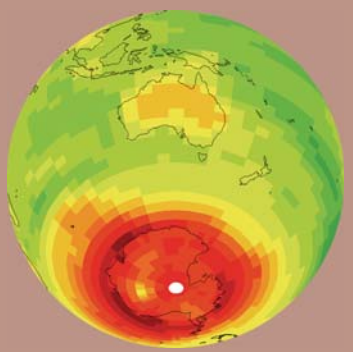
The CSIRO Mk3L Climate System Model

Steven J. Phipps

Antarctic Climate & Ecosystems CRC
University of Tasmania -
Institute of Antarctic & Southern Ocean Studies



Antarctic Climate & Ecosystems
Cooperative Research Centre

Technical Report No. 3

Steven J Phipps

Antarctic Climate & Ecosystems CRC
University of Tasmania
Private Box 80
Hobart, Tasmania 7001, Australia.
email: sjphipps@utas.edu.au

ISSN: 1833-2404

ISBN: 1-921197-03-X

October 2006

Published by The Antarctic Climate & Ecosystems Cooperative Research Centre, Hobart, Tasmania, Australia, 236 pp.

Established and supported under the Australian Government's Cooperative Research Centres Programme.

The CSIRO Mk3L Climate System Model

Contents

Acknowledgements

The CSIRO Mk3L climate system model was developed as part of the author's PhD project. I would like to acknowledge my supervisors, and particularly Jason Roberts, for their support throughout the model's lengthy gestation.

Thanks should go to CSIRO Marine and Atmospheric Research for their assistance, and particularly for allowing me to use their climate system model. Mk3L was developed and tested on facilities at the Australian Partnership for Advanced Computing in Canberra. Access to these facilities was invaluable in developing the model, and I would like to thank the staff at APAC for their support.

The development of Mk3L was supported through grants of computer time from the APAC Merit Allocation Committee (project e56) and the Tasmanian Partnership for Advanced Computing (project e00). Financial support was also received from the Australian Government (an International Postgraduate Research Scholarship), the University of Tasmania (a Tasmania Research Scholarship), the Antarctic CRC (an Antarctic CRC Top-Up Scholarship) and the Trans-Antarctic Association (grant TAA/99/12).

I would like to acknowledge use of the Ferret program (www.ferret.noaa.gov) in the production of the graphics presented herein, and the NOAA-CIRES Climate Diagnostics Center (www.cdc.noaa.gov) for making a range of datasets freely available.

I would also like to thank Jason Roberts and Vicki Randell for their diligence in proof-reading this document.

List of Acronyms

ADCP	Acoustic Doppler Current Profiler
AGCM	Atmospheric General Circulation Model
AMIP	Atmospheric Model Intercomparison Project
APAC	Australian Partnership for Advanced Computing
ASCII	American Standard Code for Information Interchange
BP	Before Present
CFL	Courant-Friedrichs-Lewy
CIRES	Cooperative Institute for Research in Environmental Sciences
CPU	Central Processing Unit
CSIRO	Commonwealth Scientific and Industrial Research Organisation
CXML	Compaq Extended Math Library
DOE	Department of Energy
ERBE	Earth Radiation Budget Experiment
FFT	Fast Fourier Transform
FFTW	Fastest Fourier Transform in the West
GISS	Goddard Institute for Space Studies
HP	Hewlett-Packard
IDL	Interactive Data Language
I/O	Input/Output
LW	Longwave
MPI	Message Passing Interface
NASA	National Aeronautics and Space Administration
NCAR	National Center for Atmospheric Research
NCEP	National Centers for Environmental Prediction
netCDF	network Common Data Form
NOAA	National Oceanic & Atmospheric Administration
ODAM	Ocean Direct Access Manager
OGCM	Ocean(ic) General Circulation Model
OpenMP	Open Multi Processing
PBS	Portable Batch System
PMIP	Paleoclimate Modelling Intercomparison Project
RAM	Random Access Memory
RH	Relative Humidity
SGI	Silicon Graphics Inc.
SSS	Sea Surface Salinity
SST	Sea Surface Temperature
SW	Shortwave
UKMO	U.K. Meteorological Office

Chapter 1

Introduction

1.1 The CSIRO Mk3L climate system model

The CSIRO Mk3L climate system model is a computationally-efficient coupled atmosphere-sea ice-ocean general circulation model, suitable for studying climate variability and change on millennial timescales.

The atmospheric component of Mk3L comprises a spectral general circulation model, a sea ice model and a land surface model. A coarse horizontal resolution of R21 is employed, giving zonal and meridional resolutions of 5.625° and $\sim$3.18° respectively. A hybrid vertical coordinate is used, with 18 vertical levels. The model incorporates both a cumulus convection scheme and a prognostic stratiform cloud scheme. The radiation scheme treats longwave and shortwave radiation independently, and is able to calculate the cloud radiative forcings. Code has been incorporated to calculate the values of the Earth's orbital parameters, enabling them to be varied dynamically at runtime.

The sea ice model includes both ice dynamics and ice thermodynamics. The land surface model allows for 13 land surface and/or vegetation types and nine soil types, and incorporates prognostic soil and snow models. The vegetation types and land surface properties are pre-determined, however, and are therefore static.

The oceanic componemt of Mk3L is a coarse-resolution, z-coordinate general circulation model. The horizontal grid matches the Gaussian grid of the atmosphere model, and there are 21 vertical levels. The prognostic variables are potential temperature, salinity, and the zonal and meridional components of the horizontal velocity. The vertical velocity is diagnosed through the application of the continuity equation. The scheme of Gent and McWilliams (1990) is employed, in order to parameterise the adiabatic transport of tracers by mesoscale eddies.

The coupling between the atmosphere and ocean models within Mk3L rigorously conserves both heat and freshwater. Flux adjustments are applied within the coupled model, to ensure that the control climate of the model is stable on millennial timescales.

The source code has been designed to ensure that Mk3L is portable across a wide range of computer architectures, whilst also being computationally efficient. Dependence on external libraries is restricted to the netCDF and FFTW libraries, both of which are freely available and open source, while a high degree of shared-memory parallelism is achieved through the use of OpenMP directives. On an Intel Pentium 4 processor, Mk3L completes around 5 model years per day of walltime. Much greater throughput can be

achieved on parallel computing facilities.

The simulated present-day climate is both stable and realistic. The control climate of Mk3L, and the response to external forcing, is studied in detail by Phipps (2006).

1.2 Overview

This report constitutes both technical documentation and a user's guide, and has been written with both beginners and experienced users in mind.

Chapter 2 describes the model physics; beginners can happily skip this chapter. Chapter 3 explains how to compile and run Mk3L, while Chapter 4 explains how to configure the model via the control file. Chapter 5 describes the restart and auxiliary files that are required, while Chapter 6 describes the output files, including the processing of model output.

A number of appendices are also included, which provide further information regarding the development of Mk3L. Appendix A outlines the modifications which were made to the original model source code, both to enable compilation and execution of the model on a number of new computer platforms, and to optimise the runtime performance. Appendices B, C and D document the remaining modifications which were made to the atmosphere model, the ocean model and the coupled model respectively; these modifications largely relate to the model physics.

Appendix E describes the procedures which were followed to generate the default restart and auxiliary files which are supplied with Mk3L, while Appendix F provides sample control files and run scripts. Relevant source code is presented in Appendix G.

Chapter 2

Model description

2.1 Introduction

The CSIRO Mk3L climate system model comprises two components: an atmospheric general circulation model, which incorporates both a sea ice model and a land surface model, and an oceanic general circulation model. The atmospheric general circulation model is a low-resolution version of the atmospheric component of the CSIRO Mk3 coupled model (Gordon et al., 2002), while the oceanic general circulation model is the oceanic component of the CSIRO Mk2 coupled model (Gordon and O'Farrell, 1997).

This combination takes advantage of the rapid execution times of the Mk2 coupled model, which result from the low horizontal resolution, while also taking advantage of the enhanced physics of the Mk3 atmosphere model. Relative to the Mk2 atmosphere model, enhancements to the physics include:

- an increase in the vertical resolution from 9 to 18 levels
- the incorporation of a prognostic scheme for stratiform cloud
- the incorporation of a new cumulus convection scheme
- an enhanced land surface model

In developing Mk3L from these components, extensive modifications were made to the source code. These modifications were intended to make Mk3L as portable across computer architectures as possible, whilst also ensuring its computational efficiency. Details are provided in Appendix A.

Some modifications were also made to the model physics, to ensure that Mk3L is suitable for conducting multi-millennial climate simulations. The most significant enhancement has been to modify the coupling between the atmosphere and ocean models, such that it rigorously conserves heat and freshwater. Other modifications have added features to the model, and sought to improve the realism of the simulated climate. The modifications to the physics of the atmosphere and ocean models, and to the coupling between them, are documented in Appendices B, C and D respectively.

The Mk3L atmosphere, ocean and coupled models are described in Sections 2.2, 2.3 and 2.4 respectively. The derivation of the flux adjustments, which are applied within the coupled model, is described in Section 2.5.

2.2 Atmosphere model

The CSIRO Mk3 atmosphere model consists of three components: an atmospheric general circulation model, a multi-layer dynamic-thermodynamic sea ice model and a land surface model. As each of these components is documented in detail by Gordon et al. (2002), only a brief summary is provided here. This summary concentrates on those features which are unique to Mk3L.

The standard configuration of the Mk3 atmosphere model employs a spectral resolution of T63. However, a spectral resolution of R21 is also supported for research purposes, and it is this resolution which is used within Mk3L. The zonal and meridional resolutions are therefore 5.625° and $\sim$3.18° respectively.

2.2.1 Atmospheric general circulation model

The dynamical core of the atmosphere model is based upon the spectral method, and uses the flux form of the dynamical equations (Gordon, 1981). Physical parameterisations and non-linear dynamical flux terms are calculated on a latitude-longitude grid, with Fast Fourier Transforms used to transform fields between their spectral and gridded forms. Semi-Lagrangian transport is used to advect moisture (McGregor, 1993), and gravity wave drag is parameterised using the formulation of Chouinard et al. (1986).

A hybrid vertical coordinate is used, which is denoted as the η-coordinate. The Earth's surface forms the first coordinate surface, as in the σ-system, while the remaining coordinate surfaces gradually revert to isobaric levels with increasing altitude. The 18 vertical levels used in the Mk3L atmosphere model are listed in Table 2.1 (Gordon et al., 2002, Table 1).

The topography is derived by interpolating the $1^\circ \times 1^\circ$ dataset of Gates and Nelson (1975a) onto the model grid. Some modifications are then made, in order to avoid areas of significant negative elevation upon fitting to the (truncated) resolution of the spectral model (Gordon et al., 2002). The resulting topography is shown in Figure 2.1.

Time integration is via a semi-implicit leapfrog scheme, with a Robert-Asselin time filter (Robert, 1966) used to prevent decoupling of the time-integrated solutions at odd and even timsteps. The Mk3L atmosphere model uses a timestep of 20 minutes.

The radiation scheme treats solar (shortwave) and terrestrial (longwave) radiation independently. Full radiation calculations are conducted every two hours, allowing for both the annual and diurnal cycles. Clear-sky radiation calculations are also performed at each radiation timestep. This enables the cloud radiative forcings to be determined using Method II of Cess and Potter (1987), with the forcings being given by the differences between the radiative fluxes calculated with and without the effects of clouds.

The shortwave radiation scheme is based on the approach of Lacis and Hansen (1974), which divides the shortwave spectrum into 12 bands. Within each of these bands, the radiative properties are taken as being uniform. Ozone concentrations are taken from the AMIP II recommended dataset (Wang et al., 1995). Additional code has been inserted into Mk3L, enabling both the solar constant and the epoch to be specified within the model control file, with the Earth's orbital parameters being calculated at runtime (Section B.2.1).

The longwave radiation scheme uses the parameterisation developed by Fels and Schwarzkopf (Fels and Schwarzkopf, 1975, 1981; Schwarzkopf and Fels, 1985, 1991), which divides the longwave spectrum (wavelengths longer than 5 μm) into seven bands. Values for the CO_2 transmission coefficients must be provided via an auxiliary file (Section 5.2).

Level (k)	η	Approximate height (m)
18	0.0045	36355
17	0.0216	27360
16	0.0542	20600
15	0.1001	16550
14	0.1574	13650
13	0.2239	11360
12	0.2977	9440
11	0.3765	7780
10	0.4585	6335
9	0.5415	5070
8	0.6235	3970
7	0.7023	3025
6	0.7761	2215
5	0.8426	1535
4	0.8999	990
3	0.9458	575
2	0.9784	300
1	0.9955	165

Table 2.1: The hybrid vertical levels used within the Mk3L atmosphere model: the value of the η-coordinate, and the approximate height (m).

The cumulus convection scheme is based on the U.K. Meteorological Office scheme (Gregory and Rowntree, 1990), and generates both the amount and the liquid water content of convective clouds. This scheme is coupled to the prognostic cloud scheme of Rotstayn (1997, 1998, 2000), which calculates the amount of stratiform cloud, using the three prognostic variables of water vapour mixing ratio, cloud liquid water mixing ratio and cloud ice mixing ratio.

In the stand-alone atmosphere model, four types of surface gridpoint are employed: land, sea, mixed-layer ocean and sea ice. The temperatures of the sea gridpoints are determined by monthly-mean observed sea surface temperatures, which are provided via an auxiliary file (Section 5.2). Linear interpolaton in time is used to estimate values at each timestep, with no allowance for any diurnal variation. At high latitudes, sea gridpoints may be converted to mixed-layer ocean gridpoints, with self-computed temperatures; these can then evolve into sea ice gridpoints. This is discussed further in the following description of the sea ice model.

2.2.2 Sea ice model

The sea ice model includes both ice dynamics and ice thermodynamics, and is described by O'Farrell (1998). Internal resistance to deformation is parameterised using the cavitating fluid rheology of Flato and Hibler (1990, 1992). The thermodynamic component is based on the model of Semtner (1976), which splits the ice into three layers, one for snow and two for ice. Sea ice gridpoints are allowed to have fractional ice cover, representing the presence of leads and polynyas.

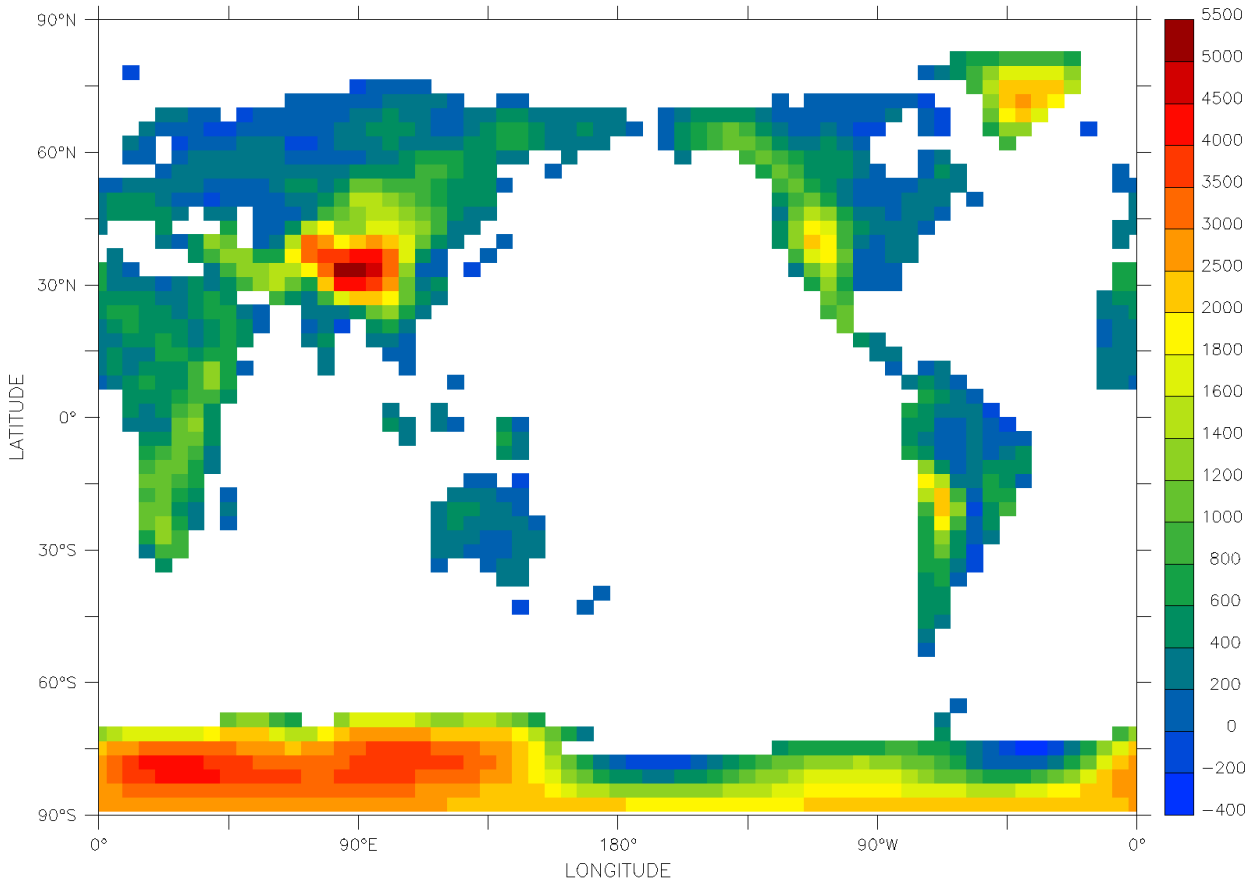


Figure 2.1: The topography of the Mk3L atmosphere model: the elevation of land gridpoints (m).

Ice advection arises from the forcing from above by atmospheric wind stresses, and from below by oceanic currents. The currents are obtained from the ocean model when running as part of the coupled model; in the stand-alone atmosphere model, climatological ocean currents are provided via an auxiliary file (Section 5.2).

The advance and retreat of the ice edge in the stand-alone atmosphere model is controlled by using a mixed-layer ocean to compute water temperatures for those sea gridpoints which lie adjacent to sea ice. The mixed-layer ocean has a fixed depth of 100 m, and the evolution of the water temperature T_s is calculated using the surface heat flux terms and a weak relaxation towards the prescribed sea surface temperature T_{SST}, as follows (Gordon et al., 2002, Equation 19.6):

$$\gamma_0 \frac{dT_s}{dt} = (1-\alpha_s)S_s^{\downarrow} + R_s^{\downarrow} - \epsilon_s \sigma T_s^4 - (H_s + E_s) + \lambda_c (T_{SST} - T_s) \tag{2.1}$$

Here, γ_0 represents the areal heat capacity of a 100 m-thick layer of water, t represents time, α_s represents the surface albedo, $S_s^{\downarrow}$ and $R_s^{\downarrow}$ represent the net downward surface fluxes of shortwave and longwave radiation respectively, ϵ_s represents the surface emissivity, σ represents the Stefan-Boltzmann constant, H_s and E_s represent the net upward surface fluxes of sensible and latent heat respectively, and λ_c represents a relaxation constant. Mk3L uses a relaxation timescale of 23 days.

A mixed-layer ocean gridpoint can become a sea ice gridpoint either when its temperature falls below the freezing point of seawater, which is taken as being -1.85°C, or when ice is advected from an adjacent sea ice gridpoint. When a mixed-layer ocean gridpoint is converted to a sea ice gridpoint, the initial ice concentration is set at 4%. The neighbouring equatorward gridpoint, if it is a sea gridpoint, is then converted

to a mixed-layer ocean gridpoint.

Within a gridpoint that has fractional sea ice cover, both in the stand-alone atmosphere model and the coupled model, the water temperature is calculated using a mixed-layer ocean with a fixed depth of 100 m. The surface heat flux is given by Equation 2.1, except that the final relaxation term is replaced with a basal heat flux F_i, which is calculated as follows (Gordon et al., 2002, Equation 19.8):

$$F_i = \frac{k_{frz}\rho_w c_w dz(T_{SST} - T_f)}{(dz/2)^2} + F_{geog} \tag{2.2}$$

Here, k_{frz} = 0.15×10^{-4} s^{-1} is the heat transfer coefficient, ρ_w and c_w are the density and specific heat capacity of seawater respectively, dz = 25 m is the thickness of the upper layer of the ocean model, and T_f = -1.85°C represents the freezing point of seawater. T_{SST} represents the prescribed sea surface temperature in the case of the stand-alone atmosphere model, and the temperature of the upper level of the ocean in the case of the coupled model. The additional fixed component F_{geog} allows for the effects of subgridscale mixing. Its value is resolution-dependent and, at the horizontal resolution of R21 used in Mk3L, is equal to 2 Wm^{-2} in the Northern Hemisphere, and 15 Wm^{-2} in the Southern Hemisphere.

If, within the coupled model, the temperature of the upper level of the ocean T_{OC} falls below -2°C, an additional term is added to the ice-ocean heat flux, as follows (Gordon et al., 2002, Equation 19.9):

$$F_{frz} = \frac{k_{frz}\rho_w c_w dz(T_{OC} - T_f)}{(dz/2)^2} \tag{2.3}$$

In this case, k_{frz} is increased to 6×10^{-4} s^{-1} in order to stimulate the formation of sea ice in sub-freezing waters.

Surface processes can lead to either a decrease in ice volume, as a result of either melting or sublimation, or an increase in ice volume; this can occur either when the depth of snow exceeds 2 m, in which case the excess is converted into an equivalent amount of ice, or when the weight of snow becomes so great that the floe becomes completely submerged. When the latter occurs, any submerged snow is converted into "white" ice.

Lateral and basal ice growth and melt are determined by the temperature of the mixed-layer ocean. Additional ice can grow when the water temperature falls below the freezing point of seawater, -1.85°C, subject to a maximum allowable thickness of 6 m. Once the water temperature rises above -1.5°C, half of any additional heating is used to melt ice; once it rises above -1.0°C, all of the additional heating is used to melt ice. In the case of the stand-alone atmosphere model, a sea ice gridpoint is converted back to a mixed-layer ocean gridpoint once the sea ice has disappeared. The neighbouring equatorward gridpoint, if it is a mixed-layer ocean gridpoint, is then converted back to a sea gridpoint.

2.2.3 Land surface model

The land surface model is an enhanced version of the soil-canopy scheme of Kowalczyk et al. (1991, 1994). A new parameterisation of soil moisture and temperature has been implemented, a greater number of soil and vegetation types are available, and a multi-layer snow cover scheme has been incorporated.

The soil-canopy scheme allows for 13 land surface and/or vegetation types and nine soil types. The land surface properties are pre-determined, and are provided via auxiliary files (Section 5.2). Seasonally-varying

values are provided for the albedo and roughness length, and annual-mean values for the vegetation fraction. The stomatal resistance is calculated by the model, as are seasonally-varying vegetation fractions for some vegetation types. The soil model has six layers, each of which has a pre-set thickness. Soil temperature and the liquid water and ice contents are calculated as prognostic variables. Run-off occurs once the surface layer becomes saturated, and is assumed to travel instantaneously to the ocean via the path of steepest descent.

The snow model computes the temperature, snow density and thickness of three snowpack layers, and calculates the snow albedo. The maximum snow depth is set at 4 m (equivalent to 0.4 m of water).

2.3 Ocean model

The CSIRO Mk2 ocean model is a coarse-resolution, z-coordinate general circulation model, based on the implementation by Cox (1984) of the primitive equation numerical model of Bryan (1969). It is described by Gordon and O'Farrell (1997) and Hirst et al. (2000) and, with some slight modifications to the physics, by Bi (2002).

The prognostic variables used by the model are potential temperature, salinity, and the zonal and meridional components of the horizontal velocity. The Arakawa B-grid (Arakawa and Lamb, 1977) is used, in which the tracer gridpoints are located at the centres of the gridboxes, and the horizontal velocity gridpoints are located at the corners. The vertical velocity is diagnosed through application of the continuity equation.

The horizontal grid matches the Gaussian grid of the atmosphere model, such that the tracer gridpoints on the ocean model grid coincide with the gridpoints on the atmosphere model grid. The zonal and meridional resolutions are therefore 5.625° and $\sim$3.18° respectively. There are 21 vertical levels, which are listed in Table 2.2.

The bottom topography is derived by interpolating the $1^\circ \times 1^\circ$ dataset of Gates and Nelson (1975b) onto the model grid, with some slight smoothing to ensure that a solution is achieved when calculating the barotropic streamfunction (Cox, 1984). The resulting bathymetry is shown in Figure 2.2.

A number of changes are made to the land/sea mask, relative to the atmosphere model. The land gridpoints at the tips of South America and the Antarctic Peninsula are replaced with ocean gridpoints, ensuring that Drake Passage accommodates three horizontal velocity gridpoints. In order to ensure adequate resolution of the Greenland-Scotland sill, Iceland is removed; likewise, adequate resolution of the flows through the Indonesian archipelago is ensured through a number of modifications to the land/sea mask.

Europe, Africa, Asia, and North and South America are combined into a single landmass through the closure of Bering Strait. Greenland, Madagascar and Japan are joined to this landmass through closure of the Canadian archipelago, the Mozambique Channel and the Sea of Japan respectively. Tasmania, continental Australia and New Guinea are also combined into a single island. Svalbard, which lies to the north of Norway and which occupies a single isolated gridpoint on the model grid, is removed.

As a result of these changes, there are just four landmasses on the ocean model grid:

- Europe/Africa/Asia/North America/South America/Greenland
- Antarctica
- Australia/New Guinea

Level (k)	Thickness (m)	Depth (m)		κ_e (m^2s^{-1})
		Centre	Base	
1	25	12.5	25	0*
2	25	37.5	50	70*
3	30	65	80	180*
4	37	98.5	117	290*
5	43	138.5	160	420*
6	50	185	210	580†
7	60	240	270	770†
8	80	310	350	1000‡
9	120	410	470	1000‡
10	150	545	620	1000‡
11	180	710	800	1000‡
12	210	905	1010	1000‡
13	240	1130	1250	1000‡
14	290	1395	1540	1000‡
15	360	1720	1900	1000‡
16	450	2125	2350	1000‡
17	450	2575	2800	1000‡
18	450	3025	3250	1000‡
19	450	3475	3700	1000‡
20	450	3925	4150	1000‡
21	450	4375	4600	1000‡

Table 2.2: The vertical levels used within the Mk3L ocean model: the thickness, the depth of the centre and base of each gridbox, and the value of the isopycnal thickness diffusivity. *These values are hard-coded into the model. †These values are the maximum allowable values, and are hard-coded into the model; lower values may be specified via the model control file. ‡These values are specified via the model control file, and represent the default values used in Mk3L.

- New Zealand

The only other modifications to the land/sea mask are to the Hudson and Gibraltar Straits, which are too narrow on the model grid to contain any horizontal velocity gridpoints; these straits are therefore closed.

The consolidation of the Earth's land surface into four landmasses improves the computational performance of the ocean model. This is a consequence of the *rigid-lid* boundary condition (Cox, 1984), which is employed in order to remove the timestep limitation associated with high-speed external gravity waves. Under this boundary condition, the external mode of momentum is represented by a volume transport streamfunction, ψ:

$$\overline{u} = -\frac{1}{aH}\frac{\partial\psi}{\partial\phi} \tag{2.4}$$

$$\overline{v} = \frac{1}{aH\cos\phi}\frac{\partial\psi}{\partial\lambda} \tag{2.5}$$

Here, $\overline{u}$ and $\overline{v}$ represent the zonal and meridional components, respectively, of the vertically-averaged

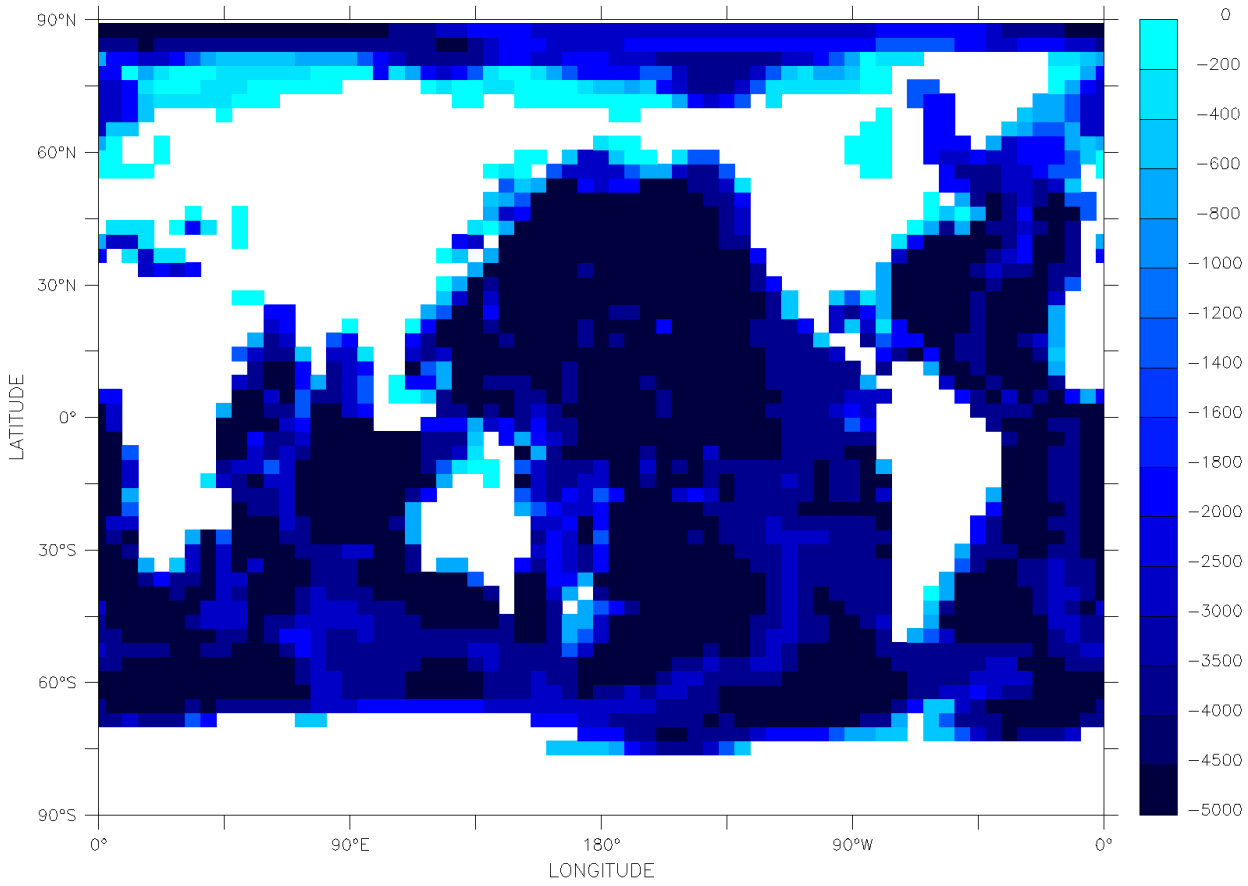


Figure 2.2: The bathymetry of the Mk3L ocean model: the depth of ocean gridpoints (m).

velocity, a represents the radius of the Earth, H represents the depth of the ocean, and ϕ and λ represent latitude and longitude respectively. At lateral walls, the boundary conditions on u and v are

$$u = v = 0 \tag{2.6}$$

The boundary condition on ψ at lateral walls is therefore

$$\frac{\partial \psi}{\partial \phi} = \frac{\partial \psi}{\partial \lambda} = 0 \tag{2.7}$$

This condition is satisfied by setting ψ constant along the boundary of each unconnected landmass. If islands are present, the constant value of ψ for each island indicates the net flow around that island, and hence must be predicted by the governing equations. The method used in the ocean model is *hole relaxation*, in which the line integral of the quantity $\nabla p_{surface}$, where $p_{surface}$ is the surface pressure, around each island is required to vanish.

By consolidating the Earth's land surface into four landmasses, and by setting the net flow around Europe-Africa-Asia-North America-South America-Greenland equal to zero, it is only necessary to calculate the line integrals around three relatively small islands (Antarctica, Australia-New Guinea and New Zealand).

The bathymetry of the Mk3L ocean model defines six basins which have no resolved connection with the world ocean: the Baltic, Black, Caspian and Mediterranean Seas, Hudson Bay and the Persian Gulf. It does not therefore adequately represent the physical connections which exist within the ocean; with the

exception of the Caspian Sea, each of these basins exchanges water with the world ocean via straits which are not resolved on the model grid. The effects of these exchanges are parameterised within the model through an imposed mixing between the gridpoints which lie to either side of each unresolved strait. This mixing has been substantially improved in Mk3L (Section C.2.1).

Time integration is via a leapfrog scheme, with mixing timesteps conducted once every 19 tracer timesteps in order to prevent decoupling of the time-integrated solutions at odd and even timesteps. Fourier filtering is used to reduce the timestep limitation arising from the CFL criterion (e.g. Washington and Parkinson, 1986) associated with the convergence of meridians at high latitudes, particularly in the Arctic Ocean (Cox, 1984). In the Mk3L ocean model, Fourier filtering is only applied northward of 79.6°N in the case of tracers, and northward of 81.2°N in the case of horizontal velocities (Section C.2.3). The ocean bottom is assumed to be insulating, while no-slip and insulating boundary conditions are applied at lateral boundaries.

The stand-alone ocean model employs an asynchronous timestepping scheme, with a timestep of 1 day used to integrate the tracer equations and a timestep of 20 minutes used to integrate the momentum equations. Within the coupled model - and during the final stage of spin-up runs, prior to coupling to the atmosphere model - a synchronous timestepping scheme is employed, with a timestep of 1 hour used to integrate both the tracer and momentum equations.

The vertical diffusivity κ_v varies as the inverse of the Brunt-Väisälä frequency, following the scheme of Gargett (1984). The minimum diffusivity is set at 3×10^{-5} m^2s^{-1}, except in the upper levels of the ocean, where it is increased in order to simulate the effects of mixing induced by surface winds. The minimum diffusivity between the upper two levels of the model is set at 2×10^{-3} m^2s^{-1}, while that between the second and third levels is set at 1.5×10^{-4} m^2s^{-1}. Whenever static instability arises, the vertical diffusivity is increased to 100 m^2s^{-1}, simulating convective mixing.

Two parameterisations are incorporated in order to represent the mixing along isopycnal surfaces (i.e. surfaces of constant density). The first of these parameterisations is the isopycnal diffusion scheme of Cox (1987), which allows for a more realistic representation of the tendency for tracers to be mixed along surfaces of constant density. In the default configuration of Mk3L, the isopycnal diffusivity is set to the depth-independent value of 1000 m^2s^{-1}.

The second parameterisation is the scheme of Gent and McWilliams (1990) and Gent et al. (1995), which parameterises the adiabatic transport of tracers by mesoscale eddies. An eddy-induced horizontal transport velocity is diagnosed, which is added to the resolved large-scale horizontal velocity to give an effective horizontal transport velocity. The continuity equation can be used to derive the vertical component of either the eddy-induced transport velocity or the effective transport velocity. In Mk3L, Gent-McWilliams eddy diffusion is employed at all latitudes; previously there was a transition to horizontal diffusion within the Arctic Ocean (Section C.2.4).

The default values for the isopycnal thickness diffusivity are shown in Table 2.2. Note that the values for levels 1 to 5 are fixed, and are hard-coded into the model. The values for levels 6 and 7 are maximum values, and these upper limits are also hard-coded into the model. Smaller values may be specified via the model control file, but the diffusivity may not exceed 580 and 770 m^2s^{-1} for levels 6 and 7 respectively. The values for the remaining levels are specified via the model control file. The decrease in the isopycnal thickness diffusivity in the upper layers, with a value of zero at the surface, is required by the continuity constraint imposed on the eddy-induced transport (Bi, 2002).

In the stand-alone ocean model, monthly values for the sea surface temperature (SST), sea surface salinity

(SSS), and the zonal and meridional components of the surface wind stress are read from auxiliary files (Section 5.2). Linear interpolation in time is used to estimate values at each timestep. The temperature and salinity of the upper layer of the model are relaxed towards the prescribed SST and SSS, using a default relaxation timescale of 20 days. In Mk3L, it is possible for a different relaxation timescale to be specified via the model control file (Section 4.3).

2.4 Coupled model

The coupling between the AGCM and OGCM is described in detail in Appendix D, and has been modified in Mk3L to ensure that heat and freshwater are rigorously conserved (Section D.4). Within the coupled model, four fields are passed from the atmosphere model (AGCM) to the ocean model (OGCM): the surface heat flux, surface salinity tendency, and the zonal and meridional components of the surface momentum flux. Four fields are also passed from the OGCM to the AGCM: the sea surface temperature (SST), sea surface salinity (SSS), and the zonal and meridional components of the surface velocity.

The Mk3L coupled model runs in a synchronous mode, with one OGCM timestep (1 hour) being followed by three AGCM timesteps (3×20 minutes). The surface fluxes calculated by the AGCM are averaged over the three consecutive AGCM timesteps, before being passed to the ocean model.

In the case of the surface fields passed from the OGCM to the AGCM, instantaneous values for the zonal and meridional components of the surface velocity are passed to the AGCM. These velocities act as the bottom boundary condition on the sea ice model for the following three AGCM timesteps. In the case of the SST and SSS, however, the OGCM passes two copies of each field: one containing the values at the current OGCM timestep, and one containing the values which have been predicted for the next OGCM timestep. The AGCM then uses linear interpolation in time to estimate the SST and SSS at each AGCM timestep.

Flux adjustments are applied to each of the fluxes passed from the AGCM to the OGCM, and also to the SST and SSS. The need to apply adjustments to the surface velocities is avoided by using climatological values, diagnosed from an OGCM spin-up run, to spin up the AGCM.

2.5 Flux adjustments

Four fields are passed from the atmosphere model (AGCM) to the ocean model (OGCM): the surface heat flux, the surface salinity tendency, and the zonal and meridional components of the surface momentum flux. Any differences between the surface fluxes calculated by the stand-alone AGCM, and those which are required to maintain the stand-alone OGCM in its equilibrium state, will represent a potential source of drift within the coupled model. Flux adjustments are therefore applied to each of these four fields.

The derivation of the flux adjustments is straightforward. If F_A is the surface flux diagnosed from an AGCM spin-up run, and F_O the surface flux diagnosed from an OGCM spin-up run (or, in the case of the components of the surface momentum flux, the flux applied to the stand-alone OGCM), then the flux adjustment ΔF is given by

$$\Delta F(\lambda, \phi, t) = F_A(\lambda, \phi, t) - F_O(\lambda, \phi, t) \tag{2.8}$$

where λ, ϕ, t represent longitude, latitude and the time of year respectively. The flux adjustments therefore vary temporally, as well as spatially. Within the coupled model, if F represents the surface flux calculated by the AGCM, then the adjusted flux F' which is passed to the OGCM is given by

$$F'(\lambda, \phi, t) = F(\lambda, \phi, t) - \Delta F(\lambda, \phi, t) \tag{2.9}$$

[Note that within Mk3L, the flux adjustments are *subtracted* from the AGCM surface fluxes.]

Four fields are also passed from the OGCM to the AGCM: the sea surface temperature (SST), the sea surface salinity (SSS), and the zonal and meridional components of the surface velocity. Any differences between the values of these fields, and the values which were imposed as the bottom boundary condition on the stand-alone AGCM, will also represent a potential source of drift within the coupled model. The need to apply adjustments to the components of the surface velocity is avoided through the use of climatological surface currents, diagnosed from an OGCM spin-up run, to spin up the AGCM. However, adjustments are applied to the SST and SSS.

The derivation of the adjustments to the SSS is straightforward. If S_{obs} is the SSS which was imposed as the surface boundary condition on the stand-alone OGCM, and S_O is the SSS which was simulated by the model, then the SSS adjustment ΔS is given by

$$\Delta S(\lambda, \phi, t) = S_{obs}(\lambda, \phi, t) - S_O(\lambda, \phi, t) \tag{2.10}$$

Within the coupled model, if S represents the SSS calculated by the OGCM, then the adjusted sea surface salinity S' which is passed to the AGCM is given by

$$S'(\lambda, \phi, t) = S(\lambda, \phi, t) + \Delta S(\lambda, \phi, t) \tag{2.11}$$

The derivation of the adjustments to the SST is more complex. If T_A is the SST which was imposed as the surface boundary condition on the stand-alone AGCM, and T_O is the SST simulated by the stand-alone OGCM, then the SST adjustment ΔT is given by

$$\Delta T(\lambda, \phi, t) = T_A(\lambda, \phi, t) - T_O(\lambda, \phi, t) \tag{2.12}$$

However, the stand-alone OGCM uses a mixed-layer ocean to calculate the SST at high latitudes. If T_{obs} is the SST which was imposed as the surface boundary condition on the stand-alone AGCM, and ΔT_{mlo} is the temperature of the mixed-layer ocean (expressed as an anomaly, relative to the value of T_{obs}), then T_A is given by

$$T_A(\lambda, \phi, t) = T_{obs}(\lambda, \phi, t) + \Delta T_{mlo}(\lambda, \phi, t) \tag{2.13}$$

Substituting this value for T_A into Equation 2.12, the SST adjustment is given by

$$\Delta T(\lambda, \phi, t) = T_{obs}(\lambda, \phi, t) + \Delta T_{mlo}(\lambda, \phi, t) - T_O(\lambda, \phi, t) \tag{2.14}$$

The SST adjustments are applied on the atmosphere model grid. Prior to using Equation 2.14 to calculate the SST adjustments, it is therefore necessary to interpolate the ocean model SST T_O onto the atmosphere model grid, in exactly the same fashion as takes place within the model.

Chapter 3

Compiling and running Mk3L

3.1 Introduction

This chapter describes how to compile and run the CSIRO Mk3L climate system model. Although there is an emphasis on the APAC National Facility (Australian Partnership for Advanced Computing, 2005), instructions are given which should enable the user to compile and run the model on any suitable facility.

Section 3.2 outlines the software which is required to compile Mk3L, while Sections 3.3 and 3.4 describe how to install and compile the model respectively. Section 3.5 describes how to test the model installation, while Section 3.6 outlines the procedure for running the model.

3.2 System requirements

The CSIRO Mk3L climate system model is designed to compile on any UNIX/Linux machine, without any modifications to the source code being required. However, the following software is required in order to compile the model:

- a Fortran/Fortran 90 compiler
- a C compiler
- the netCDF library (Unidata Program Center, 2005)
- version 2.x of the FFTW library (FFTW, 2005b)

The netCDF and FFTW libraries are both freely available and open source. Note that version 1.0 of Mk3L is not compatible with version 3.x of the FFTW library; this will be addressed in future versions.

Although not essential in order to compile the model, an auto-parallelising Fortran compiler may lead to enhanced performance on multiple processors (Section A.7).

3.3 Installation

Version 1.0 of the CSIRO Mk3L climate system model is distributed as a gzipped tar file, with the name `mk3l-1.0.tar.gz`. To install the model, extract the contents of this file using the command

```
tar zxvf mk3l-1.0.tar.gz
```

This creates a top-level directory `mk3l-1.0/`, containing the following subdirectories:

`core/`	contains the source code for Mk3L, and all the restart files, auxiliary files, control files and scripts which are needed to run the model
`doc/`	contains documentation
`post/`	contains utilities for the processing of model output
`pre/`	contains utilities for the generation of restart and auxiliary files

3.4 Compilation

3.4.1 At the APAC National Facility

The model source code is configured for compilation on `ac.apac.edu.au`, an SGI Altix Cluster located at the APAC National Facility in Canberra (Australian Partnership for Advanced Computing, 2005).

To compile the model on this facility, change to the directory `core/scripts/` and enter the command

```
./compile
```

This compiles not only Mk3L, but also the following utilities, which are used for runtime processing of ocean model output (Section 6.3):

```
annual_averages
annual_overturning
convert_averages
overturning
```

Upon compilation, the executables are installed in the directory `core/bin/`.

3.4.2 On other facilities

On other computing facilities, it will be necessary to configure the makefiles prior to compilation (a *makefile* contains a set of instructions, which describe how to compile a program). There are five makefiles, each of which will need to be configured:

```
core/src/annual_averages/Makefile
core/src/annual_overturning/Makefile
```

core/src/convert_averages/Makefile
core/src/model/Makefile
core/src/overturning/Makefile

The makefile for Mk3L (core/src/model/Makefile) is as follows:

```
# Purpose
# -------
# Makefile for the CSIRO Mk3L climate system model.
#
# Usage
# -----
# 'make' builds the model
# 'make clean' tidies up afterwards
#
# The Makefile must be configured for the machine on which the model is to be
# built. The values of the following variables must be set:
#
# FC            Fortran compiler
# F90           Fortran 90 compiler
# FFLAGS        Options to pass to the Fortran compiler
# F90FLAGS      Options to pass to the Fortran 90 compiler
#
# AUTO          Auto-parallelising compiler (if available)
# AUTOFLAGS     Options to pass to the auto-parallelising compiler
#
# CC            C compiler
# CFLAGS        Options to pass to the C compiler
#
# INC           The directory containing the netCDF header file
# LIB           The directories containing the netCDF and FFTW libraries
#               (note that Mk3L is only compatible with FFTW 2.x)
#
# History
# -------
# 2006 May 16   Steven J Phipps   Original version

##############################################################################
#                                                                            #
#                           START OF USER INTERFACE                          #
#                                                                            #
#          IN THIS SECTION, CONFIGURE THE MAKEFILE FOR THE MACHINE           #
#                   ON WHICH YOU WISH TO COMPILE THE MODEL                   #
#                                                                            #
##############################################################################

######################
#   ac.apac.edu.au   #
######################
```

```
FC = ifort
F90 = $(FC)
FFLAGS = -openmp -align dcommons -r8 -fpp -warn nouncalled
F90FLAGS = $(FFLAGS) -fixed

AUTO = $(FC)
AUTOFLAGS = $(FFLAGS)

CC = gcc
CFLAGS = -O

INC = -I/opt/netcdf-3.6.0-p1/intel-8.1/include
LIB = -L/opt/netcdf-3.6.0-p1/intel-8.1/lib -L/opt/fftw-2.1.5/intel-8.1/lib

######################
#   lc.apac.edu.au   #
######################

# FC = ifort
# F90 = $(FC)
# FFLAGS = -O3 -xW -align dcommons -r8 -fpp -warn nouncalled
# F90FLAGS = $(FFLAGS) -fixed

# AUTO = $(FC)
# AUTOFLAGS = $(FFLAGS)

# CC = gcc
# CFLAGS = -O

# INC = -I/opt/netcdf-3.5.1/Intel/include
# LIB = -L/opt/netcdf-3.5.1/Intel/lib -L/opt/fftw-2.1.5/Intel/lib

########################################
#   sc.apac.edu.au (serial execution)  #
########################################

# FC = f90
# F90 = $(FC)
# FFLAGS = -math_library fast -fp_reorder -arch host -tune host -align dcommons
-r8 -fpp -DALPH -warn nouncalled
# F90FLAGS = $(FFLAGS) -fixed

# AUTO = $(FC)
# AUTOFLAGS = $(FFLAGS)

# CC = cc
# CFLAGS = -O
```

```
# INC = -I/opt/netcdf-3.5.1/include
# LIB = -L/opt/netcdf-3.5.1/lib -L/opt/fftw-2.1.5/lib -lcxml

########################################
#   sc.apac.edu.au (parallel execution)  #
########################################

# FC = f90
# F90 = $(FC)
# FFLAGS = -omp -math_library fast -fp_reorder -arch host -tune host -align dcom
mons -r8 -fpp -DALPH -warn nouncalled
# F90FLAGS = $(FFLAGS) -fixed

# AUTO = kf90
# AUTOFLAGS = -fkapargs='-conc -ur=1 -so=2 -r=0 -fuse -fuselevel=1 -mc=2000' -no
_fp_reorder -r8 -v

# CC = cc
# CFLAGS = -O

# INC = -I/opt/netcdf-3.5.1/include
# LIB = -L/opt/netcdf-3.5.1/lib -L/opt/fftw-2.1.5/lib -lcxml

#############################################################################
#                                                                           #
#                            END OF USER INTERFACE                          #
#                                                                           #
#           YOU SHOULDN'T HAVE TO CHANGE ANYTHING BELOW THIS LINE           #
#                                                                           #
#############################################################################

####################
#   Object files   #
####################

OBJ =   hstring.o aplota.o  aploti.o  ateday.o  atemon.o  atstart.o  atstep.o \
        checkl.o  cldblk.o  clddia.o  cldset.o  clo89.o  cloud.o  cloud2.o  \
        cloudm.o  co2_read.o  collst.o  collstx.o  conv.o  cvmix.o c_aero.o \
        datard.o   dynmnl.o  dynmst.o  dynmvo.o  e1e288.o  e3v88.o errcheck.o \
        ecread.o  esbda.o  esibda.o  extend.o  filerd.o filest.o  \
        filewr.o  finterp.o  fst88.o  gauleg.o  gaussv.o  gwdrag.o  hadvect.o \
        hconst.o  hinterp.o  hist_acc.o  hist_cld.o  hist_save.o  hist_wlat.o \
        hkuo.o  hmread.o  hsflux.o  hvertmx.o  icefhx.o  icestat.o  icefall.o \
        icetau.o  inital.o  initax.o  initice.o  initfs.o  landrun.o lgndre.o \
        lwr88.o main.o matinv.o matset.o  mmtx.o  mtxmtx.o  ncinit.o  ncput.o \
        nestarc.o newcloud.o newrain.o o3blk.o o3_read.o o3set.o \
        ocicurr.o ocforce.o ocntau.o \
        openfl.o  openhist.o  ordleg.o   prdaily.o  prmlomap.o \
        progcld.o  prsmap.o  prtcd.o  prtcl.o  prtt.o  przav.o physgm.o  \
```

```
        radcoupl.o  radfs.o  radhyb.o  radin.o  radrhs.o radsgrg.o radslab.o  \
        radstres.o  radvars.o  rainda.o  readnml1.o  reset.o \
        seaice.o  setqcld.o  solargh.o  source.o  spa88.o specam.o  \
        surfdiag.o  surfset.o  surfa.o  surfupa.o  surfb.o  \
        surfupb.o   surfupl.o  swr89.o  table.o  timet.o  tmread.o \
        traceout.o  tracera.o  transf.o  trim.o  trim3x.o \
        xtchemie.o  xtemiss.o  xtsink.o  xtwetdep.o  \
        zenith.o   zerost.o  convukmo.o  ukall.o  ocsave1.o just_fm.o \
        rads_l.o radh_l.o comb_l.o orbpar.o conserve.o conserve_fw.o \
        ngwdrag.o bvnfcalc.o o3read_amip.o  o3set_amip.o \
        assel.o  diffn.o  energy.o  icecon.o  jmcgslt.o \
        linear.o uvharm.o uvreal.o  vadvect.o  vinterp.o zerogi.o \
        dtogcray.o  dynm.o  ptogcray.o  phys.o  mfftga.o  mfftma.o \
        ftospec.o  ftospec2.o semiis.o \
        advect.o    cavit.o     cavit2.o    dynice.o  \
        flatme.o    flatset.o   icebound.o  icediag.o   icedrive.o  icefree.o \
        icesetup.o  ltoh.o      ltou.o      polefilt.o  timefilt.o  wrapu.o \
        filter.o   findex.o   matrix.o   ocean.o    ocemon.o   \
        ocend.o    ocfinal.o  ocinit.o   odam.o     relax.o    \
        tracer1.o  ocdatra.o  ocdatro.o  r2fcfld.o  step0.o \
        clinic.o dencal.o state.o step.o tracer.o

model : $(OBJ)
        $(FC) $(FFLAGS) $(OBJ) $(LIB) -lrfftw -lfftw -lnetcdf -o model

clean :
        /bin/rm *.o

.f.o :
        $(FC) $(FFLAGS) -c $(INC) $<

.c.o :
        $(CC) $(CFLAGS) -c $<

ngwdrag.o : ngwdrag.f90
        $(F90) $(F90FLAGS) -c ngwdrag.f90

bvnfcalc.o : bvnfcalc.f90
        $(F90) $(F90FLAGS) -c bvnfcalc.f90

o3read_amip.o : o3read_amip.f90
        $(F90) $(F90FLAGS) -c o3read_amip.f90

o3set_amip.o : o3set_amip.f90
        $(F90) $(F90FLAGS) -c o3set_amip.f90

clinic.o : clinic.f
        $(AUTO) $(AUTOFLAGS) -c clinic.f
```

```
dencal.o : dencal.f
        $(AUTO) $(AUTOFLAGS) -c dencal.f

state.o : state.f
        $(AUTO) $(AUTOFLAGS) -c state.f

step.o : step.f
        $(AUTO) $(AUTOFLAGS) -c step.f

tracer.o : tracer.f
        $(AUTO) $(AUTOFLAGS) -c tracer.f
```

Although this file is lengthy, it is only the “User Interface” section at the start which needs to be modified. For the facility on which the model is to be compiled, it will be necessary to supply values for the following variables:

`FC`	The command which invokes the Fortran compiler
`F90`	The command which invokes the Fortran 90 compiler
`FFLAGS`	The options to pass to the Fortran compiler
`F90FLAGS`	The options to pass to the Fortran compiler
`AUTO`	The command which invokes the auto-parallelising compiler
`AUTOFLAGS`	The options to pass to the auto-parallelising compiler
`CC`	The command which invokes the C compiler
`CFLAGS`	The options to pass to the C compiler
`INC`	The directory containing the netCDF header file `netcdf.inc`
`LIB`	The directories containing the netCDF and FFTW libraries

Note that the makefile already contains the values which are needed for compilation on `lc.apac.edu.au`, the Linux Cluster located at the APAC National Facility. To use these values, delete the comment character (#) at the start of each line, and insert a comment character in front of each of the values for `ac.apac.edu.au`.

Although `sc.apac.edu.au` (an AlphaServer SC which used to be located at the APAC National Facility) no longer exists, the values required for compilation on this facility are still provided. They provide an example as to how to compile the model on this architecture and, in particular, as to how to make use of an auto-parallelising compiler.

The other four makefiles are simpler, but require modification in a similar fashion. Once this has been done, the model can be compiled by changing to the directory `core/scripts/` and entering the command

```
./compile
```

3.5 Testing the installation

Mk3L includes three scripts, located in the directory `core/scripts/`, which enable the user to test that the model has been compiled successfully. From within this directory, enter any of the following three commands:

`./test_atm`	Runs the atmosphere model for one day
`./test_cpl`	Runs the coupled model for one day
`./test_oce`	Runs the ocean model for one month

Each of these tests will typically take $\sim$1 minute. If the model runs correctly, it will write a succession of diagnostic information to standard output. In the case of `test_atm` and `test_cpl`, the test is successful if the last line of this output is:

```
Stopped after  1 days.
```

In the case of `test_oce`, the test is successful if the last line is:

```
 Timestep was    3600.00000000000
```

3.6 Running the model

3.6.1 The basics

The command which runs Mk3L is simply

```
./model < input
```

`model` is the executable, and `input` is the model control file (Chapter 4). The model writes various diagnostic information to standard output; this is usually redirected to an output file by running the model using a command such as

```
./model < input > output
```

In order to run successfully, the model requires that various restart and auxiliary files also be provided; these are described in Chapter 5.

3.6.2 Queueing systems

For short jobs, such as the test scripts above, the model can be run interactively. However, for production purposes, the model is typically run for several hours at a time. For longer jobs such as this, it will be necessary to use a queueing system when running on facilities such as the APAC National Facility.

Within the directory `core/scripts/` are three scripts which demonstrate how to run the model using PBS, which is the queueing system employed at APAC. From within this directory, enter any of the following three commands:

`qsub qsub_test_atm`	Runs the atmosphere model for one day
`qsub qsub_test_cpl`	Runs the coupled model for one day
`qsub qsub_test_oce`	Runs the ocean model for one month

The `qsub` command is part of PBS, and submits a job for execution under the control the queueing system. The command `qsub qsub_test_cpl` therefore requests that PBS run the script `qsub_test_cpl`.

The three scripts `qsub_test_atm`, `qsub_test_cpl` and `qsub_test_oce` are equivalent to `test_atm`, `test_cpl` and `test_oce` respectively, except that they use the queueing system on `ac.apac.edu.au`. On other facilities, these scripts may require modification before they can be submitted to the queue, and may require the use of a command other than `qsub`.

The script `qsub_test_cpl` is as follows, and contains directives (the lines beginning with #PBS) which pass information to the queueing system:

```
#!/bin/tcsh
#PBS -q express
#PBS -l walltime=0:05:00
#PBS -l vmem=200MB
#PBS -l ncpus=1
#PBS -l jobfs=1GB
#PBS -wd
#
# Purpose
# -------
# Runs the CSIRO Mk3L climate system model for one day, in coupled mode, using
# PBS.
#
# Usage
# -----
# qsub qsub_test_cpl
#
# History
# -------
# 2006 May 21   Steven J Phipps   Original version

# Create a temporary directory, if it doesn't already exist, into which to copy
# the model output at the end. Delete the contents if it does already exist.
set TMP_DIR = ~/mk3l_tmp/
if (-e $TMP_DIR) /bin/rm $TMP_DIR/*
if (! -e $TMP_DIR) mkdir $TMP_DIR

# Copy the model executable to the run directory
cp ../bin/model $PBS_JOBFS
```

```
# Copy the control file to the run directory
cp ../control/input_cpl_1day $PBS_JOBFS/input

# Copy the restart files to the run directory
cp ../data/atmosphere/restart/rest.start_default $PBS_JOBFS/rest.start
cp ../data/coupled/restart/oflux.start_default $PBS_JOBFS/fort.23
cp ../data/ocean/restart/orest.start_sync $PBS_JOBFS/fort.21

# Copy the basic data files to the run directory
cp ../data/atmosphere/basic/* $PBS_JOBFS
cp ../data/ocean/basic/* $PBS_JOBFS

# Copy the CO2 radiative data file for 280ppm to the run directory
cp ../data/atmosphere/co2/co2_data.280ppm.18l $PBS_JOBFS/co2_data.18l

# Copy the sea surface temperatures to the run directory (these are only used
# for diagnostic purposes)
cp ../data/atmosphere/sst/clim3f.sst_default $PBS_JOBFS/clim3f.sst

# Copy the flux adjustments to the run directory
cp ../data/coupled/flux_adjustments/hfcor.dat12_default $PBS_JOBFS/hfcor.dat12
cp ../data/coupled/flux_adjustments/sfcor.dat12_default $PBS_JOBFS/sfcor.dat12
cp ../data/coupled/flux_adjustments/txcor.dat12_default $PBS_JOBFS/txcor.dat12
cp ../data/coupled/flux_adjustments/tycor.dat12_default $PBS_JOBFS/tycor.dat12
cp ../data/coupled/flux_adjustments/t1coravth_default $PBS_JOBFS/t1coravth
cp ../data/coupled/flux_adjustments/s1coravth_default $PBS_JOBFS/s1coravth
cp ../data/coupled/flux_adjustments/dtm1av_default $PBS_JOBFS/dtm1av

# Change to the run directory
cd $PBS_JOBFS

# Run the model
./model < input > output

# Copy everything back to the temporary directory
/bin/cp * $TMP_DIR
```

The directives have the following meanings:

```
#PBS -q express
```

Instructs the queueing system to place the job in the `express` queue. This is more expensive than the `normal` queue, but is useful for ensuring that short jobs run promptly.

```
#PBS -l walltime=0:05:00
```

Sets a walltime limit of 5 minutes for the job.

```
#PBS -l vmem=200MB
```

Sets a memory limit of 200 MB for the job.

```
#PBS -l ncpus=1
```

Specifies that only one processor is required.

```
#PBS -l jobfs=1GB
```

Requests that temporary disk space of 1 GB be created. This is created in a directory with the name `$PBS_JOBFS`.

```
#PBS -wd
```

Starts the job in the directory from which it was submitted.

Although these scripts only run the model for a very short time, they demonstrate the essential steps that must always be taken in order to run the model:

1. Create a run directory.
2. Copy the executable, control file, restart file and auxiliary files to this directory.
3. Run the model.
4. Move the output to a more permanent directory.

3.6.3 Advanced

Scripts which have been used to run the model for production purposes are provided in Section F.3. They perform the same steps as the simple scripts above, but also perform processing and archiving of model output. The processing of model output is described in Chapter 6.

Chapter 4

The control file

4.1 Introduction

The control file configures the model for a particular simulation, determining the mode in which the model is to run, the duration of the simulation, the physical configuration of the atmosphere and ocean models, and which model variables are to be saved to file. For most purposes, the user will be able to configure the model via the control file, rather than having to make any changes to the source code.

Section 4.2 describes the parameters which are read from the control file by the atmosphere model, while Section 4.3 describes the parameters which are read by the ocean model. Sample control files, which have been used to run the model for production purposes, are provided in Section F.2.

This chapter represents an updated and expanded version of the information contained within Elliott et al. (1997).

4.2 Atmosphere model

The atmosphere model reads five `namelist` groups from the control file:

`control`	controls the basic features of each simulation
`diagnostics`	controls the diagnostic output
`statvars`	specifies which monthly-mean statistics are to be saved to file
`histvars`	specifies which daily statistics are to be saved to file
`params`	obsolete

Although `params` is obsolete, it must still be present within the control file; it can, however, be empty. The remaining `namelist` groups are now described in detail.

4.2.1 control

Overview

The parameters contained within control determine the basic features of each model simulation. Detailed descriptions of each of these parameters are provided below; however, there are only a few parameters that the user is likely to want to change:

locean, lcouple

These parameters determine the mode in which the model is to run:

locean=.true.	Stand-alone ocean mode (this overrides lcouple)
locean=.false./lcouple=.false.	Stand-alone atmosphere mode
locean=.false./lcouple=.true.	Coupled mode

nsstop, ndstop, lastmonth, months

For the coupled model and stand-alone atmosphere model, these parameters determine the duration of the simulation:

nsstop	Stop after nsstop timesteps
ndstop	Stop after ndstop days
lastmonth	Stop at the end of calendar month lastmonth (1=January, 2=February, ..., 12=December)
months	Stop after months months

The model examines each of these parameters in turn, in the above order (i.e. nsstop first, and months last). The first parameter to have a non-zero value is the one that takes effect. The following points should be noted, however:

- The coupled model and stand-alone atmosphere model cannot be run for more than 12 months at a time.
- For runs longer than 31 days, lastmonth or months must be used to specify the duration of the run, and *not* nsstop or ndstop.
- Irrespective of the required duration of the run, at least one of lastmonth and months must be greater than zero. The model will not run otherwise; this is a bug which will be fixed in future versions of Mk3L.

For the stand-alone ocean model, the duration of the simulation is specified by the values of iocmn and iocyr (Section 4.3).

`bpyear, csolar`

These parameters control the insolation. The value of `bpyear` specifies the epoch, in years before present (where the "present" is the year AD 1950); the value of `csolar` specifies the solar constant in Wm^{-2}.

`runtype`

This parameter, which consists of a string of three alphanumeric characters, specifies the name of the simulation. `runtype` is appended to the names of the output files generated by the model.

Detailed description

The parameters contained within `control` consist of a number of different data types. The `logical` parameters are listed in Table 4.1, while the `integer` and `real` parameters are listed in Table 4.2, and the `character` parameters in Table 4.3. As a result of modifications made to the model over time, some of the parameters are obsolete, and no longer have any effect. These parameters do not need to be present within the control file.

