1+\left(\frac{\varepsilon L_v}{C_p P}\right)\frac{\varepsilon L_v e_s}{RT^2}} \qquad 2.46$$

where e_S is the saturation vapor pressure obtained from the Clausius-Clapeyron relation

$$e_s = (6.11mb)\exp\left[\frac{\varepsilon L_v}{R}\left(\frac{1}{T_o}-\frac{1}{T}\right)\right]$$

In the above T_O=273K, ε=.622, and γ_d=9.8 K/km is the dry adiabatic lapse rate. Stone and Carlson (1979) note that the baroclinic stability of a two layer model predicts that the lapse rate will not exceed a critical value given by,

$$\gamma_c = \gamma_d + \frac{\tan\phi}{H_2}\frac{\partial T_2}{\partial \phi} \qquad 2.47$$

where H_2 and T_2 are the scale height and temperature in the middle atmosphere (approximately 600 mb) and ϕ is the latitude. For each latitude we calculate the moist adiabatic lapse rate and the critical lapse rate from baroclinic theory using (2.46) and (2.47). We then use the convective adjustment scheme of Manabe and Wetherald (1967) to force the atmospheric lapse rate to be less than the smaller of the two calculated lapse rates.

2.5 Modifications of the Radiative Convective Model.

As mentioned previously, at the heart of our two-dimensional model is the one dimensional radiative convective model (1DRCM) described in detail by MacKay and Khalil (1991) and MacKay (1990). We have modified several aspects of the original 1D-RCM to make it more physically realistic, more amenable to the two dimensional model's structure, and more computationally efficient. Below we give a brief overview of the original 1-D RCM along with a description of the modifications made to it for use in the two dimensional model described in this work.

2.5.1 1-D RCM review

The 1D RCM of MacKay and Khalil (1991) has eighteen layers and calculates the absorption of solar radiation due to H_2O, O_3, O_2, and CO_2 for both clear and cloudy conditions following the methods described by Lacis and Hansen (1974) for H_2O and O_3; and by Sasamori (1972) for O_2 and CO_2. Infrared fluxes for H_2O, CO_2, O_3, N_2O, CH_4, F-11, and F-12 are calculated using parameterizations previously published by: Ramanathan (1976), Roberts et al. (1976), and Kuo (1977) for H_2O; Kiehl and Ramanathan (1983) for CO_2; Kuo (1977) for O_3; Donner and Ramanathan (1980) for N_2O, Ramanathan et al. (1985) for CH_4; and Ramanathan et al. (1985) and Ramanathan (1976) for F-11 and F-12. Convection is simulated by performing a convective adjustment similar to that used by Manabe and Wetherald (1967) using either a constant critical lapse rate or the moist adiabatic lapse rate. A single cloud is included in one of the atmospheric layers to simulate the average cloudiness of the atmosphere. This cloud occupies one complete layer in the vertical direction and approximately 50% of the layer in the horizontal direction. The cloud is assumed to be a black body to terrestrial radiation and its altitude is typically chosen to be between roughly 3 to 4 km. This type of cloud is similar to that described by Stephens (1984) for 1-D models. The surface heat capacity is an adjustable parameter that has been included to simulate the thermal inertia of the oceans.

2.5.2 Modifications of the Radiation Model

We have reduced the vertical resolution of the 1-D RCM from eighteen layers to nine layers. This change alone increases the speed of the radiation part of the calculations by a factor of four. We have changed the parameterization for the calculation of the infrared absorptivity of CO_2 to that used by Ramanathan et al. (1983) in the NCAR general circulation model. Since the parameterization used in the original model was very computationally intensive, this change also reduced the total computational time for radiative flux calculations by about 30%. The net affect of these changes on the sensitivity of the surface temperature to a doubling of CO_2 was to increase it by about

10%. This corresponds to a change in surface temperature of about 2.2 K for a doubling of CO_2 in the modified 9 layer model as opposed to an approximate 2.0 K change in the original 18 layer model, assuming a constant 6.5 K/km lapse rate.

The concentration profiles for carbon dioxide, methane, nitrous oxide, F-11, and F-12 used in our 2-D model are assumed to be independent of latitude and season. The vertical dependence of the concentration of these gases is as described by MacKay and Khalil (1991). We use the ozone concentration profiles generated by the Lawerence Livermore National Laboratory two dimensional ozone photochemical model obtained from Dr. Doug Kinnison (personal communication). We have interpolated his data for each month to fit our model grid and assume that the ozone concentration profile during a given month remains fixed. The Data we use are included in Appendix B along with graphs of the January and July profiles.

Two cloud layers were added for a total of three cloud layers (high, middle, and low) in the 9 layer model. The clouds also have an adjustable emissivity instead of the eighteen layer model assumption of a black body cloud. We use a cloud emissivity of 1.0 for low and middle clouds, and 1.0 for high tropical clouds and 0.5 for high clouds that are outside of the tropical latitudes ($|\phi| > 30°$). We follow the treatment described in Stephens (1984) for the calculation of radiative fluxes in regions of multiple cloud. In regions of the sky that have both cloud and gas infrared emissivities (ε_c and ε_g)the total emissivity ε is given by

$$\varepsilon \approx 1-(1-\varepsilon_g)(1-\varepsilon_c) \qquad (2.48)$$

In regions of the sky that have several layers of cloud above, we assume that the cloud layers overlap randomly and that the total cloud cover can be written as

$$A_T=1-(1-A_1)(1-A_2).....$$

where A_i represents the amount of the sky covered by the ith cloud. In future versions of the model we will predict cloud cover hydrodynamically and increase the sophistication of the treatment of cloud radiation physics.

2.6 Deep Ocean Model

Our two-dimensional atmospheric model can be coupled to an energy balance deep ocean model of the type described by Watts and Morantine (1990). In this study we cut off the flow of energy to the deep ocean and hence use only a mixed layer ocean model with horizontal transport of energy. We do plan to eventually connect the model mixed layer to the deep ocean and hence will give a brief description of the model below for future reference. Figure 2.4 is a graphical description of the ocean model structure. The ocean mixed layer has a depth D and a horizontal turbulent diffusion constant K_{hs} that simulates the horizontal turbulent transport of heat. Deep water is assumed to form at some latitude ϕ, sink to the bottom of the ocean and then spread uniformly over the bottom of the ocean into a very thin layer. An upwelling velocity w recirculates this water back to the surface. Below we present only the fundamental equations of the model. A complete description of the derivation of these equations is given by Watts and Morantine (1990).

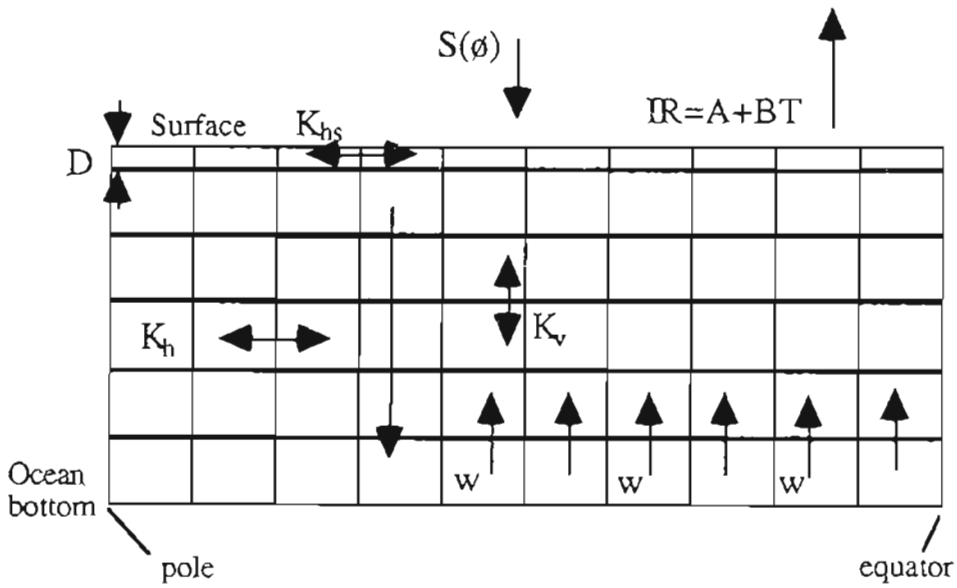

Figure 2.4. The Deep ocean Model used in the GCRC 2D climate model. After Watts and Morantine (1990)

The surface of the Watts and Morantine model has no atmospheric model to couple to so they have to crudely parameterize the net flux of solar radiation as a function of latitude, S(Ø), and the net flux of infrared radiation leaving the surface by A+BT. Our detailed radiation code allows us to explicitly calculate these fluxes.

The respective energy balance equations for the surface and interior,

$$\frac{\partial T}{\partial t} = -\frac{w}{D}\frac{\partial T}{\partial z} - \frac{w}{D}\tan\phi\frac{\partial T}{\partial \phi} - \frac{K_v}{D}\frac{\partial T}{\partial z} + \frac{K_{hS}}{a^2\cos\phi}\frac{\partial}{\partial\phi}\left[\cos\phi\frac{\partial T}{\partial\phi}\right] + \frac{1}{\rho_w DC_w}\left[(1-\alpha)S(\phi) - (A+BT)\right] \tag{2.49}$$

and

$$\frac{\partial T}{\partial t} = -w\frac{\partial T}{\partial z} + K_v\frac{\partial^2 T}{\partial^2 z} + \frac{K_h}{a^2\cos\phi}\frac{\partial}{\partial\phi}\left[\cos\phi\frac{\partial T}{\partial\phi}\right] \tag{2.50}$$

In the above a is the Earth's radius, D is the depth of the mixed layer (50-100 m). In their original model Watts and Moratine used w=4.0 m/yr for the upwelling velocity which was assumed to be independent of latitude up to 60 degrees from the equator (where the deep water is assumed to form) and zero for latitudes above that region, Kv (=2000 m^2/yr) for the vertical diffusivity of heat, K_{hS} for the horizontal diffusivity of heat at the surface (=1.2×10^{12} m^2/yr), and K_h for the diffusivity of heat in the internal ocean (=2×10^{10} m^2/yr). The variables T, ϕ, and z are the temperature, latitude, and vertical depth respectively.

In the present work the above deep ocean model is used as a simple mixed layer model since we set both K_v and w equal to zero. We have used this full deep ocean model in a simple energy balance model as described by Watts and Morantine (1990) and found, as they suggested, that the mean surface temperature was highly sensitive to changes in upwelling velocity. Although their original study was of the Younger Dryas reversal, we believe that this sensitivity of mean surface temperature to upwelling velocity w is also very important in terms of present and future climate change scenarios over decades to centuries. Preliminary investigations with the model of Watts and Moratine suggest that if w decreases as the equator to pole temperature gradient decreases

(due to increases in greenhouse gases) then this would constitute a possible negative feedback mechanism. A decrease in w for a decrease in equator to pole temperature gradient seems intuitively plausible if we believe that the driving force of ocean circulations is the atmosphere. More work needs to be done however, before we can incorporate this full deep ocean model reliably to our climate model and pursue this idea.

2.7 Sea Ice

We have performed some preliminary experiments with the development of a prognostic thermodynamic sea ice model but have found that it adds too many additional processes for us to clearly understand the basics of our 2D atmosphere model. That is, the much longer response time of the integrated sea ice model and atmosphere model and the additional feedbacks associated with a variable sea ice extent seemed to overshadow the fundamental aspects of the atmosphere model. Thus in this first presentation of our 2D model, we have prescribed the seasonal sea ice area to agree with the observations reported by Peixoto and Oort (1992), page 213. Following the work of Parkinson and Washington (1980) we have used a mean sea ice thickness of 3.0 meters for the Arctic and 1.5 meters for the Antarctic regions.

2.8 Summary of the Model.

The primitive equations (2.7-2.12) in two-dimensional zonally averaged form to be solved are,

the equation of continuity,

$$\frac{\partial \rho}{\partial t} = -\frac{1}{a\cos\phi}\frac{\partial(\rho v\cos\phi)}{\partial\phi} - \frac{\partial \rho w}{\partial z} \tag{2.7}$$

the horizontal equations of motion,

$$\frac{\partial \rho u}{\partial t} = -\frac{1}{a\cos\phi}\frac{\partial[\rho(uv+\overline{u'v'})\cos\phi]}{\partial\phi} - \frac{\partial[\rho(uw+\overline{u'w'})]}{\partial z} + \rho f v + \rho\frac{\tan\phi}{a}(uv+\overline{u'v'}) + \rho F_\lambda \tag{2.8}$$

$$\frac{\partial \rho v}{\partial t} = -\frac{1}{a}\frac{\partial p}{\partial\phi} - \frac{1}{a\cos\phi}\frac{\partial[\rho v^2\cos\phi]}{\partial\phi} - \frac{\partial[\rho(vw+\overline{v'w'})]}{\partial z} - \rho f u - \rho\frac{u^2\tan\phi}{a} + \rho F_\phi \tag{2.9}$$

the first law of thermodynamics

$$\frac{\partial[\rho C_p T]}{\partial t} = -\frac{1}{a\cos\phi}\frac{\partial[\rho C_p\cos\phi(vT+\overline{v'T'})]}{\partial\phi} + \rho(Q+Q_L) + 1.4RT\left[\frac{\partial\rho}{\partial t} + \frac{v}{a}\frac{\partial\rho}{\partial\phi}\right] + 1.4RTw\frac{\partial\rho}{\partial z} - \frac{\partial[\rho C_p(wT+\overline{w'T'})]}{\partial z} \tag{2.10}$$

the equation of state,

$$P = \rho RT \tag{2.11}$$

the assumed condition of hydrostatic equilibrium

$$dP = -\rho g dz \tag{2.12}$$

We solve the above equations as an initial valued problem for the zonally averaged values of temperature, T; density ρ (or pressure, p), vertical, meridional, and zonal velocities (w,v, and u) using a spatial grid of 9 vertical layers and 18 horizontal zones.

Simulations of surface fluxes of heat, momentum and moisture are included as described in sections 2.1.5, 2.3.3, and 2.4.5. For reference, we outline the parameterization of the turbulent fluxes that appear in the above equations below

Equation 2.8 ($\overline{u'w'}$ and $\overline{u'v'}$)

$\overline{u'w'}$

$$\frac{\partial \rho \overline{u'w'}}{\partial z} = \frac{\partial}{\partial z} \rho K_z \frac{\partial u}{\partial z} \tag{2.16}$$

(see section 2.3.3)

$\overline{u'v'}$

$$K_{vy} = K_{BC} + K_{NL} \tag{2.38}$$

$$K_{NL} = \begin{cases} K_o, [\bar{u}] > 0 \\ \\ K_o e^{y/L}, [\bar{u}] < 0 \end{cases} \tag{2.39}$$

$$\frac{\partial}{\partial y} \int_0^{P_s} \overline{u'v'} dp = \int_0^{P_s} \left\{ K_{vy} \left(\beta - \frac{\partial^2 u}{\partial y^2} \right) - \frac{(1+M)f}{\sigma_e} \frac{\partial \theta}{\partial y} \frac{\partial K_{vy}}{\partial p} \right\} dp \tag{2.40}$$

(see section 2.4.1)

Equation 2.9 ($\overline{v'w'}$)

$\overline{v'w'}$

$$\frac{\partial \rho \overline{v'w'}}{\partial z} = \frac{\partial}{\partial z} \rho K_z \frac{\partial v}{\partial z} \tag{2.17}$$

(see section 2.3.4)

Equation 2.10 ($\overline{v'T'}$, $\overline{w'T'}$, and Q_L)

$\overline{v'T'}$

$$\overline{v'\theta'} = 0.6\frac{gd^2\langle N\rangle}{\langle\bar{\theta}\rangle f^2}\left(\left\langle\frac{\partial\bar{\theta}}{\partial y}\right\rangle\right)^2 \exp\left\{-\frac{z}{\langle H\rangle}(1.48\gamma + 0.48)\right\} \times\left[1-\exp\left\{-\frac{z}{.45km}\right\}\right] \tag{2.42}$$

$$\overline{v'\theta'}_2 = -C_T\left(\frac{\partial\bar{\theta}}{\partial y}\right)\left|\frac{\partial\bar{\theta}}{\partial y}\right|\left[1-\exp\left\{-\frac{z}{.45km}\right\}\right] \tag{2.43}$$

$$\overline{v'T'} = (P/P_s)^{\kappa}\left\{\overline{v'\theta'} + \overline{v'\theta'}_2\right\}$$

(see section 2.4.2)

$\overline{w'T'}$

$$\overline{w'\theta'} = -0.6\frac{g^2d^2}{\langle\theta\rangle^2\langle N\rangle f^2}\left(\frac{\partial\bar{\theta}}{\partial y}\right)^2\left|\frac{\partial\bar{\theta}}{\partial y}\right|\chi(r)\left[\frac{z}{D}-\frac{z^2}{4D^2}\right]e^{-z/D}$$

where

$$r = \frac{\langle\Gamma_m\rangle}{\langle\Gamma_d\rangle}\frac{\left\langle\frac{\partial\bar{T}}{\partial z}\right\rangle + \langle\Gamma_m\rangle}{\left\langle\frac{\partial\bar{T}}{\partial z}\right\rangle + \langle\Gamma_d\rangle} \tag{2.44}$$

$$\chi = \frac{1+r/\lambda}{1+\lambda} \quad \textit{and} \quad \lambda = \frac{.573\sqrt{r}}{1-0.427r^{0.302}}$$

(see section 2.4.3) As noted in section 2.4.3 the actual form of the first law of thermodynamics used in the calculation relies on a knowledge of $\overline{w'\theta'}$ instead of $\overline{w'T'}$.

Q_L

The horizontal flux of latent heat Q_L is calculated according to,

$$Q_L = \rho L_v\left(\overline{s}\,\overline{v} + \overline{s'v'}\right)$$

while the vertical flux of latent heat is,

$Q_L = \rho L_v\left(\overline{\bar{s}\bar{w}} + \overline{s'w'}\right)$ with $\left(\overline{v'\theta'}\right)$ given by 2.42. Latent heat is also transfered during the convective adjustment and we assume that this process simulates $\overline{s'w'}$.

In the next chapter we explore the actual performance of the GCRC 2-D model. We pay particular attention to the simulation of zonally averaged temperature and zonal velocity for summer (JJA), winter (DJF), and annual mean conditions.

CHAPTER 3

MODEL PERFORMANCE

3.1 Model Overview

3.1.1 Introduction

In this chapter we present and discuss the performance of the GCRC 2D zonally averaged statistical dynamical model, beginning with a general overview of its performance followed by a presentation of the outputs for the model's control climate for summer (JJA), Winter (DJF), and annual averages compared to observations. Our model does very well at simulating the observed state of Earth's atmosphere, however below we will not only focus on the model's strengths but will also pay particular attention to its weaknesses. We hope that this will identify areas that require improvements and identify areas of future work for other 2D climate modelers.

The order of our presentation of model outputs will be: section 2, zonal mean temperatures and energy balance; section 3, zonal mean velocities u and angular momentum budget; section 4, model kinetic energy and mass stream function; section 5 zonal mean precipitation and surface pressure, and section 6 a summary of the important results of this chapter.

3.1.2 General Discussion

Our 2D model runs on an IBM RISC 6000 work station taking approximately 2 CPU hours for each simulated year. Stability of the model is an important consideration. As described in the previous chapter, if the initial conditions (initial state of the model's atmosphere) are too far away from the steady state solution, large amplitude oscillations can grow without bound destroying the model output. After getting into a stable run with

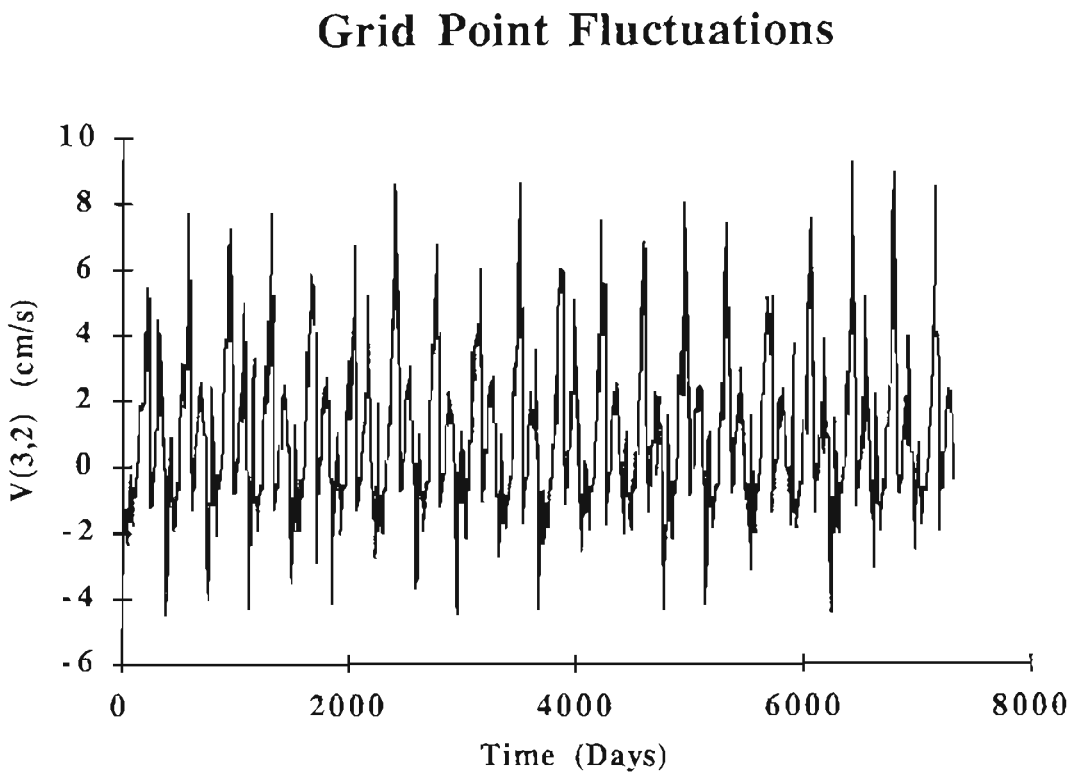

Figure 3.1. Example of grid point fluctuations.

a particular initial condition, starting as described in chapter 2, care must be taken not to introduce extraordinarily large perturbations into the initial state. Our technique has been to use the output from one run as the initial state of the next run (with a perturbation) to explore the phase space of the model. As shown in the next chapter, the model is stable to perturbation studies such as doubling of CO_2 and indeed works well with much larger perturbations of the initial state. For example, the model runs stablely when the pressure of the whole system (each grid point) is altered by 4.0% but would most likely "blow up" if the pressure of an isolated grid point were altered by 4.0%.

Figure 3.1 shows the output of the meridional velocity in cm/s at grid point 3,2 (approx. 200 mb, 75°N) for a 20 year run. For this particular grid point and output

variable, the model appears to have a distinct stochastic signal superimposed upon a strong periodicity of 1 year. In Figure 3.2 the global averaged surface air temperature is displayed for the same 20 year run. Notice that it is highly periodic with essentially no stochastic signal. One point that can be gleaned from these figures is that the amount of noise in the model output depends on either the scale size of interest or the output variable. We experimented with this and found that the 3,2 grid point temperature signal and zonal velocity were both fairly noise free so it seems that noise in figure 3.1 is primarily due to the fact that we are looking at the meridional velocity v. An inspection of Figure 3.1 also reveals why the model may become unstable, especially for perturbations of a single grid point. The strong oscillations that exist in the meridional velocity are undoubtedly the first to become unstable. Since the meridional velocity at the gridpoint level is not a deterministic output variable, it is useful to keep track of it as a diagnostic check to see if the model was set-up correctly for a stable run. For example, if we were getting ready to make a long 24 to 48 hour run, we could first run the model for 2 hours (1 year of simulated time) and look at the grid point meridional velocity time series. If it appeared to have a steadily increasing amplitude, we would suspect that the model was on the way to uncontrollable oscillations and hence make adjustments accordingly. Increasing the magnitude of the variable C_T of equation 2.40 was one way to increase the stability of the model, since increasing it tends to decrease the temperature gradients everywhere.

All of this talk about model stability may make the reader wonder if it is even possible to get a stable solution out of the model. Actually in its present state the model is fairly stable to substantial perturbations and could be made more stable with an increase in the filtering of the model equations, beyond the present use of the eighth order Shapiro filter, by adding a damping constant for changes in meridional velocity. Ideally the model must be stable enough for the perturbation studies of interest, without being too stable by over filtering. Too much filtering of the model equations could cause the model to be biased towards the initial state as a steady state solution, which would be very bad for perturbation studies. We have attempted to achieve the appropriate balance between model stability and freedom of the model physics to completely determine the output of the model. More work could be done by us in quantifying the above ideas, but at present

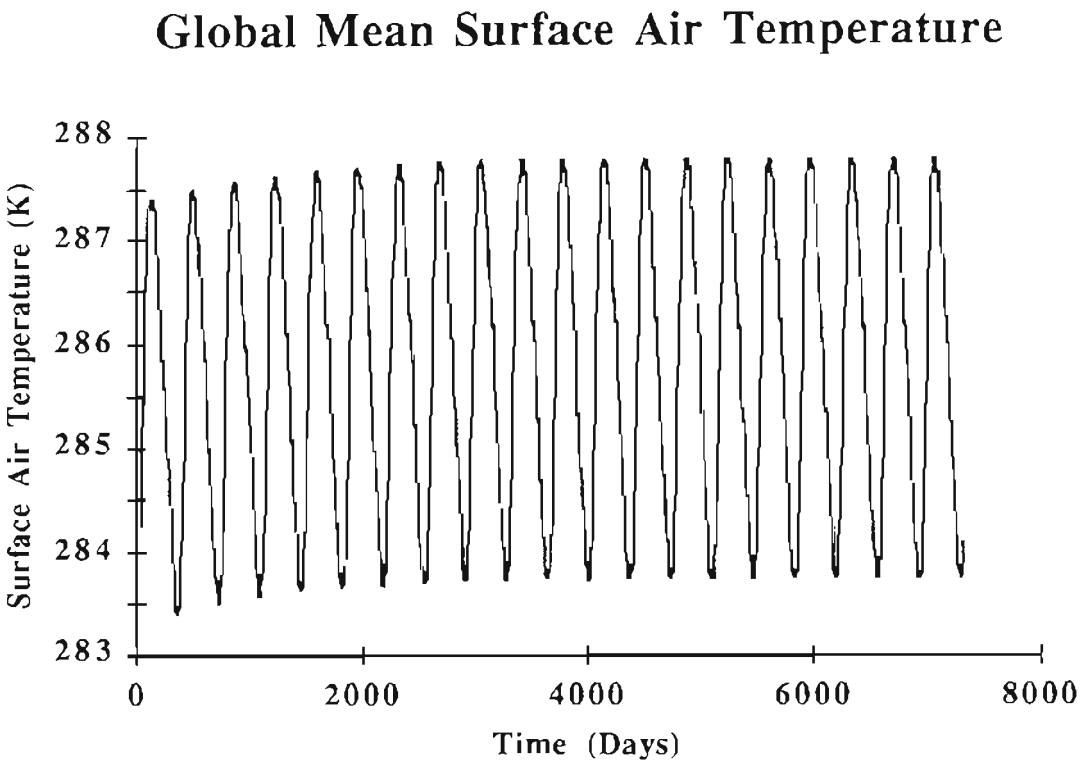

Figure 3.2. Surface air temperature annual cycle and approach to equilibrium.

we feel that we have achieved a good balance. The next sections dealing with the presentation of model output data lend support to this claim.

3.2 The Thermal Structure of the Atmosphere and Energy Balance

3.2.1 Temperature Field

Several interesting features of the model can be extracted from a careful inspection of Figure 3.2. First, it appears that a run length of 20 years is more than

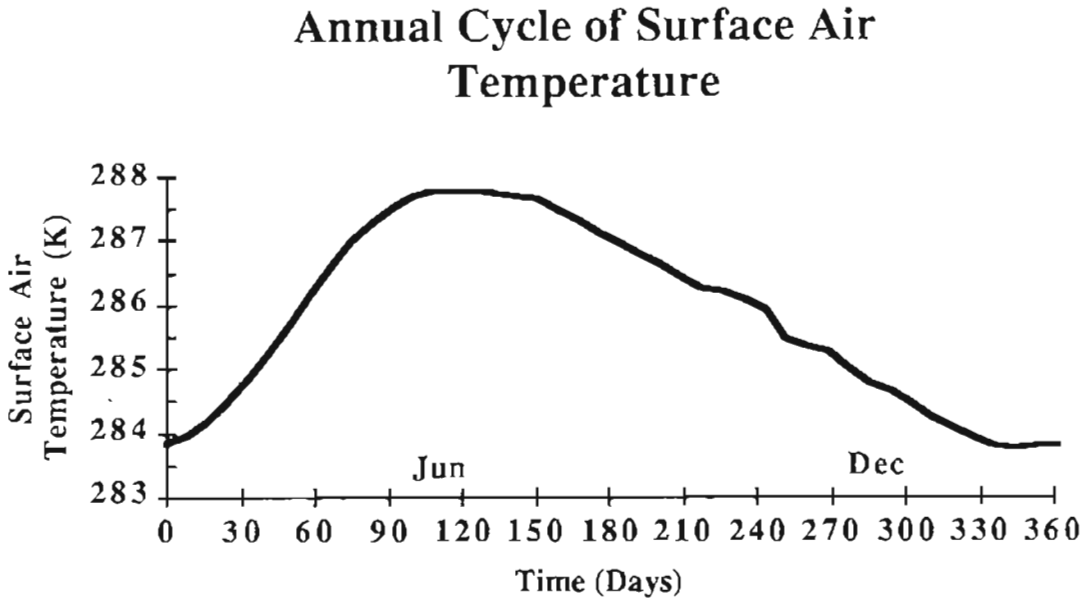

Figure 3.3. Annual cycle of surface air temperature. Note that March first corresponds to time=0.0.

adequate for the model's surface air temperature to reach equilibrium and that even 10 years (3650 days) would be sufficient. Through experimentation we have found that fairly small differences exist in the overall temperature structure, or dynamical structure of the atmosphere between year twenty and year two of the model simulation even though the perturbation from the initial state may have been substantial. The differences that do exist between years two and twenty are mostly concentrated in the polar regions. Thus for quick model sensitivity studies, aimed at investigating the effects of changes in model parameterizations or physics, two year simulations are efficient and seem to work well. However for perturbation studies involving changes in the physical state of the atmosphere such as a doubling of CO_2, where the greatest precision possible is desired, then we have concluded that a twelve year simulation is optimum from the view point of efficiency and precision. This twelve year simulation takes 24 CPU hours to run. It should be stressed here that as other physical processes are introduced, such as a

thermodynamic sea ice model, the time to reach a steady state solution will undoubtedly increase and hence should be reassessed.

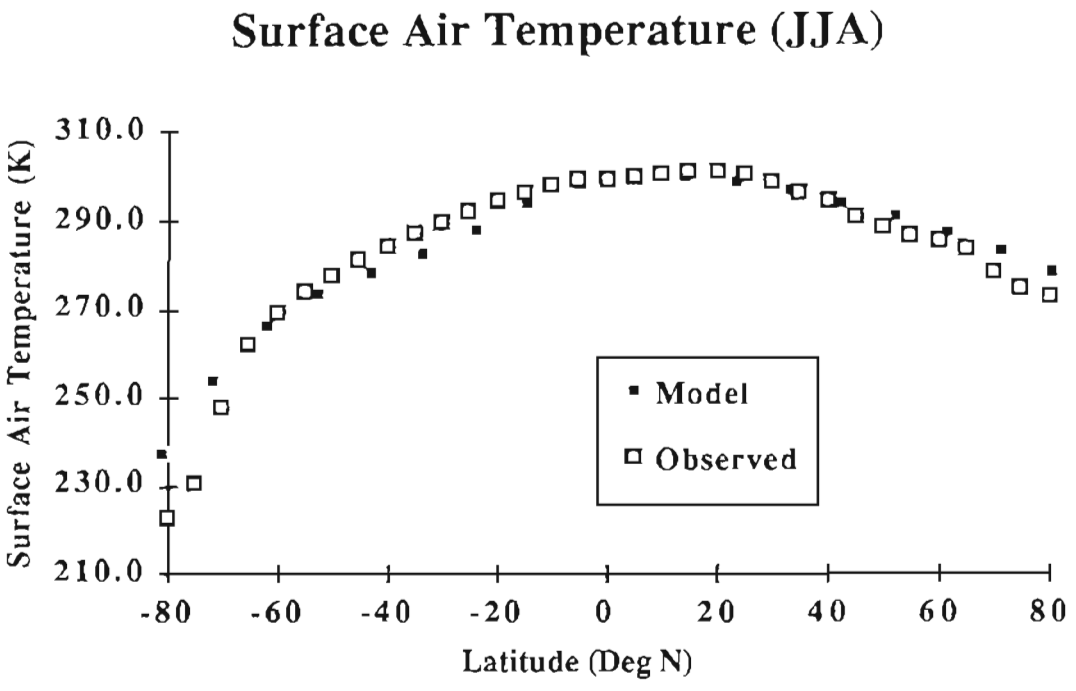

Figure 3.4. Zonal-mean summer (JJA) surface air temperature. Observations are from NCAR Data Support Section, Scientific Computing Division (Roy Jenne contact); Units are Kelvin. Model values are for z=100 m.

Also from Figure 3.2 it can be seen that the model predicts a strong 1.0 year periodicity in global surface air temperature with an amplitude of approximately 2.0 K. In Figure 3.3 we show the last year of the 20 year control run from Figure 3.2 for a better look at the yearly cycle of model temperature. Willmott and Legates (1993) present an updated surface air temperature climatology using data from the period 1920 to 1980. They calculate global averages of 285.7 K and 289.1 K for January and July respectively giving a range of 3.4 K between the two months. In comparison, our control run January and July surface (z=100 m) air temperatures are 283.9 K and 287.4 K with a range of 3.5

K which are in good agreement with their observations. The yearly cycle is dominated by the asymmetrically larger amount of land mass in northern hemisphere compared to that in the southern hemisphere.

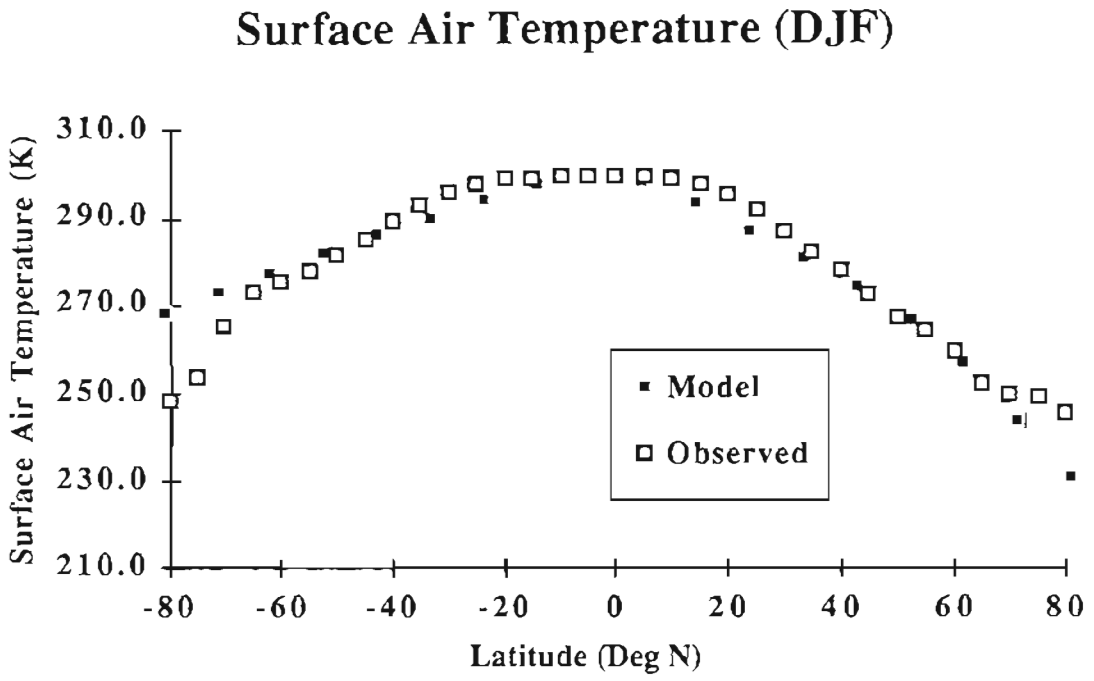

Figure 3.5. Zonal-mean winter (DJF) surface air temperature. Observations are from NCAR Data Support Section, Scientific Computing Division (Roy Jenne contact); Units are Kelvin. Model values are for z=100 m.

In figures 3.4 through 3.6 we compare the latitudinal dependency of the model's zonal mean surface air temperature (for the last year of the 20 year run) with observed values obtained from the NCAR Data Support Section, Scientific Computing Division for summer, winter and annual averages respectively. The model agrees with the observations well for all latitudes except near Antarctica where we have close to a 10 %

discrepancy for the DJF and annual averages. For all three figures the Model's surface air temperature is higher than the observed values over Antarctica. The reason for this is most likely because our model does not at present simulate the orography of the Antarctic continent. Consistent with observations the model does predict slightly cooler Antarctic winters than Arctic winters since no heat energy is allowed to flow from the Southern Ocean onto the continent. Inspection of these three figure lends credibility to our model's ability to simulate the real climate. Uncertainties in actual cloud amounts and cloud optical properties are probably responsible for the majority of the deviation between observed and model output; especially around 30 S latitude.

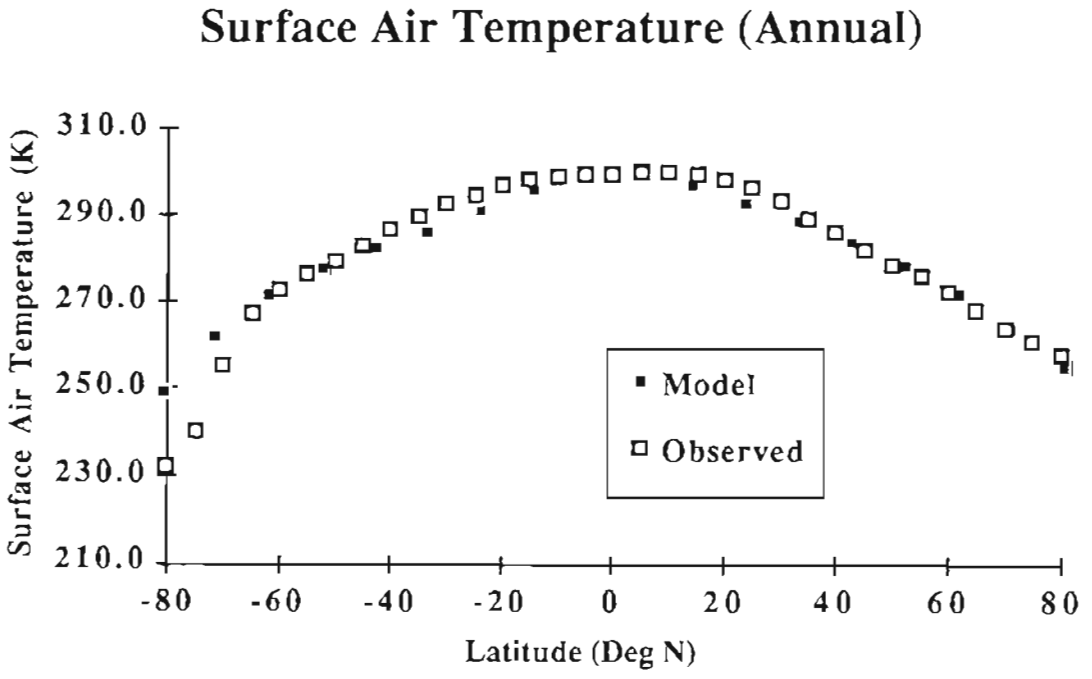

Figure 3.6. Zonal-mean annual average surface air temperature. Observations are from NCAR Data Support Section, Scientific Computing Division (Roy Jenne contact); Units are Kelvin. Model values are for z=100 m.

The vertical and latitudinal profiles of the zonal mean temperatures for summer, winter, and annual averages compared to the observed values obtained from the NCAR Data Support Section, Scientific Computing Division are shown in figures 3.7 through 3.9 respectively. The overall temperature structure predicted by the model agrees well with observations particularly in the troposphere and in the northern hemisphere. This is particularly impressive since there are very few adjustable parameters in the model that can be used to force it to agree with observations.

There are some noteable differences between the observed and modeled temperature structure that were not apparent from the surface temperature comparison of figures 3.4-3.6. First, at present we fail at reproducing the substantial temperature depression in the tropical stratosphere; although we do have a low temperature spot in the DJF tropical stratosphere. One reason for this may be that the boundary condition set on the meridional velocity v in the upper most model layer does not allow enough advective energy to be transported out of the tropical stratosphere. Another possibility for this discrepancy may be that the amount of water vapor assumed to be in the model's tropical stratosphere may be too low, resulting in less radiative cooling there. It is also likely that the vertical resolution of the model is just not adequate above 100 mb to simulate this low temperature region. Further work needs to be done to understand the exact source of this difference between model and observations.

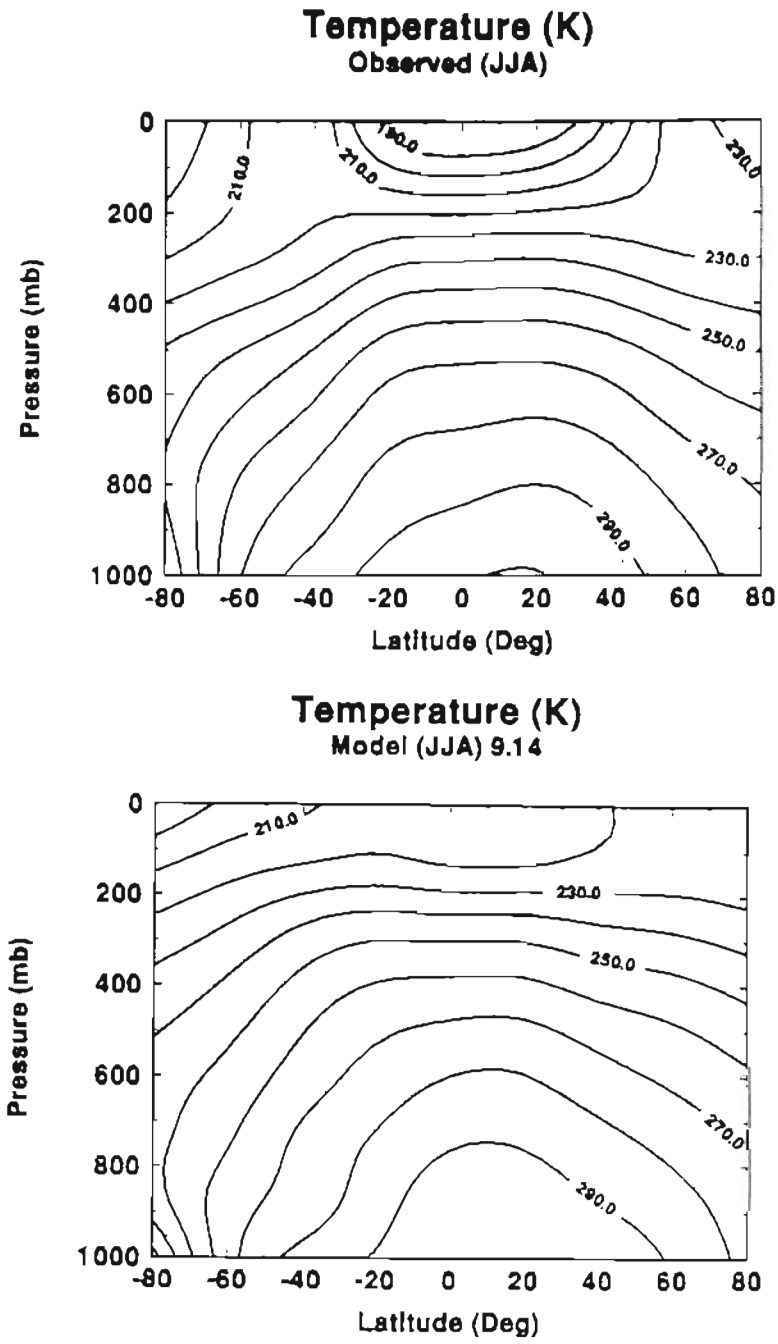

Figure 3.7. Observed zonal-mean (JJA) average air temperature (top) and Modeled (bottom). Observations are from NCAR Data Support Section, Scientific Computing Division; Units are Kelvin.

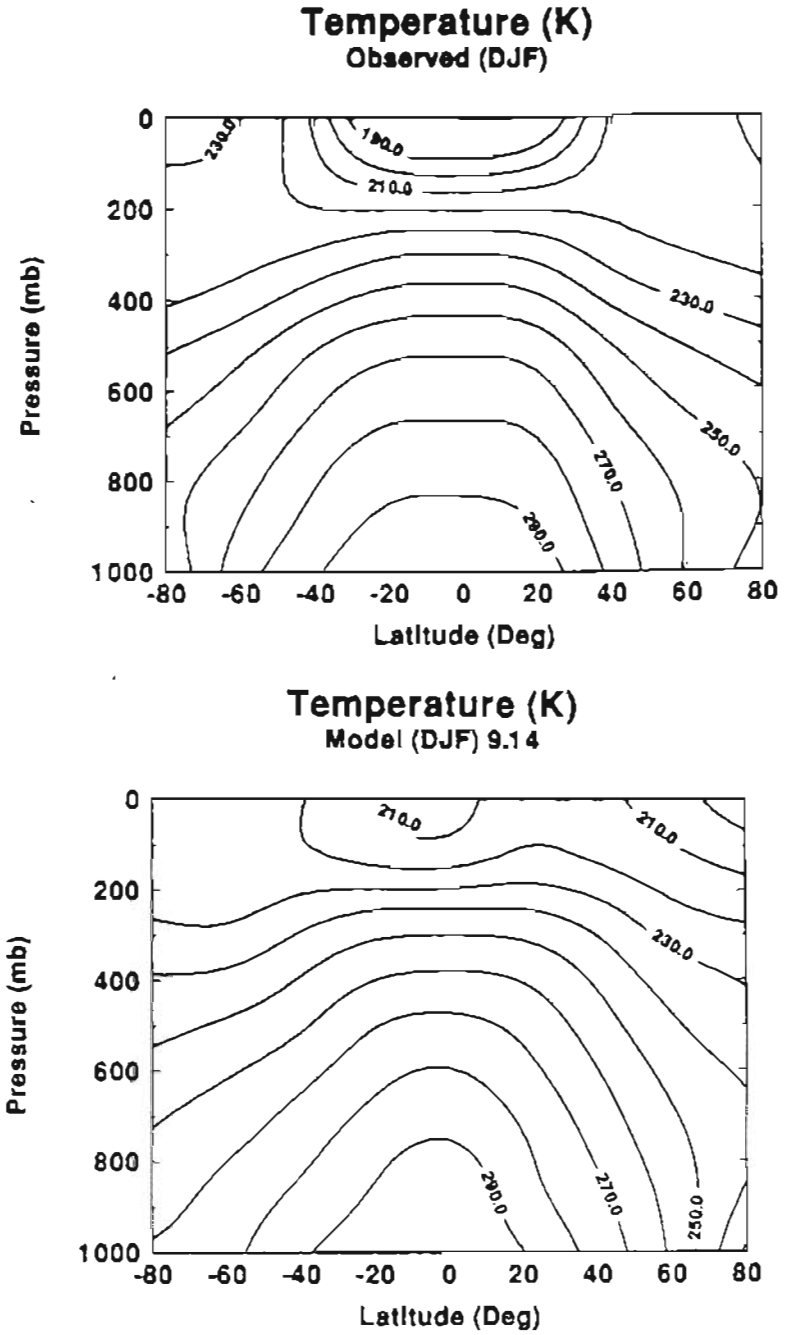

Figure 3.8. Observed zonal-mean (DJF) average air temperature (top) and Modeled (bottom). Observations are from NCAR Data Support Section, Scientific Computing Division; Units are Kelvin.

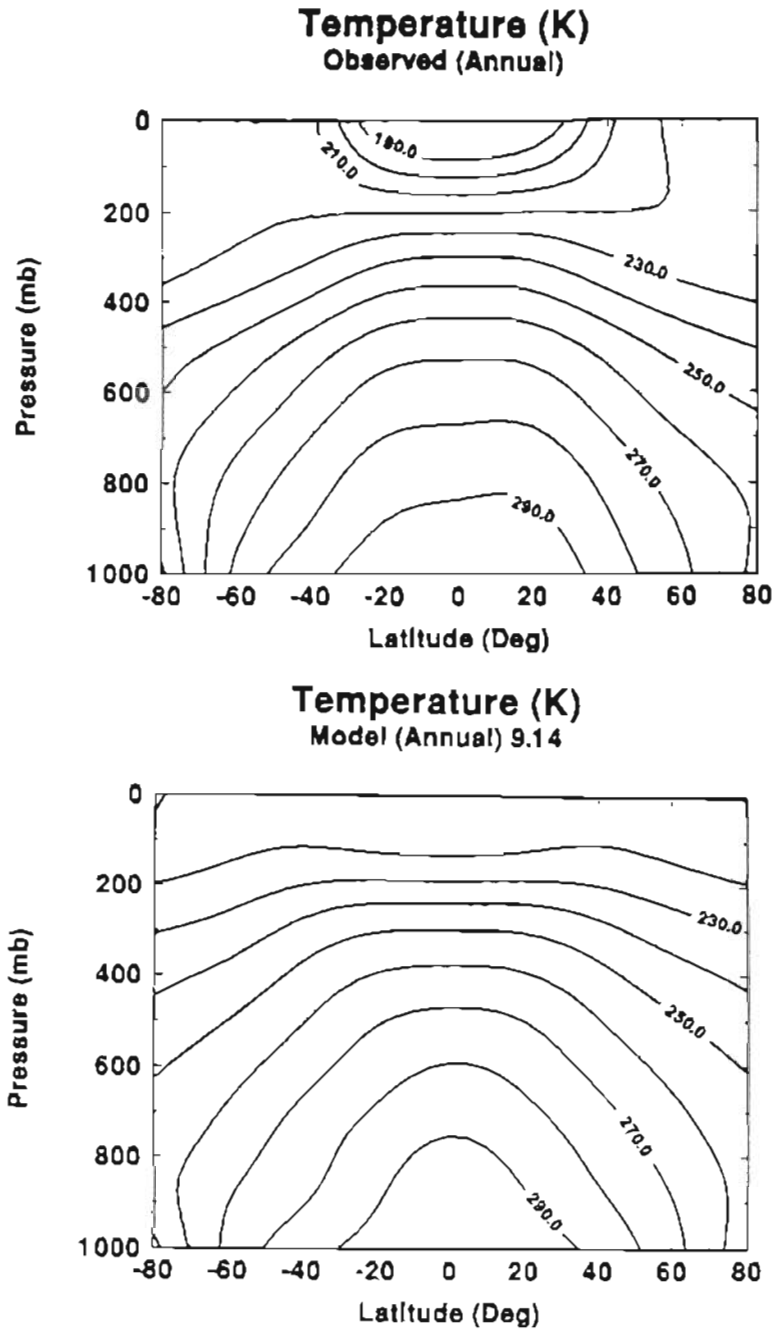

Figure 3.9. Observed zonal-mean annual average air temperature (top) and Modeled (bottom). Observations are from NCAR Data Support Section, Scientific Computing Division; Units are Kelvin.

Another region of difference between observations and the model is in the lower troposphere of the tropics, where the model does not simulate the meridional temperature gradient very well. Note that the meridional temperature gradient 10 to 20 degrees on either side of the equator is relatively constant in the observed atmosphere but our model at present predicts gradients that change significantly in either direction from the equator. This is particularly noticeable in the annual mean temperature field of figure 3.9. The model's meridional gradients do flatten out however in the mid to high tropical troposphere in agreement with observations. Also the model does fairly well at reproducing the seasonal cycle of the overall temperature structure. Reasons for the differences between the model and observations may be due to uncertainties in cloud amount and physics, unrealistic parameterization of tropical convection with associated latent heat releases, or problems with the parameterization of polar heat fluxes. All of the above areas need work for future model improvements.

In figure 3.10 we compare the model's parameterization of the vertical-mean zonal transient eddy sensible heat flux to observed values. The parameterization scheme is described in section 2.4.2 in which we used the Stone and Yao parameterization coupled with an additional term given in equation 2.43. The term C_T in equation 2.43 is an adjustable parameter that we have introduced to keep polar temperatures from becoming too low in winter, resulting in very large zonal winds as a consequence of the thermal wind relation,

$$R\frac{p^{\kappa-1}}{p_o^{\kappa}}\frac{\partial\overline{\theta}}{\partial y} = f\frac{\partial\overline{u}}{\partial p}K_{BC} = 0 \qquad if \quad |\phi| > \phi^*$$

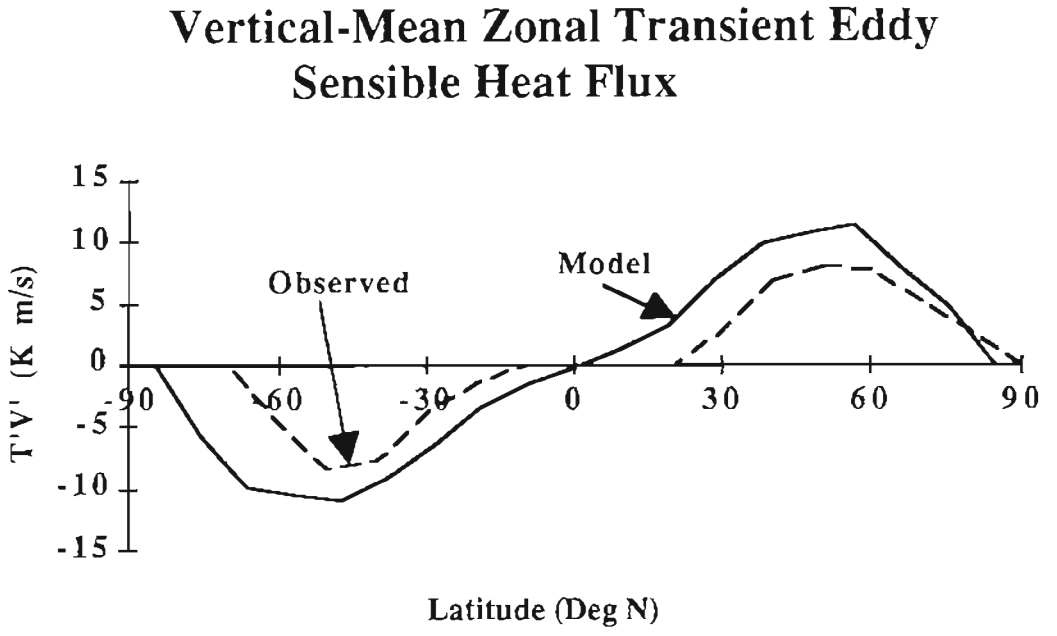

Figure 3.10. Model versus Observed flux of sensible heat due to transient eddies. O Observations were digitized from Peixoto and Oort (1992) page 328 and units are Km/s.

As can be seen by figure 3.10, the meridional transport of eddy sensible heat predicted by our parameterization is high at almost all latitudes. We found however that the parameterization of Stone and Yao was inadequate for our model and gave results that disagreed with observations. The weakness of their parameterization scheme is that it does not reproduce the secondary maximum that is observed near the tropopause in both hemispheres of the real atmosphere. Stone and Yao (1990) point out this fact but were able to use the parameterization without any apparent problem in their 2D model simulations. This may be due to the fact that their were primarily designed to understand the parameterization schemes and were performed for perpetual January conditions and fixed sea surface temperatures.

For our control run, we used a value of C_T=0.008 $m^3/(K^3sec)$ and when we reduced C_T to 0.004 $m^3/(K^3$-hr) Arctic polar temperatures dropped by about 3 K, the global kinetic energy increased by about 20 %, and the strength of the subtropical jets increased by roughly 10 %. Figure 3.11 summarizes some of the essential features of the model's sensitivity to a change in the parameter C_T. The change in temperature resulting from a decrease in C_T from 0.004 $m^3/(K^3$-hr) to 0.008 $m^3/(K^3$-hr) is consistent with intuition, the tropics become warmer and the poles become cooler as the gradient driven transport of energy is decreased. The increase in the strength of the subtrocial jets is also consistent with the thermal wind relation given above. It is also worth noting that as C_T decreases the strength of the Hadley circulation increases by up to 20 % and the magnitude of the parameterized u'v' flux increases by up to 20 %. Also, when we attempted to run the model from the same initial conditons that were used for the control run (C_T=0.008 $m^3/(K^3sec)$) using a value of C_T=0.000, the polar temperatures (particularly in the stratosphere) reached very low values and the model became numerically unstable after 4 years.

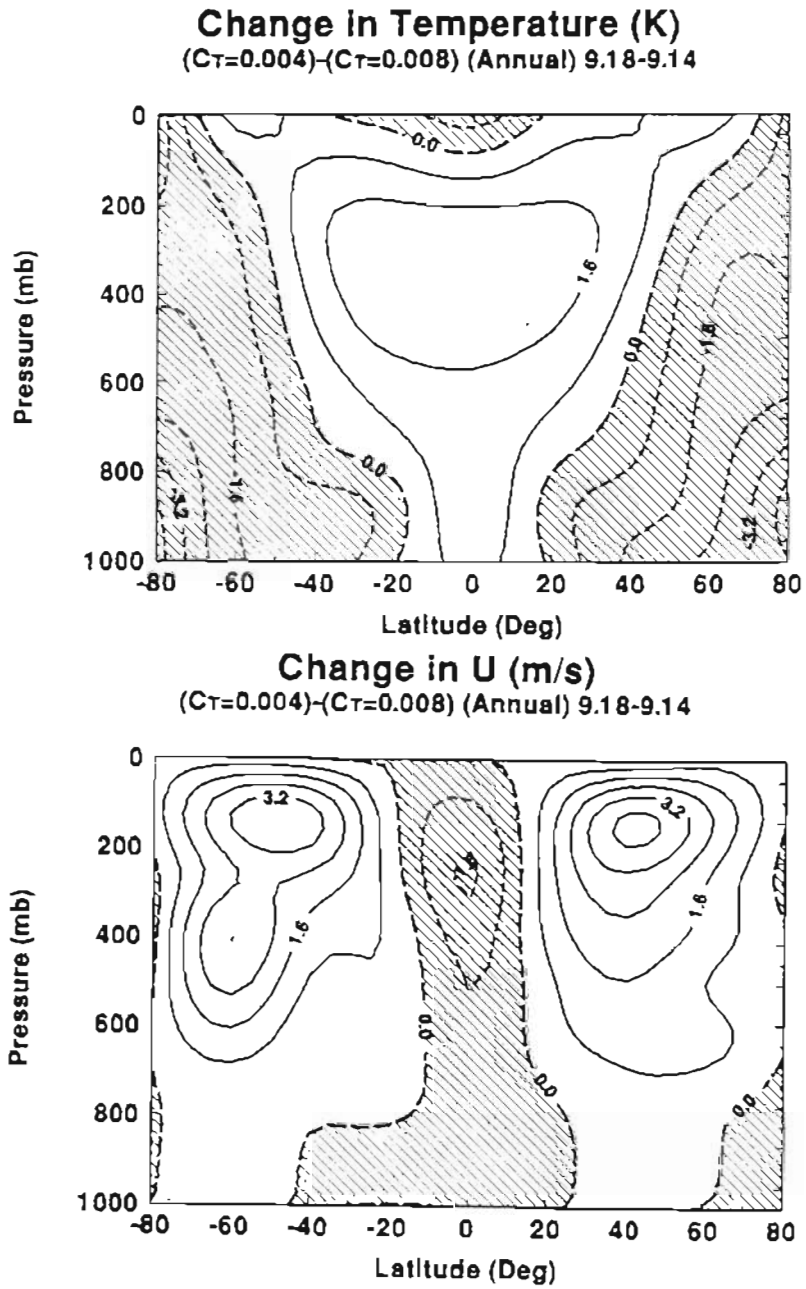

Figure 3.11 The change in mean annual temperature (top) and zonal velocity (bottom) as the turbulent sensible heat diffusivity parameter C_T changes from 0.008 $m^3/(K^3 sec)$ to 0.004 $m^3/(K^3 sec)$.