`logical` **parameters**

The `logical` parameters (Table 4.1) specify the configuration in which the model is to run, and which physical schemes are to be used within the atmosphere model. It is recommended that the following configuration be used (as Phipps, 2006):

- use of the prognostic cloud scheme [`dcflag` and `qcloud` set to `.true.`]
- use of the Semtner sea ice model, incorporating leads and sea ice dynamics [`semice`, `leads` and `idyn` set to `.true.`]
- use of the "New SIB" land surface scheme [`nsib` set to `.true.`]
- use of the UK Meteorological Office convection scheme [`ukmo` set to `.true.`]
- use of the NCAR boundary layer scheme [`ncarpbl` set to `.true.`]
- implicit treatment of the vorticity equation [`impvor` set to `.true.`]
- use of the hybrid vertical coordinate [`hybrid` set to `.true.`]
- use of AMIP 2 ozone data [`amipo3` set to `.true.`]
- application of flux adjustments within the coupled model [`fluxadj` set to `.true.`]

One `logical` parameter, `subice`, has been added in Mk3L. In order to allow for the effect of sub-gridscale mixing, an additional component is added to the sea ice-ocean heat flux (Section 2.2). The magnitude of this component is hard-coded into the model, and depends upon the horizontal resolution. Mk3L uses a horizontal resolution of R21, in which case a flux of 2 Wm^{-2} is applied in the Northern Hemisphere, and 15 Wm^{-2} in the Southern Hemisphere. The parameter `subice` enables the user to specify whether or not this additional heat flux should be applied; however, it is recommended that the default behaviour of the model be used.

Variable name	Description	Value	
		Default	Rec'd
amipo3	If true, use AMIP 2 ozone data	.true.	.true.
autoname	If true, compute name of output restart file automatically	.false.	.false.
coupled_aero	If true, use coupled aerosol model	.false.	.false.
dcflag	If true, compute cloud amounts (otherwise use observational dataset)	.true.	.true.
filewrflag	If true, write restart file at end of run	.false.	.true.
fluxadj	If true, and in coupled mode, apply flux adjustments	.true.	.true.
hybrid	If true, use hybrid vertical coordinate	.true.	.true.
idyn	If true, and semice and leads also true, use dynamical sea ice model	.true.	.true.
impvor	If true, use implicit treatment of vorticity equation	.true.	.true.
jsfdiff	If true, use Frederiksen spectral diffusion (only available at a horizontal resolution of T63)	.false.	.false.
lcouple[1]	If true, run in coupled mode, otherwise run in stand-alone AGCM mode (if locean true, lcouple has no effect)	.false.	.true./ .false.
leads	If true, and semice also true, allow leads in sea ice model.	.true.	.true.
locean[1]	If true, run in stand-alone OGCM mode	.false.	.true./ .false.
ncarpbl	If true, use NCAR boundary layer scheme	.false.	.true.
ncepagcm	If true, use NCEP data to initialise model	.false.	.false.
nestflag	If true, save boundary conditions for limited area model	.false.	.false.
newriver	If true, use new river routing scheme (only available at a horizontal resolution of T63)	.false.	.false.
nsib	If true, use "New SIB" land surface scheme	.true.	.true.
pi_emissions	If true, and coupled_aero also true, use pre-industrial emissions	.false.	.false.
qcloud	If true, and dcflag also true, use prognostic cloud scheme	.true.	.true.
qflux	If true, use a slab ocean, otherwise use observed SSTs (stand-alone AGCM only)	.false.	.false.
semice	If true, use Semtner sea ice model	.true.	.true.
subice*	If true, add additional fixed component to sea ice-ocean heat flux	.true.	.true.
ukconv	If true, use UKMO convection scheme	.true.	.true.

Table 4.1: The logical parameters contained within the namelist group control: variable names, brief descriptions, the default values and the recommended values. [1]The values of lcouple and locean determine the mode in which the model runs. The physical schemes are described by Gordon et al. (2002). *Added in Mk3L.

`integer` and `real` parameters

In addition to the `logical` parameters, the `integer` and `real` parameters (Table 4.2) further specify the configuration of the atmosphere model. The following configuration is recommended (as Phipps, 2006):

- a timestep of 20 minutes [`mstep` set to 20]
- the radiation scheme is called once every six timesteps [`nrad` set to 6]
- the model runs for one calendar year at a time [`lastmonth` set to 12]
- the solar constant is set equal to 1365 Wm^{-2} [`csolar` set to 1365.0]
- within the prognostic cloud scheme, the critical relative humidities for cloud formation are set equal to 75% over land, and 85% over the ocean [`rcritl` and `rcrits` set to 0.75 and 0.85 respectively]
- within the prognostic cloud scheme, the albedo reduction factors are set equal to 0.595 for convective cloud, and 0.865 for non-convective cloud [`refac1` and `refac2` set to 0.595 and 0.865 respectively]

Note that the default behaviour of the model is to call the radiation scheme once every two hours, and that `nrad`, if positive, must therefore be equal to 120/`mstep`; indeed, the model will abort execution if this is not the case. The interval between calls to the radiation scheme can be varied by setting `nrad` to a negative value; this specifies that the radiation scheme should be called once every `-nrad` timesteps. Note also that, when `mstep` is set to 0, the value of `nrad` is automatically set to the default value of 4, over-riding any value specified by the user.

The parameters `bpyear`, `csolar`, `rcritl`, `rcrits`, `refac1` and `refac2` have all been added to the namelist input in Mk3L. Previously, their values were hard-coded into the model. The parameters `bpyear` and `csolar` were added to the namelist input to enable the user to vary the epoch and the solar constant (Section B.2.1). The parameters `refac1` and `refac2` were added to the namelist input to enable their values to be “tuned”, in response to the excessive shortwave cloud forcing which was encountered during initial atmosphere model spin-up runs (Section B.2.2). The parameters `rcritl` and `rcrits` were added to the namelist input in anticipation of the fact that it might be desired to vary their values; however, it is recommended that the default values be used.

`character` parameters

The `character` parameters (Table 4.3) are largely used to specify the names of input and output files. The exception is `chtst`, which indicates whether the run should begin with a leapfrog timestep (which would be the case if using a restart file generated by a previous run) or a forward timestep (which would be the case if using a new restart file). It should therefore be replaced with a `logical` flag in future versions of the model.

Note that Mk3L does not include any code to calculate the CO_2 transmission coefficients; instead, these must be read from an auxiliary file. The model is supplied with a utility, `radint`, which can calculate the transmission coefficients (Section 5.5.1).

Variable name	Description	Value	
		Default	Rec'd
integer			
bpyear*	Epoch [years before present]	0	0
incd	Obsolete		
lastmonth	If >0, and nsstop=0 and ndstop=0, stop at end of calendar month lastmonth	0	12
months	If >0, and nsstop=0, ndstop=0 and lastmonth=0, stop after months months	1	0
mstep	Timestep [minutes]; if 0, default value is used	30	20
naerosol_d	Specifies type of direct sulphate aerosol forcing (0-3, 0 = no forcing)	2	0
naerosol_i	Specifies types of indirect sulphate aerosol forcing (0-3, 0 = no forcing)	0, 0	0, 0
ndstop	If >0 and ≤ 31, and nsstop=0, stop after ndstop days	0	0
nest_interval	If nestflag true, interval between saving boundary conditions [minutes]	480	480
nest_start	If nestflag true, interval before first saving boundary conditions [minutes]	240	240
ngwd	If $\neq 1$, use new gravity wave drag scheme	1	1
nrad	Number of timesteps between calls to radiation scheme; if 0, use no radiation; if <0, force a radiation timestep other than two hours; if mstep=0, default value is used	4	6
nsemilag	Obsolete		
nsstop	If >0, stop after nsstop timesteps; if <0, stop after zero timesteps and dump restart file; if -2, also reset day counter to zero	0	0
nthreads	Obsolete		
real			
csolar*	Solar constant [Wm^{-2}]	1366.77333333333	1365.0
rcritl*	In prognostic cloud scheme, critical RH for cloud formation over land	0.75	0.75
rcrits*	In prognostic cloud scheme, critical RH for cloud formation over the ocean	0.85	0.85
refac1*	In prognostic cloud scheme, albedo reduction factor (convective cloud)	0.85	0.595
refac2*	In prognostic cloud scheme, albedo reduction factor (other cloud)	0.95	0.865

Table 4.2: The integer and real parameters contained within the namelist group control: variable names, brief descriptions, the default values and the recommended values. *Added in Mk3L.

Variable name	Description	Value	
		Default	Rec'd
chtst	If OLD, begin with leapfrog timestep; if NEW begin with forward timestep	OLD	OLD
co2_datafile	Name of CO_2 data file	co2_data.hbg18	co2_data.18l
irfilename	Name of input restart file	(none)	rest.start
isfilename	Obsolete		
o3_datafile	Name of O_3 data file (only used if amipo3 false)	o3_data.hbg18	(none)
orfilename	Name of output restart file (only used if filewrflag true and autoname false)	(none)	rest.end
osfilename	Obsolete		
runtype[1]	Run name	(none)	*Run-dependent*
so4_direct_file	If naerosol_d equal to 1 or 2, name of aerosol data file (direct forcing)	(none)	(none)
so4_ind_rad_file	If naerosol_i(1) equal to 1 or 2, name of data file (indirect forcing - radiation)	(none)	(none)
so4_ind_rain_file	If naerosol_i(2) equal to 1 or 2, name of data file (indirect forcing - rainfall)	(none)	(none)

Table 4.3: The character parameters contained within the namelist group control: variable names, brief descriptions, the default values and the recommended values. [1] runtype specifies the run name, and should be different for each simulation.

4.2.2 diagnostics

Overview

The parameters contained within diagnostics control the diagnostic output of the model. This includes control over which statistics are to be written to standard output, and which model variables are to be saved to file. Detailed descriptions of each of these parameters are provided below; however, the most important parameters are the following:

statsflag, netcdf

statsflag must be set to true if the user wishes to save monthly-mean atmosphere model variables to file. The model then reads the parameters from the namelist group statvars (Section 4.2.3), which specifies which variables should be saved.

netcdf should also be set to true in order to specify that the output be written in netCDF (otherwise, the data is written in ASCII). One output file is generated for each variable; the filenames are of the form

`svvv_xxx.nc`, where `vvv` is the variable name, and `xxx` is the run name, as specified by the value of `runtype` (Section 4.2.1).

`savehist, hist_interval`

When `savehist` is set to true, model variables are saved to a netCDF file at a frequency determined by the value of `hist_interval`. The model reads the parameters from the `namelist` group `histvars` (Section 4.2.4), which specifies which variables should be saved to file.

The model has the capability of generating either one or two history files. For example, to save model variables once every six hours, the user would specify:

```
savehist=.true.
hist_interval=360
```

However, to save model variables once every six hours to one history file, and once every 24 hours to another, the user would specify:

```
savehist(1)=.true., savehist(2)=.true.
hist_interval(1)=360, hist_interval(2)=1440
```

Either one or two history files are generated each month. If only one history file is generated, then this has the filename `histyyyyymm.xxx.nc`, where `yyyyy` and `mm` are the year and month respectively, and `xxx` is the run name, as specified by the value of `runtype` (Section 4.2.1). If two history files are generated, then these have the filenames `histayyyyymm.xxx.nc` and `histbyyyyymm.xxx.nc`.

`savefcor`

`savefcor` should be set to true for atmosphere model spin-up runs, and false otherwise. This parameter specifies that the atmosphere model surface fluxes be saved to file, which enables flux adjustments to be diagnosed for use within the coupled model.

`glmean_interval`

Every `glmean_interval` minutes, a single line of global statistics is written to standard output; this information can be useful for monitoring the progress of a simulation. The variables displayed on this line are listed in Table 4.4.

Detailed description

As with `control`, the parameters contained within `diagnostics` also consist of different data types. The `logical` parameters are listed in Table 4.5, while the `integer` parameters are listed in Table 4.6.

Variable	Description	Units
DAY	Day number	days since 1 January 00000
HR	Time of day	hours since 00 UT
PMSL	Global-mean sea level pressure (not displayed on every line)	hPa
SBAL	Global-mean surface energy balance	Wm^{-2}
T(9)	Global-mean temperature at top of atmosphere	K
T*	Global-mean surface temperature	K
Tmn	Minimum surface temperature	K
Tmx	Maximum surface temperature	K
Kzn	Zonal kinetic energy per unit mass	Jkg^{-1}
Ked	Eddy kinetic energy per unit mass	Jkg^{-1}
LWG	Global-mean LW radiation at the surface	Wm^{-2}
LWt	Global-mean LW radiation at top of atmosphere	Wm^{-2}
SWg	Global-mean SW radiation at the surface	Wm^{-2}
SIt	Global-mean incoming SW radiation at top of atmosphere	Wm^{-2}
SOt	Global-mean outgoing SW radiation at top of atmosphere	Wm^{-2}
CLFR	Global-mean cloud radiative forcing	Wm^{-2}
AL	Planetary albedo	%
HC	Global-mean high cloud	%
MC	Global-mean medium cloud	%
LC	Global-mean low cloud	%
TC	Global-mean total cloud	%
EVP	Global-mean evaporation	mm/day
RAI	Global-mean precipitation	mm/day
HFLX	Global-mean surface sensible heat flux	Wm^{-2}
WATR	Global-mean precipitable water	mm
SNOD	Global-mean snow depth	cm
SICD	Global-mean sea ice thickness	m
TEMP	Mean atmospheric temperature	K
K/yr	Heating rate from the diffusion correction	K/year
Umx	Maximum wind speed	ms^{-1}

Table 4.4: The statistics written to standard output every `glmean_interval` minutes by the atmosphere model.

Variable name	Description	Value	
		Default	Rec'd
cdmap	If true, print map of surface drag coefficient	.false.	.false.
clforflag	If true, calculate cloud radiative forcing	.false.	.true.
debug	If true, save timeseries of data at the gridpoint specified by the values of insdebug, lgdebug and mgdebug (see Table 4.6) to the file debug.out	.false.	.false.
dynfp	If true, do dynamical field print	.false.	.false.
iener	If true, generate detailed energy diagnostics	.false.	.false.
ispec	If true, generate spectral amplitudes of fields	.false.	.false.
laust	Obsolete		
lbrmap	Obsolete		
ltrace	If true, employ Lagrangian advection of tracers	.false.	.false.
mlomap	If true, print map of temperature of sea ice and mixed-layer ocean gridpoints	.false.	.false.
netcdf	If true and statsflag also true, use netCDF for monthly-mean output; use ASCII otherwise	.false.	.true.
rainflag	If true, save daily rainfall to the file rainyyyyymm.xxx	.false.	.false.
savefcor[1]	If true, save surface fluxes to the file fcoryyyyymm.xxx	.false.	.true./ .false.
savegbrf	If true, save global means to the file gbrfyyyyymm.xxx	.false.	.false.
saveglmean	If true, save global means to the file glmnyyyyymm.xxx	.false.	.false.
savehist	If true, save model variables to file every hist_interval minutes	2* .false.	2* .false.
saveicuv	Obsolete		
saveqflux	If true, save the "qflux" fields to the file qflxyyyyymm.xxx	.false.	.false.
savezflux	If true, save zonal means of meridional fluxes to the file zflxyyyyymm.xxx	.false.	.false.
sdiagflag	If true, save timeseries of data at the nsdiag gridpoints specified by the values of insdiag, lgdiag and mgdiag (see Table 4.6) to the file sptsyyyyymm.xxx	.false.	.false.
sltrace	If true, employ semi-Lagrangrian advection of tracers	.false.	.false.
statsflag	If true, save monthly-mean model variables to file	.false.	.true.
tempflag	If true save daily maximum and minimum surface and screen temperatures to the files tgmx-, tgmn-, tsmx- and tsmnyyyyymm.xxx	.false.	.false.
uvtflag	If true, save daily zonal wind, meridional wind and temperature to the files uday-, vday- and tdayyyyyymm.xxx	.false.	.false.
gmap1	If false, set the following four flags to false	.false.	.false.
cldm	If true, print maps of cloud	.false.	
cvrnm	If true, print maps of convection and rainfall	.false.	
gwicm	If true, print maps of snow depth and ice depth	.false.	
rhnmm	If true, print maps of relative humidity and mixing ratio	.false.	

Variable name	Description	Value	
		Default	Rec'd
gmap2	If false, set the following four flags to false	.false.	.false.
evapm	If true, print map of average evaporation	.false.	
pmslm	If true, save sea level pressure to the file pmslyyyyymm.xxx	.false.	
rainm	If true, print map of average rainfall	.false.	
surfm	If true, print maps of average surface heat fluxes	.false.	
zavgp	If false, set the following sixteen flags to false	.true.	.true.
conzp	If true, print map of zonal-mean cloud fraction	.true.	.true.
dynzp	If true, print zonal-mean dynamical fields and heating rates	.true.	.true.
phz1p	If true, print zonal-mean moisture and heat flux statistics	.true.	.true.
phz2p	If true, print zonal-mean cloud, heat flux, temperature and moisture statistics	.true.	.true.
plotclds	If true, plot zonal-, monthly-mean cloud amounts	.false.	.true.
plotevrn	If true, plot zonal-, monthly-mean evaporation and rain	.false.	.true.
plotheat	If true, plot zonal-, monthly-mean energy balances	.false.	.true.
plotnetr	If true, plot zonal-, monthly-mean top of atmosphere radiation balance	.false.	.true.
savezcfr	If true, save zonal-mean cloud fraction to the file zcfryyyyymm.xxx	.false.	.false.
savezonr	If true, save zonal-mean relative humidity to the file zonryyyyymm.xxx	.false.	.false.
savezont	If true, save zonal-mean temperature to the file zontyyyyymm.xxx	.false.	.false.
savezonu	If true, save zonal-mean zonal wind to the file zonuyyyyymm.xxx	.false.	.false.
savezonv	If true, save zonal-mean meridional wind to the file zonvyyyyymm.xxx	.false.	.false.
savezonw	If true, save zonal-mean "sigma dot" to the file zonwyyyyymm.xxx	.false.	.false.
savezqcl	If true, and qcloud also true, save zonal-mean prognostic cloud statistics to the files zqcl-, zqcf-, zqev-, zqsb-, zaut-, zcol- and zacryyyyymm.xxx	.false.	.false.
savezqlp	Obsolete		

Table 4.5: The logical parameters contained within the namelist group diagnostics: variable names, brief descriptions, the default values and the recommended values. In the names of the output files, yyyy and mm represent the year and month respectively, and xxx represents the run name, as specified by the value of runtype (Section 4.2.1). [1]savefcor should be set to true for atmosphere model spin-up runs, and false otherwise.

`logical` **parameters**

The `logical` parameters (Table 4.5) largely control which statistics are written to standard output, and which model variables are saved to file. They should generally be set to `false`, in order to prevent large amounts of unnecessary diagnostic information from being generated.

Note that the parameters `gmap1`, `gmap2` and `zavgp` are "multi-control" flags. When set to false, they override the values of each of the parameters which follow them in Table 4.5, effectively setting them all to false as well. When `gmap1`, `gmap2` and `zavgp` are set to true, however, the values of each of these parameters are examined in turn.

`integer` **parameters**

The `integer` parameters (Table 4.6) largely control the frequency at which statistics are generated.

4.2.3 `statvars`

`statvars` contains logical flags which determine which monthly-mean model variables are to be saved to file; they only have effect if `statsflag` in `diagnostics` is set to true. Table 4.7 lists the flags which are contained within `statvars`, and the monthly-mean variables which will be saved to file if each flag is set to true.

Generally, each flag controls one variable. However, there are some exceptions, particularly the variables which are saved on vertical levels. If `c_sflg` is set to true, for example, then 18 output files will be created, containing the variables `c01`, `c02`, ..., `c18`. These variables contain the cloud amounts on each model level.

Note that some variables can only be saved when the corresponding physical schemes are being used: the "New SIB" land surface scheme (`nsib`), the prognostic cloud scheme (`qcloud`), and the Semtner sea ice model, incorporating leads and sea ice dynamics (`semice`, `leads` and `idyn`).

Note also that Table 4.7 excludes the flags which relate to the new river routing scheme (`newriver`) and the coupled aerosol model (`coupled_aero`), as neither of these are supported in Mk3L.

4.2.4 `histvars`

`histvars` contains logical flags which determine which model variables are to be saved to the history file(s); they only have effect if `savehist` in `diagnostics` is set to true. Table 4.8 lists the flags which are contained within `histvars`, and the variables which will be saved if each flag is set to true. Note that for some of the variables, instantaneous values are saved, whereas for the others, the values are accumulated over the history period (with the value saved being either the total, average, maximum or minimum value, as appropriate).

As with `statvars`, each flag generally controls one variable. However, the variables which are saved on vertical levels are again an exception, with each of these flags controlling 18 variables.

There are some important differences between `statvars` and `histvars`. Some variables are avilable within `statvars`, but not within `histvars`, and vice versa. `histvars` also contains the flag `all_hflg`,

Variable name	Description	Value	
		Default	Rec'd
glmean_interval	Interval between prints of global means to standard output [minutes]	480	240
hist_interval	If savehist true, interval between the saving of model variables to file [minutes]	2*1440	
idayp	Interval between calls to printing routines; if zero, print at end of run only [days]	0	0
insdebug	See debug (Table 4.5)	1	
insdiag	See sdiagflag (Table 4.5)	15*integer	
lgdebug	See debug (Table 4.5)	1	
lgdiag	See sdiagflag (Table 4.5)	15*integer	
mgdebug	See debug (Table 4.5)	1	
mgdiag	See sdiagflag (Table 4.5)	15*integer	
minw	If dynfp true, interval between dynamical field prints [minutes]	480	
nsdiag	See sdiagflag (Table 4.5)	11	

Table 4.6: The integer parameters contained within the namelist group diagnostics: variable names, brief descriptions, the default values and the recommended values.

Flag	Model variable	Description	Units
Surface statistics			
dtm_sflg	dtm	Mixed-layer ocean temperature anomaly	K
evp_sflg	evp	Evaporation	mm/day
hfl_sflg	hfl	Surface sensible heat flux	Wm^{-2}
int_sflg[1]	int	Interception of rainfall by foliage	mm/day
per_sflg[1]	per	Soil percolation	mm/day
pev_sflg[1]	pev	Potential evaporation	mm/day
psl_sflg	psl	Sea-level pressure	hPa
rnc_sflg	rnc	Convective precipitation	mm/day
rnd_sflg	rnd	Precipitation	mm/day
run_sflg[1]	run	Runoff	mm/day
sev_sflg[1]	sev	Scaling evaporation	mm/day
tax_sflg	tax	Zonal surface wind stress	Nm^{-2}
tay_sflg	tay	Meridional surface wind stress	Nm^{-2}
tb2_sflg	tb2	Soil temperature at level 2	K
tb3_sflg	tb3	Soil temperature at lowest level	K
tgf_sflg[1]	tgf	Vegetated ground temperature	K
tgg_sflg[1]	tgg	Bare ground temperature	K
thd_sflg	thd	Daily maximum screen temperature	K
thf_sflg[1]	thf	Daily maximum vegetated ground temperature	K

Flag	Model variable	Description	Units
thg_sflg[1]	thg	Daily maximum bare ground temperature	K
thm_sflg	thm	Extreme maximum screen temperature	K
tld_sflg	tld	Daily minimum screen temperature	K
tlf_sflg[1]	tlf	Daily minimum vegetated ground temperature	K
tlg_sflg[1]	tlg	Daily minimum bare ground temperature	K
tlm_sflg	tlm	Extreme minimum screen temperature	K
tsc_sflg	tsc	Screen temperature	K
tsu_sflg	tsu	Surface temperature	K
vmo_sflg	vmo	Surface wind speed	ms^{-1}
wfb_sflg	wfb	Upper level soil moisture	fraction
wfg_sflg	pmc	Moisture puddles	mm
	wfg	Lower level soil moisture	fraction
Cloud statistics			
clc_sflg[2]	clc	Convective cloud	fraction
cld_sflg	cld	Total cloud	fraction
clh_sflg	clh	High cloud	fraction
cll_sflg	cll	Low cloud	fraction
clm_sflg	clm	Medium cloud	fraction
lwp_sflg[2]	iwp	Ice water path	kgm^{-2}
	lwp	Liquid water path	kgm^{-2}
pwc_sflg[2]	pwc	Precipitable water content	mm
ref_sflg[2]	cli	Liquid cloud fraction	fraction
	ref	Effective radius for liquid clouds	μm
rev_sflg[2]	rev	Re-evaporation of rain	mm/day
sno_sflg[2]	sno	Snowfall	mm/day
ssb_sflg[2]	ssb	Sublimation of snow	mm/day
Radiation statistics			
als_sflg[1]	als	Surface albedo	fraction
rgc_sflg	rgc	Net LW radiation at ground (clear sky)	Wm^{-2}
rgd_sflg	rgd	Downward LW radiation at ground	Wm^{-2}
rgn_sflg	rgn	Net LW radiation at ground	Wm^{-2}
rtc_sflg	rtc	LW radiation out at top of atmosphere (clear sky)	Wm^{-2}
rtu_sflg	rtu	LW radiation out at top of atmosphere	Wm^{-2}
sgc_sflg	sgc	Net SW radiation at ground (clear sky)	Wm^{-2}
sgd_sflg	sgd	Downward SW radiation at ground	Wm^{-2}
sgn_sflg	sgn	Net SW radiation at ground	Wm^{-2}
soc_sflg	soc	SW radiation out at top of atmosphere (clear sky)	Wm^{-2}
sot_sflg	sot	SW radiation out at top of atmosphere	Wm^{-2}
Snow and ice statistics			
div_sflg[4]	wdf	Sea ice divergence removed by rheology	s^{-1}
	wls	Sea ice residual divergence	s^{-1}
gro_sflg[4]	gro	Monthly sea ice growth	m

Flag	Model variable	Description	Units
ich_sflg[3]	ich	Sea ice advection	m
ico_sflg[3]	icd	Sea ice thickness times concentration	m
	ico	Sea ice concentration	fraction
icu_sflg[4]	icu	Sea ice zonal velocity	ms^{-1}
icv_sflg[4]	icv	Sea ice meridional velocity	ms^{-1}
ire_sflg[4]	ire	Sea ice redistribution	m
isf_sflg[3]	isf	Sea ice-ocean salt flux	mm/day
itf_sflg[3]	itf	Sea ice-ocean heat flux	Wm^{-2}
sid_sflg	sid	Sea ice thickness	m
snd_sflg	snd	Snow depth	cm
Freshwater flux into ocean statistics			
fwf_sflg	fw1	Precipitation minus evaporation	mm/day
	fw2	Ice water A	mm/day
	fw3	Ice water B	mm/day
	fw4	River outflow	mm/day
Statistics on vertical levels			
c_sflg	c01–c18	Cloud amount on hybrid vertical levels	fraction
g_sflg	g01–g18	Geopotential height on pressure levels	kg/kg
l_sflg	l01–l18	Latent heating on hybrid vertical levels	K/day
q_sflg	q01–q18	Specific humidity on pressure levels	kg/kg
rh_sflg	r01–r18	Relative humidity on pressure levels	fraction
t_sflg	t01–t18	Temperature on pressure levels	K
u_sflg	u01–u18	Zonal wind on pressure levels	ms^{-1}
v_sflg	v01–v18	Meridional wind on pressure levels	ms^{-1}

Table 4.7: The flags contained within the `namelist` group `statvars`: the name of the flag, and the names, descriptions and units of the variable(s) which will be saved to file if each flag is set to true. [1]Only saved if `nsib` is true. [2]Only saved if `qcloud` is true. [3]Only saved if `semice` is true. [4] Only saved if `semice`, `leads` and `idyn` are all true.

Flag	Model variable	Description	Units	Inst. /Acc.
all_hflg	Save all the available variables			
Two-dimensional fields				
als_hflg	als	Surface albedo	fraction	I
cld_hflg	cld	Total cloud	fraction	I
clda_hflg	clda	Average total cloud	fraction	A
clh_hflg	clh	High cloud	fraction	I
cll_hflg	cll	Low cloud	fraction	I
clm_hflg	clm	Medium cloud	fraction	I
evp_hflg	evp	Evaporation	mm/day	A
hfl_hflg	hfl	Surface sensible heat flux	Wm^{-2}	A

Flag	Model variable	Description	Units	Inst. /Acc.
icb_hflg	icb	Bottom of low cloud	1–18	I
ich_hflg	ich	Level of high cloud	1–18	I
icm_hflg	icm	Level of medium cloud	1–18	I
ico_hflg	ico	Sea ice concentration	fraction	I
ict_hflg	ict	Top of low cloud	1–18	I
imsl_hflg	imsl	Surface type	1–4	I
int_hflg	int	Interception of rainfall by foliage	mm/day	A
pev_hflg	pev	Potential evaporation	mm/day	A
psf_hflg	psf	Surface pressure	mb	I
psl_hflg	psl	Sea-level pressure	mb	I
rgc_hflg	rgc	Net LW radiation at ground (clear sky)	Wm^{-2}	A
rgd_hflg	rgd	Downward LW radiation at ground	Wm^{-2}	A
rgn_hflg	rgn	Net LW radiation at ground	Wm^{-2}	A
rhsa_hflg	rhsa	Screen-level relative humidity	percent	A
rnc_hflg	rnc	Convective precipitation	mm/day	A
rnd_hflg	rnd	Precipitation	mm/day	A
rtc_hflg	rtc	LW radiation out at top of atmosphere (clear sky)	Wm^{-2}	A
rtu_hflg	rtu	LW radiation out at top of atmosphere	Wm^{-2}	A
run_hflg	run	Runoff	mm/day	A
sev_hflg	sev	Scaling evaporation	mm/day	A
sgc_hflg	sgc	Net SW radiation at ground (clear sky)	Wm^{-2}	A
sgd_hflg	sgd	Downward SW radiation at ground	Wm^{-2}	A
sgn_hflg	sgn	Net SW radiation at ground	Wm^{-2}	A
sid_hflg	sid	Sea ice thickness	m	I
sit_hflg	sit	Downward SW radiation at top of atmosphere	Wm^{-2}	A
snd_hflg	snd	Snow depth	cm	I
soc_hflg	soc	SW radiation out at top of atmosphere (clear sky)	Wm^{-2}	A
sot_hflg	sot	SW radiation out at top of atmosphere	Wm^{-2}	A
tax_hflg	tax	Zonal surface wind stress	Nm^{-2}	A
tay_hflg	tay	Meridional surface wind stress	Nm^{-2}	A
tb2_hflg	tb2	Soil temperature at level 2	K	I
tb3_hflg	tb3	Soil temperature at lowest level	K	I
tgf_hflg	tgf	Vegetated ground temperature	K	I
tgg_hflg	tgg	Bare ground temperature	K	I
tgh_hflg	tgh	Maximum surface temperature	K	A
tgl_hflg	tgl	Minimum surface temperature	K	A
tsc_hflg	tsc	Screen temperature	K	I
tsca_hflg	tsca	Average screen temperature	K	A
tsh_hflg	tsh	Maximum screen temperature	K	A
tsl_hflg	tsl	Minimum screen temperature	K	A
tsu_hflg	tsu	Surface temperature	K	I
tsua_hflg	tsua	Average surface temperature	K	A

Flag	Model variable	Description	Units	Inst. /Acc.
v10ma_hflg	v10ma	Average 10m wind speed	ms^{-1}	A
wfb_hflg	wfb	Upper level soil moisture	fraction	I
wfg_hflg	wfg	Lower level soil moisture	fraction	I
zht_hflg	zht	Surface height	m	Once[1]
Three-dimensional fields				
q_hflg	q01–q18	Specific humidity on hybrid vertical levels	kg/kg	I
t_hflg	t01–t18	Temperature on hybrid vertical levels	K	I
u_hflg	u01–u18	Zonal wind on hybrid vertical levels	ms^{-1}	I
v_hflg	v01–v18	Meridional wind on hybrid vertical levels	ms^{-1}	I

Table 4.8: The flags contained within the `namelist` group `histvars`: the name of the flag, and the names, descriptions and units of the variable(s) which will be saved to file if each flag is set to true. The final column indicates whether the values saved are instantaneous (I) or accumulated (A). [1]The surface height is only saved once to each history file.

which simply specifies that all the available variables be saved; an equivalent flag is not available within `statvars`. Another significant difference is that, in the case of `histvars`, the vertical fields are all saved on the hybrid vertical levels whereas, in the case of `statvars`, many of the vertical fields are saved on pressure levels.

Note also that, in the case of `histvars`, variables will be saved even if the corresponding physical scheme is not being used. When this occurs, the values which are saved will not be meaningful.

4.3 Ocean model

The ocean model reads ten `namelist` groups from the control file:

`contrl`	controls the basic features of the run
`eddy`	specifies various physical parameters
`eddy2`	obsolete in Mk3L
`etrans`	configures Gent-McWilliams eddy diffusion
`tsteps`	specifies the timesteps
`parms`	specifies various physical parameters
`icple`	specifies the duration of stand-alone ocean model runs
`pltg`	specifies whether monthly-mean statistics should be saved to file
`coefs`	specifies various physical parameters
`accel`	specifies acceleration factors

Each of these `namelist` groups is briefly discussed below.

It should be noted that the ocean model source code has been heavily modified relative to the original code of Cox (1984). Many additional parameters are now read from `namelist` input, while many of the original parameters are now obsolete. All ten `namelist` groups must be present within the control file, but those

parameters that are now obsolete can be omitted. There is considerable scope for simplification of the current source code.

4.3.1 Overview

The parameters contained within the ocean model `namelist` groups are largely used to determine the physical configuration of the model. Detailed descriptions of each of the parameters are provided below; however, there are only a few parameters that the user is likely to want to change:

`iocmn, iocyr`

When running in stand-alone ocean mode, the duration of the simulation is determined by the values of `iocmn` and `iocyr`. If `iocmn` is less than 12, the simulation stops after `iocmn` months; if `iocmn` is equal to 12, it stops after `iocyr` years.

`dttsf, dtuvf, dtsff`

`dttsf`, `dtuvf` and `dtsff` specify the tracer, velocity and streamfunction timesteps respectively. It is recommended that they all be set to 3600 seconds (1 hour); for the asynchronous stage of ocean model spin-up runs, however, it is recommended that `dttsf` be set to 86400 seconds (1 day), and that `dtuvf` and `dtsff` be set to 1200 seconds (20 minutes).

`ntsi, nnergy`

Every `ntsi` tracer timesteps, a single line of statistics is written to standard output. The variables displayed on this line are listed in Table 4.9. For some variables, more than one quantity is displayed; in the case of `DDMM`, the second of these quantities is always equal to 13. This reflects a bug within the diagnostic source code, which will be fixed in future versions of Mk3L.

Every `nnergy` tracer timesteps, a variety of energy, salinity and transport statistics is also written to standard output.

4.3.2 Basic features

The parameters contained within the groups `contrl`, `tsteps`, `icple` and `pltg` control the basic features of the run. These parameters are listed in Table 4.10.

`contrl`

The parameter `nmix` specifies the number of tracer timesteps to be conducted between each mixing timestep (Cox, 1984); the default value in Mk3L is 19. The parameters `ntsi` and `nnergy` specify the frequency at which diagnostic output should be written to standard output, while the parameter `na` specifies whether or not a restart file should be written at the end of each run.

Variable	Description	Units
ITT	Timestep counter	-
DDMM	Number of timesteps since boundary conditions last read	-
	13	-
EN	Total kinetic energy per unit volume	Jm^{-3}
MSCAN	Relaxation scan counter	-
QN	Surface heat flux at 118°W, 53°S (positive = down)	Wm^{-2}
T	Ocean temperature at 118°W, 53°S, level 1	°C
	Ocean temperature at 118°W, 53°S, level 17	°C
	Ocean temperature at 118°W, 53°S, level 21	°C
US	Zonal component of surface velocity at 118°W, 53°S	cms^{-1}
VS	Meridional component of surface velocity at 118°W, 53°S	cms^{-1}
TS28	Ocean temperature at 141°W, 5°S, level 1	°C

Table 4.9: The statistics written to standard output every ntsi tracer timesteps by the ocean model.

In the original version of the model, a restart file was written every nwrite tracer timesteps. This behaviour was considered unnecessary and has been removed from Mk3L, rendering the parameter nwrite obsolete.

tsteps

The parameters dttsf, dtuvf and dtsff specify the tracer, velocity and streamfunction timesteps respectively. It is recommended that all these timesteps be set to 3600 seconds (1 hour). For the asynchronous stage of ocean model spin-up runs, it is recommended that dttsf be set to 86400 seconds (1 day), and that dtuvf and dtsff be set to 1200 seconds (20 minutes).

icple

The parameters iocmn and iocyr specify the duration of stand-alone ocean model runs; the recommended values are 12 and 50 respectively. These parameters have no effect in the coupled model; in this case, the duration of the run is specified by the values of nsstop, ndstop, lastmonth and months (Section 4.2).

pltg

The only non-obsolete parameter within this group is iyes, which specifies whether or not monthly-mean statistics should be saved to file.

4.3.3 Physical configuration

The parameters contained within the groups eddy, eddy2, etrans, parms, coefs and accel determine the physical configuration of the ocean model. These parameters are listed in Table 4.11.

Variable name	Description	Value	
		Default	Rec'd
contrl			
na	If equal to 1, write restart file at end of run	1	1
nb	Obsolete		
nc	Obsolete		
ndw	Obsolete		
nfirst	Obsolete		
nlast	Obsolete		
nmix	Number of tracer timesteps between mixing timesteps	(none)	19
nnergy	Number of tracer timesteps between writes of energy diagnostics to standard output	(none)	3650^{A}/ $87600^{S,C}$
ntsi	Number of tracer timesteps between writes of run statistics to standard output (stand-alone ocean model only)	(none)	30^{A}/ $720^{S,C}$
nwrite	Obsolete in Mk3L		
tsteps			
dtsff	Streamfunction timestep [seconds]	(none)	1200.0^{A}/ $3600.0^{S,C}$
dttsf	Tracer timestep [seconds]	(none)	86400.0^{A}/ $3600.0^{S,C}$
dtuvf	Velocity timestep [seconds]	(none)	1200.0^{A}/ $3600.0^{S,C}$
icple			
iocmn	If < 12, stop after iocmn months (stand-alone ocean model only)	(none)	12
iocyr	If iocmn equal to 12, stop after iocyr years (stand-alone ocean model only)	(none)	50
isync	Obsolete		
nato	Obsolete		
pltg			
itdb	Obsolete		
iyes	If $\neq 0$, write monthly-mean statistics to file	1	1
ndiag	Obsolete		

Table 4.10: The parameters contained within the namelist groups contrl, tsteps, icple and pltg: variable names, brief descriptions, the default values and the recommended values. ^{A} Asynchronous ocean model. ^{S} Synchronous ocean model. ^{C} Coupled model.

Variable name	Description	Value	
		Default	Rec'd
eddy			
ahf	Horizontal diffusivity [cm^2s^{-1}]	(none)	7.0e6
ahi1f	Isopycnal tracer diffusivity at surface [cm^2s^{-1}]	(none)	1.0e7
ahi2f	Isopycnal tracer diffusivity at infinite depth [cm^2s^{-1}]	(none)	1.0e7
ahi3f	Scaling distance for varying diffusivity [cm]	(none)	5.0e4
amf	Horizontal viscosity [cm^2s^{-1}]	(none)	9.0e9
fkphf	Obsolete		
fkpmf	Vertical viscosity [cm^2s^{-1}]	(none)	20.0
slmxrf	Inverse maximum slope	(none)	100.0
eddy2			
ahh1f	Obsolete in Mk3L		
ahh2f	Obsolete in Mk3L		
ahh3f	Obsolete in Mk3L		
etrans			
ahe1f	Isopycnal thickness diffusivity at surface [cm^2s^{-1}]	(none)	1.0e7
ahe2f	Isopycnal thickness diffusivity at infinite depth [cm^2s^{-1}]	(none)	1.0e7
ahe3f	Scaling distance for varying diffusivity [cm]	(none)	5.0e4
igm	Obsolete in Mk3L		
itm	If equal to 1, use McDougall enhancement to Gent-McWilliams eddy diffusion	0	0
parms			
acorf	If > 0.0, treat Coriolis term implicitly, with forward component weighted by acorf	(none)	0.55
critf	Relaxation convergence criterion [cm^3s^{-1}]	(none)	1.0e8
mxscan	Maximum number of relaxation scans permitted	(none)	80
sorf	Over-relaxation coefficient	(none)	1.5
trelax*	Relaxation timescale [days] (stand-alone ocean model only)	20.0	20.0
coefs			
cdrag	Coefficient of bottom drag [dimensionless]	(none)	2.6e-3
difrat	Obsolete		
itset	If equal to 1, reset time counters to zero at start (stand-alone ocean model only)	1	1
urat	Obsolete in Mk3L		
accel			
dtxf	Acceleration factors	21*1.0	21*1.0

Table 4.11: The parameters contained within the namelist groups eddy, eddy2, etrans, parms, coefs and accel: variable names, brief descriptions, the default values and the recommended values. *New in Mk3L.

eddy

The parameters amf and fkpmf set the horizontal and vertical viscosities respectively. ahf sets the horizontal diffusivity, but is essentially obsolete; it is only used within the coupled model, and is only used to generate diagnostic output. fkphf, which specifies the vertical diffusivity in the original source code of Cox (1984) is also obsolete; the vertical diffusivity is now calculated as a function of the stability of the water column (Section 2.3).

The parameters ahi1f, ahi2f and ahi3f are used to specify a depth-dependent isopycnal tracer diffusivity, with the diffusivity ahi being calculated within the model as follows (zdzz specifies the depth of the centre of each gridbox, in centimetres):

$$\mathtt{AHI(K)} = \mathtt{AHI2F} + (\mathtt{AHI1F} - \mathtt{AHI2F}) * \mathtt{EXP}(-\mathtt{ZDZZ(K)}/\mathtt{AHI3F}) \tag{4.1}$$

It should be noted that the recommended values of ahi1f and ahi2f, which are both equal to 1×10^7, specify a depth-independent isopycnal tracer diffusivity of 1×10^7 cm^{2}s^{-1}.

slmxrf sets the inverse maximum slope. This parameter specifies the upper limit on the isopycnal slope to be used when calculating the mixing tensor, with values being capped at 1/slmxrf.

eddy2

The parameters ahh1f, ahh2f and ahh3f specify a depth-dependent horizontal diffusivity, in an analagous manner to ahi1f, ahi2f and ahi3f. However, horizontal diffusion, which was previously only employed within the Arctic Ocean, has been completely removed from Mk3L (Section C.2.4), rendering eddy2 obsolete.

etrans

The namelist group etrans contains parameters which control Gent-McWilliams eddy diffusion (Gent and McWilliams, 1990). In the original version of the model, the parameter igm was used to specify whether or not eddy diffusion should be applied. However, the use of Gent-McWilliams eddy diffusion has been hard-coded into Mk3L in order to enhance runtime performance (Section A.6.1), rendering igm obsolete.

The parameter itm is used to specify whether or not an enhancement to Gent-McWilliams eddy diffusion should be used. However, the enhancement has been found to have little effect in this implementation of the model (Hirst, pers. comm.), and so it is recommended that it not be used.

The parameters ahe1f, ahe2f and ahe3f specify a depth-dependent isopycnal thickness diffusivity, in an analagous manner to ahi1f, ahi2f and ahi3f (zdz specifying the depth of the base of each gridbox, in centimetres):

$$\mathtt{ahe(k)} = \mathtt{ahe2f} + (\mathtt{ahe1f} - \mathtt{ahe2f}) * \mathtt{exp}(-\mathtt{zdz(k-1)}/\mathtt{ahe3f}) \tag{4.2}$$

The recommended values of ahe1f and ahe2f, which are both equal to 1×10^7, specify a depth-independent isopycnal thickness diffusivity of 1×10^7 cm^{2}s^{-1}. However, it should be noted that, for the upper seven levels of the ocean, the diffusivities specified by ahe1f, ahe2f and ahe3f are over-ridden by

values which are hard-coded into the model. These values specify an isopycnal thickness diffusivity which decreases from 7.7×10^6 cm^{2}s^{-1}, at a depth of 270 m, to zero at the surface (Section 2.3).

`parms`

The `namelist` group `parms` specifies the values of a number of physical parameters; `acorf`, `mxscan`, `sorf` and `critf` are documented by Cox (1984).

The parameter `trelax` has been added in Mk3L. It enables the user to specify the relaxation timescale to be used by the stand-alone ocean model when relaxing the sea surface temperature and salinity towards observed values (Section C.2.5).

`coefs`

The only non-obsolete physical parameter specified within the `namelist` group `coefs` is `cdrag`, which sets the coefficient of bottom drag.

In the original version of the model, the parameter `urat` enabled the user to specify a globally-uniform factor by which to multiply the observed wind stresses in the case of the stand-alone ocean model. This ability has been removed from Mk3L, rendering `urat` obsolete.

`accel`

The `namelist` group `accel` specifies time acceleration factors for each level of the model. These factors can be set to values larger than 1, if the user wishes to use the method of Bryan (1984) to accelerate the convergence of the ocean towards equilibrium. However, it is recommended that they be set to 1.0.

Chapter 5

Input files

5.1 Introduction

This chapter describes the restart and auxiliary files which are required in order to run Mk3L. It also outlines how to generate new restart and auxiliary files, in order to configure the model for alternative scenarios.

Section 5.2 describes the restart and auxiliary files which are required by the model, while Section 5.3 documents the origin of the default restart and auxiliary files that are supplied with the model source code. Sections 5.4 and 5.5 describe the utilities that are also supplied with the model, for the generation of restart and auxiliary files respectively.

5.2 Overview

The restart and auxiliary files required by Mk3L depend not only upon the mode in which it is being run (i.e. stand-alone atmosphere model, stand-alone ocean model or coupled model), but also upon its physical configuration. Certain physical schemes require that the model read in datasets at runtime; for example, the “New SIB” land surface scheme (Gordon et al., 2002) reads data from a number of files.

It is beyond the scope of this document to describe all the restart and auxiliary files which might be required by the model, in all the possible configurations. This section, however, lists the files required by the model in its default configuration. Particular attention is given to the differences between Mk3L and the original version of the model.

Sections 5.2.1 and 5.2.2 describe the restart and auxiliary files required by the atmosphere and ocean models respectively, both in stand-alone mode and when running as part of the coupled model. Section 5.2.3 describes the additional files which are required by the coupled model.

Many of the restart and auxiliary files are in a machine-dependent binary format; ideally, the model should be modified such that all input and output files are in netCDF (Unidata Program Center, 2005), which is a completely machine-independent format. Not only would this allow restart and auxiliary files generated on one machine to be used on any other, but it would also simplify the analysis and visualisation of these datasets.

Filename	Format	Description	Required?	
			Atm	Cpl
(User-specified)	Binary	Restart file	Yes	Yes
(User-specified)	Text	CO_2 transmission coefficients	Yes	Yes
albnew21f	Text	Land surface albedos	Yes	Yes
amip2o3.dat	Text	AMIP 2 ozone volume mixing ratios	Yes	Yes
clim3f.sss*	Text	Observed sea surface salinities	Yes†	No
clim3f.sst	Text	Observed sea surface temperatures	Yes	Yes‡
icedivl.R21	Text	Sea ice divergence limiter mask	Yes	Yes
landrun21	Text	Run-off relocation data	Yes†	Yes
ocuv.3st*	Text	Observed ocean currents	Yes	No
psrk21f.dat	Text	Surface pressure data ($\sim$topography)	Yes	Yes
sibrs.dat	Text	Stomatal resistance (“New SIB” scheme)	Yes	Yes
sibsig.dat	Text	Vegetation fraction (“New SIB” scheme)	Yes	Yes
sibsoil.dat	Text	Soil type (“New SIB” scheme)	Yes	Yes
sibvegt.dat	Text	Vegetation/land type (“New SIB” scheme)	Yes	Yes
sibz0.dat	Text	Roughness length (“New SIB” scheme)	Yes	Yes

Table 5.1: The restart and auxiliary files required by the Mk3L atmosphere model: the filename, the format, a brief description, and whether it is required in stand-alone mode (Atm) and in coupled mode (Cpl). *New and/or modified in Mk3L. †Only required by the stand-alone atmosphere model if calculating surface fluxes for the purpose of diagnosing flux adjustments for the coupled model. ‡Only required for diagnostic purposes.

5.2.1 Atmosphere model

The names of the restart file, and of the file containing the CO_2 transmission coefficients, are specified by the parameters `irfilename` and `co2_datafile` respectively, which are contained within the `namelist` group `control` (Section 4.2). The names of the other auxiliary files are hard-coded into the model, and are shown in Table 5.1.

The auxiliary file `clim3f.sss` is new in Mk3L, and contains observed sea surface salinities. The sea surface salinity is required in order to convert surface freshwater fluxes to equivalent surface salinity tendencies (Section D.4). The stand-alone atmosphere model must be able to perform these calculations if flux adjustments are to be diagnosed for use within the coupled model; it must therefore be provided with observed sea surface salinities.

The auxiliary file `clim3f.sst` contains observed sea surface temperatures, which act as the bottom boundary condition on the stand-alone atmosphere model. This file is required by the atmosphere model even when it is running as part of the coupled model; however, the coupled model only uses the data for diagnostic purposes, using it to compare the ocean model sea surface temperatures (after the application of any flux adjustments) with observed values.

The auxiliary file `ocuv.3st` is also new in Mk3L, containing the ocean currents which act as the bottom boundary condition on the sea ice model. It replaces the 12 previous auxiliary files `ocuv01.3st`, `ocuv02.3st` ... `ocuv12.3st` (Section B.3.1).

The auxiliary files `hhm01`, `hhm02` ... `hhm12` and `c9details`, required by the original version of the model,

Filename	Format	Description	Required?	
			Oce	Cpl
`fort.21`	Binary	Restart file	Yes	Yes
`bsnmask.nc`*	netCDF	Ocean basin masks	Yes†	Yes†
`sss.dat`*	Text	Observed sea surface salinities	Yes	No
`sst.dat`*	Text	Observed sea surface temperatures	Yes	No
`stress.dat`*	Text	Observed wind stresses	Yes	No
`sttop.bot_ind`	Text	Bathymetry	Yes	Yes

Table 5.2: The restart and auxiliary files required by the Mk3L ocean model: the filename, the format, a brief description, and whether it is required in stand-alone mode (Oce) and in coupled mode (Cpl). *Modified and/or re-named in Mk3L. †The file `bsnmask.nc` is only required if it is desired to use the utility `overturning` to calculate the meridional overturning streamfunctions at runtime.

are not required by Mk3L (Section B.3.1).

All the auxiliary files read by the atmosphere model are in text format, and hence are fully portable. The restart file is in binary format, and is therefore machine-dependent. If a restart file generated on one machine is to be used to initialise the model on another, this may require that the file be converted from one binary format to another (Section 5.4).

5.2.2 Ocean model

The ocean model requires that the restart file have the name `fort.21`, although this filename is not specified within the source code. Instead, the model relies upon Fortran's implicit file naming whereby, if data is read from unit `nn`, and this unit has not been connected to an external file using the `open` statement, then data is read from the file `fort.nn`. It is dangerous to rely upon this behaviour, however, as it does not form part of the Fortran standard (Standards Association of Australia, 1983), and it will therefore be changed in future versions of Mk3L.

The names of the auxiliary files *are* hard-coded into the model. The auxiliary files `sss.dat`, `sst.dat` and `stress.dat`, which contain surface boundary conditions for the stand-alone ocean model, have been given more logical and consistent names in Mk3L (Section C.3.1).

The netCDF auxiliary file `bsnmask.nc`, which divides the world ocean into the Atlantic and Pacific Oceans, is new in Mk3L, replacing the binary file `bsnmask.cif`. It is not required by the ocean model itself; however, it is required by the ocean model utility `overturning` (Section C.4.2), and it must therefore be supplied if it is desired to calculate the meridional overturning streamfunctions at runtime.

The auxiliary file `cwice.dat12`, required by the original version of the model, is not required by Mk3L (Section C.3.1).

As with the atmosphere model, the ocean model restart file is in a machine-dependent binary format, while the auxiliary files are in a machine-independent format (either text or netCDF).

Filename	Format	Description
fort.23	Binary	Restart file
dtm1av‡	Binary	Climatological mixed-layer ocean temperature anomalies
hfcor.dat12	Binary	Surface heat flux adjustments
s1coravth*	Binary	Sea surface salinity adjustments
sfcor.dat12	Binary	Surface salinity tendency adjustments
t1coravth	Binary	Sea surface temperature adjustments
txcor.dat12	Binary	Zonal wind stress adjustments
tycor.dat12	Binary	Meridional wind stress adjustments

Table 5.3: The restart and auxiliary files required by the Mk3L coupled model, in addition to those required by the atmosphere and ocean models: the filename, the format, and a brief description. *New in Mk3L. ‡Only required for diagnostic purposes.

5.2.3 Coupled model

The coupled model also requires a restart file, containing the surface fluxes to be passed to the ocean at the first timestep. At the start of each run, the ocean model is executed before the atmosphere model; no surface fluxes will have been calculated, and they must therefore be read from file instead.

The model requires that the restart file have the name `fort.23`; as with the ocean model restart file `fort.21`, this name is not hard-coded into the model, but instead relies upon Fortran's implicit file naming.

The auxiliary files `hfcor.dat12`, `sfcor.dat12`, `txcor.dat12` and `tycor.dat12` contain the flux adjustments to be applied to the surface heat flux, the surface salinity tendency, and the zonal and meridional components of the surface momentum flux, respectively. These adjustments are applied to the surface fluxes passed from the atmosphere model to the ocean model.

The auxiliary files `t1coravth` and `s1coravth` contain the adjustments to be applied to the sea surface temperature and sea surface salinity respectively, and are applied to the surface fields passed from the ocean model to the atmosphere model. The file `s1coravth` is new in Mk3L, and is required because the sea surface salinity is now passed to the atmosphere model, to enable the surface freshwater fluxes to be converted to equivalent surface salinity tendencies (Section D.4).

The auxiliary file `dtm1av` contains climatological mixed-layer ocean temperature anomalies, diagnosed from an atmosphere model spin-up run. These are only used for diagnostic purposes, and are used to adjust the observed sea surface temperatures read from the file `clim3f.sst`, to allow for the fact that the stand-alone atmosphere model uses a mixed-layer ocean at high latitudes (Section 2.2).

All the restart and auxiliary files required by the coupled model are in a machine-dependent binary format.

5.3 Default input files

Default versions of all the restart and auxiliary files are supplied with the Mk3L source code. These files can be found in the following directories:

`core/data/atmosphere/`	The files required by the atmosphere model
`core/data/coupled/`	The files required by the coupled model
`core/data/ocean/`	The files required by the ocean model

The origins of each of the files are outlined in the following sections.

5.3.1 Atmosphere model

`core/data/atmosphere/albnew21f`

This file contains land surface albedos for the present day, and was supplied by CSIRO Marine and Atmospheric Research.

`core/data/atmosphere/amip2o3.dat`

This file contains AMIP 2 ozone concentrations (Wang et al., 1995), and was supplied by CSIRO Marine and Atmospheric Research.

`core/data/atmosphere/icedivl.R21`

This file contains a sea ice divergence limiter mask, which specifies gridpoints where the sea ice divergence should be set equal to zero. It was supplied by CSIRO Marine and Atmospheric Research.

`core/data/atmosphere/landrun21`

This file contains run-off relocation data for the present-day topography; this data is used to determine the path that run-off follows to the ocean. The original version was supplied by CSIRO Marine and Atmospheric Research; one modification was made during the development of Mk3L, to correct an error at the tip of the Antarctic Peninsula.

`core/data/atmosphere/psrk21f.dat`

This file contains surface pressure data for the present-day topography. It was supplied by CSIRO Marine and Atmospheric Research.

`core/data/atmosphere/sibrs.dat`
`core/data/atmosphere/sibsig.dat`
`core/data/atmosphere/sibsoil.dat`
`core/data/atmosphere/sibvegt.dat`
`core/data/atmosphere/sibz0.dat`

These files contain the data required by the “New SIB” land surface scheme, for the present-day distribution of vegetation. They were supplied by CSIRO Marine and Atmospheric Research.

`core/data/atmosphere/co2/co2_data.280ppm.18l`

This file contains the CO_2 transmission coefficients for an atmospheric carbon dioxide concentration of 280 ppm, and was generated using the utility `radint` (Section 5.5.1).

`core/data/atmosphere/currents/ocuv.3st_default`

This file contains climatological ocean currents diagnosed from the final 100 years of Mk3L ocean model spin-up run O-DEF (Phipps, 2006). It was generated using the procedure outlined in Section E.4.

`core/data/atmosphere/restart/rest.start_default`

This is a Mk3L atmosphere model restart file, and contains the state of the model at the end of spin-up run A-DEF (Phipps, 2006).

`core/data/atmosphere/sss/clim3f.sss_default`

This file contains climatological Levitus 1998 sea surface salinities for the start of each month, and was generated using the procedure outlined in Section E.4.

`core/data/atmosphere/sss/clim3f.sst_default`

This file contains climatological Levitus 1998 sea surface temperatures for the start of each month, and was generated using the procedure outlined in Section E.4.

5.3.2 Ocean model

`core/data/ocean/data/basic/bsnmask.nc`

This file defines the ocean basins which are used to calculate the meridional overturning streamfunctions, and was generated during the development of Mk3L (Section C.4.2).

`core/data/ocean/data/basic/ocean/sttop.bot_ind`

This file defines the ocean model bathymetry, and configures the model for present-day conditions. It was supplied by CSIRO Marine and Atmospheric Research.

`core/data/ocean/data/ocean/restart/orest.start_async`

This restart file was generated by integrating the Mk3L ocean model to equilibrium under asynchronous timestepping, using the default sea surface salinities, sea surface temperatures and surface wind stresses as the surface boundary conditions. It contains the state of the ocean model at the end of the asynchronous stage of spin-up run O-DEF (Phipps, 2006).

`core/data/ocean/data/ocean/restart/orest.start_levitus1998`

This restart file represents a state in which the ocean is at rest, and the temperatures and salinities are set equal to the annual-mean Levitus 1998 values. It was generated using the procedure outlined in Section E.6.

`core/data/ocean/data/ocean/restart/orest.start_sync`

This restart file was generated by integrating the Mk3L ocean model to equilibrium under synchronous timestepping, using the default sea surface salinities, sea surface temperatures and surface wind stresses as the surface boundary conditions. It contains the state of the ocean model at the end of spin-up run O-DEF (Phipps, 2006).

`core/data/ocean/data/ocean/sss/sss.dat_annual`

This file contains annual-mean Levitus 1998 sea surface salinities, and was generated using the procedure outlined in Section E.3.

`core/data/ocean/data/ocean/sss/sss.dat_monthly`

This file contains climatological Levitus 1998 sea surface salinities for the midpoint of each month, and was generated using the procedure outlined in Section E.3.

`core/data/ocean/data/ocean/sst/sst.dat_annual`

This file contains annual-mean Levitus 1998 sea surface temperatures, and was generated using the procedure outlined in Section E.3.

`core/data/ocean/data/ocean/sst/sst.dat_monthly`

This file contains climatological Levitus 1998 sea surface temperatures for the midpoint of each month, and was generated using the procedure outlined in Section E.3.

`core/data/ocean/data/ocean/stress/stress.dat_annual`

This file contains climatological annual-mean NCEP-DOE Reanalysis 2 wind stresses, and was generated using the procedure outlined in Section E.3.

`core/data/ocean/data/ocean/stress/stress.dat_monthly`

This file contains climatological NCEP-DOE Reanalysis 2 wind stresses for the midpoint of each month, and was generated using the procedure outlined in Section E.3.

5.3.3 Coupled model

`core/data/coupled/flux_adjustments/dtm1av_default`
`core/data/coupled/flux_adjustments/hfcor.dat12_default`
`core/data/coupled/flux_adjustments/s1coravth_default`
`core/data/coupled/flux_adjustments/sfcor.dat12_default`
`core/data/coupled/flux_adjustments/t1coravth_default`
`core/data/coupled/flux_adjustments/txcor.dat12_default`
`core/data/coupled/flux_adjustments/tycor.dat12_default`

These files contain flux adjustments derived from Mk3L spin-up runs, and were generated using the procedure outlined in Section E.5. The atmosphere model surface fluxes were diagnosed from the final 40 years of spin-up run A-DEF (Phipps, 2006), and the ocean model surface fluxes were diagnosed from the final 100 years of spin-up run O-DEF (Phipps, 2006).

`core/data/coupled/restart/oflux.start_default`

This is a coupled model restart file, in which all the surface fluxes are set equal to zero. It was generated using the procedure outlined in Section E.6.

5.4 Generating restart files

A variety of utilities are supplied to assist the user in the generation of new restart files; these are located in the directory `pre/restart/`.

5.4.1 Atmosphere model

Given the short timescale on which the atmosphere responds to changes in the external boundary conditions, the initial state of the atmosphere model is essentially irrelevant. The atmosphere model restart file does, however, also set the model's internal calendar. At the start of a simulation, the model reads the date from the restart file; if this file was written by a previous simulation at the end of (say) year 00100, then the first day of the new simulation will be regarded as being 1 January 00101.

When beginning a fresh simulation, the user will generally wish to reset the date; the Fortran 90 program `redate_restart_mk3.f90` enables the user to do this. This program reads an existing Mk3/Mk3L restart file, and prompts the user to enter a date. A new restart file is then generated, in which the date has been reset to the specified value, but which is otherwise identical. Note that the date must be specified in days since 1 January 00000; it is recommended that the value 365, which corresponds to 1 January 00001, be entered. The default restart value `rest.start_default` has the date set to this value.

The atmosphere model restart file is written in a machine-dependent binary format. The Fortran 90 program `convert_atmos_mk3_i8_to_i4.f90` is supplied, and was used to convert an existing restart file from CRAY binary format to little-endian (Section A.4.6). This program could readily be adapted to convert atmosphere model restart files between two alternative binary formats, in order to port Mk3L to a new architecture.

5.4.2 Ocean model

The Fortran 90 program `generate_orest.f90` (Section E.6) is supplied. Given an exisiting ocean model restart file, and netCDF files containing ocean temperatures and salinities, this program can be used to generate a new restart file. This file will represent a state in which the ocean is at rest, and in which the temperatures and salinities are set equal to the specified values.

The ocean model restart file is written in a machine-dependent binary format. The Fortran 90 program `convert_ocean_i8_to_i4.f90` is supplied, and was used to convert an existing restart file from CRAY binary format to little-endian. Note, however, that it can only be used to convert restart files generated by the original model source code; the format of the restart files has been changed in Mk3L. The Fortran 90 program `convert_ocean_to_new.f90` is also supplied; this converts ocean model restart files from the original format to the format used by Mk3L. See Sections A.4.5 and A.4.6 for further information regarding these utilities.

5.4.3 Coupled model

The surface fluxes contained with the coupled model restart file are only used for the first timestep of a simulation, and hence the values are essentially irrelevent. The Fortran 90 program `make_zero_oflux.f90` is supplied, and creates a coupled model restart file in which the surface fluxes are set to zero. This file can be used to initialise a new coupled model simulation.

The coupled model restart file is written in a machine-dependent binary format. The Fortran 90 program `convert_flux.f90` is supplied, and was used to convert an existing restart file from CRAY binary format to little-endian (Section A.4.6). This program could readily be adapted to convert coupled model restart files between two alternative binary formats, in order to port Mk3L to a new architecture.

5.5 Generating auxiliary files

A variety of utilities are supplied to assist the user in the generation of new auxiliary files; these are located in the directories `pre/auxiliary/` and `pre/co2/`.

5.5.1 Atmosphere model

CO_2 transmission coefficients

Files containing the CO_2 transmission coefficients should be generated using the utility `radint`; this is provided in the directory `pre/co2/`. To compile `radint`, change to this directory and enter the commands

```
make
make clean
```

[Note that the makefile may require modification before `radint` can be compiled on facilities other than `ac.apac.edu.au`.]

Two executables will be produced: `pset` and `radint`. `pset` generates data files which are required by `radint`, and must therefore be executed before `radint` can be used. To generate the necessary data files, enter the command

```
./pset -n 18
```

Note that these data files only need to be generated once, and that `pset` therefore does not have to be executed again.

`radint` can now be used to generate auxiliary files containing the CO_2 transmission coefficients. To generate the auxiliary file for an atmospheric CO_2 concentration of `<concentration>` ppm, enter the command

```
./radint -c <concentration>
```

This will generate a file, `co2_data`, containing the transmission coefficients. Note that it is only possible to generate these coefficients for atmospheric CO_2 concentrations between 165 and 1320 ppm.

Sea surface salinities

An IDL program, `construct_sss.pro`, is supplied. This reads monthly sea surface salinities from a netCDF file, and generates an auxiliary file in the format read by the model.

Sea surface temperatures

An IDL program, `construct_sst.pro`, is supplied. This reads monthly sea surface temperatures from a netCDF file, and generates an auxiliary file in the format read by the model.

Ocean currents

An IDL program, `construct_currents.pro`, is supplied. This reads monthly ocean currents from a netCDF file, and generates an auxiliary file in the format read by the model.

5.5.2 Ocean model

Sea surface salinities

An IDL program, `construct_levitus1998_sss_ogcm_v2d.pro`, is supplied. Given annual- and monthly-mean Levitus 1998 sea surface salinities, interpolated onto the model grid as described in Section E.3, it generates an auxiliary file in the format read by the model. This program could readily be adapted to generate an auxiliary file from any set of climatological sea surface salinities.

Sea surface temperatures

An IDL program, `construct_levitus1998_sst_ogcm_v2a.pro`, is supplied. Given annual- and monthly-mean Levitus 1998 sea surface temperatures, interpolated onto the model grid as described in Section E.3, it generates an auxiliary file in the format read by the model. This program could readily be adapted to generate an auxiliary file from any set of climatological sea surface temperatures.

Wind stresses

An IDL program, `construct_ncep2_stress_ogcm.pro`, is supplied. Given annual- and monthly-mean NCEP-DOE Reanalysis 2 wind stresses, interpolated onto the model grid as described in Section E.3, it generates an auxiliary file in the format read by the model. This program could readily be adapted to generate an auxiliary file from any set of climatological wind stresses.

5.5.3 Coupled model

Flux adjustments

`hfcor.dat12, sfcor.dat12, txcor.dat12, tycor.dat12`

Two IDL programs are supplied, which can be used to generate the auxiliary files `hfcor.dat12`, `sfcor.dat12`, `txcor.dat12` and `tycor.dat12`:

```
generate_flux_adjustments.pro
construct_flux_adjustments.pro
```

`generate_flux_adjustments.pro` reads netCDF files containing climatological surface fluxes, derived from atmosphere and ocean model spin-up runs. It then calculates the flux adjustments, and saves them to another netCDF file.

`construct_flux_adjustments.pro` reads the flux adjustments from the file generated by `generate_flux_adjustments.pro`, and generates an auxiliary file in the format read by the model.

`dtm1av, t1coravth`

Two IDL programs are supplied, which can be used to generate the auxiliary files `dtm1av` and `t1coravth`:

```
generate_dsst.pro
construct_dsst.pro
```

`generate_dsst.pro` reads netCDF files containing climatological sea surface temperatures, derived from observations and from atmosphere and ocean model spin-up runs. It then calculates the SST corrections required by the coupled model, and saves them to another netCDF file.

`construct_dsst.pro` reads the SST corrections from the file generated by `generate_dsst.pro`, and generates the auxiliary files `dtm1av` and `t1coravth`.

`s1coravth`

Two IDL programs are supplied, which can be used to generate the auxiliary file `s1coravth`:

`generate_dsss.pro`
`construct_dsss.pro`

`generate_dsss.pro` reads netCDF files containing climatological sea surface salinities, derived from observations and from an ocean model spin-up run. It then calculates the SSS corrections required by the coupled model, and saves them to another netCDF file.

`construct_dsss.pro` reads the SSS corrections from the file generated by `generate_dsss.pro`, and generates the auxiliary file `s1coravth`.

Conversion of auxiliary files

The coupled model auxiliary files are written in a machine-dependent binary format. The Fortran 90 programs `convert_fcorr.f90` and `convert_tscorr.f90` are supplied, and were used to convert existing auxiliary files from CRAY binary format to little-endian (Section A.4.7). `convert_fcorr.f90` converts the files `hfcor.dat12`, `sfcor.dat12`, `txcor.dat12` and `tycor.dat12`, while `convert_tscorr.f90` converts the files `t1coravth`, `s1coravth` and `dtm1av`.

These two programs could readily be adapted to convert coupled model auxiliary files between two alternative binary formats, in order to port Mk3L to a new architecture.

Chapter 6

Output files

6.1 Introduction

This chapter covers the output which is generated by Mk3L. The types of output produced by the model are summarised in Section 6.2. Section 6.3 describes the utilities which are supplied with Mk3L, and which are intended for the runtime processing of ocean model output. Section 6.4 lists the IDL utilities which are also supplied, and which are intended to assist with the analysis of model output.

6.2 Overview

The model generates four types of output:

- Diagnostic information
- Atmosphere model output
- Ocean model output
- Restart files

Each of these types of output is now discussed in turn.

6.2.1 Diagnostic information

Various diagnostic information is written to standard output by the model. This information largely consists of statistics generated by the atmosphere and ocean models, and can be controlled via the `namelist` parameters described in Sections 4.2 and 4.3. It is usually redirected to an output file, by running the model using a command such as

```
./model < input > output
```

6.2.2 Atmosphere model output

The output of the atmosphere model is controlled via the `namelist` parameters described in Section 4.2. Although a variety of different statistics can be generated, the user will generally only specify that monthly-mean model variables be saved. Table 4.7 lists the variables that are available.

The data can be saved in netCDF, in which case no processing of the model output is required. One file is generated for each variable; the filenames are of the form `svvv_xxx.nc`, where `vvv` is the variable name and `xxx` the run name.

6.2.3 Ocean model output

The ocean model writes monthly-mean variables to the binary file `fort.40`. This needs to be converted into a more portable and user-friendly format before it can be analysed, and a number of utilities are therefore provided with Mk3L for the processing of ocean model output. These utilities are described in Section 6.3.

6.2.4 Restart files

The atmosphere model, ocean model and coupled model each generate restart files at the end of a simulation; these files are described in Section 5.2.

The only difference between the restart files read by Mk3L at the start of a simulation, and those written at the end, is that the coupled model reads its initial state from the file `fort.23`, but saves its final state to the file `fort.13`. This file must therefore be renamed to `fort.23` before it can be used to initialise the model for another simulation.

6.3 Processing of ocean model output

Mk3L is supplied with a number of utilities, intended for runtime processing of ocean model output: `convert_averages`, `overturning`, `annual_averages` and `annual_overturning` (Section C.4).

6.3.1 `convert_averages`

Two slightly different versions of the utility `convert_averages` are supplied: `convert_averages`, which processes the output of the coupled model, and `convert_averages_ogcm`, which processes the output of the stand-alone ocean model. The need for these different versions arises from the fact that the coupled model can only be run for one year at a time, in which case the ocean model output file will only contain data for one year, whereas the stand-alone ocean model can be run for multiple years at a time, in which case the output file will contain data for multiple years.

The two versions of `convert_averages` are called differently:

```
convert_averages <input_file> <output_file>
```

converts one year of ocean model output to netCDF, while

Variable	Description	Units
itt	Timestep counter	-
dtts	Tracer timestep duration	s
relyr	Year counter	years
kmt	Ocean depth	model levels
smfzon	Zonal component of surface wind stress	Nm^{-2}
smfmer	Meridional component of surface wind stress	Nm^{-2}
stfht	Surface heat flux	Wm^{-2}
stfsal	Surface salinity tendency	kg/kg s^{-1}
temp	Potential temperature	°C
sal	Salinity	psu
u	Zonal component of velocity	ms^{-1}
v	Meridional component of velocity	ms^{-1}
w	Vertical component of velocity	ms^{-1}
uedd	Zonal component of eddy-induced velocity	ms^{-1}
vedd	Meridional component of eddy-induced velocity	ms^{-1}
wedd	Vertical component of eddy-induced velocity	ms^{-1}
res	Barotropic streamfunction	Sv
cdepthm	Maximum depth of convection	m

Table 6.1: The variables contained in the file generated by the utility `convert_averages`.

```
convert_averages_ogcm <run> <start_year> <end_year>
```

takes the output of ocean model run `run`, for years `start_year` to `end_year`, and generates one netCDF file for each year of the run.

The files produced by the two versions of `convert_averages` are identical, with one netCDF file being generated for each year of model output. The variables that this file contains are listed in Table 6.1.

6.3.2 `overturning`

The utility `overturning` calculates the oceanic meridional overturning streamfunctions, and is called as follows:

```
overturning <infile> <outfile>
```

This reads one year of ocean model output from the netCDF file `<infile>`, generated by `convert_averages`, and generates the netCDF file `<outfile>`. This contains the following monthly-mean meridional overturning streamfunctions:

`mola, molp, moli, molg`	Large-scale streamfunctions (Sv)
`moea, moep, moei, moeg`	Eddy-induced streamfunctions (Sv)
`mota, motp, moti, motg`	Total streamfunctions (Sv)

The suffixes a, p, i and g indicate the Atlantic, Pacific, Indian and World Oceans respectively.

The netCDF file also contains the annual means of each of these streamfunctions, with the following variable names:

```
molaann, molpann, moliann, molgann
moeaann, moepann, moeiann, moegann
motaann, motpann, motiann, motgann
```

6.3.3 annual_averages

The utility annual_averages reads monthly-mean ocean model output from the netCDF files generated by convert_averages, and calculates the annual means of each of the fields. (The variable cdepthm is an exception, with the annual maximum being calculated instead.) The data is then written to a single netCDF file.

annual_averages is called as follows:

```
annual_averages <run> <start_year> <end_year>
```

This reads the output of run run, for years start_year to end_year, and generates a single netCDF file containing annual-mean model output.

6.3.4 annual_overturning

The utility annual_overturning reads the annual-mean meridional overturning streamfunctions from the netCDF files generated by the utility overturning, and writes the data to a single netCDF file. It is not necessary to perform any calculations, as the annual-mean streamfunctions have already been calculated by overturning.

annual_overturning is called as follows:

```
annual_overturning <run> <start_year> <end_year>
```

This reads the meridional overturning streamfunctions for years start_year to end_year of run run, and generates a single netCDF file containing the annual-mean streamfunctions.

6.4 Analysis

Some programs, written in IDL, are provided in the directory post/. These utilities are intended to assist with the analysis of model output.

`average_fcor.pro`

Reads monthly-mean `fcor` output from the atmosphere model, derives climatological surface fluxes, and writes the values to a netCDF file. This program requires the data file `csiro_ogcm_ts_landsea.nc`, which is also supplied.

`average_ocuv.pro`

Reads monthly-mean output from the ocean model, derives climatological surface velocities, and writes the values to a netCDF file.

`average_sss.pro`

Reads monthly-mean output from the ocean model, derives climatological sea surface salinities, and writes the values to a netCDF file.

`average_sst.pro`

Reads monthly-mean output from the ocean model, derives climatological sea surface temperatures, and writes the values to a netCDF file.

`average_stfht.pro`

Reads monthly-mean output from the ocean model, derives the climatological surface heat flux, and writes the values to a netCDF file.

`average_stfsal.pro`

Reads monthly-mean output from the ocean model, derives the climatological surface salinity tendency, and writes the values to a netCDF file.

`calc_dmsl.pro`

Using the output of `rho_annual.pro`, this program calculates the change in mean sea level arising from changes in the density of the ocean. This program requires the data file `csiro_ogcm_ts_area_volume.nc`, which is also supplied.

`csiro_annual_climat.pro`

For a given atmosphere model variable, this program reads monthly-mean output from multiple netCDF files, and generates a single netCDF file containing annual means.

`csiro_annual_extent.pro`

This program reads monthly-mean sea ice concentrations from multiple netCDF files, and generates a single netCDF file containing annual-mean sea ice extent.

`csiro_annual_maximum.pro`

For a given atmosphere model variable, this program reads monthly-mean output from multiple netCDF files, and generates a single netCDF file containing annual maxima.

`csiro_annual_minimum.pro`

For a given atmosphere model variable, this program reads monthly-mean output from multiple netCDF files, and generates a single netCDF file containing annual minima.

`csiro_climat_agcm_stats_multi.pro`

For a given atmosphere model variable, this program reads monthly-mean output from multiple netCDF files, derives a climatology, and writes the values to a netCDF file.

`csiro_detrend_climat.pro`

Using the output of `csiro_annual_climat.pro`, this program calculates two timeseries: one high-pass filtered, and one low-pass filtered. This program requires the function `lowpass_3d.pro`, which is also supplied.

`csiro_detrend_mth_climat.pro`

Using the output of `csiro_monthly_climat.pro`, this program calculates two timeseries: one high-pass filtered, and one low-pass filtered. This program requires the function `lowpass_3d.pro`, which is also supplied.

`csiro_detrend_mth_sst.pro`

Using the output of `sst_monthly.pro`, this program calculates two timeseries of monthly-mean sea surface temperature: one high-pass filtered, and one low-pass filtered. This program requires the function `lowpass_3d.pro`, which is also supplied.

`csiro_detrend_sst.pro`

Using the output of `ts_annual.pro`, this program calculates two timeseries of annual-mean sea surface temperature: one high-pass filtered, and one low-pass filtered. This program requires the function `lowpass_3d.pro`, which is also supplied.

`csiro_month_climat.pro`

For a given atmosphere model variable, this program reads monthly-mean output from multiple netCDF files, and generates a single netCDF file containing the data for a particular calendar month.

`csiro_monthly_climat.pro`

For a given atmosphere model variable, this program reads monthly-mean output from multiple netCDF files, and writes the data to a single netCDF file.

`enso_monthly.pro`

Using high-pass filtered mean sea level pressure and sea surface temperature data, this program calculates various El Niño-Southern Oscillation statistics.

`find_elnino.pro`

Using the output of `enso_monthly.pro`, this program detects El Niño events. The definition of Trenberth (1997) is employed, with an El Niño event being defined as a period of at least six consecutive months when the five-month running mean of the sea surface temperature anomaly in the Niño 3.4 region (170–120°W, 5°S–5°N) exceeds +0.4°C.

`rho_annual.pro`

Reads monthly-mean output from the ocean model, derives annual-mean density, and writes the values to a netCDF file. This program requires the function `rho.pro`, which is also supplied.

`sst_monthly.pro`

Reads monthly-mean output from the ocean model, and generates a single output file containing the sea surface temperature data.

`sst_pcs_detrend.pro`

Using the output of `csiro_detrend_sst.pro`, this program derives the principal components of annual-mean sea surface temperature.

`ts_annual.pro`

Reads monthly-mean output from the ocean model, and produces netCDF files containing the annual-mean temperature, salinity, sea surface temperature, sea surface salinity, surface heat flux and surface salinity tendency.

`ts_stats.pro`

Reads monthly-mean output from the ocean model, and produces a single netCDF file containing various temperature and salinity statistics. This program requires the data file `csiro_ogcm_ts_area_volume.nc`, which is also supplied.

`tsc_pcs_detrend.pro`

Using high-pass filtered screen temperatures, this program derives the principal components of annual-mean screen temperature.

Appendix A

Porting and optimisation

A.1 Introduction

Extensive modifications were made to the original model source code during the development of Mk3L. This appendix documents the modifications which were made to the source code in order to port it to a number of new platforms, and in order to optimise the runtime performance. Appendices B, C and D document the remaining modifications which were made to the atmosphere model, the ocean model and the coupled model respectively; these modifications largely relate to the model physics.

The modifications to the source code were made in an incremental fashion and over an extended period of time (between November 2001 and May 2005). In the interests of clarity, the modifications are presented here in a logical order, and *not* the chronological order in which they were made. However, this appendix remains a complete and accurate summary of the changes which were made to the source code during the process of porting and optimisation.

A.2 The original source code

The original source code (v6.2, released November 2001) consists of 325 files, which are summarised in Table A.1. Data is shared between routines through the use of `common` blocks, and global parameters are defined within header files. Of the 235 Fortran source files, 100 contain `implicit none` statements, with the remainder of the source code relying, in part, on implicit typing. A high degree of shared-memory parallelism is achieved within the atmosphere model through the use of proprietary compiler directives (Section A.2.3). However, no parallelisation is specified within the ocean model.

Table A.2 shows the combinations of platforms and operating systems which are supported; in each case, the operating systems represent proprietary versions of UNIX. Of the four versions, the SGI version is intended to be relatively generic, and is designed to run on other UNIX platforms with minimal, if any, changes. However, the use of proprietary compiler directives within this version means that parallel execution can only be achieved on SGI machines; on other machines, only serial execution is possible. The SGI version is optimised for scalar architectures, whereas the other three versions are optimised for vector architectures.

Contents	Number of files	Total number of lines
Fortran 77 source code	231	85,720
Fortran 90 source code	4	682
C source code	1	34
Header files	88	3,369
Makefile	1	566
Total	**325**	**90,371**

Table A.1: A summary of the original model source code.

Platform	Operating system
CRAY	UNICOS
Fujitsu	UXP/V
NEC	SUPER-UX
SGI	IRIX

Table A.2: The platforms and operating systems supported by the original version of the model.

Support for multiple platforms, and optimal runtime performance of the code on each platform, is achieved through the following:

- the detection of the machine type at runtime
- the use of platform-specific sections of the source code, and of platform-specific external libraries
- the use of proprietary compiler directives, particularly in order to specify manual parallelisation of loops
- the specification of different compilers (including auto-parallelising compilers) and compiler options in the makefile

Each of these features are discussed in detail in the following sections.

A.2.1 Detection of machine type

The machine type is detected at runtime through examination of the environment variable `HOSTX`, which must be set equal to the value of the UNIX `hostname` command prior to execution. The value of `HOSTX` is compared to a list of known hosts and, if a match is found, the variable `machine` is set equal to either `CRAY`, `FUJI`, `NEC` or `SGI`, as appropriate. If this process fails, execution is aborted.

This approach has the disadvantage of requiring advance knowledge of the machine(s) on which the model will be run. The ‘look up’ table of known hosts is contained within the source code, requiring modification of the code, and hence production of a fresh executable, before a new machine can be used. This is necessary even when the machine belongs to a supported platform, and even if an executable has already been produced for that platform.

Furthermore, as the path through the source code is not determined until runtime, this approach requires that all of the source code for all of the supported platforms be compiled into the executable. This is both

Scalar-optimised version	Vector-optimised version
dtog	dtogcray
dynmseca	dynm
physseca	phys
ptog	ptogcray
semiis	semii

Table A.3: The names of scalar- and vector-optimised versions of subroutines in the original model source code, where alternate versions exist.

unnecessary, and creates a large number of undefined references to external routines. These references can only be resolved through the inclusion of dummy routine definitions, which are contained within the files `cnecdums.f`, `fujidums.f`, `necdums.f` and `sgicalls.f`. An example of a dummy subroutine definition, taken from `cnecdums.f`, is as follows:

```
subroutine physseca
print*,'Error, dummy routine physseca called'
stop
end
```

A.2.2 Platform-specific source code

In order for the source code to give optimal runtime performance across a number of scalar and vector platforms, platform-specific versions of some sections of the source code are provided. There are essentially two reasons why this is the case:

- loop structure
- the use of proprietary scientific libraries

Loop structure

Different loop structures give optimal performance on scalar and vector architectures (e.g. Ford and Snelling, 1997; Ashworth, 2000; Michalakes et al., 2002). As a result, the source code includes scalar- and vector-optimised versions of the subroutines listed in Table A.3, as well as of sections of some other routines.

The use of proprietary scientific libraries

Proprietary scientifc libraries are employed on the CRAY, Fujitsu and NEC platforms, largely to carry out Fast Fourier Transforms (FFTs), but also to carry out matrix operations. These libraries employ algorithms and source code that have been designed to give optimal performance on their intended platforms.

Table A.4 lists the subroutines which are called on each platform to calculate FFTs. On the CRAY, Fujitsu and NEC platforms, these subroutines make calls to external scientifc libraries. On the SGI platform,

Subroutine	Platform
asmrv	SGI
cool	SGI
factr	SGI
fftfax	SGI
fixrl	SGI
mfftg	CRAY/NEC
mfftgf	Fujitsu
mfftgs	SGI
mfftm	CRAY/NEC
mfftmf	Fujitsu
mfftms	SGI
smfac	SGI
symrv	SGI

Table A.4: Fast Fourier Transform routines in the original model source code, and the platforms on which they are called.

however, the subroutines contain a set of native FFT routines. These have not been optimised for any particular platform, and hence can be expected to perform less well than their proprietary equivalents.

As the names of routines vary between libraries, the use of proprietary scientific libraries requires that different calls to external routines be made on different platforms. An example of this, taken from the subroutine `gauleg`, is as follows:

```
if(machine.eq.'CRAY')then
  call  fftfax(lon,ifax,trigs)
elseif(machine.eq.'NEC')then
  call  rftfax(lon,ifax,trigs)
elseif(machine.eq.'SGI ')then
  call fftfax
endif
```

In this code fragment, calls are made to different FFT initialisation routines on each platform. The proprietary routines `fftfax` and `rftfax` are called on the CRAY and NEC platforms respectively; on the SGI platform, the native FFT initialisation routine `fftfax` is called instead. No call is made on the Fujitsu platform, as the Fujitsu FFT library does not require any call to an initialisation routine.

A.2.3 Proprietary compiler directives

On the CRAY, NEC and SGI platforms, shared-memory parallelism is achieved within the atmosphere model through the use of proprietary compiler directives. The Fujitsu Fortran compiler does not support explicit parallelisation directives, performing automatic parallelisation instead (Section A.2.4). An important feature of all compiler directives is that they are treated as comments, and hence ignored, by all compilers other than those for which they are intended.

The directives instruct the compiler that certain loops should be executed in parallel, and indicate the scope of the variables referenced within those loops (i.e. whether they should be private to each thread, or shared between them). If a variable is private, then each thread has its own temporary copy of that variable. Each copy will be located at a different memory address, and will only exist for the lifetime of the thread. Such variables contain temporary data which is not required outside of a parallel region, such as the results of intermediate calculations.

If a variable is shared, however, then only one copy of that variable will exist. Each thread will access this copy of the variable which, unlike private variables, will exist outside of the parallel region. Shared variables typically contain the data which forms the input to the calculations performed within a parallel region, or which represents the results of those calculations.

Within the subroutine `dynm`, the following directives declare that each of these `common` blocks should be private to each thread:

```
CDIR$ TASK COMMON GIANT1
*PDIR TASKLOCAL ( /GIANT1/ )
CDIR$ TASK COMMON GIANT4
*PDIR TASKLOCAL ( /GIANT4/ )
CDIR$ TASK COMMON LEGND
*PDIR TASKLOCAL ( /LEGND/ )
CDIR$ TASK COMMON WORK1X
*PDIR TASKLOCAL ( /WORK1X/ )
CDIR$ TASK COMMON WORKNSD
*PDIR TASKLOCAL ( /WORKNSD/ )
```

The `CDIR$ TASK COMMON` directives apply on the CRAY platform, while the `*PDIR TASKLOCAL` directives apply on the NEC platform. On the SGI platform, an equivalent role is performed by compiler directives specified within the makefile.

The main loop within the routine is then parallelised using the following directives:

```
c$doacross LOCAL(lg,x2,x3,mm,ll,k,mg,x2mm1,zonpsl,lgn),
c$&        SHARE(efor,efei,efer,efoi,gfor,gfei,gfer,gfoi,
c$&              ffor,ffer,hfoi,hfei,pbor,pbei,pber,pboi,
c$&              xaor,xaei,xaer,xaoi,paor,paei,paer,paoi,
c$&              xbor,xbei,xber,xboi)
cmic$ do all
cmic$&  shared (efor,efei,efer,efoi,gfor,gfei,gfer,gfoi,
cmic$&          ffor,ffer,hfoi,hfei,pbor,pbei,pber,pboi,
cmic$&          xaor,xaei,xaer,xaoi,paor,paei,paer,paoi,
cmic$&          xbor,xbei,xber,xboi,
cmic$&          lstat,tdt,nl,
cmic$&          rmg,machine,zpslk,
cmic$&          wocs,w,plmg,cplmg,iphyst,ronmx,sonmx,
cmic$&          savegrid,ugd,vgd,pgd,impvor,rampm)
cmic$&  private(lg,x2,x3,mm,ll,k,mg,x2mm1,zonpsl,nex,
cmic$&          lgn,
```

```
cmic$&          aon,afnr,afni,
cmic$&          bon,bfnr,bfni,
cmic$&          eon,efnr,efni,
cmic$&          fon,ffnr,ffni,
cmic$&          gon,gfnr,gfni,
cmic$&          hon,hfnr,hfni,
cmic$&          bpnr,bpni,apni,apnr,
cmic$&          cplm,pn,un,vn,rmn,z4,ron,son,plm)
cfpp$   cncall
*PDIR PARDO FOR
```

The directives beginning with `c$`, `cmic$`/`cfpp$` and `*PDIR` specify parallel execution on the SGI, CRAY and NEC platforms respectively.

The `LOCAL`/`private` and `SHARE`/`shared` clauses declare the scope of each variable. On the NEC platform, this is achieved through the following directives, which are placed at the start of the subroutine:

```
*PDIR SAVE (efor,efei,efer,efoi,gfor,gfei,gfer,gfoi)
*PDIR SAVE (ffor,ffer,hfoi,hfei,pbor,pbei,pber,pboi)
*PDIR SAVE (xaor,xaei,xaer,xaoi,paor,paei,paer,paoi)
*PDIR SAVE (xbor,xbei,xber,xboi)
*PDIR SAVE ( lstat, iphyst )
```

In this case, SAVE declares that the specified variables should be shared between each thread.

It should be noted that the list of variables given an explicit scope varies between platforms. This can be attributed to the differing behaviour of the compilers on each platform, particularly with regard to the default scope which is assigned to a variable when none is declared explicitly.

Compiler directives are also placed within the source code in order to control the auto-parallelising (Section A.2.4) and vectorising abilities of the compilers on some platforms. For example, the following directive within the subroutine `conv` indicates that the loop should not be parallelised by the CRAY Fortran compiler:

```
CDIR$ NEXTSCALAR
         do 70 j=kb,kt
 70         sumqt=sumqt+qtg(mg,j)*dskm(mg,j)
```

The following directives within the subroutine `convecu`, however, indicate that the loop can safely be vectorised by the CRAY and Fujitsu Fortran compilers:

```
CDIR$ IVDEP
*vdir nodep
      DO 130 I=1,NCONV
        BINIT(INDEX1(I)) = BWORK(I,4)
  130 CONTINUE
```

Platform	Compiler and options
CRAY	`f90 -O task1,overindex -dp -f fixed`
NEC	`f90 -Pmulti -Wf"-tasklocal micro" -Cvopt -float0 -f0`
SGI	`f90 -r8 -i8 -mp -O2 -LNO:fusion=2:cs2=4m -OPT:Olimit=0 -fixedform`

Table A.5: The compilers and options used to compile routines containing parallel compiler directives, in the original version of the model.

A.2.4 Compilers and compiler options

On the Fujitsu platform, the Fortran source code is compiled using

```
frt -Wv,-p2200 -Sw -Ad
```

The `frt` compiler is an auto-parallelising compiler, meaning that it considers all of the `do` loops within the source code as being candidates for parallelisation. Each loop is assessed against the following criteria:

- that there are no data dependencies between iterations of the loop
- that there is sufficient work contained within the loop

The first of these criteria is satisfied if the compiler determines that the calculations performed during one iteration of the loop do not depend on the calculations performed during any other iteration. If this is the case, then each iteration of the loop can be performed independently of the others. The second criterion is fullfilled if the compiler determines that the performance gain arising from execution of the loop in parallel will exceed the performance loss arising from the overheads associated with parallel execution (such as launching and terminating threads, and creating and merging multiple copies of data).

If a loop satisfies both criteria, then it is transformed such that it will execute in parallel on shared-memory systems.

On the other platforms, compiler directives within the source code are used to determine which loops should be executed in parallel. Routines which contain these directives are compiled using the compilers and options shown in Table A.5. However, for certain atmosphere model routines which have not already been parallelised, the auto-parallelising capabilities of the Fortran compilers are exploited. These routines are compiled using the options shown in Table A.6. On the CRAY, NEC and SGI platforms, the ocean model routines are always executed in serial; no parallelisation is specified with the source code, and the auto-parallelising capabilities of the Fortran compilers are not exploited.

A.3 Hardware used

Mk3L was developed and tested at the Australian Partnership for Advanced Computing National Facility (Australian Partnership for Advanced Computing, 2005) in Canberra. Two different machines were used:

Platform	Compiler and options
CRAY	`f90 -O task2,overindex -dp -f fixed`
NEC	`f90 -Pauto -Wf"-tasklocal micro -fopp res=no" -Cvopt -R1 -float0 -f0`
SGI	`f90 -r8 -i8 -mp -O2 -LNO:fusion=2:cs2=4m -OPT:Olimit=0 -pfa -fixedform`

Table A.6: The compilers and options used to auto-parallelise routines which do not contain parallel compiler directives, in the original version of the model.

AlphaServer SC 126 Compaq AlphaServer SC nodes, each containing:
- 4 $\times$ 1GHz EV68 (Alpha 21264C) CPUs
- between 4 and 16GB of RAM

Tru64 UNIX operating system

Linux Cluster 152 Dell Precision 350 nodes, each containing:
- 1 $\times$ 2.66GHz Intel Pentium 4 CPU
- 1GB RAM

Linux operating system

The AlphaServer SC was decommissioned in June 2005, while the Linux Cluster is still in service at the time of writing (August 2006).

The source code was also tested on a CRAY SV1 (using the UNICOS operating system) and an SGI Origin 3400 (using the IRIX operating system). Both of these machines were located at the University of Tasmania in Hobart.

A.4 Porting

In developing Mk3L, it was intended that the source code should satisfy the following criteria:

- it is portable across as many architectures as possible, without any modifications to the source code being required
- it is optimised for scalar architectures
- it is as “safe” as possible, conforming to the Fortran standard, containing no array-bounds violations, and generating no floating-point exceptions at runtime
- the output of the model should be fully reproducible

This necessitated the following actions:

- minimising the dependence on external libraries, and removing all dependence on proprietary libraries
- removing all proprietary compiler directives

Package	Author/Vendor	Equivalent speed ("mflops")	
		Forward transforms	Backward transforms
FFTW	FFTW	1649.3	1581.0
CXML	Hewlett-Packard	994.49	938.79
Ooura FFTs	Takuya Ooura	959.77	959.77
FFTs for RISC 2.0	John Green	762.99	749.67
JMFFT	Jean-Marie Teuler	377.30	377.58

Table A.7: FFTW benchmarks.

- removing all platform-specific source code
- ensuring that the source code can be compiled using a range of compilers
- extensive checking of the source code
- conversion of the original restart files to a more generic binary format

Each of these actions are discussed in detail in the following sections.

A.4.1 Minimising the dependence on external libraries

The original version of the source code is dependent on the netCDF library (network Common Data Form, Unidata Program Center, 2005), and on a number of proprietary scientific libraries.

NetCDF is a freely-available, open-source library, providing a self-describing, machine-independent format for representing scientific data. These properties satisfy the above criteria (Section A.4), and so use of the netCDF library was retained. However, in order to bring the source code up-to-date and to improve runtime performance, the netCDF routine calls were updated from version 2.4 to 3.x of the Fortran interface.

Much of the dependence on proprietary scientific libraries relates to the use of Fast Fourier Transforms (FFTs), as discussed in Section A.2.2. It was decided to use the FFTW library (Fastest Fourier Transform in the West, FFTW, 2005b) instead. As with the netCDF library, FFTW is freely available, open source and portable across a wide range of platforms. Furthermore, FFTW is typically faster than other publicly-available FFT software, and is comparable even with proprietary FFT libraries.

The superior performance of FFTW is demonstrated by Table A.7, which shows benchmarks for the transformation of 64x64 64-bit `real` arrays on an 833MHz Alpha EV6, taken from the FFTW website (FFTW, 2005b). These particular timings are reproduced here, as they most closely resemble the actual transformations performed within the model, and the actual hardware on which Mk3L was developed. FFTW is compared with four other FFT packages: CXML (Hewlett-Packard Company, 2005), the Ooura FFTs (Ooura, 2005), FFTs for RISC 2.0 (FFTW, 2005b) and JMFFT (Teuler, 2005).

The benchmarks provided were obtained using benchFFT (FFTW, 2005a), and are supplied as equivalent speeds in "mflops". If N is the number of data points, and T μs is the time taken to complete one FFT operation, then the equivalent speed in "mflops" is given by

$$\frac{2.5Nlog_2(N)}{T} \tag{A.1}$$

Multiple benchmarks are provided for FFTW; of these, the ones presented here are for version 2.x, when it is initialised using the `fftw_measure` option. These timings are the most appropriate, as version 2.x is the one against which Mk3L is linked; the fastest runtime performance is also obtained when FFTW is initialised using the `fftw_measure` option, and this is therefore how it is initialised within Mk3L. Multiple benchmarks are also provided for the Ooura FFTs; of these, the ones presented here are for the function `ooura-4f2d`, which provides the fastest timings.

It can be seen that FFTW is considerably faster than all the other packages, and is more than 1.6 times faster even than the CXML routines, which are optimised for Alpha systems.

All calls to FFT routines within the model code were replaced with equivalent calls to version 2.x of the FFTW Fortran interface. For example, the above section of the subroutine `gauleg` (Section A.2.2), which initialises the FFT routines, was replaced with:

```
      call rfftw_f77_create_plan(plan_backward, lon,
     &    fftw_complex_to_real, fftw_measure+fftw_threadsafe)
      call rfftw_f77_create_plan(plan_forward, lon,
     &    fftw_real_to_complex, fftw_measure+fftw_threadsafe)
```

All of the subroutines listed in Table A.4 were replaced with just two new subroutines, `mfftga` and `mfftma`.

After these changes, a few calls still remained to proprietary scientific libraries, such as the following calls to the proprietary matrix multiplication routines `sgemm` and `mmtx`, which are contained within the subroutine `ftospec`:

```
      if(machine.eq.'CRAY')then
        call sgemm('n','n',nstack,(llmax+1)/2,lat,1.0,fm,nstack
     &             ,plmo,lat,0.,prodo,nstack)
      elseif(machine.eq.'NEC')then
        call mmtx(fm,plmo,prodo,nstack,lat,(llmax+1)/2)
      endif
```

In all cases, equivalent routines are included within the model source code. By using these instead, all the remaining calls to external libraries could be removed.

Dependence on external libraries was therefore reduced to just two libraries, both of which are freely available, open source and portable across a wide range of platforms:

- netCDF 3.x
- FFTW 2.x

As a final measure, care was also taken to ensure that the precision of all variables passed as arguments to these external libraries is specifically declared as being either 16-, 32- or 64-bit, as appropriate, ensuring complete portability across 32- and 64-bit architectures.

A.4.2 Removing proprietary compiler directives

All of the shared-memory parallelism achieved through the use of platform-specific proprietary compiler directives can be achieved through the use of equivalent OpenMP directives (OpenMP Architecture Review Board, 2005). OpenMP is fully portable, and provides for shared-memory parallel programming across a wide range of platforms.

The proprietary parallel compiler directives were removed, and replaced with the equivalent directives from the OpenMP Fortran interface. For example, in the subroutine `dynm`, the directives which declare that the appropriate `common` blocks should be made private to each thread (Section A.2.3) were replaced with the following:

```
!$OMP THREADPRIVATE ( /GIANT1/ )
!$OMP THREADPRIVATE ( /GIANT4/ )
!$OMP THREADPRIVATE ( /LEGND/ )
!$OMP THREADPRIVATE ( /WORK1X/ )
!$OMP THREADPRIVATE ( /WORKNSD/ )
```

The main loop is then parallelised using the directives:

```
!$OMP  PARALLEL
!$OMP& PRIVATE (i, j, k, lg, lgn, ll, ll_lim, mg, mm, nex, x2, x2mm1,
!$OMP&          x3, zonpsl)
!$OMP& SHARED  (cplmg, cplmge, cplmgo, efei, efer, efoi, efor, eli,
!$OMP&          elr, ffer, ffor, gfei, gfer, gfoi, gfor, hfei, hfoi,
!$OMP&          impvor, iphyst, ipi, ipr, iti, itr, ixi, ixr, lstat,
!$OMP&          paei, paer, paoi, paor, pbei, pber, pboi,
!$OMP&          pbor, pgd, plmg, plmge, plmgo, rampm, rmg, ronmx,
!$OMP&          savegrid, sonmx, tdt, ugd, vgd, w, wocs, xaei, xaer,
!$OMP&          xaoi, xaor, xbei, xber, xboi, xbor, zpslk)
```

Careful checking was carried out in order to ensure that all variables were given an explicit scope, with only two exceptions:

- variables which have already been declared as being private to each thread, through the use of `!$OMP THREADPRIVATE` directives
- variables which have been declared as having the `parameter` attribute; these are automatically regarded by the HP Fortran compiler as being shared between each thread, and the compiler displays an error message if these variables are explicitly declared as being `SHARED`

If multiple parallel regions exist within a single subroutine, these were merged into a single parallel region wherever possible, in order to minimise the overheads associated with the creation and destruction of threads.

The remaining proprietary compiler directives, which control the auto-parallelising and vectorising abilities of the Fortran compilers on the previously-supported platforms, were left in place. As they are ignored by other compilers, it was unnecessary to remove them, and they were retained so as to enable Mk3L to be used on these platforms in future.

File	Description	Platform(s)
MACHINE.f	Header file	All
asmrv.f	Contains the subroutine asmrv	SGI
cnecdums.f	Contains dummy subroutines	NEC
cool.f	Contains the subroutine cool	SGI
dtog.f	Contains the subroutine dtog	SGI
dynmseca.f	Contains the subroutine dynmseca	SGI
factr.f	Contains the subroutine factr	SGI
fftfax.f	Contains the subroutine fftfax	SGI
fixrl.f	Contains the subroutine fixrl	SGI
fujidums.f	Contains dummy subroutines	Fujitsu
mfftg.f	Contains the subroutine mfftg	CRAY
mfftgf.f	Contains the subroutine mfftgf	Fujitsu
mfftgs.f	Contains the subroutine mfftgs	SGI
mfftm.f	Contains the subroutine mfftm	CRAY
mfftmf.f	Contains the subroutine mfftmf	Fujitsu
mfftms.f	Contains the subroutine mfftms	SGI
necdums.f	Contains dummy subroutines	NEC
physseca.f	Contains the subroutine physseca	SGI
ptog.f	Contains the subroutine ptog	SGI
semii.f	Contains the subroutine semii	CRAY/Fujitsu/NEC
sgicalls.f	Contains dummy subroutines	SGI
smfac.f	Contains the subroutine smfac	SGI
symrv.f	Contains the subroutine symrv	SGI

Table A.8: Platform-dependent source files which were removed from the source code.

A.4.3 Removing platform-specific source code

The above changes (Sections A.4.1 and A.4.2) removed many of the platform-specific sections of the source code. The only remaining examples related to the provision of alternate sections of source code for scalar and vector architectures. In each of these cases, a single code path was chosen. Given the desire to produce a version of the model that is optimised for scalar architectures, it was generally the scalar-optimised version of the code that was retained.

All platform dependence had now been removed from the source code, enabling the section of code which determines the platform at runtime to be removed. A number of source files (Table A.8) were now redundant, and were removed from the source code.

A.4.4 Compilation

The modified source code was compiled and checked on the machines shown in Table A.9. The minimum compiler options that must be specified in order to compile the model on these machines were:

Machine	Compiler
AlphaServer SC	HP Fortran 90 compiler v5.5a (Hewlett-Packard Development Company, 2005a)
Linux Cluster	Intel Fortran compiler v8.1 (Intel Corporation, 2005)

Table A.9: The machines and compilers used to check the model source code.

AlphaServer SC	`f90 [-omp] -r8 -align dcommons`
Linux Cluster	`ifort -r8 -align dcommons`

[The single C routine is an execption, being compiled using `cc -O` and `gcc -O` on the AlphaServer SC and Linux Cluster respectively; this routine will not be discussed further.]

The `-r8` option ensures the use of 64-bit precision for all `real` variables, unless specified otherwise within the source code, while the `-align dcommons` option ensures the appropriate alignment of variables within `common` blocks. On the AlphaServer SC, `-omp` enables shared-memory parallel processing, as directed through the insertion of OpenMP directives (Section A.4.2).

The `-warn nouncalled` compiler option was also specified on both machines, in order to disable a small number of warnings - which are of no concern - arising from the use of statement functions within the source code.

It was necessary to modify a few variable declarations before the code would compile successfully. However, beyond these minor changes, it was possible to produce an executable on both machines, without any error or warning messages being generated by the compilers.

A.4.5 Checking the source code

Code syntax

As the source code consists almost entirely of Fortran 77 and fixed-format Fortran 90, it is important that none of the executable statements (i.e. excluding comments) extend past column 72. Should this happen, one possible outcome is that a compiler error will arise, due to the truncated line representing an invalid Fortran statement. A far more dangerous outcome, however, is that the trunacted line will represent a valid Fortran statement, but not the one required for correct execution of the code. This will create an error which might remain undetected.

awk was used to check the source code. Only one line, contained within the subroutine `xtconv`, was found to contain executable statements extending beyond column 72. This line was fixed; however, it was later found that this routine is not compiled into the model anyway (Section A.6.2).

Floating-point exceptions

The default behaviour of the HP Fortran 90 compiler is to generate an executable which will terminate if a floating-point exception takes place. Although the complier can be instructed to handle floating-point

exceptions differently, this can have a significant impact on runtime performance. Furthermore, floating-point exceptions indicate that a problem has arisen within the model, and it is dangerous to allow execution to continue in this case.

When initial attempts were made to run the model on the AlphaServer SC, floating-point exceptions caused execution to terminate during the first timestep. These exceptions were traced to two lines within the source code; the first, contained within the subroutine `newrain`, was:

```
fcol=min(1.,clfra(mg)/clfr(mg,k))
```

It is possible for `clfr(mg,k)` to be equal to zero, in which case a floating-point exception arises. Inspection of the source code, however, revealed that the variable `fcol` is never referenced. This line is therefore redundant, and was deleted.

The second line, contained within the subroutine `surfa`, was:

```
ewwp(mg)=min(pmc(mg,lg)/dt*hl,max(beta*pevap(mg),0.))
```

The cause of floating-point exceptions in this case was found to be more subtle, arising from the fact that some of the elements of the array `pmc` contain denormalised numbers. These are numbers whose values lie between zero and the smallest finite value that can be represented by a normalised number (being one in which the leading bit of the mantissa is 1). In the case of 64-bit `real` variables such as `pmc`, a denormalised number has a magnitude that lies within the approximate range 4.94×10^{-324} to 2.23×10^{-308}. The default behaviour of the HP Fortran 90 compiler is to generate an executable which will terminate if such a value is present in an arithmetic expression. The denormalised numbers were found to originate from within the atmosphere model restart file, and were found to be restricted to the array `pmc`.

The solution to this problem was to modify the program `convert_atmos_mk3_i8_to_i4` (Section A.4.6) such that it sets all the elements of the array `pmc` equal to zero. `pmc` is used within the land surface scheme, and contains puddle depths. Setting these values equal to zero at the beginning of a run might be expected to be of little consequence and, indeed, it was establised that these values recover rapidly during an atmosphere model spin-up run.

After making these changes, the atmosphere model was found to run for 50 years on the AlphaServer SC, without any floating-point exceptions taking place.

Rounding errors

Comparison of the output of the model with that of runs conducted on a CRAY machine, using the same source code, revealed that some model variables were being set to incorrect values. This was found to be the result of rounding errors arising from lines such as the following, contained within the subroutine `ocean`:

```
knitd=24.*3600./DTTSF
```

In this line, the number of ocean model timesteps per day is calculated by dividing the duration of the tracer timestep in seconds, contained within the `real` variable `dttsf`, into the number of seconds in a day. The result is then stored in the `integer` variable `knitd`.

Such statements, which represent an implicit conversion of data from one type to another, require that the compiler insert additional code. In this case, the compiler must insert code to convert the value of the expression `24.*3600./dttsf` from `real` to `integer` type. As `dttsf` is always positive, the result is that the value of `24.*3600./dttsf` is rounded down to the next smallest integer e.g. if it is equal to 29.999999, then `knitd` is set equal to 29, and not 30.

This is exactly what was found to occur. When `dttsf` is set equal to 2880 seconds in the model control file, `knitd` should be set equal to 30. However, in the runs conducted on the CRAY machine, it was found that `knitd` was being set equal to 29 instead. The solution was to replace the above line with

```
knitd = int(24.0 * 3600.0 / DTTSF + 0.1)
```

A number of similar lines were modified in the same way.

Array-bounds violations

An array-bounds violation occurs if a reference is made to an element of an array which lies outside the declared bounds of that array. For example, if the array `data` is declared as `real data(10)`, then the following code fragment gives rise to array-bounds violations:

```
sum = 0.0
do i = 1, 20
  sum = sum + data(i)
end do
```

Such code might be expected to give both incorrect and inconsistent results, as there is no way of knowing what values will lie at the memory addresses corresponding to the array elements `data(11)`, `data(12)` ... `data(20)`.

The following code fragment is even more dangerous:

```
do i = 1, 20
  data(i) = data(i) + 1.0
end do
```

Changes are made to the values stored at the memory addresses corresponding to `data(11)` ... `data(20)`, even though it is not known what is been stored at these addresses. This can result in unpredictable changes being made to the values of other variables, or even to the binaries of programs which are currently running.

The default behaviour of the HP Fortran 90 compiler is to make no runtime checks for array-bounds violations, as this can significantly increase execution times. It is therefore being assumed that appropriate checks have been made during the development of the source code.

In order to establish whether or not the source code contains any array-bounds violations, the model was recompiled on the AlphaServer SC, with the addition of the `-check bounds` compiler option. It was then run in three modes:

co21(imax, 1, 1)	≡	co21(imax, 1, 1)
co21(imax+1, 1, 1)	≡	co21(1, 2, 1)
co21(imax+2, 1, 1)	≡	co21(2, 2, 1)
...		...
co21(2*imax, 1, 1)	≡	co21(imax, 2, 1)
co21(2*imax+1, 1, 1)	≡	co21(1, 3, 1)
co21(2*imax+2, 1, 1)	≡	co21(2, 3, 1)
...		...
co21(lp1*imax, 1, 1)	≡	co21(imax, lp1, 1)
co21(lp1*imax+1, 1, 1)	≡	co21(1, 1, 2)
...		...
co21(2*lp1*imax, 1, 1)	≡	co21(imax, lp1, 2)
co21(2*lp1*imax+1, 1, 1)	≡	co21(1, 1, 3)
...		...
co21(lp1*lp1*imax, 1, 1)	≡	co21(imax, lp1, lp1)

Table A.10: References to array elements which can be regarded as being equivalent. The references on the left-hand side generate array-bounds violations, while those on the right-hand side do not.

- stand-alone atmosphere model
- stand-alone ocean model
- coupled atmosphere-ocean model

Array-bounds violations: atmosphere model

Array-bounds violations were found to occur within the subroutines `ptogcray`, `clo89`, `fst88`, `e1e288`, `e3v88`, `spa88`, `phys`, `dtogcray` and `dynm`.

All of these violations were found to arise from cases where loops have been unrolled in order to optimise performance on vector architectures. An example of this, taken from the subroutine `fst88`, is as follows:

```
      do 605 i=1,imax*lp1*lp1
      co21(i,1,1)=co21(i,1,1)*over(i,1,1)
605   continue
```

Both `co21` and `over` are `real` variables with dimensions `(imax, lp1, lp1)`. Array-bounds violations arise because the first dimension of these variables only has size `imax`, while index values as large as `imax*lp1*lp1` are specified. In practice, this code fragment gives correct results on the platforms on which the source code has been run in the past. This is because the elements of each array are ordered in memory such that the references shown in Table A.10 are equivalent.

Although modifying loops in this way can enhance performance, it is also highly dangerous. Firstly, it cannot be assumed that, in general, all compilers will order the elements of each array in memory such that the above loop restructuring gives correct results. Thus, if the source code is ported to a new platform, or even if the compiler is upgraded on an existing platform, it cannot be guaranteed that the source code will give correct results.

Secondly, by generating widespread array-bounds violations within the source code, it prevents array-bounds checking from ever being conducted. As a result, if any genuine array-bounds violations are present within the source code, or if any are introduced through further development, they cannot be detected.

All loops that had been restructured in this way were therefore returned to a form which complies with the Fortran standard (Standards Association of Australia, 1983). For example, the above loop was replaced with:

```
      do 605 k = 1, lp1
        do 605 j = 1, lp1
          do 605 i = 1, imax
            co21(i, j, k) = co21(i, j, k) * over(i, j, k)
605   continue
```

The source code, before and after these changes, was compiled on the AlphaServer SC using the default compiler options of `f90 -r8 -align dcommons`. Short (eight-day) runs were conducted using both executables, and the output of the two runs was found to be bit-for-bit identical. This confirmed that the unrolling of the loops had had no effect on the model physics, and that no errors had been made during the process of removing the array-bounds violations. Timings also indicated that the modifications to the source code had had no measurable impact upon runtime performance.

The modified source code was now compiled on the AlphaServer SC with the addition of the `-check bounds` compiler option. The atmosphere model was run for one year, without any array-bounds violations being detected.

Array-bounds violations: ocean model

Array-bounds violations were found to arise within the ocean model as a result of the Ocean Direct Access Manager (ODAM), which creates a virtual disk within memory. This approach originates from the historical use of tape, rather than disk, for the storage of data, and seeks to enhance the performance of the model by holding all model variables within memory, rather than having to access them from tape.

ODAM consists of a set of routines, all of which are contained within the subroutine `odam`. The routines `oput` and `oget` copy model variables to and from the virtual disk, while the routines `ord` and `owrt` read the contents of the virtual disk from tape/disk at the start of the run, and write it to tape/disk again at the end.

Within `odam`, the virtual disk is created by declaring an array `big`, which is equivalenced to the array c thus:

```
      COMMON /BG/ BIG(NTB)
      DIMENSION A(8),C(8)
      EQUIVALENCE (BIG,C)
```

The value of the parameter `ntb` depends upon the precise configuration of the model, being equal to 689060 for the default configuration used in Mk3L.

`oput` and `oget` are both entry points to the subroutine `odam`, and are defined as follows:

```
      ENTRY OPUT(LU,NWRS,NFRST,A)
```

```
      ENTRY OGET(LU,NWRS,NFRST,A)
```

The `integer` arguments `lu`, `nwrs` and `nfrst` specify, respectively, a logical unit number, the number of words of data to transfer, and the address of the first word of the data within the virtual disk. The `real` array a contains the data to be transferred. `oget` and `oput` copy the data between the virtual disk and a, with the loops that perform this operation being, in the case of `oput`:

```
      DO 120 N=NS,NE
        C(N)=A(N-NS+1)
 120  CONTINUE
```

and, in the case of `oget`:

```
      DO 100 N=NS,NE
        A(N-NS+1)=C(N)
 100  CONTINUE
```

There are two senses in which this approach represents bad programming practices, which might lead to incorrect and unpredictable results.

Firstly, the array a is only declared as being of size 8. The subroutines `oget` and `oput` therefore create frequent array-bounds violations, with access being made to addresses in memory which lie well outside the addresses reserved for a.

Secondly, a is declared as being of `real` type. However, both `real` and `integer` data arrays are stored within the virtual disk, and hence are passed as arguments to `oput` and `oget`. This means that, in some cases, `integer` arrays are passed as arguments to ODAM, and yet the bits of data are received as though they correspond to an array of `real` values. When data is passed between `integer` and `real` variables in this fashion, the compiler does not insert code to convert it from one type to the other, meaning that the values stored within the virtual disk will be essentially meaningless.

In addition to these dangers, the use of ODAM is also inefficient, as there are considerable overheads associated with the frequent calls to `oput` and `oget`, and with the frequent copying of data between different memory addresses.

Given all of these problems, it was decided to remove ODAM completely from the model. The virtual disk was replaced with a `common` block containing the same model variables. A new header file, `ORESTART.f`, was added to the model source code, defining the `common` block `orestart` as follows:

```
      integer itt, isz(jmt, lseg), iez(jmt, lseg), isis(nisle),
     &        ieis(nisle), jsis(nisle), jeis(nisle),
     &        istf(njtbft, lsegf, km), ietf(njtbft, lsegf, km),
     &        isuf(njtbfu, lsegf, km), ieuf(njtbfu, lsegf, km),
     &        iszf(njtbfu, lsegf), iezf(njtbfu, lsegf),
     &        kmt(imt, jmt), kmu(imt, jmt)
      real ttsec, area, volume, pb(imt, jmt), p(imt, jmt),
     &     hr(imt, jmt), ptd2(imt, jmt, 2),
     &     odam_t(imt, jmt, km, nt, 2),
```

```
&     odam_u(imt, jmt, km, 2), odam_v(imt, jmt, km, 2)

 common /orestart/ itt, ttsec, area, volume, pb, p, hr, ptd2,
&                  isz, iez, isis, ieis, jsis, jeis, istf, ietf,
&                  isuf, ieuf, iszf, iezf, odam_t, odam_u, odam_v,
&                  kmt, kmu
```

In almost all cases, it was possible to modify the source code such that the variables contained within `orestart` are now accessed directly. This is in contrast to the original approach, whereby subroutines maintained local copies of the data. They had to make frequent calls to ODAM as a result, in order to update either their local copies of the data, or to update the values stored on the virtual disk.

The arrays `odam_t`, `odam_u` and `odam_v` are an exception, however. Some of the routines that perform calculations on this data process it one row at a time. Rather than holding the data for the entire globe in memory, they manipulate arrays containing the data for the row under consideration, and for the rows to either side. Previously, these arrays would have been updated through calls to ODAM. However, to have fully removed this behaviour from the model would have required extensive restructuring of the source code.

Thus, in the case of these routines, the calls to ODAM were replaced with loops which copy data to and from the arrays `odam_t`, `odam_u` and `odam_v`. For example, within the subroutine `step`,

```
CALL OGET(LABS(NDISKB),NSLAB,J*NSLAB+1,TBP)
CALL OGET(LABS(NDISK ),NSLAB,J*NSLAB+1,TP )
```

was replaced with

```
do kk = 1, nt
  do jj = 1, km
    do ii = 1, imt
      tbp(ii, jj, kk) = odam_t(ii, j+1, jj, kk, ndiskb)
      tp(ii, jj, kk) = odam_t(ii, j+1, jj, kk, ndisk)
    end do
  end do
end do
```

All ODAM routines, except for `ord` and `owrt`, could now be removed from the source code. `ord` and `owrt` were modified so as to read and write each of the variables to and from the restart file individually; for example, within `ord`,

```
READ(LO)BIG
```

was replaced with

```
read (lo) itt
read (lo) ttsec
read (lo) area
read (lo) volume
```

```
read (lo) pb
read (lo) p
read (lo) hr
read (lo) ptd2
read (lo) isz
read (lo) iez
read (lo) isis
read (lo) ieis
read (lo) jsis
read (lo) jeis
read (lo) istf
read (lo) ietf
read (lo) isuf
read (lo) ieuf
read (lo) iszf
read (lo) iezf
read (lo) odam_t
read (lo) odam_u
read (lo) odam_v
read (lo) kmt
read (lo) kmu
```

After removing ODAM, a number of array-bounds violations were found to remain. These generally arose from the fact that the arrays containing model variables are wrapped in the x-direction. Although the model grid has dimensions 64 and 56 in the x- and y-directions respectively, the model arrays have dimensions `(imt, jmt)`, where `imt` and `jmt` are equal to 66 and 58 respectively. In the y-direction, rows 1 and `jmt`-1 of each array are unused. In the x-direction, however, the data in column 1 replicates that in column `imt`-1, while the data in column `imt` replicates that in column 2.

This approach allows simpler coding to be employed when calculations are being conducted for gridboxes adjacent to the Greenwich Meridian; this can be seen from the following loop, which is taken from the subroutine `tracer`:

```
      DO 262 K=1,KM
      DO 262 I=1,IMT
        TEMPA(I,K)=FM(I-1,K)*(TB(I  ,K,M)-TB(I-1,K,M))
        TEMPB(I,K)=FM(I+1,K)*(TB(I+1,K,M)-TB(I  ,K,M))
 262  CONTINUE
```

However, this loop generates array-bounds violations. When `i` is equal to 1, references are made to `fm(0, k)` and `tb(0, k, m)`; likewise, when `i` is equal to `imt`, references are made to `fm(imt+1, k)` and `tb(imt+1, k, m)`. These elements lie outside of the declared bounds of these arrays, which are `fm(imt, km)` and `tb(imt, km, nt)` respectively.

The solution to this was to modify the loop as follows:

```
      DO K=1,KM
```

```
      DO I=2,IMTM1
        TEMPA(I,K)=FM(I-1,K)*(TB(I  ,K,M)-TB(I-1,K,M))
        TEMPB(I,K)=FM(I+1,K)*(TB(I+1,K,M)-TB(I  ,K,M))
      END DO
      TEMPA(1,K)=TEMPA(IMTM1,K)
      TEMPB(1,K)=TEMPB(IMTM1,K)
      TEMPA(IMT,K)=TEMPA(2,K)
      TEMPB(IMT,K)=TEMPB(2,K)
    END DO
```

`imtm1` is a model parameter which is equal to `imt`-1. The modified loop contains no array-bounds violations, while maintaining the wrapping of model variables in the x-direction.

After making these changes, the only remaining array-bounds violations were found to occur in the subroutine `matrix`, which is a diagnostic plotting routine, and in two calls to `matrix` which were located within the subroutine `tracer`. `matrix` was modified accordingly, and the offending calls within `tracer` were deleted.

The source code, before and after these changes, was compiled on the AlphaServer SC using the default compiler options of `f90 -r8 -align dcommons`. Short (one-month) runs were conducted using both executables, and the output of the two runs was found to be bit-for-bit identical. This confirmed that the array-bounds violations which had been detected had no effect on the model physics, and that no errors had been made during their removal.

The modified source code was now compiled on the AlphaServer SC with the addition of the `-check bounds` compiler option. The ocean model was run for one year, without any array-bounds violations being detected.

Array-bounds violations: coupled model

The modified source code was now compiled on the AlphaServer SC with the addition of the `-check bounds` compiler option. The coupled model was run for one year, without any array-bounds violations being detected.

A.4.6 Conversion of restart files

Binary format

The original restart files supplied by CSIRO had been generated on a CRAY machine. Before these files could be used to initialise the model on either the AlphaServer SC or Linux Cluster, they required modification in two ways:

- conversion from CRAY binary format to little-endian
- conversion of `integer` variables from 64- to 32-bit precision

Two Fortran 90 programs, `convert_atmos_mk3` and `convert_ocean`, were written to perform the first of these steps, on the atmosphere and ocean model restart files respectively. These programs were com-

piled and run on the AlphaServer SC; they read the data from the original restart files, specifying the keyword `convert='cray'` when using the `open` command to open the files, and immediately write it again to a new file in native (little-endian) format. The Linux Cluster shares the same native format, and hence restart files could be used interchangeably between the two machines. The Fortran 90 programs `convert_atmos_mk3_i8_to_i4` and `convert_ocean_i8_to_i4` were then written to perform the second step, using the `int` command to convert `integer` variables from 64- to 32-bit precision.

The coupled model also requires a restart file, containing the surface fluxes to be passed to the ocean model at the first timestep (Section 5.2). The Fortran 90 program `convert_flux` was written to convert this to little-endian format; the coupled model restart file does not contain any `integer` variables.

Structure

In removing ODAM from the ocean model (Section A.4.5), the structure of the restart files was changed completely. The original version of the model generates restart files by simply writing a section of memory directly to disk, whereas Mk3L writes each of the variables individually.

A Fortran 90 program, `convert_ocean_to_new`, was therefore written, which reads a restart file generated by the original version of the model, and writes the data to a restart file which can be used to initialise Mk3L.

A.4.7 Conversion of auxiliary files

Some of the auxiliary files required by the model are also in a machine-dependent binary format, as for the restart files (Section 5.2).

The Fortran 90 program `convert_tscorr` was written, which converts the coupled model auxiliary files `t1coravth`, `s1coravth` and `dtm1av` from CRAY to little-endian format. The program `convert_fcorr` was also written, which converts the coupled model auxiliary files `cwice.dat12`, `hfcor.dat12`, `sfcor.dat12`, `txcor.dat12` and `tycor.dat12` to little-endian format.

In order to run the ocean model, the Fortran 90 program `convert_bsnmask` was written, which converts the ocean model auxiliary file `bsnmask.cif` from CRAY to little-endian format.

A.5 Overview of code optimisation

Mk3L was designed to enable climate variability and change to be studied on millennial timescales. However, even at the coarse spatial resolution employed within the model, such simulations take months to years to complete on current computing facilities. Given these timescales, a certain amount of time was invested in optimising runtime performance.

This section outlines the approach taken to improve the performance of the model. The optimisations which were performed are discussed in Sections A.6 and A.7, with the exception of those optimisations which relate to input and output; these are discussed in Sections B.3 and C.3.

A.5.1 Serial and parallel optimisation

The optimisations which were performed on the source code can be divided into two categories: serial optimisation and parallel optimisation.

Serial optimisation aims to maximise the performance on a single processor. In order to achieve this, the following tasks were performed:

- the amount of I/O (input/output) was minimised
- redundant source code was removed
- profiling was used to identify the most time-consuming routines, which then became candidates for re-coding
- loops were restructured for greater efficiency
- benchmarking was used to determine the optimal compiler options

Parallel optimisation aims to maximise the performance on multiple processors, which can enable simulations to be completed in a small fraction of the time that would be required on a single processor. In order to achieve this, the following tasks were performed:

- profiling was used to identify the most time-consuming serial routines, which then became candidates for parallelisation
- additional OpenMP directives were inserted
- an auto-parallelising compiler was used

A.5.2 Coarse-grain and fine-grain parallelism

Parallelism can be regarded as being either *coarse-grain* or *fine-grain*. Coarse-grain parallelism represents parallelisation at the program level, typically involving the parallelisation of loops which include calls to other subroutines. This approach ensures that a very large amount of work is included within each iteration of the parallel loop, minimising the overheads associated with parallel execution. Such parallelisation can only be specified manually through the insertion of compiler directives, as it is beyond the scope of auto-parallelising compilers (Section A.2.4).