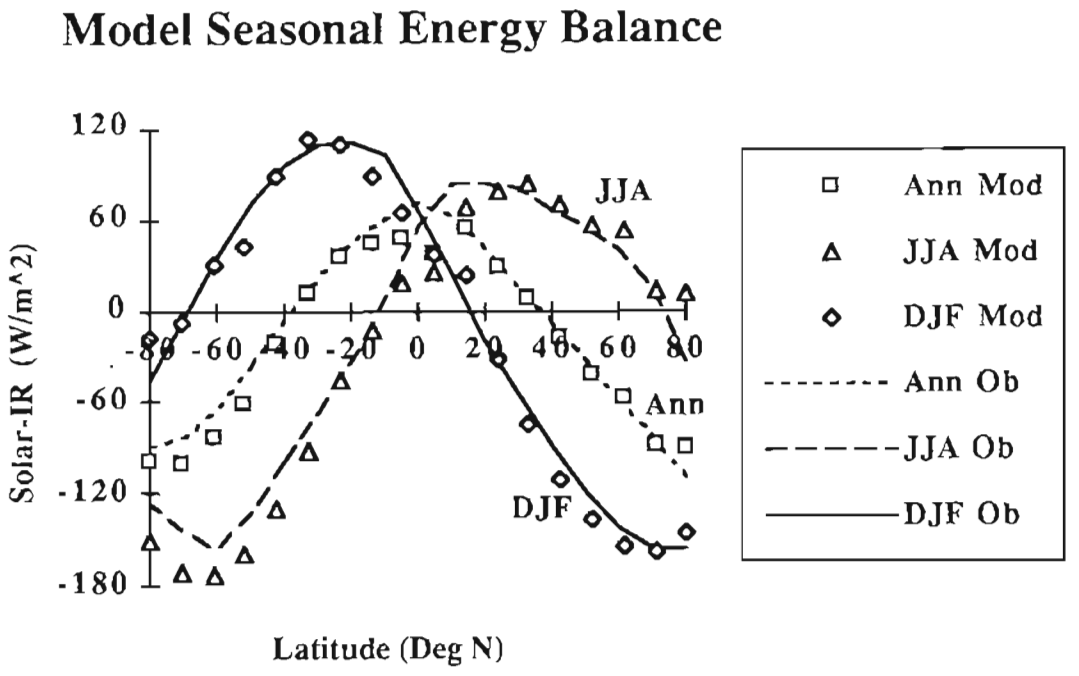

Figure 3.12. Difference between the total absorbed solar energy and the flux of infrared radiation out of the top of the atmosphere as a function of latitude. Observed values are represented as line and model output are indicated by symbols. Observations were digitized from Peixoto and Oort (1992) page 128 and Units are W/m^2.

3.2.2 Energy Balance

In our model, the lack of poleward energy transport in the upper troposphere results in extremely low polar stratospheric temperatures and unrealistically high pressure gradients . We thus added the parameter C_T and equation 2.43 to reduce this problem. However in doing so, as is evident by figure 3.10, we have overestimated the transient eddy sensible heat flux. This over estimate of sensible heat flux is most likely

compensated in our model from an underestimate of latent heat flux or advective heat flux since we reproduce the basic temperature structure of the atmosphere fairly well. However, it is likely that the existing discrepancies between our model and observations could be reduced with an improvement of the parameterization of eddy heat (both latent and sensible) fluxes . Future improvements in the parameterization of eddy sensible heat flux must try to capture the secondary maxima found in both hemispheres near the tropopause.

In figure 3.12 the difference between the total absorbed solar energy and the outward flux of terrestrial radiation at the top of the atmosphere is presented. Since when the model reaches steady state, the annual mean surface temperature remains in equilibrium, the curve for the annual average can be interpreted as the net poleward flux of energy out of a region from all transport mechanisms. Our model agrees well with the observations, hence we feel that the total energy transport simulated by the model is realistic.

The agreement between model and observations is good at all latitudes except around 5.0°N where the difference is lower than it should be in the summer months and at 80°N where it looks like the surface albedo is too low. There are two possible physical reasons for the difference between the model and observations around 5.0°N: 1) the temperatures at this latitude are over estimated causing the terrestrial radiation leaving the top of the atmosphere to be too large or 2), the amount of solar energy absorbed is underestimated. From an inspection of figure 3.4 we can rule out the first option and by an inspection of figure B.2 and B.3 of appendix B we can see that the second explanation is most likely, in that our cloud data have an unusually high value for this latitude, especially the middle cloud amount. It is possible that the cloud amount has been simply overestimated in this region. Alternately, the large middle and low cloud amounts in this region are probably of the cumulus type which may absorb radiation much differently than the broad spread out stratus type clouds which we have assumed to be in our model. Not only do we need to add a prognostic cloud prediction scheme as mentioned previously, we also need to improve the treatment of the cloud optical properties in our model and allow for a greater variety of cloud types. In fact it is our plan to improve the cloud radiation scheme in our model first and then add a scheme for the prognostic

determination of clouds. Improving the model in a step by step fashion will allow for a clearer understanding of what each improvement does to the model's performance.

A third possible reason for the discrepancy between observations and model at the 5.0 °N position in figure 3.12 is that the model values may be off by four or five degrees latitude due to interpolation of cloud amount to the model grid system. Visually it appears that if the JJA model values in figure 3.12 were all shifted to the left by 5° the observed values and modeled values would be in better agreement. After rechecking our area weighted average interpolation scheme, we feel that this source of error is unlikely. Furthermore, data on total cloud cover presented on page 174 of Peixoto and Oort (1992) support the idea that there is an observed anomalously high cloud cover from 0°N to 20°N during JJA. Thus, it seems that our cloud radiation scheme is at present unable to simulate the effects of tropical clouds very well.

The global mean flux of net radiation (solar-IR) from the top of the atmosphere is shown for one complete year in figure 3.13. Notice that during DJF when the sun is mostly over the southern oceans the net energy input into the earth-atmosphere system is maximum and is minimum during JJA when the solar energy most directly strikes zone with higher percentages of land. The amplitude of the fluctuation (approx. 10 W/m^2) is consistent with the observed difference between DJF and JJA values of 16 W/m^2 reported by Peixoto and Oort (1992) page 129. There are two reasons for this annual cycle in net absorbed enrgy: 1) The solar intensity is maximum at the top of the atmosphere in January due to the fact that the earth is at perihelion then and 2) The low thermal inertia of the land masses that dominate the northern hemisphere result in a substantial fluctuation in the output infrared radiation which is negatively correlated with the net absorbed radiation. Thus when the northern hemisphere land masses are warm in the summer months the net absorbed radiation is a minimum. In addition from figure 3.13 we can see that the annual energy imbalance for the model in steady state is less than 0.5 W/m^2.

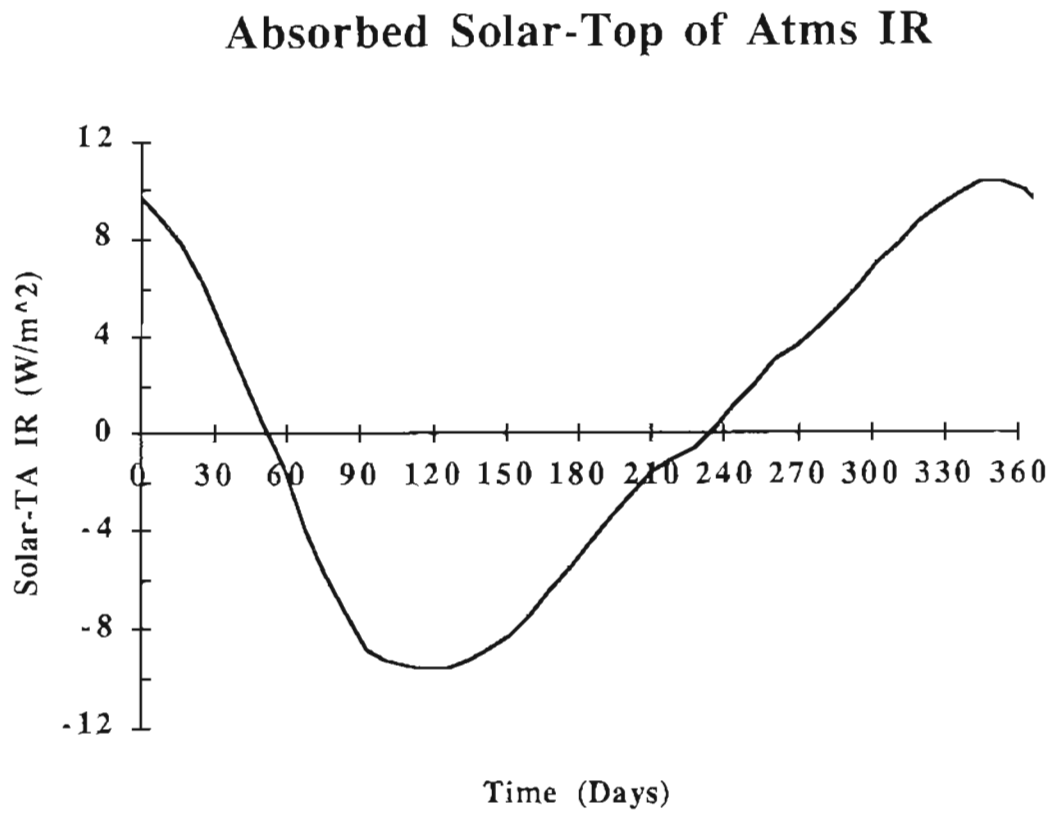

Figure 3.13 Global mean difference between the total absorbed solar energy and the net flux of IR radiation out of the top of the atmosphere. Note that March first corresponds to time=0.0.

3.3 Zonal Velocities U and Angular Momentum.

The zonal velocities predicted by the model, averaged over the last year of the control run, compared with the observations of Newell et al. (1974) are shown in figures 3.14 through 3.16 for JJA, DJF, and annual averages respectively. We have used the Newell et al. data set for comparison because it represents the primary features of the zonal velocity field well, and it was available in numerical form which made it easy to

reproduce accurately. We have also carefully scrutinized the data set published by Peixoto and Oort (1992) page 154 in the analysis of our model's performance at simulating the observed zonal velocities of Earth.

The model predicts weak northern hemisphere jets and strong southern hemisphere jets in JJA which are both consistent with observations. The magnitude and position of the model's JJA southern hemisphere jet is much more consistent with the Peixito and Oort data set in that they show a 35 m/s maximum near 200 mb at around 35° S. However the strength of our northern hemisphere JJA jet stream (15 m/s) is more consistent with the Newell et al. data set. The vertical position of the model jet streams for both hemispheres and all seasons are a little high 100-150 mb compare to 200 mb in both of the observed data sets (except for the Newell data set for the JJA southern hemisphere). We are also lacking a strong tropical easterly in the stratosphere during JJA which is present in both observed data sets. Also the tropical easterlies predicted by the model are generally too high. However, the overall structure and seasonal cycle of the modeled zonal wind field seems to agree well with the observed structure. The annual mean zonal wind field appears to match both sets of observations very well.

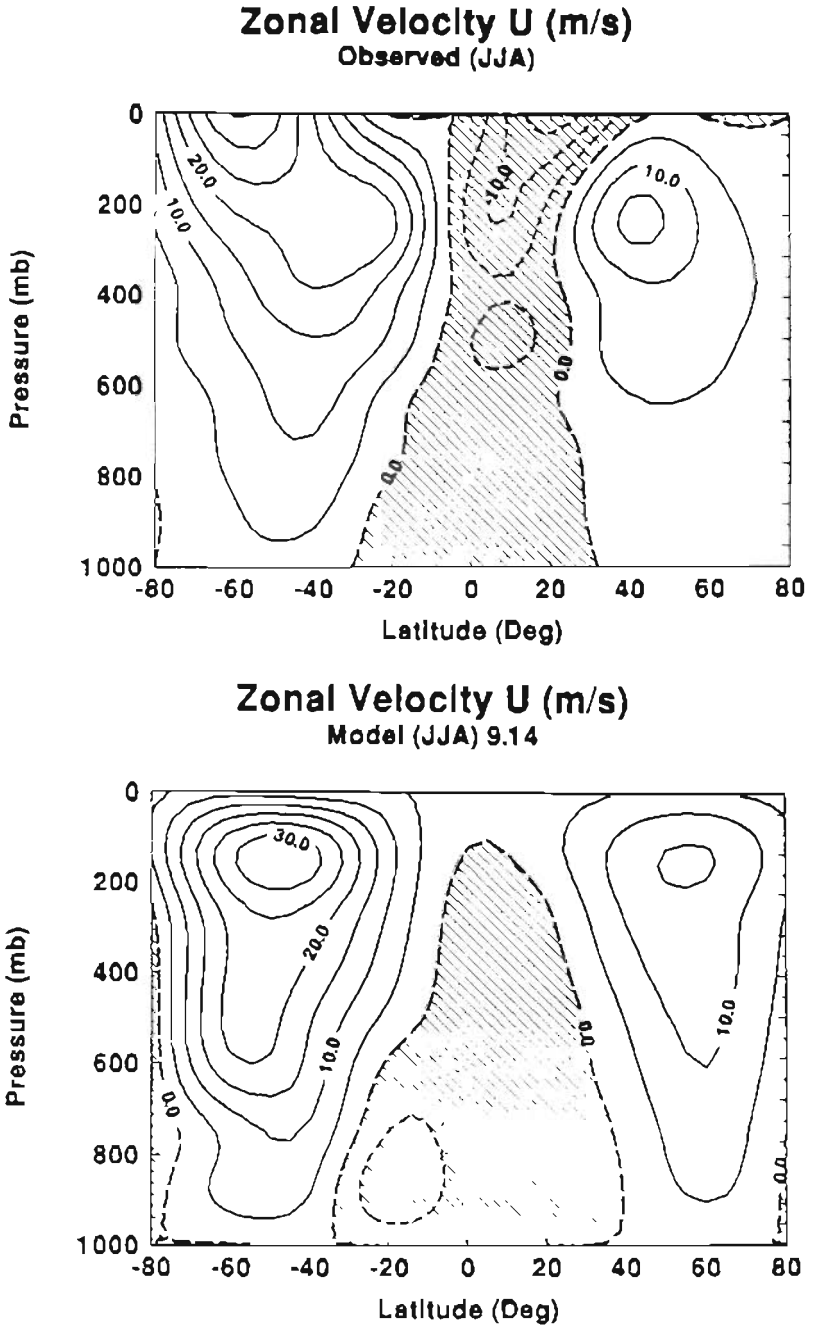

Figure 3.14. Observed zonal-mean (JJA) zonal velocities (top) and Modeled (bottom). Observations from Newell et al. 1974, units are m/s, and regions of easterly winds are shaded.

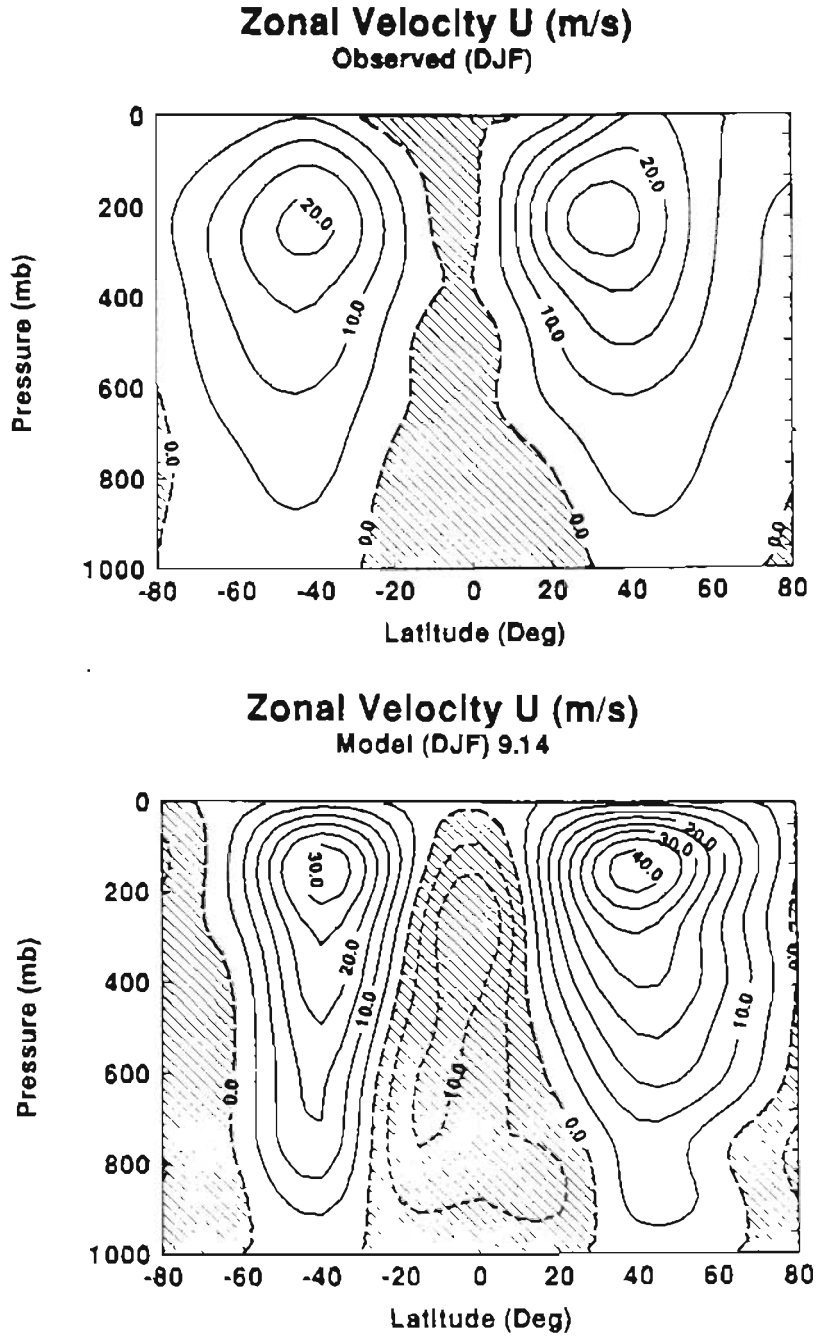

Figure 3.15. Observed zonal-mean (DJF) zonal velocities (top) and Modeled (bottom). Observations from Newell et al. 1974, units are m/s, and regions of easterly winds are shaded.

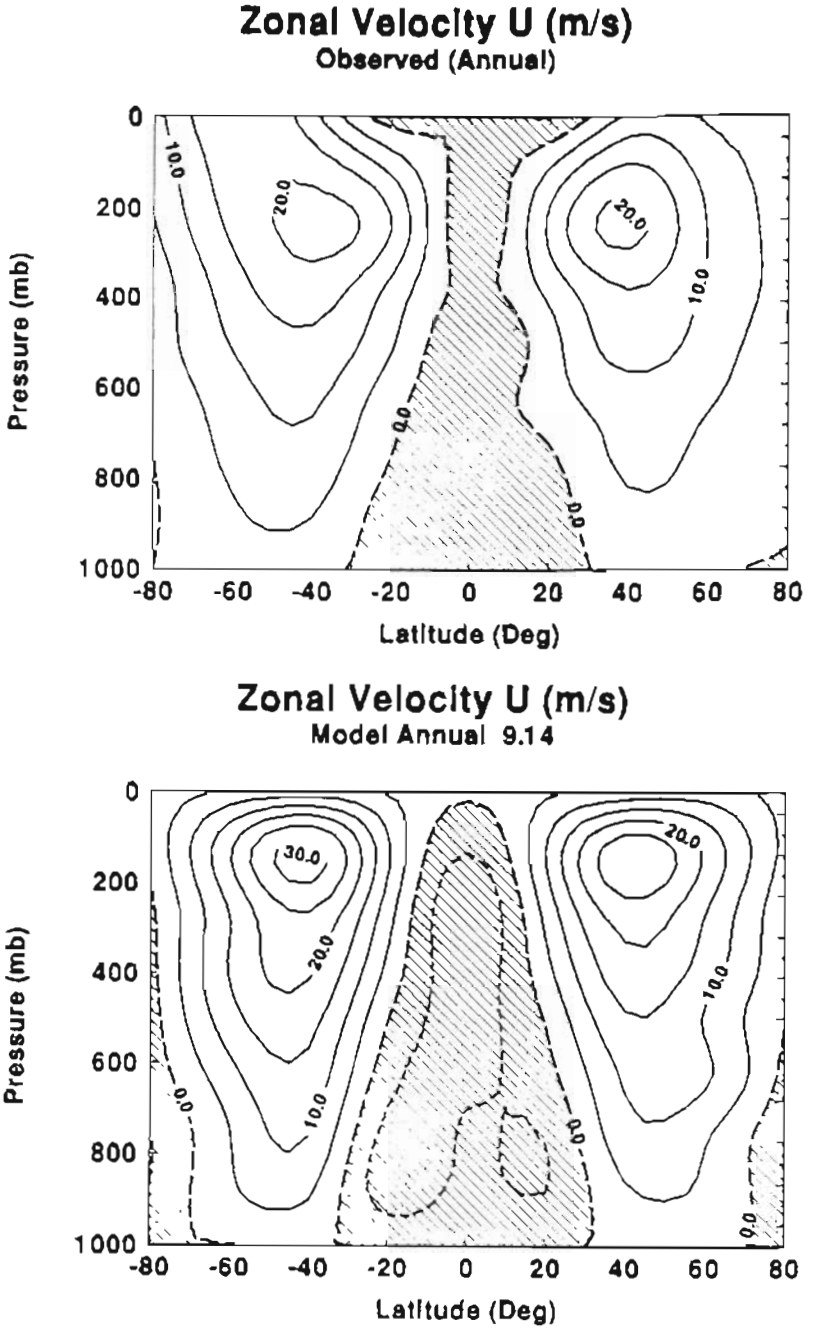

Figure 3.16. Observed zonal-mean annual zonal velocities (top) and Modeled (bottom). Observations from Newell et al. 1974, units are m/s, and regions of easterly winds are shaded.

The observed transient eddy momentum flux u'v' for JJA and DJF are compared with the model parameterized values in figures 3.17 and 3.18, and in figure 3.19 we show the vertical mean zonal average u'v' flux compared with observations. The qualitative agreement between the model and observations is good in that the position of the model's principle maxima are consistent with the observed values. The values of u'v' are dependent upon the meridional temperature gradient and hence, as mentioned previously, increase as the diffusivity parameter C_T of equation 2.43 for T'v' decreases. We have adjusted the value of the parameter C_T to C_T=0.008 $m^3/(K^3 sec)$ to obtain a good model simulation of the temperature, zonal velocity, and eddy turbulent transport of momentum.

As mentioned in the previous chapter, we have modified the Stone and Yao u'v' parameterization scheme slightly. The modification not only enhances the model's stability but also results in better agreement between modeled and observed zonal wind fields. In the modified Stone and Yao scheme, we assume that K_{BC} of equation (2.38) is zero for all latitudes poleward of 65°S and 65 °N. This assumption results in the nonlinear effects K_{NL} giving u'v' values which have stronger negative values at high northern latitudes and stronger positive values at high southern hemisphere latitudes ("reverse" polar fluxes). These "reverse" fluxes at the poles, which do appear in the observations, result in the jet stream positions being focused closer to their observed positions near 35° north and south. This seemed to also help maintain the polar easterlies and reduces the sensitivity of polar pressures and temperatures to the eddy heat flux parameterization constant C_T. Stone and Yao (1987) did not have these "reverse" fluxes at the poles in their January simulation. As can be seen from figure 3.19 (top) we still underestimate the positive u'v' flux in the Antarctic region even with the modifi-cation.

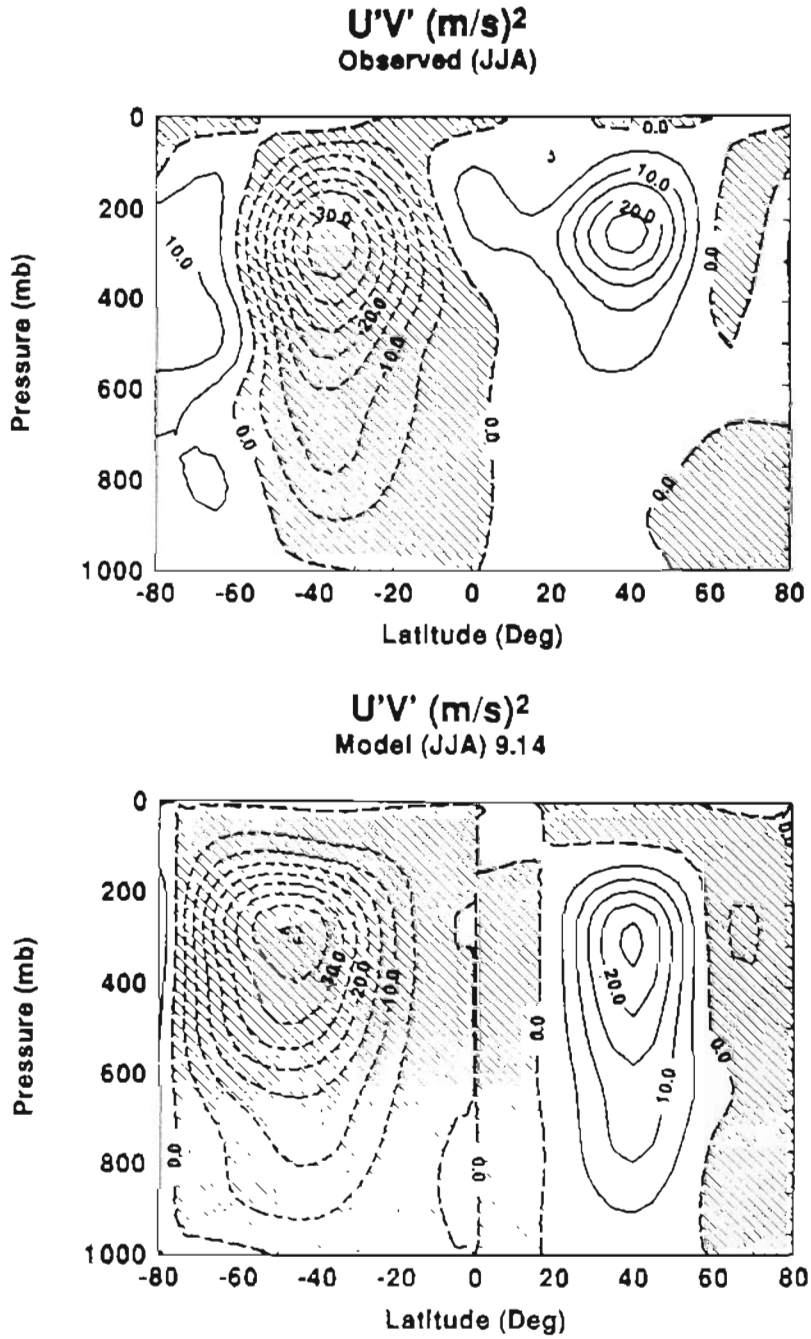

Figure 3.17. (Top) Observed (JJA) transient eddy momentum flux u'v' (Bottom) Modeled. Observations from Newell et al. 1974, units are m^2/s^2.

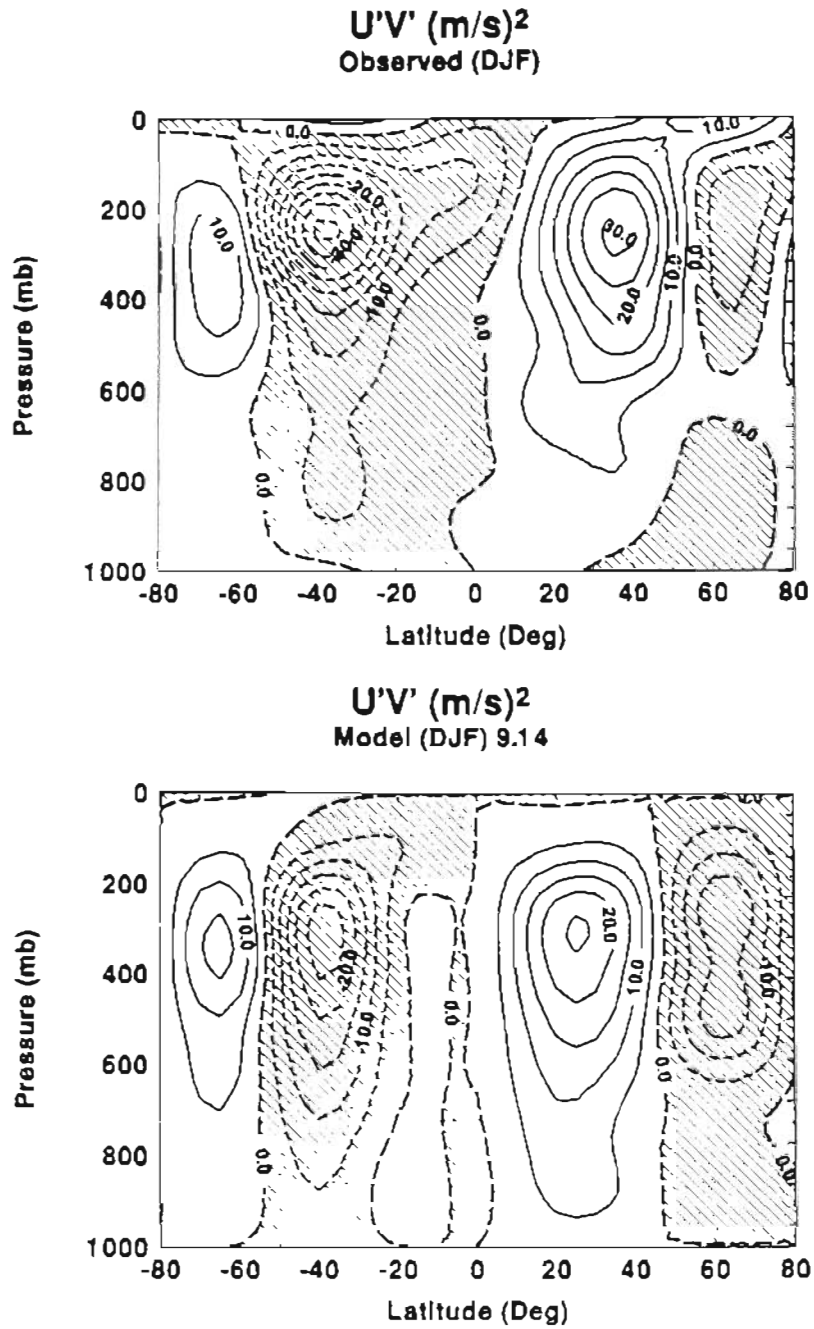

Figure 3.18. (Top) Observed (DJF) transient eddy momentum flux u'v' (Bottom) Modeled. Observations from Newell et al. 1974, units are m^2/s^2.

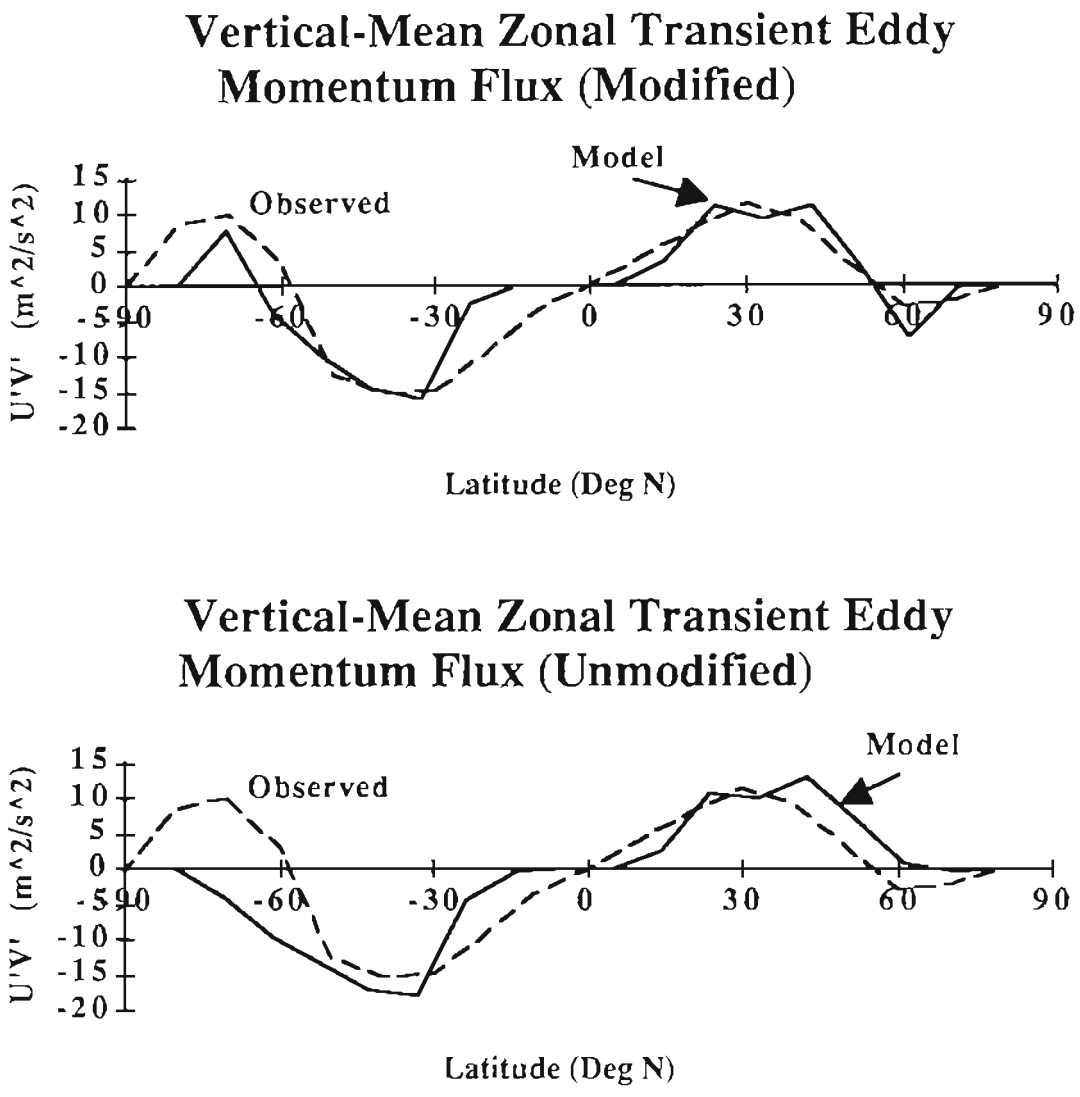

Figure 3.19. The vertical mean zonal average value of u'v' as estimated by our model using the parameterization of Stone and Yao 1987 compared with observations. The observed values were obtained by digitizing data presented by Piexoto and Oort (1992) page 258 and units are m^2/s^2. (bottom) Model with no modification of the Stone and Yao parameterization scheme, and (top) with the modification of the Stone and Yao parameterization scheme described in the text.

One justifiable reason for setting K_{BC}=0.0 for latitudes greater than some high latitude position like 65° is that the parameterization of Stone and Yao (1987) is based on quasi-geostrophic theory about some mid latitude position making it questionable at high latitudes. We have found that the significant difference between the modified and unmodified runs is apparent only in the winter hemisphere. The following discussion will focus on the southern hemisphere during (JJA) but similar features also exist in the northern hemisphere for (DJF).

Figure 3.20 compares the two-dimensional (JJA) zonal wind profile from the control run (with the modification, labeled 9.14) and an identical run that does not have the above modification (labeled 9.21). The major difference between the two is that the southern hemisphere jet stream for the unmodified run is closer to the pole than in the modified run. A close inspection of 3.20 also shows that the strength of the westerly flow at a position of 80°S and a vertical level of 200 mb is about 14 m/s in the unmodified run and about 4 m/s in the modified run indicating that the modification tends to reduce the westerly flow in the high polar atmosphere. The observed values for 80°S and 200 mb are 6 m/s (Newell et al., 1972) and 8 m/s (Peixoto and Oort , 1992). A careful inspection of figure 3.20 also shows that the position of the center of the jet in the unmodified run is about 9° (one latitude grid) farther south than in the modified run; the modified run, although an improvement, still predicts a southern hemisphere jet stream position which is about 10° farther south than the observed position of the jet reported by Piexoto and Oort (1992). Thus the modification, which gives positve (negative) u'v' fluxes in the high latitude south (north), tends to keep the jet streams closer to their observed positions and maintains polar westerly flows with magnitudes closer to those observed.

The poleward shift of the jet in the winter hemisphere in the unmodified run also seems to make the model less stable than it is in the modified run. These strong westerlies are related to large pressure gradients at the poles. Figure 3.21 which is a graph of the 100 m air pressure for the two runs supports this claim. In the unmodified run, the 100 m pressure is about 15 mb lower at 80°S than in the modified run and about 7 mb lower is at 70°S. At all other latitudes the agreement between the two runs is very

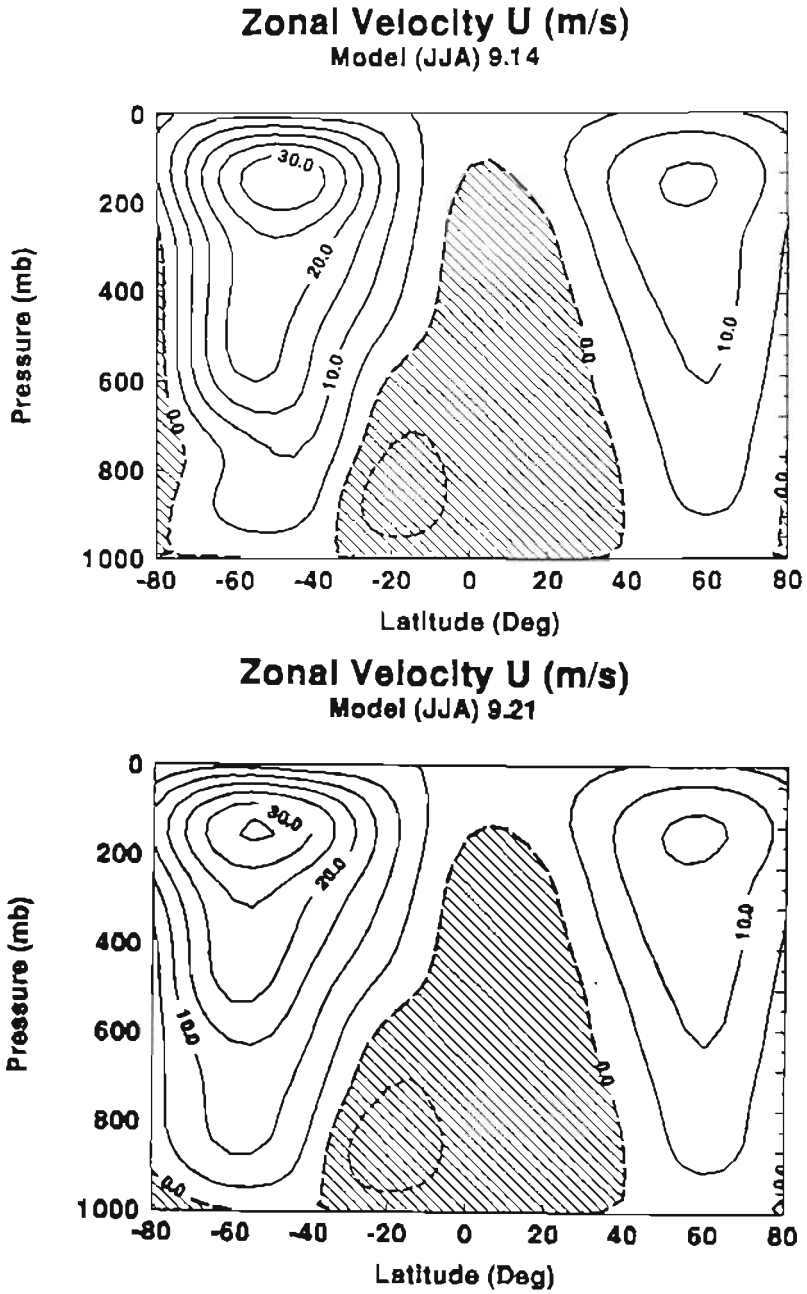

Figure 3.20. The zonal velocity for southern winter (JJA) for the control run (modified Stone and Yao u'v', 9.14) and the unmodified, 9.21.

close. These extra low pressures at the poles of the winter hemisphere add to the instability of the model and are also inconsistent with observations. The described modification of the Stone and Yao u'v' parameterization definately improves the overall performance of the model.

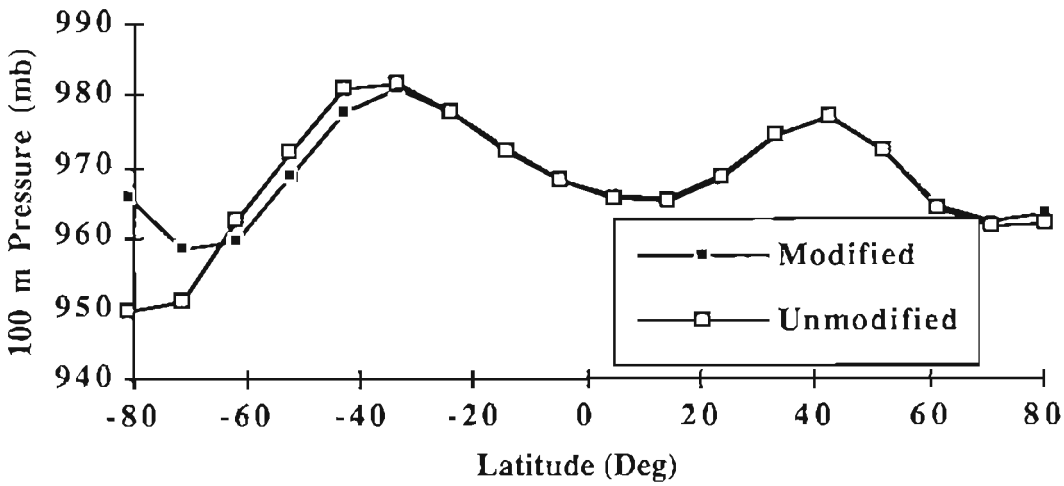

Figure 3.21. The mean 100 m air pressure during southern winter (JJA) for the control run (modified Stone and Yao u'v') and the unmodified, 9.21.

Rosen et al. (1990) performed a careful analysis of NMC (National Meteorological Center) data on global angular momentum for the years 1976 to 1987. They defined the latitudinal belt angular momentum as the total angular momentum contained within a 2.5° latitude belt. We have shown their results, with their computed standard deviations, in figure 3.22 along with the belt angular momentum simulated by our model for the last year of the 20 year control run. Aside from the larger magnitude of the tropical easterlies predicted by the model, the agreement is strikingly good.

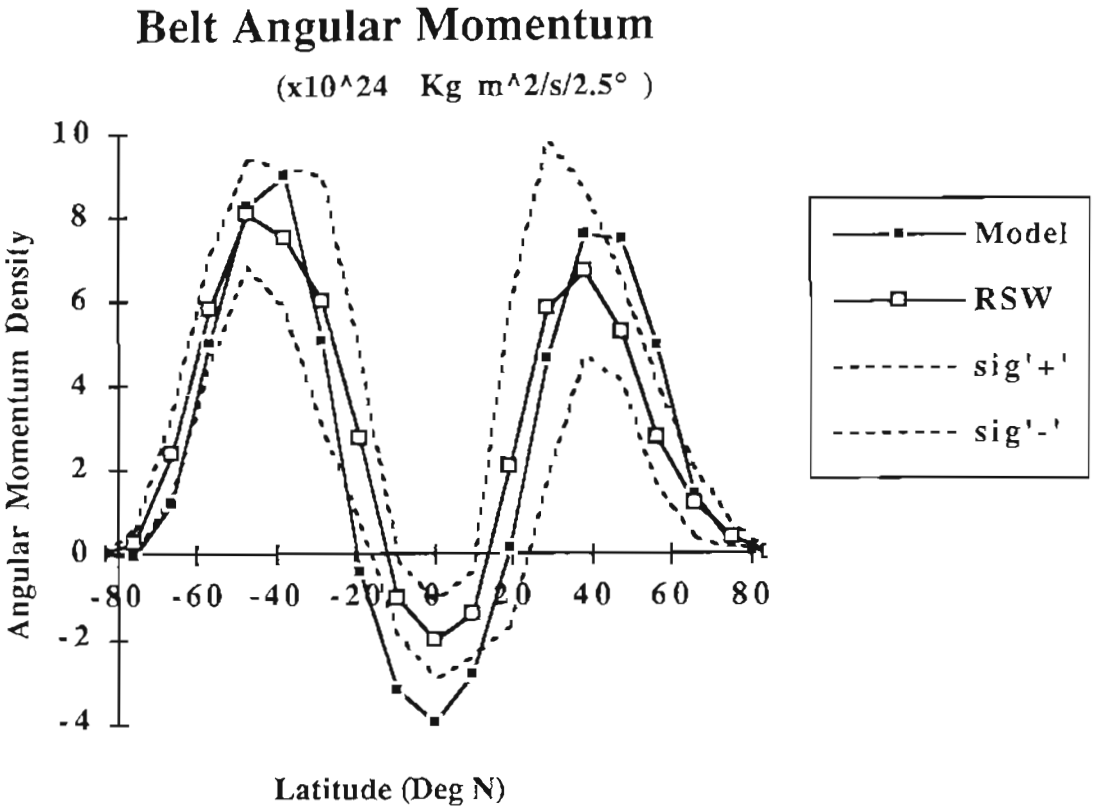

Figure 3.22. Annual mean belt angular momentum predicted by model compared with the analysis of NMC observational data for the years 1976 to 1987. RSW indicates observational analysis of Rosen, Salstein, and Wood (1991). Units are 10^{24}kgm^2/s/2.5°.

In Table 3.1 we compare the observed values of global, northern hemisphere, and southern hemisphere atmospheric angular momentum with our model predictions (in square brackets) for JJA, DJF, and annual means. The units are 10^{25} kgm^2/s and the observed values were from Rosen and Salstein reported by Peixoto and Oort (1992). The agreement between observed and modeled angular momentum is good in the southern

hemisphere. However, the model consistently predicts larger values than observed for the northern hemisphere angular momentum which is also refected in the global averages. Our modeled values should be slightly higher than the values reported by Peixoto and Oort since we have calculated the angular momentum from the surface up to a level of approximately 60 mb and the observed values were calculated only up to 100 mb. We estimate that this will make our values about 3 or 4 % higher than they should be for a direct comparison. Qualitatively our model does predict high DJF and low JJA angular momentum. This large interseasonal difference is consistent with the observations and also the fact that the low thermal inertia of the northern hemisphere allows for a much greater fluctuation in temperature gradients there.

Table 3.1 Angular momentum in units of 10^{25} kgm^2/s observed versus [model]. Observed values obtain from Peixoto and Oort (1992).

Obs [Model]	JJA	DJF	Annual	DJF-JJA
NH	1.4 [4.3]	9.4 [13.0]	5.7 [8.6]	8.0 [8.7]
SH	10.0 [12.0]	6.4 [6.4]	8.4 [9.0]	-3.6 [-5.6]
Globe	11.4 [116.3]	15.8 [19.4]	14.1 [17.6]	4.5 [3.1]

3.4 Kinetic Energy and Mass Stream Function

The vertical mean zonal average atmospheric kinetic energy per unit mass

$$K = \frac{1}{2}\left(\overline{u}^2 + \overline{v}^2\right)$$

predicted by our model is shown in figure 3.23 along with the total observed zonal average kinetic energy per unit mass,

$$K = \frac{1}{2}\left(\overline{u}^2 + \overline{u'^2} + \overline{u^{*2}} + \overline{v}^2 + \overline{v'^2} + \overline{v^{*2}}\right)$$

where u' (v') and u* (v*) are the transient and stationary variations in u (v). The latitudinal distribution of the model's kinetic agrees well with observations. Since we

have not attempted to parameterize the transient or stationary variations in u (or v) from the zonal mean, our model underestimates the kinetic energy of the atmosphere. The model actually overestimates the atmospheric kinetic energy due to mean flows since all of the model's kinetic energy must exist as mean flow kinetic energy.

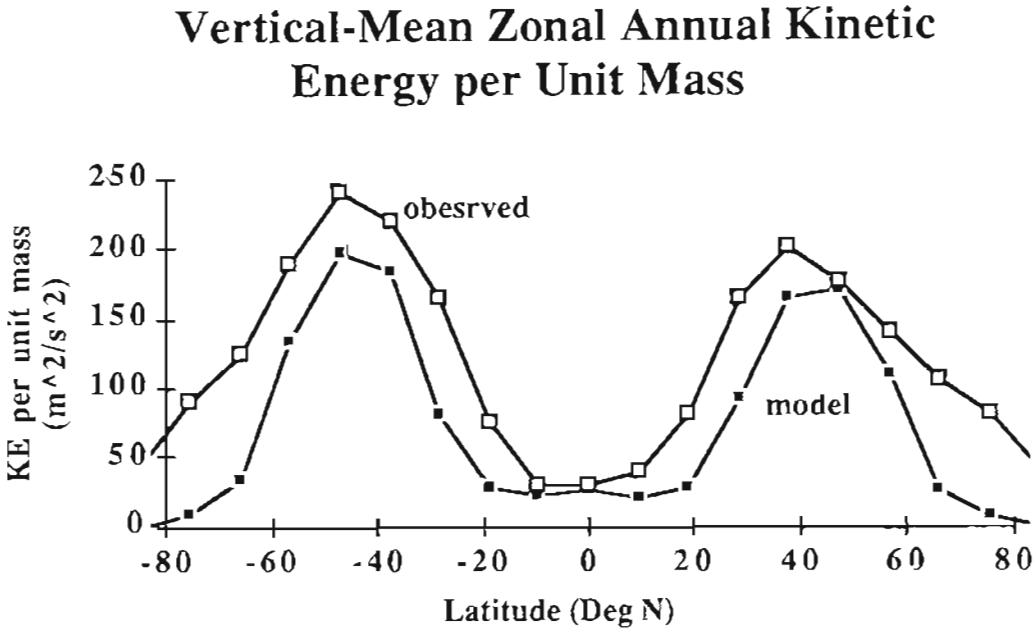

Figure 3.23 Vertical-mean zonal annual kinetic energy per unit mass model versus observed. Observed values were digitized from Peixoto and Oort (1992) page 165 and units are m^2/s^2.

We have also explored the seasonal cycle of the model's kinetic energy and have presented the results in figure 3.24. The atmospheric kinetic energy per unit area is maximum in the winter hemisphere and minimum in the summer hemisphere. The cycle of the global atmospheric kinetic energy per unit area is dominated by the larger amplitude fluctuation of the northern hemisphere and hence is maximum in DJF and

minimum during JJA. These results are qualitatively consistent with the observations reported by Peixoto and Oort (1992) which we have reproduced in table 3.2.

Table 3.2 The integrals of atmospheric kinetic energy per unit area in (J/cm^2) reported by Peixoto and Oort (1992) page 323.

Obs [Model]	JJA	DJF	Annual	DJF-JJA
NH	72	168	116	96
SH	156	100	131	-56
Globe	114	134	123	20

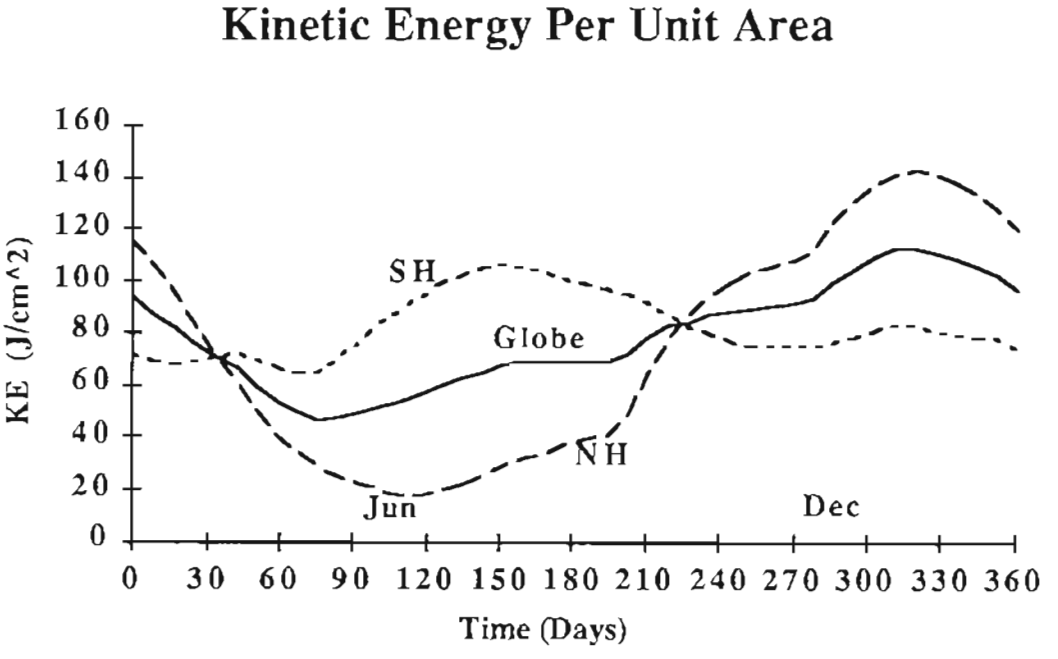

Figure 3.24 Total atmospheric kinetic energy per unit area (J/cm^2) predicted by the model.

In figures 3.25 and 3.26 we compare the model predicted mass stream function ψ for JJA and DJF with values computed by from observations of meridional velocities v,

$$\psi(p) = \frac{2\pi a}{g}\int_0^p \bar{v}\cos\phi dp$$

At present the model underestimates the mass stream function by about 50 % . The model does predict the seasonal movement of the inter tropical convergence zone (ITCZ) fairly accurately and simulates a Ferrel cell circulation reasonably well. In addition, the model is also consistent with observations in that it predicts Hadley circulations that are much stronger in the winter hemisphere.

3.5 Precipitation and Surface Pressure

To simulate the amount of high latitude snow fall for use in the sea ice model that we have been developing, we have estimated the precipitation as described in section 2.3.6. The annual precipitation estimated by the model is compared with the observed values of precipitation in figure 3.27. Despite the fact that we do not have a prognostic treatment of the water vapor cycle included in our model at present, the model estimates magnitudes of precipitation that are surprisingly close to the observed values. Our method for estimating precipitation is to produce precipitation whenever the atmosphere cools either by radiation or dynamical transport so as to keep the relative humidity of the atmosphere constant. Since the model reproduces the tropospheric temperature field well, has a water vapor profile close to the observed, and has a cloud amount that is based on observations; our method for estimating the precipitation gives a good first approximation. The model does however fail to reproduce the subtropical dry regions and the secondary maxima in the mid latitudes. However, the model's estimate of precipitation in the polar regions is good and the shape of the precipitation profile in the tropics agrees well with observations. Thus, these precipitation values can be used in estimating snow cover over sea ice in the development of a prognostic sea ice model.

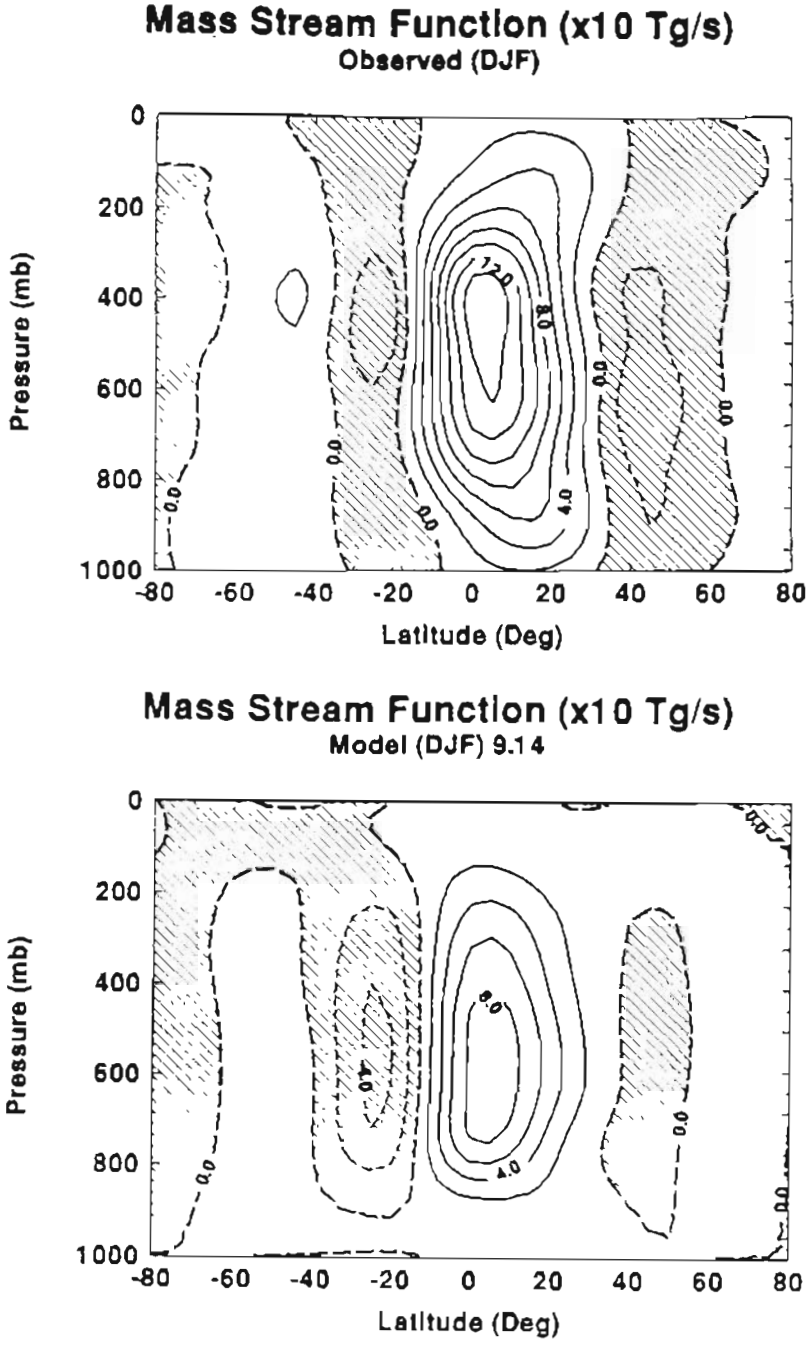

Figure 3.25 (top) DJF mass stream function from observations (bottom) Model. This figure was presented by Peixoto and Oort (1992) page 159 and was digitized by us. It should only be used for a qualitative evaluation and the original should be used for any quantitative work.