Fine-grain parallelism represents parallelisation at the level of individual loops. This will only give improved performance on multiple processors if the loop contains a relatively large amount of work; otherwise, the overheads associated with parallel execution will more than offset any performance gain arising from parallelisation of the loop. Fine-grain parallelisation can be specified manually through the insertion of compiler directives, but can also be achieved through the use of auto-parallelising compilers.

A.5.3 The atmosphere and ocean models

A high-degree of coarse-grain parallelism has already been specified within the atmosphere model (Rotstayn and Dix, 1992). This is possible because of the structure of the model, in which all the physical and

dynamical calculations for each latitude band can be conducted independently. Hence parallelism can be achieved at the timestep level, with parallelisation of both the physics loop (contained within the subroutine `phys`, and containing all the physical calculations) and the dynamics loop (contained within the subroutine `dynm`, and containing all the dynamical calculations).

The structure of the ocean model, however, precludes the specification of such coarse-grain parallelism. This is because of the rigid-lid boundary condition (Section 2.3), which requires the calculation of line integrals around each island. If large islands are present, the calculation of the integrals requires knowledge of model variables over a wide range of latitudes, precluding parallelisation in the meridional direction.

In order to achieve good parallel performance for the model as a whole, the ocean model therefore requires particular attention. Fortunately, it is much simpler than the atmosphere model. In the original version of the model, it consists of 30 source files, containing 10,609 lines of code, and it therefore accounts for only $\sim$12% of the model source code. It was therefore the focus of efforts to achieve both serial and parallel optimisation.

A.6 Serial optimisation

AlphaServer SC-specific version

Although it was the intention to produce a completely platform-independent version of the model, the initial research which was conducted using Mk3L took place on the AlphaServer SC. Therefore, some slight modifications were made to the source code in order to obtain the best possible performance on this particular machine. These changes related solely to the use of the CXML (Hewlett-Packard Company, 2005) matrix manipulation routines `dgemm`, `dgetrf` and `dgetri`, rather than the equivalent routines, `matinv` and `mmtx`, which are included within the model source code.

Changes were made to four subroutines: `ftospec`, `ftospec2`, `ftospec3` and `matset`. Preprocessor directives were inserted, ensuring that only the relevant sections of source code are compiled on each machine. For example, within `matset`,

```
        call matinv(uk2i,nl,nl,wrk1,0,d,ierr,wrk2,wrk3)
        call matinv(uk2di,nl,nl,wrk1,0,d,ierr,wrk2,wrk3)
```

was replaced with

```
#ifdef ALPH
        call dgetrf(nl, nl, uk2i, nl, ipiv, info)
        call dgetri(nl, uk2i, nl, ipiv, work, lwork, info)
        call dgetrf(nl, nl, uk2di, nl, ipiv, info)
        call dgetri(nl, uk2di, nl, ipiv, work, lwork, info)
#else
        call matinv(uk2i,nl,nl,wrk1,0,d,ierr,wrk2,wrk3)
        call matinv(uk2di,nl,nl,wrk1,0,d,ierr,wrk2,wrk3)
#endif
```

Routine	Description
atgrd	Subroutine
avgr	Subroutine
clinic1	Subroutine
dawrt	Subroutine
icewind	Subroutine
intgr	Subroutine
ocsave2	Subroutine
ocwrite	Subroutine
setz	Subroutine
step1	Entry point to subroutine step0
step2	Entry point to subroutine step0
step3	Entry point to subroutine step0
step5	Entry point to subroutine step0
tatgrd	Subroutine
tracer1	Main body of subroutine tracer1
tracer2	Entry point to subroutine tracer1
tracer3	Entry point to subroutine tracer1
tracer8	Entry point to subroutine tracer1
tracer9	Entry point to subroutine tracer1
tswrt	Subroutine
uatgrd	Subroutine
zatht	Subroutine
zheat	subroutine

Table A.11: Ocean model routines which were removed from the source code.

The use of preprocessor directives required that the compiler options `-fpp -DALPH` be specified when compiling on the AlphaServer SC, and that the option `-fpp` (or whichever option invokes the Fortran pre-processor) be specified when compiling on all other machines.

A.6.1 Ocean model

Removing redundant source code

A new header file, `OPARAMS.f`, was created, which contains all the global parameters for the ocean model. These had previously been defined in a section of code which was repeated at the beginning of every routine. By moving the definitions to a header file, it was possible to simplify the code considerably, remove a potential source of errors, and make it much easier to change parameter values.

Inspection of the source code then identified large sections which were redundant, generating data which is never used. This code, which generally produced diagnostic data which is no longer saved, was removed. As a result, a number of entire routines could be removed (Table A.11), as well as the header file `GEGST.f`.

Subroutine	% of total execution time	Time per call (ms)
tracer	51.9	1.38
ocstep	17.3	27.11
clinic	14.1	0.38
statec	7.2	0.06
dencal	2.6	0.07
filter	2.2	0.003
state	2.1	0.06
relax	1.8	2.73
stinit	0.3	0.37
tracri	0.2	2.14
ocend	0.2	0.29

Table A.12: Ocean model routines which account for more than 0.05% of the total execution time.

Profiling

`gprof` was used to profile the ocean model on the AlphaServer SC. If the model was compiled with the addition of the `-pg` option, then the compiler inserted additional instructions which counted the number of times that each routine was called, and determined the amount of time spent in that routine.

When the model was executed, the file `gmon.out` was produced, which could then be interpreted by the `gprof` command as follows:

```
gprof ./model gmon.out > model.prof
```

In this example, `./model` is the executable that has been produced with the addition of the `-pg` compiler option, and the file `model.prof` contains an execution profile. The profile gives the total number of times that each routine is called, the total amount of time spent in that routine, and the average amount of time spent in each routine per call.

The model was compiled with the addition of the `-pg` option, and the stand-alone ocean model was run for one month. Through the use of `gprof`, an execution profile was obtained. From this profile, it was determined that only 11 routines account for more than 0.05% of the total execution time each; these are listed in Table A.12. `ocstep` is an entry point to the subroutine `step`, `statec` and `stinit` are entry points to the subroutine `state`, and `tracri` is an entry point to the subroutine `tracer`.

The subroutines `tracer`, `step` and `clinic` account for 83.5% of the total execution time; these three routines were therefore the focus of attempts to optimise the ocean model.

Loop optimisation

In order to optimise the performance of the subroutines `tracer`, `ocstep` and `clinic`, adjacent loops were fused wherever possible. For example, within `tracer`,

```
      do k=1,km
```

```
      do i=1,imt
        utm(i,k) = -fxb*dz2rq(i,k)*(aryrz(i,k)-aryrz(i,k+1))
        vtm(i,k) =  fxb*dz2rq(i,k)*(arxrz(i,k)-arxrz(i,k+1))
      enddo
      enddo

      do k=1,km
      do i=1,imt
        uedd(i,k) = uedd(i,k) + utm(i,k)
        vedd(i,k) = vedd(i,k) + vtm(i,k)
      enddo
      enddo
```

was replaced with

```
      do k = 1, km
        do i = 1, imt
          utm(i,k) = -fxb*dz2rq(i,k)*(aryrz(i,k)-aryrz(i,k+1))
          vtm(i,k) =  fxb*dz2rq(i,k)*(arxrz(i,k)-arxrz(i,k+1))
          uedd(i,k) = uedd(i,k) + utm(i,k)
          vedd(i,k) = vedd(i,k) + vtm(i,k)
        end do
      end do
```

However, these efforts were hampered by a number of `if` constructs relating to the parameter `igm`, the value of which is specified in the model control file. If it is equal to 1, then Gent-McWilliams eddy diffusion (Gent and McWilliams, 1990) is employed; otherwise, horizontal diffusion is employed. Large numbers of `if` constructs impede the runtime performance of the model, not least by restricting the ability of the compiler to optimise the source code.

It was decided to hard-code the use of the Gent-McWilliams eddy diffusion scheme into Mk3L. This required modifications to the subroutines `tracer`, `step`, `ocean` and `ocend`, and enabled the removal of all of the relevant `if` constructs. For example, within `tracer`,

```
      if(igm.eq.1)then

      DO K=1,KM
      DO I=1,IMT
        TEMPA(I,K)=FUW(I,K)*(T(I,K,M)+T(I-1,K,M))
     *                + fuwedd(i,k)*(t(i,k,m)+t(i-1,k,m))
      ENDDO
      ENDDO

      else

      DO 810 K=1,KM
```

File	Description
airsea.f	Contains the subroutine airsea
dwrt.f	Contains the subroutine dwrt
establ.f	Contains the subroutine establ
fname.f	Contains the function fname
xtconv.f	Contains the subroutine xtconv

Table A.13: Source files included with the original model source code, but not compiled.

```
      DO 810 I=1,IMT
        TEMPA(I,K)=FUW(I,K)*(T(I,K,M)+T(I-1,K,M))
 810  CONTINUE

      endif
```

could be replaced with

```
      DO K=1,KM
      DO I=1,IMT
        TEMPA(I,K)=FUW(I,K)*(T(I,K,M)+T(I-1,K,M))
     *                 + fuwedd(i,k)*(t(i,k,m)+t(i-1,k,m))
      ENDDO
      ENDDO
```

This enabled a number of additional loops to be fused.

The source code, before and after these changes, was compiled on the AlphaServer SC, using the default compiler options of `f90 -r8 -align dcommons`. Short (one-month) runs were conducted using both executables, and the output of the two runs was found to be bit-for-bit identical, confirming that no errors had been made.

Compiling the modified code on the AlphaServer SC using the optimal compiler options (Section A.6.3), it was found that a performance gain of $\sim$6% on one processor had been achieved.

A.6.2 Coupled model

Five source files were included with the model source code, but not compiled into the model (Table A.13). These were removed.

Another four source files were found to be compiled into the model, but to contain subroutines which were never called (Table A.14). These files were also removed.

A.6.3 Compilation for serial execution

AlphaServer SC

The minimum compiler options required for serial execution were:

File	Description
qnegfix.f	Contains the subroutine qnegfix
sclset.f	Contains the subroutine sclset
timer.f	Contains the subroutine timer
trim3.f	Contains the function trim3

Table A.14: Source files compiled into the original version of the model, but never called.

```
f90 -align dcommons -r8 -fpp DALPH -warn nouncalled
```

The additional options `-math_library fast`, `-fp_reorder`, `-arch host` and `-tune host` were found to be optimal, improving runtime performance by ∼12%. The `-math_library fast` option specifies that, for certain mathematical functions, the compiler select the version of the library routine which provides optimal runtime performance. The `-fp_reorder` option instructs the compiler to reorder floating-point operations for improved performance, while the `-arch host` and `-tune host` options instruct the compiler to generate instructions for the host architecture, and to tune them for the host architecture, respectively.

Thus, for serial execution on the AlphaServer SC, the model was compiled using:

```
f90 -math_library fast -fp_reorder -arch host -tune host -align dcommons -r8
    -fpp DALPH -warn nouncalled
```

Linux Cluster

The minimum compiler options required were:

```
ifort -align dcommons -r8 -fpp -warn nouncalled
```

The additional options `-O3 -xW` were found to be optimal, improving runtime performance by ∼21%. The `-O3` option selects the highest level of general optimisation, while the `-xW` option instructs the compiler to optimise the code for Intel Pentium 4 processors.

Thus, for execution on the Linux Cluster, the model was compiled using:

```
ifort -O3 -xW -align dcommons -r8 -fpp -warn nouncalled
```

A.7 Parallel optimisation

A.7.1 Atmosphere model

Profiling

The model was compiled on the AlphaServer SC with the addition of the `-pg` compiler option, and the stand-alone atmosphere model was run for eight days. Through the use of `gprof`, an execution profile was

Subroutine	% of total execution time	Time per call (ms)
enforce_conq	2.0	4.65
cavit	0.5	0.11
jmcgslt	0.4	3.04
assel	0.4	2.68
uvreal	0.3	1.90
uvharm	0.2	1.05
diffn	0.1	0.70
icedrive	0.1	1.35
dynice	0.1	1.34
advect	0.1	0.08

Table A.15: Atmosphere model subroutines which are executed in serial, and which account for more than 0.05% of the total execution time.

obtained. From this profile, it was estimated that 95.6% of the work was contained within parallel regions. Of the subroutines containing the remaining 4.4% of the work, Table A.15 shows the ten routines which were identified as accounting for more than 0.05% of the total execution time each.

Manual parallelisation

Each of these subroutines was inspected in turn, and parallelised wherever possible through the insertion of appropriate OpenMP directives.

`enforce_conq`

This subroutine consists essentially of three loops, each of which could easily be parallelised. However, the first two loops calculate global integrals, which would require that the variables containing the sums be declared as having the `REDUCTION` data scope. For example, the first loop, and the OpenMP directives which would enable it to be executed in parallel, are as follows:

```
!$OMP  PARALLEL DO
!$OMP& PRIVATE   (k, lg, lgns, ma, mg, ns, rhf)
!$OMP& REDUCTION (+ : delqneg, delqpos)
!$OMP& SHARED    (delq, wdsk)

      do k=1,nl
        do ns=1,2
          do lg=1,lat
            lgns=lg*(ns-1)+(lat2p-lg)*(2-ns)
            do mg=1,lon
              ma=mg+(ns-1)*lon
              rhf=max(0.0,delq(ma,lg,k))
              delqpos=delqpos+     rhf *wdsk(mg,k,lgns)
```

```
            delqneg=delqneg+(delq(ma,lg,k)-rhf)*wdsk(mg,k,lgns)
          enddo
        enddo
      enddo

!$OMP END PARALLEL DO
```

The REDUCTION clause indicates that each thread should hold a private copy of the variables delqneg and delqpos and that, at the end of the parallel region, the values of delqneg and delqpos should be set equal to the sum of each of these private copies.

However, because of rounding errors associated with the calculation of the sums, the precise values of delqneg and delqpos will depend not only on which threads execute which iterations of the loop, but also on which threads complete their share of the work first. Thus, even if the loop is executed twice on the same machine, with each of the variables containing exactly the same data at the start of the parallel region, the final values of delqneg and delqpos can still differ in the least significant bits.

One of the stated aims in developing Mk3L (Section A.4) was that the output of the model should be fully reproducible. This precludes the use of the REDUCTION clause, meaning that it was not possible to parallelise the first two loops in enforce_conq. It was straightforward, however, to parallelise the third loop, as follows:

```
!$OMP  PARALLEL DO
!$OMP& PRIVATE (delqq, facm, k, lg, lgns, ma, mg, ns, rgtx)
!$OMP& SHARED  (alphax, delq, fact, muf, ncepx, ncons, qq, qqm)

      do lg=1,lat
        do ns=1,2
          lgns=lg*(ns-1)+(lat2p-lg)*(2-ns)
          do k=1,nl
            do mg=1,lon
              ma=mg+(ns-1)*lon
              facm=1.0
              if(delq(ma,lg,k).gt.0)facm=alphax
              delqq=facm*delq(ma,lg,k)*fact(ncons)
              rgtx=qqm(ma,k,lg)+delqq/muf(mg,k,lgns)
              if(rgtx.lt.1.0e-20)rgtx=0.0
              if(.not.ncepx)
     &        qqm(ma,k,lg)=asfx*qq(ma,k,lg)+
     &             asf*qqm(ma,k,lg)
              qq(ma,k,lg)=rgtx
            enddo
          enddo
        enddo
      enddo
```

```
!$OMP END PARALLEL DO
```

This was found to give a speed gain on multiple processors.

cavit

This subroutine contains only a small amount of work, accounting for just 0.11ms of execution contain per call. Furthermore, this work is split between nine loops.

By restructuring the code, it was possible to reduce the number of loops to just five, which were then parallelised. However, it was found that the model now ran more slowly on multiple processors than previously, indicating that there is insufficient work contained within the loops to compensate for the overheads associated with parallelisation.

The restructuring of the routine was retained, as this was found to reduce the serial execution time from 0.11ms to 0.09ms per call. However, the OpenMP directives were removed.

jmcgslt

This subroutine contains a large number of loops. However, for the default configuration of Mk3L, only five of these are executed. It was possible to merge two of these, leaving just four to parallelise. One of these loops calculates a global integral, and hence could not be parallelised without use of the `REDUCTION` clause. However, the other three loops were parallelised.

This was found to give a speed gain on multiple processors.

assel

This subroutine contains four loops. Two of these were merged, and each of the remaining three loops was parallelised.

This was found to give a speed gain on multiple processors.

uvreal

This subroutine consists of two short loops. One of these loops calculates a global integral, and hence could not be parallelised without use of the `REDUCTION` clause. However, the other loop was parallelised.

This was found to give a speed gain on multiple processors.

uvharm

This subroutine contains three loops, one of which contains a negligible amount of work. The other two loops were parallelised.

This was found to give a speed gain on multiple processors.

```
diffn, icedrive, dynice, advect
```

None of these subroutines contain any large loops, and hence they were not candidates for parallelisation.

Profiling the modified code

The modified source code was compiled on the AlphaServer SC with the addition of the `-pg` option, and the stand-alone atmosphere model was run for one month. Through the use of `gprof`, a fresh execution profile was obtained.

By assuming, as a best estimate, that $\frac{1}{3}$, $\frac{3}{4}$ and $\frac{1}{2}$ of the work within the subroutines `enforce_conq`, `jmcgslt` and `uvreal` respectively is executed in parallel, it was estimated that 97.5% of the work within the atmosphere model is now executed in parallel.

OpenMP scheduling

OpenMP supports four different scheduling types, which are specified using the `SCHEDULE` clause. These types are `STATIC`, `DYNAMIC`, `GUIDED` and `RUNTIME`. A chunk size may also be specified, although this feature is not used; the following discussion of the scheduling types only discusses the behaviour in the absence of a chunk size being specified.

In `STATIC` scheduling, which is the default if no explicit type is specified, the iterations of each loop are shared equally between each thread, and are dispatched to the threads before execution of the loop begins.

In `DYNAMIC` scheduling, the iterations of each loop are passed to the threads one at a time. As each thread completes an iteration, the next iteration is dispatched.

In the absence of a chunk size being specified, `GUIDED` is equivalent to `DYNAMIC`.

In `RUNTIME` scheduling, the schedule type is specified at runtime via the `OMP_SCHEDULE` environment variable.

Benchmarking of the atmosphere model on the AlphaServer SC revealed that `DYNAMIC` scheduling gave significantly better performance than `STATIC` scheduling, with a speed gain of $\sim$10% on four processors. This performance gain can be attributed to the fact that the parallelism specified within the atmosphere model is generally very coarse-grain, with a large amount of work being contained within each iteration of the parallelised loops. Hence the improved load balancing obtained through the use of `DYNAMIC` scheduling outweighs any additional overheads incurred.

The `SCHEDULE(DYNAMIC)` clause was therefore added to the OpenMP directives at the start of each parallel region e.g.

```
!$OMP  PARALLEL DO
```

was replaced with

```
!$OMP  PARALLEL DO SCHEDULE(DYNAMIC)
```

A.7.2 Ocean model

Manual parallelisation

The three subroutines which account for the bulk of total execution time, `tracer`, `step` and `clinic`, are all lengthy routines, consisting of a large number of loops. As such, none were candidates for manual parallelisation.

Although `statec` only accounts for 0.06ms of execution time per call, it accounts for 7.2% of total execution time, and consists of a single loop. It was therefore parallelised through the insertion of OpenMP directives. However, this was found to lead to diminished performance, with the model now running more slowly on multiple processors on the AlphaServer SC than on a single processor.

Therefore, there appeared to be no scope for manual parallelisation.

Automatic parallelisation

An auto-parallelising Fortran compiler was available on the AlphaServer SC. The KAP Fortran/OpenMP optimiser (Hewlett-Packard Development Company, 2005b) is a source-to-source preprocessor which optimises Fortran source code and performs automatic parallelisation, inserting OpenMP directives in order to enable parallel execution on shared-memory systems.

The simplest way to use KAP Fortran/OpenMP on the AlphaServer SC was to use the `kf90` command, as follows:

```
kf90 -fkapargs='-concurrent -unroll=1' -c -r8 sample.f
```

This command passes the options `-conc -unroll=1` to the `kapf90` source translator, which produces an optimised and parallelised source file with the name `sample.cmp.f`. The `f90` Fortran compiler is then invoked with the options `-c -r8`, which compiles the source file `sample.cmp.f` and produces an object file, `sample.o`.

To enable individual subroutines to be compiled using `kf90`, it was necessary to split up some of the source files, as they contained multiple subroutines. Hence `tracer.f` was split into `tracer.f` and `tracer1.f`, with these files containing the subroutines `tracer` and `tracer1` respectively. Likewise, `step.f` was split into `ocdatra.f`, `ocdatro.f`, `r2fcfld.f`, `step.f` and `step0.f`.

Beginning with the subroutines which account for the largest share of total execution time (Table A.12), `kf90` was used on each of these in turn. It was found that performance on multiple processors was improved when `kf90` was used on each of the five most time-consuming subroutines i.e. `tracer`, `step`, `clinic`, `state` and `dencal`. The other routines contain insufficient work, and parallelising these led to diminished performance.

The optimal options to pass to the `kapf90` source translator were found to be

```
-concurrent -unroll=1 -scalaropt=2 -roundoff=0 -fuse -fuselevel=1
-minconcurrent=2000
```

The `-concurrent` option directs `kapf90` to restructure the source code for parallel processing, through the insertion of OpenMP directives.

The `-unroll=1` option disables loop unrolling: better performance was obtained when this particular optimisation was left to the `f90` Fortran compiler.

The `-scalaropt=2` option specifies the level of scalar optimisation to be performed on the source code. The level can be either 0, 1, 2, 3; level 2 was found to give optimal performance.

The `-roundoff=0` option specifies the level of roundoff error that can be tolerated. The level can be either 0, 1, 2, 3. Level 0 disables all optimisations that can change the results of arithmetic operations, while higher levels enable increasingly aggressive optimisations. These include the recognition of arithmetic reduction operations (Section A.7.1). Only level 0 was found to give acceptable results; higher levels were found to lead to irreproducible results on multiple processors.

The `-fuse` option instructs `kapf90` to perform loop fusion, whereby two adjacent loops are combined into a single loop. The `-fuselevel=1` option instructs `kapf90` to move non-adjacent loops to adjacent positions, subject to a data dependence test. Loop fusion can then be attempted on loops that could not be combined otherwise.

The `-minconcurrent` option sets the minimum amount of work (in arbitrary units) that must be contained within a loop before `kapf90` executes it in parallel. The default amount is 1000, but specifiying `-minconcurrent=2000` was found to give optimal performance.

By default, `kf90` passes the following options to the `f90` Fortran compiler:

```
-fast -automatic -omp -tune host -call_shared
```

The `-automatic` option instructs the compiler to place local variables on the runtime stack, the `-omp` option enables shared-memory parallel processing, as directed through the insertion of OpenMP directives (Section A.4.2), and the `-tune host` option instructs the compiler to tune the code for the host architecture. The `-call_shared` option instructs the compiler to link against shared libraries; however, `kf90` was only being used to generate object files, and hence this option was irrelevant.

The `-fast` option is equivalent to

```
-align dcommons -align sequence -arch host -assume bigarrays -assume nozsize
-fp_reorder -math_library fast -tune host
```

Of these options, `-align dcommons` was essential (Section A.4.4), while it had already been established that the `-arch host`, `-fp_reorder`, `-math_library fast` and `-tune host` options gave optimal performance (Section A.6.3). The remaining options had no effect in this case.

The options used with the `kf90` command were therefore

```
kf90 -fkapargs='-concurrent -unroll=1 -scalaropt=2 -roundoff=0 -fuse
     -fuselevel=1 -minconcurrent=2000' -no_fp_reorder -r8
```

A.7.3 Compilation for parallel execution

AlphaServer SC

For parallel execution on the AlphaServer SC, the subroutines `clinic`, `dencal`, `state`, `step` and `tracer` were compiled using

Contents	Number of files	Total number of lines
Fortran 77 source code	195	81,256
Fortran 90 source code	4	682
C source code	1	34
Header files	91	3,449
Makefile	1	190
Total	**292**	**85,611**

Table A.16: A summary of the Mk3L source code.

```
kf90 -fkapargs='-concurrent -unroll=1 -scalaropt=2 -roundoff=0 -fuse
     -fuselevel=1 -minconcurrent=2000' -no_fp_reorder -r8
```

The remainder of the source code was compiled using

```
f90 -omp -math_library fast -fp_reorder -arch host -tune host -align dcommons
    -r8 -fpp DALPH -warn nouncalled
```

A.8 Modified source code

A summary of the final Mk3L source code, after the modifications documented within this appendix, and within Appendices B, C and D, is shown in Table A.16. The number of source files has been reduced from 236 to 200, and the total length of the source code has been reduced from 90,371 to 85,611 lines.

A.9 Benchmarks

The final version of the source code was benchmarked on the following machines (see Section A.3 for descriptions of the hardware):

- APAC AlphaServer SC
- APAC Linux Cluster

Timings were obtained for each of the four different modes in which the model was to be run:

- stand-alone ocean model (asynchronous timestepping)
- stand-alone ocean model (synchronous timestepping)
- stand-alone atmosphere model
- coupled atmosphere-ocean model

The durations of the benchmarking runs, and the timestep durations employed within the atmosphere and ocean models, are shown in Table A.17.

Mode		Timestep duration			Duration of run
		Atmosphere	Ocean		
			T, S	ψ, u, v	
Ocean model	Asynchronous		1 day	20 minutes	10 years
	Synchronous		1 hour	1 hour	1 year
Atmosphere model		20 minutes			1 month
Coupled model		20 minutes	1 hour	1 hour	1 month

Table A.17: The configuration of the model for benchmarking runs.

Machine	Speed (years/day)			Speed gain		
	Number of processors			Number of processors		
	1	2	4	1	2	4
Ocean model (asynchronous timestepping)						
AlphaServer SC	1560	2080	2440	1.00	1.33	1.56
Linux Cluster	1850					
Ocean model (synchronous timestepping)						
AlphaServer SC	70	94	112	1.00	1.35	1.60
Linux Cluster	82					
Atmosphere model						
AlphaServer SC	4.4	7.9	13.4	1.00	1.80	3.07
Linux Cluster	4.9					
Coupled model						
AlphaServer SC	4.0	7.2	11.7	1.00	1.79	2.89
Linux Cluster	4.6					

Table A.18: Benchmarks for Mk3L.

On both the Linux Cluster and the AlphaServer SC, the model was compiled for serial execution and benchmarked on one processor. On the AlphaServer SC, the model was also compiled for parallel execution and benchmarked on two and four processors. Each benchmarking run was conducted three times, with the time of the fastest run being used. As well as enabling more accurate timings to be obtained, this also made it possible to check that the output of the model was reproducible (Section A.10).

In order to avoid any unnecessary overheads, the model was always run on a local filesystem, rather than one which was mounted via a network. In the case of both the AlphaServer SC and Linux Cluster, this meant that the model was run on the `/jobfs` filesystem. Furthermore, an entire node was reserved for each of the runs. While nodes on the Linux Cluster consisted of a single processor, on the AlphaServer SC this meant that an entire four-processor node was reserved for each run, irrespective of the number of processors being used.

Timings were obtained using the `timex` command on the AlphaServer SC, and the `time` command on the Linux Cluster. The times for each run were then converted into an equivalent speed, in model years per day of walltime. The resulting benchmarks are shown in Table A.18.

The performance of the model, even on one processor, is consistent with the completion of millennial-scale climate simulations on a realistic timescale. Assuming unrestricted access to processors, the serial version of the coupled model could be used to complete a one thousand-year simulation in $\sim$250 days on the AlphaServer SC, and in $\sim$220 days on the Linux Cluster. The parallel version could be used to complete such a simulation in just $\sim$85 days on the AlphaServer SC.

The performance on multiple processors is largely a consequence of the very high degree of parallelism achieved within the atmosphere model. On the AlphaServer SC, there was a speed gain of 3.07 on four processors, representing a parallel efficiency of 77%. This is less, however, than the theoretical maximum speed gain. Amdahl's Law (Amdahl, 1967) states that if a fraction f of a calculation is sequential, and hence that the fraction $(1-f)$ is conducted in parallel, then the maximum speed gain S that can be achieved by using N processors is

$$S = \frac{1}{f + \frac{(1-f)}{N}} \tag{A.2}$$

As N tends towards infinity, the theoretical maximum speed gain tends asymptotically towards $\frac{1}{f}$. The rationale behind Amdahl's Law is straightforward: the time taken to complete the serial fraction f of the calculation will be independent of the number of processors, while, assuming that there are no overheads associated with parallel execution, the time taken to complete the parallel fraction $(1-f)$ will vary as the inverse of the number of processors.

For the atmosphere model, it was estimated that 97.5% of the work is executed in parallel (Section A.7.1), giving a value for f of 0.025. Thus the thereotical maximum speed gain is 1.95 on two processors, and 3.72 on four processors. The speed gains that have been achieved in practice are considerably smaller, indicating the magnitude of the overheads associated with parallel execution.

For the ocean model, there is no straightforward way of determining the extent to which the code has been parallelised. However, Amdahl's Law can be used to determine a lower limit. Re-arranging Equation A.2, f can be obtained as a function of S:

$$f = \frac{\frac{N}{S} - 1}{N - 1} \tag{A.3}$$

The speed gains achieved on two processors on the AlphaServer SC give upper limits on f of 0.50 and 0.48, for the asynchronous and synchronous ocean models respectively. Under asycnchronous timestepping, *at least* 50% of the work is therefore executed in parallel, while, under synchronous timestepping, *at least* 52% is executed in parallel. This is respectable, given that only fine-grain parallelism is possible, and given that the auto-parallelising compiler is being applied to only five subroutines.

A.10 Reproducibility

One of the stated aims in developing Mk3L (Section A.4) was that the output of the model should be fully reproducible, i.e. that if a simulation is conducted twice using the same executable, the output of the two runs should be bit-for-bit identical.

Reproducibility is strongly desirable, primarily because it enables some or all of a simulation to be repeated at a later date. This could be necessary if, for example, some of the output from the original simulation is

lost, or if it is desired to extract additional variables.

Reproducibility also acts as a check against certain types of errors within the source code. Array-bounds violations might cause the output of the model to be irreproducible; this could also occur if variables that are referenced within parallel regions are declared as having the incorrect scope. If the output of the model is fully reproducible, this provides confidence that such errors are not present.

It should be noted that different compilers will perform different optimisations on the source code, and that these optimisations can be controlled through the specification of compiler options (e.g. Section A.6.3). As a result of the rounding errors associated with arithmetic operations, many compiler optimisations will affect the output of the model. If two different compilers are used to compile identical source code, it is exceptionally unlikely that the two executables will give identical output.

It should therefore be emphasised that reproducibility can only expected to apply to simulations conducted using a given executable. This means that it should be possible to repeat a simulation on another machine of the same architecture, but that it should not be expected to be able to repeat a simulation on a machine of a different architecture.

A.10.1 AlphaServer SC

For each of the modes in which benchmarking runs were conducted (Section A.9), the output of all the runs was bit-for-bit identical, irrespective of the number of processors on which the runs were conducted.

A.10.2 Linux Cluster

For each of the modes in which benchmarking runs were conducted (Section A.9), the output of all the runs was bit-for-bit identical.

A.10.3 Summary

The output of the model appears to be fully reproducible, as intended.

A.11 Scope for further optimisation

Once the work documented within this appendix had been completed, it was considered that the point of diminishing returns had been reached. Specifically, it was considered that the model had been sufficiently highly optimised that any potential further performance gains would not justify the investment of time that would be required in order to achieve them.

Nonetheless, ideas for possible further optimisations include:

- the use of MPI
- further use of preprocessor directives
- upgrading to version 3.x of FFTW

- manual loop restructuring within the atmosphere model
- the use of “masking” within the ocean model

A.11.1 MPI

Shared-memory parallelism is currently achieved within the model through the use of OpenMP directives. However, this restricts parallel execution to machines on which memory is shared between multiple processors. This is a major restriction, given that supercomputing facilities are increasingly tending towards a cluster architecture, in which there is no sharing of memory between processors at all (e.g. the APAC Linux Cluster), or in which memory is only shared between small numbers of processors (e.g. the APAC AlphaServer SC, which was composed of four-processor nodes, with memory only shared between the processors on each node).

This restriction can be overcome through the use of MPI (Message Passing Interface, MPI Forum, 2005), a library which enables parallel execution on distributed-memory systems, such as clusters. However, to convert the model to MPI would require extensive restructuring of the code, and would involve an enormous investment of time. Given the limitations on parallel performance which result from the inability to achieve coarse-grain parallelism within the ocean model (Section A.5), this investment of time would not be justified.

A.11.2 Preprocessor directives

Preprocessor directives have already been introduced into the source code, in order to improve performance on the AlphaServer SC (Section A.6).

The model, and particularly the atmosphere model, contains a very large number of `if` constructs. These select code paths according to the spatial resolution, the physical schemes which have been selected within the model control file, or the variables which have been selected to be saved. In many cases, and particularly in the case of the spatial resolution, it is highly unlikely that the user would wish to vary the configuration of the model.

The use of preprocessor directives, in which the configuration of the model would be selected at compile time, rather than at runtime, could therefore enable a large number of `if` constructs to be removed. This should lead to increased performance, partly as a result of the overheads associated with the execution of the `if` constructs, and partly because their presence can restrict the ability of compilers to perform certain optimisations.

However, the performance gains would be unlikely to exceed a few percent, and would be unlikely to justify the time that would be required to restructure the source code.

A.11.3 FFTW

The model currently uses version 2.x of the FFTW Fortran interface. This could readily be updated to version 3.x which, according to the release notes (FFTW, 2005b), can be more than 20% faster than version 2.x.

This would require a minimal investment of time; however, as profiling indicates that FFTs account for less than 2% of the total execution time, any potential performance gain would also be minimal.

A.11.4 Loop restructuring

A performance gain of $\sim$6% was achieved through manual restructuring of loops within the ocean model (Section A.6.1). Although this suggests that a similar performance gain might be achieved within the atmosphere model, as indeed was the case for the subroutine `cavit` (Section A.7.1), the size of the atmosphere model ($\sim$75,000 lines of source code) would make this a very time-consuming task.

A.11.5 Masking

In order to achieve optimal performance on vector architectures, the ocean model currently carries out calculations for every single gridbox, irrespective of whether it contains land or ocean. For the default configuration of Mk3L, only 41,717 out of a total of 75,264 gridboxes (on the tracer grid) are regarded as forming part of the ocean. This means that $\sim$45% of the calculations that are performed are unnecessary.

For optimal performance on serial architectures, it is possible that a performance gain could be achieved through the use of `if` constructs to restrict calculations to those gridboxes which form part of the ocean. However, this would be a time-consuming task, and, as `if` constructs would introduce additional overheads and restrict compiler optimisations, it is not obvious that any performance gain could be achieved.

Appendix B

Modifications to the atmosphere model

B.1 Introduction

This appendix documents the modifications which were made to the atmosphere model during the development of Mk3L. Section B.2 describes the changes which were made to the model physics, while Section B.3 describes the changes which relate to input and output.

B.2 Model physics

B.2.1 Insolation

The original version of the model uses hard-coded values for the solar contant and the Earth's orbital parameters. The solar constant, in Langleys per minute (ly/min), is specified by the variable `csolar`, which is set equal to 1.96 by the subroutine `initfs`. 1 Langley is exactly equal to 41840 Jm^{-2} (Lide, 1999), and so the original value for the solar constant is 1366.77 Wm^{-2}.

The epoch, in years BP (Before Present, where "present" is taken as the year 1950), is specified by the variable `bpyear`, which is set equal to 0 by the subroutine `initfs`. `bpyear` is passed as an argument to the subroutine `solargh`, which sets the Earth's orbital parameters to appropriate values. However, the only supported values for `bpyear` are 0, 6000 and 21000; `solargh` returns pre-set values for the Earth's orbital parameters for each of these three epochs, and aborts execution otherwise. The values returned by `solargh` for each of these three epochs are shown in Table B.1.

Variable	Description	bpyear		
		0	**6000**	**21000**
ec	Eccentricity of Earth's orbit	0.016724	0.018682	0.018994
peril	Longitude of perihelion (°)	102.04	0.87	114.42
oblqty	Obliquity of Earth's axis (°)	23.446	24.105	22.949

Table B.1: The original values for the Earth's orbital parameters.

Years BP	PMIP2	Original model	orbpar	Error
0	102.04	102.04	102.07	+0.03
6,000	0.87	0.87	0.83	-0.04
21,000	114.42	114.42	114.15	-0.27

Table B.2: The longitude of the perihelion (degrees) as specified by PMIP2 experimental design, as originally hard-coded into the model, as calculated by the subroutine `orbpar`, and the error in the values calculated by `orbpar`.

Mk3L was designed to enable climate variability and change to be studied on millennial timescales, including palaeoclimate simulations. It should therefore be possible to specify the solar constant and the epoch via the model control file, and to be able to vary them continuously over as wide a range of values as possible, rather than to have the values hard-coded into the model.

The NASA/GISS Atmosphere-Ocean Model subroutine `ORBPAR.SUB` (Goddard Institute for Space Studies, 2001) calculates the Earth's orbital parameters for any given epoch, using the method of Berger (1978); this method is accurate for dates extending $O(10^6)$ years into the past and future. `ORBPAR.SUB`, which is written in Fortran 77, was re-named `orbpar.f` and modified as follows:

- the longitude of the perihelion is now returned, rather than the longitude of the sun at perihelion (which differs from the longitude of the perihelion by 180°)
- angles are now returned in degrees, rather than radians
- some array-bounds violations were rectified

For the epochs 0, 6 and 21 thousand years BP, `orbpar` was found to return values for `ec` and `oblqty` which agree, to the number of decimal places shown, both with the values which had originally been hard-coded into the model (Table B.1), and with the values which are specified by PMIP2 (Paleoclimate Modelling Intercomparison Project, 2005) experimental design. The values returned for `peril` are shown in Table B.2. While these differ slightly from the values specified by PMIP2, the discrepancies were considered to be sufficiently small as to be negligible. The subroutine `orbpar` was therefore incorporated into the model.

A new header file, `ORBRAD.f`, was then added to the source code. This file defines the `common` block `orbrad`, as follows:

```
integer bpyear
real csolar, ec, peril, oblqty
common /orbrad/ bpyear, csolar, ec, peril, oblqty
```

The subroutine `initfs` was modified to call `orbpar` at the start of each run, and to obtain the value of `csolar` via `orbrad`. The solar constant is then converted from Wm^{-2} to ly/min, as required by the existing radiation code; this enables the user to specify the value of `csolar` in Wm^{-2}. The subroutine `solargh` was modified to obtain the values for the Earth's orbital parameters via `orbrad`, rather than the previous approach whereby they were hard-coded into the model.

The subroutine `readnml2` was then modified to read the values of `bpyear` and `csolar` from the model control file, as part of the `namelist` group `control`. The subroutine `readnml1` was modified to give `bpyear` and `csolar` default values of 0 and 1366.77333333333 respectively.

Flux	ERBE	Run A	Run B	Run C
Outgoing shortwave radiation (60°S–60°N)				
Net	-100.20	-109.00	-104.87	-101.89
Clear sky	-49.81	-51.56	-51.52	-51.49
Cloud forcing	-50.41	-57.44	-53.35	-50.40
Outgoing longwave radiation (60°S–60°N)				
Net	-242.90	-246.11	-246.36	-246.55
Clear sky	-274.30	-279.47	-279.62	-279.75
Cloud forcing	+31.39	+33.36	+33.27	+33.20

Table B.3: Annual-mean heat fluxes (Wm^{-2}, positive downward) at the top of the atmosphere, for ERBE (1985–1990 average), and for AGCM spin-up runs A, B and C (averages for years 11–50).

These modifications give the user complete control over the values of the solar constant and the epoch, while ensuring that the default behaviour of the model remains unchanged.

The source code for `orbpar` is provided in Section G.4.

B.2.2 Cloud albedo

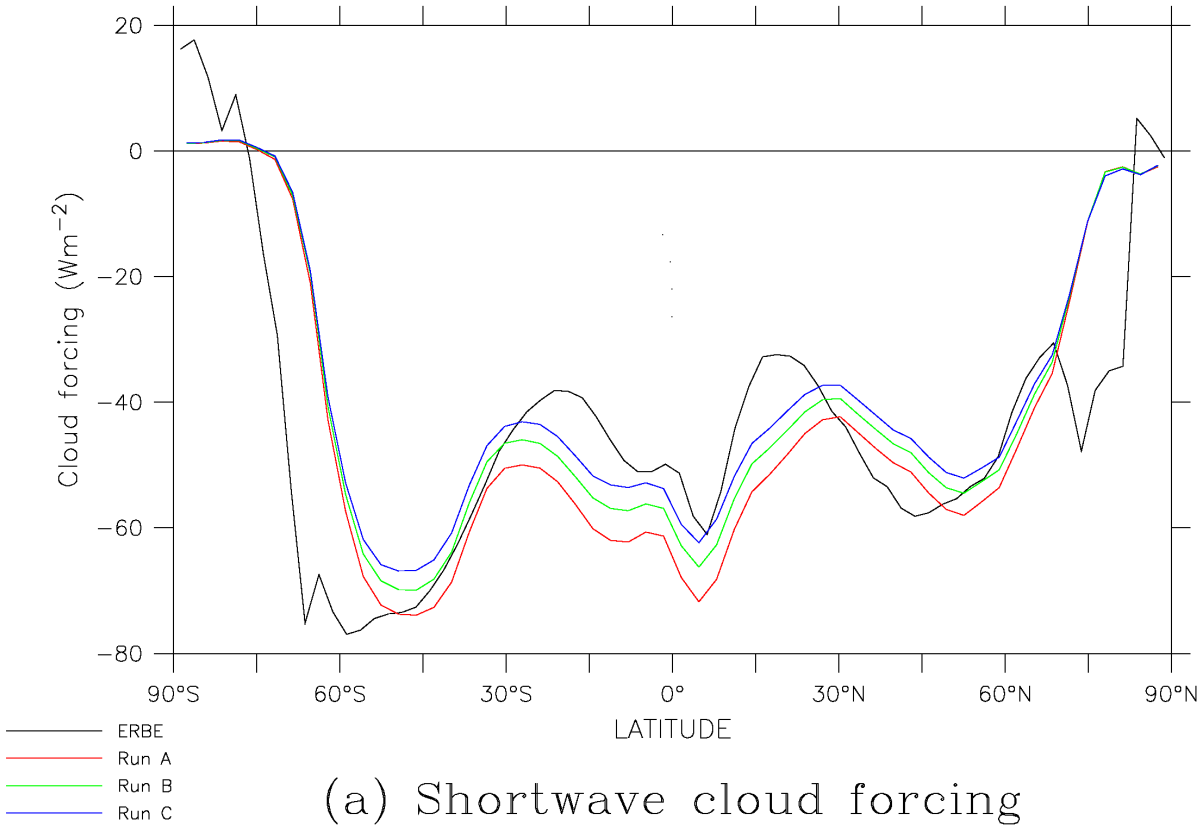
An initial atmosphere model (AGCM) spin-up run revealed a large surface energy imbalance, with the annual-, global-mean atmosphere-ocean heat flux being diagnosed as -8.83 Wm^{-2}. Pre-industrial conditions are used to initialise the model on the assumption that the Earth's climate was in, or very close to, thermal equilibrium at this time. The climatological mean atmosphere-ocean heat flux should therefore be approximately equal to zero.

Any net atmosphere-ocean heat flux simulated by the AGCM during a spin-up run will, in the absence of flux adjustments being employed within the coupled model, represent a source of drift upon coupling to the ocean model. A small net heat flux can be corrected for through the use of a global-mean heat flux adjustment; however, a surface energy imbalance as large as 8.83 Wm^{-2} indicates a significant deficiency within the model physics.

The most likely source of the imbalance was excessive shortwave cloud forcing (Rotstayn, pers. comm.). Table B.3 shows the outgoing fluxes of shortwave and longwave radiation at the top of Earth's atmosphere. Observed values are diagnosed from ERBE (the Earth Radiation Budget Experiment, Barkstrom, 1984), and represent the 5-year mean from February 1985 to January 1990. The averages are calculated over the latitude range 60°S–60°N, owing to the unreliable nature of the ERBE data at higher latitudes (Rotstayn, pers. comm.). The zonal-mean cloud forcings are plotted in Figure B.1.

The ERBE data is compared with three AGCM spin-up runs, of which Run A was the one conducted using the original configuration of the model. The values for the AGCM represent 40-year means from years 11 to 50 of each run. While the longwave cloud forcing can be seen to be in good agreement with ERBE, excessive shortwave cloud forcing in the tropics leads to excessive outgoing shortwave radiation at the top of the atmosphere.

Rotstayn (pers. comm.) suggested that a reduction in the cloud albedo would reduce the shortwave cloud forcing. Two parameters within the model, `refac1` and `refac2`, reduce the cloud albedo to compensate for



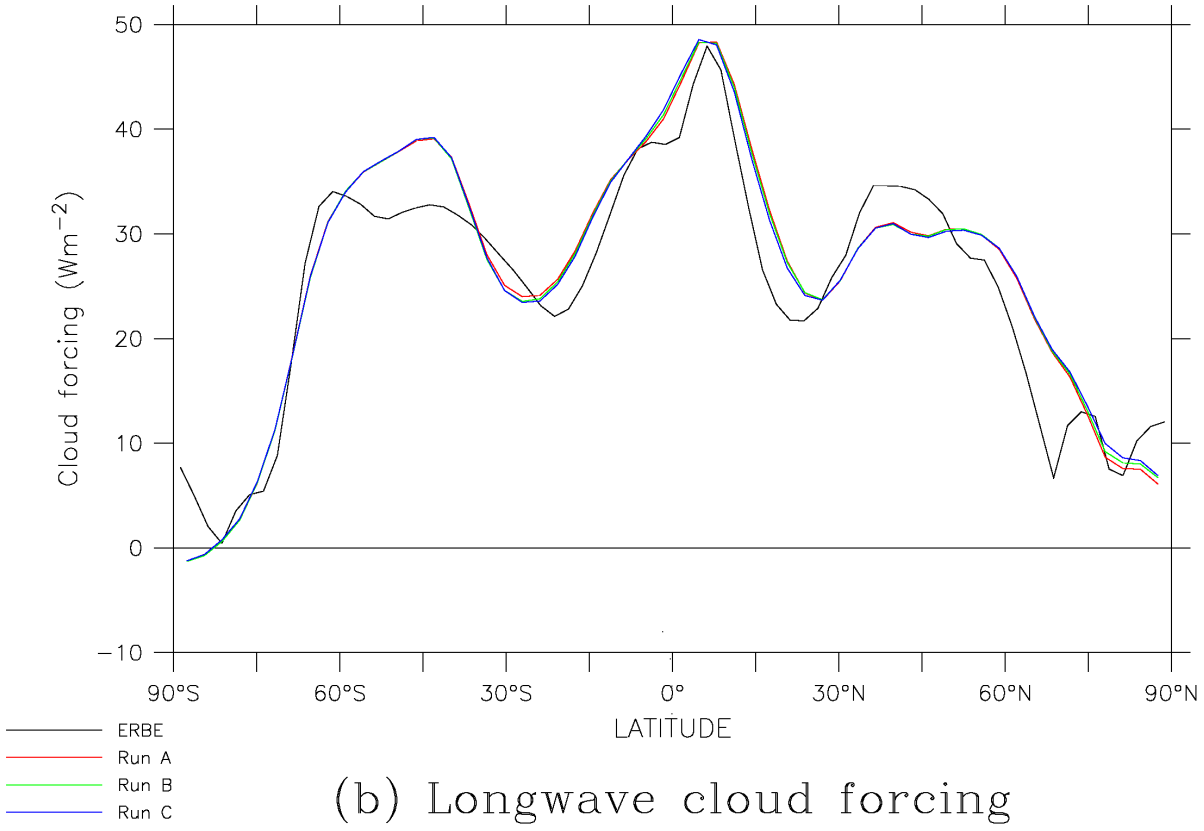


Figure B.1: The annual-, zonal-mean cloud forcing (positive downward) for ERBE (1985–1990 average), and for AGCM spin-up runs A, B and C (averages for years 11–50): (a) shortwave, and (b) longwave.

	Run A	Run B	Run C
refac1	0.85	0.7	0.595
refac2	0.95	0.9	0.865
Heat flux (Wm^{-2})	-8.83	-3.79	-0.17

Table B.4: The values of the parameters refac1 and refac2 for AGCM spin-up runs A, B and C, and the resulting net annual-, global-mean heat flux into the ocean.

the fact that the upper surfaces of clouds are neither smooth nor horizontal. The cloud albedo is multiplied by refac1 in the case of convective clouds, and by refac2 for all other clouds; these parameters have the values 0.85 and 0.95 respectively in the original version of the model.

To enable the values of refac1 and refac2 to be varied, the subroutine cloud2 was modified so that the values of these parameters are obtained via the common block cloudpar. The subroutine readnml2 was modified to read the values of the parameters from the model control file, as part of the namelist group control. The subroutine readnml1 was then modified so that the default values of refac1 and refac2 are set equal to the original values of 0.85 and 0.95 respectively.

For AGCM spin-up run B, the values of refac1 and refac2 were reduced to 0.7 and 0.9 respectively, being the values that were used in the Mk2 atmosphere model (Rotstayn, pers. comm.). While this reduced the magnitude of the shortwave cloud forcing in the tropics, it remained excessive. Furthermore, the net atmosphere-ocean heat flux remained unacceptably large in magnitude, at -3.79 Wm^{-2}.

For Run B, the values of refac1 and refac2 were reduced by 0.15 and 0.05 respectively, relative to Run A. For Run C, they were reduced by an additional 0.105 and 0.035 (representing 70% of the previous reductions), to 0.595 and 0.865 respectively. The mean shortwave cloud forcing was in excellent agreement with ERBE, and it can be seen from Figure B.1 that, while the zonal-mean cloud forcing does not reproduce the finer-scale variations of the observed values, the fit is much better than previously.

Table B.4 summarises the differences between the three AGCM spin-up runs. For Run C, the net atmosphere-ocean heat flux was reduced in magnitude to -0.17 Wm^{-2}; based on this consideration, and on the fit to observed values for the shortwave cloud forcing, values for refac1 and refac2 of 0.595 and 0.865 were selected for future use within the model.

B.2.3 Sea ice model

White ice formation

Analysis of preliminary AGCM spin-up runs for pre-industrial conditions, conducted using Mk3L, revealed excessive fluxes of freshwater into the ocean at high latitudes. Figure B.2 shows that the zonal-mean surface salinity tendency (Section D.2) is negative at latitudes higher than $\sim 40^{\circ}$.

Analysis of the surface freshwater fluxes due to precipitation, evaporation and run-off (Table B.5) revealed that the net flux of freshwater into the ocean is -4.2$\times 10^{11}$ m^3/year. Using Equation D.4, this can be converted into an equivalent annual-, global-mean surface salinity tendency of +1.6$\times 10^{-3}$ psu/year. The actual annual-, global-mean surface salinity tendency calculated by the AGCM, however, is -17.1$\times 10^{-3}$ psu/year.

The salinity tendency calculated by the AGCM takes into account not only the freshwater fluxes due to

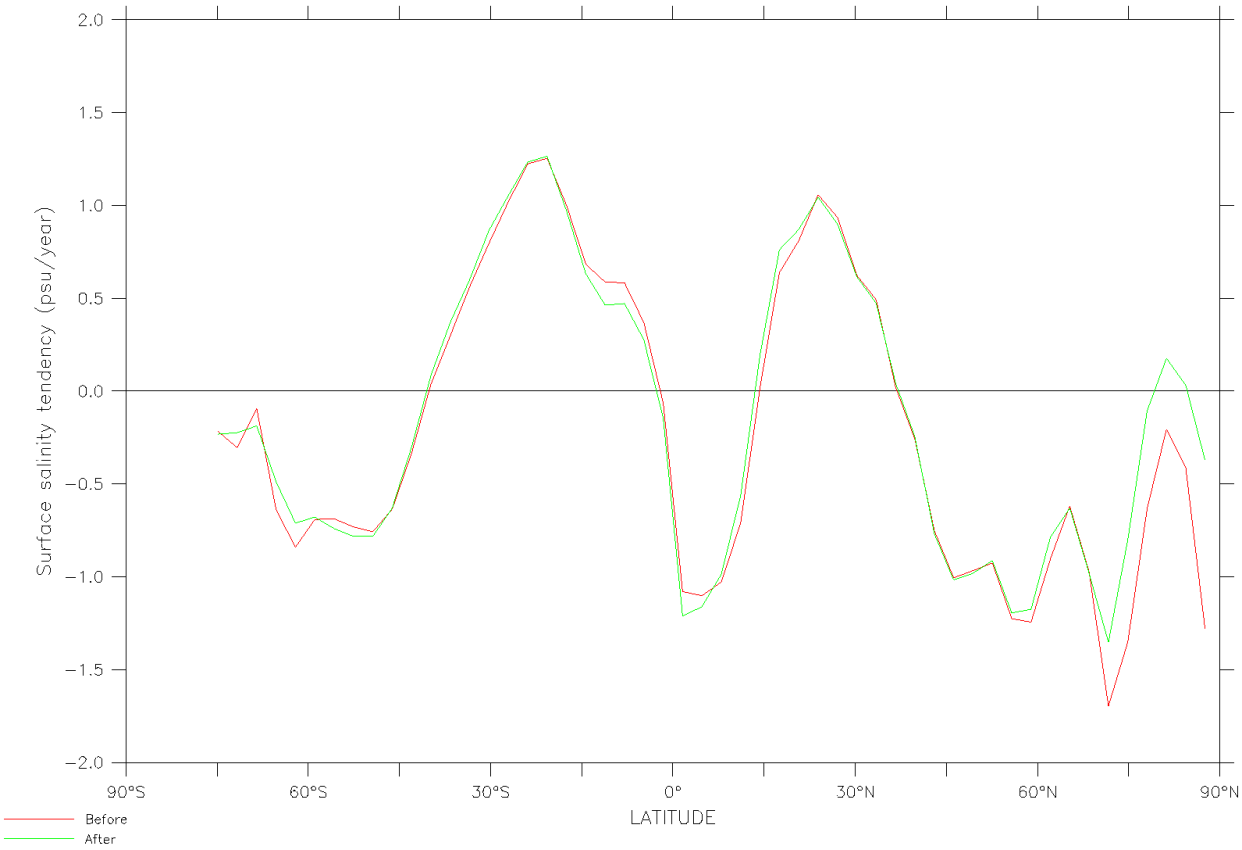


Figure B.2: The climatological annual-, zonal-mean surface salinity tendency diagnosed from AGCM spin-up runs, before (red, 50-year mean) and after (green, 40-year mean) the modifications to the treatment of white ice.

precipitation, evaporation and run-off, but also those associated with the formation and decay of sea ice. The discrepancy between the two values for the surface salinity tendency can therefore be attributed to a freshwater conservation error within the sea ice model.

Inspection of the source code identified an error in the treatment of white ice, which forms when the weight of snow upon sea ice becomes sufficiently great that part of the snow becomes submerged. The following section of code within the subroutine `seaice` deals with the freshwater fluxes associated with the formation of white ice:

```
xxx=hi+hs-(ps*hs+rhowi*hi)/pw
if(xxx.lt.hs) then
  excess=(hs-xxx)*0.1          ! m of water
  snowmi(mg)=snowmi(mg)+excess ! m of water
  hn=hi+excess
  hb=(0.5*hi)+excess ! Assume 2 levels of ice, hence 0.5*hi
  tn1(mg)=((0.5*hi)*t(1)+excess*t0)/hb + tfrz
  hs=xxx
  hi=hn
endif
```

`excess` represents the depth of snow (expressed as metres of freshwater) which is converted to white

Quantity	Units	Original code	Modified code
Precipitation over the ocean	10^{12} m^3/year	403.50	403.84
Evaporation over the ocean	10^{12} m^3/year	441.79	441.99
Run-off	10^{12} m^3/year	37.87	37.68
Net freshwater flux into the ocean	10^{12} m^3/year	-0.42	-0.48
Equivalent surface salinity tendency	10^{-3} psu/year	+1.6	+1.9
Actual surface salinity tendency	10^{-3} psu/year	-17.1	+4.1

Table B.5: Climatological surface freshwater fluxes diagnosed from AGCM spin-up runs, before and after the modifications to the treatment of white ice; the equivalent global-mean surface salinity tendency; and the actual global-mean surface salinity tendency calculated by the model. The figures represent 50-year means in the case of the original code, and 40-year means in the case of the modified code.

ice during the current timestep. However, this depth of freshwater is *both* added to the ocean as a flux of freshwater (by being added to the surface freshwater flux, `snowmi`) *and* added to the sea ice thickness. This erroneously adds freshwater to the model whenever white ice forms.

A modified version of the subroutine `seaice`, which rectifies this error, was supplied by Gordon (pers. comm.). Figure B.2 shows that this modification had a significant impact on the net freshwater flux into the Arctic Ocean. Table B.5 also indicates much improved conservation of freshwater within the sea ice model. A small discrepancy appears to remain, and deserves further investigation. However, it should be noted that the above calculations are not exact, owing to the fact that AGCM fields are saved as 16-bit integers, and that this discrepancy may not therefore represent a conservation error within the model.

Anomalous surface freshwater fluxes

Preliminary coupled model simulations conducted using Mk3L were found to contain occasional anomalous "events", characterised by instantaneous and very large fluxes of freshwater out of the ocean at high latitudes. This is illustrated by Figure B.3, which shows the maximum annual-mean surface salinity tendency for each year of a preliminary coupled model control simulation.

The values for years 270, 317, 352 and 398 are clearly anomalous. The largest "event" occurs during year 317, when the annual-mean surface salinity tendency at gridpoint (27, 8), which lies adjacent to the coast of Antarctica at 146°E, 65°S, is +108.8 psu/year. Investigation of the monthly salinity tendencies revealed that the anomaly arose during January, for which the monthly-mean salinity tendency was +1275 psu/year. This is equivalent to a total monthly freshwater flux out of the ocean of $\sim$80 m.

Through insertion of additional diagnostic source code into the model, these "events" were found to be instantaneous, with fluxes of freshwater out of the ocean of up to $\sim$400 m taking place during a single timestep. Such excessive fluxes were giving rise to highly unrealistic ocean properties; they were also creating very large failures of freshwater conservation, not least because the approximations used to derive Equation D.4 break down when such large fluxes arise.

These anomalous freshwater fluxes were found to originate within the subroutines `surfupl` and `radcoupl`. The following section of code within `surfupl` deals with the snow cover on sea ice:

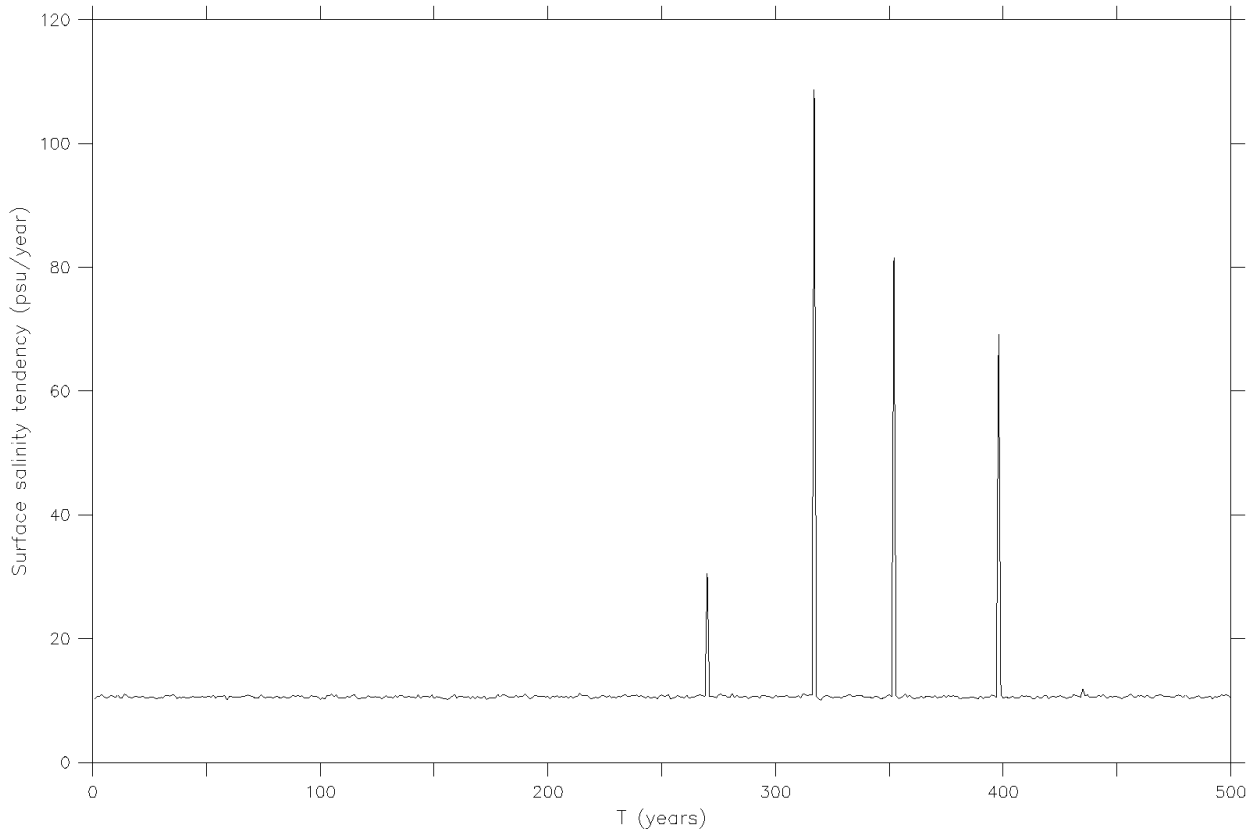


Figure B.3: The maximum annual-mean surface salinity tendency for a preliminary coupled model control simulation. For each year, the value plotted is $\max_{i,j} F_{i,j}$, where $F_{i,j}$ is the mean surface salinity tendency at each gridpoint for that year.

```
        if(pc.gt.0.0)then
c if pc>0 (ice cover has grown) then
c transfer volume of current snow across new ice
          dsn(mg)=dsn(mg)*(1.0-plold)/(1.0-(pl(mg)-pc))
        else
c if pc=0 (heating into leads), no ice change
c   and hiinc=hinew=pc=smelt=0.0
c if pc<0 (ice cover has melted) then
c melt the snow on the fraction (pc) of melted ice
          smelt=0.1*dsn(mg)*pc  ! < 0
          snowmi(mg)=snowmi(mg)-smelt/(1.0-(pl(mg)-pc)) ! m water
        endif
      endif
```

`pc` represents the change in sea ice concentration during the current timestep, while `pl(mg)` represents the fraction of open water. When `pc` is negative, it is possible for `(pl(mg)-pc)` to exceed 1, and hence for `1.0-(pl(mg)-pc)` to become negative. When this occurs, the fraction in the line

```
          snowmi(mg)=snowmi(mg)-smelt/(1.0-(pl(mg)-pc)) ! m water
```

becomes positive (as `smelt` is negative). If `1.0-(pl(mg)-pc)` is small in magnitude, then `snowmi(mg)`,

which represents the flux of freshwater into the ocean arising from snow melt, can become very large and negative.