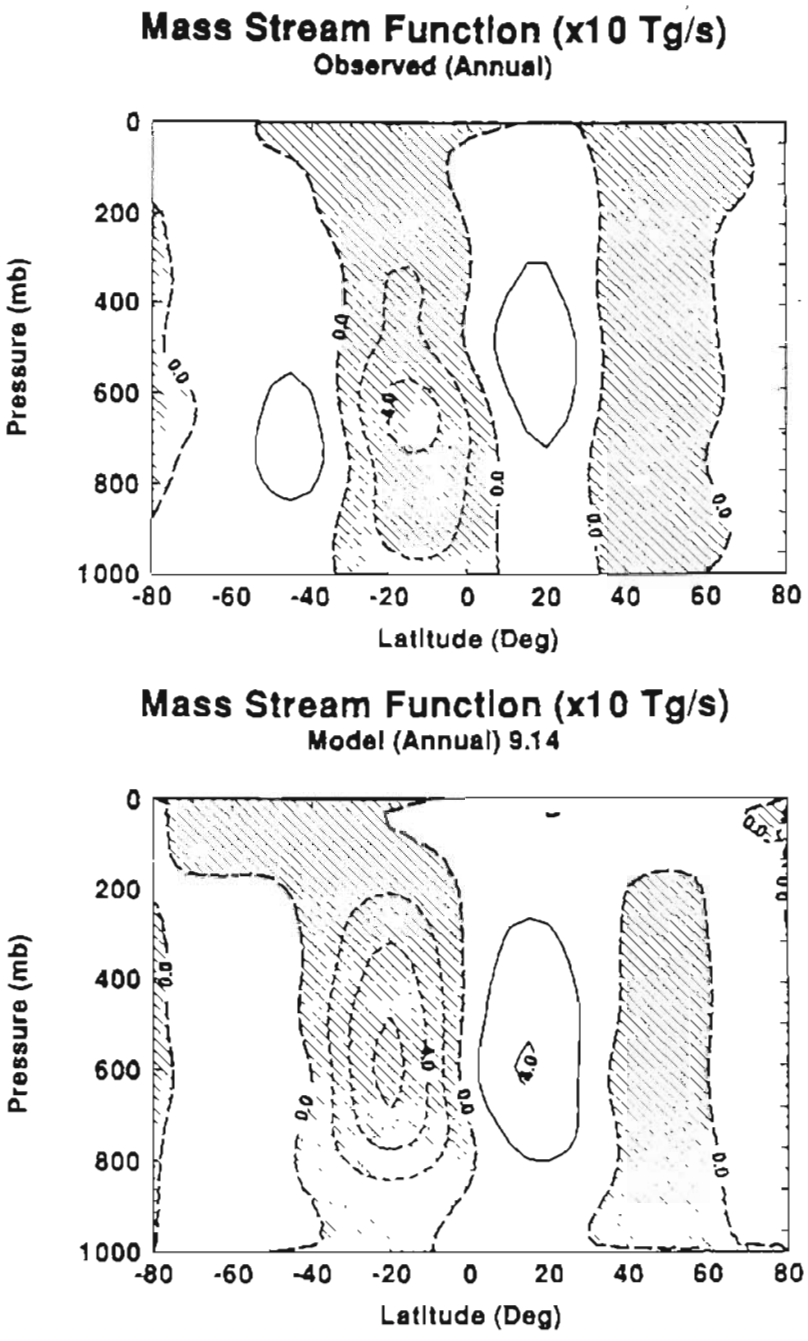

Figure 3.26 (top) Annual mass stream function from Model (bottom) Model. This figure was presented by Peixoto and Oort (1992) page 159 and was digitized by us. It should only be used for a qualitative evaluation and the original should be used for any quantitative work.

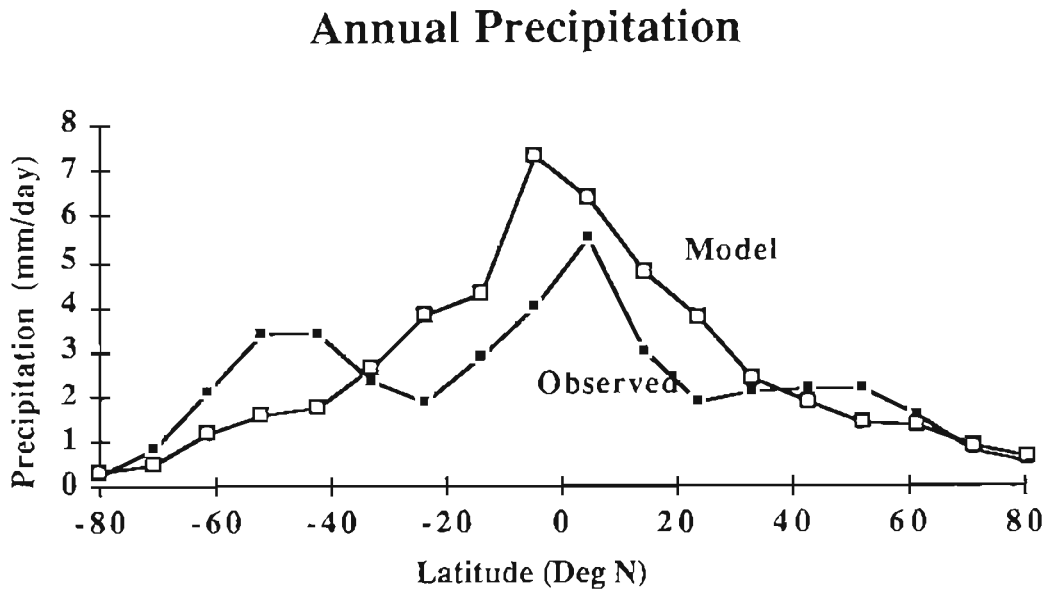

Figure 3.27 Annual latitudinal precipitation profiles; Observed values were digitized from data presented by Peixoto and Oort (1992) page 168 and units are in mm/day.

The model predicted latitudinal surface pressure profile is shown in figure 3.28 along with the observed surface pressure profile and the air pressure at 100 m above the surface with an added offset of 40 mb. The model's average surface pressure is slightly higher than observed but the positions of the tropical low and subtropical highs in both hemispheres are consistent with observations. The amplitude of the surface pressure disturbance simulated by the model is much larger than observations although at 100 m the pressure amplitude is more comparable to observations. Since the z=100 m pressures are actually used in the prognostic calculations for momentum transfer, they tend to give a better indication of the model's performance than the calculated pressures at the surface. It is worth emphasizing at this point that when the model is run without the modification

to the Stone and Yao u'v' flux parameterization scheme described previously, the model fails to reproduce the slight increase in pressure observed at the poles (see figure 3.21).

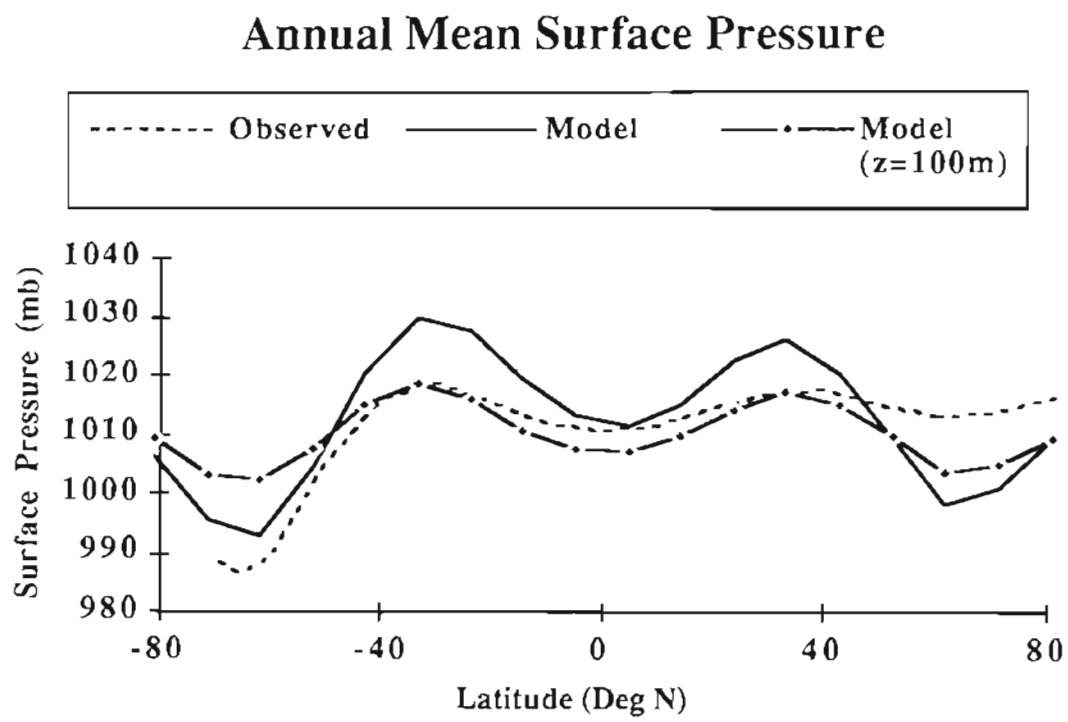

Figure 3.28 Annual model surface pressure and pressure at z=100 m (+40 mb), compared with observations of sea level air pressure digitized from Peixoto and Oort (1992) page 135. Note: an offset of 40 mb has been added to the z=100m pressure for easier comparison.

3.6 Summary of the Model's Performance

To provide a quick reference to the model's performance, the results of this chapter are summarized in this section in outline form.

Temperature

Global Averages

Table 3.3. Model vs. observed surface air temperatures for JJA, DJF, and Annual averages. Observations are from NCAR Data Support Section, Scientific Computing Division; Units are Kelvin. Model values are for z=100 m.

Temperature (K)	Model	Obs. NCAR
JJA	288.5	289.1
DJF	284.3	285.6
Annual	286.3	287.4

Latitudinal summary for surface air temperature.

- All model values are within 2% of observations except near the poles where the discrepancy is a maximum of 8% at 80°S for DJF mean.

Two-dimensional summary

- Annual mean temperature profile has less than 5 % error at all positions except for the surface near 80°S (8%) and above 100 mb in the tropics where the model is approximately 10 % too warm.

Energy Balance

- The difference between the net absorbed solar radiation and the outgoing infrared radiation (Solar-IR) is less than 0.5 W/m^2 for the annual mean.

- The difference between observed (Solar-IR) and modeled (Solar-IR) is less than 15 W/m^2 at all latitudes and all seasons except:

  a) At the poles where the maximum difference is at 80°N for JJA average and is 45 W/m^2 difference or about 19 % of the global mean absorbed flux of solar energy (S_{ab}). At 70° N or S the values of (Solar-IR) immediately drop to less than 5 W/m2 for both winter and summer except for the JJA average at 70 S where it is about 20 W/m^2 or about 8 % of S_{ab}.

  b) In the tropics during JJA where the maximum difference is 31 W/m^2 or about 13 % of S_{ab}.

Mean Zonal Velocity u

- The latitudinal and vertical profile of zonal velocities predicted by the model are in good qualitative agreement with observations; see figure 3.22.
- The strength of the model's mean tropical easterlies is about twice as strong as it should be during all seasons;see figure 3.22.

Table 3.4 summarizes the comparison between the maximum zonal velocities (subtropical jets) predicted by the GCRC 2D model and the observations of Newell et al. (1972) and Peixoto and Oort (1992).

- Our results are in better agreement with the observations of Peixoto and Oort (1992) being within 25 % during all seasons and in both hemispheres.
- There is considerable disagreement between the two data sets.

Table 3.5 shows the latitudinal position of the jets predicted by our 2D model compare to the observations of Newell et al. (1972) and Peixoto and Oort (1992).

- Our model tends to predict jet stream positions that are too far poleward, especially during JJA.
- The vertical position of the center of the jets in both hemispheres predicted by our model is closer to 100 or 150 mb where the observed vertical positions are about 200 mb.

Table 3.4 Summary of zonal velocities (in m/s) predicted by our 2D model as compared to the observations of Newell et al. [N. et al.] (1972) and Peixoto and Oort [P & O] (1992) for summer (JJA), winter (DJF), and annual averages. Top three rows are for the southern hemisphere and the bottom three rows are for the northern hemisphere. The columns headed by Δ% (N), Δ% (P&O), and Δ% (N-(P&O)) are the percent differences between the model and the data from Newell et al. (1972), the model and the data from Peixoto and Oort (1992), and the Newell et al. data and the Peixoto and Oort Data.

u	m/s	Model	N. et al.	P & O	Δ% (N)	Δ% (P&O)	Δ% (N-(P&O))
	JJA	32	23	35	39	-9	-34
SH	DJF	30	20	24	50	25	-17
	Annual	30	22	25	36	20	-12
	JJA	15	15	20	0	-25	-25
NH	DJF	40	27	39	48	3	-31
	Annual	25	22	25	14	0	-12

Table 3.5 Summary of latitudinal position of the jets predicted by our 2D model as compared to the observations of Newell et al. [N. et al.] (1972) and Peixoto and Oort [P & O] (1992) for summer (JJA), winter (DJF), and annual averages. Top three rows are for the southern hemisphere and the bottom three rows are for the northern hemisphere.

		Model	N. et al.	P & O
	JJA	48°S	35°S	30°S
SH	DJF	40°S	42°S	41°S
	Annual	42°S	40°S	35°S
	JJA	50°N	41°N	42°N
NH	DJF	41°N	35°N	30°N
	Annual	42°N	40°N	35°N

Eddy Flux of Angular Momentum $\overline{u'v'}$

Table 3.6 shows the maxima of the model zonal mean flux of angular momentum, $\overline{u'v'}$ in m^2/s^2, compared to those observed by Newell et al. (1972) for JJA and DJF.

- The magnitudes of the maxima predicted by the model are within 30 % of those observed.
- The latitudinal position of the maxima predicted by the model are within 10° of the observed position for all seasons; see figures 3.17 and 3.18.
- The vertical positions of the model's maxima are about 300 mb, consistent with observations; see figures 3.17 and 3.18.
- With the modification of the Stone and Yao $\overline{u'v'}$ parameterization described in section 3.3, the model is able to simulate the secondary reverse maxima observed near the poles.
- The modification to the $\overline{u'v'}$ parameterization also helps the model simulate the high pressures observed near the poles.

Table 3.6. The hemispherical maxima of the model zonal mean flux of angular momentum, $\overline{u'v'}$ in m^2/s^2, compared to those observed by Newell et al. (1972) for JJA and DJF

$\overline{u'v'}$	m^2/s^2	Model	N. et al.	Δ%
	JJA	-40	-35	14
SH	DJF	-25	-35	-28
	JJA	25	20	25
NH	DJF	25	30	-17

Mass Stream Function, Surface Pressure, and Precipitation.

Mass Stream Function

- The seasonal movement and position of the ITCZ is simulated very well by the model.
- The model does simulate a weak Farrel circulation in both hemispheres.
- The strength of the Hadley circulation (mass stream function) is about 50 % too low.

Precipitation

- The global mean annual precipitation rate estimated by the model is 3.3 mm/day compared to the observed 2.7 mm/day reported by Peixoto and Oort (1992), which is about 20 % too high.
- The model estimates that the precipitation is maximum in the tropics and minimum at the poles in agreement with observations.
- The model fails to simulate the subtropical dry regions.

Surface Pressure

- The latitudinal profile of the annual mean surface air pressure agrees very well with observations in that the model simulates: 1) the surface pressure low of the tropics, 2) the subtropical highs, and 3) the polar highs.

- The amplitude of the model's surface pressure is much larger (2 to 3 times) than the observed, but the amplitude of the model's air pressure at a height of 100 meters is much more consistent with observations.

CHAPTER 4

MODEL SENSITIVITY

In this chapter we explore the equilibrium response of the model to a doubling of carbon dioxide (the $2xCO_2$ experiment). Many other groups have performed this same experiment so this offers a way to compare our model's sensitivity to that of other climate models. In the first section we will focus on the change in the temperature structure of the atmosphere for a $2xCO_2$ experiment and in the second section we follow the investigation of Rosen and Gutowski (1992) and analyze the change in model's zonal wind field and angular momentum due to a doubling of CO_2. In the third section, in the hope of understanding the model's response more completely, we present the simulated changes in the mass stream function, meridional temperature gradients, and eddy momentum flux due to a doubling of atmospheric CO_2. Finally, in the last section we summarize the results obtained throughout this chapter.

4.1 Temperature Changes due to $2xCO_2$

The changes in the annual mean zonally averaged surface temperatures due to a doubling of CO_2 predicted by five different general circulation models (GCM) are shown in figure 4.1. Most all of the models are qualitatively consistent in that they predict changes in surface temperature which are greater in the high latitudes than in the tropics for both hemispheres. In a recent Intergovernmental Panel on Climate Change report (IPCC 1990), this was also one of the features noted as being common to most climate model predictions. In figure 4.2 we present the latitudinal and seasonal response of our model's zonal surface air (z=100 m) temperature and surface temperature (z=0.0) for the

$2xCO_2$ experiment. First, focusing in on the annual mean case, our model is consistent with the results of others in that the high latitudes experience more warming than the lower latitudes. Notice that this is most pronounced for the surface temperature (z=0.0). Schlesinger and Mitchell (1987) in a review of the response of climate models to changes in CO_2, also note this similarity in the predictions of general circulation models (GCMs). They attribute this preferential warming at high latitudes to the ice albedo feedback, changes in snow cover, and changes in sea ice thickness. Simulation of these processes has been intentionally suppressed in our model and hence some other mechanism must be responsible for the preferential high latitude warming observe in our model.

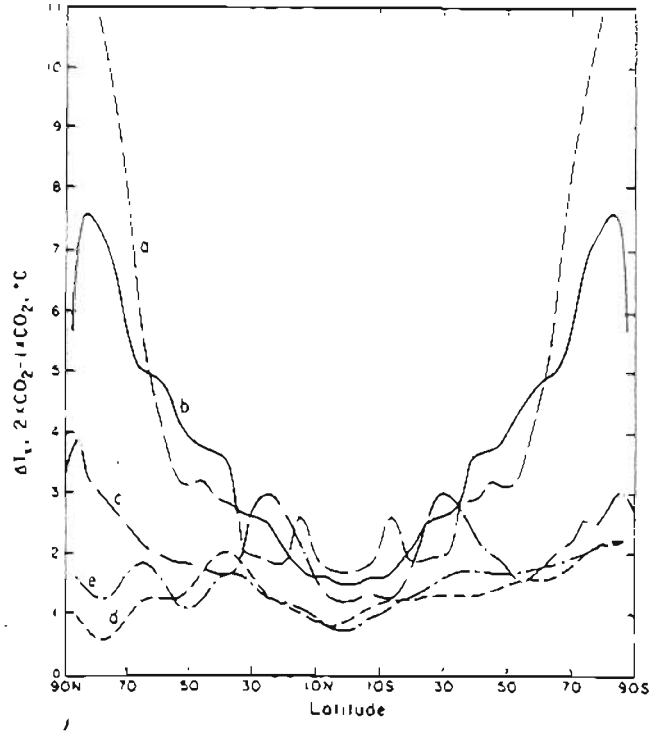

Figure 4.1. Changes in zonal mean surface air temperature (ΔTs) simulated by five GCMs for $2xCO_2$. Curve (a) from data by Manabe and Wetherald (1975) and curve (b) from data by Manabe and Wetherald (1980) are symmetrically plotted about the equator. Other curves: c, Schlesinger (1982); d, Washington and Meehl (1983); computed clouds; e, Washington and Meehl (1983), prescribed clouds. Source: Schlesinger (1984)

One possible mechanism for the polar amplification of the climate sensitivity is that since the absolute humidity at the cool high latitudes is much smaller than it is in the tropics, CO_2 may play a much more important role in the greenhouse warming at high latitudes. We explored this idea using the 1D-RCM described by MacKay and Khalil (1991) and found that its climate sensitivity actually decreased as the model became cooler (less water vapor). We found that when the equilibrium surface temperature for the 1D-RCM control run is 289 K, the increase in surface temperature predicted due to a doubling of CO_2 concentration is 1.9 K. The surface temperature increase is 1.6 K when the control surface temperature is decreased to 272 K by decreasing the solar constant, and 0.9 K for a 235 K control surface temperature. Thus the mechanism proposed above for the polar amplification of the climate sensitivity seems unlikely.

Ramanathan (1977) noted the importance of atmospheric static stability in determing its climate sensitivity. In the tropical regions, where the atmosphere is very unstable the sensitivity is decreased since surface warming will cause an enhancement of convection resulting in enhanced surface cooling, a negative feedback. The opposite is true of a highly stable atmosphere such as the polar atmosphere. Thus, the difference between the static stability of the polar atmosphere and the tropical atmosphere results in this preferential warming in the polar regions.

Certainly the other processes discussed by Schlesinger and Mitchell are very important and should be included in any comprehensive model. However, care must be taken to model positive feedbacks, such as the ice albedo feedback, correctly. Errors in the physics of feedbacks can easily result in an overestimation of the true sensitivity of the climate system to perturbations. For example, in our preliminary work with a thermodynamic prognostic version of a sea ice model we found a polar warming in excess of 12 K for a $2xCO_2$ experiment. This may indeed be a realistic response of the real climate system but this will require closer scrutiny by us.

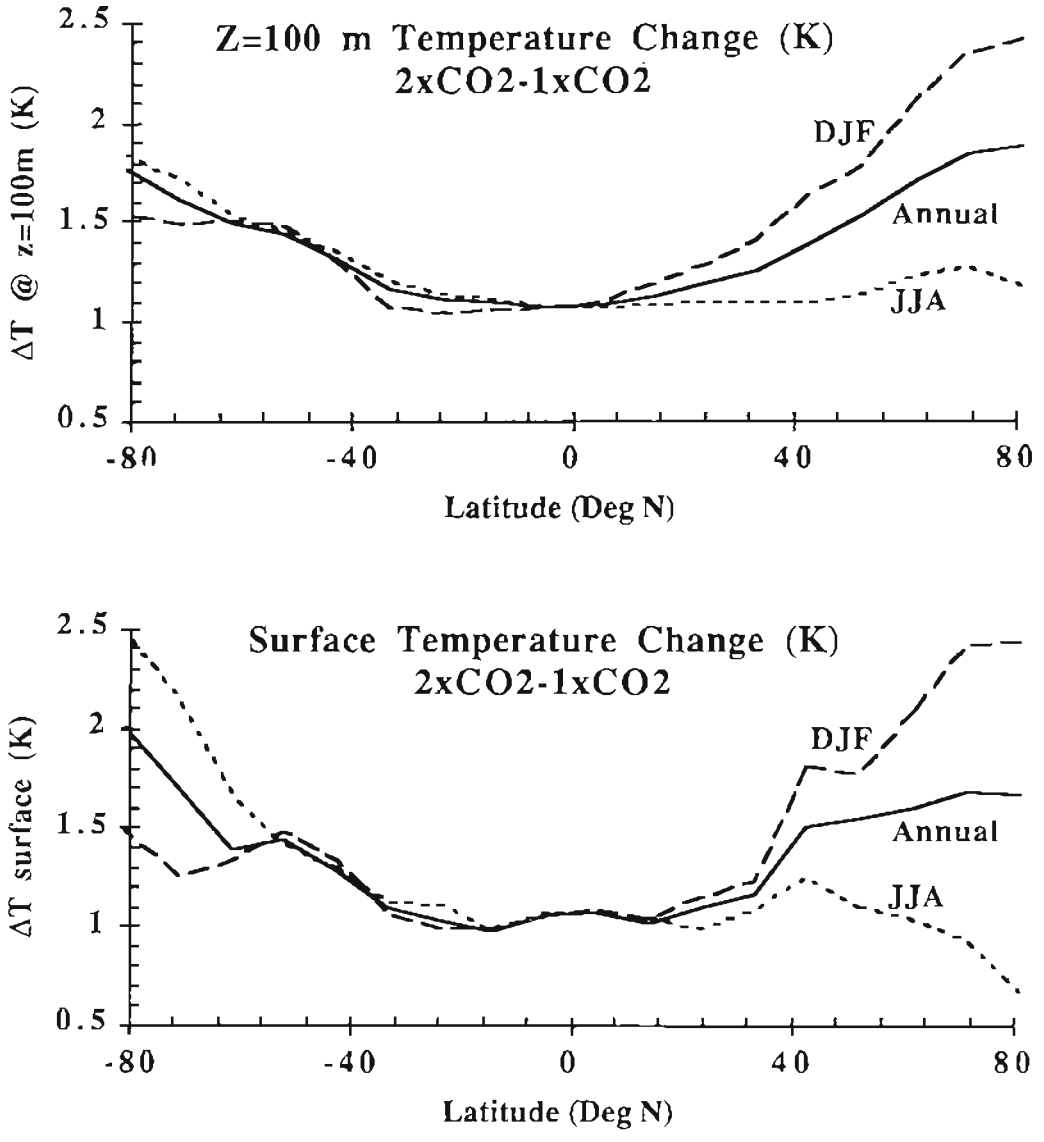

Figure 4.2. The latitudinal and seasonal response of the GCRC 2D climate model to a doubling of CO_2. (top) z=100m air temperature change (K) (bottom) Surface temperature change (K).

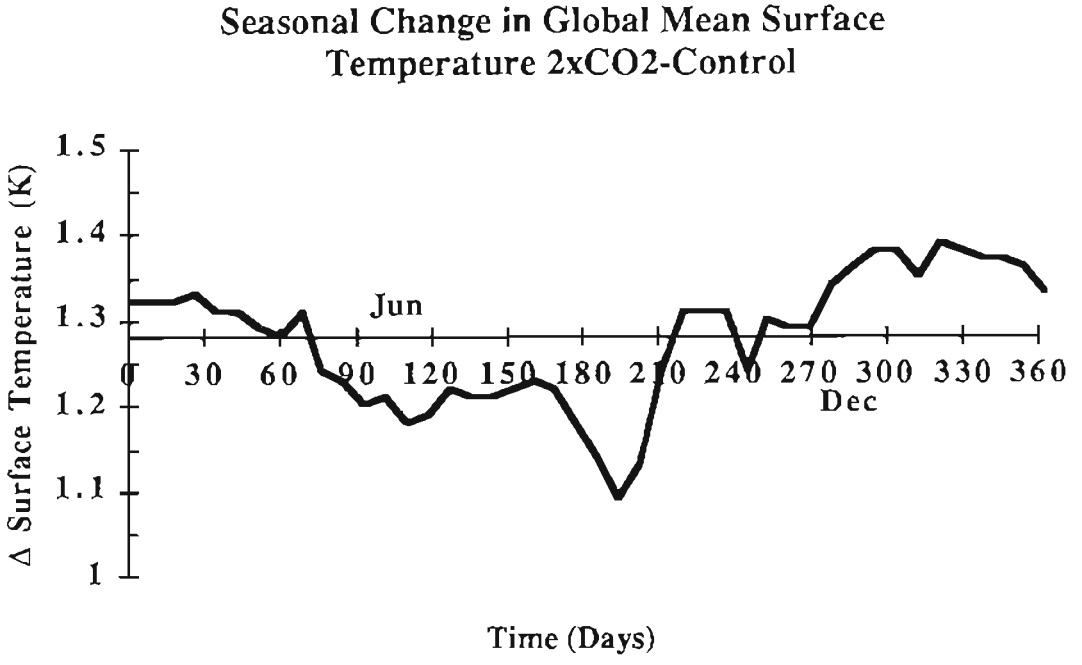

Figure 4.3. One year time series of the change in global mean surface air temperature, ΔT_S , due to a doubling of CO_2 predicted by the OGI 2D climate model.

Another feature of the climate sensitivity that is qualitatively consistent between most models, see for example Schlesinger and Mitchell (1987) or IPCC (1990), is that the warming is greatest during the winter for each hemisphere and least during the summer. This is clearly evident in the response of our model and, at least for our model, is likely a result of the enhanced static stability of the winter atmosphere.

In figure 4.3 we show the change in global average surface air temperature predicted by our model for the $2xCO_2$ experiment, as a function of time for one complete year. The change in global mean surface (z=100 m) air temperature is 1.20 (JJA), 1.35 K (DJF), and 1.28 K (Annual). Our model's climate sensitivity is low compared to the range of ΔT_S between +1.5K and +4.5K with a "best guess" of 2.5 K reported by IPCC

(1990) for the climate sensitivity of most GCMs for a doubling of CO_2. This is expected since we have attempted to suppressed all feedbacks except for the water vapor feedback.

In figure 4.4 we present the changes in the two-dimensional temperature field predicted by our model for JJA and DJF. Held (1993) gave a review of climate models which included a brief description of their sensitivity to a doubling of carbon dioxide. We present his qualitative results in figure 4.5 along with the change in the annual average temperature field predicted by our model. Our results have several features in common with Held's qualitative interpretation of what most climate models predict for the changes in temperature field due to a doubling of CO_2. As noted previously, we have an increase in magnitude of the surface warming as the latitude increases towards the poles in both hemispheres. Note also the maximum in temperature change predicted to occur in the middle to upper tropical troposphere. Held (1993) attributes this maximum warming in the upper tropical troposphere simulated by most climate models to an increase in the vertical transport of latent heat energy in the tropics due to surface warming. Finally, as with most climate models, we have a substantial cooling in the stratosphere. Our model predicts changes in the two-dimensional temperature fields that are qualitatively consistent with most climate models of this or greater sophistication. Although we do have good qualitative agreement between our results and the results of other climate models, our model is quantitatively less sensitive to changes in carbon dioxide than the average climate model. As mentioned previously this is primarily due to the lack of positive feedback processes included in our model.

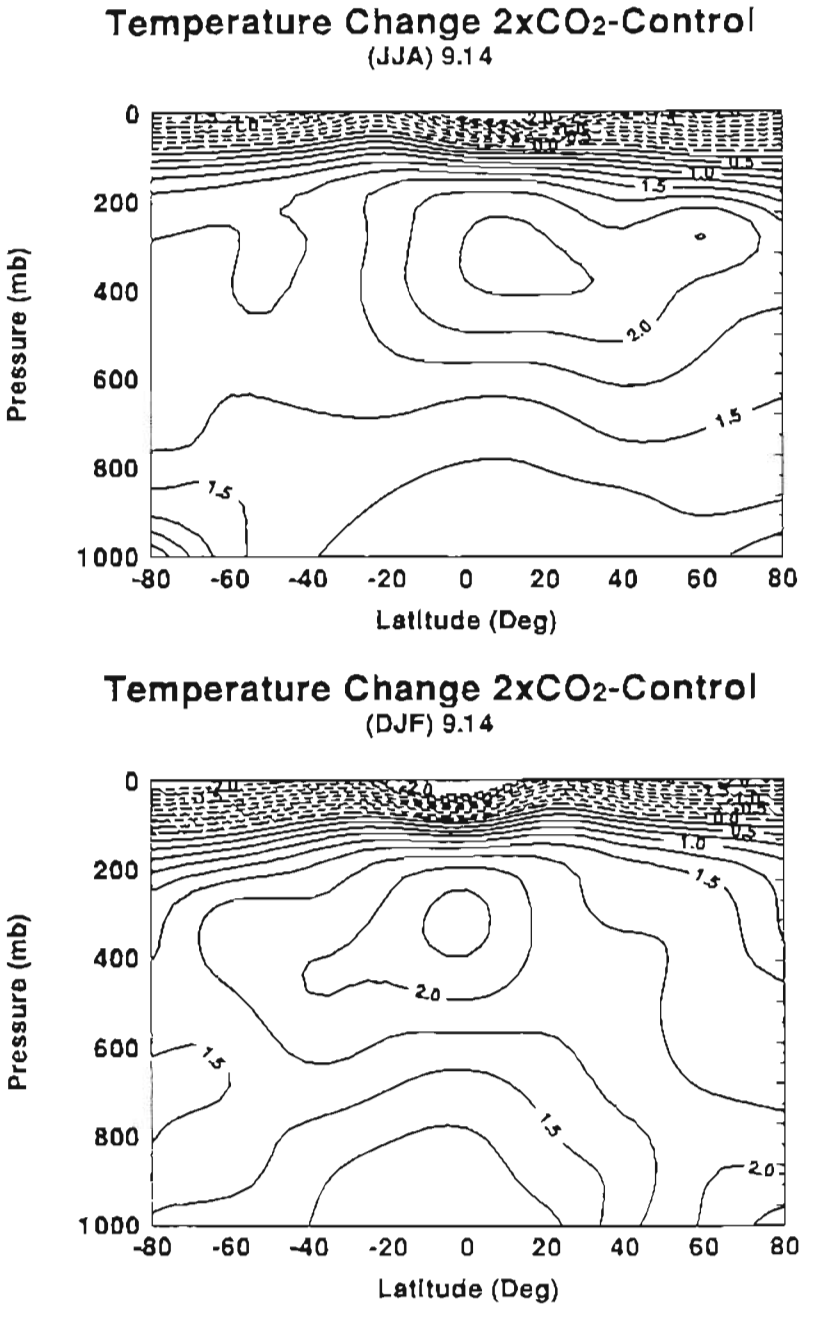

Figure 4.4. The two-dimensional change in the temperature field predicted by the GCRC 2D climate model for (top) JJA, (bottom) DJF averages.

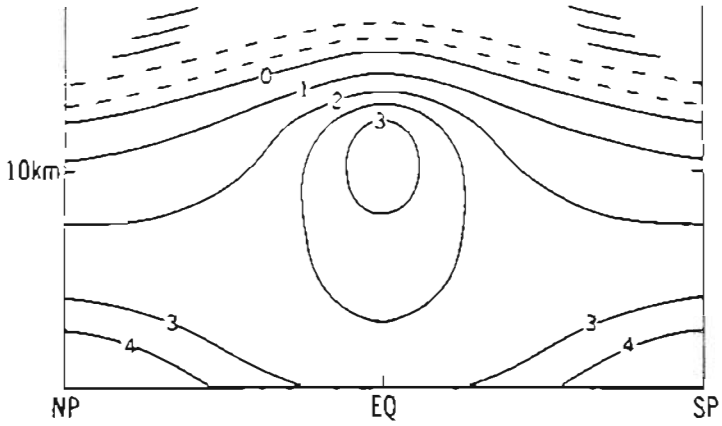

FIG. 6. A schematic of the equilibrium annual mean temperature response to a doubling of CO_2, as typically predicted by GCMs, emphasizing the maxima at upper-tropospheric levels in the tropics and at low levels in the polar regions. Polar amplification is present only in winter.

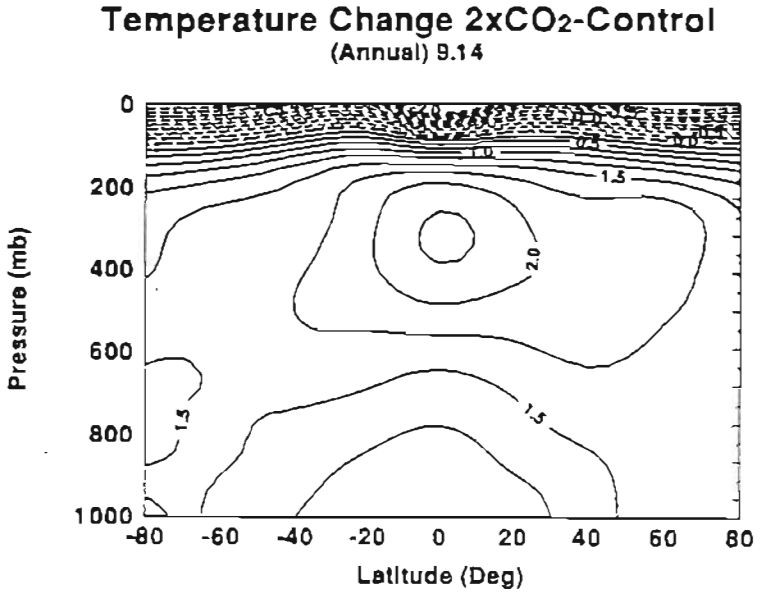

Figure 4.5. (top) Figure 6 of the review article of Held (1993). His figure has been reproduced directly from the Bulletin of the American Meteorological Society with its original caption. (bottom) The two-dimensional change in the annual average temperature field predicted by the GCRC 2D climate model.

With these general features incorporated into our model, the real pay off for the development of this model will come when we use it to explore climate feedback processes such as land and sea ice dynamics, atmosphere-ocean interactions, and cloud processes. With a model of this degree of sophistication and simplicity, we believe it is an essential tool for the exploration of the complex interactions present in the real climate system. In the next section we put our model to work and investigate the changes in zonal velocity predicted by our model for a $2xCO_2$ experiment.

4.2 Changes in Zonal Velocity and the Length of Day?

Because we have a good simulation of the zonal velocity field, we will follow the lead of Rosen and Gutowski (1992) and explore how the model's mean zonal velocity field changes for a $2xCO_2$ experiment. Rosen and Gutowski (1992) investigated the response of the zonal velocity field to a doubling of CO_2 for three different GCM outputs archived by the Goddard Institute for Space Studies (GISS), the National Center for Atmospheric Research (NCAR), and the Geophysical Fluid Dynamics Laboratory (GFDL).

In Figure 4.6 we have reproduced Rosen and Gutowski's figure 2 with the addition of the results from our model for the change in total global angular momentum due to a doubling of CO_2 for JJA and DJF. Also shown are the model differences between the JJA and DJF global angular momentum which should be compared to the observed range of the seasonal cycle of approximately -4.4×10^{25} kg m^2/s reported by Peixoto and Oort (1992) in reference to earlier work of Rosen and Salstein (1983).

The changes in the mean zonal velocity, u, predicted by the three GCMs due to a doubling of CO_2 are shown in figures 4.7 and 4.8 for DJF and JJA respectively. These figures, which have been reproduced from Rosen and Gutowski's 1992 paper, also include the interannual standard deviations of u as an estimate of the noise of the observed signal. An inspection of these figures reveals that it is very difficult to identify any consistent patterns between the simulated change in zonal velocity predicted by the GCMs in either season. Rosen and Gutowski are very conservative about drawing any concrete conclusions regarding the consistent response of the GCM's zonal wind fields to a doubling of CO_2. They do say however that "only in JJA are the patterns of the tropical

wind field response similar enough to yield significant correlation coefficients among the three models, and yet even these coefficients are not impressively high". They also note that if the NCAR results were valid, then the change in the length of day accompanying the change in angular momentum resulting from the doubling of atmospheric carbon dioxide would be on the order of 0.3 ms which is detectable with modern instrumentation. This estimate was based on the conservation of the angular momentum of the solid earth-atmosphere system and an assumed moment of inertia of the solid earth. Rosen et al. (1990) give

$$\Delta l.o.d. = 1.68 \times 10^{-29} \Delta M$$

for the relationship between the change in length of day and change in total angular momentum of the atmosphere.

In figure 4.9 we show the change in zonal velocity during JJA and DJF as simulated by our 2D model for the $2xCO_2$ experiment. In both seasons we have a very small change in total global angular momentum $+0.16 \times 10^{25} kgm^2/s$ (JJA), -0.14×10^{25} kgm^2/s (DJF), and for the anual case the model predicts an angular momentum change of 0.02×10^{25} kgm^2/s. Hence, our model predicts that the change in length of day will be insignificant. However in all cases we find very distinct patterns of change predicted by our model: 1) an increase in tropical easterlies ($\Delta u<0$), 2) an increase westerly flow in the mid to high latitude regions, and 3) an increase in easterly flow in the Antarctic region. It is interesting to note that despite the zonal patterns of change, the net change in angular momentum predicted by our model is nearly zero. This is most likely due to the fact that we have neglected orography in our present model and have assumed that the frictional drag at the surface is independent of latitude. Hence, there is a relatively weak coupling between the atmosphere and surface making it difficult to have a net transfer of angular momentum between the two.

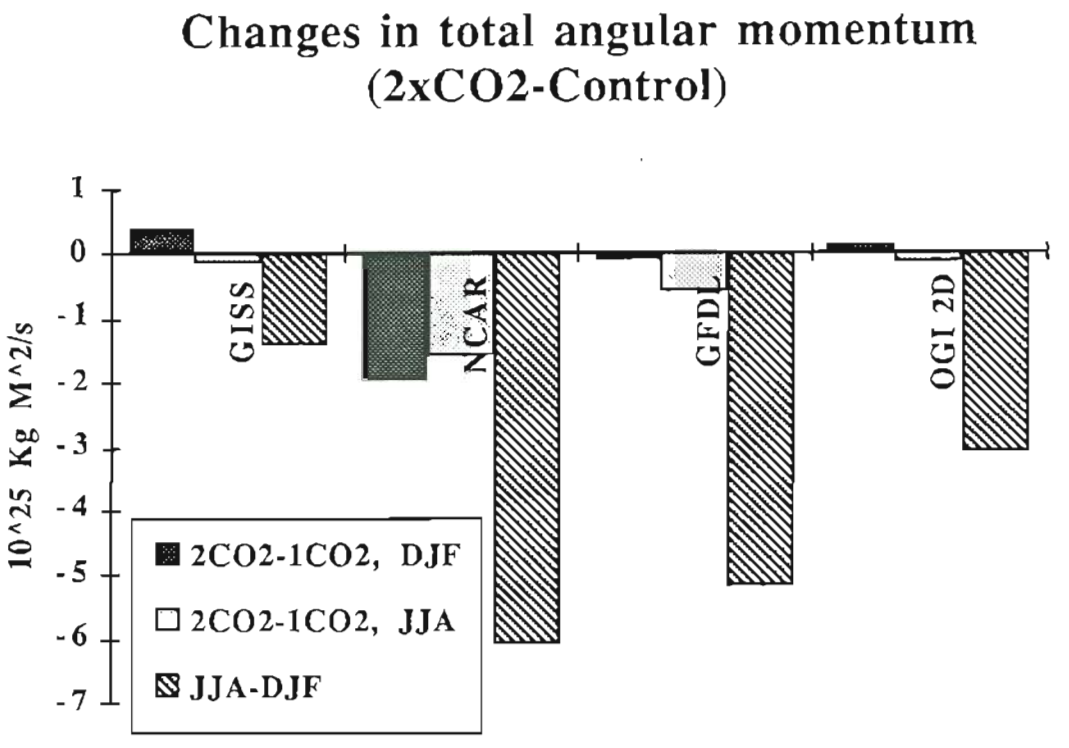

Figure 4.6. Differences in total global angular momentum ($2xCO_2$-$1xCO_2$) predicted by each model, for JJA and DJF averages, along with JJA minus DJF determined from control runs of each model. The Observed JJA -DJF angular momentum difference reported by Peixoto and Oort (1992) from Rosen's data up to 100 mb is -4.4 $x10^{25}$ kg m^2/s and Rosen and Gutowski give a standard deviation of this difference for the angular momentum up to 200 mb of approximately 1.0 $x10^{25}$ kg m^2/s. GCM data was digitized from data presented by Rosen and Gutowski (1992).

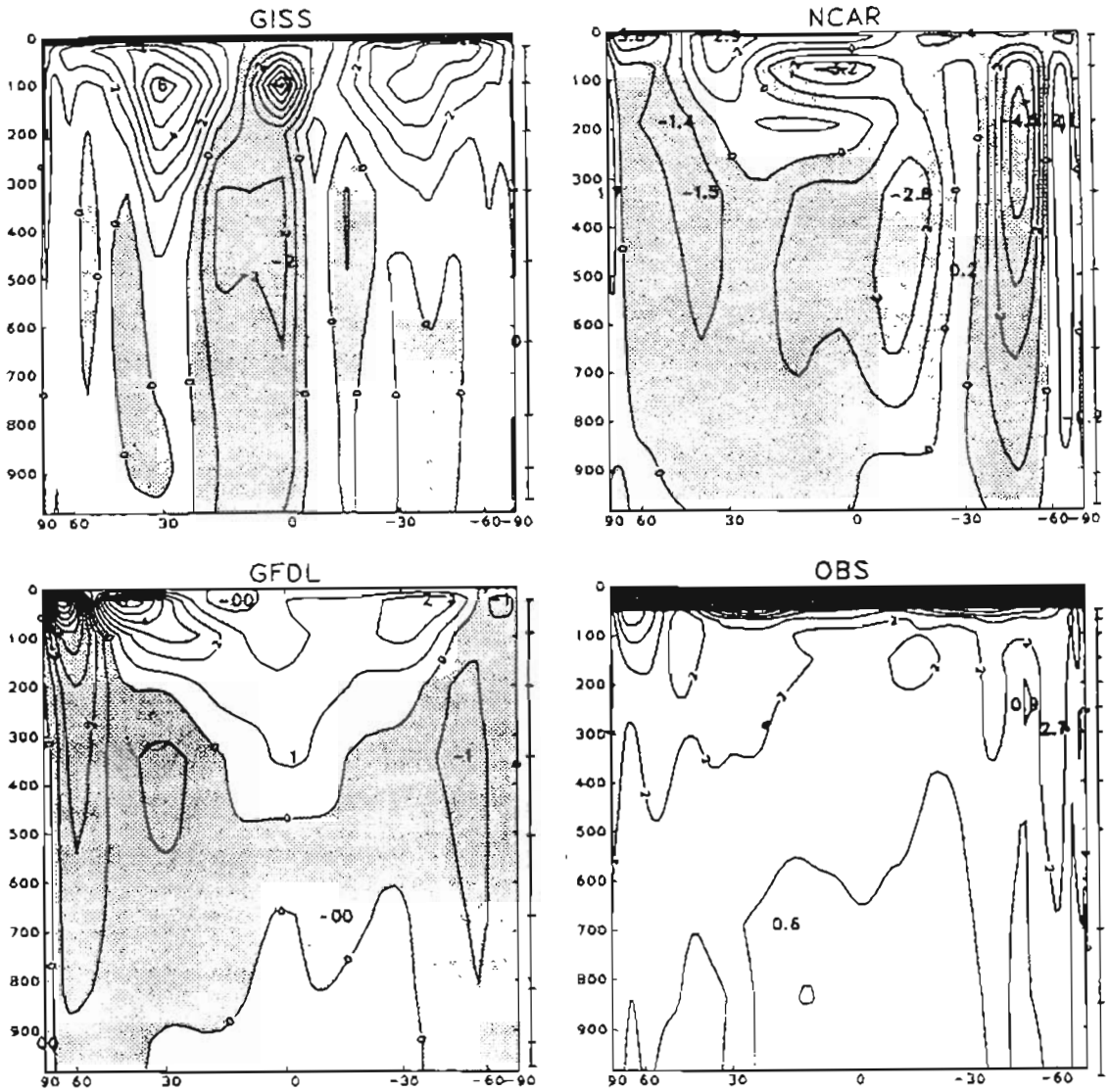

Figure 4.7. Differences in zonal velocity resulting from a $2xCO_2$ experiment predicted by the three GCMs of Rosen and Gutowski's (1992) study for DJF. Also shown are the interannual standard deviations of the zonal velocity based on 14 years of NMC data. Units are meters per second and shaded regions represent decreases in zonal velocity. The ordinate is the vertical pressure in units of millibars and the abscissa is the latitude in degrees north.

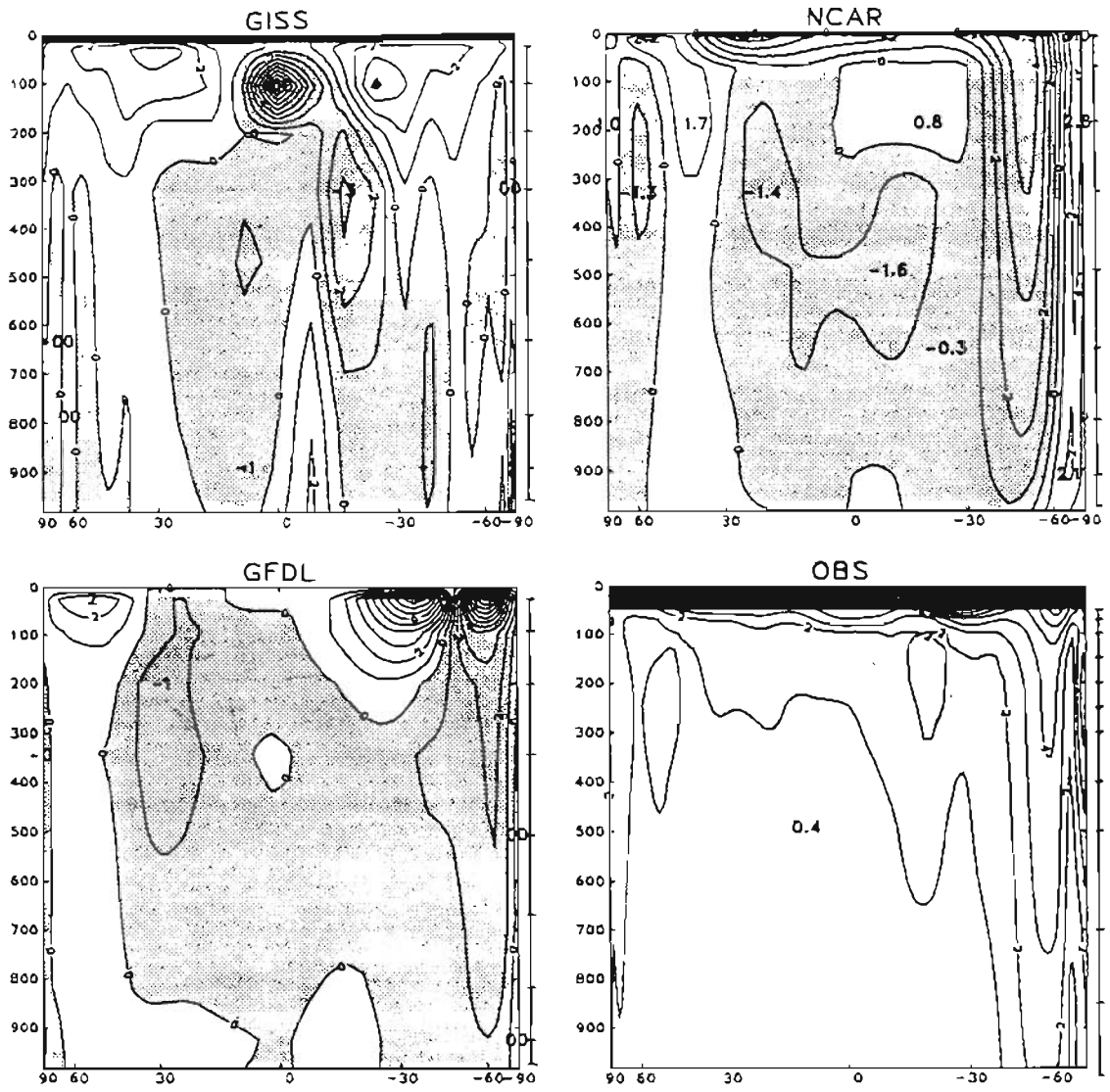

Figure 4.8. Differences in zonal velocity resulting from a $2xCO_2$ experiment predicted by the three GCMs of Rosen and Gutowski's (1992) study for JJA. Also shown are the interannual standard deviations of the zonal velocity based on 14 years of NMC data. Units are meters per second and shaded regions represent decreases in zonal velocity. The ordinate is the vertical pressure in units of millibars and the abscissa is the latitude in degrees north.

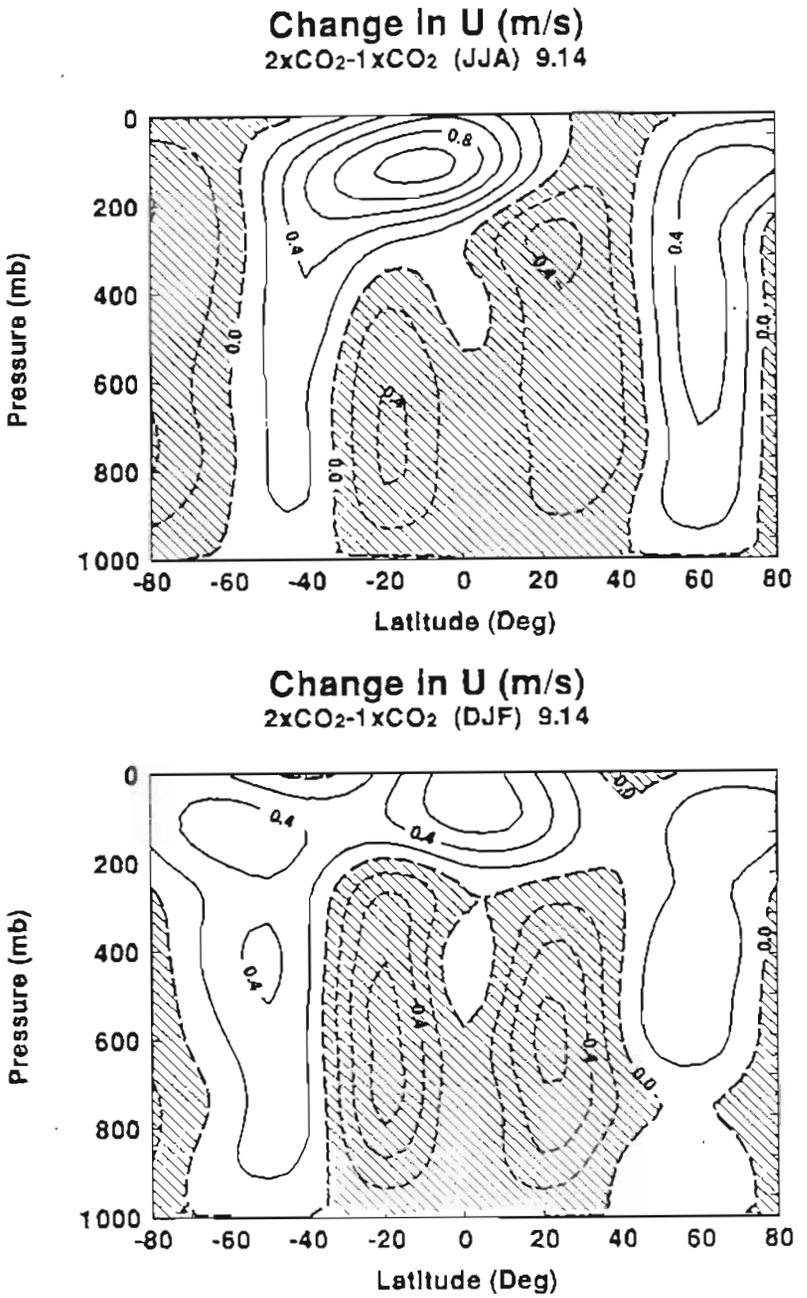

Figure 4.9. Differences in zonal velocity resulting from a $2xCO_2$ experiment predicted by the GCRC 2D model for JJA and DJF. Units are meters per second and shaded regions represent decreases in zonal velocity.

4.3 Other Changes for $2xCO_2$

To gain further insight into the model's sensitivity to the doubling of atmospheric carbon dioxide it is interesting to look at the corresponding changes in other variables of the model such as global kinetic energy, meridional temperature gradient, mass stream function, and eddy momentum flux, u'v'. The time series of the change in global kinetic energy per unit area calculated from the last full year of the control run and the $2xCO_2$ run is shown in figure 4.10. Over the full year there is an average increase of kinetic energy per unit area of about 1.7 J/cm^2 which is about a 2% increase over the control run annual mean of 77 J/cm^2. This increase in kinetic energy can be explained by a close inspection of figure 4.4 or 4.11 below.

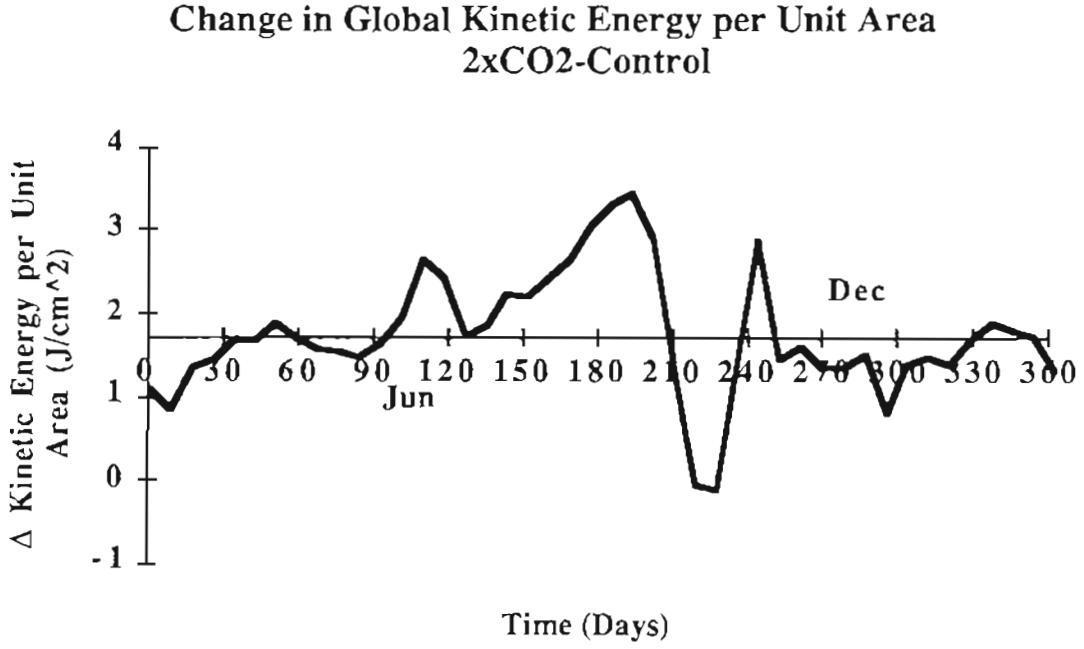

Figure 4.10. The yearly cycle of the predicted change in mean global kinetic energy per unit area. The calculated values are differences between the values in the last year of the $2xCO_2$ run and the last year of the control run. Units are Joules/cm^2.