Later within the routine, the value of `pl(mg)` is updated:

```
      pl(mg)=pl(mg)-pc
```

At the end of the routine, the values of several variables are reset, should all the sea ice have disappeared during the current timestep:

```
c Check limits
      if((pl(mg).ge.1.0).or.(dic(mg).le.0.0))then
        snowmi(mg)=snowmi(mg)+max(dsn(mg),0.0)*0.1 ! m of water
        pl(mg)=0.0
        dsn(mg)=0.0
        dic(mg)=0.0
        tg(mg)=tfi
        tstar(mg)=tfi
c imsl and il switching occurs at next sea ice step
      endif
```

The open water fraction `pl(mg)` is erroneously reset to 0, representing 100% sea ice cover. This leads to a freshwater conservation error later, within the subroutine `radcoupl`, when `snowmi(mg)` is added to the net flux of freshwater into the ocean, `dsfw`:

```
        dsfw  (mg,lg)=(pcondx(mg)-pevap(mg))*0.001*pl(mg)
     &                    +snowmi(mg)*(1.0-pl(mg))
```

As `pl(mg)` has been set equal to 0, rather than 1, the value of `snowmi(mg)` is added in full to `dsfw`. When `snowmi(mg)` becomes very large and negative, this gives rise to a large flux of freshwater out of the ocean.

Modified versions of the subroutines `radcoupl`, `seaice` and `surfupl` were supplied by Gordon (pers. comm.), rectifying this problem.

B.2.4 Numerical instabilities

In developing Mk3L, care was taken to ensure that the source code contains no instructions which generate array-bounds violations or floating-point exceptions (Section A.4). However, floating-point exceptions were found to occur occasionally, if very rarely, during coupled model simulations. These appeared to arise as a result of local violations of the CFL criterion (e.g. Washington and Parkinson, 1986). Of 24 floating-point exceptions which occurred during initial coupled model simulations, all arose within the atmosphere model. Of these exceptions, 21 were found to have arisen at just five lines within the source code: two lines within the subroutine `gwdrag`, two within the subroutine `fst88`, and one within the subroutine `hsflux`. The other three exceptions all arose within separate routines.

Inspection of the source code revealed that floating-point exceptions could only arise at these lines as a result of catastrophic numerical stabilities having already arisen (for example, the pressure or absolute

temperature having fallen below zero). Thus, the only course of action that could be taken was to insert additional instructions into the source code. These instructions would detect floating-point exceptions before they take place, allowing the model to halt execution cleanly and to display diagnostic information.

`gwdrag`

Within the subroutine `gwdrag`, the loop

```
      do 708 mg=1,ln2
  708 bvng(mg)=sqrt(grav*dthdz(mg,1)/tg(mg))
```

will give rise to a floating-point exception if the operand of the square root is negative. `grav` is a positive constant, and so the operand can only be negative if either `dthdz(mg, 1)` or `tg(mg)` are negative.

The value of `dthdz(mg, 1)` is calculated earlier within the subroutine by the expression

```
      dthdz(mg,1)=(thf(mg,1)-tg(mg))*dzi(mg,1)
```

`(thf(mg,1)-tg(mg))` is always positive, and so `dthdz(mg, 1)` can only be negative if `dzi(mg,1)` is negative. The value of `dzi(mg,1)` is given by the expression

```
      dzi(mg,k)=(prf(mg,k)/dprf(mg,k))*dzx/ttg(mg,k)
```

`dzx` is a positive constant, and so the operand of the original square root can only be negative if at least one of `tg(mg)`, `prf(mg, 1)`, `dprf(mg, 1)` and `ttg(mg, 1)` are negative.

The original loop was therefore replaced with

```
      do mg = 1, ln2
        temp_sjp = grav * dthdz(mg, 1) / tg(mg)
        if (temp_sjp .lt. 0.0) then
          write (*, *)
          write (*, *) "ABORTING: Fatal error in GWDRAG"
          write (*, *)
          write (*, *) "mg = ", mg
          write (*, *) "lg = ", lg
          write (*, *)
          write (*, *) "thf(mg, 1)  = ", thf(mg, 1)
          write (*, *) "tg(mg)      = ", tg(mg)
          write (*, *) "prf(mg, 1)  = ", prf(mg, 1)
          write (*, *) "dprf(mg, 1) = ", dprf(mg, 1)
          write (*, *) "ttg(mg, 1)  = ", ttg(mg, 1)
          write (*, *)
          stop
        end if
        bvng(mg) = sqrt(temp_sjp)
      end do
```

Similarly, the loop

```
      do 40 k=1,nl-1
      do 710 mg=1,ln2
      bvnf(mg,k)=sqrt(  (0.5*grav)*(dthdz(mg,k)+dthdz(mg,k+1))/
     &  thf(mg,k) )
  710 continue
   40 continue
```

will give rise to a floating-point exception if the operand of the square root is negative. It was therefore replaced with

```
      do k = 1, nl-1
        do mg = 1, ln2
          temp_sjp = 0.5 * grav * (dthdz(mg, k) + dthdz(mg, k+1))
     &                           / thf(mg, k)
          if (temp_sjp .lt. 0.0) then
            write (*, *)
            write (*, *) "ABORTING: Fatal error in GWDRAG"
            write (*, *)
            write (*, *) "mg = ", mg
            write (*, *) "lg = ", lg
            write (*, *) "k  = ", k
            write (*, *)
            write (*, *) "thf(mg, k-1)  = ", thf(mg, k-1)
            write (*, *) "thf(mg, k)    = ", thf(mg, k)
            write (*, *) "thf(mg, k+1)  = ", thf(mg, k+1)
            write (*, *) "prf(mg, k)    = ", prf(mg, k)
            write (*, *) "prf(mg, k+1)  = ", prf(mg, k+1)
            write (*, *) "dprf(mg, k)   = ", dprf(mg, k)
            write (*, *) "dprf(mg, k+1) = ", dprf(mg, k+1)
            write (*, *) "ttg(mg, k)    = ", ttg(mg, k)
            write (*, *) "ttg(mg, k+1)  = ", ttg(mg, k+1)
            write (*, *)
            stop
          end if
          bvnf(mg, k) = sqrt(temp_sjp)
        end do
      end do
```

`fst88`

Within the subroutine `fst88`, the loop

```
      do 481 j = 1, lp1
```

```
        do 481 i = 1, imax
          vtmp3(i, j) = exp(hm1ez*totvo2(i, j))
          totevv(i, j) = one / vtmp3(i, j)
481   continue
```

will give rise to a floating-point exception if `vtmp3(i, j)` is equal to zero. It was therefore replaced with

```
      do 481 j = 1, lp1
        do 481 i = 1, imax
          vtmp3(i, j) = exp(hm1ez*totvo2(i, j))
          if (vtmp3(i, j) .eq. 0.0) then
            write (*, *)
            write (*, *) "ABORTING: Fatal error in FST88"
            write (*, *)
            write (*, *) "i = ", i
            write (*, *) "j = ", j
            write (*, *)
            write (*, *) "totvo2(i, j) = ", totvo2(i, j)
            write (*, *)
            stop
          end if
          totevv(i, j) = one / vtmp3(i, j)
481   continue
```

The loop

```
      do 619 k=1,l
      do 619 i=1,imax
      rlog(i,k)=log(over(i,k,k+1)*co2nbl(i,k))
619   continue
```

will also give rise to a floating-point exception if the operand of the logarithm is negative. It was therefore replaced with

```
      do 619 k=1,l
      do 619 i=1,imax
      temp_sjp = over(i, k, k+1) * co2nbl(i, k)
      if (temp_sjp .lt. 0.0) then
        write (*, *)
        write (*, *) "ABORTING: Fatal error in FST88"
        write (*, *)
        write (*, *) "i = ", i
        write (*, *) "k = ", k
        write (*, *)
        write (*, *) "over(i, k, k+1) = ", over(i, k, k+1)
        write (*, *) "co2nbl(i, k)    = ", co2nbl(i, k)
```

```
        write (*, *)
        stop
      end if
      rlog(i,k)=log(temp_sjp)
619   continue
```

`hsflux`

Within the subroutine `hsflux`, the line

```
          rich = ( -b + sqrt(b*b-4.*a*c) ) /(2.*a)
```

will give rise to a floating-point exception if the operand of the square root is negative. It was therefore replaced with

```
          temp_sjp = b*b - 4.0*a*c
          if (temp_sjp .lt. 0.0) then
            write (*, *)
            write (*, *) "ABORTING: Fatal error in HSFLUX"
            write (*, *)
            write (*, *) "mg = ", mg
            write (*, *) "lg = ", lg
            write (*, *)
            write (*, *) "tsurf(mg)  = ", tsurf(mg)
            write (*, *) "tstarx(mg) = ", tstarx(mg)
            write (*, *)
            stop
          end if
          rich = ( -b + sqrt(temp_sjp) ) /(2.*a)
```

B.3 I/O

B.3.1 Input files

Mixed layer depths

Within the stand-alone AGCM, each ocean gridpoint which lies immediately equatorward of a sea ice gridpoint is treated as a mixed-layer ocean (Section 2.2). These mixed-layer ocean gridpoints act as a buffer around the sea ice zone, controlling its horizontal extent. They have a fixed depth of 100 m, and their temperature is determined through relaxation towards observed SSTs.

In an earlier version of the model, the mixed-layer gridpoints had variable depths, with climatological depths for each month of the year being read from the auxiliary files `hhm01`, `hhm02`, ..., `hhm12`. These files are still read by the Mk3 atmosphere model, even though the data is no longer used. The subroutine `atstart` was therefore modified to remove the redundant section of code.

Ocean currents

The stand-alone AGCM must be provided with climatological ocean currents, which act as a bottom boundary condition on the sea ice model (Section 2.2). The subroutine `flatset` reads climatological ocean currents for each month of the year from the auxiliary files `ocuv01.3st`, `ocuv02.3st` ... `ocuv12.3st`.

For simplicity, and for consistency with the other auxiliary files read by the model, `flatset` was modified so that the ocean currents are read from the single auxiliary file `ocuv.3st`.

Run description

The original version of the subroutine `atstart` contains a section of code which reads the text file `c9details` at the start of each run, and writes the contents to standard output. This file is user-generated, and is intended to contain a short description of the run. This behaviour was considered to be unnecessary, and so the relevant section of code was removed.

B.3.2 Output files

Year numbers

The model was modified such that the year counter has five digits, rather than four. This enables runs of up to 100,000 years in duration, as opposed to the previous limit of 10,000 years.

Pressure levels

PMIP2 (Paleoclimate Modelling Intercomparison Project, 2005) experimental design requires that atmosphere model output be provided on 17 standard pressure levels. The subroutine `ncinit` was modified such that temperature, the zonal and meridional wind speeds, the specific and relative humidities, and the geopotential height are saved on the 18 pressure levels shown in Table B.6. These comprise the 17 standard pressure levels, plus an additional level at 5 hPa.

The two other variables which are saved on vertical levels (the cloud fraction and the rate of latent heating) are still saved on the hybrid vertical levels used by the model (Section 2.2).

Mixed-layer ocean temperatures

As discussed in Section B.3.1, a mixed-layer ocean is used by the stand-alone AGCM at high latitudes. The flag `dtm_sflg` is used to determine whether or not the mixed-layer ocean temperatures are saved to a netCDF file. However, in the original version of the model, the values could only be saved if the flag `savefcor` was also set equal to `.true.`. This causes a large amount of unrelated diagnostic output to be generated.

The subroutines `filest` and `ncinit` were modified so that the mixed-layer ocean temperatures can be saved, irrespective of the value of `savefcor`.

Level	Pressure (hPa)
1	1000
2	925
3	850
4	700
5	600
6	500
7	400
8	300
9	250
10	200
11	150
12	100
13	70
14	50
15	30
16	20
17	10
18	5

Table B.6: The pressure levels on which AGCM output is saved in Mk3L.

Date and time

The original version of the model generates a one-line text output file, `f30.his`, containing the current model date and time. This file is written at an interval specified, in minutes, by the variable `glmean_interval`. This has a default value of 480, in which case the file `f30.his` is generated every eight hours of model time; this typically corresponds to just a few seconds of walltime.

To open a file, write to it, and close it again so frequently will impede runtime performance. Furthermore, this file was considered to be unnecessary. The relevant section of code was therefore removed from the subroutine `atstep`.

Appendix C

Modifications to the ocean model

C.1 Introduction

This appendix documents the modifications which were made to the ocean model during the development of Mk3L. Section C.2 describes the changes which were made to the model physics, while Section C.3 describes the changes which relate to input and output. The utilities which were developed for the processing of ocean model output are described in Section C.4

C.2 Model physics

C.2.1 Mixing across unresolved straits

The ocean model bathymetry, as configured for Mk3L, defines six basins which have no resolved connection with the world ocean (Section 2.3):

- the Baltic Sea
- the Black Sea
- the Caspian Sea
- Hudson Bay
- the Mediterranean Sea
- the Persian Gulf

Figure C.1 shows the locations and extents of these basins, and illustrates the lack of any resolved connections with the world ocean.

This model bathymetry does not adequately represent the physical connections which exist within the ocean. With the exception of the Caspian Sea, water *is* exchanged between each of the above basins and the world ocean. In order to represent these connections within the model, mixing is imposed between the gridboxes lying on either side of each of the unresolved straits.

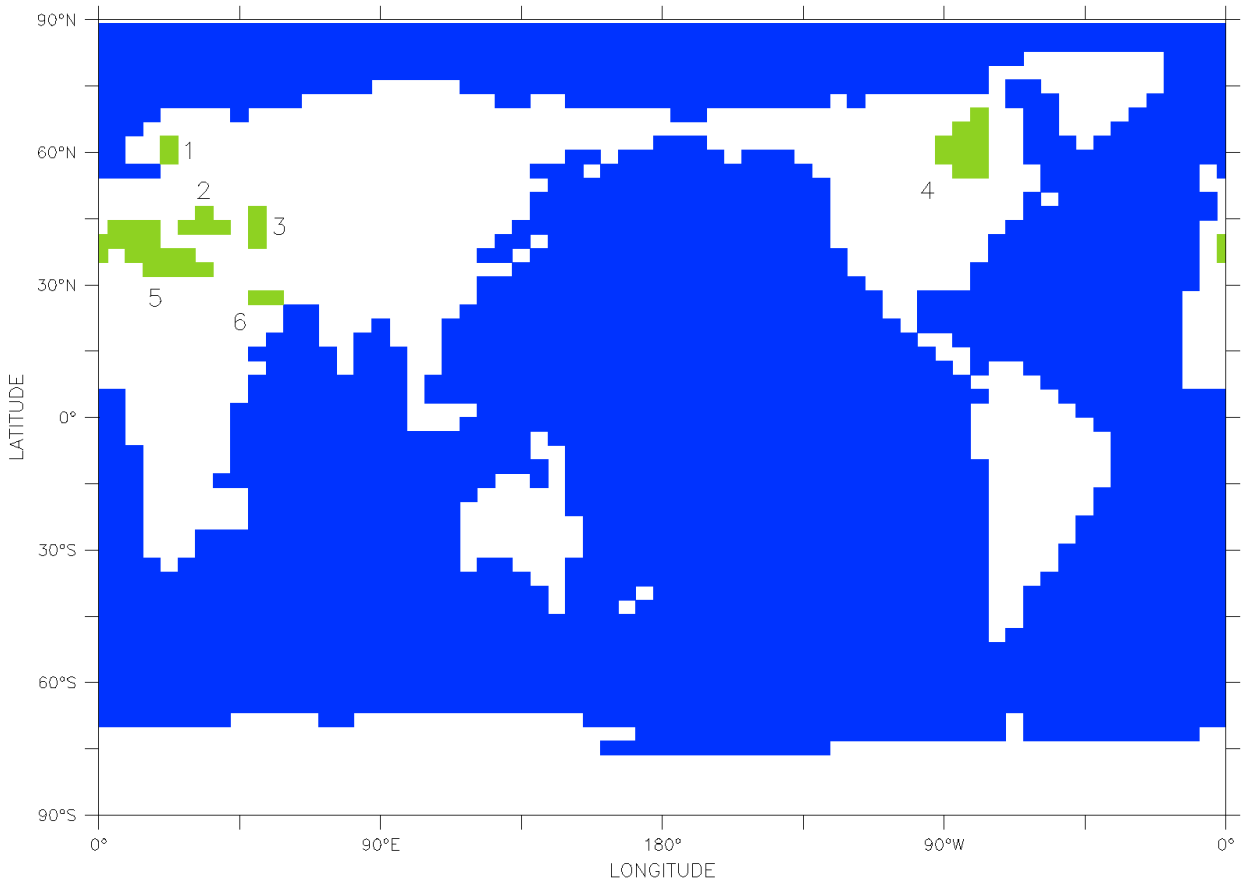


Figure C.1: The six ocean basins (green) which have no resolved connection with the world ocean (blue) in Mk3L: (1) the Baltic Sea, (2) the Black Sea, (3) the Caspian Sea, (4) Hudson Bay, (5) the Mediterranean Sea, and (6) the Persian Gulf.

Consider two masses of water, with volumes V_1, V_2. Let the values of the tracer T (where T represents temperature, salinity, or any other tracer) be T_1, T_2, and let water be exchanged between the two masses at a rate F. After an increment of time dt, a volume of water Fdt will have been exchanged, and the updated value of the tracer T_1 will be

$$T_1' = \left(1 - \frac{Fdt}{V_1}\right) T_1 + \frac{Fdt}{V_1} T_2 \quad \text{(C.1)}$$

$$= T_1 + \frac{Fdt}{V_1}(T_2 - T_1) \quad \text{(C.2)}$$

Defining exchange coefficients $\alpha_1 = F/V_1$ and $\alpha_2 = F/V_2$, the changes in the tracer values T_1, T_2 after an increment of time dt can therefore be written as

$$dT_1 = \alpha_1 dt(T_2 - T_1) \quad \text{(C.3)}$$

$$dT_2 = \alpha_2 dt(T_1 - T_2) \quad \text{(C.4)}$$

Applying these results to the model, let each unresolved strait have a sill depth H. If the gridboxes on either side of the strait have horizontal surface areas A_1, A_2, then the volumes of the two water masses are $V_1 = A_1 H$ and $V_2 = A_2 H$ respectively. The exchange coefficients can therefore be expressed as

Basin	F	H	Gridpoints		α_1, α_2 (10^{-9} s^{-1})	
	(Sv)	(m)	(i, j)	λ, ϕ	Original	Modified
Baltic Sea	0.02	160	(5, 47)	23°E, 59°N	-	1.09
			(4, 46)	17°E, 56°N	-	1.00
Black Sea	0.01	160	(6, 42)	28°E, 43°N	-	0.39
			(4, 42)	17°E, 43°N	-	0.39
Hudson Bay	0.1	160	(51, 48)	79°W, 62°N	36.00	6.03
			(54, 48)	62°W, 62°N	36.00	6.03
Mediterranean Sea	0.75	800	(1, 41)	0°, 40°N	7.20	5.51
			(63, 41)	11°W, 40°N	7.20	5.51
Persian Gulf	0.2	160	(10, 37)	51°E, 27°N	0.79*	3.17*
			(11, 37)	56°E, 27°N		
			(12, 36)	62°E, 24°N	1.55*	6.17*

Table C.1: The exchange coefficients used to mix water properties across unresolved straits. *Mixing is conducted between gridpoint (12, 36) and *both* of gridpoints (10, 37) and (11, 37).

$$\alpha_1 = \frac{F}{A_1 H} \tag{C.5}$$

$$\alpha_2 = \frac{F}{A_2 H} \tag{C.6}$$

At each timestep, Equations C.3 and C.4 can be used to modify the tracer values on either side of each unresolved strait, simulating the exchange of water. This takes place within the subroutine `tracer`, immediately after updated tracers have been calculated for each timestep.

The original version of the model only simulates the exchange of water between the world ocean and Hudson Bay, the Mediterranean Sea and the Persian Gulf; the Baltic and Black Seas are left isolated. The exchange coefficients are also hard-coded into the model, with the values used being shown in Table C.1.

The subroutine `tracer` was modified so that the Black and Baltic Seas are also connected with the world ocean. The exchange coefficients are now calculated at runtime, based on the values of F and H, and the volumes of the gridboxes. Although F and H are hard-coded into the model, as were the exchange coefficients previously, this approach is more transparent. It also removes the possibility of errors being introduced, should future users wish to change the values of either F or H; indeed, the exchange coefficients used for Hudson Bay in the original version of the model are inconsistent with comments indicating that the intended rate of exchange is 1 Sv, and they may therefore be incorrect.

The values for the rates of exchange across each strait were chosen to be typical of published values, as outlined in the following sections, with the rate of exchange across each unresolved strait being estimated by averaging the observed rates of inflow and outflow. The precise value of the rate of exchange is not critical, as the rate of transport of heat and salt across a strait is not proportional to the rate of exchange; an increase in the rate of exchange will tend to reduce any gradients, and hence will not lead to a corresponding increase in the rates of transport.

For each basin, the sill depth is determined by the model bathymetry. The depth is set equal to the shallower of the depths of the water column on either side of each unresolved strait.

In the case of the Persian Gulf, the original behaviour of the model has been retained, whereby the mixing takes place between the Indian Ocean and *both* of the gridpoints which constitute the Gulf. According to comments within the source code, this approach was introduced in order to avoid numerical problems.

Baltic Sea

Wulff et al. (2001) use Knudsen's hydrographic theorem (Knudsen, 1900) to estimate the rates of inflow and outflow for the Baltic Sea. Assuming equilibrium with its surroundings, the climatological net fluxes of both salt and freshwater into the Baltic Sea must be equal to zero. Observed values for precipitation, evaporation and run-off can be used to estimate the net flux of freshwater into the surface, whle observed salinities can be used to estimate the salinities of the inflow and outflow. By requiring that the net rate of exchange of salt with the world ocean be zero, and that the net rate of exchange of freshwater with the world ocean balances the net flux of freshwater into the surface, the rates of inflow and outflow can be determined.

Wulff et al. (2001) formulate a four-box model for the Baltic Sea. Using data on freshwater inputs over the period 1970-1991, and using observed salinities, the model is used to derive a mean rate of inflow of 238 km^3/year, and a mean rate of outflow of 746 km^3/year. The mean rate of exchange is therefore estimated to be 492 km^3/year, which is equivalent to 0.016 Sv.

Black Sea

Table C.2 shows published estimates of the rates of exchange of water between the Black and Mediterranean Seas, as listed by Black Sea Environmental Internet Node (2004). Full citations are not provided; the methods used to derive these estimates are therefore unknown, and the references are not included in the bibliography within this document.

The mean values for the rates of inflow and outflow are 200 km^3/year and 395 km^3/year respectively. These equate to an average rate of exchange of water of 297 km^3/year, which is equivalent to 0.0094 Sv.

Hudson Bay

Unlike the other basins discussed in this section, Hudson Bay is different in that it is not connected to the world ocean via a single channel. Instead, it is connected at its northern end to both Foxe Basin, which in turn is connected to the Arctic Ocean via Fury and Hecla Strait, and to Hudson Strait, via which it is connected to the Atlantic Ocean.

Using current meter measurements, both Sadler (1982) and Fissel et al. (1988) estimate the rate of inflow via Fury and Hecla Strait as being $\sim$0.04 Sv. Drinkwater (1988), using data from an array of current meters situated near the eastern end of Hudson Strait, estimates that the net rate of outflow to the Atlantic Ocean is $\sim$0.11 Sv.

These observational datasets do not indicate the rate of exchange between the waters of Hudson Bay and those of Foxe Basin and Hudson Strait. However, Saucier et al. (2004) use a three-dimensional coastal-ice ocean model to simulate the seasonal cycle of water masses and sea ice in the Hudson Bay system over the two-year period from August 1996 to July 1998. Analysis of this simulation reveals a net rate of inflow into Hudson Bay at depths below 100 m of 0.25 Sv. A net freshwater transport of 0.035 Sv is simulated

Reference	Rate of exchange (km^3/year)	
	Inflow	**Outflow**
Shpindler (1896, 1899)	-	416
Merz (1928)	193	398
Sverdrup (1942)	192	397
Zenkevich (1947)	202	348
Rojdestvenskiy (1953)	195	575
Neumann and Roseman (1954)	193	462
Leonov (1960)	193	392
Bruevich (1960)	175	400
Berenbeim (1960)	193	398
Solyankin (1963)	176	340
Okeanograf Entsikl. (1966)	202	398
Tixeront (1970)	-	400
Ozturgut (1971)	249	548
Rojdestvenskiy (1971)	229	485
Sitnikov (1972)	-	190
Serpoianu (1973)	123	260
Rozengurt and Sitnikov (1973)	-	241
Pora and Oos (1974)	229	485
Fonselius (1974)	200	400
Entsiklopedia Okean./Atm. (1983)	188	388
Altman (1984)	181	366
Bondar (1986)	203	371
Unluata et al (1990)	312	612
Altman (1991)	176	371
Reshetnikov (1992)	-	227

Table C.2: Estimated rates of exchange of water between the Black and Mediterranean Seas.

through the mouth of Hudson Strait, suggesting that the rate of outflow from Hudson Bay via the surface layers slightly exceeds the rate of inflow at depth.

Based on these studies, it seems reasonable to state that the average rate of exchange between Hudson Bay and the world ocean is of order 0.1 Sv. Note that within Mk3L, the mixing takes place between Hudson Bay and the Atlantic Ocean, with no attempt made to allow for the inflow via Fury and Hecla Strait.

Mediterranean Sea

Bryden et al. (1994) use current meter measurements to estimate an average rate of inflow from the Atlantic Ocean to the Mediterranean Sea of 0.72 Sv, and an average rate of outflow of 0.68 Sv. Tsimplis and Bryden (2000), using an acoustic doppler current profiler (ADCP), estimate the rates of inflow and outflow as being 0.78 Sv and 0.67 Sv respectively.

The average rate of exchange of water is therefore taken as being $\sim$0.75 Sv.

Persian Gulf

Ahmad and Sultan (1991) use Knudsen's hydrographic theorem (Knudsen, 1900) to compute an average rate of inflow into the Persian Gulf of 0.186 Sv, and an average rate of outflow of 0.169 Sv. Johns et al. (2003), using a combination of ADCP data, current meter data and Knudsen's hydrographic theorem, estimate the rates of inflow and outflow as being 0.23 Sv and 0.21 Sv respectively.

The average rate of exchange of water is therefore taken as being $\sim$0.2 Sv.

C.2.2 Density calculations

The subroutine `dencal` calculates the density of the ocean, as follows:

```
  ROE=999.842594+6.793952D-2*T1-9.095290D-3*T1*T1+1.001685D-4*T1*T1
&*T1-1.120083D-6*T1*T1*T1*T1+6.536332D-9*T1*T1*T1*T1*T1
  SIG=8.24493D-1-4.0899D-3*T1+7.6438D-5*T1*T1-8.2467D-7*T1*T1*T1+5.
&3875D-9*T1*T1*T1*T1
  SAL=(-5.72466D-3+1.0227D-4*T1-1.6546D-6*T1*T1)*S1**1.5+4.8314D-4*
&S1**2
  DEN=ROE+SIG*S1+SAL
```

where `T1` is the temperature (^{o}C), `S1` is the salinity (psu) and `DEN` is the density (kgm^{-3}).

Should the salinity be negative, the term `S1**1.5` generates a floating-point exception. Although non-physical, it is possible for negative salinities to arise within the model. Reasons why this might occur include: numerical instabilities arising from violation of the CFL criterion (e.g. Washington and Parkinson, 1986); "overshoot", whereby the calculated trend in salinity causes the predicted value for the next timestep to be negative; temporary freshening arising from a large flux of freshwater into the ocean; and long-term freshening arising from drift within the coupled model.

In practice, negative salinities were not encountered during ocean model spin-up runs, and were only encountered during coupled model runs as a result of long-term drift. The modified method for converting the surface freshwater flux to an equivalent surface salinity tendency (Section D.4) makes negative salinities much less likely, but not impossible. (Negative salinities occurred much more frequently in the original version of the model, where there was no feedback mechanism which might tend to constrain any drift in salinity.)

`dencal` was therefore modified as follows, so that negative salinities are treated as though the salinity is equal to zero:

```
  ROE=999.842594+6.793952D-2*T1-9.095290D-3*T1*T1+1.001685D-4*T1*T1
&*T1-1.120083D-6*T1*T1*T1*T1+6.536332D-9*T1*T1*T1*T1*T1

  if (s1 .gt. 0.0) then

  SIG=8.24493D-1-4.0899D-3*T1+7.6438D-5*T1*T1-8.2467D-7*T1*T1*T1+5.
&3875D-9*T1*T1*T1*T1
  SAL=(-5.72466D-3+1.0227D-4*T1-1.6546D-6*T1*T1)*S1**1.5+4.8314D-4*
```

Parameter	Value	
	Original	Modified
JFT0	51	54
JFT1	7	3
JFT2	52	55
JFU0	50	54
JFU1	7	3
JFU2	52	55

Table C.3: The parameters which control the Fourier filtering at high latitudes.

```
&S1**2
  DEN=ROE+SIG*S1+SAL

  else

    den = roe

  end if
```

While this modification allows execution to continue in the event that negative salinities arise, it should be noted that it also has the potential to obscure numerical problems within the model.

C.2.3 Fourier filtering

Ocean model spin-up runs conducted using the original version of the model were found to converge towards an equilibrium solution under asynchronous timestepping. This was not the case under synchronous timestepping, however, where drift was found to occur within the Arctic Ocean. This drift consisted of ongoing warming at depth, which would give rise to numerical instabilities after several thousand years. Hirst (pers. comm.) advised that this was due to numerical noise generated by the interaction between the Fourier filtering applied at high latitudes, and the complex model bathymetry in the vicinity of Svalbard (an island group which lies to the north of Norway, at around 20°E, 78°N).

Fourier filtering is used to reduce the timestep limitation arising from the CFL criterion and the convergence of meridians at high latitudes, particularly in the Arctic Ocean (Cox, 1984). The subroutine `filter` applies a low-pass filter to the tracers and velocities at high latitudes, increasing the effective zonal dimension of the gridboxes. The parameters which control the extent of the filtering are shown in Table C.3.

The parameters `JFT1` and `JFT2` specify the northernmost latitude row in the Southern Hemisphere, and the southernmost latitude row in the Northern Hemisphere, respectively, of the domain where filtering is applied. The filtering ensures that the gridboxes have an effective zonal dimension equal to that of the gridboxes at the latitude row specified by `JFT0`. The parameters `JFU0`, `JFU1` and `JFU2` have equivalent meanings to `JFT0`, `JFT1` and `JFT2`, and apply to the filtering of horizontal velocities.

The values of these parameters were modified as shown in Table C.3, removing all filtering in the Southern Hemisphere, and shifting the southern limit of the filtering in the Northern Hemisphere from 70.1°N to

79.6°N for tracers, and from 71.6°N to 81.2°N for horizontal velocities. These limits lie to the north of Svalbard, and were found to eliminate the drift encountered under synchronous timestepping, without requiring any change to the timestep.

C.2.4 Eddy diffusion in the Arctic Ocean

The original version of the model features a transition from Gent-McWilliams eddy diffusion (Gent and McWilliams, 1990) to horizontal diffusion within the Arctic Ocean. This is achieved through the following section of code within the subroutine `ocean`:

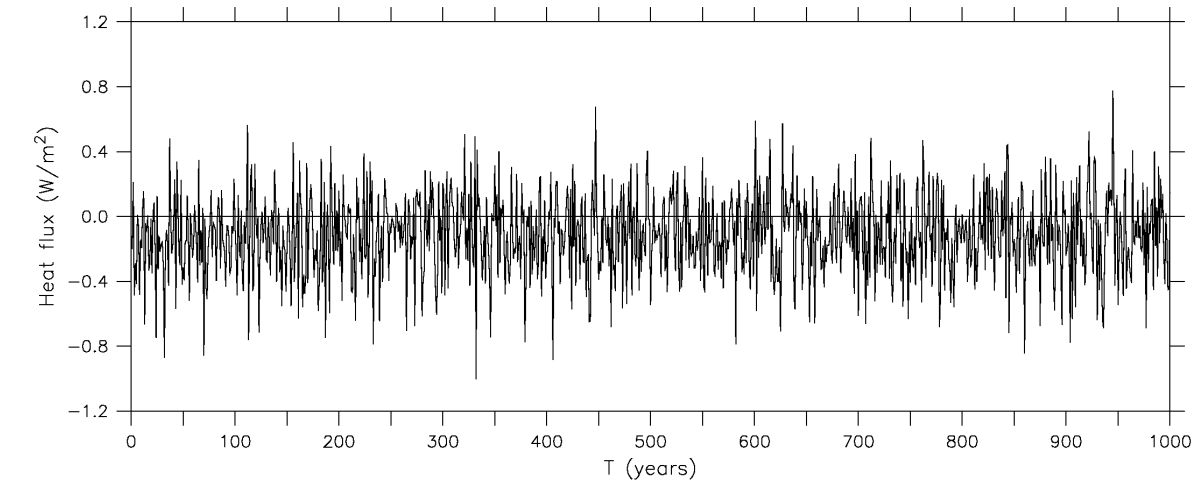
```
do j = 1,jmt
  ahifac(j,k) = 1.
  ahhfac(j,k) = 1.
  if(igm.eq.1)then
    ahefac(j,k) = 1.
    if(j.eq.53)ahefac(j,k) = 0.5
    if(j.ge.54)ahefac(j,k) = 0.
    if(j.eq.53)ahhfac(j,k) = 0.5
    if(j.le.52)ahhfac(j,k) = 0.
  else
    ahefac(j,k) = 0.
  endif
enddo
```

When the parameter `igm` is equal to 1, which specifies that Gent-McWilliams eddy diffusion is to be used, this loop sets the values of the arrays `ahefac` and `ahhfac`. These variables specify the extent to which diffusion of tracers should be determined by Gent-WcMilliams eddy diffusion, and by horizontal diffusion, respectively. The values are set such that the transition occurs linearly over two latitude rows, with the northern limit of eddy diffusion lying at 76.4°N, and the southern limit of horizontal diffusion lying at 73.2°N.

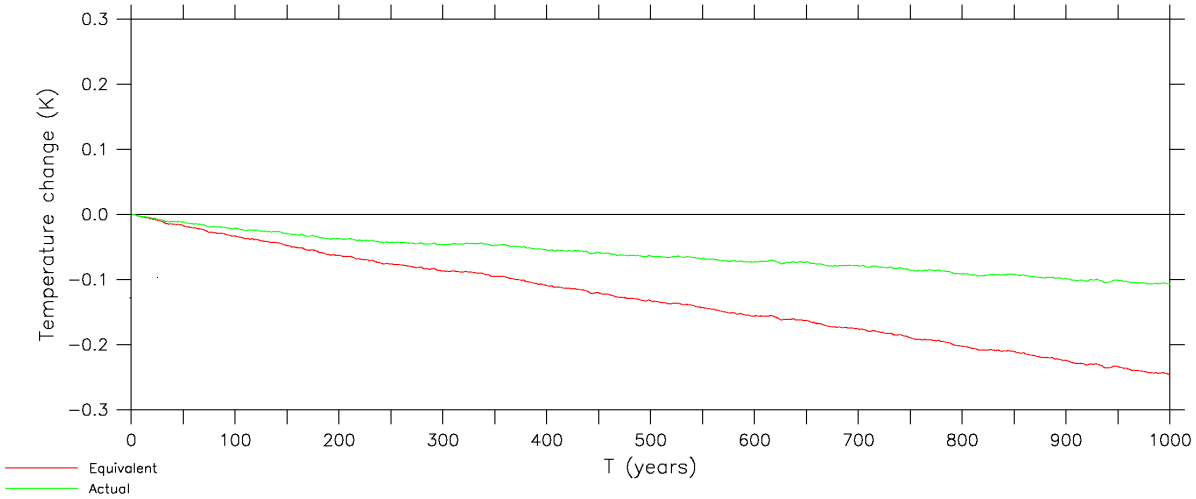
Hirst (pers. comm.) advised that this transition was introduced because of numerical instabilities caused by Gent-McWilliams eddy diffusion in an earlier version of the model, and indicated that it might no longer be required. The transition to horizontal diffusion was therefore removed, by modifying `ocean` such that `ahefac` is always equal to `1.0`, and `ahhfac` is always equal to `0.0`. No numerical instabilities were found to arise as a result of this modification, and it was found to reduce the excessive spatial variability in the surface fluxes over the Arctic Ocean (Section E.3).

Energy conservation

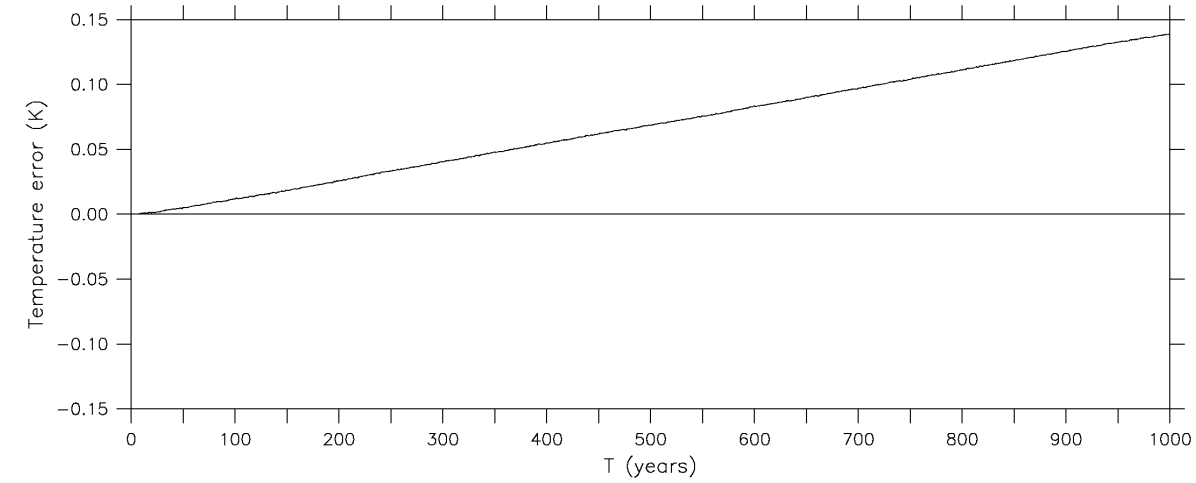
Preliminary coupled model simulations conducted using Mk3L exhibited a large energy conservation error within the ocean. Figure C.2a shows the annual-, global-mean heat flux into the ocean for a preliminary coupled model control simulation. By integrating this flux with respect to time, and dividing by the heat capacity of the ocean (which, for the bathymetry used in Mk3L, is 5.24×10^{24} JK^{-1}), the heat flux can be converted into an equivalent change in the mean temperature of the ocean. This value is plotted in Figure C.2b, along with the actual change in the ocean temperature.



(a) Average surface heat flux



(b) Change in mean ocean temperature



(c) Error in mean ocean temperature

Figure C.2: Energy conservation within the ocean for a preliminary coupled model control simulation: (a) the annual-, global-mean heat flux into the ocean, (b) the equivalent (red) and actual (green) changes in the mean temperature of the ocean, and (c) the error in the mean temperature of the ocean.

As there are no sources or sinks of energy within the model ocean, any difference between the actual and equivalent temperature changes represents an energy conservation error. This error, which amounts to +0.14 K after 1000 years, is plotted in Figure C.2c. Such an error represents a gain of 7.3×10^{23} J in the heat content of the ocean, equal to an average rate of energy gain of 2.3×10^{13} W.

Figure C.3 illustrates the lack of any equivalent conservation error during the final century of an ocean model spin-up run. Owing to the manner in which the atmosphere and ocean models are "fused" together, different paths are followed through the ocean model source code in the case of the coupled model and the stand-alone ocean model. Inspection of the source code revealed that such differences in the code paths occur within the subroutines `clinic`, `ocean`, `step` and `tracer`. There are a number of reasons for the differences:

- differing surface boundary conditions, with surface fluxes being obtained from the atmosphere model when running in coupled mode, and with climatological surface boundary conditions being read from auxiliary files when the ocean model is running in stand-alone mode
- differing timestepping arrangements
- differences in the diagnostic output

While there should clearly be no difference in the internal physics of the ocean, one such discrepancy was located. Within the subroutine `tracer`, the following loop derives updated tracers at the end of each timestep:

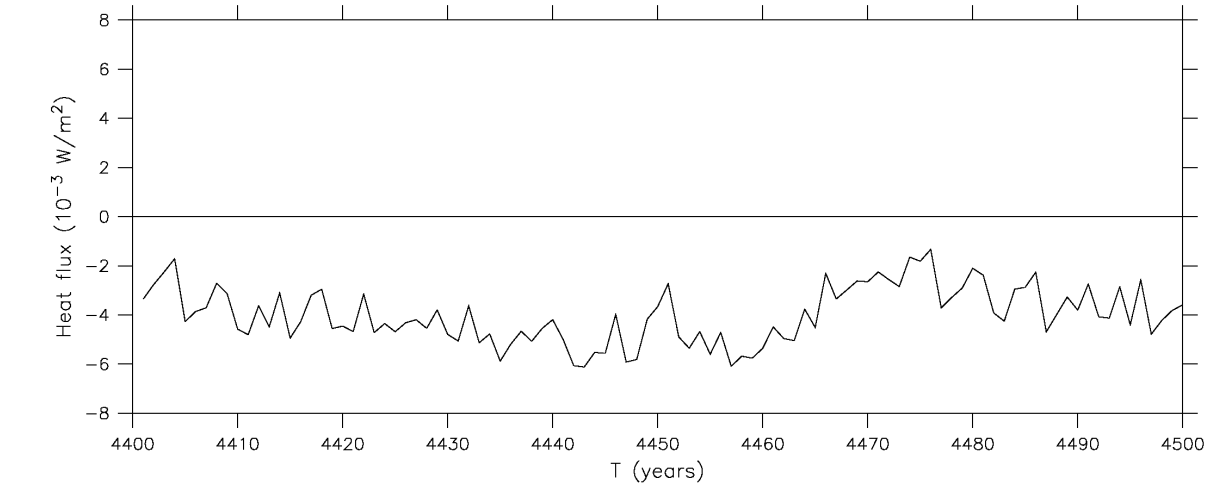
```
      DO 850 K=1,KM
      DO 850 I=1,IMT
        TA(I,K,M)=(TB(I,K,M)+C2DTTS*DTXQ(I,K)*TA(I,K,M))*FM(I,K)
        If(lcouple)Then
          tfc=tfi-273.15
          if(TA(I,k,M).lt.tfc)TA(I,k,M)=tfc-.0001
        End If
 850  CONTINUE
```

The flag `lcouple` is `.true.` in the case of the coupled model, and `.false.` otherwise. Thus the contents of the `if` construct are only executed when the ocean model is running as part of the coupled model. `tfi` is a global parameter which specifies the water temperature beneath sea ice, in Kelvin, and which is set equal to 271.3. `tfc` is the equivalent value in degrees Celsius, and is equal to -1.85.

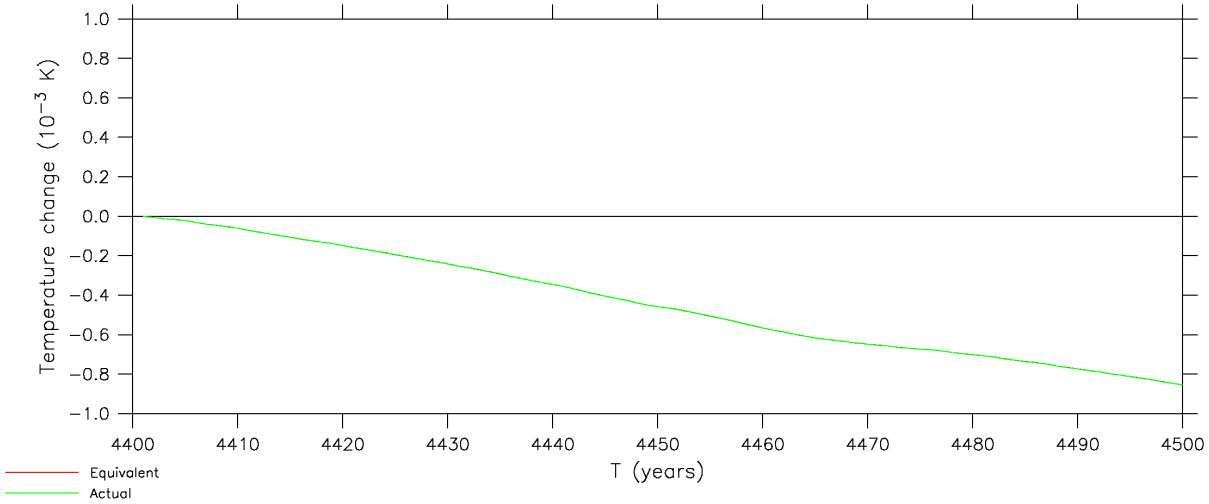
The effect of the above loop, therefore, is to reset ocean model temperatures to -1.8501°C whenever they fall below -1.85°C. There are two significant flaws with this approach:

- it is non-conserving, as the resetting of the ocean temperature represents an input of energy to the model
- it represents a difference in the model physics between the stand-alone ocean model and the coupled model

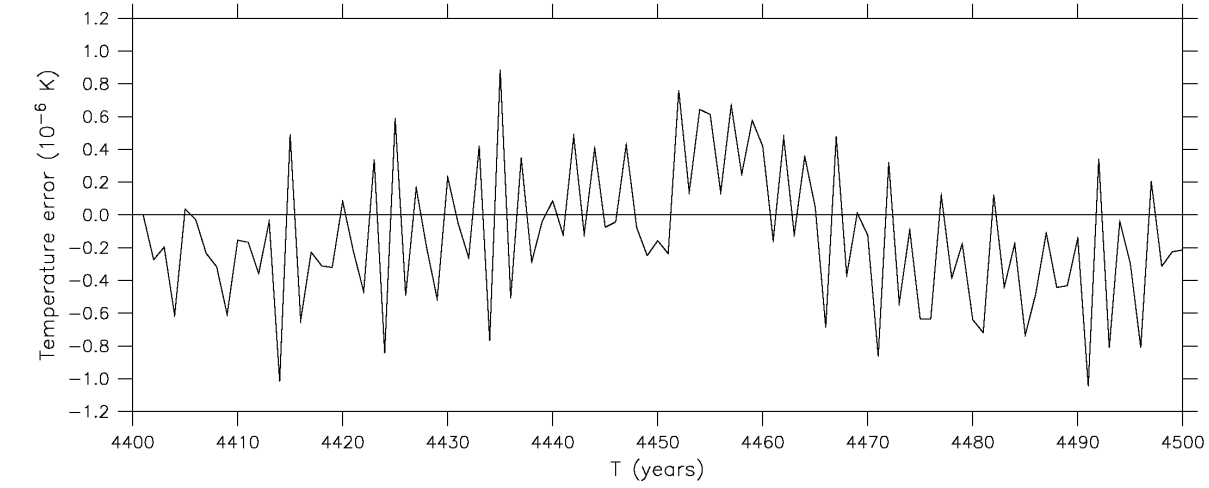
Regarding the first of these flaws, if it is necessary to modify the ocean temperature at a particular gridpoint, then it must be done in such a fashion that energy is conserved. For example, the energy added to the ocean



(a) Average surface heat flux



(b) Change in mean ocean temperature



(c) Error in mean ocean temperature

Figure C.3: Energy conservation during the final century of an ocean model spin-up run: (a) the annual-, global-mean heat flux into the ocean, (b) the equivalent (red) and actual (green) changes in the mean temperature of the ocean, and (c) the error in the mean temperature of the ocean.

at that gridpoint could be extracted from the surrounding gripdoints, which would have their temperatures reduced in order to conserve energy.

Regarding the second of these flaws, the ocean model physics should be invariant, and should not depend on the mode in which the model is running. If flux adjustments are derived on the basis that the control climate of the ocean within the coupled model should remain as close as possible to its control climate during spin-up, then the flux adjustments will only be applicable if the model physics is consistent. A change in the physics, such as that identified here, will represent a source of drift within the coupled model.

It was considered that such modifications to the ocean temperatures are unnecessary. The model already contains sufficient physics to represent the processes that will warm the ocean whenever it reaches a super-cooled state. Both the conduction of heat, and the mixing arising from the convection of dense super-cooled water to depth, will tend to warm such water. Furthermore, within the coupled model, additional sea ice will form whenever the surface waters become super-cooled, adding latent heat to the ocean and hence raising the sea surface temperature.

The ocean model was therefore modified by simply removing the `if` construct from the above section of code, ensuring consistent physics between the stand-alone ocean model and the coupled model.

C.2.5 Relaxation timescale

The model was modified to enable the user to vary the timescale used by the stand-alone ocean model to relax the sea surface temperatures and salinities towards observed values. The subroutine `ocean` was modified to read a new parameter `trelax`, as part of the existing `namelist` group `parms`. `trelax` is then made available to the subroutine `tracer` via the `common` block `trelax`.

`tracer` now calculates the relaxation coefficient `gamma` as follows:

```
gamma = 1.0 / (86400.0 * trelax)
```

Previously, `gamma` was set to the value 5.787×10^{-7} s^{-1}, corresponding to a relaxation timescale of 20 days.

C.3 I/O

C.3.1 Input files

Surface boundary conditions

Within the subroutine `step`, the auxiliary files read by the model when running in stand-alone mode were given more logical and consistent names, as shown in Table C.4.

Wind stress reduction factors

In the original version of the model, the stand-alone ocean model requires the auxiliary file `cwice.dat12`, which contains spatially-varying wind stress reduction factors. The climatological wind stresses used to

Field	Original filename	New filename
Sea surface temperature	`sstd`	`sst.dat`
Sea surface salinity	`stsall_fix6.dat`	`sss.dat`
Surface momentum flux	`hr4seas.dat`	`stress.dat`

Table C.4: The original and new names of the files containing the surface boundary conditions for the stand-alone ocean model.

force the model are multiplied by these factors, in order to allow for the effects of sea ice.

However, it was not considered to be desirable to modify the wind stresses in this fashion. The section of code within the subroutine `ocdatro`, which reads the file `cwice.dat12` and which modifies the wind stresses, was therefore removed.

The file `cwice.dat12` is also read by the original version of the coupled model, even though the values are not used. The relevant section of code was therefore removed from the subroutine `ocdatra`.

C.3.2 Output files

Model output

The original version of the ocean model generates a large number of output files. In addition to the creation of restart files at the end of each month, data is written to 14 different files (`fort.1`, `fort.32`, `fort.40`, `fort.67`, `fort.76`, `fort.81`, `fort.91`, `fort.92`, `fort.93`, `fort.94`, `fort.95`, `fort.96`, `fort.97` and `fort.99`). These files are not given explicit names, with the model relying on Fortran's implicit file naming: if data is written to unit `nn`, and this unit has not been connected to an external file using the `open` statement, then a file with the name `fort.nn` is created. It is dangerous to rely upon this behaviour, however, as it does not form part of the Fortran standard (Standards Association of Australia, 1983).

Some of these output files are binary, and some text. Data is written at varying intervals, although generally at the end of each month, and in a variety of different formats. Many of the files contain unnecessary diagnostic output; furthermore, there is considerable duplication, with some model variables being written to up to three different output files. Such unnecessary I/O will impede the runtime performance of the model.

For those ocean model variables that were deemed to be required, the source code was modified such that monthly averages are written to the single output file `fort.40`. This is a binary data file, and the `write` statements were modified such that the data is always written at 32-bit precision, irrespective of the precision used within the model. All other `write` statements, apart from those which create restart files, were removed from the source code.

In future versions of Mk3L, further modifications will be made to ocean model output, so that the model does not rely on Fortran's implicit file naming.

C.4 Utilities

C.4.1 Monthly-mean model output

A utility, `avgcnv`, is supplied with the original version of the model. This utility is written in Fortran 77, and generates a netCDF (Unidata Program Center, 2005) file containing the data written to `fort.40`. It was re-named `convert_averages` and completely rewritten, with the following modifications:

- the model output is read from the expanded output file `fort.40`
- lengths are converted from cm to m, velocities from cms^{-1} to ms^{-1}, the surface momentum fluxes from dynes/cm^2 to Nm^{-2}, the streamfunction from cm^{3}s^{-1} to Sv, and salinities from kg/kg to psu
- the horizontal velocity grid is defined within the netCDF file, in addition to the tracer grid
- missing values and the surface salinity tendencies are now treated correctly
- the source code has been converted to Fortran 90
- the netCDF Fortran interface has been updated to version 3.x, for improved runtime performance

`convert_averages` is intended to be executed at runtime, by the batch script that executes the model. This ensures that all ocean model output is stored in a single output file, and in a completely machine-independent format.

For further information on `convert_averages`, see Section 6.3.

C.4.2 Meridional overturning streamfunctions

Flows within the ocean can be regarded as being non-divergent, enabling a meridional overturning streamfunction ψ to be defined in the latitude-depth plane such that

$$V = -\frac{\partial \psi}{\partial z} \tag{C.7}$$

$$W = \frac{\partial \psi}{\partial y} \tag{C.8}$$

where V, W are the zonal integrals of the meridional velocity v and the vertical velocity w respectively.

The original version of the ocean model is supplied with a utility, `overturning`, which calculates the meridional overturning streamfunction by integrating v thus:

$$\psi = -\iint v \, dx dz \tag{C.9}$$

Three streamfunctions are calculated:

ψ	the total streamfunction
ψ_e	the streamfunction calculated from eddy-induced transport only
ψ_l	the streamfunction calculated from large-scale transport only

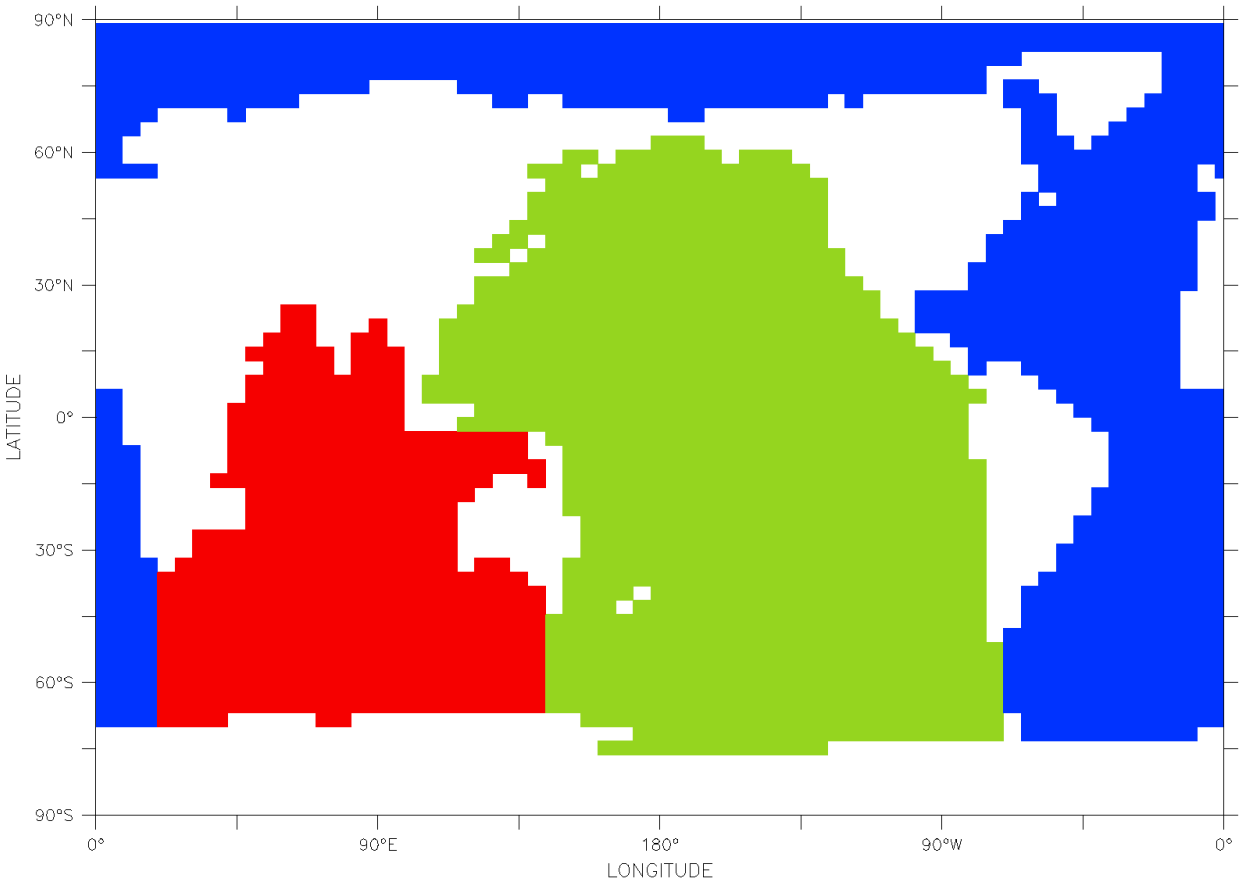


Figure C.4: The ocean basins defined for the purposes of calculating the meridional overturning streamfunctions: the Atlantic Ocean (blue), the Pacific Ocean (green), and the Indian Ocean (red).

For each of these streamfunctions, values are calculated for the Atlantic, Pacific and Indian Oceans, and for the world ocean as a whole.

The original version of `overturning` is written in Fortran 77. It was completely rewritten, with the following modifications:

- the velocities are read from the netCDF file created by `convert_averages`, whereas the original version read them from the binary output file `fort.67`
- the masks which divide the world ocean into the Atlantic, Pacific and Indian Oceans are now read from the netCDF file `bsnmask.nc`, rather than the binary file `bsnmask.cif`
- all of the streamfunctions are written to a single netCDF output file, whereas the original version wrote each streamfunction to a separate binary output file
- the source code has been converted to Fortran 90

The masks defined in `bsnmask.cif` excluded the Mediterranean, Black and Caspian Seas from the world ocean. In deriving the file `bsnmask.nc`, the masks were modified such that the Baltic Sea, Hudson Bay and the Persian Gulf are also excluded. This is appropriate, as none of these basins have any resolved connection with the world ocean within the model (Section C.2.1). The modified masks are shown in Figure C.4.

For further information on `overturning`, see Section 6.3.

C.4.3 Annual-mean model output

For most analysis purposes, annual-mean model output is required; it is therefore useful to generate annual-mean output at runtime. This has the additional benefit of reducing the volume of data to be retrieved when analysing model output.

Two new utilities, `annual_averages` and `annual_overturning`, were therefore written. These take the output of `convert_averages` and `overturning` respectively, calculate the annual means of the model output, and save the data to a single netCDF file. Each utility is capable of taking any number of years of model output, and generating a single output file.

For further information on `annual_averages` and `annual_overturning`, see Section 6.3.

Appendix D

Modifications to the coupled model

D.1 Introduction

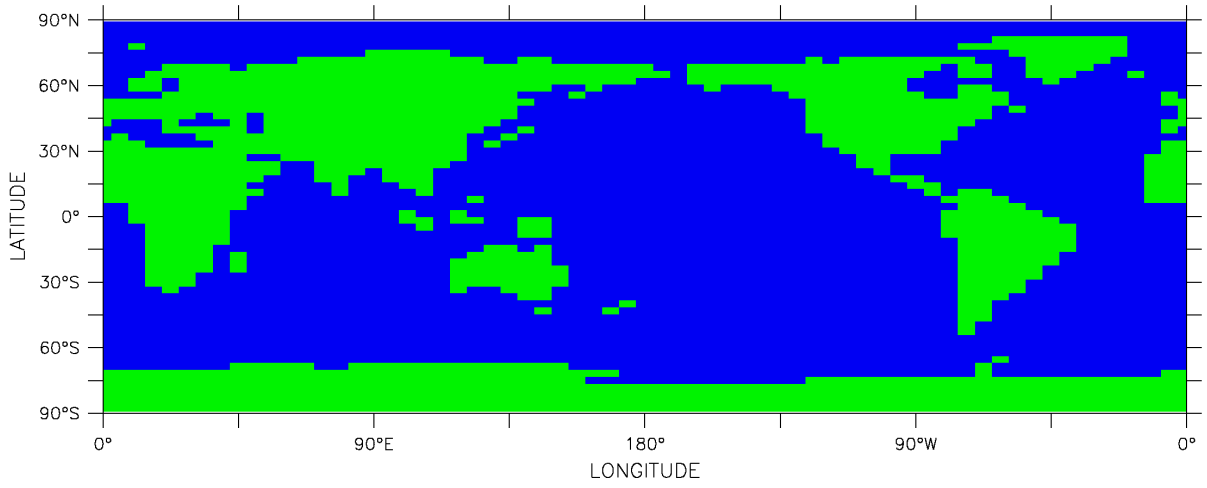
Although the atmosphere model (AGCM) and the ocean model (OGCM) use the same horizontal grid, there are differences between the land/sea masks used by the two models. These arise from deliberate modifications made to the ocean model bathymetry, in order to ensure adequate resolution of the Drake Passage, the Greenland-Scotland sill, and the flows through the Indonesian archipelago (Section 2.3).

The land/sea masks used by the two models, and the differences between them, are shown in Figure D.1. Some statistics on the two masks are shown in Table D.1, and on the differences between them in Table D.2. Of the 3,584 gridpoints which comprise the Earth's surface, there are 11 gridpoints which are treated as land by the AGCM, but as ocean by the OGCM. Conversely, there are 18 gridpoints which are treated as ocean by the AGCM, but as land by the OGCM. The total area affected amounts to slightly over 1% of the Earth's surface.

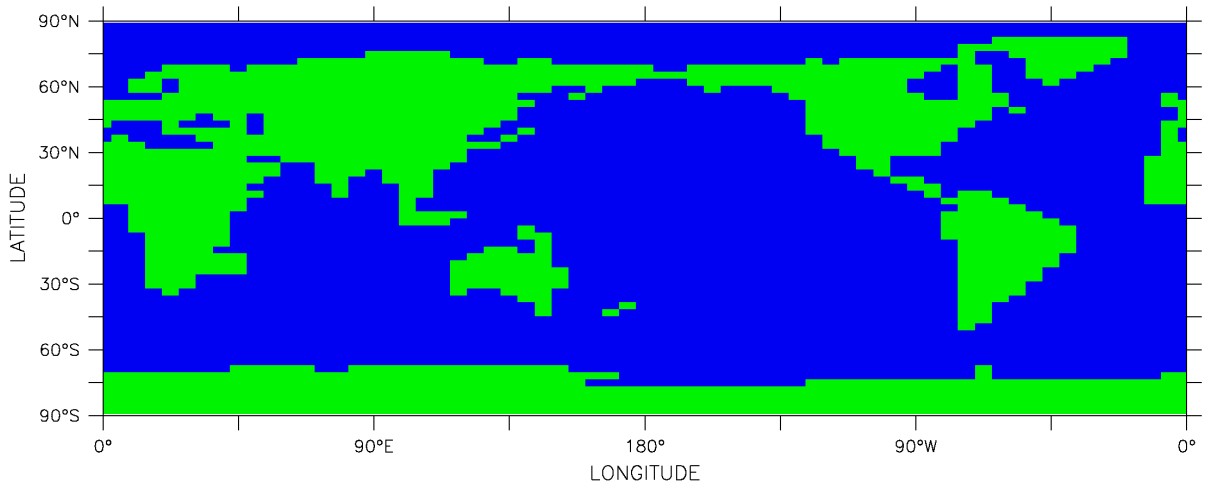
The methodology used by the original version of the model to interpolate the surface fields between the AGCM and OGCM grids is outlined in Section D.2, and examined in detail in Section D.3. The modifications which were made to the coupling during the development of Mk3L are documented in Section D.4.