The change in meridional temperature gradient resulting from the $2xCO_2$ experiment is shown in figure 4.11 for JJA and DJF. From this figure we can see that the magnitude of the meridional temperature gradient decreases in both hemispheres near the surface. This fact is also evident from figure 4.2. However, at higher altitudes the magnitude of the meridional temperature gradient typically increases. This same result is also suggested by figure 4.5, which shows the qualitative change in temperature field from Held's review article. That is, the maximum in the change of temperature predicted to occur in the mid to high level tropical troposphere results in an upper level temperature gradient increase. Since the upper atmosphere is not directly affected by surface drag, the increase in kinetic energy associated with the upper level temperature gradient increase is larger in magnitude than the decrease in kinetic energy that would be expected from the near surface temperature gradient reduction. This results in a net global kinetic energy increase due to a doubling of atmospheric carbon dioxide.

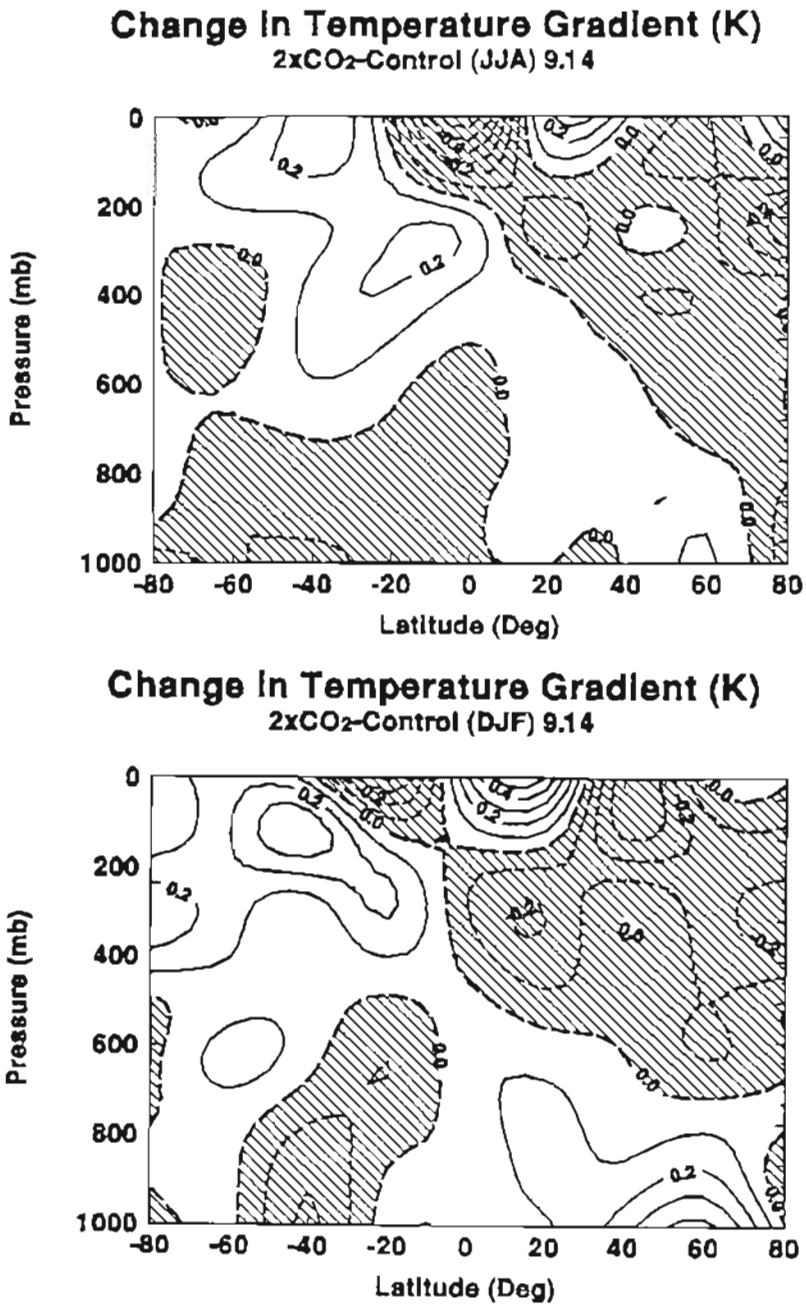

Figure 4.11. The change in Meridional temperature gradient for JJA (top) and DJF (bottom) due to the $2xCO_2$ experiment. In the above the units are K and can be converted to true gradient units by dividing be the product of the radius of the earth and 0.165 radians (9.44 deg).

The changes in the mass stream function for JJA and DJF resulting from the $2xCO_2$ experiment are shown in figure 4.12. Consistent with the increase in global kinetic energy, the Hadley circulation increases in both hemispheres by up to 3%. This certainly would have consequences in our model if we had a prognostic determination of cloud amount. Finally in figure 4.13 we show the $2xCO_2$ changes in the eddy flux of momentum (u'v') predicted by our model for JJA and DJF. In both hemispheres the magnitude of the poleward flux of westerly momentum increases in the mid latitude regions. Since the parameterized eddy momentum flux is highly dependent upon the temperature gradient, this increase in mid latitude eddy momentum transfer predicted by our model is a direct consequence of the meridional temperature gradient changes shown in figure 4.11.

This change in eddy momentum flux is also consistent with the changes in zonal velocity predicted by our model in figure 4.9. That is, the increase in u'v' flux in the mid latitude regions will result in more transfer of westerly momentum out of the tropics and into higher latitudes. This will result in an increase in the tropical easterly flow and a corresponding increase in high latitude westerly flow. Since in general our model's results are qualitatively consistent with the results of GCMs, we suspect that this same basic mechanism is responsible for the changes predicted by them.

Although Rosen and Gutowski (1992) stressed the weakness of the correlations between the changes in zonal velocity predicted by the GCMs of their study, they did state "Comparisons among the three models of the difference in zonal-mean zonal winds between $2xCO_2$ and $1xCO_2$ simulations indicate a common tendency when CO_2 is doubled for winds to become more easterly in much of the tropics during June-July-August." They were reluctant to propose any mechanism for the observed changes predicted by the GCMs. The mechanism described above is one possible mechanism driving the response of the GCMs.

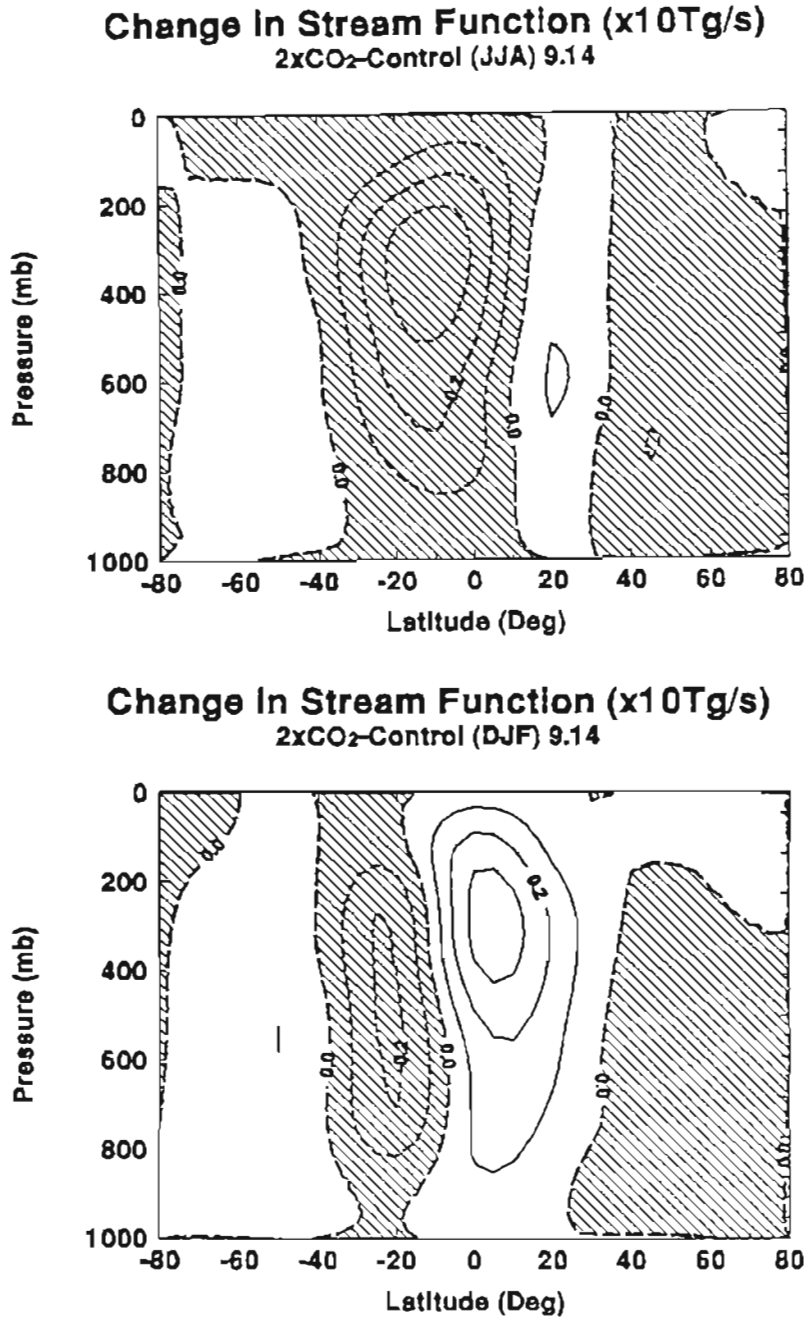

Figure 4.12. The change in the mass stream function predicted by our model for JJA (top) and DJF (bottom) averages resulting from the $2xCO_2$ experiment. Units are 10Tg/s.

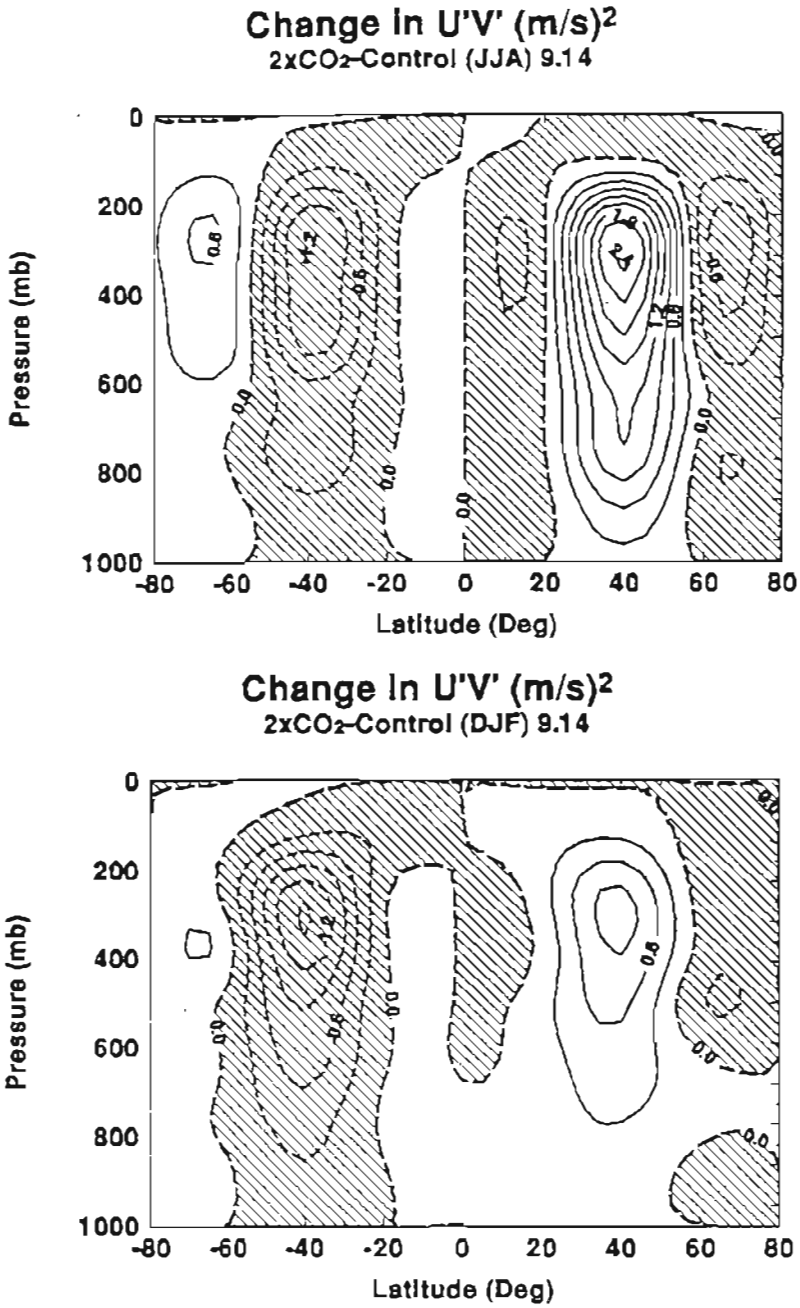

Figure 4.13. The change in the eddy momentum flux predicted by our model for JJA (top) and DJF (bottom) averages resulting from the $2xCO_2$ experiment. Units are m^2/s^2.

4.4 Summary.

The sensitivity of the GCRC 2D statistical dynamical climate model to the doubling of atmospheric carbon dioxide is investigated and we find that the change in temperature structure predicted by the model has several features that are also common to more sophisticated GCMs. In particular:

- The annual mean warming is greatest at the poles and smallest at the equator.
- There is a substantial cooling found in the stratosphere.
- The warming at the poles is greatest during the winter months for the particular hemisphere.
- There is a maximum in warming predicted to occur in the mid to upper tropical troposphere.

We suggest that the greater warming at the poles predicted by our model is likely the result of the enhanced upper level temperature gradients driving more energy from the equatorial regions towards the poles. The enhanced upper level temperature gradients are a direct result of the local maximum in temperature change found to occur in the upper tropical troposphere. This local maximum is believed to be associated with an increase in the vertical transport of latent heat energy. Since the change in temperature gradient is greatest in the winter hemisphere, this would also suggest a cause for the greater warming at the poles predicted for the winter months.

Following the work of Rosen and Gutowski (1992), we investigated the change in zonal mean zonal velocity predicted by our model for a doubling of CO_2. We find that our model predicts:

- An increase in easterly flow from about 35°S to 40°N.
- An increase in westerly flow from 60°S to 35°S and 40°N to 70°N.
- An increase in easterly flow for latitudes poleward of about 65°S and 75°N.
- The net angular momentum of the atmosphere remains relatively unchanged.

With our model we are able to recognize patterns of change in the zonal velocity for the $2xCO_2$ experiment that are consistent with the change in temperature structure of the atmosphere. We suggest that the increase in upper level temperature gradients

predicted by our model result in an increase in the simulated eddy flux of zonal momentum $\overline{u'v'}$, and that this causes more westerly momentum to be transferred from the tropics to the higher latitudes. In addition, we believe that the lack of a significant change in total atmospheric angular momentum is probably due to the lack of orography in our model and the relatively weak coupling between the atmosphere and the solid earth. Hence, our model atmosphere should conserve angular momentum.

We also attribute the 2% increase in global kinetic energy per unit are to the enhanced upper level temperature gradients as well as the 3% increase in the Hadley circulation.

CHAPTER 5

SUMMARY OF RESULTS

The GCRC two-dimensional statistical dynamical climate model presented in this dissertation has been developed using the one-dimensional radiative convective model of MacKay and Khalil (1991) as its basic core. Although the mid to late 1970s saw an abundance of two-dimensional climate models, their use in the 1980s and 1990s has been limited with few exceptions (MacCraken, 1987; and Stone and Yao,1990. There are two possible reasons for the apparent lack of interest in the use of two-dimensional climate model over the last decade. First, the tremendous advances in computing power and the increase in the sophistication of GCMs over the last several years has made their use very attractive for many climate modelers. Second, the difficulties encountered in attempting to parameterize the fluxes of eddy heat and momentum have been formidable. Recently however, Stone and Yao (1987) and (1990) following the work of others, have made substantial progress towards obtaining realistic parameterization schemes for calculating the eddy fluxes of heat and momentum. We have exploited their success by incorporating their parameterization scheme into the GCRC 2-D model.

The intent of this work was to develop a 2-D statistical dynamical climate model (base model) that can be used for a variety of climate simulations. In chapter one we give several arguments to support the usefulness of a suite of climate models of varying complexities for the study of climate change and the interactions between the various components of the climate system. It is also noted that the model presented in this dissertation is the most recent and most sophisticated addition to our climate modeling package at the Global Change Research Center. In Chapter two we describe our 2-D model in detail. The model has latitude and height as spatial coordinates and we use a numerical time stepping procedure to prognostically solve the primitive equations for zonal-mean temperatures, air density, zonal velocity, and meridional velocity. The

vertical velocity, pressure, and precipitation are calculated diagnostically. Cloud amounts, relative humidity profile, surface albedo, and the atmospheric concentrations of CO_2, CH_4, N_2O, O_3, $CFCl_3$, and CF_2Cl_2 have all been prescribed.

The model performance is evaluated in chapter 3, where we stress that it reproduces the two-dimensional seasonal temperature field of the earth to fairly well. Namely: 1) all surface temperatures predicted by the model are within 2% of observations except at the poles where there is a maximum discrepancy of 8%, 2) the annual mean two-dimensional temperature profile is within 5% of observations except at 80°S (8% difference) and the stratosphere where the model is about 10 % too warm. The good agreement between the modeled and observed temperatures at most positions is impressive considering the limited number of adjustable parameters that we have included in our model. It was suggested that improved boundary conditions in the model stratosphere for the meridional velocity or an improved estimate of the relative humidity of the tropical stratosphere may improve the simulation of the temperature minimum in the tropical stratosphere. The model's temperature gradient in the lower tropical troposphere seems to disagree somewhat with the observations there. We suspect that an over simplified parameterization of the cloud radiation physics and/or an unrealistic simulation of tropical convection is responsible for this difference between model and observations. The fact that the agreement between modeled and observed radiation balance of absorbed solar energy and outgoing long wave radiation is poor in the tropics for JJA and annual averages, also suggests that improvements in the model's tropical cloud radiative parameterization are needed. Also, we attribute a lack of model orography to be the major reason that the near surface temperatures in the Antarctic region simulated by the model are high in all seasons.

To improve the model's stability and performance, we have added an additional term to the Stone and Yao (1990) parameterization of the eddy flux of sensible heat, T'v', that depends on the square of the meridional temperature gradient (equation 2.43). We have found that the inability of their parameterization to simulate the secondary maximum of T'v' near the tropopause causes our model's polar winter stratospheric temperatures to be much lower than observed and the winter polar pressure gradients much larger than observed . It has been stressed that future improvements can be made in our parameterization scheme for the eddy flux of sensible heat.

The agreement between the model's zonal velocity field and observations is good. The magnitudes of the subtropical jet streams predicted by our model are within 25% of the observed values reported by Peixoto and Oort (1992) for JJA, DJF, and annual means. We do however note that the model overestimates the tropical easterlies, the vertical level of the model's jet stream is about 50 mb too high, and the latitudinal positions of the model's jet streams are typically a little too far poleward. We have also described a simple modification in the Stone and Yao (1987) parameterization scheme for the eddy momentum flux, $\overline{u'v'}$,which enhances the model's stability by reducing polar pressure gradients, focuses the jets closer to their observed positions at 35 degrees north and south latitude, and gives a better estimate of the vertical-mean u'v' flux in the polar regions. We find that this modification also reduces the need of the extra term of equation 2.43 for the T'v' flux. We find that our parameterization of $\overline{u'v'}$ does a good job at simulating the general features of the observed $\overline{u'v'}$ field.

Our model was able to give a good qualitative simulation of the mass stream function despite the fact that the maximum magnitude of the Hadley circulation was about 50 % of that observed. The seasonal movement of the ITCZ and seasonal variations in Hadley circulation strength also had good qualitative agreement with observations and we did obtain a weak Ferrel circulation.

Chapter 4 was devoted to exploring the equilibrium response of our 2-D model to a doubling of atmospheric carbon dioxide and comparing it to that of more sophisticated GCMs. The change in temperature profile predicted by our model is in excellent qualitative agreement with that typically simulated by GCMs in that our model simulated: 1) larger than average warming at the poles in winter months and not in the summer, 2) a maximum in the change in temperature simulated for the mid to upper tropical troposphere, and 3) stratospheric cooling. The change in mean annual global surface temperature for the $2xCO_2$ experiment for our model is about half of the IPCC (1990) best guess value of 2.5 K. We attribute our reduced sensitivity to the lack of positive feedback processes such as sea ice thickness, ice albedo, and cloud dynamics. Although we have experimented with coupling a simplified thermodynamic sea ice model to our atmospheric model and have obtained much greater polar warming than in our base model, we stress the importance of moving cautiously when adding positive feedback processes to a climate model. Small errors in assumptions and/or model physics of the

positive feedbacks may result in the model greatly over estimating the response of the real climate system. We plan to couple a sea ice model to the base model.

The good performance of the model at simulating the observed zonal velocity field has prompted us to follow the work of Rosen and Gutowski (1992) and explore the equilibrium response our model's zonal velocity field to a $2xCO_2$ experiment. Rosen and Gutowski (1992) investigated the response of three GCMs (NCAR, GISS, and GFDL) to the same experiment and found very few patterns that were similar between them. Because our model has much less inherent noise than GCMs, it is an ideal tool for investigating the basic mechanism behind the expected response of the real climate system and the observed response of the GCMs. We find that doubling CO_2 results in an increase in tropical easterly flow and an increase in the westerly flow at higher latitudes. The increase in tropical easterly flow in JJA was the only pattern that Rosen and Gutowski were willing to acknowledge as being consistent between the GCMs. We show that the change in zonal velocity predicted by our model for the $2xCO_2$ experiment is closely related to the change in temperature gradient simulated by our model which in turn drives the change in calculated $\overline{u'v'}$ flux. Since the change in the structure of the two-dimensional temperature gradient predicted by our model is qualitatively consistent with that simulated by most GCMs and the $\overline{u'v'}$ flux parameterization was designed by Stone and Yao (1987) primarily to agree with the $\overline{u'v'}$ flux of the GISS model, we conclude that the response of our model is indicative of the basic response expected of the GCMs and possibly the real earth. Of course there are other factors that complicate the output of the GCMs, but this is precisely the reason that a model of the sophistication of our two-dimensional model is so attractive.

In summary the model presented in this dissertation is a good solid working two-dimensional model of the earth-atmosphere climate system. Its has been intentionally designed modularly to aid in the implementation of future improvements. Although, as shown here, the present version of the model can be used for legitimate scientific investigations, the real pay-off for the development of this model will come when we begin implementing improvements into the model, such as (but not limited to) a deep ocean model, cloud dynamics and radiative properties, a prognostic sea ice model, prognostic hydrodynamics, and improvements in the calculation of the eddy fluxes of

heat and momentum. With each addition the model's performance should improve and the affect of each component of the whole can be carefully explored.

REFERENCES

Branscome, L.E., 1983a: A Parameterization of transient Eddy Heat Flux on a Beta-Plane. J. of the Atmospheric Sciences, Vol. 40, 2508-2521.

Branscome, L.E., 1983b: The Charney Baroclinic Stability Problem: Approximate Solutions and Modal Structures. Journal. of the Atmospheric Sciences, Vol. 40, 1393-1409.

Budyko, M.I., 1969. The Effect of Solar Radiation Variations on the Climate of the Earth. Tellus, 21, 611-619.

Charney, J.G., 1947. The Dynamics of Long Waves in a Baroclinic Westerly Current. Journal of Meteorology, Vol. 4, No. 5, 135-163.

Donner, L., and V. Ramanathan, 1980. Methane and Nitrous Oxide: Their effects on the Terrestrial Climate. Journal Of Atmospheric Sciences, Vol. 37, 119-124.

Eady, E.T., 1949. Long Waves and Cyclone Waves. Tellus, Vol. 1, No. 3, 33-52.

Eagleman, J.R., 1985. Meteorology: The Atmosphere in Action. Wadsworth Publishing, Belmont California.

Forrester, F. H., 1981. 1001 Questions Answered About the Weather. Dover Publications, New York.

Ghan, S.J., J.W. Lingaas, M.E. Schlesinger, R.L. Mobley, and W.L. Gates, 1982. A Documentation of the OSU Two-Level Atmospheric General Circulation Model. Climatic Research Institute Report No. 35, Oregon State University, Corvallis, Oregon .

Goldstein, H. 1982. Classical Mechanics Second Edition (chapter 3). Addison Wesley Publishers, Reading Mass.

Goody, R.M. and J.C.G. Walker, 1972. Atmospheres. Prentice-Hall, Englewood Cliffs, N.J.

Green, J.S.A., 1970. Transfer Properties of the Large-Scale Eddies and the General Circulation of the Atmosphere. Quarterly Journal of the Royal Meteorological Society. Vol. 96, No. 408, 157-185.

Hahn, C.J., S.G. Warren, J. London, R.L. Jenne and R.M. Chervin, 1987: Climatological Data for Clouds over the Globe from Surface Observations, Report NDP-026, Carbon Dioxide Information Center, Oak Ridge, TN 37831-6050

Hansen, J., G. Russell, D. Rind, P. Stone, A. Lacis, S. Lebedeff, R. Ruedy and L. Travis, 1983. Efficient Three-Dimensional Global Models for Climate Studies: Models I and II. Monthly Weather Review, Vol. 111, No. 4, 609-662.

Harvey, L.D.D., 1988. Development of a Sea Ice Model for Use in Zonally Averaged Energy Balance Climate Models. Journal of Climate, Vol. 1, 1221-1238.

Held, I.M., 1978. The Vertical Scale of an Unstable Baroclinic Wave and Its Importance fo Eddy Heat Flux Parameterizations. Journal of the Atmospheric Sciences, Vol 35, 572-576.

Held, I.M. and M.J. Suarez, 1978. A Two-Level Primitiv Equation Atmospheric Model Designed for Climate Sensitivity Experiments. Journal of the Atmospheric Sciences, Vol 35, 206-229.

Held, I.M., 1993. Large-Scale Dynamics and Global Warming. Bulletin of the American Meteorological Society. Vol. 74, 228-241.

Henderson-Sellers, A. and K. McGuffie. 1987. A Climate Modeling Primer. John Wiley & Sons, New York.

Holton, J.R., 1979. An Introduction to Dynamic Meteorology, Second Edition. International Geophysics Series Vol. 23. Academic Press, Inc. Harcourt Brace Jovanovich, Publishers, New York.

Hunt, B.G., 1973, Zonally Symmetric Global General Circulation Models With and Without the Hydrologic Cycle. Tellus XXV, 4, 337-354.

IPCC, 1990. Climati Change, The IPCC Scientific Assessment. J.T. Houghton, G.J. Jenkins, and J.J. Ephraums Editors. Cambridge University Press, Cambridge U.K.

Karl, T.R., J.D. Tarpley, R.G. Quayle, H.F. Diaz, D.A. Robinson, and R.S. Bradley, 1989. The Recent Climate Record: What it Can and Cannot Tell Us. Reviews of Geophysics, 27, 3, 405-430

Kasahara, A., 1977. Computational Aspects of Numerical Models for Weather Prediction and Climate Simulation. In Methods in Computational Physics Volume 17 J. Chang Editor. Academic Press, New York.

Khalil, M.A.K. and R.A. Rasmussen, 1985. Modeling Chemical Transport and Mass Balances in the Atmosphere. Chapter 2 in Environmental Exposure From Chemicals Volume II W.B. Neely and G.E. Blau Editors. CRC Press, Inc. Boca Raton, Florida.

Kiehl, J.T. and V. Ramanathan, 1983. CO2 Radiative Parameterization Used in Climate Models: Comparison With Narrow Band Models and With Laboratory Data. Journal of Geophysical Research, Vol 88, No. C9, 5191-5202

Kuo, H.L., 1977. Analytic Infared Transmissivities of the Atmosphere. Beitr. Phys. Atmos., Vol. 50, 331-349.

Lacis, A.A., and J.E. Hansen, 1974. A Parameterization for the absorption of Solar Radiation in the Earth's Atmosphere. Journal Of Atmospheric sciences, Vol. 31, 118-133.

Liu, S.C., J.R. McAfee, and R.J. Cicerone, 1984. Radon 222 and Tropospheric Vertical Transport. J. of Geophysical Research, Vol. 89, No. D5, 7291-7297.

Liou, Kuo-Nan. 1980. An Introduction to Atmospheric Radiation.
Academic Press.

Lorentz, E.N., 1967. The Nature and Theory of the General Circulation of the Atmosphere. Published by The World Meteorological Organization.

MacCracken, M.C., 1972. Zonal Atmospheric Model ZAM2. Lawrence Livermore National Lab. UCRL-74254 UNCLAS

MacCracken, M. C. and S. Ghan. 1987. Design and Use of Zonally-Averaged Climate Models, UCRL-94338, University of California, Livermore CA, 44pp.\

MacKay, R.M., 1990. The Oregon Graduate Institute One-Dimensional Time-Dependent Radiative Convective Model: Theory and Application. M.S. Thesis, The Oregon Graduate Institute of Science & Technology, Department of Environmental Science and Engineering, Global Change Research Center, Beaverton, Oregon.

MacKay, R.M., and M.A.K. Khalil, 1991. Theory and Development of a One-Dimensional Time-Dependent Radiative Convective Climate Model. Chemosphere, Vol. 22. No. 3-4 318-417.

MacKay, R.M. and M.A.K. Khalil, 1992. Global Change Science and Education. Paper presented at the 1992 Aspen Global Change Institute, Aspen Colorado.

MacKay, R.M. and M.A.K. Khalil, 1993. Computer Based Activities for Understanding Global Change. In the Proceedings of the 1st International Conference on Computer-aided Learning and Distance Learning in Meteorology, Hydrology, and Oceanography (CALMet).

Manabe, S. and R.F. Strickler, 1964. Thermal Equilibrium of the Atmosphere with a Convective Adjustment. Journal Of Atmospheric sciences, Vol. 21, 361-385

Manabe, S. and R.T. Wetherald, 1967. Thermal Equilibrium of the Atmosphere with a given Distribution of Relative Humidity. Journal of Atmospheric Sciences, Vol. 24, No.3, 241-259

Manabe, S. and K. Bryan. 1969. Climate Calculations with a Combined Ocean Atmosphere Model. Journal of the Atmospheric Sciences, Vol. 26, 786-789.

Mitchell, John F.B., 1989. The "Greenhouse" Effect and Climate Change. Reviews of Geophysics Vol. 27 No. 1 115-139.

North, G.R., 1975. Theory of Energy Balance Climate Models. Journal of the Atmospheric Sciences, Vol. 32, 3-15

Newell, R.E., J.W. Kidson, D.G. Vincent, and G.J. Boer, 1972. The General Circulation of the Tropical Atmosphere and Interactions with Extratropical Latitudes Vol.1. MIT Press, Cambridge, Mass.

Ohring, G. and S. Adler. 1978. Some Experiments with a Zonally Averaged Climate Model. Journal of the Atmospheric Sciences, Vol. 35, 186-205

Parkinson, C.L. and W.M. Washington, 1979. A Large-Scale Numerical Model of Sea Ice. Journal of Geophysical Research, Vol. 84, No. C1, 311-337.

Pedlosky, J. 1979. Geophysical Fluid Dynamics. Springer-Verlag Publishers, New York.

Peixoto, J. P. and A.H. Oort, 1992. Physics of Climate. American Institue of Physics.

Peng, L., M-D Chou, and A. Arking, 1982. Climate Studies with a Multilayer Energy Balance Model. Part I: Model Description and Sensitivity to the solar Constant. Journal of the Atmospheric Sciences, Vol. 39, 2639-2656.

Phillips, N. A., 1954. Energy Transformations and Meridional Circulations Associated With Simple Baroclinic Waves in a Two-level, Quasi-geostrophic Model. Tellus Vol. 6, pp 273-286.

Ramanathan, V. 1975. Greenhouse Effect Due to Chlorofluorocarbons: Climatic Implications. Science Vol. 190 50-51.

Ramanathan, V., 1976. Radiative Transfer Within the Earth's Troposphere and Stratosphere: A simplified Radiative-Convective Model. Journal of Atmospheric Sciences, Vol. 33, 1330-1346.

Ramanathan, V., 1977. Interactions between Ice-Albedo, Lapse-Rate and Cloud-Top Feedbacks: An Analysis of the Nonlinear Response of a GCM Climate model. Journal of Atmospheric Sciences, Vol. 34, 1885-1897.

Ramanathan, V., and J.A. Coakley Jr., 1978. Climate Modeling Through Radiative Convective Models. Rev. of Geophysics and Space Physics, 6, 465-489.

Ramanathan, V., E.J.Pitcher, R.C. Malone, and M.L. Blackmon, 1983. The Response of a Spectral General Circulation Model to refinements in Radiative Processes.Journal of Atmospheric sciences, Vol. 40, 605-631.

Ramanathan, V., R.J. Cicerone, H.B. Singh, and J.T. Kiehl, 1985. Trace Gas Trends and Their Potential Role in Climatic Change. Journal of Geophysical Research, Vol. 90, 5547-5566.

Rasool, S.I. and S.H. Schneider, 1971. Atmospheric Carbon Dioxide and Aerosols: Effects of Large Increases on Global Climate. Science, Vol. 173, 138-141

Richardson, L. F., 1922. Weather Prediction by Numerical Process, Cambridge University Press, reprinted Dover, New York, 236 pp.

Roberts, R.E., J.E.A. Selby, and L.M. Biberman, 1976. Infared Continuum Absorption by Atmospheric Water Vapor in the 8-12 μm Window. Applied Optics, Vol.15, No. 9, 2085-2090.

Rosen, R. D. and W. J. Gutowski Jr., 1992. Response to Zonal Winds and Atmospheric Angular Momentum to a doubling of CO2. Journal of Climate, Vol. 5, No. 12, 1391-1404

Rosen, R.D., D.A. Salstein, and T.M. Wood, 1990. Discrepancies in the Earthe-Atmosphere Angular Momentum Budget, Journal of Geophysical Research, Vol. 95, No. B1, 265-279.

Rosen, R.D., D.A. Salstein, and T.M. Wood, 1991. Zonal Contributions to Global Momentum Variations on Intraseasonal Through Interannual Time Scales, Journal of Geophysical Research, Vol. 96, No. D3, 5145-5151.

Saltzman, B., 1978. A Survey of Statistical-Dynamical Models of the Terrestrial Climate. In Advances in Geophysics Vol. 20 Academic Press, New York.

Sasamori, T., J. London, and D.V. Hoyt, 1972. Radiation Budget of the Southern Hemisphere. Meteorological Monographs, Vol.13, No.35.9-23

Schlesinger, M.E. and J.F.B. Mitchell, 1985. Model Projections of the Equilibrium Climatic Response to Increased Carbon Dioxide. Chapter 4 of The Potetial Climatic Effects of Increasing Carbon Dioxide. DOE/ER-0237.

Schlesinger, M.E. and J.F.B. Mitchell, 1987. Climate Model Simulations of the Equilibrium Climatic Response to Increased Carbon Dioxide. Reviews of Geophysics, Vol. 25, No.4, 760-798.

Schneider, E.K. and R.S. Lindzen, 1977. Axially Symmetric Steady-State Models of the Basic State for Instability and Climate Studies. Part I. Linearized Calculations. Journal of the Atmospheric Sciences, Vol. 34, 263-279.

Schneider, E. K., 1984. Response of the Annual and Zonal Mean Winds and Temperatures to Variations in the Heat and Momentum Sources. Journal of the Atmospheric Sciences, Vol. 41, 1093-1115.

Schneider, S. H., 1989. Global Warming: Are We Entering the Greenhouse Century? Sierra Club Books, San Francisco, Calif.

Schoeberl, M. R. and D. F. Strobel, 1978. The Zonall Averaged Circulation of the Middle Atmosphere. Journal of the Atmospheric Sciences, Vol. 35, 577-591.

Sellers, W.D., 1973. A New Global Climate Model, Journal of Applied Meteorology, Vol. 12, 241-254.

Semtner, A.J. , 1976. A Model for the thermodynamic Growth of Sea Ice in Numerical Investigations of Climate. Journal of Physical Oceanography, ^ pp379-389.

Shapiro, R., 1970. Smoothing, Filtering, and Boundary Effects. Reviews of Geophysics and Space Physics, Vol. 8, No. 2, 359-387

Smagorinsky, J., 1983. The Beginnings of Numerical Weather Prediction and General Circulation Modeling: Early Recollections. In B. Saltzmann (ed.) Theory of Climate, Academic Press, New York, pp3-38.

Stephens, G.L., 1984. The Parameterization of Radiation for Numerical Weather Prediction and Climate Models. Monthly Weather Review, Vol. 112, 826-867.

Stone, P. H., 1972. The Effect of Large-Scale Eddies on Clamatic Change. Journal of the Atmospheric Sciences, Vol. 30, 521-529

Stone, P. H., 1973. A Simplified Radiative-Dynamical Model for the Static Stability of Rotating Atmospheres. Journal of the Atmospheric Sciences, Vol. 29, 405-418.

Stone, P. H., 1978. Baroclinic Adjustment. Journal of the Atmospheric Sciences, Vol. 35, 561-571.

Stone, P.H., and J.H. Carlson, 1979. Atmospheric Lapse Rate Regimes and Their Parameterization. Journal of Atmospheric Sciences, Vol. 36, 415-423.

Stone, P.H., and M.-S. Yao. 1987. Development of a Two-Dimensional Zonally Averaged Statistical-Dynamical Model. Part II: The Role of Eddy Momentum Fluxes in the General Circulation and their Parameterization. Journal of the Atmospheric Sciences, Vol. 44, No. 24, 3769-3786.

Stone, P.H., and M.-S. Yao. 1990. Development of a Two-Dimensional Zonally Averaged Statistical-Dynamical Model. Part III: The Parameterization of Eddy Fluxes of Heat and Moisture. Journal of Climate Volume 3, 726-740.

Stone, P.H., 1992. Forecast Cloudy: The Limits of Global Warming Models. Technology Review, Feb/Mar 1992, pp. 32-40.

Temkin, R. L. and F.M. Snell, 1976. An Annual Zonally Averaged Hemispherical Climate Model with Diffuse Cloudiness Feedback. Journal of the Atmospheric Sciences, Vol. 33, 1671-1685.

Vernekar, A. D. and H. D. Chang, 1978. A Statistical-Dynamical Model for Stationary Perturbations in the Atmosphere. Journal of the Atmospheric Sciences, Vol. 35, 433-444.

Wang, W-C, M.P. Dudek, X-Z. Liang, and J.T. Kiehl, 1990a. Inadequacy of Effective CO2 as a Proxy in Simulating the Greenhouse Effect of Other Radiatively Active Gases. Nature, Vol. 350, pp573-577.

Wang, W.-C., G. Molnar, M.K.W. KO, S. Goldenberg, and N.D. Sze, 1990b. Atmospheric Trace Gases and Global Climate: A Seasonal Model Study. Tellus, 42B, 2, 149-161.

Washington, W.M., and C.L. Parkinson, 1986. An Introduction to Three-Dimensional Climate Modeling. University Science Books.

Watts, R.g. and M. Morantine, 1990. Rapid Climatic Change and the Deep Ocean. Climatic Change, Vol. 16, 83-97.

Willmott, C. J. and D. R. Legates, 1993. A Comparison of GCM-Simulated and Observed Mean January and Jully Surface Air Temperature. Journal of Climate, Vol. 6, No. 2, 274-291

Yao, M.-S., and P.H. Stone, 1987. Development of a Two-Dimensional Zonally Averaged Statistical-Dynamical Model. Part I: The Parameterization of Moist Convection and its Role in the General Circulation. Journal of the Atmospheric Sciences, Vol. 44, No. 1, 65-82.

APPENDIX A
LIST OF SYMBOLS

Q_S	Daily average flux of incident solar radiation per unit surface area,
S(t)	Tme dependent solar constant at the top of the atmosphere
H	Half day length in hours
r(t)	Earth-sun distance at time t
r_m	Mean Earth-Sun distance 1.49×10^{11}m
δ	Solar declination
ϕ	Latitude
μ	Cosine of daily average solar zenith angle
α_L	Snow-free land surface albedo
α_L*	Snow-covered land surface albedo
ξ	% Ocen fraction
α_O	Ocean surface albedo
x	1-μ
V_S	Surface wind speed
X_I	Mean sea ice thickness
C_w	Specific heat of water 4.19 J/K/g
C_I	Specific heat of ice 2.0 J/K/g
D	Depth of the mixed layer (typically 50 m)
H_S	Snow thickness over sea ice
ρ_w	Density of water (1.0 g/cm^3)
ρ_I	Density of ice (0.9 g/cm^3)
T_O	Ocean temperature,
$F\downarrow$	Downward flux of IR radiation at the Earth's surface from the atmosphere
σ	5.67×10^{-8}W/m^2/K^4
$S\downarrow$	Downward flux of solar radiation at the Earth's surface

$S\uparrow$	Upward flux of solar radiation at the Earth's surface
SH	Flux of sensible heat leaving the Earth's surface
LH	Flux of latent heat leaving the Earth's surface
A_O	Surface area of the ocean
A_I	Area of sea ice
K_I	Thermal conductivity ice (2.2 J/K/m)
K_s	Thermal conductivity snow (.3 J/K/m)
T_I	Temperature of the top of the sea ice
a	Radius of the earth
K_h	Horizontal turbulent difusivity constant for ocean heat transport
T_L	Land surface temperature
χ	Fraction of ocean covered by sea ice
λ	Longitude
z	vertical height
p	zonal average pressure
ρ	zonal average air density
R	gas constant (287 J/(g K) for dry air
T	zonal average atmospheric temperature
f	coriolis parameter
w	vertical velocity
u	zonal velocity
v	meridional velocity
t	Time
Δt	time step for dynamical numerical integration (0.25 hr)
Δt_r	time step for radiation numerical integration (2.0 hr)
C^*	Rate of condensation
E^*	Evaporation rate
i	Vertical center of model grid
i+1/2	Bottom boundary of model grid i
j	Horizontal center of model grid
j+1/2	Southern boundary of model grid j

$g_{j+1/2}$	$\left(\frac{\rho_j + \rho_{j+1}}{2}\right) v_{j+1/2} \cos\phi_{j+1/2}$
K_Z	Vertical eddy diffusivity
τ_ϕ	Surface shear stress in the meridional direction
τ_λ	Surface shear stress in the longitudinal direction
F_X	frictional force per unit mass in x ($\phi or \lambda$) direction
C_D	Surface drag coefficient (taken as 0.001)
C_p	specific heat capacity of air at constant pressure
L_v	Latent heat of vaporization
Lf	Latent heat of fusion for ice (330 J/g)
Q	Heating rate per unit mass
κ	R/C_p
α	Specific volume per unit mass ($1/\rho$)
C_v	Specific heat capacity at constant volume for dry air (5/2R)
γ	C_p/C_v
g	the acceleration of gravity (9.8 m/s^2)
$\Delta m_{i+\frac{1}{2}}$	mass transport from layer i+1 to layer i during hydrostatic adjustment
s	specific humidity
s^*	Saturation specific humidity
r	Relative humidity
r_o	Surface relative humidity
ς	Relative vorticity $\left[\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\right]$
y	Latitude position, a ϕ
β_o	$\left.\frac{\partial f}{\partial y}\right\|_{y=y_o}$
δp	First order perturbation of pressure from the stationary state
$\aleph_S$	An arbitrary variable of the stationary state of the atmosphere $\aleph$ =(density, temperature, pressure,...)
θ	potential temperature, $T\left(\frac{P_o}{P}\right)^\kappa$

N^2	Square of the *Brunt – Väisälä* frequency $\frac{g}{\theta}\frac{d\theta}{dz}$
Ψ	Velocity stream function, $\left[\varsigma + \beta_o y + \frac{f_o}{\rho_s}\frac{\partial}{\partial z}\left(\frac{\rho_s}{N_s^2}\frac{\partial}{\partial z}\frac{\delta p}{\rho_s}\right)\right]$
J(A,B)	Jacobian of A and B, $\frac{\partial A}{\partial x}\frac{\partial B}{\partial y} - \frac{\partial A}{\partial y}\frac{\partial B}{\partial x}$
H	Scale height of the atmosphere
σ	Stability parameter $\frac{\partial \theta}{\partial p}$
γ_d	Dry adiabatic lapse rate (9.8 K/km)
γ_m	Moist adiabatic lapse rate
K_{hs}	Horizontal diffusivity of heat in ocean mixed layer (=1.2x10^{12} m^2/yr)

APPENDIX B

CLOUD AND OZONE DATA USED IN THE GCRC 2D MODEL

B.1. Clouds

B.1.1 High Clouds

Table B.1. High cloud fraction (%) used in the GCRC 2D model for each season. Values were interpolated from NCAR cloud data set; See Hahn et al. (1988).

Latitude (Deg)	Season			
	MAM	JJA	SON	DJF
-80.3	22.3	20.2	37.2	28.1
-70.8	22.9	20.4	26.6	23.2
-61.4	24.0	23.8	28.1	18.8
-51.9	17.6	14.0	17.5	18.3
-42.5	14.8	13.4	15.7	16.5
-33.1	11.7	10.4	11.6	10.3
-23.6	9.9	7.0	9.3	11.4
-14.2	12.4	6.7	10.0	16.7
-4.7	16.4	12.1	13.6	16.8
4.7	21.6	23.0	20.9	19.5
14.2	15.4	24.1	19.6	12.8
23.6	12.5	16.2	11.3	8.9
33.1	20.6	14.0	14.0	16.8
42.5	20.8	15.5	16.1	18.0
51.9	24.5	20.5	22.5	24.0
61.4	25.7	21.1	24.1	25.2
70.8	21.3	18.8	18.0	17.7
80.3	29.2	26.4	23.0	16.8

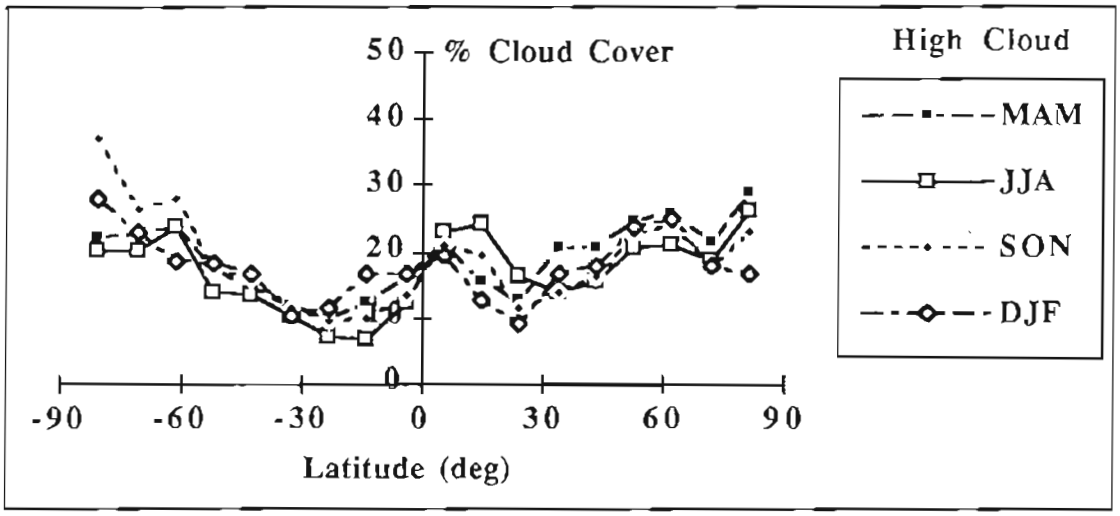

Figure B.1. Graphical display of high cloud fraction (%) from sata in Table B.1.

B.1.2. Middle Clouds

Table B.2. Middle cloud fraction (%) used in the GCRC 2D Model for each season. Values were interpolated from NCAR cloud data set obtained from Roy Jenny. See Hahn et al. 1988.

Latitude (Deg)		Season		
	MAM	JJA	SON	DJF
-80.3	19.4	15.1	20.3	30.5
-70.8	38.2	31.0	32.9	38.5
-61.4	47.5	45.4	39.9	49.7
-51.9	40.7	31.8	29.0	46.6
-42.5	28.5	27.7	27.7	32.1
-33.1	25.2	23.3	25.5	23.5
-23.6	20.6	17.9	20.5	22.7
-14.2	21.3	16.5	19.6	28.2
-4.7	28.2	23.5	25.3	32.0
4.7	31.3	39.8	36.7	30.2
14.2	15.3	34.0	26.2	15.6
23.6	22.4	23.4	21.5	21.8
33.1	27.5	22.7	24.6	28.0

42.5	31.7	27.7	29.4	33.3
51.9	32.6	33.5	36.4	36.2
61.4	34.3	35.4	42.2	39.7
70.8	32.5	37.4	41.0	33.5
80.3	25.6	36.7	37.5	26.9

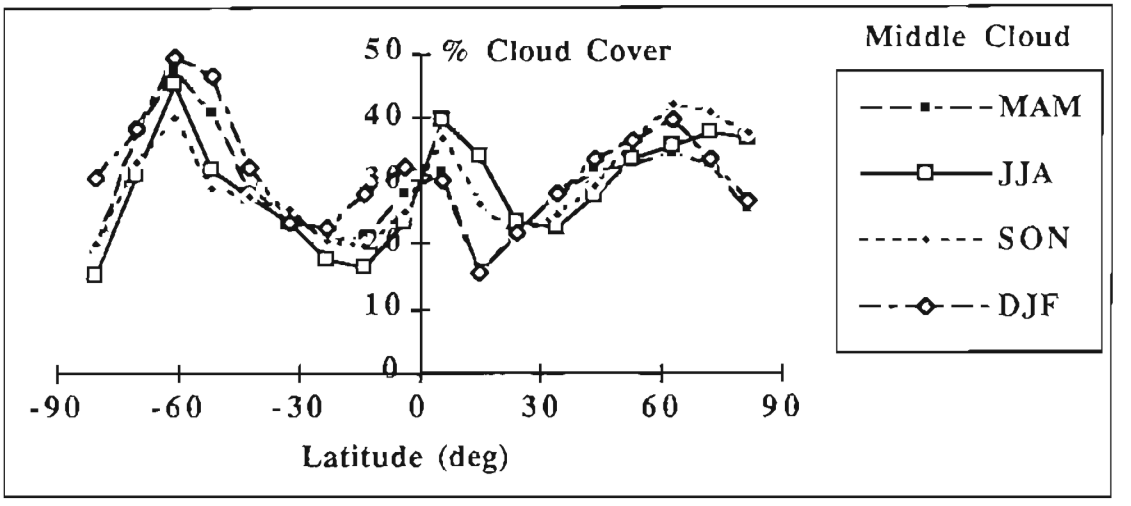

Figure B.2. Graphical display of middle cloud fraction (%) from sata inTable B.2

B.1.3 Low Cloud

Table B3. Low cloud fraction (%) used in the GCRC 2D Model for each season. Values were interpolated from NCAR cloud data set obtained from Roy Jenny. See Hahn et al. 1988.

Latitude (Deg)		Season		
Low	MAM	JJA	SON	DJF
-80.3	11.0	7.5	9.7	12.6
-70.8	33.2	30.9	28.5	36.0
-61.4	64.9	66.0	67.2	64.7
-51.9	68.6	63.6	69.2	70.0
-42.5	58.5	57.0	55.1	57.2
-33.1	47.6	47.5	48.2	46.6
-23.6	40.9	40.2	42.8	42.5
-14.2	41.6	43.5	45.5	44.2

-4.7	40.2	41.4	43.6	41.3
4.7	40.0	45.9	45.5	39.9
14.2	29.8	40.2	36.3	30.7
23.6	40.6	39.6	37.7	40.9
33.1	39.3	36.3	36.1	42.0
42.5	45.0	45.5	43.6	47.7
51.9	46.4	50.8	50.7	49.0
61.4	39.9	49.1	50.4	39.7
70.8	47.3	62.5	57.9	39.6
80.3	28.0	64.2	44.2	19.0

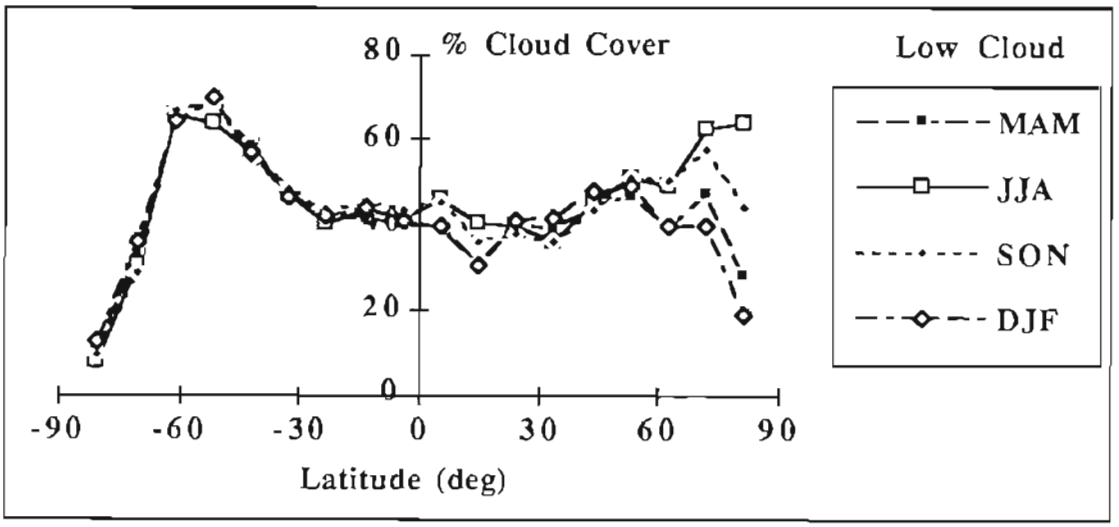

Figure B.3. Graphical display of low cloud fraction (%) from sata inTable B.3.

Table B.4. Reduced ozone data from LLNL 2-D ozone photochemical model. From January through December 1990. Each row corresponds to the ozone (cm of O_3 at STP) amount down to the vertical boundary between adjacent model layers. (From top row down: 40, 20, 14, 10, 8, 6, 4, 2, 0.2, and 0.0 km) Each column corresponds to a model grid (zone) of latitude. From left to right approximate grid horizontal centers are 80.3° S, 70.8°S,...0..., 70.8° N, 80.3* N. Data were interpolated from raw values to fit our model grid.