Model	Number of gridpoints		Surface area of ocean (% of Earth's surface)
	Land	Ocean	
Atmosphere	1220	2364	70.61
Ocean	1227	2357	70.35

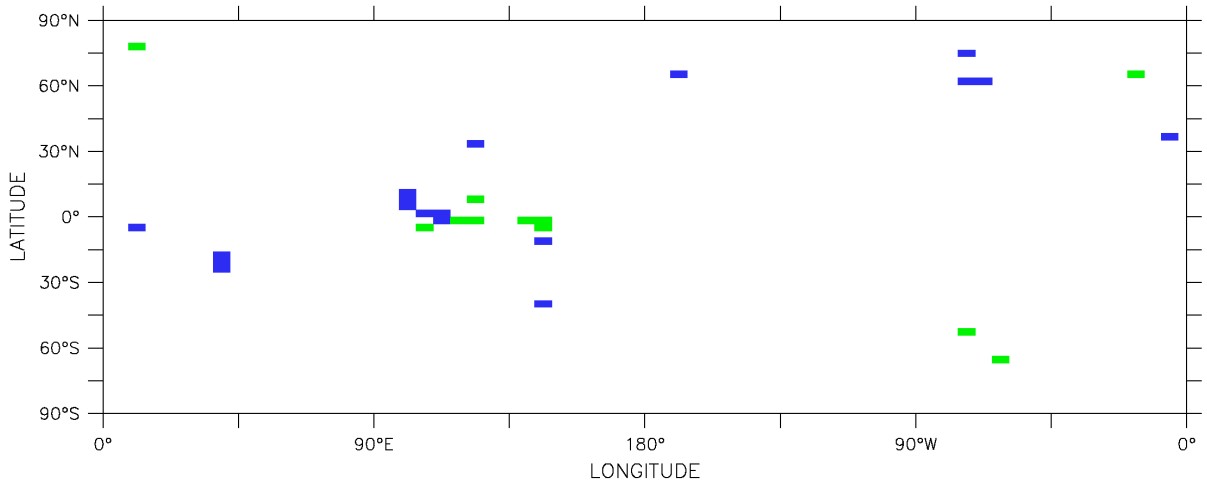
Table D.1: Statistics on the land/sea masks used by the AGCM and OGCM.



(a) AGCM



(b) OGCM



(c) Difference between AGCM and OGCM

Figure D.1: The land/sea masks: (a) AGCM, (b) OGCM, and (c) the difference between the two models. In (c), green represents points which are treated as land by the AGCM, but as ocean by the OGCM; vice versa for blue points.

Difference between masks	Number of gridpoints	Area (10^{12} m^2)	% of Earth's surface
AGCM=land OGCM=ocean	11	1.91	0.37
AGCM=ocean OGCM=land	18	3.27	0.64
Total	**29**	**5.18**	**1.02**

Table D.2: Statistics on the differences between the land/sea masks used by the AGCM and OGCM.

D.2 Coupling between the AGCM and OGCM

D.2.1 Fields passed from the AGCM to the OGCM

In the coupled model, four fields are passed from the AGCM to the OGCM: the surface heat flux, the surface salinity tendency and the zonal and meridional components of the surface momentum flux.

Surface heat flux

The interpolation of the surface heat flux onto the OGCM grid is conducted within the subroutine `ocforce`. This routine is incorporated into the AGCM, enabling the heat fluxes on the OGCM grid to be obtained even when the AGCM is being run in stand-alone mode. This is essential if flux adjustments are to be diagnosed for use within the coupled model (Section 2.5).

The method used to interpolate the surface heat flux onto the OGCM grid is *ad hoc* and is hard-coded into the model. Where there is no mismatch between the land/sea masks, the values are simply transferred from one grid to the other. For the 11 gridpoints which are treated as land by the AGCM, but as ocean by the OGCM, values must be estimated. The approach taken for each of these gridpoints differs, and is shown in Table D.3. Wherever possible, the average is taken of the values for the neighbouring gridpoints to the east and west, avoiding errors associated with strong latitudinal temperature gradients; this approch is employed for six of the 11 gridpoints. For the remaining five gridpoints, all of which lie within the Indonesian region, a nearest-neighbour approach is taken.

Surface salinity tendency

The use of the rigid-lid boundary condition within the OGCM (Section A.5), which does not allow for any variation in the thickness of the upper layer of the ocean, prevents a freshwater flux from being applied directly to the ocean. Instead, the surface freshwater flux must be converted into an equivalent rate of change in the salinity of the upper layer of the ocean, which shall be referred to herein as the *surface salinity tendency*.

The method used to convert a freshwater flux into an equivalent surface salinity tendency is as follows. Take a layer of water of salinity S_O and thickness Δz, and apply a flux of water of salinity S_A at a rate F. After an increment of time dt, which is sufficiently short that $Fdt \ll \Delta z$, the total depth of water will be $(\Delta z + Fdt)$, and the mean salinity will be

Target gridpoint		Interpolation method	Source gridpoint(s)	
(i, j)	λ, ϕ		(i, j)	λ, ϕ
(54, 8)	62°W, 65°S	E-W interpolation	(53, 8)	68°W, 65°S
			(55, 8)	56°W, 65°S
(52, 12)	73°W, 53°S	E-W interpolation	(51, 12)	79°W, 53°S
			(53, 12)	68°W, 53°S
(20, 27)	107°E, 5°S	E-W interpolation	(19, 27)	101°E, 5°S
			(21, 27)	113°E, 5°S
(27, 27)	146°E, 5°S	nearest neighbour to E	(28, 27)	152°E, 5°S
(22, 28)	118°E, 2°S	nearest neighbour to S	(22, 27)	118°E, 5°S
(23, 28)	124°E, 2°S	nearest neighbour to E	(24, 28)	129°E, 2°S
(26, 28)	141°E, 2°S	nearest neighbour to W	(25, 28)	135°E, 2°S
(27, 28)	146°E, 2°S	nearest neighbour to E	(28, 28)	152°E, 2°S
(23, 31)	124°E, 8°N	E-W interpolation	(22, 31)	118°E, 8°N
			(24, 31)	129°E, 8°N
(62, 49)	17°W, 65°N	E-W interpolation	(61, 49)	23°W, 65°N
			(63, 49)	11°W, 65°N
(3, 53)	11°E, 78°S	E-W interpolation	(2, 53)	6°E, 78°N
			(4, 53)	17°E, 78°N

Table D.3: The method used to estimate surface fluxes for those gridpoints which are treated as land by the AGCM, but as ocean by the OGCM.

$$S'_O = \frac{\Delta z S_O + F dt S_A}{\Delta z + F dt} \tag{D.1}$$

The change in salinity is therefore

$$dS_O = S'_O - S_O = \frac{\Delta z S_O + F dt S_A - \Delta z S_O - F dt S_O}{\Delta z + F dt} \tag{D.2}$$

$$= \frac{F dt}{\Delta z + F dt}(S_A - S_O) \tag{D.3}$$

However, $F dt \ll \Delta z$, and so the rate of change of salinity can be approximated as

$$\frac{dS_O}{dt} \approx \frac{(S_A - S_O)}{\Delta z} F \tag{D.4}$$

If Δz is taken as representing the thickness of the upper layer of the ocean model, and S_O the salinity, then Equation D.4 enables a freshwater flux F to be converted into an equivalent surface salinity tendency.

The surface freshwater flux comprises four components, listed in Table D.4. As with the surface heat flux, the interpolation onto the OGCM grid is conducted within the AGCM subroutine `ocforce`, enabling flux adjustments to be diagnosed for use within the coupled model. Each component of the freshwater flux is then converted into an equivalent surface salinity tendency, and the salinity tendencies summed to obtain the total surface salinity tendency, which is passed to the OGCM.

In calculating the equivalent surface salinity tendencies, the model assumes that sea ice has a constant

Component	Description	$\mathbf{S_A}$ (psu)	$\mathbf{S_O}$ (psu)	$\mathbf{S_A - S_O}$ (psu)
F_{p-e}	Precipitation minus evaporation	0	35	-35
F_{ice}	Change in sea ice volume	10	35	-25
F_{sub}	Sublimation of sea ice	10	0	+10
F_{run}	Run-off	0	35	-35

Table D.4: The components of the surface freshwater flux, and the salinities used to calculate an equivalent surface salinity tendency.

salinity of 10 psu, and that the ocean has a constant salinity of 35 psu. Note that in the case of F_{sub}, the flux arising from the sublimation of sea ice, S_O is set equal to zero. In this case, the flux represents the rate at which sea ice sublimes. The sublimation gives rise to a flux of pure salt into the ocean, which can be regarded as having zero thickness. The term Fdt therefore disappears from the denominator of Equation D.1, and Equation D.4 becomes

$$\frac{dS_O}{dt} = \frac{S_A}{\Delta z} F_{sub} \tag{D.5}$$

The method used to interpolate F_{p-e}, F_{ice} and F_{sub} onto the OGCM grid is exactly the same as for the surface heat flux.

The run-off must be handled differently, however, as this field is calculated over land by the AGCM, rather than over the ocean. Within `ocforce`, the run-off is relocated to the ocean using a simple method in which the run-off at each land gridpoint is instantaneously spread amongst any neighbouring gridpoints which lie at a lower elevation. This relocation is performed repeatedly, until all the run-off has reached the ocean. These calculations are performed on the OGCM grid, ensuring that when the run-off reaches the ocean, it will always arrive at a gridpoint which is treated as ocean by the OGCM.

This requires that the run-off be interpolated onto the OGCM grid, prior to its downslope relocation. In the original version of the model, however, this interpolation is not performed, meaning that the run-off at those 11 gridpoints which are treated as land by the AGCM, but as ocean by the OGCM, is lost. The run-off at those 18 gridpoints which are treated as ocean by the AGCM, but as land by the OGCM, is treated as being zero.

Surface momentum flux

The interpolation of the zonal and meridional components of the surface momentum flux onto the OGCM grid is conducted within the subroutine `ocntau`. Again, this routine is incorporated into the AGCM, enabling flux adjustments to be diagnosed for use within the coupled model.

The surface momentum flux calculated by the AGCM has two components: the atmosphere-ocean momentum flux, and the sea ice-ocean momentum flux. The method used to interpolate the atmosphere-ocean flux onto the OGCM grid is the same as for the surface heat flux and the surface salinity tendency. However, an additional step must be carried out. The OGCM uses the “B-grid” configuration of Arakawa and Lamb (1977), in which the tracer gridpoints are located at the centres of the gridboxes, and the horizontal velocity gridpoints are located at the corners. The AGCM calculates the atmosphere-ocean momentum flux

at the centre of each gridbox, while the OGCM requires that the momentum flux be specified at the velocity gridpoints. The fluxes at the corners of each gridbox must therefore be estimated by interpolation.

The velocity gridpoints are offset to the north-east of the tracer gridpoints; thus, if $\tau_{i,j}$ represents the momentum flux at the tracer gridpoint (i, j), the estimated flux at the velocity gridpoint (i, j) is given by

$$\tau'_{i,j} = \frac{\tau_{i,j} + \tau_{i+1,j} + \tau_{i,j+1} + \tau_{i+1,j+1}}{4} \tag{D.6}$$

The AGCM calculates the sea ice-ocean momentum flux on an Arakawa B-grid, and these values can therefore be passed directly to the OGCM.

D.2.2 Fields passed from the OGCM to the AGCM

In the original version of the coupled model, three fields are passed from the OGCM to the AGCM: the sea surface temperature, and the zonal and meridional components of the surface velocity.

Sea surface temperature

The interpolation of the sea surface temperature from the OGCM grid to the AGCM grid takes place within the subroutine `step0`, which is incorporated into the ocean model. As with the interpolation in the opposite direction, the method is *ad hoc* and is hard-coded into the model. The approach taken for those gridpoints which are treated as land by the OGCM, but as ocean by the AGCM, is shown in Table D.5. In descending order of preference, the approach employed is:

- interpolation in the east-west direction
- interpolation in the north-south direction
- nearest neighbour interpolation

Surface velocity

The ocean currents are only used by the sea ice component of the atmosphere model. These values are passed directly from the OGCM to the AGCM, without any interpolation taking place. While this approach is not entirely satisfactory, as there will therefore be some ocean gridpoints on the AGCM grid for which no ocean currents are available, there is relatively little mismatch between the AGCM and OGCM grids at high latitudes (Figure D.1).

D.3 Assessing the coupling

D.3.1 Interpolation

The interpolation methodology outlined in Section D.2 is inadequate in two regards:

Target gridpoint(s)		Interpolation method	Source gridpoint(s)	
(i, j)	λ, ϕ		(i, j)	λ, ϕ
(27, 16)	146°E, 40°S	E-W interpolation	(26, 16)	141°E, 40°S
			(28, 16)	152°E, 40°S
(8, 21)	39°E, 24°S	N-S interpolation	(8, 20)	39°E, 27°S
(8, 22)	39°E, 21°S		(8, 24)	39°E, 14°S
(8, 23)	39°E, 18°S			
(27, 25)	146°E, 11°S	E-W interpolation	(26, 25)	141°E, 11°S
			(28, 25)	152°E, 11°S
(3, 27)	11°E, 5°S	nearest neighbour to W	(2, 27)	6°E, 5°S
(21, 28)	113°E, 2°S	N-S interpolation	(21, 27)	113°E, 5°S
(21, 29)	113°E, 2°N		(21, 30)	113°E, 5°N
(20, 29)	107°E, 2°N	nearest neighbour to N	(20, 30)	107°E, 5°N
(19, 30)	101°E, 5°N	E-W interpolation	(18, 30)	96°E, 5°N
			(20, 30)	107°E, 5°N
(19, 31)	101°E, 8°N	E-W interpolation	(18, 31)	96°E, 8°N
			(20, 31)	107°E, 8°N
(19, 32)	101°E, 11°N	nearest neighbour to W	(18, 32)	96°E, 11°N
(23, 39)	124°E, 33°N	N-S interpolation	(23, 38)	124°E, 30°N
			(23, 40)	124°E, 37°N
(64, 40)	6°W, 37°N	E-W interpolation	(63, 40)	11°W, 37°N
			(1, 40)	0°, 37°N
(52, 48)	73°W, 62°N	E-W interpolation	(51, 48)	79°W, 62°N
(53, 48)	68°W, 62°N		(54, 48)	62°W, 62°N
(35, 49)	169°W, 65°N	N-S interpolation	(35, 48)	169°W, 62°N
			(35, 50)	169°W, 68°N
(52, 52)	73°W, 75°N	E-W interpolation	(51, 52)	79°W, 75°N
			(53, 52)	68°W, 75°N

Table D.5: The method used to estimate sea surface temperatures for those gridpoints which are treated as land by the OGCM, but as ocean by the AGCM.

- no attempt is made to ensure conservation of the total fluxes of either heat, freshwater or momentum
- in the case of the run-off and the surface velocity, the interpolation fails even to provide values for all of the gridpoints on the target grid

Of these, the failure to conserve physical quantities is particularly significant, as will now be shown by considering each of the fields in turn.

Conservation and the surface heat flux

If F_A is the surface heat flux, as calculated by the AGCM, then the total heat flux Φ_A can be calculated by intregating over the 2,364 gridpoints which are treated as ocean by the AGCM:

$$\Phi_A = \int\int_{AGCM\ ocean} F_A \, dxdy \tag{D.7}$$

Likewise, if F_O is the surface heat flux on the OGCM grid, after the above interpolation operations have been carried out, then the total heat flux Φ_O can be calculated by integrating over the 2,357 gridpoints which are treated as ocean by the OGCM:

$$\Phi_O = \int\int_{OGCM\ ocean} F_O \, dxdy \tag{D.8}$$

If Φ_O and Φ_A differ, then the interpolation operations have not conserved the total heat flux from the atmosphere to the ocean. For the 2,346 gridpoints which are treated as ocean by both the AGCM and the OGCM, $F_A = F_O$. Any failure of conservation is therefore due to the 29 gridpoints where there is a mismatch between the two land/sea masks. If an interpolation error $d\Phi = \Phi_O - \Phi_A$ is defined, then $d\Phi$ has two components:

- the loss of the heat flux at those 18 gridpoints which are treated as ocean by the AGCM, but as land by the OGCM
- the gain of the estimated heat flux at those 11 gridpoints which are treated as land by the AGCM, but as ocean by the OGCM

To investigate this further, one year of a coupled model control simulation was repeated, with the surface fields on the source grids being saved at each timestep. This data could then be used to reproduce the values that would have been produced on the target grids, had the original interpolation methodology, as outlined in Section D.2, been used.

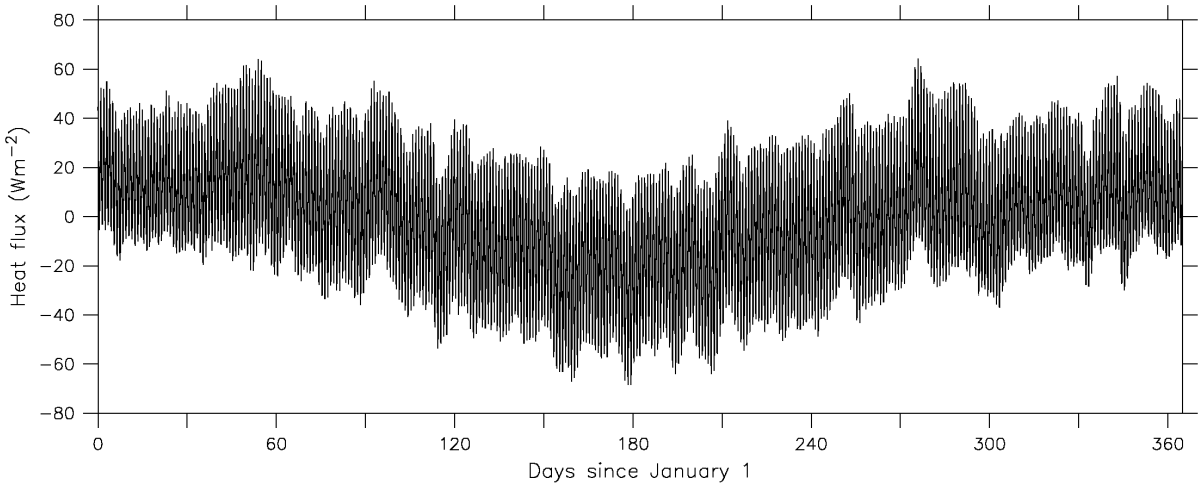
The total heat flux Φ_A, and the interpolation error $d\Phi$, at each timestep are shown in Figures D.2a and D.2b respectively. Both quantities have been normalised by the surface area of the ocean. The temporal integral of the interpolation error is shown in Figure D.2c.

The interpolation error varies between -3.38 and +2.10 Wm^{-2}, with an annual-mean value of -0.39 Wm^{-2}; the mean errors for each month are shown in Table D.6. The total interpolation error for the year is -4.37$\times$10^{21} J, equivalent to a change in the mean temperature of the ocean of -8.3$\times$10^{-4} K. Over the course of a one thousand-year simulation, the interpolation error is therefore potentially equivalent to a decrease in the mean temperature of the ocean of $\sim$0.8 K.

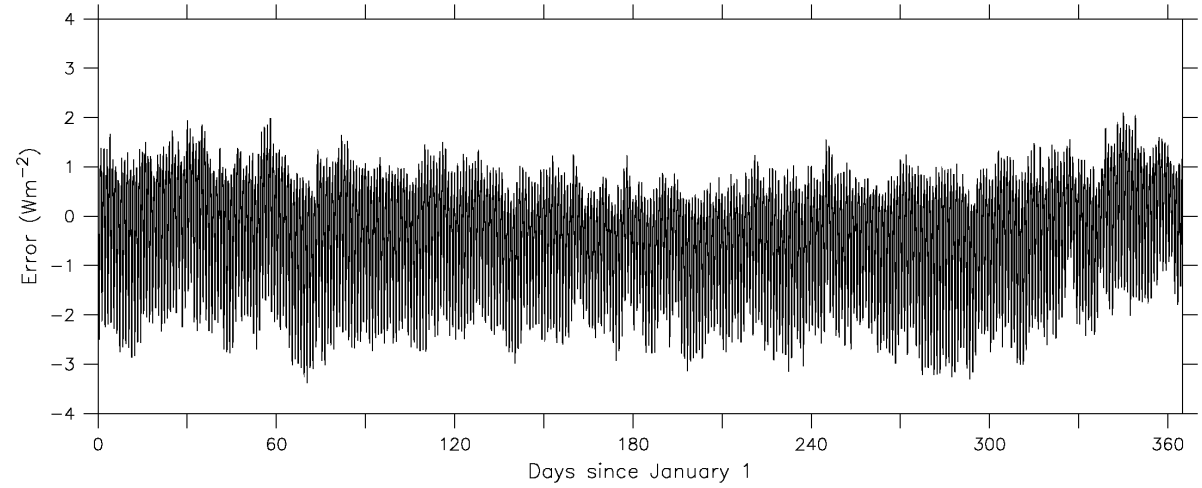
Figure D.3 shows the contributions to the mean interpolation error of each of the 29 gridpoints where there is a mismatch between the AGCM and OGCM land/sea masks. The bulk of the error (75%) arises between 30°S and 30°N; although only 18 of the 29 gridpoints lie between these latitudes, the larger areas represented by these gridpoints mean that they account for 76% of the mismatch between the two land/sea masks.

Conservation and the surface freshwater flux

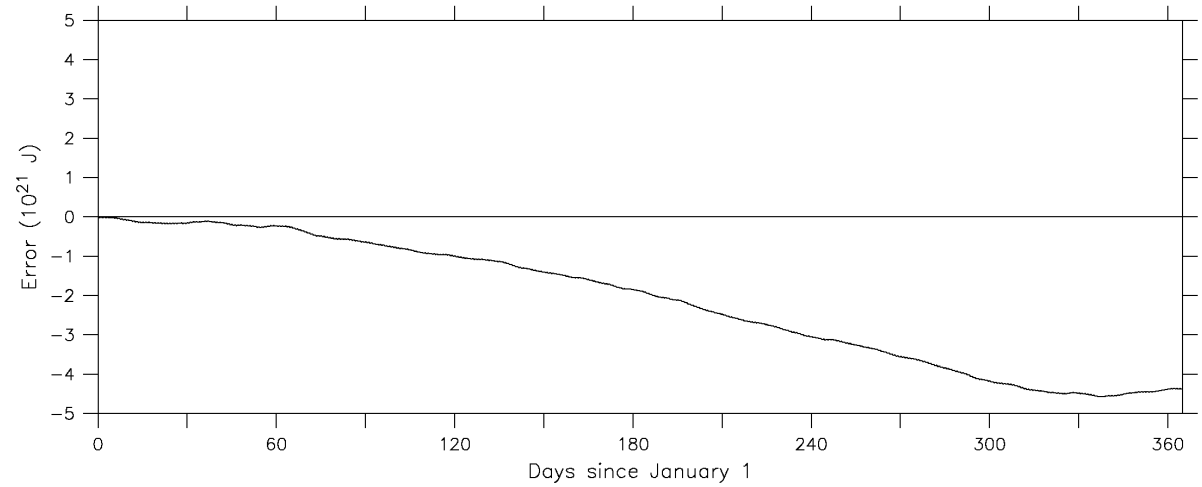
The surface freshwater flux also gives rise to large interpolation errors. Figure D.4 shows the total flux of precipitation minus evaporation, the interpolation error, and the temporal integral of the interpolation error. All quantities have been normalised by the surface area of the ocean. The total interpolation error for the year amounts to a net flux of freshwater into the ocean of 5.6 mm.



(a) Surface heat flux



(b) Interpolation error



(c) Integrated interpolation error

Figure D.2: The surface heat flux for one year of a coupled model control simulation: (a) the average heat flux on the AGCM grid, (b) the interpolation error, normalised by the surface area of the ocean, and (c) the temporal integral of the interpolation error.

Month	Interpolation error (Wm^{-2})
January	-0.15
February	-0.10
March	-0.43
April	-0.38
May	-0.44
June	-0.48
July	-0.69
August	-0.59
September	-0.54
October	-0.66
November	-0.32
December	+0.16
Annual mean	**-0.39**

Table D.6: The mean interpolation errors for the surface heat flux, diagnosed from one year of a coupled model control simulation.

Component	Annual interpolation error (10^{12} m^3)	Equivalent change in mean ocean salinity (10^{-3} psu)
F_{p-e}	+1.99	-0.054
F_{ice}	+1.02	-0.020
F_{sub}	-0.01	-0.000
F_{run}	-1.31	+0.036
Total		**-0.038**

Table D.7: The annual interpolation errors, and the equivalent changes in the mean salinity of the ocean, for the components of the surface freshwater flux, diagnosed from one year of a coupled model control simulation.

The interpolation errors for each of the four components of the surface freshwater flux are shown in Table D.7. Note that, in the case of the run-off, the total fluxes Φ_A and Φ_O, from which the interpolation error $d\Phi$ is derived, are calculated by integrating over land, rather than the ocean. The equivalent changes in the mean salinity of the ocean are also given. These can be calculated using Equation D.4, taking $S_O = 34.7$ psu as being the mean salinity of the ocean and using $\Delta z = 3565$ m, which is the mean depth of the ocean according to the model bathymetry. The total interpolation error amounts to a potential decrease of $\sim$0.04 psu in the mean salinity of the ocean over the course of a one thousand-year simulation.

Conservation and the sea surface temperature

The interpolation of sea surface temperature (SST) from the OGCM grid to the AGCM grid is also non-conserving. Although the sea surface temperature is not a flux field, and hence there is no total flux to be conserved, an interpolation error can still be defined in terms of the global mean. If T_O is the mean SST on the OGCM grid, and T_A is the mean SST on the AGCM grid, then the interpolation error can be defined as $dT = T_A - T_O$. Figure D.5 shows this error at each timestep.

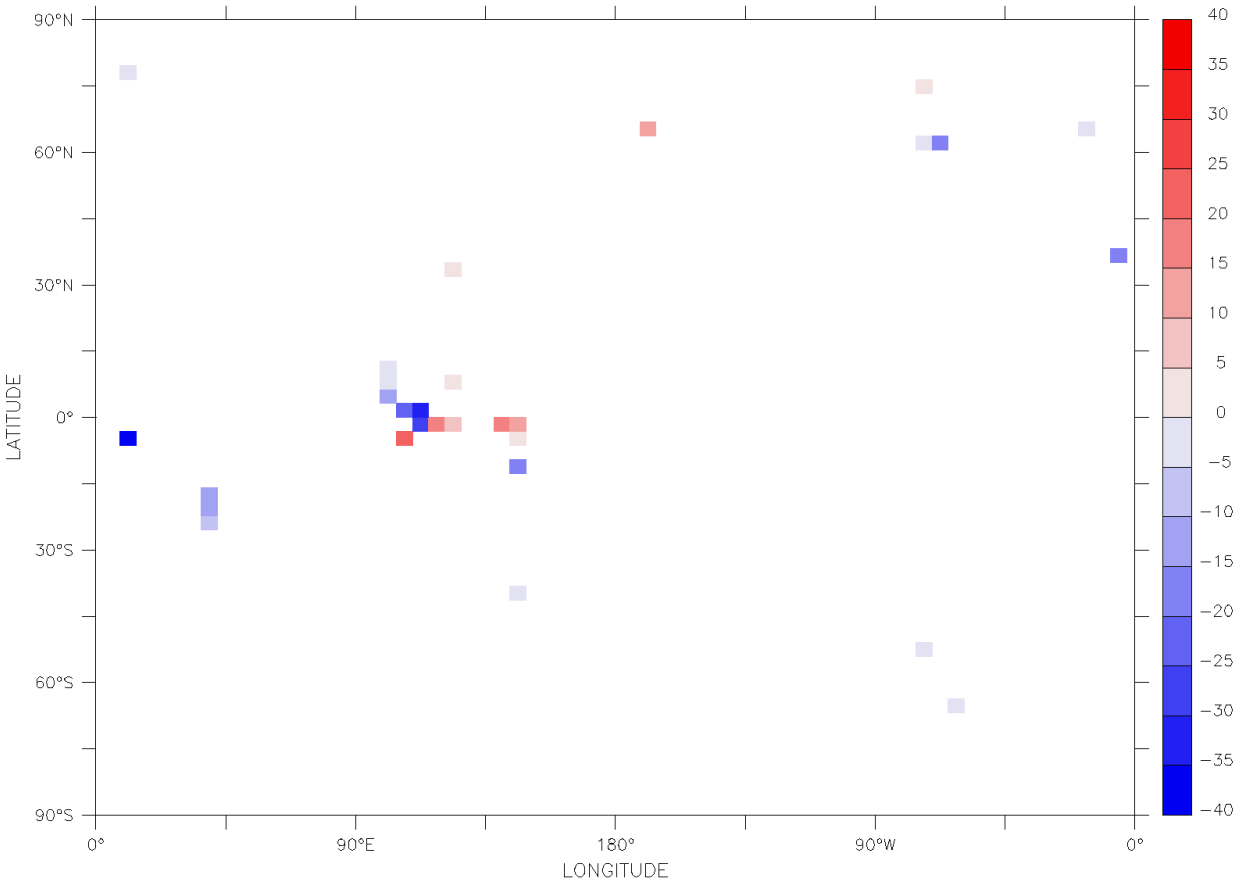


Figure D.3: The annual-mean contributions to the interpolation error for the surface heat flux, diagnosed from one year of a coupled model control simulation (in units of 10^{12} W).

While the errors can be seen to be small, not exceeding 0.007 K in magnitude, the annual-mean error of +0.003 K nonetheless represents a slight warm bias. The lack of an apparent annual cycle, however, suggests that there is interannual variability in the magnitude of the interpolation error, and that these values may not therefore be typical of the long-term mean.

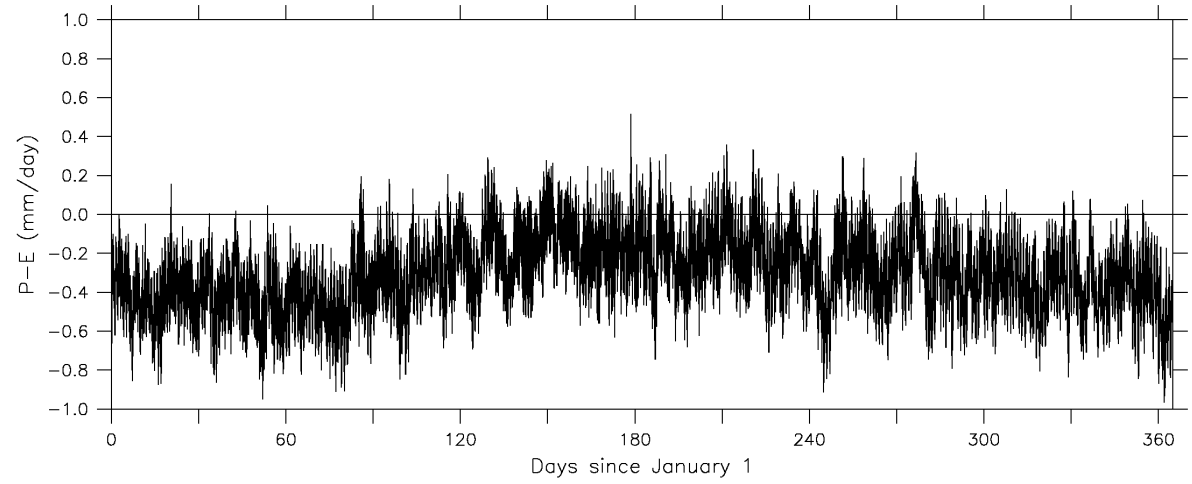
D.3.2 The surface salinity tendency

In addition to the problems regarding the interpolation between the AGCM and OGCM grids, there are further problems with regard to the method used to convert the components of the surface freshwater flux to equivalent surface salinity tendencies (Section D.2.1). This method fails to conserve freshwater on either a local or a global basis.

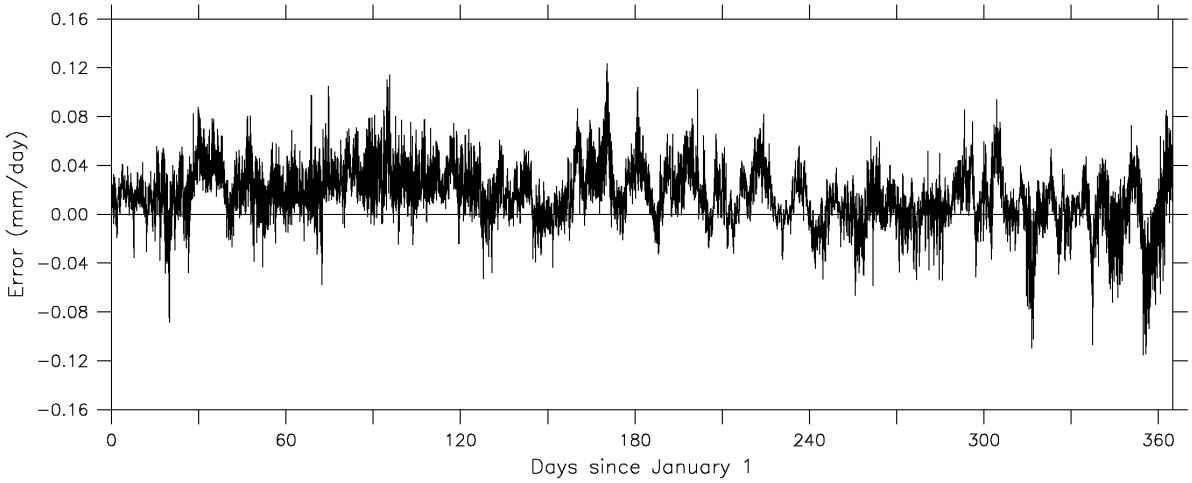
Local conservation of freshwater

The conversion of the surface freshwater flux to an equivalent surface salinity tendency is not exact, as the term Fdt is neglected in the derivation of Equation D.4. Furthermore, the use of a uniform value of S_O = 35 psu in Equation D.4 can lead to very large local freshwater conservation errors.

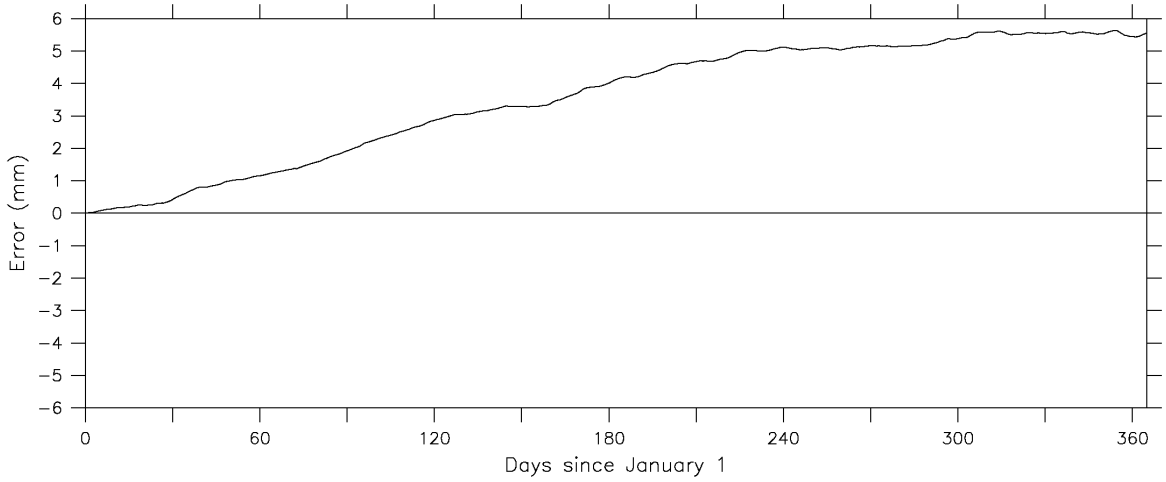
Consider climatological values for the sea surface salinity (Section E.3). $S_O \approx 35$ psu is generally a reasonable approximation, with the global- and annual-mean sea surface salinity (SSS) being equal to 34.68 psu,



(a) Precipitation minus evaporation



(b) Interpolation error



(c) Integrated interpolation error

Figure D.4: Precipitation minus evaporation for one year of a coupled model control simulation: (a) the average freshwater flux on the AGCM grid, (b) the interpolation error, normalised by the surface area of the ocean, and (c) the temporal integral of the interpolation error, also normalised by the surface area of the ocean.

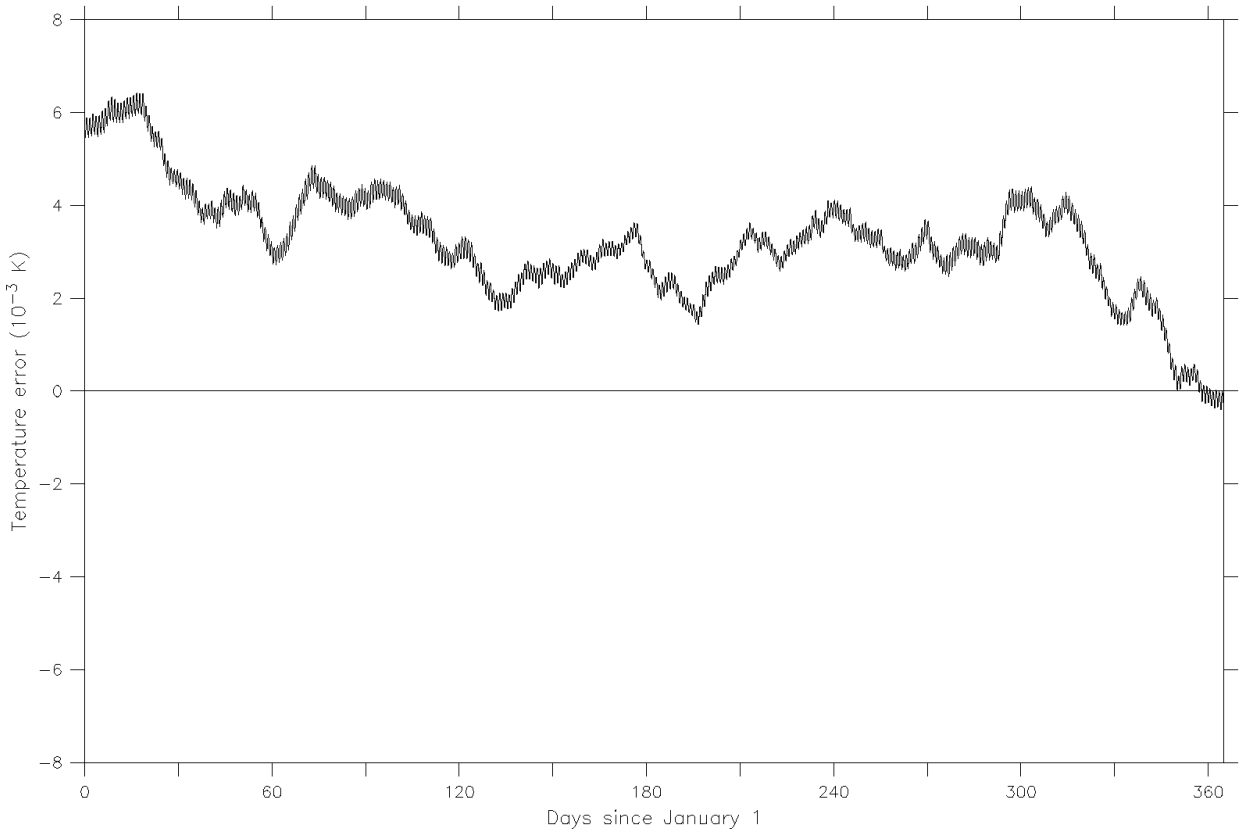


Figure D.5: The interpolation error for the sea surface temperature, diagnosed from one year of a coupled model control simulation.

and with the annual-mean SSS exceeding 30 psu over 99.0% of the surface of the ocean.

However, lower salinities are encountered, particularly in the Arctic Ocean, and in the Baltic, Black and North Seas. This is of particular concern in the case of the Baltic and Black Seas as, in the original version of the model, these seas are not connected to the world ocean (Section C.2.1). This removes the only feedback mechanism which has the potential to alleviate any local freshwater conservation errors, and can therefore lead to an ongoing and unconstrained drift in the salinity of these seas.

The Baltic Sea has the lowest SSSs, with the monthly means varying from 5.76 psu in July to 6.12 psu in January (World Ocean Atlas 1998, National Oceanographic Data Center, 2002, after interpolation onto the OGCM grid). Consider therefore the fluxes of freshwater into the Baltic Sea. Table D.8 shows the values of $(S_A - S_O)$ used to calculate the surface salinity tendency, both for the value of $S_O = 35$ psu used within the model, and for a value of $S_O = 6$ psu, which is typical for the Baltic Sea.

For F_{p-e} and F_{run}, it can be seen that $S_O = 35$ psu causes the surface salinity tendency to be over-estimated by a factor of $\sim$6. An even larger error arises in the case of F_{ice}; not only is the magnitude of the surface salinity tendency over-estimated by a factor of $\sim$6, but it also has the wrong sign.

Global conservation of freshwater

By applying Equation D.4 to the ocean as a whole, rather than simply to the upper level of the model, it can be seen that the value of S_O should reflect the mean salinity of the ocean at each timestep. Otherwise, the

Component	$\mathbf{S_A}$ (psu)	$\mathbf{S_A - S_O}$ (psu)	
		$\mathbf{S_O}$ = 35 psu	$\mathbf{S_O}$ = 6 psu
F_{p-e}	0	-35	-6
F_{ice}	10	-25	+ 4
F_{sub}	10	+10	+10
F_{run}	0	-35	-6

Table D.8: The values of $(S_A - S_O)$ for the components of the surface freshwater flux, for the cases $S_O = 35$ psu and $S_O = 6$ psu.

resulting change in the mean salinity will not exactly reflect any net flux of freshwater into the ocean.

Consider Φ_O, the total freshwater flux into the ocean as defined by Equation D.8. Integrating Equation D.4 over the surface of the ocean, Φ_O can be converted into an equivalent rate of change in the mean salinity of the ocean:

$$\left.\frac{dS_{ocean}}{dt}\right|_{fluxes} \approx \frac{(S_A - S_{ocean})}{V}\Phi_O \tag{D.9}$$

where V is the volume of the ocean.

However, let dS_O/dt be the surface salinity tendency at each gridpoint, as derived by applying Equation D.4 to the upper level of the ocean model. The equivalent rate of change in the mean salinity of the ocean is given by

$$\left.\frac{dS_{ocean}}{dt}\right|_{tendencies} = \frac{\Delta z}{V}\iint_{ocean}\frac{dS_O}{dt}\,dxdy \tag{D.10}$$

where Δz is the thickness of the upper layer of the ocean model.

If freshwater is conserved, then the rate of change in the mean salinity of the model ocean at each timestep will be equal to the value given by Equation D.9, which represents the salinity change arising from any net flux of freshwater into the ocean. However, the actual rate of change in the mean salinity of the model ocean, which arises from the surface salinity tendencies calculated by the AGCM, is given by Equation D.10. Any difference between these two values represents a freshwater conservation error, which can be expressed as

$$d\left(\frac{dS_{ocean}}{dt}\right) = \left.\frac{dS_{ocean}}{dt}\right|_{tendencies} - \left.\frac{dS_{ocean}}{dt}\right|_{fluxes} \tag{D.11}$$

Using the surface fields from the one year of coupled model output which was analysed in Section D.3.1, Figure D.6a shows the temporal integrals of Φ_O for each of the four components of the surface freshwater flux. Figure D.6b shows the equivalent changes in the mean salinity of the ocean, as derived using Equations D.9 (red) and D.10 (green). The value used for S_{ocean} in the right-hand side of Equation D.9 is 34.52 psu, being the mean salinity of the ocean for this year. The error in the mean salinity of the ocean is shown in Figure D.6c.

The total error in the mean salinity of the ocean after one year is 1.05×10^{-6} psu, indicating that the magnitude of the conservation error is of $O(10^{-6})$ psu/year. The conservation error has two sources: the

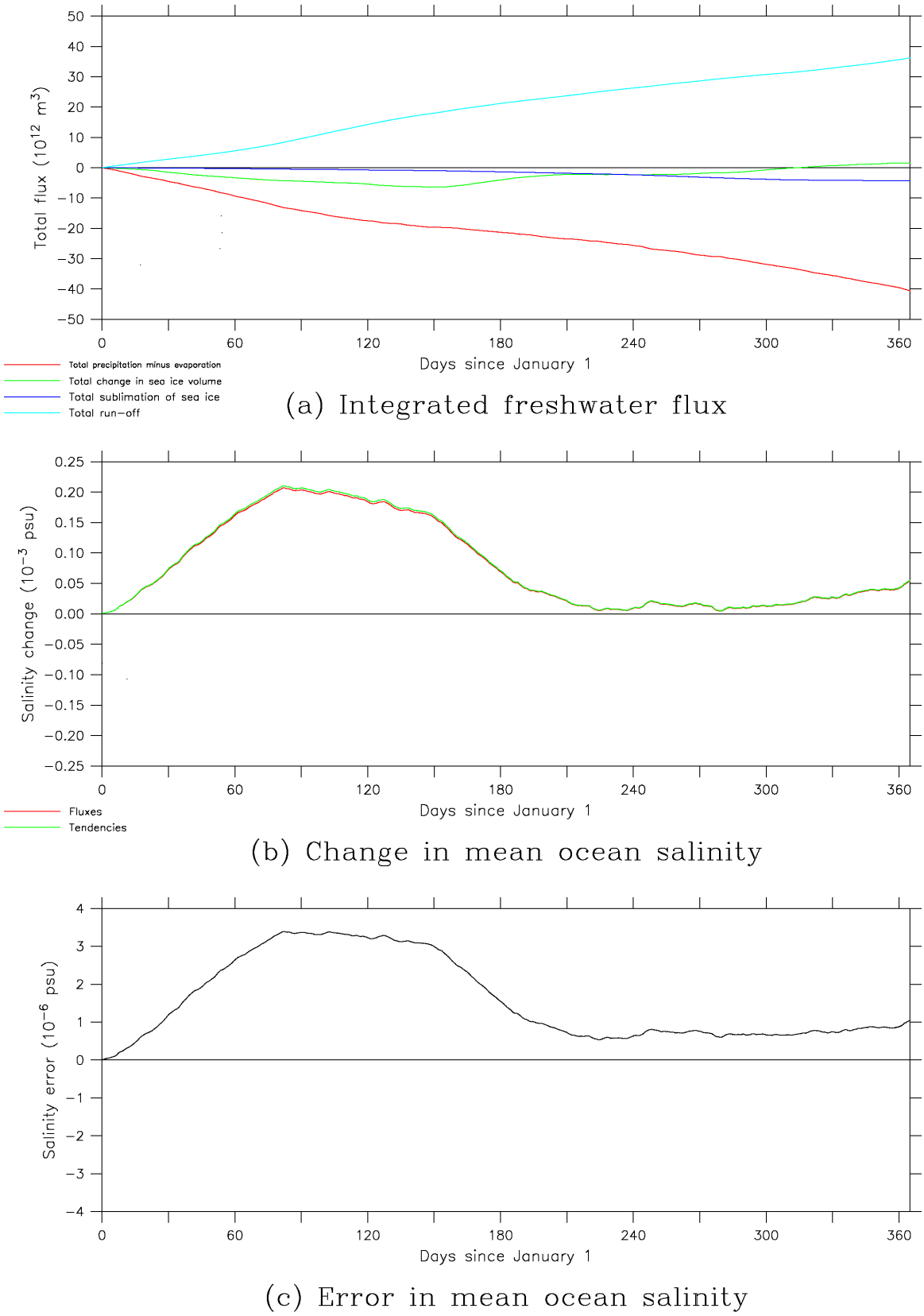


Figure D.6: The changes in the mean salinity of the ocean, diagnosed from one year of a coupled model control simulation: (a) the components of the surface freshwater flux - precipitation minus evaporation (red), change in sea ice volume (green), sublimation of sea ice (dark blue) and run-off (light blue), (b) the changes in the mean salinity derived from the freshwater fluxes (red), and from the surface salinity tendencies (green), and (c) the error in the mean salinity.

approximation used to derive Equation D.4, and the fact the uniform value of S_O = 35 psu used in the model does not accurately reflect the mean salinity of the ocean. The surface salinity tendency given by Equation D.4 is a linear function of S_O, accounting for the high correlation between the change in the mean salinity of the ocean and the conservation error, which is apparent from Figure D.6.

D.3.3 The role of flux adjustments

In Sections D.3.1 and D.3.2, a number of significant conservation errors are identified. The use of flux adjustments in the coupled model, however, will tend to mask their existence. In the case of the surface heat flux, for example, a mean interpolation error of -0.39 Wm^{-2} was diagnosed (Table D.6). However, it is the surface fluxes *after* interpolation onto the OGCM grid which are used to diagnose flux adjustments. The flux adjustments will therefore compensate for any interpolation errors. If the interpolation errors remain constant throughout a simulation, they will not represent a source of drift within the coupled model.

However, it was shown in Section D.3.1 that the interpolation error is determined by the sum of the fluxes at just 29 gridpoints. Any change in the mean fluxes at these gridpoints will lead to a change in the mean interpolation error. As can be seen from the temporal variability in Figure D.2b, and from the differences between the monthly-mean errors in Table D.6, the interpolation error is very sensitive to any change in the model climate.

Therefore, any shift in the climate of the coupled model away from its initial state, be it either as a result of drift within a control simulation, or as a result of an imposed change in external forcing, would be expected to lead to a change in the mean interpolation error. In turn, this will represent a source of drift within the model. In the case of the surface heat flux, natural feedback mechanisms will tend to counteract any drift; however, in the case of the surface freshwater flux, any drift in the mean salinity of the ocean will be unconstrained.

It is therefore hypothesised that the interpolation error is the dominant source of the salinity drift encountered by Bi (2002) in his transient climate change simulations.

D.4 Modifications to the coupling

Mk3L was designed to enable climate variability and change to be studied on millennial timescales. This requires a control climate which is both realistic and stable. The conservation errors documented in Sections D.3.1 and D.3.2 have been identified as a probable source of drift within the coupled model, and it was therefore necessary to rectify them.

The following sections document the modifications that were made to the coupling between the AGCM and OGCM, in order to ensure the rigorous conservation of both heat and freshwater. It should be noted that the conservation of momentum was not addressed, as the offset between the tracer and velocity grids within the OGCM renders any solution non-trivial; furthermore, conservation of momentum is less important than conservation of heat and freshwater, as a failure to conserve momentum does not have the potential to lead to ongoing drift within the model.

Type	Source model	Surface	Quantity conserved	Correction applied	Fields
1	AGCM	Ocean	Integral	$+(\Phi_A - \Phi_O)/A$	Heat flux, F_{p-e}, F_{ice}, F_{sub}
2	AGCM	Land	Integral	$\times(\Phi_A/\Phi_O)$	F_{run}
3	OGCM	Ocean	Mean	$+(T_O - T_A)$	SST

Table D.9: The three types of conservation operation supported by the subroutine `conserve`, and the fields conserved by each operation.

D.4.1 Conservation

The interpolation within the AGCM subroutine `ocforce` was modified to ensure that the values for the run-off are interpolated onto the OGCM grid, as for the other fields. To estimate values for the run-off for those 18 gridpoints which are treated as ocean by the AGCM, but as land by the OGCM, the same interpolation operations are performed as for the sea surface temperature, i.e. the operations listed in Table D.5.

The AGCM subroutine `ocforce` and OGCM subroutine `step0` were then modified so that, after each of the fields has been interpolated onto the target grid, it is passed to a new subroutine, `conserve`. The values on the source grids, and the estimated values on the target grids, are passed as arguments. `conserve` then performs the following operations:

- the areal integrals are calculated on both the surface and target grids (except in the case of sea surface temperature, where the areal means are calculated)
- the interpolation error is calculated
- a correction is applied to the values on the target grid

In addition to the values on the source and target grids, a third argument is passed to `conserve`. This is an integer between 1 and 3, and indicates the type of conservation operation to be performed. A single argument is returned, which is an array containing the corrected values on the target grid. After the calls to `conserve`, execution of the model continues as it did previously.

Table D.9 summarises the three types of conservation operation which are supported, with further details being given below. Table D.10 summarises the corrections applied by `conserve` during the one year of the coupled model control simulation that was analysed in Section D.3.1.

The source code for `conserve` is provided in Section G.2.

Type 1

Type 1 is used to conserve the global integral of fluxes from the atmosphere to the ocean. It is therefore used to conserve the total surface heat flux, and the totals of the components F_{p-e}, F_{ice} and F_{sub} of the surface freshwater flux.

`conserve` calculates the interpolation error, $d\Phi = \Phi_O - \Phi_A$, where Φ_A and Φ_O are as defined in Equations D.7 and D.8 respectively. $d\Phi$ is then divided by the surface area of the ocean A, and subtracted from the values on the OGCM grid.

Field	Units	Corrections applied		
		Minimum	Maximum	Mean
Heat flux	Wm^{-2}	-2.10	+3.38	+0.39
F_{p-e}	mm/day	-0.123	+0.115	-0.015
F_{ice}	mm/day	-0.118	+0.580	-0.008
F_{sub}	mm/day	-0.003	+0.005	+0.000
F_{run}	-	1.004	1.085	1.034
SST	K	-0.006	+0.000	-0.003

Table D.10: A summary of the corrections applied by the subroutine `conserve` during one year of a coupled model control simulation.

Type 2

Type 2 is used to conserve run-off. It is similar to Type 1, except that the total flux is conserved over land, rather than over the ocean. In this case, the total fluxes on the AGCM and OGCM grids, Φ_A and Φ_O respectively, can be defined by the integrals

$$\Phi_A = \iint_{AGCM\,land} F_A \, dxdy \tag{D.12}$$

$$\Phi_O = \iint_{OGCM\,land} F_O \, dxdy \tag{D.13}$$

Given that the run-off cannot be negative, it is not appropriate to conserve the global integral through the application of a uniform offset, which might generate negative values. Instead, run-off is conserved by multiplying the values on the AGCM grid by a uniform ratio, given by Φ_A/Φ_O.

Type 3

Type 3 is used to conserve sea surface temperature (SST). As the sea surface temperature is not a flux field, the global mean is conserved in this case. If T_O is the mean SST on the OGCM grid, and T_A is the mean SST on the AGCM grid, then the interpolation error can be defined as $dT = T_A - T_O$. The global-mean SST is conserved by subtracting dT from the values on the AGCM grid.

D.4.2 Surface salinity tendency

In order to avoid the problems associated with the conversion of the components of the surface freshwater flux to equivalent surface salinity tendencies, the coupled model was modified in two ways:

- the actual sea surface salinity at each gridpoint is used to calculate an equivalent surface salinity tendency, ensuring local conservation of freshwater
- a globally-uniform correction is applied to the surface salinity tendencies, ensuring global conservation of freshwater

Sea surface salinities

The surface salinity tendencies are calculated within the AGCM subroutine `ocforce`. The use of the actual sea surface salinity (SSS) to calculate the surface salinity tendencies requires that an additional field be passed from the OGCM to the AGCM within the coupled model.

This required relatively minor modifications to the model, as the SSS is already available to the AGCM. The values are copied into the array `osal` within the OGCM subroutined `tracer1`, and this array is available to the AGCM via the `common` block `bcogcm`. The calculation of the surface salinity tendencies takes place on the OGCM grid, and hence no interpolation is necessary.

For consistency with the other fields exchanged within the coupled model, it should be possible to apply "flux" adjustments to the SSSs. The coupled model subroutine `tmread` already contained the code to read these values from the file `s1coravth`, but they were not used. The coupled model subroutine `ocinit` was therefore modified, so as to apply these adjustments to the SSSs.

To enable flux adjustments to be diagnosed for the coupled model, the stand-alone AGCM must also be able to calculate the surface salinity tendencies. The AGCM subroutine `datard` was therefore modified so as to read monthly-mean climatological SSSs from the file `clim3f.sss`. The values are read into the array `osal`, the same array used to pass SSSs from the OGCM to the AGCM. `ocforce` was modified to use linear interpolation in time to estimate values at each timestep.

Conservation of freshwater

The AGCM subroutine `ocforce` now converts each of the components of the surface freshwater flux into an equivalent surface salinity tendency, using the actual SSS at each gridpoint. The total salinity tendency at each gridpoint can then be calculated by adding the tendencies derived from each component of the freshwater flux.

In order to ensure global conservation of freshwater, the components of the surface freshwater flux, and the total surface salinity tendency are then passed to a new subroutine, `conserve_fw`. This performs the following operations:

- for each component of the surface freshwater flux, the global integral Φ_O is calculated, as defined by Equation D.8
- Φ_O is converted into the equivalent rate of change in the mean salinity of the ocean, using Equation D.9
- these values are summed, obtaining the total rate of change in the mean salinity of the ocean
- the surface salinity tendencies calculated by `ocforce` are integrated, and the integrals converted into the equivalent rate of change in the mean salinity of the ocean using Equation D.10
- the conservation error is calculated using Equation D.11
- the conservation error is converted into an equivalent rate of change in the mean salinity of the upper layer of the ocean model, by multiplying it by $H/\Delta z$, where H is the mean depth of the ocean and Δz is the thickness of the upper layer of the ocean model

- this figure is then subtracted from the surface salinity tendencies, ensuring that they give the correct rate of change in the mean salinity of the ocean

The value of S_{ocean} used in Equation D.9 is 34.7 psu. For exact conservation, the actual mean salinity of the ocean at each timestep should be used. However, when the AGCM is being spun up in stand-alone mode, no ocean salinity is available. A value of 34.7 psu is used in this case, being the mean salinity of the ocean according to the World Ocean Atlas 1998 (National Oceanographic Data Center, 2002). In the coupled model, the actual mean salinity of the ocean could be used instead. However, it would be time-consuming to calculate this, and it is unlikely to differ significantly from 34.7 psu. For these reasons, and for consistency with the stand-alone AGCM, a constant value of 34.7 psu is used.

It should be noted, therefore, that while the freshwater conservation error within Mk3L should be smaller than the value of $O(10^{-6})$ psu/year diagnosed in Section D.3.2 for the original model, an error still remains.

The source code for `conserve_fw` is provided in Section G.3.

Appendix E

Restart and auxiliary files

E.1 Introduction

Mk3L must be supplied with restart files, which specify the initial conditions, and with auxiliary files, which specify the boundary conditions; these files are described in detail in Chapter 5. This appendix documents the procedures which were followed to generate the default restart and auxiliary files which are supplied with the model source code. The generation of the auxiliary files is described in Sections E.2, E.3, E.4 and E.5, while the generation of the restart files is described in Section E.6.

To enable intercomparison with other climate system models, the default spin-up procedure for Mk3L is consistent with PMIP2 experimental design (Paleoclimate Modelling Intercomparison Project, 2005). For coupled model experiments, it is specified that a control simulation be conducted for pre-industrial conditions, which are taken as being those which existed around the year AD 1750. The experimental design is summarised in Table E.1.

E.2 Generation of auxiliary files

This section summarises the process whereby the auxiliary files were generated: the sources of the data, the software packages which were used, and the principles which were followed. The precise steps taken to generate each auxiliary file are documented in the sections which follow.

E.2.1 Datasets

World Ocean Atlas 1998

For consistency, the World Ocean Atlas 1998 (National Oceanographic Data Center, 2002, commonly referred to as the “Levitus 1998” dataset) was used not only to initialise the ocean model, but also to provide the sea surface temperatures and salinities used to spin up both the ocean and atmosphere models.

The World Ocean Atlas 1998 dataset contains analysed ocean temperatures and salinities, on a $1^\circ \times 1^\circ$ latitude-longitude grid in the horizontal direction, and on 33 levels in the vertical direction, at depths ranging

Boundary condition	Value
Vegetation	Fixed
Ice sheets	Modern
Topography/coastlines	Modern
CO_2 concentration [ppm]	280
CH_4 concentration [ppb]	760
N_2O concentration [ppb]	270
Chlorofluorocarbons	None
O_3 concentration	Modern
Solar constant [Wm^{-2}]	1365
Epoch [years BP]	0
Eccentricity of Earth's orbit	0.016724
Obliquity of Earth's axis [°]	23.446
Longitude of perihelion [°]	102.04
Initial ocean state	World Ocean Atlas 1998

Table E.1: PMIP2 experimental design for coupled model control simulations.

from 0 to 5500 m. Annual means are available for all vertical levels, while monthly means are available for the uppermost 24 levels (which range in depth from 0 to 1500 m). The World Ocean Atlas 1998 dataset was obtained in netCDF (Unidata Program Center, 2005) from the NOAA-CIRES Climate Diagnostics Center (NOAA-CIRES, 2005).

NCEP-DOE Reanalysis 2

The wind stresses used to spin up the ocean model were obtained from the NCEP-DOE Reanalysis 2 (Kanamitsu et al., 2002). Monthly-mean wind stresses for the period 1979-2003 were obtained in netCDF from the NOAA-CIRES Climate Diagnostics Center.

E.2.2 The definition of "surface"

The upper layer of the Mk3L ocean model has a thickness of 25 m. The temperature and salinity of this layer therefore simulate the average temperature and salinity of the upper 25 m of the water column, and not the sea surface temperature (SST) and sea surface salinity (SSS) *per se*.

It is therefore inappropriate to use the observed SST and SSS to spin up the ocean model. Relaxing the temperature and salinity of a 25 m-thick layer of water towards these values, rather than towards an equivalent observational quantity, will result in unrealistic surface fluxes. In turn, this will influence the magnitude of the flux adjustments diagnosed for the coupled model.

The surface boundary conditions on the ocean model are *not* therefore the observed SST and SSS, but are instead the averages of the World Ocean Atlas 1998 temperature and salinity over the upper 25 m of the water column. For consistency, the same surface boundary conditions are applied to the atmosphere model.

E.2.3 Conservation

Spatial conservation

After vertical averaging over the upper 25 m of the water column, the annual-, global-mean SST and SSS on the World Ocean Atlas 1998 grid are 18.032°C and 34.680 psu respectively.

After interpolating the SST and SSS onto either the Mk3L ocean or atmosphere model grids, a temporally- and spatially-uniform correction was applied, in order to ensure that the annual-, global-mean SST and SSS had been conserved. The corrections applied were small, being uniform offsets of +0.049°C and +0.047 psu respectively for the interpolation of the SST and SSS onto the ocean model grid, and +0.039°C for the interpolation of the SST onto the atmosphere model grid.

Temporal conservation: atmosphere model

The Mk3L stand-alone atmosphere model requires that four surface boundary conditions be provided via auxiliary files: SST, SSS, and the zonal and meridional components of the surface velocity. In each case, it is required that the values supplied be climatological values for the *start* of each month. The model then uses linear interpolation in time to estimate the values at each timestep.