Lat	-80.3	-70.8	-61.4	-51.9	-42.5	-33.1	-23.6	-14.2	-4.8	4.8	14.2	23.6	33.1	42.5	51.9	61.4	70.8	80.3
Ht (km)									**Jan**									
40	0.008	0.008	0.008	0.009	0.010	0.010	0.010	0.010	0.011	0.011	0.010	0.010	0.009	0.008	0.008	0.007	0.007	0.007
20	0.282	0.279	0.268	0.258	0.254	0.251	0.246	0.241	0.239	0.249	0.256	0.258	0.254	0.242	0.221	0.195	0.177	0.174
14	0.439	0.433	0.408	0.368	0.336	0.315	0.295	0.275	0.261	0.272	0.287	0.298	0.302	0.299	0.292	0.274	0.257	0.255
10	0.486	0.480	0.447	0.398	0.357	0.331	0.307	0.283	0.267	0.278	0.294	0.307	0.312	0.311	0.308	0.291	0.275	0.272
8	0.499	0.492	0.458	0.408	0.366	0.339	0.313	0.287	0.271	0.281	0.297	0.311	0.317	0.316	0.313	0.296	0.280	0.277
6	0.509	0.502	0.468	0.417	0.375	0.347	0.319	0.292	0.274	0.284	0.301	0.316	0.322	0.321	0.317	0.300	0.284	0.281
4	0.519	0.512	0.478	0.426	0.384	0.354	0.326	0.297	0.277	0.287	0.304	0.319	0.326	0.325	0.321	0.303	0.288	0.285
2	0.526	0.519	0.485	0.433	0.390	0.360	0.331	0.301	0.280	0.289	0.307	0.323	0.329	0.328	0.324	0.306	0.290	0.288
0.2	0.531	0.523	0.489	0.437	0.394	0.365	0.336	0.305	0.284	0.292	0.310	0.326	0.333	0.331	0.326	0.308	0.293	0.290
0.0	0.531	0.524	0.490	0.438	0.395	0.365	0.336	0.305	0.284	0.293	0.311	0.326	0.333	0.332	0.326	0.309	0.293	0.290
									Feb									
40	0.008	0.008	0.009	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.009	0.009	0.009	0.008	0.008
20	0.276	0.275	0.271	0.265	0.260	0.256	0.250	0.243	0.239	0.246	0.252	0.255	0.252	0.242	0.224	0.203	0.189	0.186
14	0.460	0.456	0.434	0.393	0.349	0.323	0.301	0.278	0.262	0.268	0.282	0.293	0.297	0.293	0.286	0.278	0.267	0.264
10	0.518	0.512	0.481	0.427	0.373	0.340	0.313	0.286	0.268	0.273	0.288	0.301	0.307	0.305	0.301	0.296	0.286	0.284
8	0.532	0.526	0.494	0.438	0.383	0.349	0.320	0.290	0.271	0.276	0.291	0.305	0.312	0.310	0.306	0.301	0.291	0.289
6	0.543	0.537	0.505	0.449	0.392	0.357	0.327	0.296	0.274	0.279	0.294	0.309	0.316	0.315	0.310	0.305	0.296	0.293
4	0.554	0.547	0.515	0.458	0.401	0.365	0.333	0.301	0.278	0.281	0.297	0.313	0.320	0.319	0.314	0.309	0.299	0.297
2	0.562	0.556	0.523	0.466	0.408	0.372	0.339	0.305	0.281	0.284	0.300	0.316	0.323	0.322	0.317	0.312	0.302	0.300
0.2	0.567	0.560	0.527	0.470	0.413	0.376	0.344	0.309	0.284	0.287	0.303	0.319	0.326	0.325	0.319	0.314	0.304	0.302

0.0	0.567	0.561	0.528	0.471	0.413	0.377	0.344	0.310	0.285	0.287	0.303	0.319	0.327	0.325	0.320	0.314	0.305	0.302
									Mar									
40	0.007	0.008	0.009	0.009	0.009	0.009	0.009	0.009	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.011	0.010	0.010
20	0.269	0.271	0.272	0.267	0.264	0.262	0.257	0.249	0.244	0.246	0.251	0.254	0.251	0.242	0.229	0.215	0.203	0.202
14	0.462	0.461	0.445	0.401	0.356	0.331	0.309	0.284	0.267	0.268	0.279	0.291	0.295	0.292	0.290	0.289	0.285	0.284
10	0.524	0.522	0.497	0.439	0.380	0.348	0.321	0.293	0.273	0.273	0.285	0.299	0.305	0.305	0.306	0.309	0.307	0.306
8	0.539	0.537	0.509	0.450	0.391	0.357	0.328	0.298	0.276	0.275	0.288	0.303	0.310	0.310	0.311	0.314	0.312	0.312
6	0.550	0.548	0.520	0.460	0.401	0.366	0.336	0.303	0.280	0.278	0.292	0.307	0.314	0.314	0.315	0.318	0.317	0.316
4	0.560	0.558	0.530	0.470	0.409	0.374	0.342	0.308	0.283	0.281	0.295	0.310	0.318	0.318	0.319	0.322	0.320	0.320
2	0.568	0.566	0.538	0.477	0.417	0.381	0.348	0.313	0.286	0.283	0.297	0.314	0.322	0.321	0.322	0.325	0.323	0.323
0.2	0.573	0.571	0.543	0.482	0.422	0.386	0.353	0.317	0.290	0.286	0.300	0.317	0.325	0.324	0.324	0.327	0.325	0.325
0.0	0.574	0.572	0.543	0.483	0.422	0.386	0.354	0.317	0.290	0.286	0.301	0.317	0.325	0.325	0.324	0.327	0.326	0.325
									Apr									
40	0.007	0.008	0.008	0.008	0.009	0.009	0.009	0.010	0.010	0.010	0.010	0.011	0.011	0.012	0.013	0.012	0.010	0.009
20	0.255	0.257	0.263	0.265	0.266	0.266	0.261	0.253	0.246	0.244	0.247	0.250	0.250	0.248	0.243	0.232	0.225	0.225
14	0.429	0.430	0.417	0.389	0.359	0.337	0.315	0.290	0.269	0.265	0.275	0.287	0.296	0.301	0.309	0.314	0.318	0.319
10	0.489	0.489	0.469	0.426	0.384	0.356	0.328	0.298	0.275	0.270	0.282	0.296	0.306	0.315	0.326	0.337	0.345	0.347
8	0.502	0.503	0.481	0.436	0.395	0.365	0.335	0.304	0.278	0.273	0.285	0.300	0.311	0.321	0.332	0.344	0.352	0.354
6	0.512	0.513	0.491	0.446	0.404	0.373	0.342	0.309	0.282	0.276	0.289	0.304	0.316	0.326	0.337	0.349	0.357	0.359
4	0.522	0.522	0.500	0.455	0.413	0.381	0.349	0.314	0.285	0.278	0.292	0.308	0.321	0.330	0.341	0.353	0.362	0.364
2	0.529	0.529	0.507	0.462	0.420	0.388	0.355	0.318	0.288	0.281	0.295	0.312	0.324	0.333	0.345	0.357	0.365	0.368
0.2	0.534	0.534	0.512	0.467	0.425	0.393	0.359	0.322	0.292	0.284	0.298	0.315	0.327	0.336	0.347	0.359	0.367	0.370
0.0	0.534	0.534	0.512	0.468	0.425	0.393	0.360	0.323	0.292	0.284	0.298	0.315	0.328	0.336	0.347	0.359	0.368	0.370
									May									
40	0.007	0.007	0.007	0.008	0.008	0.009	0.009	0.010	0.010	0.010	0.010	0.011	0.012	0.013	0.013	0.010	0.008	0.009
20	0.217	0.220	0.235	0.254	0.265	0.269	0.266	0.257	0.247	0.241	0.243	0.246	0.250	0.252	0.251	0.245	0.248	0.252
14	0.352	0.354	0.355	0.353	0.348	0.339	0.321	0.296	0.270	0.262	0.272	0.287	0.300	0.311	0.323	0.338	0.355	0.361
10	0.402	0.403	0.397	0.385	0.374	0.359	0.335	0.304	0.276	0.267	0.279	0.296	0.312	0.326	0.342	0.365	0.387	0.394
8	0.414	0.415	0.408	0.396	0.384	0.368	0.342	0.309	0.280	0.271	0.283	0.301	0.318	0.333	0.349	0.373	0.396	0.404
6	0.423	0.425	0.418	0.406	0.394	0.377	0.349	0.314	0.283	0.274	0.287	0.306	0.324	0.339	0.356	0.380	0.403	0.411

4	0.432	0.433	0.426	0.414	0.402	0.384	0.356	0.319	0.286	0.277	0.291	0.311	0.329	0.344	0.361	0.386	0.410	0.417
2	0.439	0.440	0.433	0.421	0.409	0.391	0.361	0.323	0.289	0.280	0.294	0.315	0.333	0.348	0.365	0.390	0.414	0.422
0.2	0.443	0.445	0.438	0.426	0.414	0.396	0.366	0.326	0.292	0.283	0.298	0.318	0.336	0.351	0.368	0.393	0.417	0.425
0.0	0.444	0.445	0.439	0.426	0.415	0.396	0.366	0.327	0.293	0.283	0.298	0.319	0.337	0.351	0.368	0.393	0.418	0.425
									Jun									
40	0.007	0.007	0.007	0.008	0.008	0.009	0.010	0.010	0.011	0.011	0.010	0.011	0.011	0.012	0.011	0.008	0.008	0.008
20	0.194	0.197	0.215	0.241	0.259	0.267	0.267	0.260	0.249	0.240	0.241	0.245	0.249	0.252	0.254	0.260	0.271	0.276
14	0.282	0.285	0.298	0.319	0.331	0.330	0.319	0.297	0.273	0.263	0.273	0.289	0.304	0.318	0.338	0.366	0.391	0.398
10	0.316	0.318	0.328	0.345	0.354	0.348	0.331	0.305	0.278	0.268	0.280	0.300	0.318	0.335	0.360	0.394	0.425	0.433
8	0.328	0.330	0.339	0.356	0.364	0.357	0.337	0.309	0.281	0.272	0.285	0.306	0.325	0.343	0.368	0.403	0.436	0.444
6	0.338	0.339	0.349	0.366	0.373	0.365	0.344	0.314	0.284	0.275	0.290	0.312	0.332	0.351	0.376	0.412	0.444	0.453
4	0.346	0.348	0.357	0.374	0.381	0.372	0.349	0.317	0.287	0.279	0.294	0.318	0.338	0.357	0.383	0.420	0.452	0.461
2	0.353	0.355	0.364	0.380	0.388	0.378	0.354	0.321	0.290	0.282	0.299	0.322	0.343	0.362	0.389	0.426	0.459	0.467
0.2	0.358	0.359	0.369	0.385	0.393	0.383	0.359	0.325	0.293	0.285	0.302	0.326	0.347	0.365	0.392	0.429	0.462	0.471
0.0	0.358	0.360	0.369	0.386	0.394	0.384	0.359	0.325	0.294	0.285	0.303	0.327	0.347	0.366	0.392	0.430	0.463	0.471
									Jul									
40	0.007	0.007	0.008	0.008	0.009	0.010	0.011	0.011	0.011	0.011	0.010	0.010	0.010	0.010	0.009	0.007	0.007	0.007
20	0.188	0.191	0.209	0.235	0.256	0.265	0.267	0.261	0.250	0.242	0.243	0.247	0.249	0.249	0.251	0.262	0.278	0.283
14	0.256	0.259	0.276	0.300	0.317	0.320	0.313	0.296	0.274	0.265	0.277	0.295	0.309	0.324	0.348	0.379	0.408	0.416
10	0.282	0.284	0.299	0.320	0.335	0.335	0.323	0.303	0.279	0.271	0.285	0.306	0.324	0.342	0.370	0.408	0.443	0.452
8	0.292	0.295	0.309	0.330	0.344	0.342	0.329	0.307	0.282	0.275	0.290	0.312	0.332	0.350	0.379	0.418	0.454	0.463
6	0.302	0.304	0.318	0.338	0.352	0.349	0.334	0.311	0.285	0.278	0.295	0.319	0.339	0.359	0.388	0.428	0.464	0.473
4	0.309	0.312	0.325	0.346	0.359	0.355	0.338	0.314	0.288	0.282	0.300	0.325	0.346	0.366	0.396	0.437	0.473	0.483
2	0.316	0.318	0.331	0.352	0.365	0.360	0.343	0.317	0.291	0.286	0.305	0.330	0.351	0.371	0.403	0.444	0.481	0.491
0.2	0.320	0.323	0.336	0.357	0.370	0.365	0.347	0.321	0.294	0.289	0.309	0.334	0.355	0.375	0.407	0.448	0.485	0.495
0.0	0.321	0.323	0.337	0.357	0.371	0.366	0.347	0.321	0.295	0.289	0.309	0.334	0.355	0.375	0.407	0.449	0.486	0.496
									Aug									
40	0.009	0.009	0.009	0.009	0.010	0.010	0.011	0.011	0.011	0.010	0.010	0.010	0.010	0.010	0.009	0.007	0.005	0.005
20	0.195	0.197	0.213	0.234	0.253	0.263	0.264	0.260	0.252	0.246	0.249	0.253	0.255	0.253	0.245	0.247	0.257	0.262
14	0.258	0.260	0.274	0.293	0.308	0.313	0.307	0.292	0.275	0.271	0.285	0.304	0.322	0.335	0.348	0.369	0.391	0.399

10	0.280	0.282	0.293	0.310	0.324	0.325	0.316	0.299	0.281	0.278	0.293	0.316	0.337	0.353	0.369	0.397	0.425	0.434
8	0.289	0.291	0.302	0.318	0.331	0.331	0.320	0.302	0.283	0.281	0.298	0.322	0.344	0.361	0.378	0.407	0.436	0.445
6	0.297	0.299	0.309	0.326	0.338	0.337	0.325	0.305	0.286	0.285	0.303	0.329	0.352	0.369	0.387	0.416	0.445	0.455
4	0.304	0.306	0.316	0.332	0.344	0.342	0.329	0.308	0.289	0.289	0.309	0.335	0.359	0.376	0.395	0.425	0.455	0.464
2	0.310	0.312	0.322	0.338	0.349	0.347	0.332	0.311	0.292	0.293	0.314	0.341	0.364	0.382	0.401	0.432	0.462	0.472
0.2	0.314	0.316	0.326	0.342	0.354	0.351	0.336	0.315	0.295	0.296	0.318	0.345	0.368	0.386	0.405	0.436	0.467	0.476
0.0	0.314	0.316	0.327	0.343	0.355	0.352	0.337	0.315	0.296	0.296	0.318	0.345	0.368	0.386	0.405	0.436	0.467	0.477
									Sep									
40	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.011	0.010	0.010	0.010	0.010	0.010	0.010	0.010	0.008	0.006	0.005
20	0.212	0.214	0.224	0.238	0.250	0.259	0.260	0.256	0.251	0.251	0.255	0.262	0.267	0.263	0.246	0.233	0.230	0.230
14	0.277	0.278	0.285	0.296	0.305	0.307	0.301	0.286	0.273	0.276	0.293	0.316	0.337	0.348	0.348	0.350	0.359	0.362
10	0.298	0.299	0.304	0.313	0.319	0.319	0.309	0.292	0.278	0.282	0.301	0.328	0.351	0.366	0.368	0.377	0.391	0.395
8	0.307	0.307	0.312	0.321	0.326	0.324	0.313	0.295	0.281	0.286	0.306	0.334	0.359	0.374	0.377	0.386	0.401	0.406
6	0.315	0.315	0.320	0.328	0.333	0.330	0.317	0.298	0.284	0.289	0.311	0.341	0.366	0.382	0.386	0.395	0.411	0.415
4	0.322	0.323	0.327	0.335	0.339	0.335	0.321	0.301	0.287	0.293	0.316	0.347	0.373	0.389	0.393	0.403	0.419	0.424
2	0.328	0.329	0.333	0.341	0.344	0.340	0.325	0.304	0.289	0.297	0.321	0.352	0.379	0.394	0.399	0.410	0.427	0.431
0.2	0.332	0.333	0.337	0.345	0.349	0.344	0.329	0.307	0.292	0.300	0.325	0.356	0.383	0.398	0.403	0.414	0.431	0.436
0.0	0.333	0.333	0.337	0.345	0.350	0.345	0.329	0.308	0.293	0.300	0.325	0.357	0.383	0.398	0.403	0.415	0.431	0.436
									Oct									
40	0.010	0.011	0.012	0.012	0.012	0.011	0.011	0.010	0.010	0.010	0.010	0.010	0.009	0.009	0.009	0.008	0.006	0.006
20	0.234	0.234	0.238	0.246	0.252	0.255	0.255	0.251	0.248	0.253	0.261	0.268	0.272	0.267	0.249	0.222	0.209	0.207
14	0.309	0.308	0.307	0.309	0.309	0.305	0.295	0.281	0.270	0.279	0.299	0.322	0.340	0.349	0.346	0.330	0.323	0.322
10	0.332	0.330	0.327	0.327	0.323	0.316	0.304	0.287	0.275	0.285	0.307	0.333	0.354	0.365	0.365	0.353	0.350	0.349
8	0.341	0.339	0.335	0.334	0.330	0.322	0.308	0.290	0.278	0.288	0.312	0.340	0.361	0.373	0.373	0.361	0.359	0.359
6	0.349	0.347	0.343	0.342	0.337	0.328	0.312	0.293	0.281	0.292	0.317	0.346	0.368	0.380	0.381	0.369	0.367	0.367
4	0.356	0.354	0.351	0.349	0.343	0.333	0.317	0.296	0.284	0.296	0.322	0.352	0.375	0.386	0.388	0.377	0.374	0.374
2	0.363	0.360	0.357	0.355	0.349	0.338	0.321	0.300	0.287	0.299	0.327	0.357	0.380	0.391	0.393	0.382	0.380	0.380
0.2	0.367	0.364	0.361	0.359	0.354	0.343	0.325	0.303	0.290	0.302	0.330	0.361	0.383	0.395	0.396	0.385	0.384	0.383
0.0	0.367	0.365	0.361	0.359	0.354	0.343	0.325	0.303	0.290	0.303	0.331	0.362	0.384	0.395	0.396	0.386	0.384	0.384
									Nov									

40	0.009	0.009	0.010	0.012	0.012	0.011	0.010	0.010	0.010	0.010	0.010	0.009	0.008	0.008	0.008	0.007	0.006	0.006
20	0.265	0.262	0.257	0.256	0.254	0.251	0.248	0.245	0.246	0.254	0.263	0.267	0.263	0.254	0.241	0.218	0.198	0.194
14	0.361	0.356	0.343	0.330	0.317	0.304	0.290	0.275	0.267	0.280	0.299	0.316	0.324	0.327	0.333	0.321	0.305	0.301
10	0.388	0.382	0.367	0.350	0.333	0.316	0.299	0.282	0.273	0.286	0.308	0.327	0.337	0.342	0.352	0.342	0.328	0.324
8	0.397	0.392	0.375	0.358	0.340	0.323	0.304	0.286	0.276	0.290	0.312	0.333	0.343	0.349	0.358	0.348	0.335	0.331
6	0.406	0.400	0.383	0.366	0.348	0.329	0.309	0.289	0.279	0.293	0.317	0.338	0.350	0.355	0.365	0.355	0.341	0.338
4	0.413	0.408	0.391	0.373	0.354	0.335	0.314	0.293	0.282	0.297	0.322	0.344	0.355	0.360	0.370	0.360	0.347	0.343
2	0.420	0.414	0.397	0.379	0.360	0.340	0.319	0.297	0.285	0.300	0.326	0.348	0.359	0.365	0.374	0.364	0.351	0.348
0.2	0.424	0.418	0.401	0.383	0.364	0.345	0.323	0.300	0.288	0.303	0.329	0.352	0.363	0.368	0.377	0.367	0.354	0.350
0.0	0.424	0.418	0.401	0.384	0.365	0.345	0.323	0.301	0.289	0.303	0.329	0.352	0.363	0.368	0.377	0.368	0.354	0.351
									Dec									
40	0.008	0.008	0.008	0.010	0.011	0.010	0.010	0.010	0.011	0.011	0.010	0.009	0.009	0.008	0.007	0.007	0.006	0.006
20	0.286	0.282	0.268	0.257	0.252	0.249	0.245	0.242	0.243	0.253	0.261	0.263	0.258	0.246	0.225	0.201	0.186	0.183
14	0.411	0 405	0.379	0.346	0.323	0.307	0.291	0.275	0.265	0.278	0.296	0.308	0.312	0.310	0.306	0.292	0.280	0.278
10	0.446	0.440	0.409	0.370	0.342	0.322	0.302	0.282	0.270	0.284	0.303	0.318	0.323	0.323	0.323	0.312	0.301	0.298
8	0.457	0.450	0.418	0.378	0.350	0.329	0.307	0.286	0.274	0.288	0.307	0.323	0.329	0.329	0.329	0.317	0.306	0.304
6	0.466	0.459	0.427	0.387	0.359	0.336	0.313	0.291	0.277	0.291	0.312	0.328	0.335	0.334	0.334	0.322	0.311	0.309
4	0.474	0.468	0.436	0.395	0.366	0.343	0.319	0.295	0.280	0.294	0.315	0.333	0.339	0.338	0.338	0.326	0.315	0.313
2	0.481	0.474	0.442	0.401	0.372	0.349	0.324	0.299	0.283	0.297	0.319	0.336	0.343	0.342	0.341	0.329	0.319	0.316
0.2	0.485	0.478	0.446	0.405	0.376	0.353	0.328	0.303	0.287	0.300	0.322	0.340	0.346	0.345	0.344	0.332	0.321	0.318
0.0	0.485	0.479	0.447	0.406	0.377	0.353	0.329	0.303	0.287	0.300	0.322	0.340	0.347	0.346	0.344	0.332	0.321	0.319

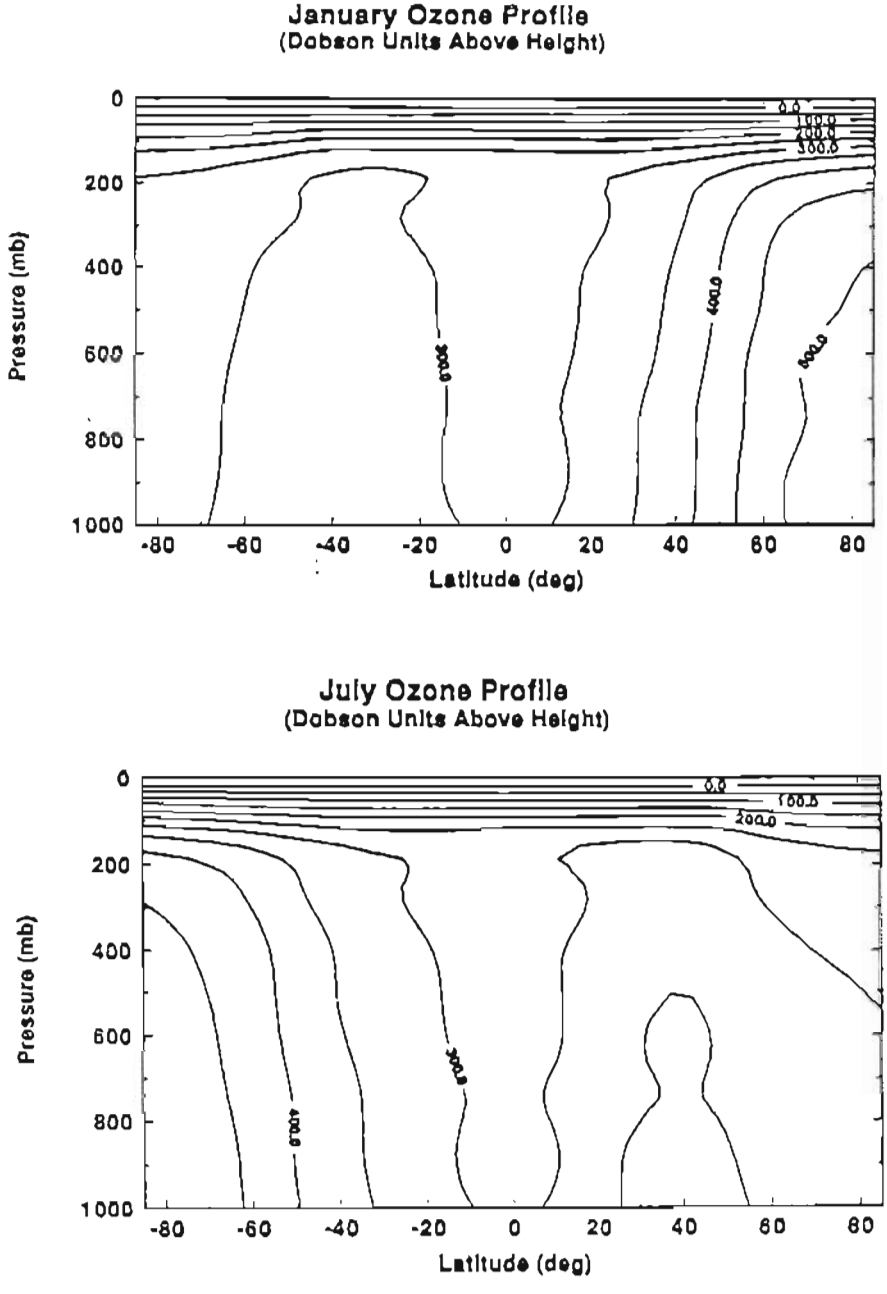

Figure B.4. Ozone profiles used for our 2D model simulations (Top) January (Bottom) July