However, this method can fail to conserve the annual mean. For example, let the number of days in calendar month i be n_i, and let the climatological mean SST (or other surface field) for this month be T_i. The observed annual-mean SST, $\overline{T}_{obs}$, is therefore given by

$$\overline{T}_{obs} = \frac{\sum_{i=1}^{12} n_i T_i}{\sum_{i=1}^{12} n_i} \tag{E.1}$$

Observational datasets such as the World Ocean Atlas 1998 contain climatological *mean* values for each month. Let $\mathcal{T}_i$, the best estimate of the climatological SST for the start of month i, be given by the average of the mean SST for months $i-1$ and i, as follows:

$$\mathcal{T}_i = \frac{1}{2}\left(T_{i-1} + T_i\right) \tag{E.2}$$

During month i, the SST estimated by the model at each timestep will vary linearly from $\mathcal{T}_i$ to $\mathcal{T}_{i+1}$. The mean SST applied during this month will therefore be $\frac{1}{2}(\mathcal{T}_i + \mathcal{T}_{i+1})$, and the annual-mean SST applied by the model will be

$$\overline{T}_{mod} = \frac{\sum_{i=1}^{12} \frac{1}{2} n_i (\mathcal{T}_i + \mathcal{T}_{i+1})}{\sum_{i=1}^{12} n_i} \tag{E.3}$$

Substituting Equation E.2 into Equation E.3:

$$\overline{T}_{mod} = \frac{\sum_{i=1}^{12} n_i \left(\frac{1}{4} T_{i-1} + \frac{1}{2} T_i + \frac{1}{4} T_{i+1} \right)}{\sum_{i=1}^{12} n_i} \tag{E.4}$$

If the lengths of the calendar months are equal, then Equations E.1 and E.4 both simplify to

$$\overline{T}_{obs} = \overline{T}_{mod} = \frac{1}{12} \sum_{i=1}^{12} T_i \tag{E.5}$$

However, Mk3L uses a 365-day calendar, in which the lengths of the months vary from 28 to 31 days. In this case, $\overline{T}_{mod}$ and $\overline{T}_{obs}$ differ, and the interpolation in time fails to conserve the annual mean.

When auxiliary files are generated for use by the Mk3L atmosphere model, the climatological value for the start of month i is estimated by calculating the average of the mean values for months $i-1$ and i. The conservation error $\Delta\overline{T} = \overline{T}_{mod} - \overline{T}_{clim}$ is then calculated at each gridpoint. This error is subtracted from the observed values, ensuring that the annual-mean values applied by the model are equal to the observed annual mean. When generating the default auxiliary files supplied with Mk3L, however, these corrections were found to be small, not exceeding 0.013°C in magnitude in the case of the SST, or 0.011 psu in the case of the SSS.

Temporal conservation: ocean model

The Mk3L stand-alone ocean model requires that four surface boundary conditions be provided via auxiliary files: SST, SSS, and the zonal and meridional components of the surface wind stress. In each case, it is required that the values supplied be climatological values for the *midpoint* of each month (in contrast to the atmosphere model, where they are required to be climatological values for the *start* of each month). The model then uses linear interpolation in time to estimate the values at each timestep. As with the atmosphere model, the differing lengths of the calendar months can result in a failure to conserve the annual mean.

When auxiliary files are generated for use by the Mk3L ocean model, the *mean* value for each month is taken as being the best estimate of the climatological value for the *midpoint* of that month. The conservation error $\Delta\overline{T} = \overline{T}_{mod} - \overline{T}_{clim}$ is then calculated at each gridpoint. This error is subtracted from the observed values, ensuring that the annual-mean values applied by the model are equal to the observed annual mean. When generating the default auxiliary files supplied with Mk3L, however, these corrections were found to be small, not exceeding 0.007°C in magnitude in the case of the SST, 0.007 psu in the case of the SSS, or 1.2×10^{-4} Nm^{-2} in the case of the components of the surface wind stress.

E.2.4 Software

Two software packages were used to generate the auxiliary files which are supplied with Mk3L: Ferret (Pacific Marine Environmental Laboratory, 2005) and IDL (Research Systems Inc., 2005).

Ferret was used to perform the interpolation operations on the data. The @AVE transformation, which calculates length-, area- and volume-weighted averages (in one, two and three dimensions respectively), was used to interpolate the observational datasets onto the Mk3L atmosphere and ocean model grids.

The model grids are coarser than the grids on which the observational datasets were supplied, and hence spatial averaging is the appropriate interpolation operation.

Ferret was also used to fill any missing values in the observational datasets. In the case of both the World Ocean Atlas 1998 and the NCEP-DOE Reanalysis 2 datasets, this was necessary because of differences between the positions of the coastlines on the Mk3L grid and on the grids on which these datasets were supplied. By using spatial interpolation to fill missing values over land, it could be ensured that the interpolation onto the model grid would generate values for all ocean gridpoints.

The @FAV transformation, which replaces missing values with the average of the values for the adjacent gridpoints, was used to fill the observational datasets over land. If no values are available for any of the adjacent gridpoints, no value can be calculated; it was therefore necessary to apply the @FAV transformation repeatedly in order to fill all the missing values.

IDL was used to perform all the other operations on the data, including the generation of auxiliary files in the formats read by Mk3L.

E.3 Ocean model spin-up

E.3.1 Sea surface temperature

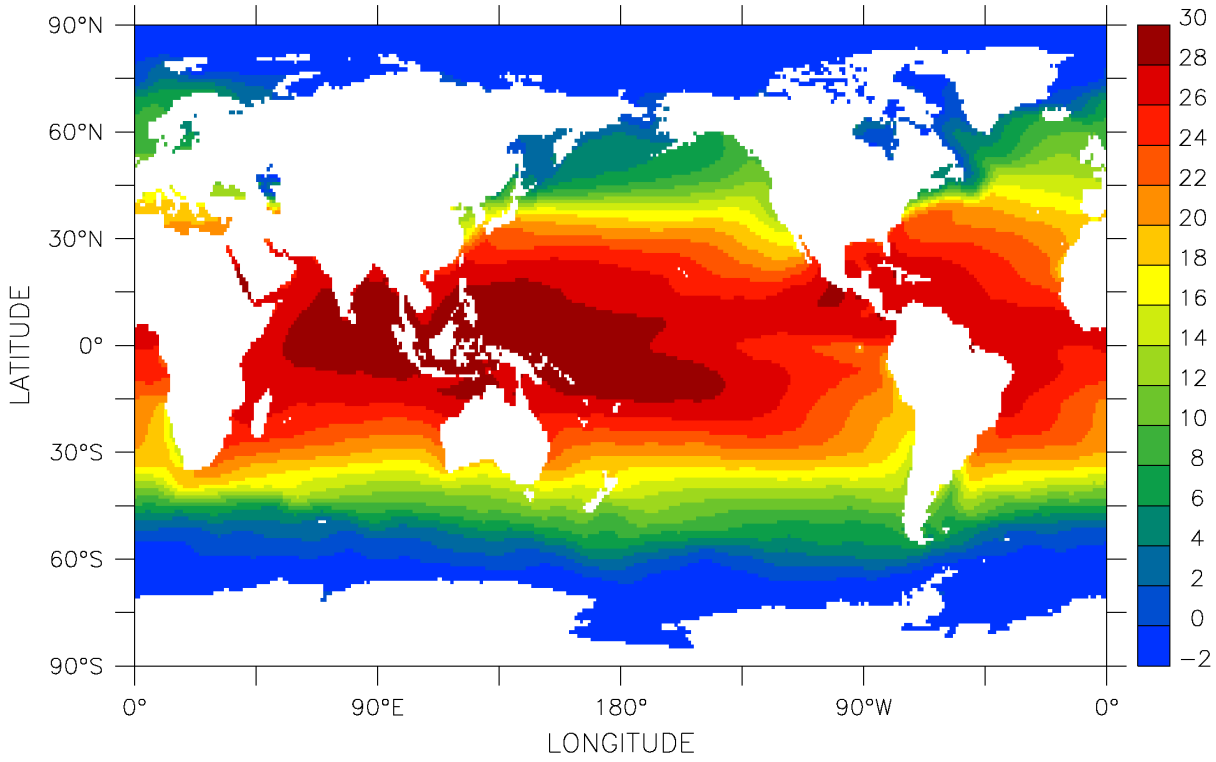
The annual-mean World Ocean Atlas 1998 sea surface temperature, after taking the average over the upper 25 m of the water column, is shown in Figure E.1. Values are shown on the original $1^\circ \times 1^\circ$ grid, and after interpolation onto the Mk3L ocean model grid.

The steps taken to generate the auxiliary files required by the ocean model were as follows:

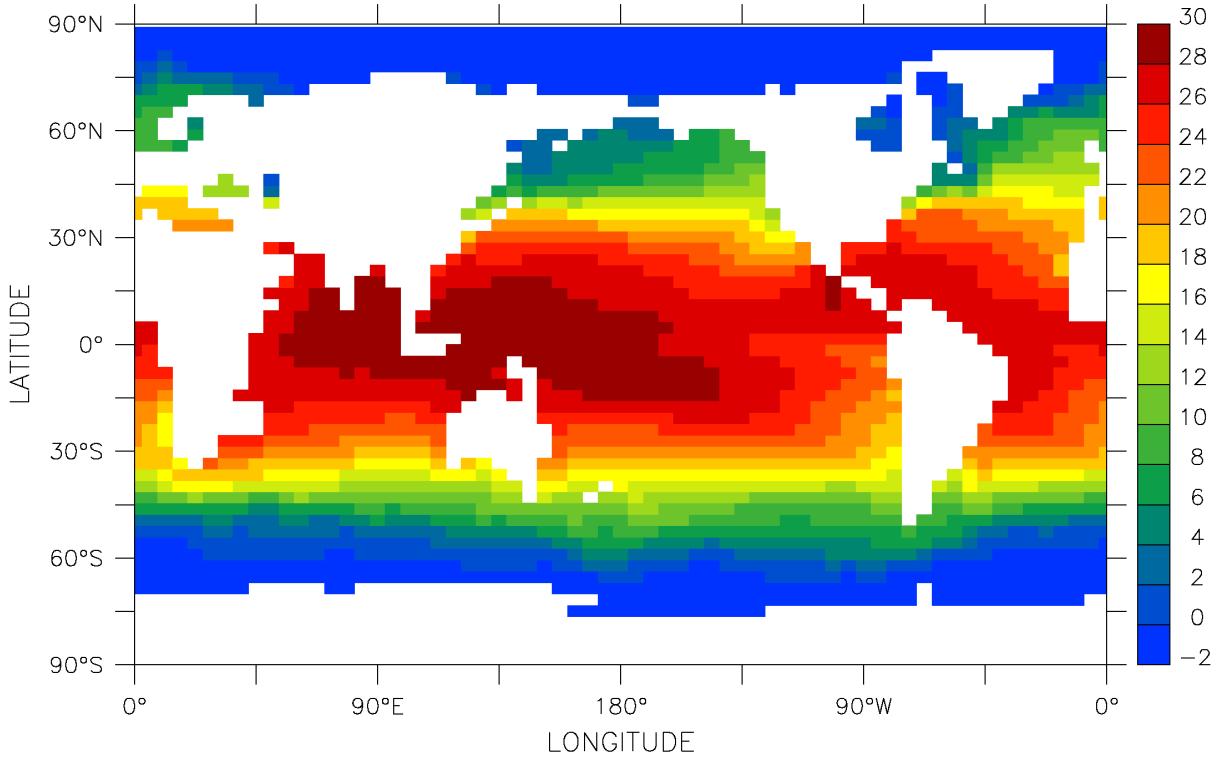
1. The monthly-mean World Ocean Atlas 1998 ocean temperatures were averaged over the upper 25 m of the water column.
2. The missing values over land were filled.
3. The values were interpolated onto the Mk3L ocean model grid.
4. A temporally- and spatially-uniform correction of $+0.049^\circ$C was applied to the values on the model grid, in order to conserve the annual-, global-mean sea surface temperature (SST) on the World Ocean Atlas 1998 grid of 18.032°C.
5. An annual-mean correction was applied to the monthly values at each gridpoint, ensuring that the annual-mean SST is conserved upon the temporal interpolation conducted by the model.
6. The data was written to file in the format read by the Mk3L ocean model. Two auxiliary files were generated: one containing annual-mean SSTs, for the initial stage of spin-up runs, and the other containing monthly SSTs.

E.3.2 Sea surface salinity

The annual-mean World Ocean Atlas 1998 sea surface salinity (SSS), after taking the average over the upper 25 m of the water column, is shown in Figure E.2. Values are shown on the original $1^\circ \times 1^\circ$ grid, and



(a) World Ocean Atlas 1998



(b) Mk3L ocean model

Figure E.1: The annual-mean World Ocean Atlas 1998 sea surface temperature (°C): (a) on the original grid, and (b) after interpolation onto the Mk3L ocean model grid.

after interpolation onto the Mk3L ocean model grid.

The steps taken to generate auxiliary files for the Mk3L ocean model were equivalent to those taken to generate the auxiliary files containing the SST. However, between Steps 3 and 4, two modifications were made to the SSSs. The first of these modifications was required by the fact that the Mk3L ocean model grid does not resolve the Amazon Delta; the second was required by the highly spatially-variable nature of the World Ocean Atlas 1998 sea surface salinities over the Arctic Ocean. Each of the modifications is outlined below.

The correction applied at Step 4 was +0.047 psu, ensuring that the annual-, global-mean SSS on the World Ocean Atlas 1998 grid of 34.680 psu was conserved. The SSSs shown in Figure E.2b are the values which were used to spin up the model, after the various modifications had been made.

Sea surface salinity and the Amazon Delta

Initial spin-up runs for the Mk3L atmosphere and ocean models exhibited a very large mismatch between the surface salinity tendencies in the vicinity of the Amazon Delta (Figures E.3a and E.3b). As a consequence, very large flux adjustments were being diagnosed for the coupled model.

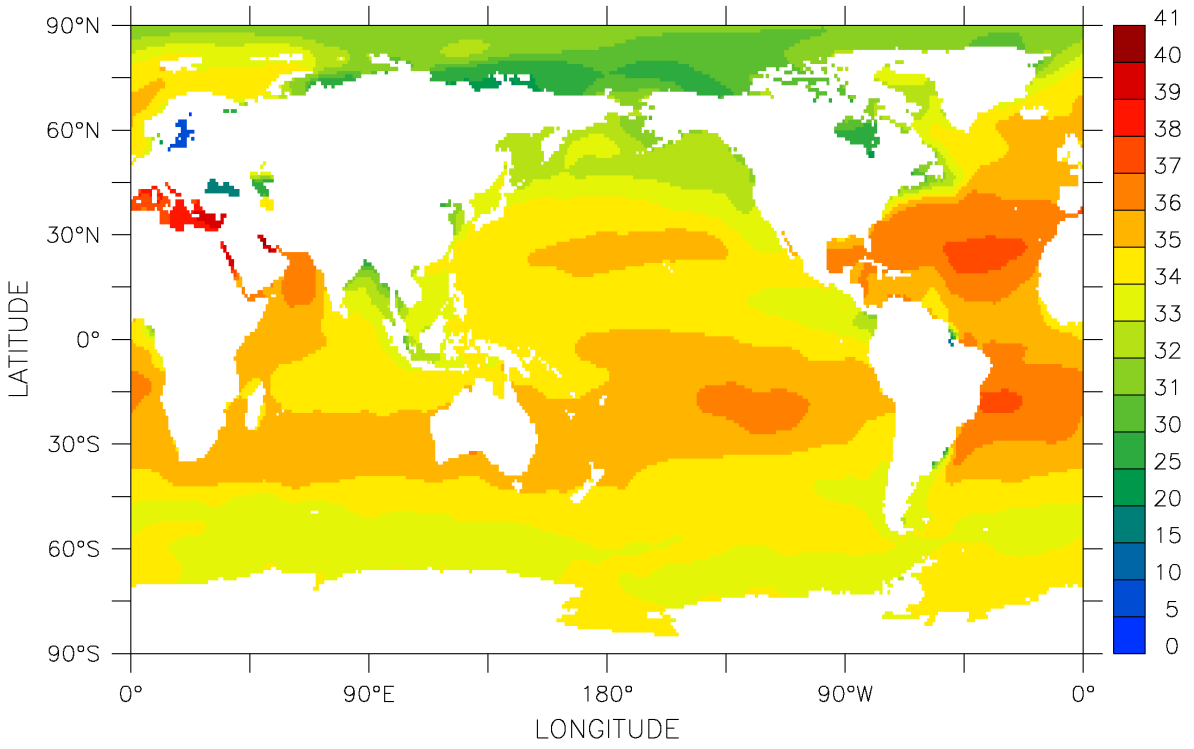
Comparison of the World Ocean Atlas 1998 and Mk3L ocean model grids revealed that the Amazon Delta is not resolved on the model grid (Figure E.4). The relatively fresh waters of the Amazon Delta are not therefore represented on the model grid, and the salinities in this region therefore experience negligible freshening relative to those of the surrounding open ocean (Figure E.4b). As a consequence, the surface salinity tendencies are small (Figure E.3b). In contrast, the atmosphere model simulates a large amount of run-off, and the surface salinity tendencies are large and negative (Figure E.3a).

Prior to attempting to resolve this mismatch, an algorithm was developed to search for any other significant estuaries which may be unresolved on the Mk3L ocean model grid. This algorithm operated as follows:

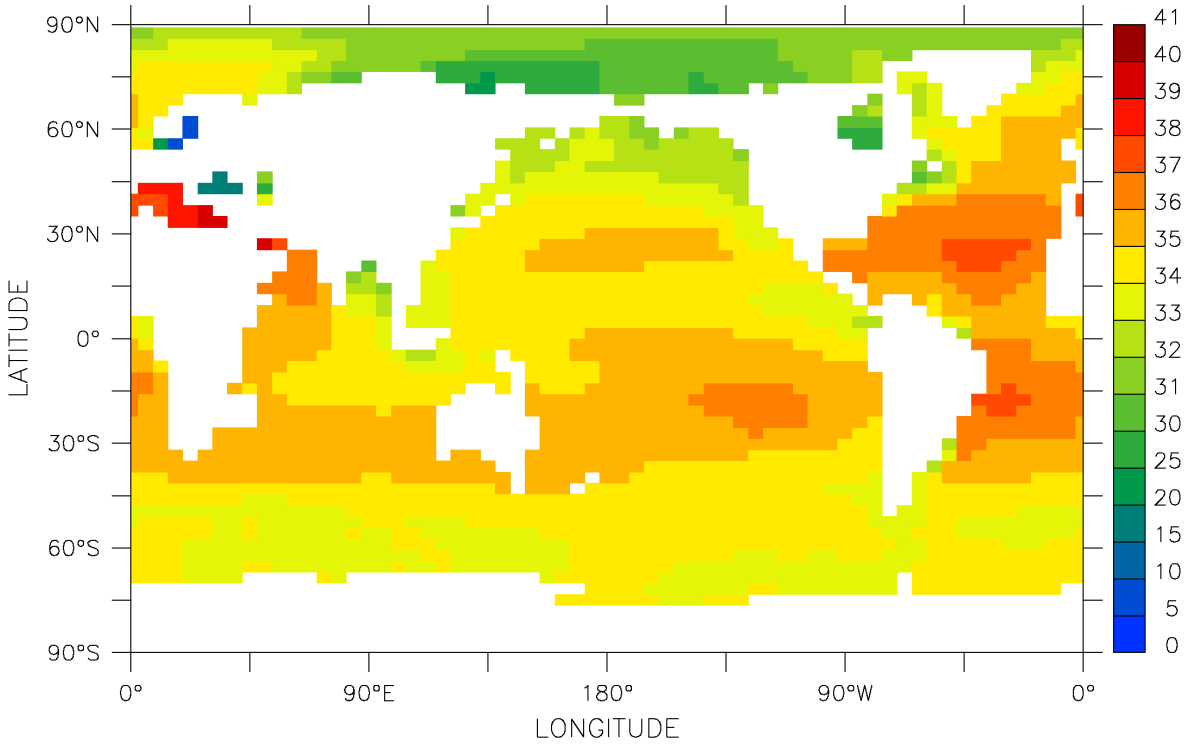
1. Each *land* gridpoint on the Mk3L ocean model grid was checked to see if it overlaps with *ocean* gridpoints on the World Ocean Atlas 1998 grid.
2. If so, then the following checks were performed:
   (a) The SSS for that gridpoint [the interpolation process estimates SSSs for all gridpoints, whether they are land or ocean] was compared with that of any surrounding ocean gridpoints.
   (b) If, for any month of the year, the gridpoint had an SSS less than 30 psu, and it was surrounded by ocean gridpoints which all had SSSs greater than 34 psu, then it was identified as representing a possible unresolved estuary.

The upper salinity threshold of 30 psu for an unresolved estuary was chosen so as only to identify gridpoints where the seawater has been significantly diluted by the outflow of freshwater from the mouth of a river. Likewise, the lower salinity threshold of 34 psu for the surrounding ocean gridpoints was chosen in order to ensure that the outflow had not already been resolved. It was assumed that, if the diluting effect of the freshwater outflow was already represented on the Mk3L ocean model grid, then at least one of surrounding ocean gridpoints would have a salinity lower than the typical open water value of 34 psu.

The four gridpoints which were identified by this algorithm are shown in Table E.2. Of these, gridpoint (9, 33) represents the Red Sea. In February, the World Ocean Atlas 1998 indicates the presence of a small region



(a) World Ocean Atlas 1998



(b) Mk3L ocean model

Figure E.2: The annual-mean World Ocean Atlas 1998 sea surface salinity (psu): (a) on the original grid, and (b) after interpolation onto the Mk3L ocean model grid (including the modifications which were made in the vicinity of the Amazon Delta, and over the Arctic Ocean).

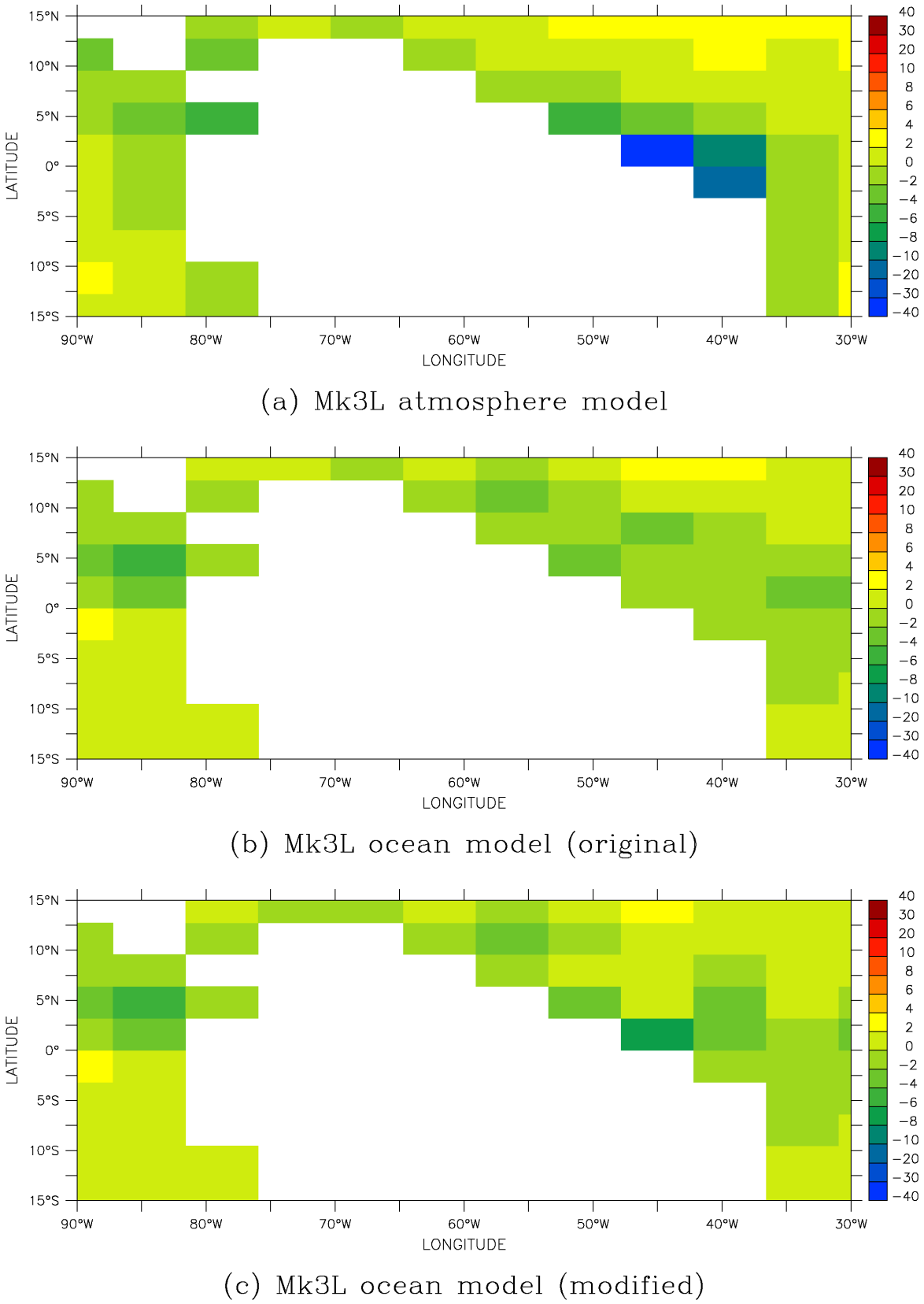
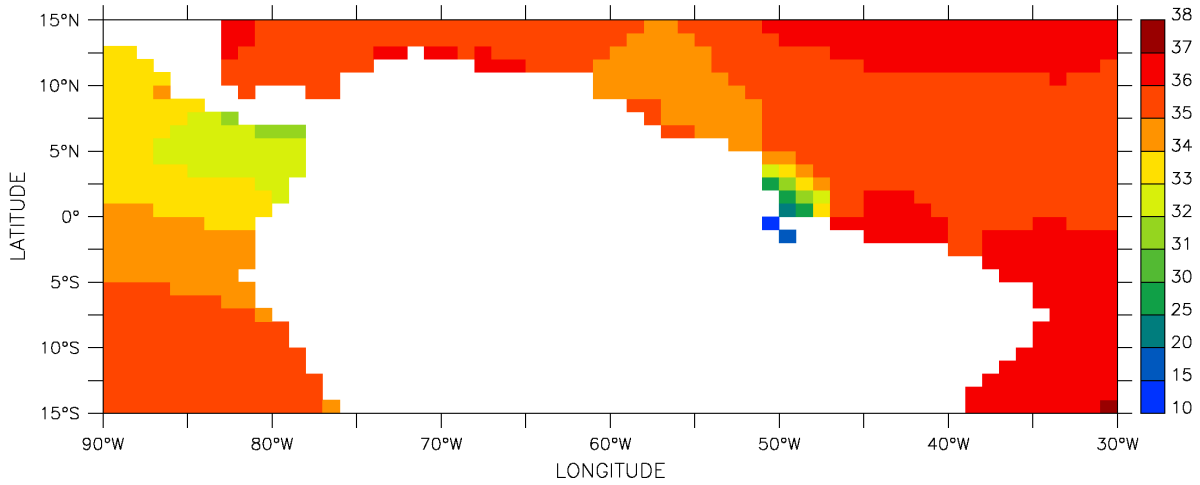
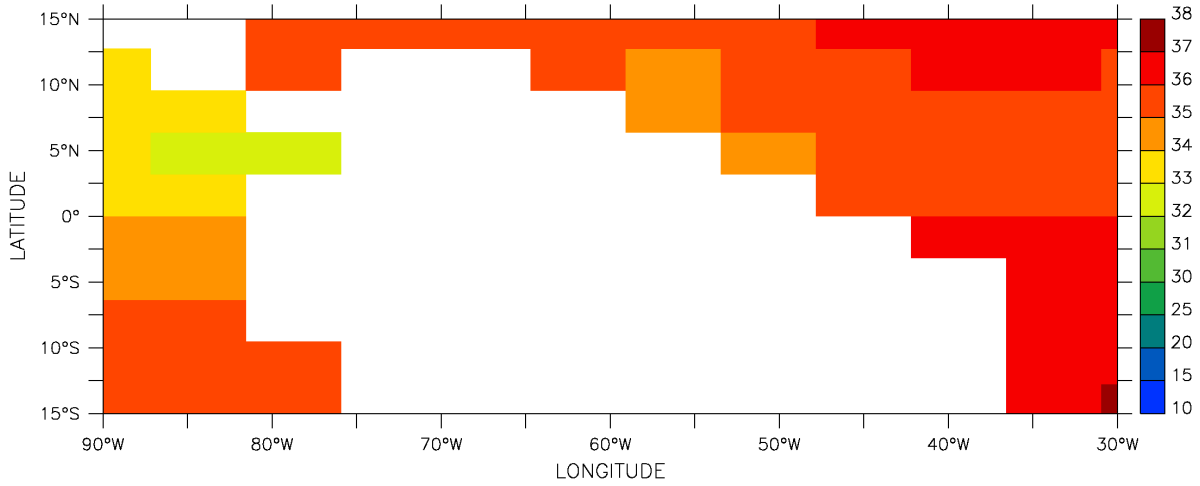


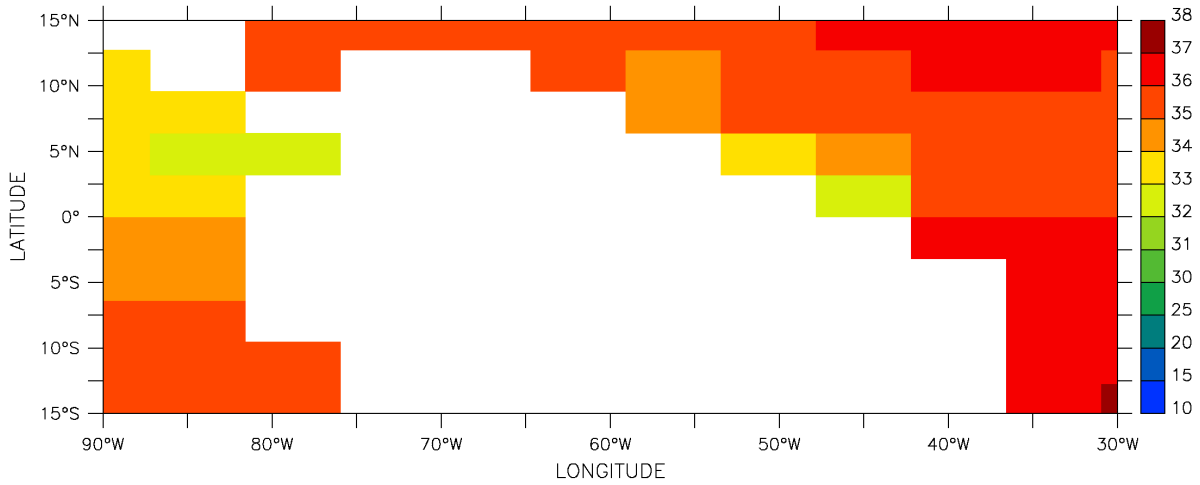
Figure E.3: Annual-mean surface salinity tendencies (psu/year) in the vicinity of the Amazon Delta: (a) diagnosed from an atmosphere model spin-up run (40-year average), (b) diagnosed from an ocean model spin-up run (using the original sea surface salinities, 100-year average), and (c) diagnosed from an ocean model spin-up run (using the modified sea surface salinities, 100-year average).



(a) World Ocean Atlas 1998



(b) Mk3L ocean model (original)



(c) Mk3L ocean model (modified)

Figure E.4: The annual-mean World Ocean Atlas 1998 sea surface salinity (psu) in the vicinity of the Amazon Delta: (a) on the original grid, (b) after interpolation onto the Mk3L ocean model grid, and (c) after relocation of the freshwater content of the Amazon Delta.

Gridpoint			Number
(i, j)	λ, ϕ	Water body	of months
(56, 28)	51°W, 2°S	Amazon Delta	11
(56, 29)	51°W, 2°N	Amazon Delta	7
(9, 33)	45°E, 14°N	Red Sea	1
(5, 50)	23°E, 68°N	Lake Inari	12

Table E.2: The Mk3L ocean model gridpoints which were identified as potentially containing unresolved estuaries: the co-ordinates on the model grid; the longitude and latitude; the water body to which they correspond; and the number of calendar months for which the sea surface salinity criteria were satisfied.

Gridpoint		Land/	Ocean	SSS (psu)	
(i, j)	λ, ϕ	Ocean	fraction	Original	Modified
(56, 28)	51°W, 2°S	Land	0.109	(17.283)	(17.283)
(56, 29)	51°W, 2°N	Land	0.450	(25.096)	(25.096)
(56, 30)	51°W, 5°N	Ocean	0.723	34.429	33.461
(57, 30)	45°W, 5°N	Ocean	1.000	35.788	34.770
(57, 29)	45°W, 2°N	Ocean	0.998	35.540	32.583

Table E.3: The Mk3L ocean model gridpoints which contain the freshwater content of the Amazon Delta, either before or after its relocation: the co-ordinates on the model grid; the longitude and latitude; whether it is treated as land or ocean by the model; the fraction of the gridpoint which is covered by ocean, according to the World Ocean Atlas 1998; and the annual-mean sea surface salinity before and after the relocation of the freshwater content of the Amazon Delta.

of relatively fresh water at the mouth of the Red Sea. This is not present during the other months of the year, during which this region has *higher* sea surface salinities than those of the surrounding ocean. The February salinities may therefore represent an anomaly within the dataset; irrespective of whether or not this is the case, this gridpoint does not represent an unresolved estuary.

The relatively fresh salinities at gridpoint (5, 50) arise from Lake Inari in northern Finland, which occupies a single gridpoint on the World Ocean Atlas 1998 grid, and which has no resolved connection with the world ocean.

The gridpoints (56, 28) and (56, 29) represent the Amazon Delta, which therefore appears to be the only significant estuary which is unresolved on the Mk3L ocean model grid. In order to rectify this, the freshwater content of these gridpoints (according to the World Ocean Atlas 1998) was relocated to the neighbouring ocean gridpoints on the Mk3L ocean model grid. The gridpoint (56, 28) only has one neighbouring ocean gridpoint, (57, 29), to which its freshwater content was tranferred in its entirety. The gridpoint (56, 29), however, has three neighbouring ocean gridpoints: (56, 30), (57, 30) and (57, 29). Its freshwater content was therefore shared equally between these three gridpoints.

Table E.3 shows the locations of each of these gridpoints; the fraction of each gridoint which is covered by ocean according to the World Ocean Atlas 1998; and the annual-mean SSS at each gridpoint, before and after the relocation of the freshwater content of gridpoints (56, 28) and (56, 29).

The relocation was performed in a manner which strictly conserves freshwater. Let two gridpoints have areas A_1, A_2, ocean fractions f_1, f_2, and sea surface salinities S_1, S_2. In order to relocate the freshwater

content of the first gridpoint to the second, the sea surface salinity for the second gridpoint was replaced with

$$S_2' = \frac{(f_2 A_2 - f_1 A_1) S_2 + f_1 A_1 S_1}{f_2 A_2} \tag{E.6}$$

Figure E.4c shows that, while this modification leads to an improvement in the ocean model surface salinity tendencies, a significant mismatch with the atmosphere model remains.

Sea surface salinity and the Arctic Ocean

Initial spin-up runs for the Mk3L ocean model experienced surface salinity tendencies over the Arctic Ocean which were large in magnitude and highly spatially variable (Figure E.5c). The magnitude of the annual-mean salinity tendency over the Arctic Ocean was as large as 15.1 psu/year, which is equivalent to a net freshwater flux of $\sim$10 m/year; over the remainder of the world ocean, the magnitude of the annual-mean salinity tendency did not exceed 13.7 psu/year, despite the larger fluxes of precipitation, evaporation and run-off that are encountered at low latitudes.

Such large surface salinity tendencies were hypothesised to arise from the highly spatially-variable nature of the World Ocean Atlas 1998 SSSs over the Arctic Ocean (Figure E.5a). This give rises to horizontal salinity gradients which are larger than those which can be sustained by a coarse-resolution ocean model such as Mk3L, without the application of large, and highly spatially-variable, surface freshwater fluxes.

To test this hypothesis, the SSSs were "smoothed" over the Arctic Ocean. For the northernmost seven latitude rows on the Mk3L ocean model grid, representing the region north of 66.9°N, the SSS at each gridpoint was replaced with the area-weighted average of the SSSs at all gridpoints lying within a radius of 400 km. The southern limit of the averaging was placed at 66.9°N, as this represents the position of Bering Strait on the model grid; the radius of 400 km was chosen, as this just exceeds the meridional spacing of the gridpoints on the Mk3L ocean model grid, which is $\sim$350 km. This ensured that, when a modified value was calculated for each gridpoint, the values for the gridpoints immediately to the north and south would always contribute towards the average.

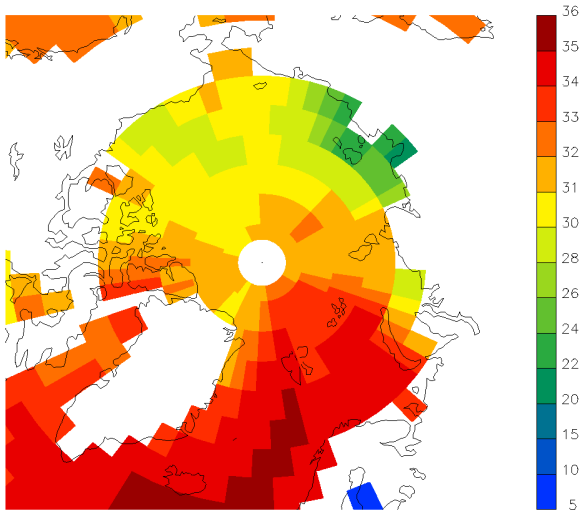
The modified SSSs are shown in Figure E.5b. The changes in the annual means were as large in magnitude as 2.64 psu, while those in the monthly means were as large as 3.95 psu. When the modified SSSs were used to spin up the ocean model, the annual-mean salinity tendency over the Arctic Ocean (Figure E.5d) was found to exhibit less spatial variability, and did not exceed 10.6 psu/year in magnitude.

Thus the hypothesis was confirmed, and the modified sea surface salinities were used thereafter to spin up the ocean model.

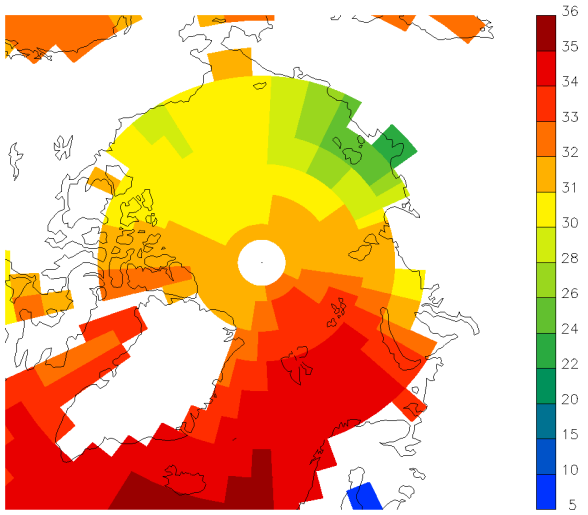
E.3.3 Surface wind stress

The annual-mean NCEP-DOE Reanalysis 2 surface wind stresses are shown in Figure E.6. Values are shown on the original 1.875°$\times$1.875° latitude-longitude grid, and after interpolation onto the Mk3L ocean model grid.

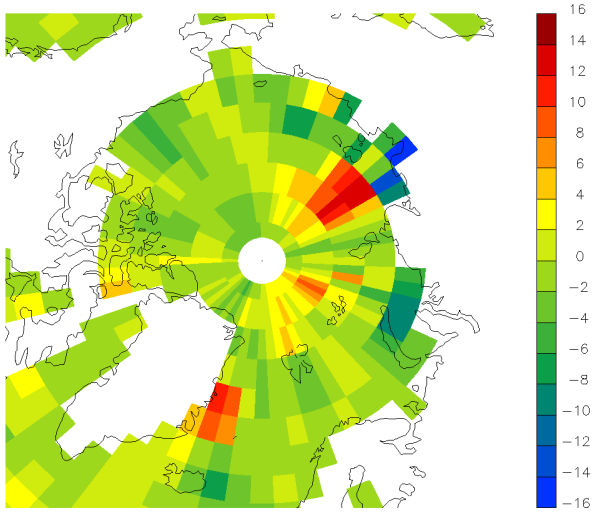
The steps taken to generate the auxiliary files required by the ocean model were as follows:



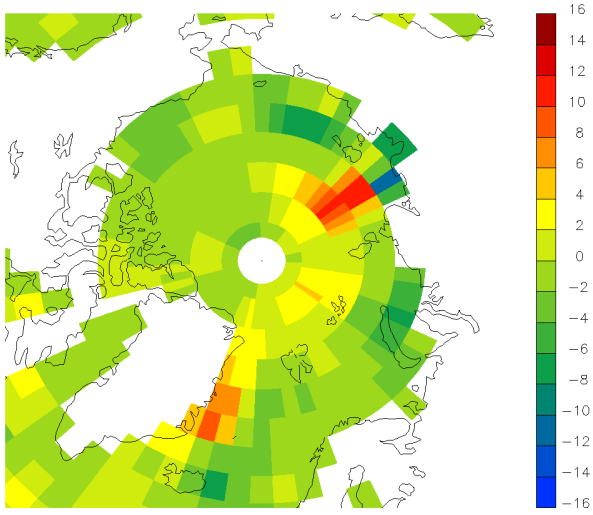
(a) Sea surface salinity (original)



(b) Sea surface salinity (modified)



(c) Surface salinity tendency (original)



(d) Surface salinity tendency (modified)

Figure E.5: The annual-mean World Ocean Atlas 1998 sea surface salinity (psu): (a) after interpolation onto the Mk3L ocean model grid, and (b) after the values over the Arctic Ocean have been smoothed. The annual-mean surface salinity tendencies diagnosed from ocean model spin-up runs (psu/year, 100-year averages): (c) using the original sea surface salinities, and (d) using the sea surface salinities after the values over the Arctic Ocean have been smoothed.

1. Climatological monthly-mean NCEP-DOE Reanalysis 2 surface wind stresses were calculated for the period 1979–2003.

2. The values were multiplied by -10, to convert them from momentum fluxes in Nm^{-2} to surface stresses in dynes/cm^2, as required by the ocean model.

3. The values over land were masked out.

4. Interpolation was used to generate “pseudo-data” over land.

5. The values were interpolated onto the Mk3L ocean model grid.

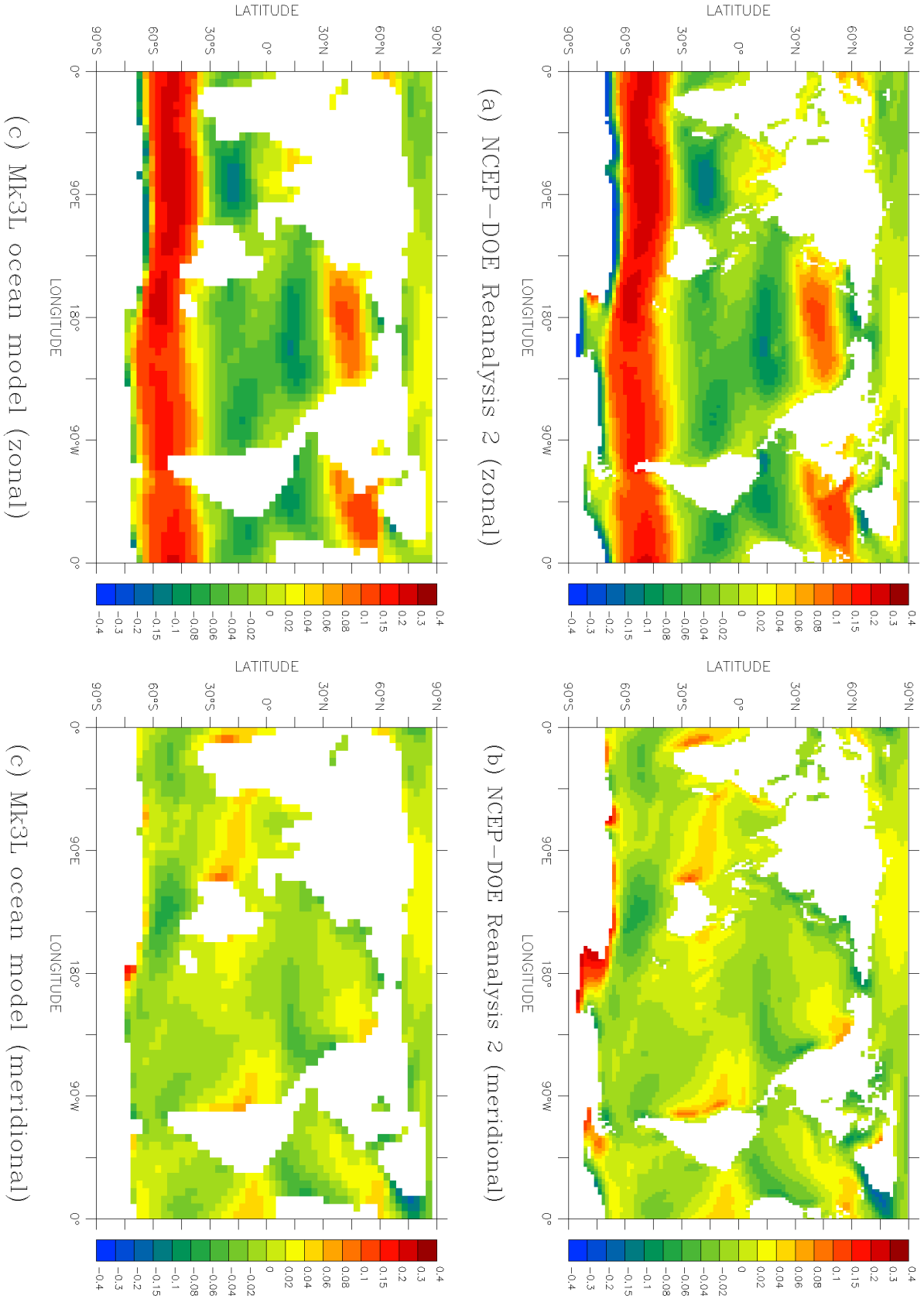


Figure E.6: The annual-mean NCEP-DOE Reanalysis 2 surface wind stresses (Nm^{-2}, 1979–2003 average): (a), (b) the zonal and meridional components, respectively, on the original grid; (c), (d) the zonal and meridional components, respectively, after interpolation onto the Mk3L ocean model grid.

6. An annual-mean correction was applied to the values at each gridpoint, ensuring that the annual-mean surface wind stress is conserved upon the temporal interpolation conducted by the model.

7. The data was written to file in the format read by the Mk3L ocean model. Two auxiliary files were generated: one containing annual-mean wind stresses, for the initial stage of spin-up runs, and the other containing monthly wind stresses.

These steps are similar to those which were taken to derive the auxiliary files containing the SST and SSS, but with some important differences:

- The surface wind stresses over land were masked out, and then replaced with "psuedo-data" generated by interpolation from the values for the surrounding ocean gridpoints. This is necessary because of the differences in the positions of the coastlines on the NCEP-DOE Reanalysis 2 and Mk3L ocean model grids. If no masking was carried out, then NCEP-DOE Reanalysis 2 wind stresses over land would contribute towards the values used to force the ocean model at some gridpoints. However, the surface wind stresses can be much larger in magnitude over land than over the ocean, as a result both of topography and of the greater surface roughness lengths, and this would therefore be inappropriate.

- The observed values were interpolated onto the horizontal velocity grid, as required by the model; in contrast, the SSTs and SSSs were interpolated onto the tracer grid, as also required by the model.

- Unlike the SSTs and SSSs, no correction was applied to the interpolated values in order to conserve the global-mean wind stress. The difference in extent between the domain on which the data is supplied, and the domain on which the values are applied within the model (Figure E.6), was considered to render this inappropriate. This difference in treatment is also consistent with the treatment of the surface fluxes within the coupled model (Section D.4), where the coupling ensures rigorous conservation of heat and freshwater, but not of momentum.

E.4 Atmosphere model spin-up

E.4.1 Sea surface temperature

The steps taken to generate the auxiliary file required by the Mk3L atmosphere model were as follows:

1. The monthly-mean World Ocean Atlas 1998 ocean temperatures were averaged over the upper 25 m of the water column.

2. The missing values over land were filled.

3. The values were interpolated onto the Mk3L atmosphere model grid.

4. A temporally- and spatially-uniform correction of +0.039°C was applied to the values on the model grid, in order to conserve the annual-, global-mean sea surface temperature on the World Ocean Atlas 1998 grid of 18.032°C.

5. Climatological sea surface temperatures for the *start* of each month were estimated, by averaging the climatological means for consecutive months.

6. An annual-mean correction was applied to the monthly values at each gridpoint, ensuring that the annual-mean sea surface temperature is conserved upon the temporal interpolation conducted by the model. These corrections did not exceed 0.013°C in magnitude.

7. The data was written to file in the format read by the Mk3L atmosphere model.

These are similar to the steps which were taken to generate the auxiliary file required by the ocean model, with just two differences:

- The World Ocean Atlas 1998 sea surface temperatures are interpolated onto the atmosphere model grid, rather than the ocean model grid.
- Climatological sea surface temperatures are generated for the *start* of each month, rather than the *midpoint* of each month.

E.4.2 Sea surface salinity

The sea surface salinity is required by the Mk3L atmosphere model to convert the surface freshwater fluxes to equivalent surface salinity tendencies. As these calculations are performed upon the ocean model grid, the steps taken to generate the auxiliary file required by the atmosphere model were very similar to those taken to generate the auxiliary file required by the ocean model.

The only difference between the two auxiliary files is that the atmosphere model file contains climatological sea surface salinities for the *start* of each month, while the ocean model file contains climatological sea surface salinities for the *midpoint* of each month.

E.4.3 Ocean currents

The ocean currents used to spin up the atmosphere model are derived from ocean model spin-up runs, avoiding any need to apply flux adjustments to the ocean currents within the coupled model. No spatial interpolation is required, as the data is already on the model grid. However, the following steps were taken to generate the auxiliary file required by the Mk3L atmosphere model:

1. Climatological monthly-mean ocean currents were calculated from the output of an ocean model spin-up run.

2. The values were multiplied by 100, to convert them from ms^{-1} to cms^{-1}, as required by the atmosphere model.

3. The climatological ocean currents for the start of each month were estimated, by averaging the climatological means for consecutive months.

4. An annual-mean correction was applied to the values at each gridpoint, ensuring that the annual-mean currents are conserved upon the temporal interpolation conducted by the model.

5. The data was written to file in the format read by the Mk3L atmosphere model.

E.5 Flux adjustments

E.5.1 Fields passed from the atmosphere model to the ocean model

Flux adjustments are applied to each of the four fields which are passed from the atmosphere model to the ocean model: the heat flux, the surface salinity tendency, and the zonal and meridional components of the surface momentum flux.

The derivation and application of the flux adjustments is outlined in Section 2.5. If F_A and F_O are the climatological surface fluxes diagnosed from atmosphere and ocean model spin-up runs respectively - in the case of the surface momentum fluxes, F_O is equal to the climatological fluxes applied to the ocean model during the spin-up run - then the flux adjustments ΔF are given by:

$$\Delta F = F_A - F_O \tag{E.7}$$

The flux adjustments are applied within the ocean model; as with the boundary conditions on the stand-alone ocean model, the coupled model requires that the flux adjustments supplied be climatological values for the midpoint of each month. The model than uses linear interpolation in time to estimate values at each timestep.

The steps taken to generate the auxiliary files containing the flux adjustments were therefore as follows:

1. Climatological monthly-mean surface fluxes were diagnosed from atmosphere and ocean model spin-up runs.
2. Monthly-mean flux adjustments were diagnosed, by subtracting the ocean model fluxes from the atmosphere model fluxes.
3. An annual-mean correction was applied to the values at each gridpoint, ensuring that the annual-mean flux adjustment is conserved upon the temporal interpolation conducted by the model.
4. The data was written to file in the format read by the Mk3L coupled model.

E.5.2 Fields passed from the ocean model to the atmosphere model

Four fields are passed from the ocean model to the atmosphere model: the sea surface temperature (SST), the sea surface salinity (SSS), and the zonal and meridional components of the surface velocity. Of these, the coupled model applies adjustments to the SST and the SSS.

The derivation and application of the adjustments is outlined in Section 2.5. The adjustments which are applied to the SST are given by:

$$\Delta T = T_{obs} + \Delta T_{mlo} - T_O \tag{E.8}$$

while the adjustments which are applied to the SSS are given by:

$$\Delta S = S_{obs} - S_O \tag{E.9}$$

The adjustments to the SST and the SSS are applied within the atmosphere model; as with the boundary conditions on the stand-alone atmosphere model, the coupled model requires that the adjustments supplied be climatological values for the *start* of each month. The model then uses linear interpolation in time to estimate values at each timestep.

The steps taken to generate the auxiliary file containing the adjustments to the SST were therefore as follows:

1. Climatological monthly-mean SSTs were diagnosed from an ocean model spin-up run, while climatological monthly-mean mixed-layer ocean temperature anomalies (ΔT_{mlo}) were diagnosed from an atmosphere model spin-up run.
2. The ocean model SSTs were interpolated onto the atmosphere model grid, in exactly the same rigorously-conserving fashion as takes place within the Mk3L coupled model (Section D.4).
3. Monthly-mean adjustments were diagnosed, using Equation E.8.
4. Climatological adjustments for the start of each month were estimated, by averaging the climatological means for consecutive months.
5. An annual-mean correction was applied to the values at each gridpoint, ensuring that the annual-mean adjustment is conserved upon the temporal interpolation conducted by the model.
6. The data was written to file in the format read by the Mk3L coupled model.

Step 2 is required by the fact that the adjustments are applied to the SST on the atmosphere model grid, and they must therefore be calculated on the atmosphere model grid. In contrast, the adjustments to the SSS are applied on the ocean model grid, and so the steps taken to generate the auxiliary file were simpler:

1. Climatological monthly-mean SSSs were diagnosed from an ocean model spin-up run.
2. Monthly-mean adjustments were diagnosed, by subtracting the ocean model SSSs from observed values.
3. Climatological adjustments for the start of each month were estimated, by averaging the climatological means for consecutive months.
4. An annual-mean correction was applied to the values at each gridpoint, ensuring that the annual-mean adjustment is conserved upon the temporal interpolation conducted by the model.
5. The data was written to file in the format read by the Mk3L coupled model.

E.6 Generation of restart files

In addition to the auxiliary files which are read by Mk3L, restart files must also be supplied in order to initialise the ocean, atmosphere and coupled models.

E.6.1 Ocean model

The Fortran 90 program `generate_orest` was written, which generates restart files for the Mk3L ocean model. It performs the following tasks:

1. The data is read from an existing ocean model restart file.
2. Ocean temperatures and salinities are read from netCDF files.
3. The ocean model times, velocities and streamfunctions are reset to zero, and the temperatures and salinities are set equal to the values supplied via the netCDF files.
4. If necessary, the sizes of the index arrays `iszf`, `iezf`, `istf`, `ietf`, `isuf` and `ieuf` are reduced.
5. A new ocean model restart file is then generated.

The arrays `iszf`, `iezf`, `istf`, `ietf`, `isuf` and `ieuf` contain indices which indicate the gridpoints at which Fourier filtering should be applied (Section C.2.3). The ocean model was modified so as to reduce the size of the domain over which Fourier filtering is applied, and it was therefore necessary to make a corresponding reduction in the sizes of the index arrays.

The steps taken to generate initial temperatures and salinities on the Mk3L ocean model grid were as follows:

1. The annual-mean World Ocean Atlas 1998 temperature and salinity data had missing values - those corresponding to land - filled, using the same neighbour-averaging interpolation operation that was used to fill the sea surface temperature and salinity data.
2. These values were interpolated onto the Mk3L ocean model grid.
3. The resulting values were then saved to netCDF files: one for the temperatures, and one for the salinites.

The steps which were then taken to generate a restart file for the Mk3L ocean model were as follows:

1. The original restart file, which was supplied with the model source code, was converted to the format used by the Mk3L ocean model, using the programs `convert_ocean`, `convert_ocean_i8_to_i4` and `convert_ocean_to_new` (Section A.4.6).
2. The program `generate_orest` was then used to generate a new restart file, reading the data from the original restart file, and from the netCDF files generated above.

E.6.2 Atmosphere model

The restart file used to initialise the atmosphere model is essentially irrelevant. The only component of the model which has any significant heat capacity is the 100 m-thick mixed-layer ocean used by the sea ice model (Gordon et al., 2002), and hence it was found that the timescale required for the atmosphere model to reach equilibrium was only of order a few years. By excluding the first ten years of each atmosphere model spin-up run from any analysis, the initial conditions could be rendered irrelevant.

The atmosphere model was therefore initialised using the original restart file, which was supplied with the model source code. The Fortran 90 programs `convert_atmos_mk3` and `convert_atmos_mk3_i8_to_i4_zero_pmc` were used to convert this file to the appropriate binary format and precision, and to reset the values contained within the array `pmc` to zero (Section A.4.6). The Fortran 90 program `redate_restart_mk3` (Section 5.4) was also used to reset the date.

E.6.3 Coupled model

The atmospheric and oceanic components of the coupled model were initialised using the restart files written at the end of the appropriate spin-up runs.

The Fortran 90 program `make_zero_oflux` was written, which generates a coupled model restart file in which all the surface fluxes are set equal to zero. This file was used to initialise the coupled model for all control simulations.

Appendix F

Control files and run scripts

F.1 Introduction

This appendix reproduces control files and run scripts that were used by Phipps (2006) to spin up the atmosphere and ocean models, and to conduct a coupled model control simulation. The control files are provided in Section F.2, and the run scripts in Section F.3.

F.2 Control files

F.2.1 Atmosphere model spin-up

The following control file was used to spin up the Mk3L atmosphere model. The parameters `lcouple` and `locean`, contained within the `namelist` group `control`, are both set to `.false.`, indicating that the model is to run in stand-alone atmosphere mode.

The parameter `bpyear` is set to 0, and `csolar` is set to 1365.0, specifying a solar constant of 1365 Wm^{-2}. The model is instructed to read the CO_2 transmission coefficients from the file `co2_data.280ppm.18l`, which contains the coefficients corresponding to an atmospheric CO_2 concentration of 280 ppm. These settings configure the atmosphere model for pre-industrial conditions, consistent with PMIP2 experimental design (Section E.1).

The parameter `savefcor`, contained within the group `diagnostics`, is set to `.true.`, indicating that the surface fluxes should be calculated and saved to file. This enables flux adjustments to be diagnosed for use within the coupled model (Section 2.5).

The parameter `lastmonth`, contained within the group `control`, is set to 12, indicating that the model is to run for one calendar year at a time.

```
 &control
   lcouple=F
   locean=F
   mstep=20
   ndstop=0
```

```
  months=0
  lastmonth=12
  nsstop=0
  nrad=6
  co2_datafile='co2_data.280ppm.18l'
  o3_datafile='o3_data.18l'
  filewrflag=T
  irfilename='rest.c15.start'
  orfilename='rest.c15.end'
  runtype='c15'
  qflux=F
  ncarpbl=T
  naerosol_d=0
  fluxadj=F
  csolar=1365.0
  bpyear=0
  subice=T
  rcritl=0.75
  rcrits=0.85
  refac1=0.595
  refac2=0.865
&end

&diagnostics
  dynfp=F, minw=480
  iener=F
  ispec=F
  zavgp=T, dynzp=T, phz1p=T, phz2p=T, conzp=T, plotheat=T, plotclds=T,
    plotnetr=T, plotevrn=T
  gmap1=F, cvrnm=F, gwicm=F, rhnmm=F, cldm=F
  gmap2=F, rainm=F, evapm=F, pmslm=F, surfm=F
  cdmap=F
  mlomap=F
  idayp=0
  glmean_interval=240
  statsflag=T, netcdf=T
  saveqflux=T
  savefcor=T
  clforflag=T
&end

&statvars
  evp_sflg=T, pev_sflg=T, sev_sflg=T, rnd_sflg=T, rnc_sflg=T, hfl_sflg=T,
  wfg_sflg=T, wfb_sflg=T, run_sflg=T, per_sflg=T, int_sflg=T, psl_sflg=T,
  vmo_sflg=T, tax_sflg=T, tay_sflg=T, tsu_sflg=T, tsc_sflg=T, tb2_sflg=T,
  tb3_sflg=T, tgg_sflg=T, tgf_sflg=T, thd_sflg=T, tld_sflg=T, thg_sflg=T,
  tlg_sflg=T, thf_sflg=T, tlf_sflg=T, thm_sflg=T, tlm_sflg=T, dtm_sflg=T,
  rsv_sflg=F,
  cld_sflg=T, cll_sflg=T, clm_sflg=T, clh_sflg=T,
```

```
  rgn_sflg=T, rgd_sflg=T, rgc_sflg=T, sgn_sflg=T, sgd_sflg=T, sgc_sflg=T,
  rtu_sflg=T, rtc_sflg=T, sot_sflg=T, soc_sflg=T, als_sflg=T,
  snd_sflg=T, sid_sflg=T, ico_sflg=T, itf_sflg=T, isf_sflg=T, icu_sflg=T,
  icv_sflg=T, div_sflg=T, gro_sflg=T, ire_sflg=T, ich_sflg=T,
  fwf_sflg=F,
  sno_sflg=T, rev_sflg=T, ssb_sflg=T, clc_sflg=T, lwp_sflg=T, pwc_sflg=T,
  ref_sflg=T,
    u_sflg=T,   v_sflg=T,   t_sflg=T,   q_sflg=T,  rh_sflg=F,   g_sflg=T,
    c_sflg=T,   l_sflg=F
&end

&histvars
&end

&params
&end
```

F.2.2 Ocean model spin-up

The following control file was used for the asynchronous stage of a Mk3L ocean model spin-up run. The parameter `locean`, contained within the `namelist` group `control`, is set to `.true.`, indicating that the model is to run in stand-alone ocean mode.

The parameters `dttsf`, `dtuvf` and `dtsff`, contained within the group `tsteps`, set the tracer timestep to 1 day, and the velocity and streamfunction timesteps to 20 minutes.