APPENDIX C

SOURCE CODE OF THE GCRC 2-D MODEL

```
* Main Body of Program
      program main
      real t(10),z(10),p(10),sg(10),u(10,20),v(10,20),w(10,20)
      real wt(10,20),h(10,20),pr(10,20),uvp(10,20),xr(10),vt(10,20)
      real at(10),pa(10),za(10),cp(10),c(10),ra(10,20),rs(10,20)
      real t2d(10,20),p2d(10,20),z2d(10,20),pa2d(10,20),za2d(10,20)
      real ch(10),ce(10),f(20),sun,f1(20),sab(20),sth(20),cp2d(10,20)
      real hcld(20,4),mcld(20,4),lcld(20,4),uo2d(10,20),tco2(10,20)
      real at1(10),ch2d(10,20),ce2d(10,20),asa(20),avb(20),ava(20)
      real elev(36),ely(20),tp2d(10,20),xt(10),pha(20),phb(20)
      real c2d(10,20),dv(10,20),du(10,20),rdw(10,20)
      real dpr(10,20),dw(10,20),uav(10,20),vav(10,20),tvp(10,20)
       real ts(20),ocfra(20),ot(20),uo(10),uco2(10,20),tv1(10,20)
        real ti(20),xi(20),al(20),albl(20),vice(20),suavg(20)
        real pavg(10,20),tavg(10,20),avgv(10,20),emp(20)
       real pav(10,20),ps(20),gc(20),srg(20),tdeep,vi(20)
       real stravg(10,20),smtmp,smstr,rhp(4,20),uozon(12,10,20)
       real smuz,uzavg(10,20),cwat(10,20),dcwat(10,20),rain(20)
       real hsnow(20),twp(10,20),qwp(10,20),smuvp,auvp(10,20)
*****************************************************************
      real s0,step,s1,time,day,sout,strn,strs,dstep,dmp
      real c20,ch40,n20,abtot,re,ftop,rht,tke,dphi
      real pout,dwat,xgo,rh0(20),sk(20),tku,ytt,udrag
      real wo,vk,ho,hk,to(7,20),docean,uvam,year,area,ytr
      real stav,surtmp
       real vtpz(20),uvpz(20),kza(22),angza(22)
      integer nt,kap,nset,nturb,nyao,kozone
      integer nb,ndyn,npree,navg
*     data z/34.,20.0,14.0,12.0,10.0,8.0,6.0,4.0,2.0,0.0/
* z vertical boundary between layers
      data z/34.,20.0,14.0,10.0,8.0,6.0,4.0,2.0,0.2,0.0/
* surface elevations for zones north to south
      data elev/0,84,242,222,315,315,210,262,1000,1000
     &    ,601,599,335,195,140,112,127,111,132,115,151,176,
     & 163,129,69,31,20,12,2,0,0,283,1200,1892,2000,2640/
```

```
* earth radius
      re=6.4e6
* ndyn,navg,stav, nturb are not normally used
      ndyn=0
      navg=0
       stav=0.
       nturb=0
* year for cyclic time.  print out time =365*year+time
* 0,time,365.25 for calculations
      year=0.0
* datm is the only input (data) file
          open(unit=10,file='datm')
* oc82 gives the pressure, temp, u'v',v'T', etc at the end of the
* full run
       open(unit=20,file='oc82' )
*        open(unit=30,file='outuv' )
*  seasonal averages of 2-D u'v' are in uvp.prn
       open(unit=35,file='uvp.prn')
* time series of surfacte temp, radiation balance, kinetic energy
* and V(2,3)
      open(unit=40,file='ocw')
*  seasonal averages of 2-D temp are in tmp.prn
       open(unit=50,file='tmp.prn')
*  seasonal averages of 2-D stream func. are in str.prn
       open(unit=52,file='str.prn')
*        open(unit=55,file='ice.prn')
*       open(unit=56,file='snow.prn')
* Latitudinal profile of vertically averaged precipitation
        open(unit=57,file='rain.prn')
*  seasonal averages of 2-D zonal velocity are in uz.prn
       open(unit=58,file='uz.prn')
* Latitudinal profile of vertically averaged v'T'
       open(unit=62,file='vtpz.prn')
*  Latitudinal profile of vertically averaged u'v'
       open(unit=65,file='uvpz.prn')
*  Latitudinal profile of absorbed solar -emitted IR at top of atmosphere
        open(unit=66,file='smf.prn')
*  Latitudinal profile of vertically averaged kinetic energy per unit mass
        open(unit=67,file='kza.prn')
*  Belt angular momentum
       open(unit=68,file='angza.prn')
*
* step is the radiation time step, day is the end of run, dwat is not usually used,
* time is the start day, pout is the number of iterations between output
* c20, ch40, n20, are the initial carbon dioxide, methane, and nitrous oxide
concentrations
* trs is the energy transport coefficient when running in an EBM mode using
* subroutine tra, nb is the number of latitudinal boxes, npree is a toggle to have print-out
```

```
* to screen (1) not (0),  nyao is not normally used
      read(10,*) step,day,dwat,time,pout,c20,ch40,n20,trs,
     & nb,npree,nyao
* xt are the assumed Kz values at the vertical boundaries between boxes
          read(10,*) (xt(i),i=1,10)
* dstep is the dynamical time step, yk and uvam are not normally used
          read(10,*) yk,uvam,dstep
           read(10,*) (rh0(i),i=1,9)
*  nset Number of hydrostic adjustment iterations, dmp is not normally used
          read(10,*) nset,dmp
* ocean transport coefficients strn and strs for northern and souther hemispheres
* used by subroutine trans.  ytr and ytt are usually the same and are
* the added term to the v'T' flux not given by the Stone and Yao scheme (ytr is for the
* subtropics and ytt is for the tropics.(these are the coefficients of the temperaure
* gradient driven transport of heat)
          read(10,*) strn,strs,ytr,ytt
* sk is not normally used
          read(10,*) (sk(i),i=1,20)
* suavg is the annual mean flux of radiation reaching the top of the atmosphere for each
* zone (W/m2)
       read(10,*) (suavg(i),i=1,9)
        do i=10,18
         suavg(i)=suavg(19-i)
        end do
* initial pressure at the center of the gridbox (mb)
       do i=1,10
           read(10,*) (pa2d(i,j),j=1,nb+1)
          do j=1,nb+1
          if(i.le.2) then
           pa2d(i,j)=pa2d(i,j)*(1000.+0.0*abs(9.5-j))
           else
           pa2d(i,j)=pa2d(i,j)*(1000.+0.0*abs(9.5-j))
           end if
           end do
         end do
* initial pressure at the vertical boundary of the gridbox (mb)
          do i=1,10
           read(10,*) (p2d(i,j),j=1,nb+1)
          do j=1,nb+1
           p2d(i,j)=p2d(i,j)*1000.
           end do
          pa2d(i,19)=pa2d(i,18)
         end do
* initial meridional velocities (m/s)
          do i=1,10
         read(10,*) (v(i,j),j=1,nb+1)
          end do
* initial zonal velocities (m/s)
```

```
      do i=1,10
       read(10,*) (u(i,j),j=1,nb+1)
        end do
* initial values of (1/cp)ΔL/ΔT for calculating the effective heat capacity of a moist
* atmosphere
       do i=1,10
       read(10,*) (c2d(i,j),j=1,nb)
       end do
* initial convective energy transport (K per radiation time step)
       do i=1,10
       read(10,*) (ch2d(i,j),j=1,nb)
       end do
* initial convective energy transport (K per rad step) (ce2d is the long term
* convective transport and ch2d is a fine tuning of ce2d)  See subroutines
* tpch.f and lapadj.f called from rad.f
       do i=1,10
       read(10,*) (ce2d(i,j),j=1,nb)
       end do

* initial 2-D temperature profile
       do i=1,10
       read(10,*) (t2d(i,j),j=1,nb)
       end do
* initial sea ice fraction area coverage=al^2
       read(10,*) (al(j),j=1,nb)
* initial sea ice thickness (m)
       read(10,*) (xi(j),j=1,nb)
* initial ocean surface temperature (K)
       read(10,*) (ot(j),j=1,nb)
* initial surface temperature (K) of land surfaces
       read(10,*) (ts(j),j=1,nb)
* initial surface temperature (K) for sea ice regions
       read(10,*) (ti(j),j=1,nb)
* initial snow thickness (decimeters)
      read(10,*) (hsnow(j),j=1,nb)
      do j=1,nb
        hsnow(j)=hsnow(j)/10.
* initial sea ice volume
       vi(j)=al(j)*al(j)*xi(j)
       end do
* initial v'T' fluxes m/sK
       do i=1,10
       read(10,*) (tvp(i,j),j=1,nb+1)
     end do
* initial u'v' fluxes m2/s2
       do i=1,10
       read(10,*) (uvp(i,j),j=1,nb+1)
     end do
```

```
* Zonal frictional drag coefficients, and meridional frictional drag xr(1) and xr(2)
* xr(8) is a zonal drag on the top layer of the atmosphere see subroutine dy3.f
        read(10,*) (xr(i),i=1,10)
* vertical boudaries of each layer
        read(10,*) (z(i),i=1,10)
* deep ocean stuff not presently in use except for docean which is the depth of the deep
*  ocean.
     read (10,*) wo,rl,hk,ho,vk,docean,tdeep
* initial deep ocean temperature profile
     read(10,*) ((to(i,j),j=1,9),i=1,6)

        do i=1,6
        do j=10,18
          to(i,j)=to(i,19-j)
         end do
        end do
* high, middle , and low cloud data for spr, summ, aut, win 1,2,3,4
     do i=18,1,-1
     read(10,*) hcld(i,1),hcld(i,2),hcld(i,3),hcld(i,4)
     end do
     do i=18,1,-1
     read(10,*) mcld(i,1),mcld(i,2),mcld(i,3),mcld(i,4)
     end do
     do i=18,1,-1
     read(10,*) lcld(i,1),lcld(i,2),lcld(i,3),lcld(i,4)
     end do
* ocean fraction of each zone
      do i=1,18
*       ot(i)=to(1,i)
       read(10,*) ocfra(i)
      end do
* land albedos
        do j=1,nb
         read(10,*) albl(j)
         srg(j)=0.00
        end do
*spring ozone profile
          do i=1,10
         read(10,*) (uo2d(i,j),j=1,18)
         end do
* surface relative humidity profiles 1,18 N to South j=1,4 sp, su, atum, win
       do j=18,1,-1
        read(10,*) (rhp(i,j),i=1,4)
       end do
      do j=1,nb
       rh0(j)=rhp(1,j)+(time-0.)*(rhp(2,j)-rhp(1,j))/91.
      end do
```

```
*         do j=1,9
*         rh0(j)=rh0(19-j)
*         end do
* seasonal ozone profiles 1 Mar, 2 Apr, .. 12 Feb
        do k=1,12
        do i=1,10
          read(10,*) (uozon(k,i,j),j=1,18)
          end do
        end do

          do j=1,nb
*       uo2d(1,j)=0.10
       to(1,j)=t2d(10,j)
          end do
* No Mountains
     do 10 i=1,36
     elev(i)=0.0
10    continue
*
          do  i=1,10
          do  j=1,nb+1
* averages of pressure , temp, v,u
        pav(i,j)=pa2d(i,j)
          twp(i,j)=0.0
*
        pavg(i,j)=0.0
         tavg(i,j)=0.0
         avgv(i,j)=0.0
          end do
         t2d(i,nb+1)=t2d(i,nb)
          end do
* set up to calculate density from pressures and temperatures
     call box (nb,pha,phb,sth,asa,ava,avb,z,z2d,za2d
    &    ,pa2d,ra,rs,u,v,w,wt,h,t2d,dphi,rh0)
* calc elevation averages( not presently used)
          call avg(sth,elev,nb,ely)
         nr=0
        area=0.0
         do 45 j=1,nb
* surface are of the earth
        area=area+asa(j)
* surface boundary
         z2d(10,j)=ely(j)/1000.0
45       continue
          do i=1,10
           do j=1,nb-1
* set hydrostatic adjustment mass flow to 0.0
        rdw(i,j)=0.0
```

```
          end do
         end do

1006     format(8(f8.3,1x))
1002     format(9(f6.3,1x))
1009     format(19(f5.1,1x))

          do i=1,10
           do j=1,nb+1
        e=esat(t2d(i,j))
* e saturation vapor pressure, cwat specific humidity
       cwat(i,j)=rwat(pa2d(i,j)/1.013e5,e,1,1,rh0(j))
          end do
          end do
1008       format(2x,18(e11.4,2x))
       if(npree.eq.1) then
          print*,'Set-up Complete'
         end if
*  Below is main call to 1D model for each zone
100      s0=0.
* calculates the critical lapse rate from baroclinic theory as given by Stone and
* Carlson 1979
*gc is the critical lapse rate from baroclinic theory
* t2d is the 2D temperature profile at the grid center
* pha is the latitude of the center of the grid
* dphi is the latitudinal grid spacing
*  nb is the number of latitudinal grids
       call baroclinic (gc,t2d,pha,dphi,nb)
* ozone interpolation parameter to interpolate ozone monthly values
        kozone=int(1+time/30.5)

         do 180 j=1,nb
         do 110 i=1,10
* change 2-D to 1 D data for sending to 1DRCM subroutines
         at(i)=t2d(i,j)
*ch/ch2d are the convective adjustment terms 1D and 2D they should be close
* to zero because they are small adjustments where ce/ce2d are the convective
* adjustments that are made each time bothe ce & ch have units of (K)
         ch(i)=ch2d(i,j)
         ce(i)=ce2d(i,j)
*uo/uo2d are ozone amounts from the top of the atmosphere in cm
*        uo(i)=uo2d(i,j)
        uo(i)=uozon(kozone,i,j)
* cp is the heat capacity of the atmosphere
         cp(i)=cp2d(i,j)
* c is the correction to cp for a moist atmosphere
         c(i)=c2d(i,j)
*pa/pa2d are pressures in atms / pascals
```

```
          pa(i)=pa2d(i,j)/1.0e5
* p is the pressure between vertical layers
          p(i)=(p2d(i,j))/1.0e5
*za/z are the heights at the center and boundary of the atms layers
          za(i)=za2d(i,j)/1000.
          z(i)=z2d(i,j)/1000.
* tp2d is a temperature change calculation variable used for diagnostics
          tp2d(i,j)=t2d(i,j)
110       continue
         if(time.gt.900)then
          ch40=1.60
          end if
      if(time.lt.dwat)then
* dwat allows for the exploration of water vapor feedback
* at1 is the temperature used in water vapor calcs of radiation code
          do 150 i=1,10
          at1(i)=at(i)
150       continue
         end if
* effective surfave drag velocity
* u and v are the zonal and meridional velocities
       udrag=sqrt(((u(9,j)+u(9,j+1))/2.)**2+
     & ((v(9,j)+v(9,j+1))/2.)**2)
* radiation is the main 1DRCM program
*sun is the solar constant times the mean zenith angle
* sout is the reflected solar energy
* f is the flux of ir radiation at each level
* abtot is the total absorbed solar energy by the earth-atms system
* ac is the total cloud cover
* s1 is a diagnostic related to total temperature change of all layers
* nt is the layer corresponding to the tropopause when using the cumuls
* convection scheme
* c20,ch40,n20 are the concentrations of co2, ch4, n2o
* j is the latitudinal grid number
* sth is the sine of latitude boundaries (j+1/2 or j-1/2)
* suavg is the average solar intensity for annual mean conditions
* rhp are the seasonal profiles of surface relative humidity
          call radiation(at,at1,t,p,pa,z,za,ch,ce,sun,sout,f,
     & c,abtot,ac,s1,nt,step,c20,ch40,n20,j,sth,cp,rh0(j),
     & hcld,mcld,lcld,ts(j),ot(j),ocfra(j),uo,gc(j)
     & ,udrag,al(j),xi(j),ti(j),albl(j),srg(j),suavg(j),time
     & ,vi(j),rhp,rain(j),hsnow(j))
* rain(j) is the precipatation from radiative cooling
* return back to 2D mode
          f1(j)=f(1)
          sab(j)=sun-sout
          s0=s0+asa(j)*sun/area
          do 170 i=1,10
```

```
          tp2d(i,j)=t2d(i,j)-at(i)
          t2d(i,j)=at(i)
        rs(i,j)=(1.0+rh0(j)*esat(t2d(i,j)))
          ch2d(i,j)=ch(i)
          ce2d(i,j)=ce(i)
          cp2d(i,j)=cp(i)
          c2d(i,j)=c(i)
          p2d(i,j)=p(i)*1.0e5
            pa2d(i,j)=+ra(i,j)*287*t2d(i,j)*rs(i,j)
          za2d(i,j)=za(i)*1000.
          z2d(i,j)=z(i)*1000.
170       continue
180      continue

          do 183 i=1,9
                do 182 j=1,nb
            vt(i,j)=v(i,j)
                tp2d(i,j)=t2d(i,j)
            rdw(i,j)=0.0
182           continue
183        continue
* Ocean mixed layer horizontal transport of energy
* stm and strs are the turbulent diffusion coefficients for the northern
* hemisphere and southern hemisphere oceans
* tdeep is the effective temperature of the deep ocean (not used here)
     call trans(t2d,sth,nb,cp2d,z2d,step,
    & trs,strn,strs,c2d,ot,ocfra,tdeep,al)
* tra is EBM transport atm and Ocean
*         call tra(t2d,sth,nb,cp2d,z2d,step,trs,str,c2d)
*        if(ndy.lt.0) then
* dyn is the main dynamics portion of program (see eqns 2.7 to 2.9 of text)
* ra is the 2-D density, rs is and adjustment in ra for a moist atmosphere
* step is the radiation time step, and dstep is the dynamics time step
* phb is the cosine of the grid box boundaries (j+1/2 or j-1/2)
* uvp =u'v'  tvp=t'v'  eddy momentum and sensible heat
* qwp =s'w' vertical eddy flux of moisture
* twp =t'w' vertical eddy flux of sensible heat
* cwat is the specific humidity
* vav, uav, pav are the meridional and zonal velocities and pressure averaged over the
* dyn subroutine (i.e. rad step)
* docean is the depth of the mixed layer ocean
* sk are not used at present and should be set to zero
* asa is the earth radius*(sin(phb(j)-sin(phb(j+1))
* ava =cos(pha(j)   avb=cos(phb(j))  time is the time in days & is not used
* by dyn subroutine
     call dyn(u,v,t2d,pa2d,ra,rs,z2d,za2d,pha,phb,step
    &     ,dstep,uvp,nb,xt,xr,asa,ava,avb,c2d,p2d,time,vav,uav
    & ,docean,sk,tvp,pav,rh0,twp,qwp,cwat)
```

```
*
*
      do j=1,nb+1
       pav(10,j)=pa2d(10,j)
*       v(1,j)=0.0
       end do
* ytt is tropical CT coefficient for t'v' calculation using eqn 2.42 of text
* ytr is mid to high latitude CT
* tv1 is t'v' without the addition of eq 2.42
* yao is the stone and Yao (1987,1990) parameterization of eddy fluxes
    call yao(pa2d,t2d,uav,p2d,za2d,z2d,phb,uvp,nb,dphi,rh0
   & ,c2d,tvp,ytr,ytt,twp,cwat,qwp,tv1)

       do i=1,9
        do j=1,nb
*        tp2d(i,j)=t2d(i,j)-tp2d(i,j)
*         v(i,j)=.5*v(i,j)+.5*vt(i,j)
         end do
       end do
* hydrostatic adjustment
* rdw is the transfer of mass across the vertical boundary between two layers
* resulting from the hydrostatic adjustment
* ps is is not presently in use
* dpr is the rate of change of pressure
* nset is usually =1
       do i=1,nset
        call hys(nb,pa2d,t2d,ra,rs,pr,w,rdw,
   & dw,step*86400.,dpr,z2d,za2d,c2d,p2d,v,u,ps,cwat)
        end do
*        end if
*
     do i=1,9
      do j=1,nb
      tp2d(i,j)=t2d(i,j)-tp2d(i,j)
* it rains some more if there is cooling due to dynamic transport of energy
       if (tp2d(i,j).lt.0.0) then
        e1=esat(t2d(i,j)+1)
        e=esat(t2d(i,j))
        dr=rwat(pa2d(i,j),e1,j,1,rh0(j))
   &  -rwat(pa2d(i,j),e,j,1,rh0(j))
    rain(j)=rain(j)-(p2d(i+1,j)-p2d(i,j))*dr*tp2d(i,j)
   & /(step*9.8)
       end if
       end do
       end do
* assume that fresh snow is 10 times less dense than water
       do j=1,nb
        if(t2d(9,j).le.272) then
```

```
            hsnow(j)=hsnow(j)+10.0*rain(j)*step/1000.
          end if
          end do


         do i=1,10
          do j=1,nb+1
* calc mass stream function vav was mean meridional velocity and is tranformed into
* mass stream function (10Tg/sec)
           vav(i,j)=(ra(i,j-1)+ra(i,j))*vav(i,j)*3.14*re*avb(j)
     & *(z2d(i,j)-z2d(i+1,j))/(1e10)
          end do
         end do
*
        do i=2,9
         do j=1,nb
        vav(i,j)=vav(i-1,j)+vav(i,j)
        end do
        end do
*

       do i=1,9
        do j=1,nb
         e=esat(t2d(i,j))
* cwat is gH2O/gAir)
        cwat(i,j)=rwat(pa2d(i,j)/1.013e5,e,1,1,rh0(j))
        end do
        end do
* advection of water. Not in use really at this time
      call gasstr(ra,cwat,v,w,dcwat,phb,pha,z2d
     & ,nb,emp)
*
* seasonal zonal averages of temp, stream function, zonal velocity
* u'v' written to tmp.prn, str.prn, uz.prn, uvp.prn
        call xavge(tavg,t2d,
     & time,smtmp,nb,step,pa2d,pha,'tmp',50)
      call xavge(stravg,vav,time,smstr,
     & nb,step,pa2d,pha,'str',52)
       call xavge(uzavg,u,time,smuz
     & ,nb+1,step,pa2d,phb,'uz',58)
      call xavge(auvp,uvp,time,smuvp
     & ,nb,step,pa2d,pha,'uvp',35)
* Shiparo filter this could be written in more compact form
* but we were experimenting around with the effects
* of filtering different layers
         call shfil(t2d,nb,9)
         call shfil(t2d,nb,1)
         call shfil(t2d,nb,2)
```

```
*
        call shfil(t2d,nb,3)
        call shfil(t2d,nb,4)
        call shfil(t2d,nb,5)
        call shfil(t2d,nb,6)
        call shfil(t2d,nb,7)
        call shfil(t2d,nb,8)
*         call shfil(t2d,nb,10)
*
*         call shfil(ra,nb,1)
*         call shfil(ra,nb,2)
*         call shfil(ra,nb,9)

*        call filt1(pa2d,nb)
        call shfil(v,nb+1,9)
        call shfil(v,nb+1,2)
        call shfil(v,nb+1,3)
*  between these lines has been added since 5/17
        call shfil(v,nb+1,1)
        call shfil(v,nb+1,4)
        call shfil(v,nb+1,5)
        call shfil(v,nb+1,6)
        call shfil(v,nb+1,7)
        call shfil(v,nb+1,8)
*           call shfil(u,nb+1,5)
*            call shfil(u,nb+1,6)
*            call shfil(u,nb+1,7)

*
        call shfil(u,nb+1,1)
        call shfil(u,nb+1,2)
         call shfil(u,nb+1,3)
          call shfil(u,nb+1,4)
          call shfil(u,nb+1,8)
          call shfil(u,nb+1,9)


       do j=1,nb
        to(1,j)=ot(j)
*          to(1,j)=t2d(10,j)
        end do
*     call ocean(to,pha,phb,step/365.,vk,hk,ho,wo,docean,-.1,nb)
* total kinetic energy per unit area tke & tku
        tke=0.0
       tku=0.0
         su1=0.
        navg=navg+1
            tmax=t2d(1,1)
```

```
          do 187 i=1,10
                do 186 j=1,nb
          if(t2d(i,j).gt.tmax)then
           tmax=t2d(i,j)
          end if

           tke=tke+(z2d(i,j)-z2d(i+1,j))*dphi*
     & cos(phb(j))*vav(i,j)*vav(i,j)*ra(i,j)/40000.
           tku=tku+(z2d(i,j)-z2d(i+1,j))*dphi*
     & cos(phb(j))*uav(i,j)*uav(i,j)*ra(i,j)/40000.

*             t2d(10,j)=to(1,j)
*                  tp2d(i,j)=t2d(i,j)-tp2d(i,j)
*       su1=su1+ra(i,j)*(phb(j)-phb(j+1))
* su1 is a diagnostic check for conservation of energy
       su1=su1+ra(i,j)*(z2d(i,j)-z2d(i+1,j))
     & *tp2d(i,j)*(c2d(i,j)+1)*1005.*asa(j)/(2.*re**2*step)
186          continue
187        continue

8001       format (9(2x,f8.2))
        if(time.gt.365.25) then
         time=time-365.25
         year=year+1
        end if
*
      time=time+step
* actual ice volume m^3
      do j=1,nb
      vice(j)=ocfra(j)*2.57e14*al(j)**2*(asa(j))/6.4e6/1e13
      end do
*          write(*,2002) 365.*year+time,vice(1)+vice(2)+
*     & vice(3),vice(18)+vice(17)+vice(16)+vice(15)+vice(14)

* output is below
       if(time+365.25*year.gt.9000.and.time+365.25*year.le
     & .9000+step)then
          c20=640.
           write(20,*)'v'
           do i=1,10
           write(20,3001) (v(i,j),j=1,nb+1)
           end do
           write(20,*)'u'
           do i=1,10
           write(20,3000) (u(i,j),j=1,nb+1)
           end do
           write(20,*)'w'
           do i=1,10
```

```
            write(20,4000) (w(i,j),j=1,nb)
            end do
          write(20,*)'t2d'
            do i=1,10
            write(20,3000) (t2d(i,j),j=1,nb)
            end do
        write(20,*)'tocean'
         do i=1,6
          write(20,3000) (to(i,j),j=1,nb)
         end do
            write(20,*) 'f1'
            write(20,3000) (f1(j),j=1,nb)
            write(20,*) 'sab'
          write(20,*) 'al'
           write(20,3001) (al(j),j=1,nb)
          write(20,*) 'area'
           write(20,3001) (al(j)**2,j=1,nb)
          write(20,*) 'area x10^13 m^2'
           write(20,3001) (vice(j),j=1,nb)
          write(20,*) 'xice'
           write(20,3001) (xi(j),j=1,nb)

*          do i=1,10
*           do j=1,nb+1
*           rdw(i,j)=rdw(i,j)*6.28*re*asa(j)
*         write(30,5001) float(80.75-9.5*(j-1)),
*     & za2d(i,j)/1000,(vav(i,j)+vav(i,j+1))/2e9,rdw(i,j)/1e9
*          end do
*         end do
*         do i=2,9
*          do j=1,nb
*         vav(i,j)=vav(i-1,j)+vav(i,j)
*         end do
*         end do
*        do i=1,9
*         write(35,3000) (vav(i,j)/1e10,j=1,nb)
*        end do
         end if
*           update printer count
                  abtot=0
                  ftop=0
             surtmp=0.0
          do 200 j=1,nb
            abtot=abtot+sab(j)*asa(j)/area
            ftop=ftop+f1(j)*asa(j)/area
         surtmp=surtmp+t2d(9,j)*asa(j)/area
200       continue
                prt=prt+1.
```

```
          if (prt.gt.pout.or.time.lt.00.0+00*step) then
         if(npree.eq.1) then
           do i=1,10
           write(*,3000) (tp2d(i,j),j=1,nb)
           end do
         end if
* file 40 is called ocw time series of temp ...
        write(40,8001) 365.*year+time,100*v(3,2)
    & ,surtmp,tku,ftop-abtot,ftop,abtot,t2d(2,3),u(2,3)
     do j=1,nb
     vice(j)=ocfra(j)*2.57e14*al(j)**2*(asa(j))/6.4e6/1e13
     end do
* file 55 is ice.prn
         write(55,2002) 365.*year+time,vice(1)+vice(2)+
    & vice(3),vice(18)+vice(17)+vice(16)+vice(15)+vice(14),
    &  xi(1),xi(2),xi(3),xi(16),xi(17)
    &  ,vi(1),vi(2),vi(3),vi(16),vi(17)
*    &  ,vi(1)-al(1)*al(1)*xi(1)
*    & ,vi(2)-al(2)*al(2)*xi(2),vi(3)-al(3)**2*xi(3)
*    & ,vi(16)-al(16)**2*xi(16),vi(17)-al(17)**2*xi(17)
        write(56,2001) 365.*year+time,(hsnow(j),j=1,nb)
        write(57,2001) 365.*year+time,(rain(j),j=1,nb)
*
* zonal value of verticall integrated u'v', KE, v'T', ang mom, rain, solar-IR
       do j=1,nb
        vtpz(j)=0.0
         uvpz(j)=0.0
         kza(j)=0.0
         angza(j)=0.0
       end do
       do j=2,nb
        do  i=2,9
         vtpz(j)=vtpz(j)+tvp(i,j)*(p2d(i+1,j)-p2d(i,j))/
   & (p2d(10,j)-p2d(2,j))
* KE per unit mass in m^2/s^2
         kza(j)=kza(j)+.5*(u(i,j)**2+v(i,j)**2)
   &*(p2d(i+1,j)-p2d(i,j))/(p2d(10,j)-p2d(2,j))
* angular momentum in units of 10^25 kg m/s
         angza(j)=angza(j)+(6.37**3)*u(i,j)*6.28
   & *cos(phb(j))*(pha(j)-phb(j+1))*
   &  2.*(p2d(i+1,j)-p2d(i,j))/9.8/1.0e7
         uvpz(j)=uvpz(j)+uvp(i,j)*
   & (p2d(i+1,j)-p2d(i,j))/(p2d(10,j)-p2d(2,j))
         end do
        end do
         angza(22)=0.0
         angza(21)=0.0
         angza(20)=0.0
```

```
          do j=1,nb/2
          angza(22)=angza(22)+angza(j)
           end do
          angza(22)=angza(22)+.5*angza(10)
          do j=11,nb
          angza(21)=angza(21)+angza(j)
           end do
          angza(21)=angza(21)+.5*angza(10)
           angza(20)=angza(21)+angza(22)
* file 66 is smf.prn lat profile
* file 62 is vtpz.prn v't' vertical average lat profile
* file 65 is uvpz.prn  u'v' vertical average lat profile
* file 67 is kza.prn total kinetic energy per unit mass lat profile
* file 68 is angza.prn vertical total ang mom for each lat grid lat profile
        write(66,3002) 365.*year+time,(sab(j)-f1(j),j=1,nb)
        write(62,3002) 365.*year+time,(vtpz(j),j=1,nb)
        write(65,3002) 365.*year+time,(uvpz(j),j=1,nb)
        write(67,3002) 365.*year+time,(kza(j),j=1,nb)
        write(68,3003) 365.*year+time,(angza(j),j=1,22)
*
******** More output file 20 is oc82 gives the results of most thing for the end of the
run
      if(time.gt.60000.) then
      write(20,3000) ((pa2d(i,j)/1000.,j=1,nb+1),i=1,10)
            write(20,*)
      write(20,3000) ((p2d(i,j)/1000.,j=1,nb+1),i=1,10)
            write(20,*)
            do i=1,10
            write(20,3001) (v(i,j),j=1,nb+1)
            end do
            write(20,*)
            do i=1,10
            write(20,3000) (u(i,j),j=1,nb+1)
            end do
            write(20,*)
            do i=1,10
            write(20,3001) (c2d(i,j),j=1,nb)
            end do
            write(20,*)
            do i=1,10
            write(20,3001) (ch2d(i,j),j=1,nb)
            end do
            write(20,*)
            do i=1,10
            write(20,3001) (ce2d(i,j),j=1,nb)
            end do
          write(20,*)
           do i=1,10
```

```
          write(20,3000) (t2d(i,j),j=1,nb)
         end do
      write(20,*)
        write(20,3001) (al(j),j=1,nb)
        write(20,3001) (xi(j),j=1,nb)
        write(20,3000) (ot(j),j=1,nb)
        write(20,3000) (ts(j),j=1,nb)
        write(20,3000) (ti(j),j=1,nb)
     write(20,3001) (10.*hsnow(j),j=1,nb)
        write(20,*)
     write(20,3000)((tvp(i,j)
    & ,j=1,nb+1),i=1,10)
        write(20,*)
     write(20,3000) ((uvp(i,j),j=1,nb+1),i=1,10)
      write(20,*) 'rain in mm/day'
          write(20,3001) (rain(j),j=1,nb)
       write(20,*)'evaporation minus precipitation'
        write(20,3001) (emp(j),j=1,nb)
       write(20,*)'snow depth in cm'
        write(20,3001) (10.*hsnow(j),j=1,nb)

       write(20,*) 'al'
        write(20,3001) (al(j),j=1,nb)
       write(20,*) 'area'
        write(20,3001) (al(j)**2,j=1,nb)
       write(20,*) 'area x10^13 m^2'
        write(20,3001) (vice(j),j=1,nb)
       write(20,*) 'xice'
        write(20,3001) (xi(j),j=1,nb)

       write(20,*) 'tocean'
        write(20,3000) (ot(j),j=1,nb)
       write(20,*) 'tland'
        write(20,3000) (ts(j),j=1,nb)
       write(20,*) 'tice'
        write(20,3000) (ti(j),j=1,nb)
       write(20,*) 'tdeep', tdeep
     write(20,*)'tocean'
      do i=1,6
       write(20,3000) (to(i,j),j=1,nb)
      end do
          write(20,*)'w'
          do i=1,10
           write(20,3000) (10000.*w(i,j),j=1,nb)
          end do
       write(20,*) 'twp'
          do i=1,10
        write(20,3001) ((ra(i,j)+ra(i-1,j))*500.*twp(i,j),j=1,nb)
```

```
            end do
           write(20,*) 'qwp'
            do i=1,10
           write(20,3000) (1005.*c2d(i,j)*tvp(i,j)*qwp(i,j)
     & *(ra(i,j)+ra(i-1,j))*.5,j=1,nb)
            end do
           write(20,*) 'qvp'
            do i=1,10
           write(20,3000) (c2d(i,j)*tvp(i,j)
     & *(ra(i,j)+ra(i,j+1))*.5,j=1,nb)
            end do

         write(20,*)'tp2d'
       write(20,3000) ((10.*tp2d(i,j),j=1,nb+1),i=1,10)

            write(20,*) 'f1'
            write(20,3000) (f1(j),j=1,nb)
            write(20,*) 'sab'
            write(20,3000) (sab(j),j=1,nb)
           write(20,*)

          write(20,*) 'ftop' ,ftop
           write(20,*) 'sab', abtot
            write(20,*) 'time', time+365.25*year
      end if
********
2001     format(f7.1,1x,18(f5.2))
2002      format(13(1x,f7.2))
         if(npree.eq.1) then
         do i=1,10
           write(*,3000) (t2d(i,j),j=1,nb)
         end do
         write(*,*)'p'
           write(*,3000) (pa2d(9,j)/100,j=1,nb)
        write(*,*)'u2,u3'
        do i=1,9
        write(*,3000) (u(i,j),j=1,nb)
         end do
*          write(*,*) 'al/xi'
*         write(*,3000) (al(j),j=1,nb)
*          write(*,3000) (xi(j),j=1,nb)
*          write(*,*) 'snow (dm)'
*         write(*,3001) (10.*hsnow(j),j=1,nb)
             write(*,*) time,vav(9,8),su1
           write(*,*) 's  f   ketot',abtot,ftop,tke+tku
*          write(*,*) 'land'
*         write(*,3000) (ts(j),j=1,nb)
*          write(*,*) 'tice'
```

```
*          write(*,3000) (ti(j),j=1,nb)
*          write(*,*) 'tocean'
*          write(*,3000) (ot(j),j=1,nb)
*          write(*,*) 'SH sur temp'
*          write(*,3000) (t2d(9,j),j=10,nb)
*        write(*,3000) (ot(j)*ocfra(j)+ts(j)*(1-ocfra(j)),j=1,nb)
             write(*,*)' '
         end if

         prt=0.0

3000     format(19(f5.1,1x))
3001     format(19(f5.2,1x))
3002     format (19(f7.1,1x))
3003     format(f7.1,1x,22(f5.2,1x))
         end if
*
       if((tmax.gt.1000.0).or.(tmax.lt.50.)) then
        time=1.0e9
        end if
          if (time+365.25*year.gt.day) then
          goto 500
          else
          goto 100
          end if
*
500    continue
         write(20,*)'pa2d'
     write(20,3000) ((pa2d(i,j)/1000.,j=1,nb+1),i=1,10)
           write(20,*)
     write(20,3000) ((p2d(i,j)/1000.,j=1,nb+1),i=1,10)
           write(20,*)
           do i=1,10
           write(20,3001) (v(i,j),j=1,nb+1)
           end do
           write(20,*)
           do i=1,10
           write(20,3000) (u(i,j),j=1,nb+1)
           end do
           write(20,*)
           do i=1,10
           write(20,3001) (c2d(i,j),j=1,nb)
           end do
           write(20,*)
           do i=1,10
           write(20,3001) (ch2d(i,j),j=1,nb)
           end do
           write(20,*)
```

```
        do i=1,10
        write(20,3001) (ce2d(i,j),j=1,nb)
        end do
      write(20,*)
       do i=1,10
        write(20,3000) (t2d(i,j),j=1,nb)
       end do
   write(20,*)
     write(20,3001) (al(j),j=1,nb)
     write(20,3001) (xi(j),j=1,nb)
     write(20,3000) (ot(j),j=1,nb)
     write(20,3000) (ts(j),j=1,nb)
     write(20,3000) (ti(j),j=1,nb)
  write(20,3001) (10.*hsnow(j),j=1,nb)
     write(20,*)
  write(20,3000)((tvp(i,j)
& ,j=1,nb+1),i=1,10)
     write(20,*)
  write(20,3000) ((uvp(i,j),j=1,nb+1),i=1,10)
    write(20,*) 'rain in mm/day'
       write(20,3001) (rain(j),j=1,nb)
     write(20,*)'evaporation minus precipitation'
      write(20,3001) (emp(j),j=1,nb)
     write(20,*)'snow depth in cm'
      write(20,3001) (10.*hsnow(j),j=1,nb)

     write(20,*) 'al'
      write(20,3001) (al(j),j=1,nb)
     write(20,*) 'area'
      write(20,3001) (al(j)**2,j=1,nb)
     write(20,*) 'area x10^13 m^2'
      write(20,3001) (vice(j),j=1,nb)
     write(20,*) 'xice'
      write(20,3001) (xi(j),j=1,nb)

     write(20,*) 'tocean'
      write(20,3000) (ot(j),j=1,nb)
     write(20,*) 'tland'
      write(20,3000) (ts(j),j=1,nb)
     write(20,*) 'tice'
      write(20,3000) (ti(j),j=1,nb)
     write(20,*) 'tdeep', tdeep
  write(20,*)'tocean'
   do i=1,6
    write(20,3000) (to(i,j),j=1,nb)
   end do
       write(20,*)'w'
       do i=1,10
```

```
          write(20,3000) (10000.*w(i,j),j=1,nb)
         end do
       write(20,*) 'twp'
         do i=1,10
        write(20,3001) ((ra(i,j)+ra(i-1,j))*500.*twp(i,j),j=1,nb)
         end do
        write(20,*) 'qwp'
         do i=1,10
        write(20,3000) (1005.*c2d(i,j)*qwp(i,j)
& *(ra(i,j)+ra(i-1,j))*.5,j=1,nb)
         end do
        write(20,*) 'qvp'
         do i=1,10
        write(20,3000) (c2d(i,j)*tvp(i,j)
& *(ra(i,j)+ra(i,j+1))*.5,j=1,nb)
         end do

       write(20,*)'tp2d'
  write(20,3000) ((100.*tp2d(i,j),j=1,nb+1),i=1,10)

         write(20,*) 'f1'
         write(20,3000) (f1(j),j=1,nb)
         write(20,*) 'sab'
         write(20,3000) (sab(j),j=1,nb)
        write(20,*)

       write(20,*) 'ftop' ,ftop
        write(20,*) 'sab', abtot
         write(20,*) 'time', time+365.25*year
          write(20,*) 'U'
      do i=1,10
       do j=1,nb+1
       write(20,*) 57.3*phb(j),-pa2d(i,j)/100.,u(i,j)
       end do
        end do
       write(20,*)'delta U'
      do i=1,10
       do j=1,nb+1
  write(20,*) 57.3*phb(j),-pa2d(i,j)/100.,u(i,j)-uco2(i,j)
       end do
        end do


      do i=1,10
       do j=1,nb+1
       vav(i,j)=(ra(i,j-1)+ra(i,j))*vav(i,j)*3.14*re*avb(j)
& *(z2d(i,j)-z2d(i+1,j))
       rdw(i,j)=rdw(i,j)*6.28*re*asa(j)
```

```
*        write(30,5001) float(90-10*(j-1)),
*     & za2d(i,j)/1000,vav(i,j)/1e9,rdw(i,j)/1e9
          end do
         end do
        do i=2,9
         do j=1,nb
         vav(i,j)=vav(i-1,j)+vav(i,j)
        end do
        end do
*       write(*,35)'UVP'
         do i=1,10
           do j=1,nb+1
*      write(35,*) 57.3*phb(j),-pa2d(i,j)/100.,uvp(i,j)
          end do
           end do
*        write(35,*) 'TVP'
         do i=1,10
           do j=1,nb+1
*      write(35,*) 57.3*phb(j),-pa2d(i,j)/100.,tvp(i,j)
          end do
           end do


5001    format (f6.1,2x,f8.2,2x,f8.3,2x,f8.3)
4000   format(10(e8.2,1x))
4001   format(10(f5.1,1x))
         goto 1000

1000     continue
         if(npree.eq.1) then
         print*,'s  f',abtot,ftop
         print*,'albedo ',1-abtot/(s0)
      print*,'total kinetic energy', tke,tku,tku+tke
       write(20,*) 'tKE',tke,tku,tku+tke
       write(20,*) 'done'
      write(20,*) 's', abtot,'f',ftop
     print*,'done'
        end if
*************************************************************************
     end
*************************************************************************
* this fitler is a simple triangular filter and is not presently in use
           subroutine filt1(x,nb)
           real x(10,20),tp(10,40)
           integer nb
           do i= 1,10
            tp(i,1)=x(i,2)
            tp(i,nb+2)=x(i,nb-1)
```

```
            do j=1,nb
             tp(i,j+1)=x(i,j)
            end do
            end do
            do i=1,10
            do j=1,nb
            x(i,j)=(tp(i,j)+2*tp(i,j+1)+tp(i,j+2))/4.0
            end do
            end do
            return
            end
*****************************************************************
* Shiparo filter subroutine  see shapiro (1970)
        subroutine shfil(x,nb,ni)
        real x(10,20),tp(10,40)
        integer nb,ni
        do i= ni,ni
         tp(i,1)=2*x(i,1)-x(i,5)
          tp(i,2)=2*x(i,1)-x(i,4)
           tp(i,3)=2*x(i,1)-x(i,3)
            tp(i,4)=2*x(i,1)-x(i,2)
             tp(i,nb+8)=2*x(i,nb)-x(i,nb-4)
              tp(i,nb+7)=2*x(i,nb)-x(i,nb-3)
               tp(i,nb+6)=2*x(i,nb)-x(i,nb-2)
                  tp(i,nb+5)=2*x(i,nb)-x(i,nb-1)
             do j=1,nb
             tp(i,j+4)=x(i,j)
             end do
         end do
*
        do i=ni,ni
        do j=1,nb
        x(i,j)=(186.*tp(i,j+4)+56.*(tp(i,j+3)+tp(i,j+5))-
   & 28.*(tp(i,j+2)+tp(i,j+6))+8.*(tp(i,j+1)+tp(i,j+7))
   & -(tp(i,j)+tp(i,j+8)))/256.
        end do
        end do
        return
        end


***** Supporting Subroutines are below ********
*  this subroutine sets up the model structure it is used only once at the beginning of a run
        subroutine box (nb,pha,phb,th,asa,ava,avb,z1,z,za
   &    ,p,r,rs,u,v,w,wt,h,t,dp,rh)
*   set up of initial grid boxes and pressures/densities
        real th(20),pi,dp,asa(20),avb(20),z1(10),t(10,20)
        real z(10,20),za(10,20),p(10,20),r(10,20),rs(10,20)
```

```
      real phb(20),pha(20),ava(20),rh(20)
      real u(10,20),v(10,20),w(10,20),wt(10,20),h(10,20)
      pi=3.1415926
      re=6.4e6
      dp=(pi-pi/20.)/(float(nb))
      phb(1)=pi/2.-pi/40.
      do 10 j=2,nb+1
      phb(j)=phb(j-1)-dp
10    continue
*
      do j=1,nb
      pha(j)=(phb(j)+phb(j+1))/2.
      end do
      pha(nb+1)=pha(nb)
*
      do 20 j=1,nb+1
*     rh(j)=.75+.1*cos(6*pha(j))
      th(j)=sin(phb(j))
20    continue
*
      do 30 j=1,nb
       asa(j)=re*(th(j)-th(j+1))
30    continue
*
      do 35 j=1,nb
      avb(j)=cos(phb(j))
      ava(j)=cos(pha(j))
35    continue


      do i=1,10
      do j=1,nb+1
      z(i,j)=z1(i)*1000.
      end do
      end do

     do 42 i=10,1,-1
      do 40 j=1,nb+1
      rs(i,j)=(1+rh(j)*esat(t(i,j)))
*     rs(i,j)=1.
      if(i.eq.10)then
      za(i,j)=z1(i)
      else
      ht=287.*t(i,j)/9.8
      za(i,j)=z(i+1,j)-ht*log((exp(-(z(i,j)-z(i+1,j))/ht)+1)/2.)
      end if
*     u(i,j)=0.0
*     v(i,j)=0.0
```

```
          w(i,j)=0.0
          wt(i,j)=0.0
          h(i,j)=0.0
          if(i.eq.10)then
*         p(i,j)=1.013e5
          r(i,j)=p(i,j)/(287*t(i,j)*rs(i,j))
*         p(1,j)=p(2,j)*exp((19000-za(i,j))*.034/255)
*         r(1,j)=p(i,j)/(287*t(i,j))
          else
*         p(i,j)=p(i+1,j)*exp(-(z(i+1,j)-za(i+1,j))*.034/
*    &(t(i+1,j)*rs(i+1,j)))
*    &  *exp(-(za(i,j)-z(i+1,j))*.034/(t(i,j)*rs(i,j)))
          r(i,j)=p(i,j)/(287*t(i,j)*rs(i,j))
          end if
40        continue
42        continue
*         do i=8,1,-1
*          do j=1,nb
*           r(i,j)=(p(i+1,j)-r(i+1,j)*9.8*(z(i+1,j)-za(i+1,j)))/
*    & (287*t(i,j)+9.8*(za(i,j)-z(i+1,j)))
*           p(i,j)=r(i,j)*t(i,j)*287.
*          end do
*          end do

*
          return
          end
*************************************************************************
           subroutine avg(sth,y,nb,ya)
           real phb(20),y(36),ya(20),sth(20)
           real a,sum(20),s1(20)
           integer nb
           do 50 j=1,nb+1
           phb(j)=180/3.1415926*asin(sth(j))
           sum(j)=0.0
           s1(j)=0.0
50         continue
           do 200 i=1,36
           z1=(95-5*(i))*3.1415926/180
           z2=(95-5*(i+1))*3.1415926/180
           a=sin(z1)-sin(z2)
           z1=(92.5-5*(i))
           do 100 j=1,nb
           if(z1.lt.phb(j).and.z1.gt.phb(j+1))then
           sum(j)=sum(j)+a
           s1(j)=s1(j)+a*y(i)
           end if
100        continue
```

```
200       continue
          do 300 j=1,nb
           ya(j)=s1(j)/sum(j)
300       continue
          return
          end
*************************************************************************
* This is the cumulus convection scheme it is based on that of Lindzen et al. 1982 JAS
* V.39 1189-1205 it is not in use right now but could be put to use anytime.
      subroutine cumulus (at,t,za,z,pa,ch,jb,step,ce,c,cpt,lh,sh)
      real at(10),t(10),z(10),za(10),pa(10),ch(10),ce(10)
      real zt,tt,cp,rho0,l,qs,c(10),cpt(10),lh,sh
      real e,q0,lhf,shf,cdus,mc,lq
      integer m,jb
      cdus=1071
      cdus=.0026
      rho0=1294*(273./at(9))
      cp=1.005
       do i=1,9
        ch(i)=0.0
        ce(i)=0.0
       end do
      l=2510-2.38*(at(9)-273)
      e=esat(at(10))
      qs=rwat(1.0,e,1,1,1.0)
*          qs=rwat(1.0,e,1,1)
      e=esat(at(9))
      q0=rwat(.998,e,1,1,.77)
*         q0=rwat(.998,e,1,1)
      lq=l*q0
*       print*,lq
*       lhf=rho0*l*cdus*(qs-q0)
*       shf=rho0*cdus*cp*(at(10)-at(9))
        lhf=lh
        shf=sh
*        print*,sh,shf,lh,lhf
*         shf=rho0*cdus*cp*0.8
      call trpause (at,t,lq,z,zt,tt,m)
      mc=(lhf+shf)/lq
       s1=0.0
      do 8850 i=9,m,-1
         rho=1294*(273/at(i) )*pa(i)
       if(i.eq.m) then
        dt=(lhf+shf-s1)/((z(i)-z(i+1))*cp*mc)

       else
       dt=(t(i)-at(i))/(z(i)-za(i))+9.8/cp
```

```
      end if
       s1=s1+dt*cp*mc*(z(i)-z(i+1))
       ch(i)=cp*mc*dt*(z(i)-z(i+1))*step/cpt(i)
*       ch(i)=cp*mc*dt*step*86400/((1+c(i))*1.005*1000*rho)
*       if(i.eq.9) then
*       dt=(lhf+shf)/((z(i)-z(i+1)))
*       ch(i)=ch(i)+dt*step*86400/((1+c(i))*1000*rho)
*       end if
*      print*,i,ch(i),jb,m
8850   continue
*       print*
       do i=1,9
          at(i)=at(i)+ch(i)
       end do
*        at(10)=at(10)-(lhf+shf)*step/cpt(10)
*         print*,(lhf+shf)*step/cpt(10),s1,lhf+shf
*         print*,lhf,shf,lhf+shf,m,qs-q0,at(9),at(10),ch(m)
     return
     end
*****************************************************************
     subroutine trpause(at,t,lq,z,zt,tt,m)
     real at(10),z(10),t(10)
     real zt,tt,y,x,lq
     integer m
       cp=1.005
      do 8860 i=9,1,-1
      y=lq+cp*t(10)-cp*t(i)
      x=9.81*z(i)
     if (y.lt.x) then
      tt=t(i)
      zt=z(i)
      m=i
      goto 8870
     end if
8860   continue
8870     return
       end
*****************************************************************************
*  This is the main dynamics part of the program**
       subroutine dyn(u,v,t,p,r,rs,z,za,pha,phb,step
    & ,dstep,uvp,nb,xt,xr,asa,ava,avb,c2,pb,tr3,tav,rav,
    & doc,sk,tvp,pav,rh,twp,qwp,cwat)
      real u(10,20),v(10,20),w(10,20),t(10,20),p(10,20),dr(10,20)
       real r(10,20),z(10,20),step,dt(10,20),sum,re
       real ra(10,20),wt(10,20),dw(10,20),dpr(10,20),za(10,20)
       real phb(20),yk,xt(10),pb(10,20),tt(10,20)
      real dp(10,20),pr(10,20),dv(10,20),du(10,20),xr(10),pt(10,20)
       real vt(10,20),ut(10,20),rs(10,20),c2(10,20),cwat(10,20)
```

```
      real gv(10,20),rt(10,20),cp(10,20),tvp(10,20),twp(10,20)
      real tx(10,20),ux(10,20),vx(10,20),rx(10,20),rh(20)
      real pha(20),asa(20),ava(20),avb(20),uvp(10,20)
      real omega,fc(20),st,dstep,tr3,saveg
      real tav(10,20),rav(10,20),pav(10,20),sk(20)
      real dmp,doc,sav(10,20),qwp(10,20)
       integer nh
       time=0.0
        yk=0.0
        nh=0
       dmp=1.
       rl=24.*step
       st=3600.*dstep
       re=6.4e6
       omega=6.28/86400
       nx=0
* set up initial conditions
      do 35 j=1,nb+1
       v(1,j)=0.0
      fc(j)=2.*omega*sin(phb(j))
      pb(1,j)=0.0
       do i=1,10
       cp(i,j)=(1+c2(i,j))*1005.
       w(i,j)=0.0
       dr(i,j)=0.
       dt(i,j)=0.0
       dv(i,j)=0.
       tav(i,j)=0.0
        sav(i,j)=0.0
        rav(i,j)=0.0
         pav(i,j)=0.0
       du(i,j)=0.
*       dw(i,j)=dw(i,j)*dstep/(step*24.)
      dw(i,j)=0.0
     pr(i,j)=0.0
      end do
35     continue
      saveg=0.0
       do i=1,10
        cp(i,nb+1)=cp(i,nb)
       end do
       do j=1,nb
       cp(10,j)=doc*1e7
       end do

         call avdens(r,ra,nb)
      call vsw(t,r,u,v,tt,rt,ut,vt,nb)
*
```

```
      call uvel(w,u,v,du,pha,phb,avb,re,r,ra,dr,pr,gv,
     &  fc,xt,xr,nb,z,za,uvp)
      call vvel(p,w,u,v,dv,pha,phb,avb,re,r,ra,dr,pr,gv
     &,fc,xt,xr,nb,yk,z,za)
       do i=1,9
        do j=1,nb
         gv(i,j)=.5*(r(i,j-1)+r(i,j))*v(i,j)*avb(j)
*          if(j.eq.2) then
*           gv(i,j)=gv(i,j)/2.
*           end if
        end do
       end do
       call masst(gv,phb,r,dr,ava,re,nb,z,w)
      call heattr(r,p,t,w,v,cp,pha,phb,gv,nb,
     &  re,dt,asa,z,za,xr,xt,u,sk,tvp,twp,rh,qwp,cwat)
       sum=0.0
*     call hy(nb,p,t,r,rs,pr,dt,dr,w,dw,.67*st,dpr,dmp
*     & ,z,za,c2,pb,v,dv)
      call uvinc(v,u,dv,du,0.67*st,nb,r,dr,t,dt,tt,rt,vt,ut)
       do i=1,9
       do j=1,nb
       pr(i,j)=0.0
         dpr(i,j)=0.0
       sum=sum+dr(i,j)*asa(j)
       p(i,j)=287.*t(i,j)*r(i,j)*rs(i,j)
        if(j.eq.nb) then
*         p(i,j+1)=p(i,j)
        end if
       end do
       end do
        call avdens(r,ra,nb)
      call uvel(w,u,v,du,pha,phb,avb,re,r,ra,dr,pr,gv,
     &  fc,xt,xr,nb,z,za,uvp)
      call vvel(p,w,u,v,dv,pha,phb,avb,re,r,ra,dr,pr,gv
     &,fc,xt,xr,nb,yk,z,za)
       do i=1,9
        do j=1,nb
         gv(i,j)=.5*(r(i,j-1)+r(i,j))*v(i,j)*avb(j)
*          if(j.eq.2) then
*           gv(i,j)=gv(i,j)/2.
*           end if
        end do
       end do
       call masst(gv,phb,r,dr,ava,re,nb,z,w)
      call heattr(r,p,t,w,v,cp,pha,phb,gv,nb,
     &  re,dt,asa,z,za,xr,xt,u,sk,tvp,twp,rh,qwp,cwat)
       sum=0.0
*     call hy(nb,p,t,r,rs,pr,dt,dr,w,dw,st,dpr,dmp
```

```
*    &    ,z,za,c2,pb,v,dv)
     call uvinc(v,u,dv,du,st,nb,r,dr,t,dt,tt,rt,vt,ut)
        do i=1,9
        do j=1,nb
        pr(i,j)=0.0
          dpr(i,j)=0.0
        sum=sum+dr(i,j)*asa(j)
        p(i,j)=287.*t(i,j)*r(i,j)*rs(i,j)
         if(j.eq.nb) then
*         p(i,j+1)=p(i,j)
         end if
        end do
        end do
       time=time+dstep
      do i=1,9
       do j=1,nb+1
       tav(i,j)=tav(i,j)+v(i,j)*dstep
        rav(i,j)=rav(i,j)+u(i,j)*dstep
        sav(i,j)=sav(i,j)+t(i,j)*dstep
         pav(i,j)=pav(i,j)+p(i,j)*dstep
      end do
      end do
       saveg=saveg+dstep
**  calculations
*
*
55       continue
      call vsw(t,r,u,v,tx,rx,ux,vx,nb)
       call avdens(r,ra,nb)
      call uvel(w,u,v,du,pha,phb,avb,re,r,ra,dr,pr,gv,
   &   fc,xt,xr,nb,z,za,uvp)
      call vvel(p,w,u,v,dv,pha,phb,avb,re,r,ra,dr,pr,gv
   &,fc,xt,xr,nb,yk,z,za)
       do i=1,9
        do j=1,nb
         gv(i,j)=.5*(r(i,j-1)+r(i,j))*v(i,j)*avb(j)
*          if(j.eq.2) then
*           gv(i,j)=gv(i,j)/2.
*           end if
        end do
        end do
        call masst(gv,phb,r,dr,ava,re,nb,z,w)
       call heattr(r,p,t,w,v,cp,pha,phb,gv,nb,
   &   re,dt,asa,z,za,xr,xt,u,sk,tvp,twp,rh,qwp,cwat)
       sum=0.0
*      call hy(nb,p,t,r,rs,pr,dt,dr,w,dw,2.*st,dpr
*    & ,dmp,z,za,c2,pb,v,dv)
      if(time.ge.rl/2.0.and.nh.eq.0) then
```

```
      nh=1
*     call hy(nb,p,t,r,rs,pr,dt,dr,w,dw,2.*st,dpr
*    & ,dmp,z,za,c2,pb,v,dv)
*       do i=1,9
*       write(*,2001) (dt(i,j)*st*10.,j=1,nb+1)
*        end do
       end if
2001    format (10(f7.3,1x))
      call uvinc(v,u,dv,du,2.*st,nb,r,dr,t,dt,tt,rt,vt,ut)
       call vsw(tx,rx,ux,vx,tt,rt,ut.vt,nb)
         do i=1,9
         do j=1,nb
         pr(i,j)=0.0
           dpr(i,j)=0.0
         sum=sum+dr(i,j)*asa(j)
         p(i,j)=287.*t(i,j)*r(i,j)*rs(i,j)
*     pb(i+1,j)=pb(i,j)+r(i,j)*9.8*(z(i,j)-z(i+1,j))*rs(i,j)
      pb(i+1,j)=p(i,j)+r(i,j)*9.8*(za(i,j)-z(i+1,j))*rs(i,j)
          if(j.eq.nb) then
*          p(i,j+1)=p(i,j)
*           pb(i+1,j+1)=pb(i+1,j)
          end if
         end do
         end do

       do i=1,9
         do j=1,nb+1
       tav(i,j)=tav(i,j)+v(i,j)*dstep
        rav(i,j)=rav(i,j)+u(i,j)*dstep
        pav(i,j)=pav(i,j)+p(i,j)*dstep
        end do
        end do
        saveg=saveg+dstep
       time=time+dstep

          if(time.lt.rl-dstep/10.)then
         goto55
       end if

       do i=1,9
         do j=1,nb+1
       tav(i,j)=tav(i,j)/saveg
*         t(i,j)=tav(i,j)
       rav(i,j)=rav(i,j)/saveg
*          r(i,j)=rav(i,j)
       pav(i,j)=pav(i,j)/saveg
*          p(i,j)=pav(i,j)
       du(i,j)=0.0
```

```
          dv(i,j)=0.0
          end do
          end do
*          print*,sum,'sum'
*         call hy(nb,pav,tav,rav
*     & ,rs,pr,dt,dr,w,dw,2.,dpr,z,za,c2,pb)
          call uvinc(v,u,dv,du,2.,nb,r,dr,t,dt,t,r,v,u)
           do j=1,nb
*      v(1,j)=v(1,j)
*     & -xr(7)*tav(1,j)*sqrt(tav(1,j)**2+rav(1,j)**2)
*     & -xr(7)*v(1,j)*sqrt(v(1,j)**2+u(1,j)**2)
*     & *step*86400.
      u(1,j)=u(1,j)
*     & -xr(8)*rav(1,j)*sqrt(tav(1,j)**2+rav(1,j)**2)
     & -xr(8)*u(1,j)*sqrt(v(1,j)**2+u(1,j)**2)
     & *step*86400.
      v(2,j)=v(2,j)
*     & -xr(9)*rav(2,j)*sqrt(tav(2,j)**2+rav(2,j)**2)
     & -xr(9)*v(2,j)
     & *step*86400.
*
      v(9,j)=v(9,j)
*     &-xr(2)*tav(9,j)*sqrt(tav(9,j)**2+rav(9,j)**2)
     &-xr(2)*v(9,j)*sqrt(v(9,j)**2+u(9,j)**2)
     & *step*86400.*2./(z(9,j-1)+z(9,j))
      u(9,j)=u(9,j)
*     &-xr(1)*rav(9,j)*sqrt(tav(9,j)**2+rav(9,j)**2)
     &-xr(1)*u(9,j)*sqrt(v(9,j)**2+u(9,j)**2)
     & *step*86400.*2./(z(9,j-1)+z(9,j))
        end do
         u(9,1)=u(9,2)
          u(9,nb+1)=u(9,nb)
         return
        end
*************************************************************************
* part of dynamics average density between grid points calculation
        subroutine avdens(r,ra,nb)
        real r(10,20),ra(10,20)
        integer nb
        do i=2,9
         do j=1,nb
         if(j.gt.1)then
         ra(i,j)=.25*(r(i,j)+r(i,j-1)+r(i-1,j)+r(i-1,j-1))
          else
            ra(i,j)=.5*(r(i,j)+r(i-1,j))
         end if

         end do
```

```
      ra(10,i)=.5*(r(10,i-1)+r(10,i))
      ra(1,i)=.5*(r(1,i)+r(1,i-1))
      ra(i,nb+1)=.5*(r(i,nb)+r(i-1,nb))
      ra(i,1)=.5*(r(i-1,1)+r(i,1))
      end do
      return
      end
*****************************************************************
*part of dynamics save old value of v,u,r,w..
      subroutine uvset(w,wt,dw,u,v,ut,vt,r,p,rt,pt,dr,dp,du
     &   ,dv,nb,step,pr,dpr,tt,t)
      real u(10,20),v(10,20),ut(10,20),vt(10,20),du(10,20)
      real dv(10,20),step,dp(10,20),dr(10,20),pt(10,20)
     real rt(10,20),r(10,20),p(10,20),wt(10,20),w(10,20),dw(10,20)
      real pr(10,20),dpr(10,20),tt(10,20)
      integer nb
      do 96 i=1,9
       do 94 j=1,nb+1
       vt(i,j)=v(i,j)+0.0*dv(i,j)*step
       ut(i,j)=u(i,j)+0.0*du(i,j)*step
       wt(i,j)=w(i,j)+0.0*dw(i,j)
       rt(i,j)=r(i,j)+.0*(dr(i,j)+pr(i,j))*step
       tt(i,j)=t(i,j)
       pt(i,j)=p(i,j)+.0*(dp(i,j))
       dv(i,j)=0.
       du(i,j)=0.
94      continue
96      continue
       return
       end
*****************************************************************
*  part of dynamics mass transfer eqn of continuity
       subroutine masst(gv,phb,r,dr,ava,re,nb,z,w)
       real  gv(10,20),phb(20),r(10,20),dr(10,20),z(10,20)
       real ava(20),re
       integer nb
       do 64 j=2,9
       do 62 j=1,nb
          zs=z(i,j)-z(i+1,j)
          dr(i,j)=(+gv(i,j+1)-gv(i,j))
   & /(re*ava(j)*(phb(j)-phb(j+1)))
*    & +(w(i+1,j)*(r(i+1,j)+r(i,j))-w(i,j)*(r(i,j)+r(i-1,j)))/(4.*zs)
62      continue
64      continue
*
       return
       end
*****************************************************************
```

```
*  part of dynamics increment prognostic variables
      subroutine uvinc(v,u,dv,du,step,nb,r,dr,t,dt,tt,rt,vt,ut)
       real v(10,20),u(10,20),dv(10,20),du(10,20)
       real r(10,20),dr(10,20),t(10,20),dt(10,20)
        real vt(10,20),ut(10,20),rt(10,20),tt(10,20)
       real step
       integer nb
       do 104 i=1,10
        do 102 j=1,nb+1
         v(i,j)=vt(i,j)+dv(i,j)*step
         u(i,j)=ut(i,j)+du(i,j)*step
         r(i,j)=rt(i,j)+dr(i,j)*step
         t(i,j)=tt(i,j)+dt(i,j)*step
*          v(i,j)=0.0
*           u(i,j)=0.0
*          u(9,j)=u(9,j)/2.
102      continue
        v(i,nb+1)=0.
*          u(i,nb+1)=2.0*u(i,nb)-u(i,nb-1)
         u(i,nb+1)=u(i,nb)/2.
*          u(i,nb+1)=0
        r(i,nb+1)=r(i,nb)
         t(i,nb+1)=t(i,nb)
         v(i,1)=0.
*          v(i,10)=0.9*v(i,10)
*           u(i,1)=2.*u(i,2)-u(i,3)
           u(i,1)=u(i,2)/2.
*          u(i,1)=0.0
104      continue
*   Antartic land mass
*            v(8,nb)=0.0
*            v(9,nb)=0.0
*            v(9,nb-1)=0.0
*            u(8,nb)=0.0
*            u(9,nb)=0.0
*            u(9,nb-1)=0.0
*  No meridional flow in the two layers of the stratosphere
         do j=1,nb
*             v(2,j)=v(3,j)*0.0
*              v(2,j)=0.0*v(3,j)
           v(1,j)=v(2,j)*0.5
          end do
         return
         end
*****************************************************************
*  part of dynamics zonal velocity equation of motion
     subroutine uvel(w,u,v,du,pha,phb,avb,re,r,ra,dr,pr,gv,f
    &     ,xt,xr,nb,z,za,uvp)
```

```
      real u(10,20),v(10,20),du(10,20),r(10,20),ra(10,20),w(10,20)
      real pr(10,20),gv(10,20),dr(10,20),dru(10,20),dpr(10,20)
      real pha(20),phb(20),f(20),xt(10),avb(20),xr(10),z(10,20)
      real za(10,20),zp,zm,zs,uvp(10,20)
      real re,dz
      integer nb
      dz=2000.
      do i=1,9
        do j=2,nb
     zp=((za(i,j-1)-za(i+1,j-1))+(za(i,j)-za(i+1,j)))/2.
     zm=((za(i-1,j-1)-za(i,j-1))+(za(i-1,j)-za(i,j)))/2.
     zs=((z(i,j-1)-z(i+1,j-1))+(z(i,j)-z(i+1,j)))/2.0
        if(j.eq.2)then
        dru(i,j)=((gv(i,j+1)+gv(i,j))*(u(i,j+1)+u(i,j)))/
    & (4*re*avb(j)*(pha(j-1)-pha(j)))
        end if
        if(j.eq.nb)then
        dru(i,j)=-(gv(i,j-1)+gv(i,j))*(u(i,j-1)+u(i,j))/
    & (4*re*avb(j)*(pha(j-1)-pha(j)))
         end if
         if(j.gt.2.and.j.lt.nb)then
        dru(i,j)=((gv(i,j+1)+gv(i,j))*(u(i,j+1)+u(i,j))-
    & (gv(i,j-1)+gv(i,j))*(u(i,j-1)+u(i,j)))/
    & (4*re*avb(j)*(pha(j-1)-pha(j)))
         end if
        dru(i,j)=dru(i,j)
*    & +((r(i+1,j)+r(i,j))*w(i+1,j)*(u(i,j)+u(i+1,j))-
*    &  (r(i,j)+r(i-1,j))*w(i,j)*(u(i,j)+u(i-1,j)))/(4.*zs)
    &+v(i,j)*(r(i,j-1)+r(i,j))/2.*(f(j)+u(i,j)*tan(phb(j))/re)
    & +(xt(i)*ra(i,j)*(u(i-1,j)-u(i,j))/zm+xt(i+1)*ra(i+1,j)*
    & (u(i+1,j)-u(i,j))/zp)/(zs)

*       if(i.eq.9)then
*        dru(i,j)=dru(i,j)-xr(2)*(r(9,j)+r(9,j-1))/2.*sqrt(v(i,j)**2+
*    &u(i,j)**2)*u(i,j)/zs
*        end if
*       if(i.eq.2)then
*        dru(i,j)=dru(i,j)-xr(8)
*    &*sqrt(v(i,j)**2+u(i,j)**2)*u(i,j)*(r(i,j)+r(i,j-1))/2.
*     end if

      dru(i,j)=dru(i,j)-(r(i,j-1)*(uvp(i,j)+uvp(i,j-1))
    & *(avb(j)+avb(j-1))
    &  -(uvp(i,j)+uvp(i,j+1))*(avb(j)+avb(j+1))*r(i,j))/
    & (4.*re*avb(j)*(pha(j-1)-pha(j)))
    &  +uvp(i,j)*(r(i,j-1)+r(i,j))/2.*(tan(phb(j))/re)

        dpr(i,j)=(dr(i,j)+dr(i,j-1))/2.
```

```
*            dpr(i,j)=0.0
          du(i,j)=2.0*(dru(i,j)-u(i,j)*dpr(i,j))/(r(i,j)+r(i,j-1))
          end do
          end do
         return
           end
*****************************************************************
*  part of dynamics meridional velocity equation of motion
      subroutine vvel(p,w,u,v,dv,pha,phb,avb,re,
     &     r,ra,dr,pr,gv,f,xt,xr,nb,yk,z,za)
         real u(10,20),v(10,20),dv(10,20),r(10,20),ra(10,20)
     &     ,w(10,20),p(10,20),z(10,20),za(10,20)
         real pr(10,20),gv(10,20),dr(10,20),drv(10,20),dpr(10,20)
         real pha(20),phb(20),f(20),xt(10),avb(20),xr(10)
         real re,dz,zp,zm,zs
         integer nb
         dz=2000.
         do i=1,9
           do j=2,nb
       zp=((za(i,j-1)-za(i+1,j-1))+(za(i,j)-za(i+1,j)))/2.
       zm=((za(i-1,j-1)-za(i,j-1))+(za(i-1,j)-za(i,j)))/2.
       zs=((z(i,j-1)-z(i+1,j-1))+(z(i,j)-z(i+1,j)))/2.0
            if(j.eq.2) then
            drv(i,j)=(gv(i,j+1)+gv(i,j))*(v(i,j+1)+v(i,j))/
     &(4*re*avb(j)*(pha(j-1)-pha(j)))
            end if
            if (j.eq.nb)then
            drv(i,j)=-(gv(i,j-1)+gv(i,j))*(v(i,j-1)+v(i,j))/
     &(4*re*avb(j)*(pha(j-1)-pha(j)))
            end if
             if (j.gt.2.and.j.lt.nb)then
            drv(i,j)=((gv(i,j+1)+gv(i,j))*(v(i,j+1)+v(i,j))-
     & (gv(i,j-1)+gv(i,j))*(v(i,j-1)+v(i,j)))/
     &(4*re*avb(j)*(pha(j-1)-pha(j)))
            end if
            drv(i,j)=drv(i,j)
*     & +((r(i+1,j)+r(i,j))*w(i+1,j)*(v(i,j)+v(i+1,j))-
*     &   (r(i,j)+r(i-1,j))*w(i,j)*(v(i,j)+v(i-1,j)))/(4.*zs)
     &  -u(i,j)*(r(i,j-1)+r(i,j))/2.*(f(j)+u(i,j)*tan(phb(j))/re)
     &  -(p(i,j-1)-p(i,j))/(re*(pha(j-1)-pha(j)))
     & +(xt(i)*ra(i,j)*(v(i-1,j)-v(i,j))/zm+xt(i+1)*ra(i+1,j)*
     & (v(i+1,j)-v(i,j))/zp)/(zs)
     & -(r(i,j)+r(i,j-1))/2.*
     & yk*v(i,j)*sqrt(v(i,j)**2.+u(i,j)**2.)

*       if(i.eq.9)then
*            drv(i,j)=drv(i,j)-xr(1)*(r(9,j)+r(9,j-1))/2.
*     &*sqrt(v(i,j)**2+u(i,j)**2)*v(i,j)/zs
```

```
*       end if
*            if(i.eq.2)then
*        drv(i,j)=drv(i,j)-xr(8)
*     &*sqrt(v(i,j)**2+u(i,j)**2)*v(i,j)*(r(i,j)+r(i,j-1))/2.
*        end if


          dpr(i,j)=(dr(i,j)+dr(i,j-1))/2.
*          dpr(i,j)=0.0

          dv(i,j)=2.0*(drv(i,j)-v(i,j)*dpr(i,j))/(r(i,j)+r(i,j-1))
          end do
          end do
          return
          end
*****************************************************************
*  part of dynamics first law of thermodynamics for horizontal advection
        subroutine heattr(r,p,t,w,v,cp,pha,phb,gv,nb,
     &    re,dt,asa,z,za,xr,xt,u,sk,tvp,twp,rh,qwp,cwat)
        real r(10,20),p(10,20),t(10,20),w(10,20),rh(20)
        real v(10,20),cp(10,20),pha(20),phb(20),gv(10,20)
        real gt(10,20),vp(10,20),za(10,20),tvp(10,20)
        real dt(10,20),asa(20),z(10,20),dt1(10,20),cwat(10,20)
       real xt(10),xr(10),u(10,20),sk(10),twp(10,20),qwp(10,20)
*  gt is temporary gv
        real kappa,cpa,ta,rav,zt,zs,zr,drt,cps,rq,paq
         real e,lv
        integer nb
        kappa=287/1005

*        sum=0.
        do i=1,10
        do j=1,nb
        dt(i,j)=0.0
*         if (j.eq.1) then
*         gt(i,1)=0.0
*         vp(i,1)=0.0
*         gt(i,nb+1)=0.0
*         vp(i,nb+1)=0.0
*         else
*         gt(i,j)=gv(i,j)
*         vp(i,j)=v(i,j)
*         end if
*         ath(i,j)=t(i,j)/(p(i,j)/1.0e5)**kappa
        end do
        end do
        do i=1,9
        do j=1,nb-1
```

```
        zt=(z(i,j+1)-z(i+1,j+1))
        zs=(z(i,j)-z(i+1,j))
        zr=(zs+zt)/2.
        ta=(t(i,j)+t(i,j+1))/2.
         paq=(p(i,j)+p(i,j+1))/2.0/1.0e5
         e=esat(ta)
          lv=1000.*(2510-2.38*(ta-273))
*         rq=rwat(paq,e,1,1,(rh(j)+rh(j+1))/2.)
        rq=(cwat(i,j+1)+cwat(i,j))/2.
        cpa=(cp(i,j)+cp(i,j+1))/2.
        rav=(r(i,j)+r(i,j+1))/2.
       drt=(1.0*287+1005.)
     & *rav*ta*v(i,j+1)*zr*cos(phb(j+1))
     &  +(cpa)*rav*tvp(i,j+1)*zr*cos(phb(j+1))
     &  +lv*rq*v(i,j+1)*rav*zr*cos(phb(j+1))
      dt1(i,j)=+drt/(zs*asa(j)*r(i,j)*cp(i,j))
      dt1(i,j+1)=-drt/(zt*asa(j+1)*r(i,j+1)*cp(i,j+1))
        dt(i,j)=dt(i,j)+dt1(i,j)
        dt(i,j+1)=dt(i,j+1)+dt1(i,j+1)
*       sum=sum+r(i,j)*dt(i,j)*zs*cp(i,j)*(phb(j)-phb(j+1))*ava(j)
*     & +r(i,j+1)*dt(i,j+1)*zt*cp(i,j+1)*(phb(j+1)-phb(j+2))*ava(j+1)
        end do
        end do
       do i=3,9
        do j=1,nb
          zm=za(i-1,j)-za(i,j)
         zs=(z(i-1,j)-z(i,j))
         zt=(z(i,j)-z(i+1,j))
         rm=(r(i,j)+r(i-1,j))/2.
          cpa=(cp(i-1,j)+cp(i,j))/2.0
         drt=rm*twp(i,j)*1005.
         drt=drt+rm*(cpa-1005.)*qwp(i,j)
        dt1(i,j)=-drt/(zt*r(i,j)*cp(i,j))
        dt1(i-1,j)=+drt/(zs*r(i-1,j)*cp(i-1,j))
         dt(i,j)=dt(i,j)+dt1(i,j)
         dt(i-1,j)=dt(i-1,j)+dt1(i-1,j)
        end do
        end do
*       end if
*
*        print*,sum
1000    format (10(e7.2,2x))
       return
       end
*****************************************************************
*  part of dynamics        subroutine vsw(t,r,u,v,tx,rx,ux,vx,nb)
       real t(10,20),r(10,20),u(10,20),v(10,20)
       real tx(10,20),rx(10,20),ux(10,20),vx(10,20)
```

```
      integer nb
       do i=1,10
        do j=1,nb+1
       tx(i,j)=t(i,j)
       rx(i,j)=r(i,j)
       vx(i,j)=v(i,j)
       ux(i,j)=u(i,j)
       end do
       end do
      return
       end
**End of dynamics subroutine *****
*************************************************************************
** calculate IR fluxes of radiation
    subroutine flux (em,p,at,kap1,ac1,kap2,ac2,kap3,ac3,c,dc,f,
   > ire1,ire2,ire3)
    real trt(10,10),at(10)
    real em(10,10),c(10),cl(10,10),tr(10)
    real fd(10),fu(10),ftot(10),dc(10),f(10),p(10)
    real ac1,ac2,ac3,cf,sig,ire1,ire2,ire3
    integer kap1,kap,kap2,kap3,nt
    kap=kap1
    sig=5.67e-8
     do 203 i=1,10
        tr(i)=1.0
203   continue
     tr(kap1)=(1.0-ire1*ac1)
     tr(kap2)=(1.0-ac2*ire2)
     tr(kap3)=(1.0-ac3*ire3)
     do 206 i=1,10
        em(i,i+1)=em(i,i)
        do 205 j=i+1,10
        cl(i,j)=1.0
         do 204 k=i,j-1
        cl(i,j)=cl(i,j)*tr(k)
204     continue
        em(i,j)=em(i,j)*cl(i,j)+(1-cl(i,j))
        em(j,i)=em(i,j)
205     continue
        em(i,i)=0.0
206     continue
*
    trt(10,10)=-1.0
    do 240 i=1,9
   do 239 j=1,10
    if(i.eq.j)then
        if(i.eq.1)then
        trt(i,i)=-2.0*em(i,i+1)
```

```
          else
          trt(i,j)=-2*em(i,i+1)
      end if
      end if
          if(i.lt.j.and.j.lt.10)then
            trt(i,j)=(em(j+1,i+1)-em(j,i+1))-(em(j+1,i)-em(j,i))
          end if
          if(j.eq.10)then
          trt(i,j)=(em(10,i))-(em(10,i+1))
          end if
          if(i.gt.j) then
           trt(i,j)=(em(i,j)-em(i+1,j))-(em(i,j+1)-em(i+1,j+1))
          end if
239       continue
          trt(10,i)=(em(i,10)-em(i+1,10))
240       continue
        fd(1)=0.0
*         write(20,4050) ((trt(j,i),j=1,10),i=1,10)
          sig=5.67e-8
          fu(1)=(1.0-em(10,1))*sig*at(10)**4+sig*em(1,2)*at(1)**4
          fu(4)=(1.0-em(10,4))*sig*at(10)**4
      do 4043 j=1,10
          ftot(j)=0.0
          do 4041 i=1,10
           ftot(j)=ftot(j)+trt(j,i)*sig*at(i)**4
4041      continue
          if(j.gt.1.and.j.lt.10) then
          fu(1)=fu(1)+sig*(em(1,j+1)-em(1,j))*at(j)**4
          end if
          if(j.lt.10)then
          fu(4)=fu(4)+sig*(em(4,j+1)-em(4,j))*at(j)**4
          end if
4043      continue
          ftot(10)=ftot(10)+sig*at(10)**4
      do 4046 i=1,9
           dc(i)=.0083224*(ftot(i))/(p(i+1)-p(i))
4046   continue
      fd(10)=ftot(10)
      f(10)=fd(10)
*         print*,'10',fd(10)
*         print*,'1',fu(1)
      f(1)=fu(1)
4050   format (10(f8.5,2x))
*           write(20,*) 'em '
*           write(20,4050) ((em(j,i),i=1,10),j=1,10)
*           write(20,*) 'hi'
*           write(20,4050) ((trt(j,i),i=1,10),j=1,10)
*           write(20,*) (dc(i),i=1,10)
```

```
*          write(20,*) fd(10)
*
      return
      end
*************************************************************************
* hydrostatic adjustment**
       subroutine hys(nb,p,t,r,rs,pr,w,rdw,dw,step,dpr,z,za,c,pb,
     & v,u,ps,cwat)
            real p(10,20),t(10,20),w(10,20),pr(10,20),rs(10,20)
            real r(10,20),dw(10,20),dpr(10,20),cp(10,20),c(10,20)
            real z(10,20),za(10,20),step,pb(10,20),lv,thp(10,20)
            real sb(10),st(10),cb(10),ct(10),cm(8,8),rd(8),rd1(7)
            real zp(10),zm(10),z1(10),ga(10),z2(10),cwat(10,20)
          real v(10,20),u(10,20),rdw(10,20),dmp,ps(20),cm1(7,7)
            integer nb
          dmp=21600/step/2.
*           dmp=86400/step
            dz=2000.
             do 50 i=1,9
            do 49 j=1,nb
          thp(i,j)=t(i,j)*(p(9,j)/p(i,j))**(.287)
            cp(i,j)=(1.+c(i,j))
*           cp(i,j)=1005.
              pr(i,j)=r(i,j)
            dpr(i,j)=p(i,j)
            dw(i,j)=w(i,j)
49          continue
50          continue
          do j=1,nb
           ps(j)=p(10,j)-r(9,j)*9.8*za(9,j)
*           r(9,j)=p(9,j)/(287.*t(9,j))
          end do
*
            do 86 j=1,nb
          pb(1,j)=0.0
             do i=1,9
            zm(i)=za(i,j)-z(i+1,j)
            zp(i)=z(i,j)-za(i,j)
            z1(i)=z(i,j)-z(i+1,j)
           z2(i)=za(i,j)-za(i+1,j)
            end do
*
           do 84 i=1,9
          ct(i)=(0.0*(t(i,j)+t(i-1,j))+
     & (t(i-1,j)+t(i,j))/2.+.0098*z2(i-1)/2.)/z1(i)/r(i,j)/
*     & (thp(i-1,j)+thp(i,j))/2.)/z1(i)/r(i,j)/
     & cp(i,j)
          cb(i)=(0.0*(t(i,j)+t(i+1,j))+
```

```
     & (t(i+1,j)+t(i,j))/2.-.0098*z2(i)/2.)/z1(i)/r(i,j)/
*    & (thp(i+1,j)+thp(i,j))/2.)/z1(i)/r(i,j)/
     & cp(i,j)
       st(i)=287.*rs(i,j)*(t(i,j)/z1(i)+r(i,j)*ct(i))
       sb(i)=287.*rs(i,j)*(t(i,j)/z1(i)+r(i,j)*cb(i))
84        continue
          do i=1,8
          rd(i)=p(i+1,j)-p(i,j)-(r(i+1,j)+r(i,j))*z2(i)*4.9
          end do
          do i=2,7
          cm(i,i+1)=-sb(i+1)+9.8*z2(i)/(2.*z1(i+1))
          cm(i,i)=sb(i)+st(i+1)+9.8*z2(i)/2.*
     & (1/z1(i)-1/z1(i+1))
          cm(i,i-1)=-st(i)-9.8*z2(i)/(2.0*z1(i))
          end do
          cm(1,1)=sb(1)+st(2)
          cm(1,2)=-sb(2)
          cm(8,7)=-st(7)
          cm(8,8)=st(9)+sb(8)
*        cm(2,2)=sb(2)+st(3)
*        cm(2,3)=-sb(3)
1002     format (3x,4(f10.4,3x))
*        do i=1,7
*        rd1(i)=rd(i+1)
*         do k=1,7
*        cm1(i,k)=cm(i+1,k+1)
*         end do
*         end do
*
*
          call sle(cm,rd,8,8,.000001)
*        call sle(cm1,rd1,7,7,0.0000010)
*        do i=2,8
*          rd(i)=rd1(i-1)
*         end do
*          rd(1)=0.0

*
*        if(j.eq.nb) then
*         rd(8)=0.0
*          rd(7)=0.
*         end if
*        rd(1)=0.
*         rd(2)=0.0
*          rd(1)=0.2*rd(1)
*          rd(2)=0.6*rd(2)
*          rd(8)=rd(8)+2.*(p(9,j)-ps(j))/st(9)
           do i=1,8
```

```
        rd(i)=rd(i)/dmp
          ta=(t(i,j)+t(i,j+1))/2.
            lv=1000.*(2510-2.38*(ta-273))
            t(i,j)=t(i,j)+rd(i)*cb(i)
*           t(i,j)=t(i,j)+rd(i)*cb(i)+
*     & lv*(cwat(i,j)+cwat(i+1,j))*.5*rd(i)/cp(i,j)/1005.
*     & /r(i,j)/z1(i)
            t(i+1,j)=t(i+1,j)-rd(i)*ct(i+1)
*           t(i+1,j)=t(i+1,j)-rd(i)*ct(i+1)-
*     & lv*(cwat(i,j)+cwat(i+1,j))*.5*rd(i)/cp(i+1,j)/1005.
*     & /r(i+1,j)/z1(i+1)

            r(i,j)=r(i,j)+rd(i)/z1(i)
            r(i+1,j)=r(i+1,j)-rd(i)/z1(i+1)
       if(j.gt.1) then
       v(i,j)=v(i,j)+rd(i)*(v(i,j)+v(i+1,j))/
      &(2*z1(i)*r(i,j))/2.0
       v(i+1,j)=v(i+1,j)-rd(i)*(v(i,j)+v(i+1,j))/
      & (2*z1(i+1)*r(i+1,j))/2.0
          end if

       if(j.lt.nb) then
       v(i,j+1)=v(i,j+1)+rd(i)*(v(i,j+1)+v(i+1,j+1))/
      &(2*z1(i)*r(i,j+1))/2.
       v(i+1,j+1)=v(i+1,j+1)-rd(i)*(v(i,j+1)+v(i+1,j+1))/
      & (2*z1(i+1)*r(i+1,j+1))/2.
       end if
*       if(i.eq.8.and.j.eq.8) then
*       print*,rd(i)*(v(i,j)+v(i+1,j))/(2*z1(i)*r(i,j)),v(i,j)
*     & ,v(i+1,j)
*       end if

       if(j.gt.1) then
       u(i,j)=u(i,j)+rd(i)*(u(i,j)+u(i+1,j))/
      &(2*z1(i)*r(i,j))/2.
       u(i+1,j)=u(i+1,j)-rd(i)*(u(i,j)+u(i+1,j))/
      & (2*z1(i+1)*r(i+1,j))/2.
       end if

          if(j.lt.nb) then
       u(i,j+1)=u(i,j+1)+rd(i)*(u(i,j+1)+u(i+1,j+1))/
      &(2*z1(i)*r(i,j+1))/2.0
       u(i+1,j+1)=u(i+1,j+1)-rd(i)*(u(i,j+1)+u(i+1,j+1))/
      & (2*z1(i+1)*r(i+1,j+1))/2.0
          end if

*         dt(i,j)=dt(i,j)+rd(i)/(z1(i)*r(i,j)*cp(i,j))*ga(i)/step
*           dt(i+1,j)=dt(i+1,j)-rd(i)/(z1(i+1)*r(i+1,j)*cp(i+1,j))
```

```
*     & *ga(i)/step
*         dr(i,j)=dr(i,j)+rd(i)/z1(i)/step
*         dr(i+1,j)=dr(i+1,j)-rd(i)/z1(i+1)/step
*       w(i+1,j)=rd(i)*2./(pr(i+1,j)+pr(i,j))/step
       rdw(i+1,j)=rdw(i+1,j)+rd(i)/step
          w(i+1,j)=rdw(i+1,j)*2.0/(r(i+1,j)+r(i,j))
          end do
*
          do i=1,9
          p(i,j)=287.*rs(i,j)*r(i,j)*t(i,j)
          pb(i+1,j)=pb(i,j)+r(i,j)*9.8*z1(i)*rs(i,j)
*        pb(i+1,j)=p(i,j)+r(i,j)*9.8*zm(i)*rs(i,j)
          end do

*
86        continue
*
           do i=1,9
        pb(i,nb+1)=pb(i,nb)
         p(i,nb+1)=p(i,nb)
            do j=1,nb
          pr(i,j)=(r(i,j)-pr(i,j))/step
          dpr(i,j)=(p(i,j)-dpr(i,j))/step
          dw(i,j)=(w(i,j)-dw(i,j))
          end do
          end do
        do j=1,nb+1
         p(10,j)=pb(10,j)
        end do
         pb(10,nb+1)=pb(10,nb)
       p(10,nb+1)=p(10,nb)
*
          return
          end
*************************************************************************
* n eqns n unknowns matrix solution
          subroutine sle(a,b,n,n1,zero)
          real a(n1,n1),b(n1)
          do 100 i=1,n
          div=a(i,i)
          if(abs(div)-zero) 99,99,1
1         do 101 j=1,n
          a(i,j)=a(i,j)/div
101       continue
          b(i)=b(i)/div
          do 102 j=1,n
           if (i-j) 2,102,2
2         ratio=a(j,i)
```

```
          do 103 k=1,n
            a(j,k)=a(j,k)-ratio*a(i,k)
103       continue
          b(j)=b(j)-ratio*b(i)
102       continue
100       continue
          return
*99       print*,'matrix error',' ',i
99        end
*************************************************************************
* input program not presently used
      subroutine inmain(step,day,dwat,tme,pout,
     & c20,c40,n20,trs)
          real step,low,day,c20,c40,n20,tme,dwat,trs
          step=3.0
          low=.000001
          day=120.0
          c20=320.0
          c40=1.6
          n20=.3
          trs=.07
          tme=0.0
          dwat=2000
          pout=17
      return
      end

*********************************************************
* input called by rad.f
            subroutine inp(aoz,boz,coz,s0,theta,albedo,lap,cp)
      real aoz,boz,coz,s0,theta,albedo,cp(10),lap
          aoz=.318
          boz=23.0
          coz=5.0
          s0=680.0
          theta=60.0
          albedo=0.0
          lap=35.0
          cp(10)=50.0
          return
          end
*************************************************************************
* cloud calculations
* ire1... are infrared emissivities
* ac1... are cloud fractions 1 top 3 lowest
* dep1 cloud optical depth
* hcld, mcld, lcld are high mid and low cloud amount for each season
          subroutine cloud(ac1,kap1,dep1,ire1,ac2,kap2,dep2,ire2,
```

```
     & ac3,kap3,dep3,ire3,rg,depth,time,hcld,mcld,lcld,j,rh0,rhp)
           real ac1,ac2,ac3,dep1,dep2,dep3,ire1,ire2,ire3,rg,depth
           real time,hcld(20,4),mcld(20,4),lcld(20,4),tm(5),tc
         real rh0,rhp(4,20)
           integer kap1,kap2,kap3,j
         tc=time-21
         if(tc.lt.0.0) then
           tc=365.25+time-21
             end if
           tm(1)=0
           tm(2)=91
           tm(3)=183
           tm(4)=275
           tm(5)=365.30
           do k=1,3
           if (time.ge.tm(k).and.time.lt.tm(k+1)) then
           ac1=(hcld(j,k)+(hcld(j,k+1)-hcld(j,k))*(time-tm(k))
     &     /(tm(k+1)-tm(k)))/100
           ac2=(mcld(j,k)+(mcld(j,k+1)-mcld(j,k))*(time-tm(k))
     &     /(tm(k+1)-tm(k)))/100
           ac3=(lcld(j,k)+(lcld(j,k+1)-lcld(j,k))*(time-tm(k))
     &     /(tm(k+1)-tm(k)))/100
       rh0=rhp(k,j)+(rhp(k+1,j)-rhp(k,j))*(time-tm(k))
     &  /(tm(k+1)-tm(k))
           end if
          end do
          k=4
           if (time.ge.tm(k).and.time.lt.tm(k+1)) then
           ac1=(hcld(j,k)+(hcld(j,1)-hcld(j,k))*(time-tm(k))
     &     /(tm(k+1)-tm(k)))/100
           ac2=(mcld(j,k)+(mcld(j,1)-mcld(j,k))*(time-tm(k))
     &     /(tm(k+1)-tm(k)))/100
           ac3=(lcld(j,k)+(lcld(j,1)-lcld(j,k))*(time-tm(k))
     &     /(tm(k+1)-tm(k)))/100
       rh0=rhp(k,j)+(rhp(1,j)-rhp(k,j))*(time-tm(k))
     &  /(tm(k+1)-tm(k))

           end if
         if(abs(j-9.5).lt.3) then
           kap1=3
         ire1=1.0
         dep1=0.6-abs(0.0*(9.5-j)/9.)
* was 1.0  0.0
         else
          kap1=4
          ire1=0.5
           dep1=0.6-abs(0.0*(9.5-j)/9.0)
* was 1.0 0.0
```

```
      end if
        kap2=6
        dep2=1.2-abs(0.6*(9.5-j)/9)
* was  1.5  .5
        ire2=1.0
        kap3=8
        dep3=4.0-abs(2.0*(9.5-j)/9)
* was 5.5  2.0
        ire3=1.0
*       rg=.11
        depth=8.0
        return
        end
*************************************************************************
* simpler cloud input not use at present
        subroutine cl1(ac1,kap1,dep1,ire1,ac2,kap2,dep2,ire2,
   & ac3,kap3,dep3,ire3,rg,depth)
        real ac1,ac2,ac3,dep1,dep2,dep3,ire1,ire2,ire3,rg,depth
        integer kap1,kap2,kap3
        ac1=0.1
        kap1=3
        dep1=3.0
        ire1=0.5
        ac2=.15
        kap2=5
        dep2=8.0
        ire2=1.0
        ac3=.35
        kap3=7
        dep3=8.0
        ire3=1.0
        rg=.11
        depth=8.0
        return
        end
*********************************************
* seasonal average subroutine called from main portion of program
      subroutine xavge(xa,x,time,stav,nb,step,za2d,pha,text,nf)
       real xa(10,20),x(10,20),time,stav
       real tm(10),za2d(10,20),step,pha(20)
       character*10 text
        integer nb
        tm(1)=0
        tm(2)=91
        tm(3)=183
        tm(4)=275
        tm(5)=365.25
        do k=1,5
```

```
       if(time.ge.tm(k).and.time.lt.tm(k)+step) then
        if(stav.gt.2.0) then
          write(nf,*) text,'season',k-1
          do i=1,10
           do j=1,nb
     write(nf,*) 57.3*pha(j),-za2d(i,j)/100.,xa(i,j)/stav
            xa(i,j)=0.0
           end do
          end do
           stav=0.0
         end if
        end if
*
          if (time.ge.tm(k).and.time.lt.tm(k+1)) then
          do i=1,10
           do j=1,nb+1
            xa(i,j)=xa(i,j)+x(i,j)
           end do
          end do
           stav=stav+1.0
          end if
        end do
        return
         end
**********************************************************************
* convective lapse rate adjustment gam=gamma is the critical lapse rate for the
* convective adjustment of Manabe and Wetherald 1967
     subroutine lapadj (at,gam,za,cp,ch,nt,ce)
     real at(10),gam(10),za(10),cp(10),ch(10),ce(10)
     real s2,d1,d2,d3,db,dt
     integer nt
     nt=6
     do 7998 i=1,10
      ch(i)=0.0
7998   continue
7999   s2=0.0
     za(10)=0.0
     do 8002 i=8,2,-1
     d1=at(i+1)-at(i)
     d2=gam(i)*(za(i)-za(i+1))
      if (d1.gt.d2) then
          d3=1.0*(d1-d2)
           db=cp(i)*d3/(cp(i)+cp(i+1))
           dt=cp(i+1)*d3/(cp(i)+cp(i+1))
           at(i)=at(i)+dt
           at(i+1)=at(i+1)-db
           ch(i)=ch(i)+dt
           ch(i+1)=ch(i+1)-db
```

```
          s2=s2+d3
          nt=i
       end if
8002  continue
      if (s2.gt..005)then
        goto 7999
      endif
     do 8005 i=2,8
         if (at(10).lt.(at(9)+.04))then
         ce(i)=ce(i)*.95
     end if
         ce(i)=ce(i)+ch(i)/8.0
     if(ch(i).lt..0001) then
     ce(i)=ce(i)*.99
     end if
8005    continue
      return
      end
*
*************************************************************************
* set critical lapse rate to a fixed value such as 6.5 K/km
     subroutine lap1 (gam,lap)
     real gam(10)
     real lap
     do 8600 i=1,9
      gam(i)=lap
8600   continue
      return
      end
*************************************************************************
     subroutine lap2 (gam,at,pa,gc)
*     moist adiabatic lapse rate from Stone & Carlson 1979
*   gc is the critical baroclinic lapse rate vi stone & carlson
* 1979
     real gam(10),at(10),pa(10),gc
     real l,r,de,e
     r=.287
     do 8700 i=1,9
     l=2510.-2.38*(at(i)-273)
      e=esat(at(i))
     de=.622*l*e/(r*at(i)**2)
     gam(i)=9.8*(1+.622*l*e/(pa(i)*r*at(i)))/
    1(1+(.622*l*de)/(1.005*pa(i)))
     if(gam(i).gt.6.0) then
      gam(i)=6.0
      end if
8700  continue
*       gam(9)=1.0
```

```
      return
      end
*************************************************************************
* Stone and Carlson 1979
      subroutine baroclinic(gc,t,pha,dp,nb)
       real gc(20),t(10,20),pha(20),dp,h2
        integer nb
       do j=3,nb-2
        h2=287*t(6,j)/9.8
       gc(j)=9.8+1000.*
    & tan(pha(j))/h2*(t(6,j-1)-t(6,j+1))/(2.*dp)
        end do
         gc(2)=gc(3)
         gc(1)=gc(2)
       gc(nb-1)=gc(nb-2)
       gc(nb)=gc(nb-1)
        return
         end
*********************************************************
* chooses between critical lapse rate from baroclinic theory or moist adiabatic lapse rate
      subroutine lapstone(g,gc,p)
      real g(10),gc,p(10)
18     s1=0.
       s2=0.
      do i=1,8
      s1=s1+g(i)*(p(i+1)-p(i))
        s2=s2+p(i+1)-p(i)
       end do
        avg=s1/s2
       if (avg.gt.gc) then
         do i=9,1,-1
           g(i)=gc
          end do
        end if
        return
       end
*********************************
* calculates the average lapse rate
       function avlap(at,za,p)
        real avlap,at(10),za(10),p(10)
         s1=0.
         s2=0.
          do i=3,8
           s1=s1+(at(i+1)-at(i))*(p(i+1)-p(i))/
   & (za(i)-za(i+1))
            s2=s2+(p(i+1)-p(i))
          end do
        avlap=s1/s2
```

```
      return
      end
*************************************************************************
* 2-D ocean model from Watts and Morentine (1990)
      subroutine ocean(t,pha,phb,st,vk,hk,ho,w,d,rl,nb)
      real t(7,20),dt(7,20),dz(7,20),tp(7,20)
       real pha(20),phb(20)
      real a,b,q,x,al,st,vk,hk,ho,w,d,rc,rl,y,yp,ym
      real xm,xp
       integer nb
*      open(unit=10,file='inoc1')
*      open(unit=20,file='out')
*        open(unit=30,file='out.ext')
*      read (10,*) nb,st,w,d,rl,hk,ho,vk
*      read(10,*) ((t(i,j),j=1,nb),i=1,6)
      time=0.
      rc=4.186e6/3.15e7
          do j=1,nb
           do i=1,6
             if(i.eq.1)then
              dz(i,j)=d
             else
              dz(i,j)=1000
             end if
*           t(i,j)=t(i,1)
       if(i.eq.6)then
              dz(i,j)=0.
              t(i,j)=t(1,3)
            end if
          end do
          end do

         a=-283.4
         b=1.8
30       do i=1,1
          do j=1,nb
           x=sin(pha(j))
           y=cos(pha(j))
           yp=cos(phb(j+1))
           ym=cos(phb(j))
           xp=sin(phb(j+1))
           xm=sin(phb(j))
           if(x.gt..94)then
           al=.4
           else
           al=.75-.18*x**2
           end if
           q=al*345*(1.25-.75*x**2)
```

```
*           dt(i,j)=(q-(a+b*t(i,j)))/(rc*d)-vk*(t(i,j)-t(i+1,j))*2.
            dt(i,j)=-vk*(t(i,j)-t(i+1,j))*2.
     &     /((dz(i,j)+dz(i+1,j))*d)
          sum=sum+(q-(a+b*t(i,j)))*y
            if(j.eq.1)then
            dt(i,j)=dt(i,j)-hk*yp*(t(i,j)-t(i,j+1))/(y*.1744**2)
            end if

            if(j.eq.9)then
            dt(i,j)=dt(i,j)-hk*ym*(t(i,j)-t(i,j-1))/(y*.1744**2)
*     & -xm*w*(t(i,j))/(d*.1744*y)
*     &   +w*t(i+1,j)/d
     & -xm*w*(t(i,j)+t(i,j-1))/(2.*d*.1744*y)
     &  +w*(t(i+1,j)*30.+500.*t(i,j))/530./d
*     & -xm*w*(t(i,j))/(d*.1744*y)
            end if

            if(j.gt.1.and.j.lt.9)then
            dt(i,j)=dt(i,j)-hk*ym*(t(i,j)-t(i,j-1))/(y*.1744**2)
     & -hk*yp*(t(i,j)-t(i,j+1))/(y*.1744**2)

            if(j.gt.3.and.j.lt.9)then
*           dt(i,j)=dt(i,j)+xp*w*t(i,j+1)/(d*.1744*y)
*     & -xm*w*t(i,j)/(d*.1744*y)
*     &  +w*t(i+1,j)/(d)
          dt(i,j)=dt(i,j)+xp*w*(t(i,j+1)+t(i,j))/(2.*d*.1744*y)
     & -xm*w*(t(i,j)+t(i,j-1))/(2.*d*.1744*y)
     & +w*(30.*t(i+1,j)+500.*t(i,j))/(530.*d)
*     & -xm*w*t(i,j)/(d*.1744*y)
            end if

            if(j.eq.3)then
*           dt(i,j)=dt(i,j)+xp*w*(t(i,j+1)-t(i,j))/(d*.1744*y)
            dt(i,j)=dt(i,j)+xp*w*(t(i,j+1)-t(i,j))/(2.*d*.1744*y)

           wst=xp*w/(.1744*y)
             end if
             end if
*.1744=dphi
           end do
           end do

          do i=2,5
           do j=1,nb

           x=sin(3.14*(95-10*j)/180)
           y=cos(3.14*(95-10*j)/180)
           yp=cos(3.14*(90-10*j)/180)
```

```
          ym=cos(3.14*(90-10*(j-1))/180)


          if (i.lt.5.and.i.gt.2)then
            dt(i,j)=
     &      +vk*(t(i-1,j)-t(i,j))*2./((dz(i-1,j)+dz(i,j))*dz(i,j))
     &      -vk*(t(i,j)-t(i+1,j))*2./((dz(i+1,j)+dz(i,j))*dz(i,j))
          if(j.gt.3) then
            dt(i,j)=dt(i,j)+w*(t(i+1,j)-t(i-1,j))/(2.*dz(i,j))
          end if
           end if

          if (i.eq.5)then
            dt(i,j)=
     &      +vk*(t(i-1,j)-t(i,j))*2./((dz(i-1,j)+dz(i,j))*dz(i,j))
*    &     -vk*(t(i,j)-t(i+1,j))*2./((dz(i+1,j)+dz(i,j))*dz(i,j))
          if(j.gt.3) then
            dt(i,j)=dt(i,j)+w*(t(i+1,j)-
     &  (t(i-1,j)+t(i,j))/2.)/(dz(i,j))
          end if
           end if

          if (i.eq.2)then
            dt(i,j)=
     &      +vk*(t(i-1,j)-t(i,j))*2./((dz(i-1,j)+dz(i,j))*dz(i,j))
     &      -vk*(t(i,j)-t(i+1,j))*2./((dz(i+1,j)+dz(i,j))*dz(i,j))
          if(j.gt.3) then
            dt(i,j)=dt(i,j)+w*(t(i+1,j)+t(i,j))/(2.*dz(i,j))
     &     -w*(t(i,j)*30.+500.*t(i-1,j))/(530.*dz(i,j))
          end if
           end if


            if(j.eq.1)then
            dt(i,j)=dt(i,j)-ho*yp*(t(i,j)-t(i,j+1))/(y*.1744**2)
            end if

            if(j.eq.nb)then
            dt(i,j)=dt(i,j)-ho*ym*(t(i,j)-t(i,j-1))/(y*.1744**2)
            end if

            if(j.gt.1.and.j.lt.9)then
            dt(i,j)=dt(i,j)-ho*ym*(t(i,j)-t(i,j-1))/(y*.1744**2)
     & -ho*yp*(t(i,j)-t(i,j+1))/(y*.1744**2)
             end if
*.1744=dphi
          end do
          end do
```

```
          do i=1,5
           do j=1,nb
           t(i,j)=t(i,j)+dt(i,j)*st
*           sum=sum+t(i,j)*cos(3.14*(10*j-5)/180)*dz(i,j)/10000.
           end do
           end do

       do j=1,nb
          t(6,j)=t(1,3)
        end do

          count=count+st
          if(time.eq.0.or.count.ge.50)then
          count=0.0
*         write(20,*) 'time=',time
*         write(20,1000) ((t(i,j),j=1,nb),i=1,6)
        trade=t(1,1)
        t(1,1)=time
*         write(30,1001)(t(1,j),j=1,nb,2)
*         write(*,1001)(t(1,j),j=1,nb,2)
*         write(*,1001) t(1,1),t(1,2),t(1,3),t(1,5),t(1,7),t(1,9)
*         write(30,1001) t(1,1),t(1,2),t(1,3),t(1,5),t(1,7),t(1,9)
         t(1,1)=trade
*         print*,sum
          end if
          time=time+st
        if(time.gt.2000-10.and.time.lt.2000)then
       do j=1,nb
        tp(1,j)=t(1,j)
       end do
       end if
       if (time.gt.2000.and.time.lt.2200+st/2.) then
*       w=0.0
       end if
        sum=0.
          if (time.lt.rl+.01) goto 30
1000     format(9(f6.1,1x))
1001     format(9(f6.2,1x))
*         write(*,1000) ((t(i,j),j=1,nb),i=1,6)
*         write(30,1001) ((t(i,j),j=1,nb),i=1,6)
*        write(*,*)
*        write(*,1000) (t(1,j),j=1,nb)
*        write(*,1000) (tp(1,j),j=1,nb)
*        write (*,1000) ((t(1,j)-tp(1,j)),j=1,nb)
          end
*************************************************************************
```

```
*****************************************************************************
*
      subroutine solo3 (s0,theta,uo,p,ho,fso,abo,ac,rg)
*     Calculate ozone solar heating according to Lacis and Hansen 1974
*     Clear Skies
      real a(10),x(10),xu(10),au(10),ab(10)
      real uo(10), ho(10), fso(10),p(10)
      real mu0, mbar, m, rg, ra2, ra1, theta, ra
      real abo,rrm,rrs
      s=s0*(1-ac)
      mu0=cos(theta)
      mbar=1.9
      ra2=.144
      ra1=.219/(1+.816*mu0)
      ra=ra1+(1-ra1)*(1-ra2)*rg/(1-ra2*rg)
      m=35./sqrt(1224*mu0*mu0+1)
          do 5000 i=1,10
           x(i)=m*uo(i)
           a(i)=ao3(x(i))
5000    continue
          do 5010 i=1,10
           xu(i)=x(10)+mbar*(x(10)-x(i))/m
           au(i)=ao3(xu(i))
5010    continue
           fso(1)=s*mu0
        abo=0.0
           do 5020 i=1,9
           ab(i)=s*mu0*((a(i+1)-a(i))+ra*(au(i)-au(i+1)))
           ho(i)=.0083224*ab(i)/(p(i+1)-p(i))
           fso(i+1)=(s-s*a(i+1))*mu0
        abo=abo+ab(i)
5020    continue
       rrm=.28/(1+6.43*mu0)
       rrs=.0685
      fso(10)=s*(1-rg)*mu0*(.647-rrm-a(10))/(1-rrs*rg)
      abo=abo+fso(10)
      return
      end
*****************************************************************************
*
      subroutine sco3 (s0,theta,uo,p,ho,fso,abo,ac,rg,kap,depth)
*     Calculate ozone solar heating according to lacis and hansen 1974
*     Cloudy Skies
      real a(10),x(10),xu(10),au(10),ab(10)
      real uo(10), ho(10), fso(10),p(10)
      real mu0, mbar, m, rg, ra2, ra1, theta, ra
      real ac,depth,abo,x1,rrm,rrs
      integer kap
```

```
      s=s0*ac
      mu0=cos(theta)
      x1=.85
      mbar=1.9
      ra2=(1-x1)*depth*sqrt(3.0)/(2+(1-x1)*depth*sqrt(3.0))
      ra1=ra2
      ra=ra1+(1-ra1)*(1-ra2)*rg/(1-ra2*rg)
      m=35./sqrt(1224*mu0*mu0+1)
           do 5000 i=1,10
            x(i)=m*uo(i)
            a(i)=ao3(x(i))
5000    continue
           do 5010 i=1,10
            xu(i)=x(kap+1)+mbar*(x(kap+1)-x(i))/m
            au(i)=ao3(xu(i))
5010    continue
            fso(1)=s*mu0
         abo=0.0
            do 5020 i=1,9
            ab(i)=s*mu0*((a(i+1)-a(i))+ra*(au(i)-au(i+1)))
            ho(i)=.0083224*ab(i)/(p(i+1)-p(i))
            fso(i+1)=(s-s*a(i+1))*mu0
         abo=abo+ab(i)
5020    continue
        rrm=.28/(1+6.43*mu0)
        rrs=.0685
      fso(10)=s*(1-rg)*(1-ra1)*mu0*(.647-a(10))/(1-ra2*rg)
      abo=abo+fso(10)
      return
      end
*****************************************************************
      function ao3(x)
      a1=.02118*x/(1+.042*x+.000323*x*x)
      a2=(1.082*x/(1+138.6*x)**.805)+.0658*x/(1+(103.6*x)**3)
      ao3=a1+a2
      return
      end
*****************************************************************
*****************************************************************
*Hub of radiation calculations See Mackay (1990)
      subroutine radiation(at,at1,t,p,pa,z,za,ch,ce,sun,sout,f,
     & c,abtot,ac,s1,nt,step,c20,c40,n20,jb,sth,cp,rh0,
     & hcld,mcld,lcld,ts,ot,of,uo,gc,u,al,xi,ti,albl,rg,suavg
     & ,time,vi,rhp,rain,hs)
       real t(10),z(10),p(10),uo(10),sth(20),suavg
      real at(10),pa(10),za(10),uoa(10),cp(10),c(10)
      real ta(10,10),ut(10,10),time,lh,sh
      real gam(10),ch(10),ce(10),xmu,zave,gc
```

```
      real tr2(10,10),tco2n2(10,10),tnc4(10,10),tr5(10,10)
      real tr3(10,10),tr4(10,10),tr6(10,10)
      real dg(10),sho(10),shw(10),shoc(10),shco2(10),sho2(10)
      real g(10),gw(10),go(10),goc(10),em(10,10),c1(10,10)
      real xk(10),pk(10),at1(10)
      real dc(10),dw(10),do(10),d4(10),d5(10),ds(10)
      real df(10),f(10),fw(10),fs(10),ts,u
      real hcld(20,4),mcld(20,4),lcld(20,4),rhp(4,20)
*************************************************************************
      real s0,theta,step,albedo,s1,sout,lap,ot,of
      real sun,abw,abo,aboc,abo2,abco2,abtot
      real c20,c40,n20,rh0,rain,oalb,ialb,lalb
      real ire1,ire2,ire3
      real depth,rg,ac,ac1,ac2,ac3,dep1,dep2,dep3
      real al,xi,ti,albl,vi,hs
      integer kap1,kap2,kap3,nt,kap,jb
*************************************************************************
      data xk(1),xk(2),xk(3),xk(4),xk(5),xk(6),xk(7),xk(8)
     > /.00004, .002, .035, .377, 1.95, 9.40, 44.6, 190./
      data pk(1),pk(2),pk(3),pk(4),pk(5),pk(6),pk(7),pk(8)
     > /.647, .0698, .1443, .0584, .0335, .0225, .0158, .0087/
*************************************************************************
*       time=0.0
          call inp(aoz,boz,coz,s0,theta,albedo,lap,cp)
*         call cloud(ac1,kap1,dep1,ire1,ac2,kap2,dep2,ire2,
*      & ac3,kap3,dep3,ire3,rg,depth,21.,hcld,mcld,lcld,jb,rh0,rhp)
          call cloud(ac1,kap1,dep1,ire1,ac2,kap2,dep2,ire2,
     & ac3,kap3,dep3,ire3,rg,depth,time,hcld,mcld,lcld,jb,rh0,rhp)
*    use heat capacity units of J-Day/(m^2 K)
           kap=kap1
           ac=ac1
           cp(10)=cp(10)*4.186e6/86400
           nt=6.0
           sun=s0
*          call sunstuff(sth,tm,xmu,zave,21.,jb,s0)

           call sunstuff(sth,tm,xmu,zave,time,jb,s0)
*           print*,jb,xmu,zave,s0,tm
*         call sunavg(jb,zave,tm,s0,suavg)
         s0=s0*tm/24.00

           sun=s0*cos(zave)
*         print*,jb,sun,s0
*         write(20,*) jb,sun,zave,xmu,tm
         g(1)=sun
*          print*,j,sun,tm,zave
            theta=zave
           call suralbedo((sth(jb)+sth(jb+1))/2.,rg,ti,ts,al
```

```
     & ,albl,zave,u,of,lalb,ialb,oalb)
*          print*,jb,rg
*          s0=s0*(1.0-albedo)
*           theta=theta*3.1415926/180.0
*          sun=sun*cos(theta)
*           g(1)=s0*cos(theta)
*
*          set initial pressures,temperature,height, and reference lapse
*          rate of each layer.
           do 9 i=1,10
*        uo(i)=(aoz+aoz*exp(-boz/coz))/(1+exp((z(i)-boz)/coz))
*        uoa(i)=(aoz+aoz*exp(-boz/coz))/(1+exp((za(i)-boz)/coz))
         if(i.lt.10) then
         uoa(i)=(uo(i)*(za(i)-z(i+1))+uo(i+1)*(z(i)-za(i)))
     & /(z(i)-z(i+1))
           else
         uoa(i)=uo(i)
         end if
*           uoa(i)=(aoz+aoz*exp(-boz/coz))/(1+exp((za(i)-boz)/coz))
9      continue
       if (lap.gt.0)then
       call lap1(gam,lap)
       end if
*
*      main iterative part of the program
*      calculate pressure weighted average temperature btwn layer i and j
10         call tempave (at,p,ta)
*      calculate the n2o (1285 cm-1) ch4 overlap
           call n2och4 (tnc4,p,pa,at,c40)
*      moist adiabatic lapse rate calculation (used instead of lap1 above)
           if(lap.gt.20)then
           call lap2 (gam,at,pa,gc)
           end if
         if(gc.lt.4.0) then
          gc=4.0
         end if
         call lapstone(gam,gc,pa)
*          if(jb.eq.3) then
*          do i=1,10
*         print*,i,jb,gam(i),gc
*          end do
*           end if
*          recalc temps at top and bottom of each layer
           call tempset (t,at,p,kap)
*
*      Calculate h2o IR absortion and transmissions
       call water (pa,at,p,t,ta,dw,fw,tr2,ac,kap,tr3,
     x tr4,tr5,tr6,nt,at1,em,rh0)
```

```
*
*       update co2 concentration and calculate IR absorption transmission
        call co2 (t,p,ta,tr2,tr3,c20,tco2n2,em,c1)

850      format(10(f7.3,1x))
*
*       Calculate cloudy & clear sky solar absorption due to water vapor

     call solclod (p,pa,at,xk,pk,theta,s0,rg,kap1,ac1,kap2,ac2,kap3
    x ,ac3,dep1,dep2,dep3,shw,abw,gw,at1,rh0)
*
*       Methane IR absorption
        call ch4cool (ta,p,tr4,c40,em)
*
*       N2O (1285 cm-1) IR absorption
        call n2o5cool (ta,p,pa,t,tr5,tnc4,n20,ut,em)
*
*       N2O (590 cm-1) IR absorption
        call n2o6cool (pa,ta,t,tr6,tco2n2,n20,ut,em)
*
        do 15 i=1,10
         ds(i)=0.0
         fs(i)=0.0
15       continue
        call small (ta,p,t,275.,850.0,1965.0,em)
*
        call small (ta,p,t,275.,1075.0,736.0,em)
*
        call small (ta,p,t,468.,912.0,1568.0,em)
*
        call small (ta,p,t,468.,1090.0,1239.0,em)
*
        call small (ta,p,t,468.,1150.0,836.0,em)
*
*       call small (ta,p,t,.00028,1150.0,767.0,em)
*
*       call small (ta,p,t,.00028,1150.0,767.0,em)
*
*       Ozone planck function and IR absorption
*
         call o3cool (at,t,pa,uo,tr3,em,ta)
*       ozone clear and cloudy skies solar absorption
        ac=1-(1-ac1)*(1-ac2)*(1-ac3)
        call solo3 (s0,theta,uo,p,sho,go,abo,ac,rg)
*
    depth=(ac1*dep1+ac2*dep2+ac3*dep3)/(1-ac)
    call sco3 (s0,theta,uo,p,shoc,goc,aboc,ac,rg,kap1,depth)
*
```

```
*    solar absorption for CO2
     call sco2 (p,pa,shco2,abco2,c20,s0,theta)
*
*    solar absoprption for molecular oxygen
     call soxy (p,sho2,abo2,s0,theta)
*
*    combine IR and Solar heating rates and fluxes
          call combine(dw,dc,do,d4,d5,ds,shw,shoc
    x     ,sho2,shco2,sho,dg,df)
           call flux (em,p,at,kap1,ac1,kap2,ac2,kap3,ac3,c,df,f,
    > ire1,ire2,ire3)
          g(10)=goc(10)+gw(10)+go(10)-
    x(1-rg)*(abo2+abco2)
*         write(*,*) g(10),g(10)/(1-rg),f(10)
          abtot=abw+abo+aboc+rg*(abo2+abco2)
*        if(jb.eq.18) then
*         do i=1,10
*       print*,dg(i),shw(i),sho(i),sho2(i),shco2(i)
*          end do
*         end if
       sout=albedo*sun+(g(1)-abtot)
*      change the average temperature of each layer

      call tempchng(pa,p,cp,at,dg,df,g,step,s1,
     & f,ch,ce,kap,ft,c,at1,rh0,ts,ot,of,u,al,xi,ti,vi
     &,lh,sh,jb,rain,hs,oalb,ialb,lalb,rg,time)
*       print*,jb,at(10),ot,ti,ts
*        if(abs(jb-9.5).lt.3.) then
*       call cumulus(at,t,za,z,pa,ch,jb,step,ce,c,cp,lh,sh)
*         else
          if(lap.gt.0)then
          call lapadj (at,gam,za,cp,ch,nt,ce)
*         end if
       end if
*     recalculate the height of each layer
*          call height(z,za,p,pa,at)
*     perform the lapse rate adjustment
1003      format (10(f7.3,2x))
          return
          end
*************************************************************************
*************************************************************
*  calculates the specific humidity
      function rwat(p,e,j,kap,rh0)
       real rwat,h,e,p,rh0
       integer j,kap
       if(j.eq.kap) then
        h=rh0*(p-.02)/.98
```

```
        else
        h=rh0*(p-.02)/.98
       end if
       rwat=.622*h*e/(p-h*e)
        if (rwat.lt.3.0e-6) then
         rwat=3.0e-6
         end if
         return
        end
*****************************************************************
* stauration vapor pressure in atmospheres
     function esat(t)
     real esat,l,r,t
     l=2510.-2.38*(t-273)
     r=.287
     esat=(6.11/1012.34)*exp((.622*l/r)*(t-273)/(t*273))
     return
     end
*****************************************************************
* advection for trace gases which is not in use at present
      subroutine gasstr(r,c,v,w,dc,phb,pha,z,nb,emp)
       real r(10,20),c(10,20),v(10,20),w(10,20),dc(10,20)
       real g(10,20),za(10,20),z(10,20)
       real phb(20),pha(20),a,step,emp(20)
       integer nb
        a=6.37e6
       do i=2,9
        do j=1,nb+1
         if(i.eq.2) then
          emp(j)=0.0
          end if
        g(i,j)=.25*v(i,j)*(c(i,j)+c(i,j+1))*(r(i,j)+r(i,j+1))
    & *cos(phb(j))
        end do
          end do
*  dc is in mm of water per day
* kg Wat/m^3 air*(1m^3 h20/1000 kg h20)=m h20/m air*dz(air)
          do i=2,9
          do j=1,nb
           dc(i,j)=((g(i,j+1)-g(i,j))/(a*(phb(j+1)-phb(j)))
    &    +(w(i+1,j)*.25*(c(i+1,j)+c(i,j))*(r(i+1,j)+r(i,j))-
    &     w(i,j)*.25*(c(i-1,j)+c(i,j))*(r(i-1,j)+r(i,j)))/
    &    (z(i,j)-z(i+1,j)))*(z(i,j)-z(i+1,j))*86400
       emp(j)=emp(j)+dc(i,j)
        end do
          end do
*         do i=1,9
*         write(20,6001) (dc(i,j),j=1,nb)
```

```
*           end do
6001     format(19(f7.4,1x))
       return
      end
**************************************************
* slightly different version of Shiparo Filter
         subroutine shfil2(x,nb)
         real x(10,30),tp(10,40)
         integer nb
         do i= 1,10
          tp(i,1)=2*x(i,1)-x(i,5)
           tp(i,2)=2*x(i,1)-x(i,4)
            tp(i,3)=2*x(i,1)-x(i,3)
             tp(i,4)=2*x(i,1)-x(i,2)
              tp(i,nb+8)=2*x(i,nb)-x(i,nb-4)
               tp(i,nb+7)=2*x(i,nb)-x(i,nb-3)
                tp(i,nb+6)=2*x(i,nb)-x(i,nb-2)
                    tp(i,nb+5)=2*x(i,nb)-x(i,nb-1)
               do j=1,nb
               tp(i,j+4)=x(i,j)
               end do
          end do
*
         do i=1,10
         do j=1,nb
         x(i,j)=(186.*tp(i,j+4)+56.*(tp(i,j+3)+tp(i,j+5))-
    & 28.*(tp(i,j+2)+tp(i,j+6))+8.*(tp(i,j+1)+tp(i,j+7))
    & -(tp(i,j)+tp(i,j+8)))/256.
         end do
         end do
         return
         end
**********************************************************************
*    Calculates solar heating due to CO2 following the paramaterization
*    given by Sasamori 1972
     subroutine sco2(p,pa,shco2,abco2,c20,s0,theta)
     real p(10),pa(10),shco2(10),a2(10)
     real u2,abco2,c20,theta,s0,mu0
         mu0=cos(theta)
         abco2=0.0
         u2=0.0
         a2(1)=0.0
     do 9600 i=1,9
         u2=u2+c20*.8*(p(i+1)-p(i))*pa(i)
         a2(i+1)=(2.35e-3)*((u2+.0129)**.26)-7.5e-4
9600  continue
     do 9610 i=1,9
*         if (i.gt.kap-1) then
```

```
*         shco2(i)=(1-ac)*s0*mu0*(a2(i+1)-a2(i))
*         else
          shco2(i)=s0*mu0*(a2(i+1)-a2(i))
*         end if
          abco2=abco2+shco2(i)
          shco2(i)=shco2(i)*.0083224/(p(i+1)-p(i))
9610  continue
      return
      end
*************************************************************************
**
*     Calculates solar absorbtion due to molecular Oxygen following
*     Sasamori 1972
      Subroutine soxy(p,sho2,abo2,s0,theta)
      real p(10),sho2(10),a4(10)
      real abo2,theta,mu0,s0
      mu0=cos(theta)
      abo2=0.0
      a4(1)=0.0
      do 9700 i=1,9
          a4(i+1)=7.5e-3*(p(i+1)/(mu0+0.001))**.875
9700   continue
      do 9710 i=1,9
*         if (i.gt.kap-1) then
*         sho2(i)=(1-ac)*s0*mu0*(a4(i+1)-a4(i))
*         else
      sho2(i)=s0*mu0*(a4(i+1)-a4(i))
*         end if
      abo2=abo2+sho2(i)
      sho2(i)=sho2(i)*.0083224/(p(i+1)-p(i))
9710   continue
      return
      end
*************************************************************************
* calculates the solar intensity at the top of the atmosphere/ and mean solar
* zenith
       subroutine sunstuff(th,tm,xmu,zave,time,node,s0)
       real th(32),tm,xmu,zave,ts,tm1
       real ec,pi,del,time,cs,s0,ss,csz,za
       real t1,t2,t3,t4,r1,r2,phs1,sq,ht
       integer node,nd
          nd=node
*         time=0.
       ec=.0167504
       pi=acos(-1.00)
        ts=time-21.
         if(ts.lt.0) then
             ts=365.25+ts
```

```
          end if
        psi=2.00*pi*(ts)/365.25+1.308
         if(psi.gt.2*pi) then
          psi=psi-2*pi
         end if
        r1=atan((ec*sin(psi))/(1.-ec*cos(psi)))
        r2=r1+asin(-ec**3*sin(psi+r1)*(1+ec*cos(psi)))
        psi=psi+r2
        cs=(-ec+cos(psi))/(1.0-ec*cos(psi))
*        if(tan(psi).lt.0.0.and.cs.gt.0.) then
        if(psi.ge.0.0.and.psi.le.pi) then
         ss=sqrt(1-cs**2)
         else
          ss=-sqrt(1-cs**2)
          end if
*        if(tan(psi).gt.0.0.and.cs.lt.0.) then
*         ss=-sqrt(1-cs**2)
*         else
*          ss=sqrt(1-cs**2)
*          end if
       del=(23.5*pi/180.00)*(cs*cos(2.911)+ss*sin(2.911))
*57.******  solar constant vs Earth's orbital radius (
*  H. Goldstein  Classical Mechanics Addison Wesley 1982 2nd
*  Chapter 3
       s0=1360.0/(1.00-ec*cos(psi))**2.0
*        s0=1360.0
******** Calculate time of daylight tm and average zenith angle zave
       t1=asin(th(nd))
        t2=asin(th(nd+1))
       t3=asin((th(nd)+th(nd+1))/2.00)
        t4=tan(del)*tan(t3)
       if(t4.lt.-1.0) then
       tm=0.00
       zave=pi/2.00
       else
       if(t4.gt.1.00) then
       tm=24.00
       else
       tm=24.00-acos(t4)*24.00/pi
        endif
       ht=tm*pi/24
*  calculate the average daylight zenith angle
* This doesn't work
*         sum1=0
*         dx=ht/40
*         h1=ht/2-dx/2
*         do k=1,20
*         sum1=sum1+(sin(t3)*sin(del)+
```

```
*     & cos(t3)*cos(del)*cos(h1))/20.
*           h1=h1-dx
*          end do
*  This seems to work well for avg cos(zen) Liou p46
*          z1=sin(t3)*sin(del)+cos(t3)*cos(del)*sin(ht)/ht
* end calc of zaverage
*The following line is from Liou 1980
          sq=(s0/pi)*(sin(t3)*sin(del)*ht
     > +cos(t3)*cos(del)*sin(ht))
*         xmu=(1.00/(2.00*pi*(th(nd)-th(nd+1))))*(t1-t2+.5*
*     > (sin(2.00*(t1-del))-sin(2.00*(t2-del))))
*           print*,node,t4,del,pi*1.5,ss,cs,psi,tan(psi)
        zave=acos(24.0/tm*sq/s0)
*           tm1=(sq*24.0)/(s0*sum1)
*          za=acos(sum1)
*          print*,'zav,za',acos(z1),za,zave,tm
*           zave=acos(24.0/tm*xmu)
        endif
*       print*,tm,s0,del*180./pi,xmu
        return
        end
************************************************************************
* suface albedo
          subroutine suralbedo(sine,rg,ti,ts,al,albl,zave,u,of,
     & lalb,ialb,oalb)
          real rg,sine,ti,ts,al,albl,zave,x,albo,of,lalb
        real ialb,oalb
         x=1-cos(zave)
         if (ts.lt.273.) then
           lalb=0.6+(albl-0.6)*exp(-(273-ts)/12.)
            else
           lalb=albl
             end if
       oalb=0.021+0.0421*x*x+0.128*x**3-0.04*x**4
     & +3.12*x**5/(5.68+u)+0.074*x**6/(1+3*u)
     rl=of*((1-al*al)*oalb+al*al*.6)
     & +(1-of)*lalb
         ialb=0.6
          rg=rl
*       rg=albo*of+(1-of)*albl
*         rg=.55*sine**6+.08
*        rg=rl
*        print*,rg,rl,oalb,lalb,ialb
      return
      end
************************************
* solar energy for annual mean conditions
       subroutine sunavg(jb,zave,tm,s0,s1)
```

```
      real s1,s0,tm,zave,za
       integer jb
        tm=12.00
* tm1=8.48 gives correct s1 and za for annual mean
        tm1=8.48
       zave=acos(s1/680)
         za=acos(s1*24/(1360*tm1))
       print*,zave,za
       s0=1360.
       return
       end
*****************************************************************
*       Calculate methane transmission in N20 1200-1350 cm-1
*       region using Green's 1964 method
    function tn2c4(w)
    real dn(10),we(10)
    real w,tn2c4
    dn(1)=35.6
    dn(2)=12.0
    dn(3)=18.5
    dn(4)=13.1
    dn(5)=72.0
    we(1)=18.4
    we(2)=9.08
    we(3)=2.60
    we(4)=6.47
    we(5)=14.95
    tn2c4=0.0
    do 9400 i=1,5
    tn2c4=tn2c4+exp(-((w/we(i))**.46))*dn(i)/151.2
9400  continue
    return
    end
*****************************************************************
*    Calculate N2O CH4 overlap
    subroutine n2och4 (tnc,p,pa,at,c40)
    real tnc(10,10)
    real p(10),pa(10),at(10)
    real f,w,tn2c4,c40
*    f is ch4 mixing ratio 1.6 ppmv
    f=c40
    do 9360 i=1,9
         w=f*(1.29/1.6)*pa(i)*(p(i+1)-p(i))*sqrt(300/at(i))
         tnc(i,i)=tn2c4(w)
        do 9359 j=i+1,10
          do 9355 k=i,j-1
        w=w+f*(1.29/1.6)*pa(k)*(p(k+1)-p(k))*sqrt(300/at(k))
9355     continue
```

```
        tnc(i,j)=tn2c4(w)
9359    continue
9360   continue
        return
        end
*****************************************************************
*    Calculate CH4 cooling via Ramanathan 1980
     subroutine ch4cool (ta,p,tr4,c40,ew)
     real p(10),ew(10,10)
     real tr4(10,10),a4,ta(10,10),ut(10,10)
     real pe,bet,a0,u,c40,c,a,x,d,bl
     a=3.03
     x=0.104
     d=1.012
     c=1.66*185*c40*(1.28/1.6)
*     134 (Goody 1989) is the band strength S 1.28*(c40/1.6) *delta p
*     is the absorber amount in atm-cm c40 is the concemtration
*     of CH4 in ppmv
     call ucalc(p,ut,3,a,x,d,1.6)
          do 9005 j=1,9
           pe=(p(j)+p(j+1))/2.00
            bet=pe*.211*(300./ta(j,j+1))
*     bet0=.17 from Ramanathan 1980
          a0=68.2*(ta(j,j+1)/300)**.858
          bl=3.742e-16*(130600.**3)/((exp(1.438*1306./ta(j,j+1)))-1)
             u=(c/a0)*ut(j,j)
             a4=200.*a0*log(1+(u/(.106+sqrt(3.59+u*(1+1/bet)))))
          ew(j,j)=ew(j,j)+bl*a4*tr4(j,j)/(5.67e-8*ta(j,j+1)**4)
          do 9004 i=j+1,10
           pe=(p(j)+p(i))/2.0
            bet=pe*.211*(300./ta(j,i))
          bl=3.742e-16*(130600.**3)/((exp(1.438*1306./ta(j,i)))-1)
*     bet0=.17 from Ramanathan 1980
          a0=68.2*(ta(j,i)/300)**.858
             u=(c/a0)*ut(j,i)
             a4=200.*a0*log(1+(u/(.106+sqrt(3.59+u*(1+1/bet)))))
          ew(j,i)=ew(j,i)+bl*a4*tr4(j,i)/(5.67e-8*ta(j,i)**4)
9004    continue
9005    continue
        return
        end
*****************************************************************
*************************************************************
* CO2 emissivities from Ramanathan et al. 1983
     subroutine co2 (t,p,ta,tr2,tr3,con2,tco2n2,ew,ac)
     real ta(10,10),tr2(10,10),tco2n2(10,10)
     real tr3(10,10),b1(10)
     real a1
```

```
      real b(10),p(10),t(10)
      real ew(10,10),ac(10,10)
      real a0,a01,con2,bl
*     if (nr.eq.0.or.nr.gt.0) then
*           nr=0
            call bpl(b,t,667.0)
            call bpl(b1,t,1020.0)
      do 320 i=1,9
            a0=(22.18)*sqrt(ta(i,i)/296.0)
            a01=(1.2)*sqrt(ta(i,i)/296.0)
            c=(26.54)*(1.0-1.5*exp(-960.0/ta(i,i)))/sqrt(ta(i,i)/300.0)
            c=c*sqrt(con2/320.)
            c2=1.00+3.41*exp(-480.0/ta(i,i))+7.05*exp(-960.0/ta(i,i))
            u=1.66*.8*con2*(p(i+1)-p(i))
            pave=(p(i+1)+p(i))/2.00
*           call uco2(qc,dq,ec,vc,sc,ta(i,i),pave,u,a0,ab)
            ab=2*a0*log(1.0+c*c2*sqrt(abs(p(i+1)**2-p(i)**2)))
            tco2n2(i,i)=1-ab*100/30000.00
            ew(i,i)=ew(i,i)+tr2(i,i)*b(i)*ab*100./(5.67e-8*ta(i,i)**4)
              ac(i,i)=tr2(i,i)*b(i)*ab*100./(5.67e-8*ta(i,i)**4)
              u=1.66*.8*con2*abs(p(i+1)-p(i))
               w=.05*u/a01
            a1=200*a01*tr3(i,i)*b1(i)*
     x log(1+w/(4+w*(1+1/(pave*.1084*(298/ta(i,i))**.56)))**.5)
            ew(i,i)=ew(i,i)+a1/(5.67e-8*ta(i,i)**4)
           ac(i,i)=ac(i,i)+a1/(5.67e-8*ta(i,i)**4)
           do 319 j=i+1,10
            a0=(22.18)*sqrt(ta(i,j)/296.0)
            a01=(1.2)*sqrt(ta(i,i)/296.0)
            c=(26.54)*(1.0-1.5*exp(-960.0/ta(i,j)))/sqrt(ta(i,j)/300.0)
            c=c*sqrt(con2/320.)
            c2=1.00+3.41*exp(-480.0/ta(i,j))+7.05*exp(-960.0/ta(i,j))
            u=1.66*.8*con2*(p(j)-p(i))
            pave=(p(j)+p(i))/2.00
*           call uco2(qc,dq,ec,vc,sc,ta(i,j),pave,u,a0,ab)
            ab=2*a0*log(1.0+c*c2*sqrt(abs(p(j)**2-p(i)**2)))
            tco2n2(i,j)=1-ab*100/30000.00
            bl=3.742e-16*(64000.**3)/((exp(1.438*640./ta(i,j)))-1)
         ew(i,j)=ew(i,j)+tr2(i,j)*bl*ab*100./(5.67e-8*ta(i,j)**4)
              ac(i,j)=tr2(i,j)*bl*ab*100./(5.67e-8*ta(i,j)**4)
              bl=3.742e-16*(100000.**3)/((exp(1.438*1000./ta(i,j)))-1)
              u=1.66*.8*con2*abs(p(j)-p(i))
               w=.05*u/a01
            a1=200*a01*tr3(i,j)*bl*
     x log(1+w/(4+w*(1+1/(pave*.1084*(298/ta(i,j))**.56)))**.5)
            ew(i,j)=ew(i,j)+a1/(5.67e-8*ta(i,j)**4)
            ac(i,j)=ac(i,j)+a1/(5.67e-8*ta(i,j)**4)
319     continue
```

```
320     continue
*         else
*         do 330 i=1,9
*         ew(i,i)=ew(i,i)+ac(i,i)
*         do 325 j=i+1,10
*         ew(i,j)=ew(i,j)+ac(i,j)
*325     continue
*330     continue
*         end if
*         nr=nr+1
*
*     write (20,399) ((a1(i,j)/10000,j=1,10),i=1,10)
399     format (10(f5.3,2x))
*
    return
    end
*         Ramanathan et. al. 1983
*          a0=(4400.0)*sqrt(ta(j,i)/300.0)
*          c=(26.54)*(1.0-1.5*exp(-960.0/ta(j,i)))/sqrt(ta(j,i)/300.0)
*           c=c*sqrt(con2/320.)
*          c2=1.00+3.41*exp(-480.0/ta(j,i))+7.05*exp(-960.0/ta(j,i))
*          a(i,j)=a0*log(1.0+c*c2*sqrt(abs(p(j)**2-p(i)**2)))
*
*****************************************************************
*    Calculate N2O 1200-1350 cooling via Donner & Ramanathan 1980 see macKay
1990
    subroutine n2o5cool (ta,p,pa,t,tr5,tmc4,n20,ut,ew)
    real p(10),pa(10),t(10),ew(10,10)
    real tr5(10,10),a5,ut(10,10),ta(10,10),tmc4(10,10)
    real pe,bet,a0,u,n20,c,a,x,d,bl
    a=.559
    x=.2
    d=.096
    c=1.66*264*n20*(.239/.30)
*    264 (Goody 1989) is the band strength S .239*(n20/.30) *delta p
*    is the absorber amount in atm-cm n20 is the concemtration
*    of n2o in ppmv
    call ucalc(p,ut,3,a,x,d,.30)
        do 9005 j=1,9
         pe=(p(j)+p(j+1))/2
          bet=pe*1.12*(300./ta(j,j))**.5
        bl=3.742e-16*(128500.**3)/((exp(1.438*1285./ta(j,j)))-1)
*    bet0=1.12 from Ramanathan 1980
        a0=20.4*(ta(j,j)/300)**.5
           u=(c/a0)*ut(j,j)
           a5=200*a0*log(1+(u/sqrt(4+u*(1+1/bet))))
           ew(j,j)=ew(j,j)+bl*a5
   > *tr5(j,j)*tmc4(j,j)/(5.67e-8*ta(j,j)**4)
```

```
        do 9004 i=j+1,10
         pe=(p(j)+p(i))/2.00
          bet=pe*1.12*(300./ta(j,i))**.5
     bl=3.742e-16*(128500.**3)/((exp(1.438*1285./ta(j,i)))-1)
*    bet0=1.12 from Ramanathan 1980
        a0=20.4*(ta(j,i)/300)**.5
           u=(c/a0)*ut(j,i)
           a5=200*a0*log(1+(u/sqrt(4+u*(1+1/bet))))
           ew(j,i)=ew(j,i)+bl*a5
   > *tr5(j,i)*tnc4(j,i)/(5.67e-8*ta(j,i)**4)
9004    continue
9005    continue
        return
        end
*****************************************************************
*    Calculate N2O  520-660  cooling via Donner & Ramanathan 1980 see MacKay 1990
     subroutine n2o6cool (pa,ta,t,tr6,tco2n2,n20,ut,ew)
     real pa(10),t(10),ut(10,10)
     real tr6(10,10),a6,tco2n2(10,10),ta(10,10),ew(10,10)
     real pe,bet,a0,u,n20,c,bl
     c=1.66*24*n20*(.239/.30)
*    24 (Ramanathan 1985) is the band strength S .239*(n20/.30) *delta p
*    is the absorber amount in atm-cm n20 is the concemtration
*    of n2o in ppmv
        do 9005 j=1,9
         pe=(pa(j))
          bet=pe*1.08*(300./ta(j,j))**.5
     bl=3.742e-16*(58900.**3)/((exp(1.438*589./ta(j,j)))-1)
*    bet0=1.12 from Ramanathan 1980
        a0=23.0*(ta(j,j)/300)**.5
           u=(c/a0)*ut(j,j)
           a6=200*a0*log(1+(u/sqrt(4+u*(1+1/bet))))
           ew(j,j)=ew(j,j)+bl
   >    *a6*tr6(j,j)*tco2n2(j,j)/(5.67e-8*ta(j,j)**4)
        do 9003 i=j+1,10
         pe=(pa(j)+pa(i-1))/2.00
          bet=pe*1.08*(300./ta(j,i-1))**.5
     bl=3.742e-16*(58900.**3)/((exp(1.438*589./ta(j,i-1)))-1)
*    bet0=1.12 from Ramanathan 1980
        a0=23.0*(ta(j,i-1)/300)**.5
           u=(c/a0)*ut(j,i)
           a6=200*a0*log(1+(u/sqrt(4+u*(1+1/bet))))
           ew(j,i)=ew(j,i)+bl
   >    *a6*tr6(j,i)*tco2n2(j,i)/(5.67e-8*ta(j,i-1)**4)
9003    continue
9005    continue
        return
        end
```

```
*****************************************************************
* not in use at present.  It was used in the more detailed scheme for CO2 emissivities
* used by MacKay (1990)
      subroutine ucalc (p,ut,nt,a,x,d,c0)
      real u(10),c(10),p(10)
      real ut(10,10)
      integer nt
      do 9520 i=1,9
          if(i.lt.nt)then
            c(i)=(a*((.5*(p(i)+p(i+1)))**x)-d)/c0
           else
            c(i)=1.0
          end if
       u(i)=c(i)*(p(i+1)-p(i))
9520   continue
       do 9540 i=1,9
          ut(i,i)=u(i)
          do 9535 j=i+1,10
          ut(i,j)=0.0
      do 9530 k=i,j-1
       ut(i,j)=ut(i,j)+u(k)
9530   continue
9535   continue
9540   continue
      return
      end
*****************************************************************************
* used in ozone calc below
          function alpha(u,p)
          real alpha,u,p,au,al,beta
          au=(4.1*u)/(1+9.5*u)
          al=.8467*u*(1.9-u)/(1+2.0*u)
          if (p.ge..015) then
          beta=sqrt((p-.015)/.235)
          end if
          if (p.le..015) then
           alpha=(1.085-.085*p)*au
            else
             if((p.ge..015).and.(p.le..25)) then
              alpha=(au**(1-beta))*(al**beta)
              else
               alpha=.6667*(1.75-p)*al
             endif
          endif
          return
          end
*********************************************************
* IR Ozone calcs
```

```
      subroutine o3cool (at,t,pa,uo,tr3,ew,ta)
      dimension  tr3(10,10),ew(10,10),ta(10,10)
      dimension  b(10),  pa(10),uo(50),at(10)
      dimension  ua(10),us(10),a(10),t(10)
      real u,e1,e2,abs,y,alpha,bl
      call bpl(b,t,1042.0)
*
      do 605 i=1,9
          u=(uo(i+1)-uo(i))
         if(u.lt.0.0001) then
         u=0.0001
         end if
          a(i)=alpha(u,pa(i))
          ua(i)=u*1.66*pa(i)**a(i)
605       continue
          do 610 i=1,9
          y=.5138*ua(i)/sqrt(1+3.7145*ua(i))
          call expon (y,e1)
          call expon (17.778*y,e2)
          trn=.3476*(e1-e2)
          abs=13700*(1-trn)*tr3(i,i)
          ew(i,i)=ew(i,i)+abs*b(i)/(5.67e-8*at(i)**4)
          do 608 j=i+1,10
          ut=0.0
          do 606 k=i,j-1
          ut=ut+ua(k)
606       continue
          bl=3.742e-16*(100000.**3)/((exp(1.438*1000./ta(i,j)))-1)
          y=.5138*ut/sqrt(1+3.7145*ut)
          call expon (y,e1)
          call expon (17.778*y,e2)
          trn=.3476*(e1-e2)
          abs=13700*(1-trn)*tr3(i,j)
          ew(i,j)=ew(i,j)+abs*bl/(5.67e-8*ta(i,j)**4)
608       continue
610       continue
*
      return
      end
*
*********************************************************
**************************************************************************
* F-11 and F-12 or other gases of small concentration < 1 ppb with linear
* absorption/emissivities
       subroutine small(ta,p,t,cs0,v0,strength,ew)
       real ta(10,10),p(10),t(10),as
       real ew(10,10)
       real cs0,v0,strength,bl
```

```
      c=1.66*strength*.8*cs0/1e6
      do 9808 i=1,9
           bl=3.742e-16*((v0*100)**3)/((exp(1.438*v0/ta(i,i)))-1)
           as=100*c*(ta(i,i)/300)*abs(p(i+1)-p(i))
           ew(i,i)=ew(i,i)+bl*as/(5.67e-8*ta(i,i)**4)
          do 9807 j=i+1,10
           bl=3.742e-16*((v0*100)**3)/((exp(1.438*v0/ta(i,j)))-1)
           as=100*c*(ta(i,j)/300)*abs(p(i)-p(j))
           ew(i,j)=ew(i,j)+bl*as/(5.67e-8*ta(i,j)**4)
9807      continue
9808      continue
          return
          end
*************************************************************************
*************************************************************
*Change of temperature called by subroutine radiation
      subroutine tempchng(pa,p,cp,at,dg,df,g,step,s1,f,ch,ce,
     xkap,abt,c,at1,rh0,ts,ot,of,u,al,xice,ti,vi,lh,sh,jb,
     & rain,hs,oalb,ialb,lalb,rg,time)
      real pa(10),at(10),dg(10),df(10),f(10),g(10),ch(10)
      real p(10),cp(10),ce(10),c(10),at1(10),h0
      real dt,s1,dr,step,e1,e,q,l,rh0,ts,ot,of,dt9,qs,q0
      real shf,lhf,vi,al,xice,ti,lh,sh,rain,hs,rcond,rg
       real oalb,ialb,lalb,time
      integer kap,jb
      abt=0.0
      et=0.0
       rain=0.0
*       vi=al*al*xice
       h0=cp(10)*86400/4.186e6
      s1=0.0
      do 3400 j=1,9
       e1=esat(at1(j)+1)
       e=esat(at1(j))
*     calculate modified heat capacity Manabe & Wetherald 1967?
      dr=rwat(pa(j),e1,j,kap,rh0)-rwat(pa(j),e,j,kap,rh0)
      l=2510-2.38*(at1(j)-273)
       c(j)=.622*l*l*rwat(pa(j),e,j,kap,rh0)/(1.005*.287*at1(j)**2)
      cp(j)=(1.0+c(j))*1.038165e7*(p(j+1)-p(j))/86400
      dt=step*(dg(j)+((ce(j))*(1.0+c(j))/step)+df(j))
     x/(1.0+c(j))
       et=et+cp(j)*(ce(j))
       s1=s1+abs(dt+ch(j))
       at(j)=at(j)+dt
*
       if(df(j).lt.0.0) then
          rain=rain-10330*(p(j+1)-p(j))*dr*(df(j))/(1.0+c(j))
        end if
```

```
      if(dg(j).lt.0.0) then
        rain=rain-10330*(p(j+1)-p(j))*dr*(dg(j))/(1.0+c(j))
       end if
      if(ce(j).lt.0.0) then
        rain=rain-10330*(p(j+1)-p(j))*dr*(ce(j))/step
       end if
      if(ch(j).lt.0.0) then
        rain=rain-10330*(p(j+1)-p(j))*dr*(ch(j))/step
       end if
*
3400  continue
*
     et=et/step
      dt9=0.0
* sea ice subroutine call
      call vice(vi,al,xice,ot,ti,h0,at,cp,jb,hs,time)
       e=esat(ot)
       qs=rwat(1.0,e,1,1,1.0)
       e=esat(at(9))
       q0=rwat(.998,e,1,1,.77)
    l=(2510-2.38*(at1(9)-273))*1000.
      lhf=l*(qs-q0)*0.0010*u*1.29*(273/at(9))

       shf=0.0010*1005.*u*1.29*(273/at(9))*(ts-at(9))

      q2=(g(10)-et+f(10)-5.67e-8*ts**4-shf)/(21.1*1.)
      ts=ts+q2*step
*      ts=((0.*(ts-ot)
**     & +g(10)*(1.-lalb)/(1-rg)-et+f(10))/5.67e-8)**.25
*     & +g(10)-et+f(10))/5.67e-8)**.25
*        do k=1,3
*      ts=((0.*(ts-ot)
*     & +g(10)-et+f(10)-shf)/5.67e-8)**.25
*     & +g(10)*(1.0-lalb)/(1.-rg)-et+f(10)-shf)/5.67e-8)**.25
*       end do
      dt9=(1-of)*shf+(1.0-al**2)*of*lhf
       sh=(1-of)*shf
       lh=(1.0-al**2)*of*lhf
       if(al.gt.0.01) then
*        ti=ts
      dt9=dt9+of*al*al*shf
      shf=0.0010*1005.*u*1.29*(273/at(9))*(ot-at(9))
      rcond=xice/2.2+hs/.31
      q=(1-al**2)*(g(10)
*       q=(1-al**2)*(g(10)*(1-oalb)/(1-rg)
    & +f(10)-et-shf-lhf-5.67e-8*ot**4)
    & - (ot-at(9))*al*al/rcond
      if (hs.lt.0.1) then
```

```
      q1=(g(10)+f(10)-5.67e-8*ti**4)
*       q1=(g(10)*(1-iabl)/(1-rg)+f(10)-5.67e-8*ti**4)
     & /(21.1*0.2)
        else
      q1=(g(10)+f(10)-5.67e-8*ti**4)
*       q1=(g(10)*(1-iabl)/(1-rg)+f(10)-5.67e-8*ti**4)
     & /(21.1*hs/4.0)
        end if

        q=q*step/(cp(10)*(1-al*al*xice/h0))
        ti=ti+q1*step
        ot=ot+q
         dt9=dt9+
     & of*((1.0-al*al)*shf+(ot-at(9))*al*al/rcond)
        sh=sh+(1.0-al*al)*shf*of
       else
        shf=0.0010*1005.*u*1.29*(273/at(9))*(ot-at(9))
       q=(g(10)+f(10)-et-shf-lhf-5.67e-8*ot**4)
*       q=(g(10)*(1-oabl)/(1-rg)
*      & +f(10)-et-shf-lhf-5.67e-8*ot**4)
        q=q*step/(cp(10)*(1-al*al*xice/h0))
        ot=ot+q
         ti=ot
          dt9=dt9+of*(shf)
         sh=sh+of*shf
         end if
*         if(jb.lt.7.or.jb.gt.12) then
       at(9)=at(9)+dt9*step/cp(9)
*          end if
          ce(10)=-et*step/cp(10)
       at(10)=(of*((1-al*al)*ot**4+al*al*ti**4)
     & +(1-of)*ts**4)**.25
*         of2=of*(1-al*al)
*          at(10)=(9*of2*ot+(1-of2)*ts)/(8*of2+1)
*         at(10)=ot
      return
          end
*************************************************************************
* combine heating and cooling due to all gases
      subroutine combine (dw,dc,do,d4,d5,d6,shw,shoc,
     x sho2,shco2,sho,dg,df)
      real dw(10),dc(10),do(10),shw(10),sho(10),shoc(10)
      real dg(10),df(10),d4(10),d5(10),d6(10),sho2(10),shco2(10)
       do 3500 i=1,9
          df(i)=dw(i)+do(i)+dc(i)+d4(i)+d5(i)+d6(i)
          dg(i)=shw(i)+sho(i)+shoc(i)+shco2(i)+sho2(i)
3500  continue
      return
```

```
      end
*
**********************************************************************
* calculates an average temp between vertical layers Not in use at present
      subroutine tempave(at,p,ta)
      dimension at(10),p(10)
      dimension ta(10,10)
      real s
*
*     calculate the average temperaure to be used for each path
*     using a pressure weighted average Dp=6sigma(1-sigma)
*
      do 600 i=1,10
         do 599 j=i,10
          ta(i,j)=0.0
          s=0.0
          do 590 k=i,j
           ta(i,j)=ta(i,j)+(p(k+1)-p(k))*at(k)
           s=s+(p(k+1)-p(k))
590       continue
          if(s-0.0) 595,595,596
595        ta(i,j)=at(i)
           goto 597
596        ta(i,j)=ta(i,j)/s
597        ta(j,i)=ta(i,j)
599      continue
600   continue
      return
      end
**********************************************************************
**********************************************************
* Plank function
      subroutine bpl(b,t,nu)
      real b(10),t(10)
      real nu
      do 5500 i=1,10
         b(i)=3.742e-16*(nu*100)**3/((exp(1.438*nu/t(i)))-1)
5500  continue
      return
      end
*
**********************************************************
* Not in use at present
      subroutine tempset (t,at,p,kap)
      real t(10),at(10),p(10)
      integer kap
      t(10)=at(10)
      do 4200 i=2,9
```

```
        t(i)=((p(i+1)-p(i))*at(i)+(p(i)-p(i-1))*at(i-1))/
     x(p(i+1)-p(i-1))
4200  continue
        t(1)=at(1)+(.002/.007)*(at(1)-t(2))
      return
      end
*
*****************************************************************
* Not in use at present
      subroutine presset(sg,p,pa)
      real pa(10),sg(10),p(10)
      sg(2)=1.0/18.0
      pa(1)= (sg(2)**2)*(3.0-2.0*sg(2))
      p(1)=0.0
      p(10)=1.0
       pa(10)=1.0
      sg(1)=0.0
      do 4300 i=2,9
          p(i)=p(i-1)+6.0*sg(i)*(1-sg(i))/9.0
          sg(i+1)=sg(i)+(1.00/9.00)
          pa(i)=(sg(i+1)**2)*(3.0-2.0*sg(i+1))
4300    continue
      return
      end
*
*************************************************************
* Sea ice subroutine/model at present we have a formula in here that simulates
* the observations. i.e. sea ice seasonal cycle remains the same.
       subroutine vice(vi,al,xice,to,tice,ho,at,cp,jb,hs,
     & time)
          real vi,al,xice,tice,to,ho,ci,b1,b2,at(10)
       real cp(10),hs,time
        integer jb
         if(jb.eq.1.or.jb.eq.18) then
         ci=0.35
         end if
          if(jb.eq.2.or.jb.eq.17) then
         ci=0.35
          end if
          if(jb.eq.3.or.jb.eq.16) then
          ci=0.5
          end if
          if(jb.gt.3.and.jb.lt.16) then
         ci=0.5
          end if
*           ci=0.35
         dvi=4.186*(272.-to)*1.09*(ho-vi)/(333)
         if(dvi.gt.0.0) then
```

```
          vi=vi+dvi
          to=272.
          end if
          if(dvi.le.0.0.and.vi.gt.0.0) then
              vi=vi+dvi
               to=272
             end if
             if(vi.lt.0.0) then
               vi=0.0
                    end if
*
             if(hs.gt.0.0) then
              if(tice.gt.272) then
                hs=hs+2.1*hs*(272-tice)/333
                tice=272.
               end if
              end if
            if(hs.lt.0.) then
            hs=0.0
             end if
*
           if(vi.gt.0.2) then
           if(tice.gt.272) then
            vi=vi+(272-tice)*0.2*2.1/333
* 33.3 =L*0.1m
*             tice=272.
           end if
           end if
*
          if(vi.ge.0) then
           vcrit=(1/1.4)**(1/(1.4-1))
          if(vi.gt.vcrit) then
          al=ci*vi**0.7
          else
          al=ci*vi**0.50
          end if
****
           if(jb.eq.1) then
           al=sqrt(.82+.18*cos(0.0172*time+0.06))
            xice=2.5+al*al
           end if
            if (jb.eq.2) then
           al=sqrt(.74+.36*cos(0.0172*time+0.06))
           xice=2.5+al*al
           end if
*           if(jb.eq.3) then
*            al=sqrt(.00+.04*cos(0.0172*time+0.06))
*            end if
```

```
          if(jb.eq.16) then
           al=sqrt(.48-.37*cos(0.0172*time+0.16))
             xice=1.0+al*al
          end if
           if(jb.eq.17) then
            al=sqrt(.79-.21*cos(0.0172*time+0.16))
            xice=1.0+al*al
           end if
****
           if(al.gt.1.) then
            al=1.0
           end if
             else
             al=0.0
            end if
          if(al.gt.0.01) then
*         xice=vi/(al*al)
          end if
        return
        end
*************************************************************
*  EBM subroutine for temp gradient driven transport of heat in
* ocean and atmosphere Not in use at present
        subroutine tra(t,th,nb,cp,z2d,step,trs,str,c2)
        real t(10,20),tk(10,20),tp(10,20),th(20),t3(20)
        real trs,t4(20),t5(20),cp(10,20),c2(10,20)
        real dp2d(10,20),z2d(10,20),str
        integer nb
         do 8 i=1,9
         do 7 j=1,nb-1
         if ((z2d(i,j)-z2d(i+1,j)).lt.(z2d(i,j+1)-z2d(i+1,j+1)))then
         dp2d(i,j+1)=(z2d(i,j)-z2d(i+1,j))/100.
         else
         dp2d(i,j+1)=(z2d(i,j+1)-z2d(i+1,j+1))/100.
         end if
7        continue
8        continue
        do 20 i=1,nb
        dp2d(1,i)=dp2d(1,i)/100
     t3(i)=asin((th(i)+th(i+1))/2.0)
     t4(i)=sqrt(1-th(i)**2)
       t5(i)=th(i)-th(i+1)
20      continue
        s2=0.
        do 35 i=1,9
         do 30 j=1,nb
          tk(i,j+1)=dp2d(i,j+1)*t4(j+1)*trs/(t3(j)-t3(j+1))
          tp(i,j)=t(i,j)*cp(i,j)
```

```
*         tp(i,j)=t(i,j)*cp(i,j)/(1+c2(i,j))
30       continue
35        continue
          do 50 i=1,9
           do 45 j=1,nb-1
     t(i,j)=t(i,j)-step*tk(i,j+1)*(tp(i,j)-tp(i,j+1))/(cp(i,j)
    > *t5(j))
*
          t(i,j+1)=t(i,j+1)+step*tk(i,j+1)*(tp(i,j)-tp(i,j+1))/(cp(i,j+1)
    > *t5(j+1))
45        continue
50        continue
          return
       end
*****************************************************************
* Ocean only EBM
          subroutine trans(t1,th,nb,cp,z2d,step,trs,stn,sts,c2,ot,of,
    & tdeep,al)
          real t(10,20),tk(10,20),tp(10,20),th(20),t3(20)
          real trs,t4(20),t5(20),cp(10,20),c2(10,16),po(20),of(20),al(20)
          real dp2d(10,20),z2d(10,20),stn,sts,dmp,ot(20),t1(10,20)
          integer nb
         deepk=2.
        do j=1,nb
         t(10,j)=t1(10,j)
       end do
        do j=1,nb
         t(10,j)=ot(j)
         end do
          do 4 j=2,nb+1
       if(j.lt.10) then
*        dp2d(10,j)=stn*(of(j-1)+of(j))/2.
         dp2d(10,j)=stn
        else
           dp2d(10,j)=sts
*        dp2d(10,j)=sts*(of(j-1)+of(j))/2.
        end if
4         continue
       dp2d(10,10)=(stn+sts)/2.
      dp2d(10,nb+1)=0.0
      dp2d(10,nb)=0.0
*      dp2d(10,nb-1)=sts/4.
*      dp2d(10,nb-2)=sts/2.

         do 20 i=1,nb
         dp2d(1,i)=dp2d(1,i)/100
      t3(i)=asin((th(i)+th(i+1))/2.0)
      t4(i)=sqrt(1-th(i)**2)
```

```
        t5(i)=th(i)-th(i+1)
20      continue
        s2=0.

         do 30 j=1,nb
          tk(10,j+1)=1.0*
     & dp2d(10,j+1)*t4(j+1)*trs/(t3(j)-t3(j+1))
          tp(10,j)=t(10,j)*cp(10,j)
30      continue

       do j=1,nb
          tk(10,j+1)=dp2d(10,j+1)*t4(j+1)/(t3(j)-t3(j+1))
        end do

         do 45 j=1,nb-2
*       if(j.eq.1) then
*      t(10,j)=t(10,j)-0.333*step*
*     &  tk(10,j+1)*(tp(10,j)-tp(10,j+3))/(cp(10,j)*t5(j))
**
*         t(10,j+1)=t(10,j+1)+0.333*step*tk(10,j+1)
*     >*(tp(10,j)-tp(10,j+3))/(cp(10,j+1)
*     > *t5(j+1))
*         end if
*
*       if(j.eq.nb-2) then
*      t(10,j)=t(10,j)-0.333*step*
*     &  tk(10,j+1)*(tp(10,j-2)-tp(10,j+1))/(cp(10,j)*t5(j))
**
*         t(10,j+1)=t(10,j+1)+0.333*step*tk(10,j+1)
*     >*(tp(10,j-2)-tp(10,j+1))/(cp(10,j+1)
*     > *t5(j+1))
*         end if

*       if(j.gt.1.and.j.lt.nb-2) then
      t(10,j)=t(10,j)-step*tk(10,j+1)*(tp(10,j)-tp(10,j+1))/(cp(10,j)
     > *t5(j))
*
         t(10,j+1)=t(10,j+1)+step*tk(10,j+1)
     >*(tp(10,j)-tp(10,j+1))/(cp(10,j+1)
     > *t5(j+1))
*       end if
45      continue
       do j=1,nb-1
        ot(j)=ot(j)+(tdeep-ot(j))*deepk
        tdeep=tdeep+(ot(j)-tdeep)*(deepk/60.)*t5(j)/2.0
         end do
        do j=1,nb
         ot(j)=t(10,j)
```

```
      end do
       return
     end
**************************************************************
**********************************************************************
*  See Roberts et al. 1976
*   Calculates the transmission of water vapor continuum in 8-12
*    micro meter region for CO2 overlap
     subroutine trc2 (u4,t,e,trc)
     real trc,t,e,k,u4
         k=(4.2+5588*exp(-7.87))*exp(1800*(296-t)/(296*t))*e
         trc=(exp(-k*1.66*u4))
     return
     end

**********************************************************************
*  See Mackay (1990)
*   Calculates the transmission of water vapor continuum in 1042 cm-1
*   region for ozone overlap
     subroutine trc3 (u4,t,e,trc)
     real trc,t,e,k,u4
         k=(4.2+5588*exp(-8.20))*exp(1800*(296-t)/(296*t))*e
         trc=(exp(-k*1.66*u4))
     return
     end

**********************************************************************
* Roberts et al. 1976
* water vapor Continuum overlap between wavenumbers nu1 and nu2
     subroutine trcont (nu1,nu2,u4,t,e,trch)
     real trch,nu1,nu2,nu,t,e,k
      integer n1,n2
      n1=int(nu1)
      n2=int(nu2)
      trch=0.0
         do 3000 i=n1,n2-20,20
         nu=i+10
         k=(4.2+5588*exp(-.00787*nu))*exp(1800*(296-t)/(296*t))
    x*e
         trch=trch+(exp(-k*1.66*u4))*20/(nu2-nu1)
3000  continue
     return
     end
**********************************************************************
**************************************************
     subroutine trr1(t,p,u4,tr)
*    transmission of rotation band from 660-800 cm-1 using
*    statistical model Goody 1964, Rodgers & Walshaw 1966
```

```
      real kd,kpa,a,a1,b,b1,phi,t,psi,phb,m,mb
      m=1.66*u4
      kd=.911
      kpa=15.84
      a=.0204
      a1=.0209
      b=-4.89e-5
      b1=-6.87e-5
      phi=exp(a*(t-260)+b*(t-260)**2)
      psi=exp(a1*(t-260)+b1*(t-260)**2)
      mb=phi*m
      phb=psi*p*m/mb
      tr=exp(-(kd*mb)/sqrt(1+kpa*mb/phb))
       return
       end
*****************************************************************
* used by subroutine bcalc
      subroutine expon (y,e1)
      real gamma, e1,y
      gamma=.5772157
      e1=0
          if (y.lt.1.) then
            e1=-log(y)-gamma
            e1=e1+.9999919*y-.2499106*y*y+.0551997*y**3-.0097600*y**4
     x +.0010786*y**5
          else
          e1=(y**4)+8.5733287*y**3+18.0590170*y*y+8.6347609*y+.26777373
          e1=e1/(y**4+9.5733223*y**3+25.6329561*y*y+21.0996531*y
     x      +3.9584969)
          e1=(e1*exp(-y))/y
          endif
          return
          end
*****************************************************************
* used by subroutine overlap below
          subroutine bcalc (t,bt1,bt2,bt6,v1,v2)
*         subroutine for co2 overlap calculation using Kuo 1977
          real bt1,bt2,bt6,t,ts,ts2,v1,v2
          ts=t/100.-2.6
          ts2=ts*ts
          bt1=0.
          bt2=0.
          bt6=0.
          bt=0.
          bt=bt+1.6*.8457*(1-.2569*ts+.1191*ts2)
          bt=bt+.60*.4643*(1-.6739*ts+.36*ts2)
          bt=bt+.60*1.464*(1-.2605*ts+.1307*ts2)
          bt1=bt+(.9+v1-5.)*.927*(1-.1641*ts+.0255*ts2)
```

```
      bt6=(v2-v1)*.927*(1-.1641*ts+.0255*ts2)
      bt2=bt1+bt6
      return
      end
*************************************************
    subroutine overlap (t,p,u4,tch,tr,trx)
    real e1,e2,bt1,bt2,bt6,m
    real t,p,u4,tr,tr1,tr2,tch,trx
    m=1.66*u4*p
       call bcalc (t,bt1,bt2,bt6,5.0,8.0)
       y1=51.8845*sqrt(m)
       y5=y1*exp(-bt1)
       y6=y5*exp(-bt6)
       call expon (y5,e1)
       call expon (y6,e2)
       tch=(e2-e1)/bt6
        call trr1(t,p,u4,tr1)
        tr=tr1
       call bcalc (t,bt1,bt2,bt6,4.8,8.0)
       y1=51.8845*sqrt(m)
       y5=y1*exp(-bt1)
       y6=y5*exp(-bt6)
       call expon (y5,e1)
       call expon (y6,e2)
       trx=(e2-e1)/bt6
        return
        end
*************************************************************
*************************************************************************
*     Calculate H2O overlap in methane region1200-1650 vib
*     rotation & 950-1200 continuum
    subroutine ch4ovl (pa,u4,at,th4,e)
    real pa,u4,th4,e,tr1,tr2,at
       call trr4 (pa,u4,tr1)
       call trcont(920.,1200.,u4,at,e,tr2)
       th4=(280*tr2+450*tr1)/750
    return
    end
*****************************************************************
*    Calculates Vib-rot H20 in CH4 overlap 1200-1650
*    Rodgers & Walshaw 1966
    subroutine trr4(p,u4,tr)
    real p,u4,tr,m,kd,kpa
    m=1.66*u4
    kd=248
    kpa=1276
    tr=exp(-kd*m/sqrt(1+kpa*m/p))
    return
```

```
      end
*****************************************************************
*****************************************************************
*     Calculates Vib-rot H20 in CH4 overlap 1200-1350
*     Rodgers & Walshaw 1966
      subroutine trm(p,u4,tr)
      real p,u4,tr,m,kd,kpa
      m=1.66*u4
      kd=12.65
      kpa=142.13
      tr=exp(-kd*m/sqrt(1+kpa*m/p))
      return
      end
*****************************************************************
      subroutine tr6(t,p,u4,tr)
*     transmission of rotation band from 660-800 cm-1 using
*     statistical model Goody 1964, Rodgers & Walshaw 1966
      real kd,kpa,a,a1,b,b1,phi,t,psi,phb,m,mb
      m=1.66*u4
      kd=9.706
      kpa=162.6
      a=.0168
      a1=.0172
      b=-3.63e-5
      b1=-4.86e-5
      phi=exp(a*(t-260)+b*(t-260)**2)
      psi=exp(a1*(t-260)+b1*(t-260)**2)
      mb=phi*m
      phb=psi*p*m/mb
      tr=exp(-(kd*mb)/sqrt(1+kpa*mb/phb))
      return
      end
*****************************************************************
* water vapor emissivity calculations see MacKay (1990)
      subroutine water   (pa,at,p,t,ta,dw,fw,tr2,ac,kap,tr3,tr4,
     x tr5,tr6,nt,at1,ew,rh0)
      dimension pa(10), p(10), dw(10), t(10), fw(10),at(10)
      dimension  ta(10,10), tr2(10,10)
      dimension fu(10),fd(10),ftot(10),b(10),dr1(10),fr(10),tr3(10,10)
      real tr4(10,10),tr5(10,10),tr6(10,10),at1(10),ew(10,10)
      real sig,cf, em1,u1,e,ac,pe,rh0
      integer kap,nt
      sig=5.67e-8
      call h20trans (pa,at,p,tr2,tr3,tr4,tr5,tr6,kap,at1,ew,ta,rh0)
*
      do 740  j=1,9
              e=esat(at1(j))
              pe=.5*(p(j)+p(j+1))
```

```
            r=rwat(pe,e,j,kap,rh0)
                u2=pe*sqrt(273/at1(j))*r*1033*(p(j+1)-p(j))
            em1=(1-.5*((1/(1+19*sqrt(u2)))+(1/(1+3.5*sqrt(u2)))))
          em1=em1*.59*((273/at1(j))**.25)
            am1=.847*(u2**.022)*em1
            ew(j,j)=.5*(em1+am1)
            do 738 i=j+1,10
            u2=0.0
            do 736 k=j,i-1
            e=esat(at1(k))
            pe=(p(k)+p(k+1))/2.0
            r=rwat(pe,e,k,kap,rh0)
         u2=u2+(pe)*sqrt(273/at1(k))
     >    *r*1033*(p(k+1)-p(k))
736         continue
             em1=(1-.5*((1/(1+19*sqrt(u2)))+(1/(1+3.5*sqrt(u2)))))
           em1=em1*.59*((273/at1(k))**.25)
            am1=.847*(u2**.022)*em1
            em1=.5*(em1+am1)
            ew(j,i)=ew(j,i)+em1
738     continue
740     continue
      return
      end
*
*****************************************************************************
      subroutine h20trans (pa,at,p,tr2,tr3,tr4,tr5,
     x tr6,kap,at1,ew,ta,rh0)
*     calculate h2O transmission
*     tr1 is trans for 660 - 800 cm-1 h2O
*     tr2 is co2 h2o overlap
*     tr3 is ozone overlap
*     tr4 is ch4 h20 overlap
*     tr5 is 1200-1350 n20 h20 overlap
*      tr6 is 520-660 n2o h2o overlap
*     trc is the emissivity of the 800-1200 cm-1 h2O cont.
*     try is the emissivity of h20 cont 400-660 x tr rot
      real pa(10),at(10),p(10),at1(10),r3(10),ew(10,10)
      real tr2(10,10),tr4(10,10),tr5(10,10),tr6(10,10)
      real tr3(10,10),rc(10),ry(10),r2(10),ta(10,10),pe
      real th4,tr,tch,sig,trx,u5,rh0
      integer kap
              sig=5.67e-8
            do 839 i=1,9
                e=esat(at1(i))
                pave=(p(i)+p(i+1))/2.0
                r=rwat(pave,e,i,kap,rh0)
                 e=esat(at1(i))*.77*(pave-.02)/.98
```

```
            if(e.le.0.00) then
            e=.00001
            end if
            u5=r*1033*(p(i+1)-p(i))
          call trcont (800.,1200.,u5,at1(i),e,rc(i))
          call trcont(480.,800.,u5,at1(i),e,ry(i))
            call trcont(580.,740.,u5,at1(i),e,r2(i))
            call overlap (at1(i),pave,u5,tch,tr,trx)
           bl=3.742e-16*(73000.**3)/((exp(1.438*730./at1(i)))-1)
          ew(i,i)=14000.*bl*(1-tr)/(sig*at1(i)**4)
           bl=3.742e-16*(64000.**3)/((exp(1.438*640./at1(i)))-1)
          ew(i,i)=ew(i,i)+32000.*bl*trx*(1-ry(i))/(sig*at1(i)**4)
           bl=3.742e-16*(100000.**3)/((exp(1.438*1000./at1(i)))-1)
          ew(i,i)=ew(i,i)+40000.*bl*(1-rc(i))/(sig*at1(i)**4)
*
            tr2(i,i)=r2(i)
          call trc3(u5,at1(i),e,tr)
          tr3(i,i)=tr
          call ch4ovl(pave,u5,at1(i),th4,e)
          tr4(i,i)=th4
          call trrn (pave,u5,tr)
          tr5(i,i)=tr
          call trr6 (at1(i),pave,u5,tr)
          tr6(i,i)=tr
        do 838 j=i+1,10
            e=esat(at1(i))
            r=rwat(pave,e,i,kap,rh0)
             e=esat(at1(i))*.77*(pave-.02)/.98
            if(e.le.0.00) then
            e=.00001
            end if
            u5=r*1033*(p(i+1)-p(i))
            tave=at1(i)*r*1033*(p(i+1)-p(i))
            pave=.5*(p(i)+p(i+1))*r*1033*(p(i+1)-p(i))
         do 836 k=i+1,j-1
           e=esat(at1(k))
           pe=.5*(p(i)+p(i+1))
           r=rwat(pe,e,k,kap,rh0)
            e=esat(at1(k))*.77*(pe-.02)/.98
           if(e.le.0.00) then
           e=.000001
           end if
           u5=u5+r*1033*(p(k+1)-p(k))
           pave=pave+pe*r*1033*(p(k+1)-p(k))
           tave=tave+at1(k)*r*1033*(p(k+1)-p(k))
836      continue
           tave=tave/u5
           pave=pave/u5
```

```
                e=esat(tave)*.77*(pave-.02)/.98
            if(e.lt.(4.8e-6*pave))then
              e=(4.8e-6*pave)
                end if
                call trcont (800.,1200.,u5,tave,e,rc(i))
                call trcont(480.,800.,u5,tave,e,ry(i))
                call trcont(580.,740.,u5,tave,e,r2(i))
                call overlap (tave,pave,u5,tch,tr,trx)
               bl=3.742e-16*(73000.**3)/((exp(1.438*730./tave))-1)
              ew(i,j)=14000.*bl*(1-tr)/(sig*tave**4)
               bl=3.742e-16*(64000.**3)/((exp(1.438*640./tave))-1)
              ew(i,j)=ew(i,j)+32000.*bl*trx*(1-ry(i))/(sig*tave**4)
               bl=3.742e-16*(100000.**3)/((exp(1.438*1000./tave))-1)
              ew(i,j)=ew(i,j)+40000.*bl*(1-rc(i))/(sig*tave**4)
*
                tr2(i,j)=r2(i)
               call trc3(u5,tave,e,tr)
               tr3(i,j)=tr
               call ch4ovl(pave,u5,tave,th4,e)
              tr4(i,j)=th4
              call trrn (pave,u5,tr)
              tr5(i,j)=tr
               call trr6 (tave,pave,u5,tr)
              tr6(i,j)=tr
838       continue
*
839      continue
              tr2(10,10)=1.0
              tr3(10,10)=1.0
              tr4(10,10)=1.0
              tr5(10,10)=1.0
              tr6(10,10)=1.0
               return
               end

**********************************************************************
*  Used by subroutine solclod below
     subroutine kdist (tau,omeg,tx,rx)
     real tau,omeg,g,u,t,tx,rx
     g=.85
     if (omeg.lt.1.0) then
     t=sqrt(3*(1-omeg)*(1-g*omeg))*tau
     u=sqrt((1-g*omeg)/(1-omeg))
     bot=(u+1)**2-exp(-2*t)*(u-1)**2
     rx=(u+1)*(u-1)*(1.0-exp(-2*t))/bot
     tx=4*u*exp(-t)/bot
     else
     rx=0.0
```

```
      tx=1.0
      end if
      return
      end
*****************************************************************************
*     * solar heating for cloudy portion of the sky
*      via Lacis and Hansen 1974 (revised by Mackay 2/1991)
      subroutine solclod (p,pa,at,xk,pk,theta,s0,rg,kap1,ac1,kap2,
     x ac2,kap3,ac3,dep1,dep2,dep3,shc,abs,gw,at1,rh0)
      real gw(10),p(10),pa(10),at(10),xk(10),pk(10),shc(10)
      real up(10,10),d(10,10),ab(10,10),clh(10,10),at1(10)
      real r1(10,10),r1s(10,10),r19(10,10),t1(10,10),rx(10,10)
      real tx(10,10)
      real tt,s0,s,abs,u1,mu0,e,r,m,theta,tau,omeg,ref,ts,rs
      real ac1,ac2,ac3,ac,dep1,dep2,dep3,rh0
      integer kap1,kap2,kap3
      mu0=cos(theta)
      m=35.0/sqrt(1224.0*mu0**2+1)
       s=s0
       do 4500 j=1,10
        gw(j)=0.0
        shc(j)=0.0
4500   continue
       gw(1)=s
          abs=0.0
          ref=0.0
       do 4700 k=2,8
          do 4510 j=1,9
        rx(j,k)=0.0
        e=esat(at1(j))
           r=rwat(pa(j),e,j,kap,rh0)
        u1=pa(j)*sqrt(273/at(j))*r*1033*(p(j+1)-p(j))*m
        tau=xk(k)*u1
        tx(j,k)=exp(-tau)
4510    continue
          e=esat(at1(kap1))
          r=rwat(pa(kap1),e,kap1,kap1,rh0)
          u1=pa(kap1)*sqrt(273/at(kap1))*r*1033
     >     *(p(kap1+1)-p(kap1))*5/3

          tau=dep1+xk(k)*u1
          omeg=dep1/tau
          ts=tx(kap1,k)
           call kdist(tau,omeg,tx(kap1,k),rx(kap1,k))
          tx(kap1,k)=ac1*tx(kap1,k)+(1-ac1)*ts
          rx(kap1,k)=ac1*rx(kap1,k)
*
         do 4520 j=kap1+1,kap2
```

```
          ac=ac1
          e=esat(at1(j))
          r=rwat(pa(j),e,j,kap1,rh0)
        u1=pa(j)*sqrt(273/at(j))*r*1033*(p(j+1)-p(j))*5/3
          tau= xk(k)*u1
          ts=tx(j,k)
          tx(j,k)=ac*exp(-tau)+(1-ac)*ts
4520    continue
          e=esat(at1(kap2))
          r=rwat(pa(kap2),e,kap2,kap2,rh0)
          u1=pa(kap2)*sqrt(273/at(kap2))*r*1033
     >    *(p(kap2+1)-p(kap2))*5/3
          tau=dep2+xk(k)*u1
          omeg=dep2/tau
          ts=tx(kap2,k)
          rs=rx(kap2,k)

           call kdist(tau,omeg,tx(kap2,k),rx(kap2,k))
          tx(kap2,k)=ac2*tx(kap2,k)+(1-ac2)*ts
          rx(kap2,k)=(1-ac2)*rs+ac2*rx(kap2,k)
*
         do 4525 j=kap2+1,kap3
          ac=1-(1-ac1)*(1-ac2)
          e=esat(at1(j))
          r=rwat(pa(j),e,j,kap1,rh0)
        u1=pa(j)*sqrt(273/at(j))*r*1033*(p(j+1)-p(j))*5/3
          tau= xk(k)*u1
          ts=tx(j,k)
          tx(j,k)=ac*exp(-tau)+(1-ac)*ts
4525    continue


*
          e=esat(at1(kap3))
          r=rwat(pa(kap3),e,kap3,kap3,rh0)
          u1=pa(kap3)*sqrt(273/at(kap3))*r*1033
     >    *(p(kap3+1)-p(kap3))*5/3
          tau=dep3+xk(k)*u1
          omeg=dep3/tau
          ts=tx(kap3,k)
          rs=rx(kap3,k)
          call kdist(tau,omeg,tx(kap3,k),rx(kap3,k))
          tx(kap3,k)=ac3*tx(kap3,k)+(1-ac3)*ts
          rx(kap3,k)=(1-ac3)*rs+ac3*rx(kap3,k)
*
         do 4528 j=kap3+1,9
          ac=1-(1-ac1)*(1-ac2)*(1-ac3)
          e=esat(at1(j))
```

```
          r=rwat(pa(j),e,j,kap1,rh0)
        u1=pa(j)*sqrt(273/at(j))*r*1033*(p(j+1)-p(j))*5/3
          tau= xk(k)*u1
          ts=tx(j,k)
          tx(j,k)=ac*exp(-tau)+(1-ac)*ts
4528    continue
*
      t1(1,k)=tx(1,k)
      r1(1,k)=0.0
      rx(10,k)=rg
      r19(10,k)=rg
      tx(10,k)=0.0
      r1s(1,k)=0.0
       do 4530 j=2,10
        t1(j,k)=t1(j-1,k)*tx(j,k)
        r1(j,k)=r1(j-1,k)+rx(j,k)*t1(j-1,k)**2
        r1s(j,k)=rx(j,k)+r1s(j-1,k)*tx(j,k)**2
*   t1s(j,k) would need to be calculated if there were 2 adjacent
*   cloud layers
4530    continue
       do 4540 j=9,1,-1
        tt=tx(j,k)
      r19(j,k)=rx(j,k)+r19(j+1,k)*tt**2/(1-rx(j,k)*r19(j+1,k))
4540    continue
       do 4550 j=1,9
        up(j,k)=t1(j,k)*r19(j+1,k)/(1-r1s(j,k)*r19(j+1,k))
        d(j,k)=t1(j,k)/(1-r1s(j,k)*r19(j+1,k))
         gw(j+1)=gw(j+1)+s*pk(k)*d(j,k)
          ab(j,k)=pk(k)*(1-r1(10,k)+up(j,k)-d(j,k))
4550    continue
         abs=abs+ab(9,k)*s*mu0
         ref=ref+r19(1,k)*s*mu0*pk(k)
     clh(1,k)=ab(1,k)
        shc(1)=shc(1)+clh(1,k)*(.0083224/(p(2)-p(1)))*s*mu0
       do 4560 j=2,9
        clh(j,k)=ab(j,k)-ab(j-1,k)
        shc(j)=shc(j)+clh(j,k)*(.0083224/(p(j+1)-p(j)))*s*mu0
4560    continue
       ab(10,k)=t1(9,k)*(1-rg)*pk(k)
       shc(10)=shc(10)+ab(10,k)
4599    format (7(f6.4,3x))
4600    format (9(f6.4,3x))
4700    continue
      gw(10)=shc(10)*mu0*s
          abs=abs+gw(10)
     return
     end
*****************************************************************************
```

```
* Calculates u'v', t'v', t'w', q'w', q'v' from the algorithims given by Yao and Stone
* (1987) and Stone and Yao (1987) and (1990)
        subroutine yao(pa,t,u,p,z2,z,phb,uvp,nb,dp,rh,c2,tvp,ytr
     &  ,ytt,twp,cwat,qwp,tv1)
*  subroutine to calculate eddie transport of momentum
* via method of Yao and Stone 1987
        real pa(10,20),t(10,20),u(10,20),v(10,20),xk(10,20),p(10,20)
        real fc(20),zb(10),za(10),z(10,20),z2(10,20),dp,phb(20),pha(20)
        real a,bt(20),thp(10,20),kp,h,d1,d2,uvs(20),uv(20),ub(20)
        real uvp(10,20),rh(20),c2(10,20),sig,sg1,ytt,tv1(10,20)
      real tvp(10,20),ytr,twp(10,20),cwat(10,20),qwp(10,20)
        integer nb
*       nb=9
        kp=287./1005.
        a=6.4e6
       pa(10,nb+1)=pa(10,nb)
        do i=1,10
       pa(i,nb+1)=pa(i,nb)
       cwat(i,nb+1)=cwat(i,nb)
       t(i,nb+1)=t(i,nb)
        do j=1,nb+1
        thp(i,j)=t(i,j)*(pa(10,j)/pa(i,j))**kp
        end do
        end do
     do i=1,nb+1
     fc(i)=4.0*3.1415926*sin(phb(i))/86400.
     bt(i)=4.0*3.1415926*cos(phb(i))/86400./a
     end do
*****
     uv(1)=0.0
     uv(2)=0.0
     uv(nb)=0.0
     uv(nb-1)=0.0
*      uv(nb-2)=0.
     uv(nb+1)=0.0
     uvs(nb+1)=0.0
      uvs(10)=0.0
*
     do j=2,9
     call yaoav(xk,j,h,d1,d2,z2,z,u,(fc(j)+fc(j+1))/2.
    & ,(bt(j)+bt(j+1))/2.,t,thp,purz,pa,p,kp,uv(j),ub(j),a,dp
    & ,rh,c2,tvp,ytr,fc(j),bt(j),ytt,twp,cwat,qwp,tv1)
        end do
     uv(2)=0.0
*
        s1=0.0
        do j=9,1,-1
        uvs(j)=uvs(j+1)+a*dp*uv(j)
```

```
          if(ub(j).gt.0) then
           s1=s1+a*dp*(bt(j)+bt(j+1))/2.*1.013e5
          end if
          if (ub(j).le.0.and.(ub(j+1).gt.0.or.ub(j-1).gt.0.)) then
          s1=s1+a*dp*(bt(j)+bt(j+1))/2.*1.013e5/16.
          end if
          end do

          xk0=-uvs(2)/s1
          do j=1,9
          if(ub(j).gt.0) then
           uv(j)=uv(j)+xk0*(bt(j)+bt(j+1))/2.*1.013e5
          end if
          if (ub(j).le.0.and.(ub(j+1).gt.0.or.ub(j-1).gt.0)) then
           uv(j)=uv(j)+xk0*(bt(j)+bt(j+1))/2.*1.013e5/16.
          end if
          end do
          do j=9,1,-1
          uvs(j)=uvs(j+1)+a*dp*uv(j)
       end do
*
      do j=10,nb
      call yaoav(xk,j,h,d1,d2,z2,z,u,(fc(j)+fc(j+1))/2.
     & ,(bt(j)+bt(j+1))/2.,t,thp,purz,pa,p,kp,uv(j),ub(j),a,dp
     & ,rh,c2,tvp,ytr,fc(j),bt(j),ytt,twp,cwat,qwp,tv1)
          end do
       uv(nb)=0.0
       uv(nb-1)=0.0
*       uv(nb-2)=0.0
*
        uvs(10)=0.0
         s1=0.0
         do j=10,nb
         uvs(j+1)=uvs(j)+a*dp*uv(j)
         if(ub(j).gt.0) then
          s1=s1-a*dp*(bt(j)+bt(j+1))/2.*1.013e5
         end if
         if (ub(j).le.0.and.(ub(j+1).gt.0.or.ub(j-1).gt.0.)) then
         s1=s1-a*dp*(bt(j)+bt(j+1))/2.*1.013e5/16.
         end if
         end do
         xk0=-uvs(nb)/s1
         do j=10,nb
         if(ub(j).gt.0) then
          uv(j)=uv(j)-xk0*(bt(j)+bt(j+1))/2.*1.013e5
         end if
         if (ub(j).le.0.and.(ub(j+1).gt.0.or.ub(j-1).gt.0)) then
          uv(j)=uv(j)-xk0*(bt(j)+bt(j+1))/2.*1.013e5/16.
```

```
        end if
        end do
        do j=10,nb
        uvs(j+1)=uvs(j)+a*dp*uv(j)
      end do
*      do i=1,10
*       tvp(i,10)=0.0
*       end do
*
        call vertdis(uvp,z2,uvs,p,nb)
*       do i=8,9
*          tvp(i,18)=0.0
*          twp(i,18)=0.0
*           qwp (i,18)=0.0
*           uvp(i,18)=0.0
*            end do
*         tvp(9,17)=0.0
*         twp(9,17)=0.0
*           qwp(9,17)=0.0
*            uvp(9,17)=0.0
      goto 59
*       write(*,*)
*       write(*,1000) (1e5*fc(j),j=1,10)
*      write(*,1000) (1e5*fc(j),j=nb+1,10,-1)
*       write(*,1000) (1e10*bt(j),j=1,10)
*      write(*,1000) (1e10*bt(j),j=19,10,-1)
*       write(*,1000) (uvs(j)/1.013e5,j=1,10)
*       write(*,1000) (uvs(j)/1.013e5,j=nb+1,10,-1)
        write(*,1000) (uv(j),j=1,9)
        write(*,1000) (uv(j),j=nb,10,-1)
*       write(*,1000) (ub(j),j=1,nb+1)
*      write(*,*) xk0/1e6
      write(*,*)
*       do i=1,10
*         write(*,1000) (uvp(i,j),j=1,10)
*         write(*,1000) (uvp(i,j),j=nb+1,10,-1)
*         write(*,*)
*         end do
*       read(*,*)
1000    format (2x,19(f7.2,1x))
1001    format(2x,19(e8.1,1x))
59      end
**********************************
        subroutine yao(pa,t,u,p,z2,z,phb,uvp,nb,dp,rh,c2,tvp,ytr
   &  ,ytt,twp,cwat,qwp,tv1)
*  subroutine to calculate eddie transport of momentum
* via method of Yao and Stone 1987
        real pa(10,20),t(10,20),u(10,20),v(10,20),xk(10,20),p(10,20)
```

```
        real fc(20),zb(10),za(10),z(10,20),z2(10,20),dp,phb(20),pha(20)
        real a,bt(20),thp(10,20),kp,h,d1,d2,uvs(20),uv(20),ub(20)
        real uvp(10,20),rh(20),c2(10,20),sig,sg1,ytt,tv1(10,20)
      real tvp(10,20),ytr,twp(10,20),cwat(10,20),qwp(10,20)
       real prz(20),zr(20),xr(20)
        integer nb
*       nb=9
        kp=287./1005.
        a=6.4e6
       pa(10,nb+1)=pa(10,nb)
        do i=1,10
       pa(i,nb+1)=pa(i,nb)
       cwat(i,nb+1)=cwat(i,nb)
       t(i,nb+1)=t(i,nb)
        do j=1,nb+1
        thp(i,j)=t(i,j)*(pa(10,j)/pa(i,j))**kp
        end do
        end do
     do i=1,nb+1
* fc is the coriolis parameter
* bt is beta partial of f wrt y
     fc(i)=4.0*3.1415926*sin(phb(i))/86400.
     bt(i)=4.0*3.1415926*cos(phb(i))/86400./a
     end do
*****
* uv is the partial of u'v' wrt y the meridional coordinate
     uv(1)=0.0
     uv(2)=0.0
     uv(nb)=0.0
     uv(nb-1)=0.0
*      uv(nb-2)=0.
     uv(nb+1)=0.0
     uvs(nb+1)=0.0
      uvs(10)=0.0
*
     do j=2,10
     call yaoav(xk,j,h,d1,d2,z2,z,u,(fc(j)+fc(j+1))/2.
    & ,(bt(j)+bt(j+1))/2.,t,thp,purz,pa,p,kp,uv(j),ub(j),a,dp
    & ,rh,c2,tvp,ytr,fc(j),bt(j),ytt,twp,cwat,qwp,tv1)
* ub is the vertical mean value of u
       prz(j)=purz
        end do
      uv(2)=0.0
* zero calcs the latitude where westerlies turn into easterlies i.e. ub=0
       call zero(ub,zr)
       zr(8)=1.0*zr(8)**3
       zr(9)=1.0*zr(9)**3
       zr(10)=1.0*zr(10)**3
```

```
        call zero(prz,xr)
        do j=1,10
         uv(j)=uv(j)*xr(j)
        end do
*
        s1=0.0
        do j=9,1,-1
        uvs(j)=uvs(j+1)+a*dp*uv(j)
         s1=s1+a*dp*zr(j)*(bt(j)+bt(j+1))/2.*1.013e5
        end do

        xk0=-uvs(2)/s1
        do j=1,9
         uv(j)=uv(j)+xk0*zr(j)*(bt(j)+bt(j+1))/2.*1.013e5
        end do
        do j=9,1,-1
        uvs(j)=uvs(j+1)+a*dp*uv(j)
      end do
*
     do j=10,nb
     call yaoav(xk,j,h,d1,d2,z2,z,u,(fc(j)+fc(j+1))/2.
    & ,(bt(j)+bt(j+1))/2.,t,thp,purz,pa,p,kp,uv(j),ub(j),a,dp
    & ,rh,c2,tvp,ytr,fc(j),bt(j),ytt,twp,cwat,qwp,tv1)
      prz(j)=purz
         end do
      uv(nb)=0.0
      uv(nb-1)=0.0
      call zero(ub,zr)
       zr(10)=1.0*zr(10)**3
        zr(11)=1.0*zr(11)**3
        zr(12)=1.0*zr(12)**3
        do j=10,nb
         uv(j)=uv(j)*xr(j)
        end do

*       uv(nb-2)=0.0
*
        uvs(10)=0.0
         s1=0.0
         do j=10,nb
         uvs(j+1)=uvs(j)+a*dp*uv(j)
          s1=s1-zr(j)*a*dp*(bt(j)+bt(j+1))/2.*1.013e5
         end do
         xk0=-uvs(nb)/s1
         do j=10,nb
          uv(j)=uv(j)-zr(j)*xk0*(bt(j)+bt(j+1))/2.*1.013e5
         end do
         do j=10,nb
```

```
        uvs(j+1)=uvs(j)+a*dp*uv(j)
      end do
*     do i=1,10
*      tvp(i,10)=0.0
*      end do
*
       call vertdis(uvp,z2,uvs,p,nb)
*       do i=8,9
*          tvp(i,18)=0.0
*          twp(i,18)=0.0
*           qwp (i,18)=0.0
*           uvp(i,18)=0.0
*            end do
*         tvp(9,17)=0.0
*         twp(9,17)=0.0
*           qwp(9,17)=0.0
*            uvp(9,17)=0.0
      goto 59
*       write(*,*)
*       write(*,1000) (1e5*fc(j),j=1,10)
*      write(*,1000) (1e5*fc(j),j=nb+1,10,-1)
*       write(*,1000) (1e10*bt(j),j=1,10)
*      write(*,1000) (1e10*bt(j),j=19,10,-1)
*       write(*,1000) (uvs(j)/1.013e5,j=1,10)
*       write(*,1000) (uvs(j)/1.013e5,j=nb+1,10,-1)
       write(*,1000) (uv(j),j=1,9)
       write(*,1000) (uv(j),j=nb,10,-1)
*       write(*,1000) (ub(j),j=1,nb+1)
*      write(*,*) xk0/1e6
      write(*,*)
*       do i=1,10
*         write(*,1000) (uvp(i,j),j=1,10)
*         write(*,1000) (uvp(i,j),j=nb+1,10,-1)
*         write(*,*)
*         end do
*       read(*,*)
1000    format (2x,19(f7.2,1x))
1001    format(2x,19(e8.1,1x))
59      end
**********************************
     subroutine yaoav(xk,j,h,d1,d2,z2,z,u,f,b,t,thp,purz,pa
    & ,p,kp,uv,ub,a,dp,rh,c2,tvp,ytr,f1,b1,ytt,twp,cwat,qwp,tv1)
     real s3,s1,s2,h,yn2,f,d1,d2,xk(10,20),e,e1,tvp(10,20)
     real ga,b,purz,t(10,20),thp(10,20),z2(10,20),u(10,20)
     real p(10,20),pa(10,20),kp,a,dp,rh(20),c2(10,20)
      real thya2,thza,z(10,20),yn,ub,ptrz,ytr,f1,b1,ga1
       real da1,da2,pur1,sa1,sa2,h1,yan2,yan,ca,p1r,ytt
     real twp(10,20),thy3,adbm,l,de,ptz,s4,s5,ryao,tv1(10,20)
```

```
      real ylmd,chi,cwat(10,20),qwp(10,20),qy,qz,zd,qeff,qsy
      integer ncnt
      ncnt=0
      yn2=(9.8/thp(8,j))*(thp(5,j)-thp(8,j))/((z2(5,j)-z2(8,j))*1.0)
      purz=(u(8,j)-u(5,j))/(z2(5,j)-z2(8,j))
      yan2=yn2
       if (abs(f1).lt.1.2e-5) then
        f1=1.2e-7
       end if
      h=287.*t(4,j)/9.8
      h1=h
      d1=h/2.
      d2=h/2.
      da1=d1
      da2=d2
          ga=1.0
       if(pur1.gt.0) then
          ga1=b1*h1*yn2/(pur1*f1**2.0)
       end if
          if (ga1.gt.160.) then
          ga1=160.
          end if
       ga=ga1
          d1=h/(1.48*ga+.48)
          d2=h/(1.0+ga)
      da1=h1/(1.48*ga1+.48)
      da2=h1/(1.0+ga1)


30        d1=d1-(d1-h/(1.48*ga+.48))/2.5
          d2=d2-(d2-h/(1.0+ga))/2.5
      da1=da1-(da1-h1/(1.48*ga1+.48))/2.5
      da2=da2-(da2-h1/(1.0+ga1))/2.5
       if (j.eq.13) then
        end if
          s1=0.0
      sa1=0.0
          s2=0.0
       sa2=0.0
      s3=0.0
      s4=0.0
       s5=0.0
          do i=3,8
          s1=s1+0.50*(u(i-1,j)+u(i-1,j+1)-u(i,j+1)-u(i,j))
     & *exp(-z(i,j)/d1)
          sa1=sa1+(u(i-1,j)-u(i,j))
     & *exp(-z(i,j)/da1)
```

```
          s3=s3+0.50*(thp(i-1,j)+thp(i-1,j-1)-thp(i,j-1)-thp(i,j))
     & *exp(-z(i,j)/d1)
          s5=s5+0.50*(t(i-1,j)+t(i-1,j-1)-t(i,j-1)-t(i,j))
     & *exp(-z(i,j)/d1)

        l=2510.-2.38*(.5*(t(i-1,j)+t(i,j))-273)
        e=esat(0.5*(t(i-1,j)+t(i,j)))
         de=.622*l*e*1e5/(.287*(.5*(t(i-1,j)+t(i,j)))**2)
       s4=s4+(9.8*(1+.622*l*e*1e5/(p(i,j)*0.287*
     &  .5*(t(i-1,j)+t(i,j))))/(1+(0.622*l*de)
     &  /(1.005*p(i,j))))*
     & (z2(i-1,j)-z2(i,j))*exp(-z(i,j)/d1)

          sa2=sa2+(z2(i-1,j)-z2(i,j))*exp(-z(i,j)/da1)
          s2=s2+(z2(i-1,j)-z2(i,j))*exp(-z(i,j)/d1)
          end do
       purz=s1/s2
       pur1=sa1/sa2
       ptrz=s3/s2
       ptz=s5/s2
       adbm=s4/s2

*
          s1=0.0
       sa1=0.0
          do i=3,8
          s1=s1+(287.*(t(i-1,j)+t(i,j))*.5/9.8)
     & *exp(-z(i,j)/d1)*(z2(i-1,j)-z2(i,j))
          sa1=sa1+(287.*(t(i-1,j)+t(i,j)+t(i-1,j-1)+t(i,j-1))
     &*.25/9.8)
     & *exp(-z(i,j)/da1)*(z2(i-1,j)-z2(i,j))

          end do
          h=s1/s2
       h1=sa1/sa2
*
          s1=0.0
       sa1=0.0
          do i=3,8
          s1=s1+(2.0*9.8/(thp(i,j)+thp(i-1,j)))
     & *(thp(i-1,j)-thp(i,j))
     & *exp(-z(i,j)/d1)
          sa1=sa1+(2.0*9.8
     & /(thp(i,j)+thp(i-1,j)+thp(i,j-1)+thp(i-1,j-1)))
     & *(thp(i-1,j)+thp(i-1,j-1)-thp(i,j-1)-thp(i,j))
     & *exp(-z(i,j)/da1)

          end do
```

```
*
        yn2=s1/s2
      yan2=sa1/sa2
*       yn=s3/s2
        yn=sqrt(yn2)
        yan=sqrt(yan2)
* parameters from Stone and Yao 1990 page 733,734
       ryao=(adbm/9.8)*(ptz+adbm/1000.)/(ptz+0.0098)
    & +purz**2/yn2
       if(ryao.lt.0.1) then
         ryao=0.1
       end if
       ylmd=0.573*sqrt(ryao)/(1-0.427*ryao**.302)
       chi=(1+ylmd/ryao)/(1+ylmd)
        qeff=+1.005/1*(ptz+adbm/1000.)*(1-adbm/9.8)

*
        s1=0.0
        do i=3,8
     s1=s1+.25*(thp(i,j-1)+thp(i-1,j)+thp(i-1,j-1)+thp(i,j))
    & *exp(-z(i,j)/da1)*(z2(i-1,j)-z2(i,j))
        end do
       thza=s1/sa2
       s1=0.0
        s3=0.0
      do i=3,8
    s1=s1+((thp(i,j-1)+thp(i-1,j-1)-thp(i-1,j)-thp(i,j))
*    & /(2.0*a*dp))**2.
    & /(2.0*a*dp))
    & *exp(-z(i,j)/da1)*(z2(i-1,j)-z2(i,j))
      end do
*       thya2=(s1/sa2)
       thya2=(s1/sa2)*abs(s1/sa2)
        thy3=thya2*(s1/sa2)
*       print*,s2,d1
      if(purz.gt.0.00001) then
        ga=b*h*yn2/(purz*f**2.0)
      end if
        if (ga.gt.40.) then
        ga=40.
        end if
*
     if(pur1.gt.0.00001) then
        ga1=b1*h1*yan2/(pur1*f1**2.0)
     end if
        if (ga1.gt.40.) then
        ga1=40.0
        end if
```

```
          ncnt=ncnt+1
          if(ncnt.lt.10) goto 30
        ub=0.0
*         d2=h/(1+ga)
*        da2=h1/(1+ga1)
          uv=0.0
          s2=0.0
*        print*,j,thya2,thza,da1
*        print*,j,d1,d2

          do i=1,9
*         if(i.lt.3.or.abs(9.5-j).lt.3.)then
*         tvp(i,j)=
*      & +0.0*ytt*((t(i,j-1)-t(i,j))
*      & /(dp))**2.
*         else
**        tvp(i,j)=
*        tvp(i,j)=-exp(-z2(i,j)/h1)*
*      & ytt*((t(i,j-1)+t(i-1,j-1)-t(i-1,j)-t(i,j))
*      & /(2.0*dp))*abs(t(i,j-1)+t(i-1,j-1)-t(i-1,j)-t(i,j))
*      & /(2.0*dp)
*         end if

*           if(pur1.gt.0.) then
          if(abs(10.-j).lt.3.1) then
*         if(i.lt.4) then
*        tvp(i,j)=-1.0*(1-exp(-z2(i,j)/450))*exp(-z2(i,j)/da1)*
        tvp(i,j)=-1.0*(1-exp(-z2(i,j)/450))*
      & ytt*((t(i,j-1)-t(i,j))
      & /(1.0*dp))*abs(t(i,j-1)-t(i,j))
      & /(1.0*dp)
        else
*        tvp(i,j)=-1.0*(1-exp(-z2(i,j)/450))*exp(-z2(i,j)/da1)*
        tvp(i,j)=-1.0*(1-exp(-z2(i,j)/450))*
      & ytt*((t(i,j-1)-t(i,j))
      & /(1.0*dp))*abs(t(i,j-1)-t(i,j))
      & /(1.0*dp)
        end if
*         end if
*         else
*        tvp(i,j)=-1.0*(1-exp(-z2(i,j)/450))*
*      & ytt*((t(i,j-1)+t(i-1,j-1)-t(i-1,j)-t(i,j))
*      & /(2.0*dp))*abs(t(i,j-1)+t(i-1,j-1)-t(i-1,j)-t(i,j))
*      & /(2.0*dp)
*         end if
*      & ytt*((t(i,j-1)+t(i-1,j-1)-t(i-1,j)-t(i,j))
```

```
*     & /(2.0*dp))**2.
*           end if
      if(pur1.gt.0.0.and.j.ne.10) then
*       tvp(i,j)=(pa(i,j)/pa(10,j))**kp*
            ca=(c2(i,j)+c2(i,j-1))/2.
          p1r=(pa(i,j)+pa(i,j-1))/(pa(10,j)+pa(10,j-1))
         tv1(i,j)=-1.0*(p1r)**kp*
     &  (1+ca)*.60*9.8*da2**2.*yan*thya2/(thza*f1**2)
     & *exp(-z2(i,j)/da1)*(1-exp(-z2(i,j)/450.))
         tvp(i,j)=tvp(i,j)+tv1(i,j)
          end if
         if(purz.gt.0.0.and.j.ne.10) then
        twp(i,j)=chi*0.6*9.8*9.8*d2**2*thy3*
     & (z(i,j)/d1-(z(i,j)/(2*d1))**2)*exp(-z(i,j)/d1)
     &  /(yn*f**2*thza**2)
           e=esat((t(i-1,j-1)+t(i,j-1))*.5)*.622*1.013e5/
     & (pa(i-1,j-1)+pa(i,j-1))/2.
           e1=esat((t(i-1,j+1)+t(i,j+1))*.5)*.622*1.013e5/
     & (pa(i-1,j+1)+pa(i,j+1))/2.
         qz=(cwat(i-1,j)-cwat(i,j))/(z2(i-1,j)-z(i,j))
      qy=(cwat(i-1,j-1)+cwat(i,j-1)-cwat(i-1,j+1)-cwat(i,j+1))
     & /(2*a*dp)
       qsy=(e-e1)/(2*a*dp)
        zd=z(i,j)/d1
        if(abs(qy).gt.0.00) then
*         if(i.gt.3.and.j.gt.1.and.j.lt.nb) then
         if(i.gt.3) then
       qwp(i,j)=tv1(i,j)*(1./(1.+ylmd))*f*purz*zd/(yn2)*
     & (1.0
     &  +zd*f*purz*qz/qy/yn2/4.
     &  +(ylmd/ryao)*(1+zd/4.*f*purz/ryao/yn2*qeff/qsy))
         else
          qwp(i,j)=0.0
         end if
        end if
        end if
*         if(j.eq.6) then
*        print*,i,ptz,adbm/1000.,qeff,qsy,qz,qy
*         end if

*        if(purz.gt.0.) then
         xk(i,j)=.707*purz*yn*d2**2*exp(-z2(i,j)/d1)/f
*         else
*         xk(i,j)=0.0
*          end if
         end do
*        print*,j,thya2,yan,pur1
*        if (j.gt.9) then
```

```
*        do i=1,9
*        tvp(i,j)=-tvp(i,j)
*        end do
*        end if
*      if (j.gt.7.and.j.lt.12) then
*       print*,j,pur1,purz
*       end if
*         if(purz.le.0.) then
*         uv=0.0
*         goto 60
*         end if

          s1=0.
**********************************************************************
          do i=3,9
          sig=(thp(i,j)-thp(i-1,j))/((pa(i,j)-pa(i-1,j)))
*
      e=esat(t(6,j))
      e1=esat(t(5,j))
      sg1=(lfus(t(i,j))/1005.)*((pa(10,j)/pa(i,j))**kp)
     &*(rwat(pa(6,j)/1.013e5,e,j,1,rh(j))
     & -rwat(pa(5,j)/1.013e5,e1,j,1,rh(j)))
     & /((pa(6,j)-pa(5,j)))
*
*      print*,sig,sg1,i
      sg=sig+sg1
*        sg=sig
      if(sg.eq.0.) then
      sg=-4./(2.0*(p(i+1,j)-p(i,j)))
      end if
          xd=(thp(i,j-1)-thp(i,j+1))*f
     & *(xk(i,j)-xk(i-1,j))/(a*2.*dp)
          uv=uv+xk(i,j)*(pa(i,j)-pa(i-1,j))*(b
     &-.5*(u(i,j-1)-u(i,j)-u(i,j+1)+u(i,j+2))/(a*dp)**2.)
     &    -(1.0+c2(i,j))*xd/sg
*
*         uv=uv+1.0*xk(i,j)*(p(i+1,j)-p(i,j))*(b
*     &  -(u(i,j-1)+u(i,j+1)-2.*u(i,j))/(a*dp)**2.)
*     &  -1.0*(1.0+c2(i,j))*xd/sg
*       end if
          s1=s1+xd
*      print*,thya2,thza,yn2,d2**2.
          end do
*        print*,j,uv
*       print*,j,(u(4,j-1)-u(4,j)-u(4,j+1)+u(4,j+2))
60      do i=2,8
       ub=ub+.5*(u(i,j)+u(i,j+1))*(p(i+1,j)-p(i,j))
          s2=s2+(p(i+1,j)-p(i,j))
```

```
      end do
      ub=ub/s2
*        print*,j,uv,purz,sg1,sig

         return
         end
*************************
      subroutine vertdis(uvp,z2,uv,p,nb)
      real uv(20),uvp(10,20),p(10,20),z2(10,20)
      integer nb
      do j=2,nb
      s1=0.0
      do i=2,9
           if (z2(i,j).le.10000.) then
           s1=s1+(p(i+1,j)-p(i,j))*exp(-(z2(i,j)/1000.-10.)**2./45.)
           else
           s1=s1+(p(i+1,j)-p(i,j))*exp(-(z2(i,j)/1000.-10.)**2./15.)
            end if
      end do
      do i=2,9
           if (z2(i,j).le.10000.) then
      uvp(i,j)=uv(j)*exp(-(z2(i,j)/1000.-10.)**2./45.)/(s1*1.0)
           else
      uvp(i,j)=uv(j)*exp(-(z2(i,j)/1000.-10.)**2./15.)/(s1*1.0)
           end if
      end do
      end do

     return
     end
*****************
      function lfus(t)
      real t
      lfus=1000.*(2510.-2.38*(t-273))
      return
      end
************************************
* locates where the zero crossing of ub is
         subroutine zero( ub,zr)
       real a1,ub(20),zr(20)
          do j=1,18
          zr(j)=0.0
       if(ub(j).gt.0) then
          if(j.eq.18) then
              zr(j)=zr(j)+0.5
           end if
        if(j.lt.18) then
           if(ub(j+1).gt.0) then
```

```
                zr(j)=zr(j)+.5
              end if
             if(ub(j+1).lt.0) then
              a1=1.0+ub(j+1)/(ub(j)-ub(j+1))+1/16
               if(a1.gt.0.5) then
                    zr(j)=zr(j)+0.5
               else
                     zr(j)=zr(j)+a1
               end if
             end if
           end if
             if(j.eq.1) then
             zr(j)=zr(j)+.5
             end if
              if(j.gt.1) then
             if(ub(j-1).gt.0) then
              zr(j)=zr(j)+.5
              end if
               if(ub(j-1).lt.0) then
               a1=1.0+ub(j-1)/(ub(j)-ub(j-1))+1/16
               if(a1.gt.0.5) then
               zr(j)=zr(j)+.5
               else
               zr(j)=zr(j)+a1
               end if
             end if
            end if
          end if
*
*
*
          if(ub(j).lt.0) then
            if(j.lt.18) then
              if(ub(j+1).gt.0) then
               a1=.5+ub(j)/(-ub(j)+ub(j+1))+1/16
             if(a1.gt.0.0)then
                    zr(j)=zr(j)+a1
              end if
            end if
          end if
              if(j.gt.1) then
                    if(ub(j-1).gt.0) then
                    a1=.5+ub(j)/(-ub(j)+ub(j-1))+1/16
               if(a1.gt.0.0) then
               zr(j)=zr(j)+a1
               end if
            end if
           end if
```

```
end if
end do
return
end
```

VITA

Robert MacKay was born on 21 September 1956 in Oroville, California. He has earned a Bachelors of Art Degree in Physics and Mathematics from The California State University at Chico in 1978, a Masters of Science in Physics from Portland State University in 1983, and a Masters in Atmospheric Physics from The Oregon Graduate Institute of Science & Technology in 1991. In 1983 he accepted his present position as a tenured track faculty member at Clark College in Vancouver, and was awarded tenure in 1986. He has coupled his experience in the atmospheric sciences and teaching by developing interactive computer aided learning activities for enhancing student understanding of science concepts. Example of these learning packages have been presented recently at The Aspen Global Change Institute (1992) and 1st International Conference on Computer-aided Learning and Distance Learning in meteorology, Hydrology, and Oceanography (1993). After the completion of his Ph.D. he plans to continue his research activities in Global Change Science and Continue his career as a teacher.

Publications and Presentations

Khalil, M.A.K., Zhao W., and R.M. MacKay, 1990. The Relationship between Temperature and Precipitation. Encyclopedia of Earth System Science. Academic Press, San Diego California.

MacKay, R.M. and M.A.K. Khalil, 1990. Free Oscillations of the Earth Climate System: A Theory of the 100 kyr Climate Cycle. Annals of Glaciology 14. International Glaciological Society. (Extended Abstract)

MacKay, R.M. and M.A.K. Khalil, 1991. Theory and Development of a One Dimensional Time Dependent Radiative Convective Climate Model. Chemosphere, Vol. 22 #3-4 pp. 383-417.

Khalil, M.A.K.Khalil and R.M. MacKay, 1991. Global Warming by Trace Gases in a One Dimensional Climate Model. World Resource Review, Vol. 3 # 2.

MacKay, R.M. and M.A.K. Khalil, 8 July 1993. "Computer Based Activities for Understanding Global Change". Presented at the 1st International Conference on Computer-aided Learning and Distance Learning in meteorology, Hydrology, and Oceanography held at COMET in Boulder Colo.

MacKay, R.M. and M.A.K. Khalil, 13 August 1992. "Global Change Science and Education" Presented at the Aspen Global Change Institute, Aspen Colorado.

MacKay, R.M., 21 March 1992. "Global Warming: What we Know and What we Can Do" Presented in Salem Oregon at the Oregon state section of NATO spring 1992 meeting on the Rio Earth Summit. Sponsored by the America Association of University Women.