The parameters `iocyr` and `iocmn`, contained within the group `icple`, are set to 50 and 12 respectively, indicating that the model is to run for 50 calendar years at a time.

```
&control
  locean=T
  runtype='h53'
&end
&diagnostics
&end
&statvars
&end
&histvars
&end
&params
&end

&contrl
  nnergy=3650
  ntsi=30
  nmix=19
&end

&eddy
```

```
  amf=9.0e9
  ahf=0.7e7
  fkpmf=20.0
  fkphf=1.0
  ahi1f=1.0e7
  ahi2f=1.0e7
  ahi3f=5.0e4
  slmxrf=100.0
&end

&eddy2
  ahh1f=1.25e7
  ahh2f=0.5e7
  ahh3f=5.0e4
&end

&etrans
  ahe1f=1.0e7
  ahe2f=1.0e7
  ahe3f=5.0e4
&end

&tsteps
  dttsf=86400.0
  dtuvf=1200.0
  dtsff=1200.0
&end

&parms
  mxscan=80
  sorf=1.5
  critf=1.0e8
  acorf=0.55
&end

&icple
  iocyr=50
  iocmn=12
&end

&pltg
&end

&coefs
  cdrag=2.6e-3
&end

&accel
&end
```

For the synchronous stage of the run, the namelist groups contrl and tsteps were modified as follows. The modified values of the parameters dttsf, dtuvf and dtsff set all the timesteps to 1 hour.

```
&contrl
  nnergy=87600
  ntsi=720
  nmix=19
&end

&tsteps
  dttsf=3600.0
  dtuvf=3600.0
  dtsff=3600.0
&end
```

F.2.3 Coupled model

The following control file was used to carry out a coupled model control simulation using Mk3L. The parameter lcouple, contained within the namelist group control, is set to .true., indicating that the model is to run in coupled mode.

The parameter lastmonth, also contained within the group control, is set to 12, indicating that the model is to run for one year at a time. The ocean model parameters iocyr and iocmn have no effect in the coupled model.

The atmosphere and ocean models are configured as for the respective spin-up runs. The only physical parameter to have a different value is fluxadj, contained within the group control; this is set to .true., indicating that flux adjustments should be applied.

```
&control
  lcouple=T
  locean=F
  mstep=20
  ndstop=0
  months=0
  lastmonth=12
  nsstop=0
  nrad=6
  co2_datafile='co2_data.280ppm.18l'
  o3_datafile='o3_data.18l'
  filewrflag=T
  irfilename='rest.d73.start'
  orfilename='rest.d73.end'
  runtype='d73'
  qflux=F
  ncarpbl=T
  naerosol_d=0
  fluxadj=T
  csolar=1365.0
```

```
  bpyear=0
  subice=T
  refac1=0.595
  refac2=0.865
&end

&diagnostics
  dynfp=F, minw=480
  iener=F
  ispec=F
  zavgp=T, dynzp=T, phz1p=T, phz2p=T, conzp=T, plotheat=T, plotclds=T,
    plotnetr=T, plotevrn=T
  gmap1=F, cvrnm=F, gwicm=F, rhnmm=F, cldm=F
  gmap2=F, rainm=F, evapm=F, pmslm=F, surfm=F
  cdmap=F
  mlomap=F
  idayp=0
  glmean_interval=240
  statsflag=T, netcdf=T
  saveqflux=F
  savefcor=F
  clforflag=T
&end

&statvars
  evp_sflg=T, pev_sflg=T, sev_sflg=T, rnd_sflg=T, rnc_sflg=T, hfl_sflg=T,
  wfg_sflg=T, wfb_sflg=T, run_sflg=T, per_sflg=T, int_sflg=T, psl_sflg=T,
  vmo_sflg=T, tax_sflg=T, tay_sflg=T, tsu_sflg=T, tsc_sflg=T, tb2_sflg=T,
  tb3_sflg=T, tgg_sflg=T, tgf_sflg=T, thd_sflg=T, tld_sflg=T, thg_sflg=T,
  tlg_sflg=T, thf_sflg=T, tlf_sflg=T, thm_sflg=T, tlm_sflg=T, dtm_sflg=F,
  rsv_sflg=F,
  cld_sflg=T, cll_sflg=T, clm_sflg=T, clh_sflg=T,
  rgn_sflg=T, rgd_sflg=T, rgc_sflg=T, sgn_sflg=T, sgd_sflg=T, sgc_sflg=T,
  rtu_sflg=T, rtc_sflg=T, sot_sflg=T, soc_sflg=T, als_sflg=T,
  snd_sflg=T, sid_sflg=T, ico_sflg=T, itf_sflg=T, isf_sflg=T, icu_sflg=T,
  icv_sflg=T, div_sflg=T, gro_sflg=T, ire_sflg=T, ich_sflg=T,
  fwf_sflg=F,
  sno_sflg=T, rev_sflg=T, ssb_sflg=T, clc_sflg=T, lwp_sflg=T, pwc_sflg=T,
  ref_sflg=T,
    u_sflg=T,   v_sflg=T,   t_sflg=T,   q_sflg=T,  rh_sflg=F,   g_sflg=T,
    c_sflg=T,   l_sflg=F
&end

&histvars
&end

&params
&end
```

```
&contrl
  nnergy=87600
  ntsi=720
  nmix=19
&end

&eddy
  amf=9.0e9
  ahf=0.7e7
  fkpmf=20.0
  fkphf=1.0
  ahi1f=1.0e7
  ahi2f=1.0e7
  ahi3f=5.0e4
  slmxrf=100.0
&end

&eddy2
  ahh1f=1.25e7
  ahh2f=0.5e7
  ahh3f=5.0e4
&end

&etrans
  ahe1f=1.0e7
  ahe2f=1.0e7
  ahe3f=5.0e4
&end

&tsteps
  dttsf=3600.0
  dtuvf=3600.0
  dtsff=3600.0
&end

&parms
  mxscan=80
  sorf=1.5
  critf=1.0e8
  acorf=0.55
&end

&icple
&end

&pltg
&end

&coefs
```

```
  cdrag=2.6e-3
&end

&accel
&end
```

F.3 Run scripts

F.3.1 Atmosphere model spin-up

The following script was used to spin up the Mk3L atmosphere model on the APAC AlphaServer SC (Section A.3). It submits a job to the queueing system, and runs the atmosphere model for one year at a time. An entire node of the AlphaServer SC is used, with the model running on four processors.

The script performs the following tasks:

- The model executable, and all the necessary restart and auxiliary files, are copied from the `/short` filesystem to the `/jobfs` filesystem, using the `scfscp` command (see below).
- The model is executed for one year, and a check is performed to ensure that the run completed successfully.
- The restart files and model output are copied back to the `/short` filesystem.
- At an interval specified by the value of `SAVE_INTERVAL`, the restart files and model output are archived to tape, using the `netmv` command (see below).
- If the run is not complete (the total duration of the run is specified by the value of `LASTYR`), the script re-submits itself, and the run continues for another year.

The AlphaServer SC had a number of filesystems, and runtime performance could be enhanced by using them effectively. The `/short` filesystem could be accessed from all nodes, and was therefore used for short-term storage of model output. However, from most nodes, this was a remote filesystem, and was accessed via a network. It was therefore unsuitable for running the model, as the relatively high latency and low bandwidth associated with accessing files via a network can seriously impact upon runtime performance. It was also unsuitable for long-term storage of model output; being intended only for short-term storage, files which had not been accessed for 28 days were automatically deleted.

Instead, the `/jobfs` filesystem was used for execution of the model. This filesystem was local to each node, and therefore had much better I/O characteristics than `/short`. However, the local nature of the filesystem required that the model executables, and all the necessary restart and auxiliary files, be copied to `/jobfs` at the start of each job. The restart files and model output then had to be copied back at the end. The `scfscp` command was used to copy files between filesystems; this is a version of the `cp` command which had been optimised for the AlphaServer SC, and which gave superior data transfer rates.

For long-term storage of model output, a tape store was available. This filesystem could not be accessed directly, but the `netmv` command could be used to move files to tape.

```
#!/bin/tcsh
#PBS -P g11
#PBS -q normal
#PBS -l walltime=2:15:00
#PBS -l vmem=150MB
#PBS -l ncpus=4
#PBS -l jobfs=2GB
#PBS -m a
#PBS -M sjphipps@utas.edu.au
#PBS -N c15
#PBS -wd
#
# Purpose
# -------
# Runs the CSIRO AGCM for one year at a time on the SC
#
# History
# -------
# 2004 Sep 11   Steve Phipps    Original version
# 2004 Sep 24   Steve Phipps    Modified for five-digit year numbers

############################################################
#                                                          #
#                     USER INTERFACE                       #
#                                                          #
#  Set the following variables to the required values      #
#                                                          #
############################################################

# Run name
set run = c15

# Duration of the run, in years
set LASTYR = 00050

# How often to save model output to the mass data store, in years
set SAVE_INTERVAL = 50

# Address to which to send emails
set ADDRESS = sjphipps@utas.edu.au

##########################################################
#                                                        #
#                   RUN THE MODEL                        #
#                                                        #
#  You shouldn't have to change anything after here      #
#                                                        #
##########################################################
```

```
# Display the node on which we're running
echo This job is running on node `hostname`.

# Set the number of threads
setenv OMP_NUM_THREADS 4

# Set directory names
set copydir = /short/e56/${run}/copy
set rundir = /short/e56/${run}/run
set tmpdir = /short/e56/${run}/tmp
set atdir = /massdata/e00/csiro/${run}/atmos
set fcordir = /massdata/e00/csiro/${run}/fcor
set outdir = /massdata/e00/csiro/${run}/out
set qfluxdir = /massdata/e00/csiro/${run}/qflux
set restdir = /massdata/e00/csiro/${run}/restart

# Set the year number - $year may have leading zeroes, whereas $yr will not
set year = `cat qrun.yrmodel`
set yr = `expr $year + 0`

# Change to the run directory
cd $rundir

# Tidy up from the previous run
/bin/mv ${run}.[eo]* $tmpdir

# Copy everything to /jobfs
/bin/cp -p * $PBS_JOBFS
scfscp ${copydir}/copy.tar ${PBS_JOBFS}/copy.tar
cd $PBS_JOBFS

# Extract files from copy.tar
tar xf copy.tar
/bin/rm copy.tar

# Run the model for one year
./model < input.start > out.year$year

# Check that the run was OK
set MESSAGE = `tail out.year$year | grep termination`
if ("$MESSAGE" != '') then
  echo "Year $year terminated normally."
else
  echo "Abnormal termination of run."
  echo "CSIRO run $run stopped - abnormal termination." | mail $ADDRESS
  exit
endif

# Save restart files every SAVE_INTERVAL years
```

```
if (‘expr $yr % $SAVE_INTERVAL‘ == 0) then
  /bin/cp rest.${run}.end rest${year}12.$run
  gzip rest${year}12.$run
  chmod 400 rest${year}12.${run}.gz
  scfscp ${PBS_JOBFS}/rest${year}12.${run}.gz ${tmpdir}/rest${year}12.${run}.gz
endif

# Rename the output restart file as the new input restart file
/bin/mv rest.${run}.end rest.${run}.start

# Save the standard output of the model
gzip out.year$year
chmod 400 out.year${year}.gz
scfscp ${PBS_JOBFS}/out.year${year}.gz ${tmpdir}/out.year${year}.gz

# Save the Q-flux output of the model
tar cf qflx${year}.${run}.tar qflx*.$run
gzip qflx${year}.${run}.tar
chmod 400 qflx${year}.${run}.tar.gz
scfscp ${PBS_JOBFS}/qflx${year}.${run}.tar.gz \
       ${tmpdir}/qflx${year}.${run}.tar.gz

# Save the surface flux output of the model
tar cf fcor${year}.${run}.tar fcor*.$run
gzip fcor${year}.${run}.tar
chmod 400 fcor${year}.${run}.tar.gz
scfscp ${PBS_JOBFS}/fcor${year}.${run}.tar.gz \
       ${tmpdir}/fcor${year}.${run}.tar.gz

# Move the restart files and netCDF files back to /short
if (‘expr $yr % $SAVE_INTERVAL‘ == 0) then
  tar cf copy.tar r*.${run}.start
else
  tar cf copy.tar r*.${run}.start s*${run}.nc
endif
/bin/mv ${copydir}/copy.tar ${copydir}/copy.tar.old
scfscp ${PBS_JOBFS}/copy.tar ${copydir}/copy.tar

# Every SAVE_INTERVAL years, save all the model output to the mass data store
if (‘expr $yr % $SAVE_INTERVAL‘ == 0) then
  set yr2 = $year
  set year1 = ‘expr $yr - $SAVE_INTERVAL + 1‘
  set yr1 = $year1
  if ($year1 <= 9999) set yr1 = 0$year1
  if ($year1 <= 999) set yr1 = 00$year1
  if ($year1 <= 99) set yr1 = 000$year1
  if ($year1 <= 9) set yr1 = 0000$year1

  # Compress and tar the atmosphere model output, copy it to /short and then
```

```
  # save it to the mass data store
  gzip s*${run}.nc
  chmod 400 s*${run}.nc.gz
  tar cf netcdf.${run}.${yr1}-${yr2}.tar s*${run}.nc.gz
  scfscp ${PBS_JOBFS}/netcdf.${run}.${yr1}-${yr2}.tar \
         ${tmpdir}/netcdf.${run}.${yr1}-${yr2}.tar
  cd $tmpdir
  chmod 400 netcdf.${run}.${yr1}-${yr2}.tar
  netmv -N netcdf.$run netcdf.${run}.${yr1}-${yr2}.tar $atdir

  # Tar the restart files, and save them to the mass data store
  tar cf restart.${run}.${yr2}.tar rest*12.${run}.gz
  chmod 400 restart.${run}.${yr2}.tar
  netmv -N restart.$run restart.${run}.${yr2}.tar $restdir
  chmod 600 rest*12.${run}.gz
  /bin/rm rest*12.${run}.gz

  # Tar the standard output of the model, and save it to the mass data store
  tar cf out.${run}.${yr1}-${yr2}.tar out.year*
  chmod 400 out.${run}.${yr1}-${yr2}.tar
  netmv -N out.$run out.${run}.${yr1}-${yr2}.tar $outdir
  chmod 600 out.year*
  /bin/rm out.year*

  # Tar the Q-flux output, and save it to the mass data store
  tar cvf qflx.${run}.${yr1}-${yr2}.tar qflx*.${run}.tar.gz
  chmod 400 qflx.${run}.${yr1}-${yr2}.tar
  netmv -N qflx.$run qflx.${run}.${yr1}-${yr2}.tar $qfluxdir
  chmod 600 qflx*.${run}.tar.gz
  /bin/rm qflx*.${run}.tar.gz

  # Tar the surface flux output, and save it to the mass data store
  tar cvf fcor.${run}.${yr1}-${yr2}.tar fcor*.${run}.tar.gz
  chmod 400 fcor.${run}.${yr1}-${yr2}.tar
  netmv -N fcor.$run fcor.${run}.${yr1}-${yr2}.tar $fcordir
  chmod 600 fcor*.${run}.tar.gz
  /bin/rm fcor*.${run}.tar.gz

endif

# Change to the run directory
cd $rundir

# Increment the year number
set yrnext = ‘expr $year + 1‘
set yrp1 = $yrnext
if ($yrnext <= 9999) set yrp1 = 0$yrnext
if ($yrnext <= 999) set yrp1 = 00$yrnext
if ($yrnext <= 99) set yrp1 = 000$yrnext
```

```
if ($yrnext <= 9) set yrp1 = 0000$yrnext
/bin/rm qrun.yrmodel
echo $yrp1 > qrun.yrmodel

# Stop the run if the file stop_run exists
if (-e ${rundir}/stop_run) then
  echo "Model stopped - stop_run exists."
  echo "CSIRO run $run stopped - stop_run exists." | mail $ADDRESS
  exit
endif

# Continue the run as necessary
if ($yrnext <= $LASTYR) then
  qsub RUN_$run
  set errstat = $?
  @ n = 1
  while (($errstat != 0) && ($n <= 10))
    echo "qsub error - trying again in 60 seconds..."
    sleep 60
    qsub RUN_$run
    set errstat = $?
    @ n ++
  end
endif
```

F.3.2 Ocean model spin-up

The following script was used to conduct the asynchronous stage of a Mk3L ocean model spin-up run on the APAC Linux Cluster (Section A.3). It exhibits the following differences from the script used to run the atmosphere model:

- The Linux Cluster consists of single-processor nodes, and so the model is run on a single processor.
- The `cp` and `mv` commands are used to copy files, as the `scfscp` command is not available on the Linux Cluster.
- The model is executed for 50 years at a time, rather than one.
- After 1,000 model years, the auxiliary files `sss.dat`, `sst.dat` and `stress.dat` are replaced, in order to change from annual-mean to seasonally-varying forcing.
- The utility `convert_averages` is used to convert the model output to netCDF.
- The utility `overturning` is used to calculate the meridional overturning streamfunctions.
- The utilities `annual_averages` and `annual_overturning` are used to calculate annual-mean model output and annual-mean meridional overturning streamfunctions respectively.
- To reduce the amount of model output, the variable `MONTHYR` allows the user to specify the model year from which to begin saving monthly-mean model output; prior to this year only annual-mean output is

archived to tape. The following script specifies that monthly-mean output is only to be saved during the final 100 years of the run (from model year 3901, the value of MONTHYR, to model year 4000, the value of LASTYR).

- The standard output of the model is only archived to tape once every 1,000 model years (the value of TAR_INTERVAL); otherwise the archive files are smaller than 1 MB, which is the smallest allowable file size on the tape store.

The script used to conduct the synchronous stage of the run differed only in that the values of LASTYR and TAR_INTERVAL were changed to 4500 and 500 respectively, and the amount of walltime requested per job was increased from 2 hours to 20 hours (i.e. `#PBS -l walltime=2:00:00` was replaced with `#PBS -l walltime=20:00:00`).

```
#!/bin/tcsh
#PBS -P e00
#PBS -q normal
#PBS -l walltime=2:00:00
#PBS -l vmem=200MB
#PBS -l ncpus=1
#PBS -l jobfs=4GB
#PBS -m a
#PBS -M sjphipps@utas.edu.au
#PBS -N h53
#PBS -wd
#
# Purpose
# -------
# Runs the CSIRO stand-alone OGCM on the LC. Conversion of the model output
# to netCDF, and calculation of the meridional overturning streamfunctions, now
# takes place at runtime.
#
# History
# -------
# 2004 Aug 18   Steve Phipps    Original version
# 2004 Sep 15   Steve Phipps    Modified for 5-digit year numbers
# 2004 Sep 29   Steve Phipps    Modified for new directory structure

##########################################################
#                                                        #
#                    USER INTERFACE                      #
#                                                        #
#  Set the following variables to the required values    #
#                                                        #
##########################################################

# Run name
set run = h53

# Duration of the run, in years
```

```
set LASTYR = 04000

# Year from which to begin saving monthly output
set MONTHYR = 03901

# Length of each job, in model years
set RUN_LENGTH = 50

# Interval over which to tar standard output, in years
set TAR_INTERVAL = 1000

# Names of seasonal forcing files
set SST = sst.dat.levitus1998_v2a_mid
set SSS = sss.dat.levitus1998_v2d_mid
set STRESS = stress.dat.ncep2_v2_mid

# Address to which to send emails
set ADDRESS = sjphipps@utas.edu.au

#########################################################
#                                                       #
#                   RUN THE MODEL                       #
#                                                       #
#  You shouldn't have to change anything after here     #
#                                                       #
#########################################################

# Display the node on which we're running
echo This job is running on node `hostname`.

# Set the stacksize to unlimited
limit stacksize unlimited

# Set directory names
set datadir = /home/581/sjp581/csiro/model/data_files/ogcm
set tmpdir = /short/e56/${run}/tmp
set rundir = /short/e56/${run}/run
set comdir = /massdata/e56/${run}/com
set outdir = /massdata/e56/${run}/out
set overdir = /massdata/e56/${run}/over
set restdir = /massdata/e56/${run}/restart

# Set the year number for the first year of the run - $year1 may have leading
# zeroes, whereas $yr1 will not
set year1 = `cat qrun.yrmodel`
set yr1 = `expr $year1 + 0`

# Set the year number for the last year of the run
set yr2 = `expr $yr1 + $RUN_LENGTH - 1`
```

```
set year2 = $yr2
if ($yr2 <= 9999) set year2 = 0$yr2
if ($yr2 <= 999) set year2 = 00$yr2
if ($yr2 <= 99) set year2 = 000$yr2
if ($yr2 <= 9) set year2 = 0000$yr2

# Change to the run directory
cd $rundir

# Tidy up from the previous run
/bin/mv ${run}.[eo]* $tmpdir

# If this is year 01001, get the seasonal forcing files
if ($year1 == 01001) then
  /bin/cp ${datadir}/sss/$SSS sss.dat
  /bin/cp ${datadir}/sst/$SST sst.dat
  /bin/cp ${datadir}/stress/$STRESS stress.dat
endif

# Copy everything to /jobfs
/bin/cp -p * $PBS_JOBFS
cd $PBS_JOBFS

# Rename the restart file
/bin/mv orest.${run}.start fort.21

# Run the model
./model1 < input.start > out.${year1}-${year2}

# Check that the run was OK
set MESSAGE = ‘tail out.${year1}-${year2} | grep termination‘
if ("$MESSAGE" != ’’) then
  echo "Years $year1 to $year2 terminated normally."
else
  echo "Abnormal termination of run."
  echo "CSIRO run $run stopped - abnormal termination." | mail $ADDRESS
  exit
endif

# Convert the model output to netCDF
/bin/mv fort.40 fort.40.${year1}-${year2}
./convert_averages_ogcm $run $yr1 $yr2

# Calculate the meridional overturning streamfunctions
@ year = $yr1
while ($year <= $yr2)
  set years = $year
  if ($year < 10000) set years = 0$year
  if ($year < 1000) set years = 00$year
```

```
  if ($year < 100) set years = 000$year
  if ($year < 10) set years = 0000$year
  set com = com.${run}.${years}.nc
  set over = over.${run}.${years}.nc
  ./overturning $com $over
  @ year ++
end

# Calculate the annual means
./annual_averages $run $yr1 $yr2
./annual_overturning $run $yr1 $yr2

# If necessary, compress and tar the monthly output
set comtar = com.${run}.${year1}-${year2}.tar
set overtar = over.${run}.${year1}-${year2}.tar
if ($yr1 >= $MONTHYR) then

  # OGCM output
  chmod 400 com.${run}.?????.nc
  gzip com.${run}.?????.nc
  tar cf $comtar com.${run}.?????.nc.gz
  chmod 400 $comtar

  # Overturning data
  chmod 400 over.${run}.?????.nc
  gzip over.${run}.?????.nc
  tar cf $overtar over.${run}.?????.nc.gz
  chmod 400 $overtar

endif

# Compress the annual averages
chmod 400 com.ann.${run}.${year1}-${year2}.nc \
  over.ann.${run}.${year1}-${year2}.nc
gzip com.ann.${run}.${year1}-${year2}.nc \
  over.ann.${run}.${year1}-${year2}.nc

# Copy the restart file
/bin/cp fort.21 orest.${run}.start

# Save the restart file
/bin/mv fort.21 orest${year2}12.$run
gzip orest${year2}12.$run
chmod 400 orest${year2}12.${run}.gz

# Save standard output
gzip out.${year1}-${year2}
chmod 400 out.${year1}-${year2}.gz
```

```
# Move the output to /short
/bin/mv com.ann.${run}.${year1}-${year2}.nc.gz \
  over.ann.${run}.${year1}-${year2}.nc.gz orest.${run}.start \
  orest${year2}12.${run}.gz out.${year1}-${year2}.gz $tmpdir
if ($yr1 >= $MONTHYR) /bin/mv $comtar $overtar $tmpdir

# Change back to /short, and save the output to /massdata
cd $tmpdir
netmv -N com.ann.$run com.ann.${run}.${year1}-${year2}.nc.gz $comdir
netmv -N over.ann.$run over.ann.${run}.${year1}-${year2}.nc.gz $overdir
netmv -N rest.$run orest${year2}12.${run}.gz $restdir
if ($yr1 >= $MONTHYR) then
  netmv -N com.$run $comtar $comdir
  netmv -N over.$run $overtar $overdir
endif

# Save standard output to /massdata
if (‘expr $yr2 % $TAR_INTERVAL‘ == 0) then
  set yr0 = ‘expr $yr2 - $TAR_INTERVAL + 1‘
  set year0 = $yr0
  if ($yr0 <= 9999) set year0 = 0$yr0
  if ($yr0 <= 999) set year0 = 00$yr0
  if ($yr0 <= 99) set year0 = 000$yr0
  if ($yr0 <= 9) set year0 = 0000$yr0
  tar cf out.${run}.${year0}-${year2}.tar out.?????-?????.gz
  chmod 400 out.${run}.${year0}-${year2}.tar
  netmv -N out.$run out.${run}.${year0}-${year2}.tar $outdir
  chmod 600 out.?????-?????.gz
  /bin/rm out.?????-?????.gz
endif

# Move the restart file to the run directory
/bin/mv orest.${run}.start $rundir

# Change back to the run directory
cd $rundir

# Increment the year number
set yrnext = ‘expr $year2 + 1‘
set yrp1 = $yrnext
if ($yrnext <= 9999) set yrp1 = 0$yrnext
if ($yrnext <= 999) set yrp1 = 00$yrnext
if ($yrnext <= 99) set yrp1 = 000$yrnext
if ($yrnext <= 9) set yrp1 = 0000$yrnext
/bin/rm qrun.yrmodel
echo $yrp1 > qrun.yrmodel

# Stop the run if the file stop_run exists
if (-e ${rundir}/stop_run) then
```

```
  echo "Model stopped - stop_run exists."
  echo "CSIRO run $run stopped - stop_run exists." | mail $ADDRESS
  exit
endif

# Continue the run as necessary
if ($yrnext <= $LASTYR) then
  qsub RUN_$run
  set errstat = $?
  @ n = 1
  while (($errstat != 0) && ($n <= 10))
    echo "qsub error - trying again in 60 seconds..."
    sleep 60
    qsub RUN_$run
    set errstat = $?
    @ n ++
  end
endif
```

F.3.3 Coupled model

The following script was used to conduct a coupled model control simulation using Mk3L on the APAC AlphaServer SC (Section A.3). It combines the features of the scripts used to conduct the atmosphere and ocean model spin-up runs, executing the coupled model for one year at a time on four processors, and archiving the restart files and model output to tape once every 50 years.

```
#!/bin/tcsh
#PBS -P e56
#PBS -q normal
#PBS -l walltime=2:30:00
#PBS -l vmem=200MB
#PBS -l ncpus=4
#PBS -l jobfs=2GB
#PBS -m a
#PBS -M sjphipps@utas.edu.au
#PBS -N d73
#PBS -wd
#
# Purpose
# -------
# Runs the CSIRO coupled model for one year at a time on the SC
#
# History
# -------
# 2004 Nov 20   Steve Phipps    Original version

############################################################
#                                                          #
```

```
#                    USER INTERFACE                    #
#                                                      #
#  Set the following variables to the required values  #
#                                                      #
########################################################

# Run name
set run = d73

# Duration of the run, in years
set LASTYR = 10000

# How often to save model output to the mass data store, in years
set SAVE_INTERVAL = 50

# Address to which to send emails
set ADDRESS = sjphipps@utas.edu.au

########################################################
#                                                      #
#                   RUN THE MODEL                      #
#                                                      #
#  You shouldn't have to change anything after here    #
#                                                      #
########################################################

# Display the node on which we're running
echo This job is running on node `hostname`.

# Set the number of threads
setenv OMP_NUM_THREADS 4

# Set names of directories on the SC
set copydir = /short/e56/${run}/copy
set rundir = /short/e56/${run}/run
set tmpdir = /short/e56/${run}/tmp

# Set names of directories on the mass data store
set atdir = /massdata/e56/${run}/atmos
set comdir = /massdata/e56/${run}/com
set outdir = /massdata/e56/${run}/out
set overdir = /massdata/e56/${run}/over
set restdir = /massdata/e56/${run}/restart

# Set the year number - $year may have leading zeroes, whereas $yr will not
set year = `cat qrun.yrmodel`
set yr = `expr $year + 0`

# Change to the run directory
```

```
cd $rundir

# Tidy up from the previous run
/bin/mv ${run}.[eo]* $tmpdir

# Copy everything to /jobfs
/bin/cp -p * $PBS_JOBFS
scfscp ${copydir}/copy.tar ${PBS_JOBFS}/copy.tar
cd $PBS_JOBFS

# Extract files from copy.tar
tar xf copy.tar
/bin/rm copy.tar

# Rename ocean restart files
/bin/mv oflux.${run}.start fort.23
/bin/mv orest.${run}.start fort.21

# Run the model for one year
./model < input.start > out.year$year

# Check that the run was OK
set MESSAGE = ‘tail out.year$year | grep termination‘
if ("$MESSAGE" != ’’) then
  echo "Year $year terminated normally."
else
  echo "Abnormal termination of run."
  echo "CSIRO run $run stopped - abnormal termination." | mail $ADDRESS
  exit
endif

# Rename ocean restart files
/bin/mv fort.13 oflux.${run}.end
/bin/mv fort.21 orest.${run}.end

# Save restart files every SAVE_INTERVAL years
if (‘expr $yr % $SAVE_INTERVAL‘ == 0) then
  foreach file (rest oflux orest)
    /bin/cp ${file}.${run}.end ${file}${year}12.$run
    gzip ${file}${year}12.$run
    chmod 400 ${file}${year}12.${run}.gz
    scfscp ${PBS_JOBFS}/${file}${year}12.${run}.gz \
           ${tmpdir}/${file}${year}12.${run}.gz
  end
endif

# Rename the output restart files as the new input restart files
/bin/mv rest.${run}.end rest.${run}.start
/bin/mv oflux.${run}.end oflux.${run}.start
```

```
/bin/mv orest.${run}.end orest.${run}.start

# Save the standard output of the model
gzip out.year$year
chmod 400 out.year${year}.gz
scfscp ${PBS_JOBFS}/out.year${year}.gz ${tmpdir}/out.year${year}.gz

# Convert the ocean model output to netCDF
./convert_averages fort.40 com.${run}.${year}.nc

# Calculate the meridional overturning streamfunctions
./overturning com.${run}.${year}.nc over.${run}.${year}.nc

# Save the output of the ocean model
chmod 400 com.${run}.${year}.nc over.${run}.${year}.nc
scfscp ${PBS_JOBFS}/com.${run}.${year}.nc ${tmpdir}/com.${run}.${year}.nc
scfscp ${PBS_JOBFS}/over.${run}.${year}.nc ${tmpdir}/over.${run}.${year}.nc

# Move the restart files and netCDF files back to /short
if (‘expr $yr % $SAVE_INTERVAL‘ == 0) then
  tar cf copy.tar *.${run}.start
else
  tar cf copy.tar *.${run}.start s*${run}.nc
endif
/bin/mv ${copydir}/copy.tar ${copydir}/copy.tar.old
scfscp ${PBS_JOBFS}/copy.tar ${copydir}/copy.tar

# Every SAVE_INTERVAL years, calculate the annual means of the ocean model
# output and of the meridional overturning streamfunctions, and save all the
# model output to the mass data store
if (‘expr $yr % $SAVE_INTERVAL‘ == 0) then
  set yr2 = $year
  set year1 = ‘expr $yr - $SAVE_INTERVAL + 1‘
  set yr1 = $year1
  if ($year1 <= 9999) set yr1 = 0$year1
  if ($year1 <= 999) set yr1 = 00$year1
  if ($year1 <= 99) set yr1 = 000$year1
  if ($year1 <= 9) set yr1 = 0000$year1

  # Compress and tar the atmosphere model output, copy it to /short and then
  # save it to the mass data store
  gzip s*${run}.nc
  chmod 400 s*${run}.nc.gz
  tar cf netcdf.${run}.${yr1}-${yr2}.tar s*${run}.nc.gz
  scfscp ${PBS_JOBFS}/netcdf.${run}.${yr1}-${yr2}.tar \
         ${tmpdir}/netcdf.${run}.${yr1}-${yr2}.tar
  cd $tmpdir
  chmod 400 netcdf.${run}.${yr1}-${yr2}.tar
  netmv -N netcdf.$run netcdf.${run}.${yr1}-${yr2}.tar $atdir
```

```
  # Calculate the annual means of the ocean model output and of the meridional
  # overturning streamfunctions, compress them and save them to the mass data
  # store
  ./annual_averages $run $yr1 $yr2
  ./annual_overturning $run $yr1 $yr2
  chmod 400 com.ann.${run}.${yr1}-${yr2}.nc over.ann.${run}.${yr1}-${yr2}.nc
  gzip com.ann.${run}.${yr1}-${yr2}.nc over.ann.${run}.${yr1}-${yr2}.nc
  netmv -N com.ann.$run com.ann.${run}.${yr1}-${yr2}.nc.gz $comdir
  netmv -N over.ann.$run over.ann.${run}.${yr1}-${yr2}.nc.gz $overdir

  # Compress and tar the ocean model output, and save it to the mass data store
  chmod 400 com.${run}.?????.nc
  gzip com.${run}.?????.nc
  set comtar = com.${run}.${yr1}-${yr2}.tar
  tar cf $comtar com.${run}.?????.nc.gz
  chmod 400 $comtar
  netmv -N com.$run $comtar $comdir
  chmod 600 com.${run}.?????.nc.gz
  /bin/rm com.${run}.?????.nc.gz

  # Compress and tar the overturning data, and save it to the mass data store
  chmod 400 over.${run}.?????.nc
  gzip over.${run}.?????.nc
  set overtar = over.${run}.${yr1}-${yr2}.tar
  tar cf $overtar over.${run}.?????.nc.gz
  chmod 400 $overtar
  netmv -N over.$run $overtar $overdir
  chmod 600 over.${run}.?????.nc.gz
  /bin/rm over.${run}.?????.nc.gz

  # Tar the restart files, and save them to the mass data store
  tar cf restart.${run}.${yr2}.tar oflux*12.${run}.gz orest*12.${run}.gz \
                                   rest*12.${run}.gz
  chmod 400 restart.${run}.${yr2}.tar
  netmv -N restart.$run restart.${run}.${yr2}.tar $restdir
  foreach file (rest oflux orest)
    chmod 600 ${file}*12.${run}.gz
    /bin/rm ${file}*12.${run}.gz
  end

  # Tar the standard output of the model, and save it to the mass data store
  tar cf out.${run}.${yr1}-${yr2}.tar out.year*
  chmod 400 out.${run}.${yr1}-${yr2}.tar
  netmv -N out.$run out.${run}.${yr1}-${yr2}.tar $outdir
  chmod 600 out.year*
  /bin/rm out.year*

endif
```

```
# Change to the run directory
cd $rundir

# Increment the year number
set yrnext = ‘expr $year + 1‘
set yrp1 = $yrnext
if ($yrnext <= 9999) set yrp1 = 0$yrnext
if ($yrnext <= 999) set yrp1 = 00$yrnext
if ($yrnext <= 99) set yrp1 = 000$yrnext
if ($yrnext <= 9) set yrp1 = 0000$yrnext
/bin/rm qrun.yrmodel
echo $yrp1 > qrun.yrmodel

# Stop the run if the file stop_run exists
if (-e ${rundir}/stop_run) then
  echo "Model stopped - stop_run exists."
  echo "CSIRO run $run stopped - stop_run exists." | mail $ADDRESS
  exit
endif

# Continue the run as necessary
if ($yrnext <= $LASTYR) then
  qsub RUN_$run
  set errstat = $?
  @ n = 1
  while (($errstat != 0) && ($n <= 10))
    echo "qsub error - trying again in 60 seconds..."
    sleep 60
    qsub RUN_$run
    set errstat = $?
    @ n ++
  end
endif
```

Appendix G

Source code

G.1 Introduction

This appendix reproduces the three subroutines which are new in Mk3L: `conserve` (Section D.4), `conserve_fw` (Section D.4) and `orbpar` (Section B.2.1).

The subroutines `conserve` and `conserve_fw` represent original source code, and are laid out in a style which is consistent with the remainder of the model source code. The subroutine `orbpar` is a modified version of the NASA/GISS Atmosphere-Ocean Model subroutine `ORBPAR.SUB` (Goddard Institute for Space Studies, 2001).

The source code for `conserve`, `conserve_fw` and `orbpar` is provided in Sections G.2, G.3 and G.4 respectively.

G.2 `conserve`

```
c Purpose
c -------
c Ensures that quantities are conserved when fields are passed between the AGCM
c and the OGCM.
c
c Values on the AGCM and OGCM grids are integrated either over land or over the
c ocean, and a correction applied to the values on the target grid in order to
c ensure that either the global integral or mean (as appropriate) is conserved.
c
c Inputs
c ------
c source        Values on the source grid
c target        Initial values on the target grid
c type          Type of conservation operation to perform:
c
c               1 = Conserve global integral AGCM ocean -> OGCM ocean
c               2 = Conserve global integral AGCM land -> OGCM land
```

```
c               3 = Conserve global mean OGCM ocean -> AGCM ocean
c
c Outputs
c -------
c source        Corrected values on the target grid
c
c Notes
c -----
c (1) We use the array SOURCE for both input and output, in order to save
c     memory.
c
c (2) Over the ocean, a uniform correction is added to/subtracted from the data
c     in order to achieve conservation. Over land, however, the data is
c     multiplied by a correction factor. These methods are the most suitable
c     for the fields which are currently being conserved - heat/freshwater
c     fluxes and SST over the ocean, and runoff over land.
c
c History
c -------
c 2003 Apr 28   Steve Phipps    Original version
c 2003 May 29   Steve Phipps    Slight modifications, including an error fix
c 2003 Jun 9    Steve Phipps    Another error fix - subroutine has now been
c                               tested and definitely works OK!
c 2003 Jun 10   Steve Phipps    Added the ability to conserve over land, as
c                               well as over the ocean
c 2004 Sep 11   Steve Phipps    Moved AOCEANO, AREA and MASKO to /CONSERVATION/
c                               so that the values can be used by CONSERVE_FW
c 2004 Nov 18   Steve Phipps    Generalised in order to support conservation of
c                               global-mean SST

      subroutine conserve(source, target, type)
      implicit none

C Global parameters
      include 'PARAMS.f'
      include 'PHYSPARAMS.f'

C Argument list
      integer type
      real source(ln2, lat), target(ln2, lat)

C Local shared common blocks
      logical masko
      real aoceano, area
      common /conservation/ aoceano, area(lat), masko(ln2, lat)

C Global data blocks
      include 'GAUSL.f'
      include 'LSMI.f'
```

```
C Local work arrays and variables
      integer i, j, nland, nocean
      real corr, totala, totalo

C Local data, functions etc
      logical first, maska(ln2, lat)
      real alanda, alando, aoceana
      data first /.true./
      save alanda, alando, aoceana, first, maska

C Start code : ----------------------------------------------------------

c On the first call to this routine, set up the AGCM and OGCM masks. These are
c logical arrays, with ocean points set to .TRUE. and land points to .FALSE.
      if (first) then

c Calculate the area of gridboxes at each latitude
        do j = 1, lat
          area(j) = 2.0 * pi * w(j) * eradsq / float(lon)
        end do

c AGCM mask - get data from array IMSL. We also set the OGCM mask equal to the
c             AGCM mask at this point
        do j = 1, lat
          do i = 1, ln2
            if (imsl(i, j) .eq. 4) then
              maska(i, j) = .false.
              masko(i, j) = .false.
            else
              maska(i, j) = .true.
              masko(i, j) = .true.
            end if
          end do
        end do

c OGCM mask - derive this by adjusting the AGCM mask accordingly
c
c (1) 11 points are treated as land by the AGCM, but as ocean by the OGCM
        masko(3, 4) = .true.
        masko(62, 8) = .true.
        masko(23, 26) = .true.
        masko(84, 27) = .true.
        masko(116, 12) = .true.
        masko(118, 8) = .true.
        masko(86, 28) = .true.
        masko(87, 28) = .true.
        masko(90, 28) = .true.
        masko(91, 28) = .true.
```

```
        masko(91, 27) = .true.

c (2) 18 points are treated as ocean by the AGCM, but as land by the OGCM
        masko(91, 16) = .false.
        masko(72, 21) = .false.
        masko(72, 22) = .false.
        masko(72, 23) = .false.
        masko(91, 25) = .false.
        masko(67, 27) = .false.
        masko(85, 28) = .false.
        masko(20, 28) = .false.
        masko(21, 28) = .false.
        masko(19, 27) = .false.
        masko(19, 26) = .false.
        masko(19, 25) = .false.
        masko(23, 18) = .false.
        masko(64, 17) = .false.
        masko(52, 9) = .false.
        masko(53, 9) = .false.
        masko(35, 8) = .false.
        masko(52, 5) = .false.

c Derive and display a summary of each mask - keep the surface areas
c according to each grid, as we will need these values later
        write (*, *)
        write (*, *) "AGCM land/sea mask"
        write (*, *) "------------------"
        nland = 0
        nocean = 0
        alanda = 0.0
        aoceana = 0.0
        do j = 1, lat
          do i = 1, ln2
            if (maska(i, j)) then
              nocean = nocean + 1
              aoceana = aoceana + area(j)
            else
              nland = nland + 1
              alanda = alanda + area(j)
            end if
          end do
        end do
        write (*, *) "Number of land grid points  = ", nland
        write (*, *) "Number of ocean grid points = ", nocean
        write (*, *) "Land area  = ", alanda/1.0e6, " km^2"
        write (*, *) "Ocean area = ", aoceana/1.0e6, " km^2"
        write (*, *) "Ocean fraction = ", aoceana / (alanda + aoceana)
        write (*, *)
        write (*, *) "OGCM land/sea mask"
```

```
        write (*, *) "------------------"
        nland = 0
        nocean = 0
        alando = 0.0
        aoceano = 0.0
        do j = 1, lat
          do i = 1, ln2
            if (masko(i, j)) then
              nocean = nocean + 1
              aoceano = aoceano + area(j)
            else
              nland = nland + 1
              alando = alando + area(j)
            end if
          end do
        end do
        write (*, *) "Number of land grid points  = ", nland
        write (*, *) "Number of ocean grid points = ", nocean
        write (*, *) "Land area  = ", alando/1.0e6, " km^2"
        write (*, *) "Ocean area = ", aoceano/1.0e6, " km^2"
        write (*, *) "Ocean fraction = ", aoceano / (alando + aoceano)
        write (*, *)

c Set FIRST to .FALSE. so that this section of code is only executed once
        first = .false.

      end if

c Perform the desired conservation operation
      if (type .eq. 1) then

c (1) Conserve the global integral over the surface of the ocean, for fields
c     passed from the AGCM to the OGCM.
c
c (1.1) Integrate the total fluxes on both the AGCM and OGCM grids
        totala = 0.0
        totalo = 0.0
        do j = 1, lat
          do i = 1, ln2
            if (maska(i, j)) totala = totala + source(i, j) * area(j)
            if (masko(i, j)) totalo = totalo + target(i, j) * area(j)
          end do
        end do
C        write (*, *)
C        write (*, *) "Total value on AGCM grid = ", totala
C        write (*, *) "Total value on OGCM grid = ", totalo
C        write (*, *)
C        write (*, *) "Mean value on AGCM grid = ", totala / aoceana
C        write (*, *) "Mean value on OGCM grid = ", totalo / aoceano
```

```
C        write (*, *)

c (1.2) Derive the correction to apply to the fluxes on the OGCM grid
        corr = (totala - totalo) / aoceano
C        write (*, *) "Correction = ", corr
C        write (*, *)

c (1.3) Correct the values on the OGCM grid, putting the corrected values into
c       the array SOURCE
        do j = 1, lat
          do i = 1, ln2
            if (masko(i, j)) then
              source(i, j) = target(i, j) + corr
            else
              source(i, j) = 0.0
            end if
          end do
        end do

      else if (type .eq. 2) then

c (2) Conserve the global integral over land, for fields passed from the AGCM
c     to the OGCM.
c
c (2.1) Integrate the total fluxes on both the AGCM and OGCM grids
        totala = 0.0
        totalo = 0.0
        do j = 1, lat
          do i = 1, ln2
            if (.not. maska(i, j)) totala = totala +
     $                                      source(i, j) * area(j)
            if (.not. masko(i, j)) totalo = totalo +
     $                                      target(i, j) * area(j)
          end do
        end do
C        write (*, *)
C        write (*, *) "Total value on AGCM grid = ", totala
C        write (*, *) "Total value on OGCM grid = ", totalo
C        write (*, *)
C        write (*, *) "Mean value on AGCM grid = ", totala / alanda
C        write (*, *) "Mean value on OGCM grid = ", totalo / alando
C        write (*, *)

c (2.2) Derive the correction to apply to the fluxes on the OGCM grid
        corr = totala / totalo
C        write (*, *) "Correction factor = ", corr
C        write (*, *)

c (2.3) Correct the values on the OGCM grid, putting the corrected values into
```

```
c       the array SOURCE
        do j = 1, lat
          do i = 1, ln2
            if (.not. masko(i, j)) then
              source(i, j) = corr * target(i, j)
            else
              source(i, j) = 0.0
            end if
          end do
        end do

      else if (type .eq. 3) then

c (3) Conserve the global mean over the surface of the ocean, for fields
c     passed from the OGCM to the AGCM.
c
c (3.1) Integrate the total fluxes on both the AGCM and OGCM grids
        totalo = 0.0
        totala = 0.0
        do j = 1, lat
          do i = 1, ln2
            if (masko(i, j)) totalo = totalo + source(i, j) * area(j)
            if (maska(i, j)) totala = totala + target(i, j) * area(j)
          end do
        end do
C        write (*, *)
C        write (*, *) "Total value on OGCM grid = ", totalo
C        write (*, *) "Total value on AGCM grid = ", totala
C        write (*, *)
C        write (*, *) "Mean value on OGCM grid = ", totalo / aoceano
C        write (*, *) "Mean value on AGCM grid = ", totala / aoceana
C        write (*, *)

c (3.2) Derive the correction to apply to the values on the AGCM grid
        corr = totalo / aoceano - totala / aoceana
C        write (*, *) "Correction = ", corr
C        write (*, *)

c (3.3) Correct the values on the AGCM grid, putting the corrected values into
c       the array SOURCE
        do j = 1, lat
          do i = 1, ln2
            if (maska(i, j)) then
              source(i, j) = target(i, j) + corr
            else
              source(i, j) = 0.0
            end if
          end do
        end do
```

```
      else

c The value of TYPE that has been passed is illegal

        write (*, *)
        write (*, *) "Type = ", type
        write (*, *)
        write (*, *) "STOP: Illegal value for type"
        write (*, *)
        stop

      end if

      return
      end subroutine
```

G.3 conserve_fw

```
c Purpose
c -------
c Corrects for the conservation error that arises when fluxes of water into the
c ocean are converted into surface salinity tendencies.
c
c The three components of the surface water flux (DSIS, DSISB and DSFW) are
c each integrated over the surface of the ocean, and then converted into the
c equivalent rate of change of salinity of the entire ocean.
c
c A uniform correction is then applied to the values of ATSF, in order to
c ensure that they give the correct rate of change of salinity of the entire
c ocean, thus conserving freshwater.
c
c Inputs
c ------
c DSIS          Flux of water arising from ice growth/melt
c DSISB         Flux of water arising from ice sublimation
c DSFW          Flux of water arising from (P-E+RUNOFF)
c ATSF          Initial values for the total surface salinity tendency
c
c Outputs
c -------
c ATSF          Corrected values for the total surface salinity tendency
c
c Notes
c -----
c (1) CONSERVE must be called before the first call to this routine, in order
c     for AOCEANO, AREA and MASKO to be initialised.
c
```

```
c (2) We assume that the average salinity of the ocean is 34.70 psu when
c     calculating the rate of change of salinity of the entire ocean, this
c     being the value according to Levitus 1998. In theory, we should use the
c     actual average salinity of the ocean when we are running in coupled mode.
c     However, this should never differ significantly from 34.70 psu, and so
c     any error introduced should be utterly negligible. Furthermore, this
c     approach ensures consistency between the stand-alone AGCM and the coupled
c     model.
c
c (3) The value of 1.2787352e18 m^3 for the volume of the ocean is that
c     calculated by the model itself.
c
c History
c -------
c 2004 Sep 13   Steve Phipps    Original version

      subroutine conserve_fw(dsis, dsisb, dsfw, atsf)

      implicit none

C Global parameters
      include 'PARAMS.f'

C Argument list
      real dsis(ln2, lat), dsisb(ln2, lat), dsfw(ln2, lat),
     &     atsf(ln2, lat)

C Local shared common blocks
      logical masko
      real aoceano, area
      common /conservation/ aoceano, area(lat), masko(ln2, lat)

C Global data blocks
      include 'TIMEX.f'

C Local work arrays and variables
      integer i, j
      real corr, dsdt1, dsdt2, total, total1, total2, total3

C Local data, functions etc
      real dz
      real salice
      real saloce
      real volume
      parameter (dz = 25.0)
      parameter (salice = 0.01)
      parameter (saloce = 0.0347)
      parameter (volume = 1.2787352e18)
```

```
C Start code : ----------------------------------------------------------

c Integrate the water fluxes, and the initial values for ATSF, over the surface
c of the ocean
      total = 0.0
      total1 = 0.0
      total2 = 0.0
      total3 = 0.0
      do j = 1, lat
        do i = 1, ln2
          if (masko(i, j)) then
            total = total + atsf(i, j) * area(j)
            total1 = total1 + dsis(i, j) * area(j)
            total2 = total2 + dsisb(i, j) * area(j)
            total3 = total3 + dsfw(i, j) * area(j)
          end if
        end do
      end do

c Calculate the equivalent rates of change in the salinity of the ocean
      dsdt1 = (total1 * (salice - saloce) + total2 * salice -
     &         total3 * saloce) / (dt * volume)
      dsdt2 = total * dz / volume

c Calculate the correction to apply to the initial values of ATSF
      corr = (dsdt1 - dsdt2) * volume / (aoceano * dz)

c Correct the initial values of ATSF
      do j = 1, lat
        do i = 1, ln2
          if (masko(i, j)) then
            atsf(i, j) = atsf(i, j) + corr
          else
            atsf(i, j) = 0.0
          end if
        end do
      end do

c Write a summary to standard output
C      write (*, *)
C      write (*, *) "TOTAL1 = ", total1/dt, " m^3/s^-1"
C      write (*, *) "TOTAL2 = ", total2/dt, " m^3/s^-1"
C      write (*, *) "TOTAL3 = ", total3/dt, " m^3/s^-1"
C      write (*, *)
C      write (*, *) "TOTAL = ", total*dz, " m^3/s^-1"
C      write (*, *)
C      write (*, *) "DSDT1 = ", dsdt1, " s^-1"
C      write (*, *) "DSDT2 = ", dsdt2, " s^-1"
C      write (*, *)
```

```
C      write (*, *) "CORR = ", corr, " s^-1"
C      write (*, *)

       return
       end subroutine
```

G.4 orbpar

```
c This subroutine derives the three orbital parameters required by
c the model. It was obtained from the GISS atmosphere-ocean model
c website at http://aom.giss.nasa.gov/solar.html.
c
c It has been modified to
c (1) remove some unused data
c (2) improve the use of arrays, so that the code does not reference
c     outside array bounds
c (3) return the longitude of the perihelion, rather than the
c     longitude of the sun at perihelion
c (4) return angles in degrees, rather than radians.
c
c SJP 2001/12/23

      SUBROUTINE ORBPAR (YEAR,ECCEN,OBLIQ,OMEGVP)
C****
C**** ORBPAR calculates the three orbital parameters as a function of
C**** YEAR.  The source of these calculations is: Andre L. Berger,
C**** 1978, "Long-Term Variations of Daily Insolation and Quaternary
C**** Climatic Changes", JAS, v.35, p.2362.  Also useful is: Andre L.
C**** Berger, May 1978, "A Simple Algorithm to Compute Long Term
C**** Variations of Daily Insolation", published by Institut
C**** D'Astronomie de Geophysique, Universite Catholique de Louvain,
C**** Louvain-la Neuve, No. 18.
C****
C**** Tables and equations refer to the first reference (JAS).  The
C**** corresponding table or equation in the second reference is
C**** enclosed in parentheses.
C****
C**** Input:  YEAR   = years A.D. are positive, B.C. are negative
C**** Output: ECCEN  = eccentricity of orbital ellipse
C****         OBLIQ  = latitude of Tropic of Cancer in radians
C****         OMEGVP = longitude of perihelion =
C****                = spatial angle from vernal equinox to perihelion
C****                  in radians with sun as angle vertex
C****
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (TWOPI=6.28318530717958647d0, PI180=TWOPI/360.d0)
      REAL*8 TABL10(3,47),TABL40(3,19),TABL50(3,10)
C**** Table 1 (2).  Obliquity relative to mean ecliptic of date: OBLIQD
```

```
      DATA TABL10/-2462.22D0,  31.609970D0,  251.9025D0,
     2             -857.32D0,  32.620499D0,  280.8325D0,
     3             -629.32D0,  24.172195D0,  128.3057D0,
     4             -414.28D0,  31.983780D0,  292.7251D0,
     5             -311.76D0,  44.828339D0,   15.3747D0,
     6              308.94D0,  30.973251D0,  263.7952D0,
     7             -162.55D0,  43.668243D0,  308.4258D0,
     8             -116.11D0,  32.246689D0,  240.0099D0,
     9              101.12D0,  30.599442D0,  222.9725D0,
     0              -67.69D0,  42.681320D0,  268.7810D0,
     1               24.91D0,  43.836456D0,  316.7998D0,
     2               22.58D0,  47.439438D0,  319.6023D0,
     3              -21.16D0,  63.219955D0,  143.8050D0,
     4              -15.65D0,  64.230484D0,  172.7351D0,
     5               15.39D0,   1.010530D0,   28.9300D0,
     6               14.67D0,   7.437771D0,  123.5968D0,
     7              -11.73D0,  55.782181D0,   20.2082D0,
     8               10.27D0,   0.373813D0,   40.8226D0,
     9                6.49D0,  13.218362D0,  123.4722D0,
     0                5.85D0,  62.583237D0,  155.6977D0,
     1               -5.49D0,  63.593765D0,  184.6277D0,
     2               -5.43D0,  76.438309D0,  267.2771D0,
     3                5.16D0,  45.815262D0,   55.0196D0,
     4                5.08D0,   8.448301D0,  152.5268D0,
     5               -4.07D0,  56.792709D0,   49.1382D0,
     6                3.72D0,  49.747849D0,  204.6609D0,
     7                3.40D0,  12.058272D0,   56.5233D0,
     8               -2.83D0,  75.278214D0,  200.3284D0,
     9               -2.66D0,  65.241013D0,  201.6651D0,
     0               -2.57D0,  64.604294D0,  213.5577D0,
     1               -2.47D0,   1.647247D0,   17.0374D0,
     2                2.46D0,   7.811584D0,  164.4194D0,
     3                2.25D0,  12.207832D0,   94.5422D0,
     4               -2.08D0,  63.856659D0,  131.9124D0,
     5               -1.97D0,  56.155991D0,   61.0309D0,
     6               -1.88D0,  77.448837D0,  296.2073D0,
     7               -1.85D0,   6.801054D0,  135.4894D0,
     8                1.82D0,  62.209412D0,  114.8750D0,
     9                1.76D0,  20.656128D0,  247.0691D0,
     0               -1.54D0,  48.344406D0,  256.6113D0,
     1                1.47D0,  55.145462D0,   32.1008D0,
     2               -1.46D0,  69.000534D0,  143.6804D0,
     3                1.42D0,  11.071350D0,   16.8784D0,
     4               -1.18D0,  74.291306D0,  160.6835D0,
     5                1.18D0,  11.047742D0,   27.5932D0,
     6               -1.13D0,   0.636717D0,  348.1074D0,
     7                1.09D0,  12.844549D0,   82.6496D0/
C**** Table 4 (1).  Fundamental elements of the ecliptic: ECCEN sin(pi)
      DATA TABL40/ .01860798D0,   4.207205D0,   28.620089D0,
```

```
     2             .01627522D0,   7.346091D0,  193.788772D0,
     3            -.01300660D0,  17.857263D0,  308.307024D0,
     4             .00988829D0,  17.220546D0,  320.199637D0,
     5            -.00336700D0,  16.846733D0,  279.376984D0,
     6             .00333077D0,   5.199079D0,   87.195000D0,
     7            -.00235400D0,  18.231076D0,  349.129677D0,
     8             .00140015D0,  26.216758D0,  128.443387D0,
     9             .00100700D0,   6.359169D0,  154.143880D0,
     O             .00085700D0,  16.210016D0,  291.269597D0,
     1             .00064990D0,   3.065181D0,  114.860583D0,
     2             .00059900D0,  16.583829D0,  332.092251D0,
     3             .00037800D0,  18.493980D0,  296.414411D0,
     4            -.00033700D0,   6.190953D0,  145.769910D0,
     5             .00027600D0,  18.867793D0,  337.237063D0,
     6             .00018200D0,  17.425567D0,  152.092288D0,
     7            -.00017400D0,   6.186001D0,  126.839891D0,
     8            -.00012400D0,  18.417441D0,  210.667199D0,
     9             .00001250D0,   0.667863D0,   72.108838D0/
C**** Table 5 (3).  General precession in longitude: psi
      DATA TABL50/ 7391.02D0,  31.609970D0,  251.9025D0,
     2             2555.15D0,  32.620499D0,  280.8325D0,
     3             2022.76D0,  24.172195D0,  128.3057D0,
     4            -1973.65D0,    .636717D0,  348.1074D0,
     5             1240.23D0,  31.983780D0,  292.7251D0,
     6              953.87D0,   3.138886D0,  165.1686D0,
     7             -931.75D0,  30.973251D0,  263.7952D0,
     8              872.38D0,  44.828339D0,   15.3747D0,
     9              606.35D0,    .991874D0,   58.5749D0,
     O             -496.03D0,    .373813D0,   40.8226D0/
C****
      YM1950 = YEAR-1950.
C****
C**** Obliquity from Table 1 (2):
C****   OBLIQ# = 23.320556 (degrees)             Equation 5.5 (15)
C****   OBLIQD = OBLIQ# + sum[A cos(ft+delta)]   Equation 1 (5)
C****
      SUMC = 0.
      DO 110 I=1,47
      ARG    = PI180*(YM1950*TABL10(2,I)/3600.+TABL10(3,I))
  110 SUMC   = SUMC + TABL10(1,I)*COS(ARG)
      OBLIQ  = 23.320556D0 + SUMC/3600.
C****
C**** Eccentricity from Table 4 (1):
C****   ECCEN sin(pi) = sum[M sin(gt+beta)]           Equation 4 (1)
C****   ECCEN cos(pi) = sum[M cos(gt+beta)]           Equation 4 (1)
C****   ECCEN = ECCEN sqrt[sin(pi)^2 + cos(pi)^2]
C****
      ESINPI = 0.
      ECOSPI = 0.
```

```
      DO 210 I=1,19
      ARG    = PI180*(YM1950*TABL40(2,I)/3600.+TABL40(3,I))
      ESINPI = ESINPI + TABL40(1,I)*SIN(ARG)
  210 ECOSPI = ECOSPI + TABL40(1,I)*COS(ARG)
      ECCEN  = SQRT(ESINPI*ESINPI+ECOSPI*ECOSPI)
C****
C**** Perihelion from Equation 4,6,7 (9) and Table 4,5 (1,3):
C****   PSI# = 50.439273 (seconds of degree)         Equation 7.5 (16)
C****   ZETA =  3.392506 (degrees)                   Equation 7.5 (17)
C****   PSI = PSI# t + ZETA + sum[F sin(ft+delta)]   Equation 7 (9)
C****   PIE = atan[ECCEN sin(pi) / ECCEN cos(pi)]
C****   OMEGVP = PIE + PSI + 3.14159                 Equation 6 (4.5)
C****
      PIE = ATAN2(ESINPI,ECOSPI)
      FSINFD = 0.
      DO 310 I=1,10
      ARG    = PI180*(YM1950*TABL50(2,I)/3600.+TABL50(3,I))
  310 FSINFD = FSINFD + TABL50(1,I)*SIN(ARG)
      PSI    = PI180*(3.392506D0+(YM1950*50.439273D0+FSINFD)/3600.)
      OMEGVP = MOD(PIE+PSI,TWOPI)
      IF(OMEGVP.lt.0.)  OMEGVP = OMEGVP + TWOPI
      OMEGVP = OMEGVP / PI180
C****
      RETURN
      END
```

Bibliography

Ahmad, F., and S. A. R. Sultan (1991), Annual mean surface heat fluxes in the Arabian Gulf and the net heat transport through the Strait of Hormuz, *Atmosphere-Ocean*, *29*, 54–61.

Amdahl, G. M. (1967), Validity of the single processor approach to achieving large scale computing capabilities, in *AFIPS Conference Proceedings*, volume 30, pages 483–485.

Arakawa, A., and V. R. Lamb (1977), Computational design of the basic dynamical processes of the UCLA general circulation model, in J. Chang, editor, *General Circulation Models of the Atmosphere*, volume 17 of *Methods in Computational Physics*, pages 173–265, Academic Press.

Ashworth, M. (2000), Optimisation for vector and RISC processors, in W. Zwieflhofer and N. Kreitz, editors, *Towards Teracomputing: Proceedings of the Eighth ECMWF Workshop on the Use of Parallel Processors in Meteorology*, pages 353–359, World Scientific.

Australian Partnership for Advanced Computing (2005), APAC National Facility Home Page, URL `http://nf.apac.edu.au/`, viewed 23 April 2005.

Barkstrom, B. R. (1984), The Earth Radiation Budget Experiment (ERBE), *Bulletin of the American Meteorological Society*, *65*(11), 1170–1185.

Berger, A. L. (1978), Long-term variations of daily insolation and quaternary climatic changes, *Journal of the Atmospheric Sciences*, *35*, 2362–2367.

Bi, D. (2002), *Transient and Long-Term Behaviour of the World Ocean Under Global Warming*, PhD thesis, University of Tasmania.

Black Sea Environmental Internet Node (2004), Black Sea water balance, URL `http://www.grid.unep.ch/bsein/publish/table_1.htm`, viewed 14 January 2004.

Bryan, K. (1969), A numerical method for the study of the circulation of the world ocean, *Journal of Computational Physics*, *4*, 347–376.

Bryan, K. (1984), Accelerating the convergence to equilibrium of ocean-climate models, *Journal of Physical Oceanography*, *14*, 666–673.

Bryden, H. L., J. Candela, and T. H. Kinder (1994), Exchange through the Strait of Gibraltar, in *Progress in Oceanography*, volume 33, pages 201–248, Pergamon Press.

Cess, R. D., and G. L. Potter (1987), Exploratory studies of cloud radiative forcing with a general circulation model, *Tellus*, *39A*, 460–473.

Chouinard, C., M. Béland, and N. McFarlane (1986), A simple gravity wave drag parameterization for use in medium-range weather forecast models, *Atmosphere-Ocean*, *24*(2), 91–110.

Cox, M. D. (1984), A primitive equation, 3-dimensional model of the ocean, Technical Report 1, Geophysical Fluid Dynamics Laboratory Ocean Group.

Cox, M. D. (1987), Isopycnal diffusion in a z-coordinate ocean model, *Ocean Modelling*, *74*, 1–5.

Drinkwater, K. F. (1988), On the mean and tidal currents in Hudson Strait, *Atmosphere-Ocean*, *25*(2), 252–266.

Elliott, T. I., L. D. Rotstayn, and M. R. Dix (1987), Using the CSIRO GCM Mark 2, Unpublished CSIRO Atmospheric Research manuscript.

Fels, S. B., and M. D. Schwarzkopf (1975), The simplified exchange approximation: A new method for radiative transfer calculations, *Journal of the Atmospheric Sciences*, *32*, 1475–1488.

Fels, S. B., and M. D. Schwarzkopf (1981), An efficient, accurate algorithm for calculating CO_2 15μm band cooling rates, *Journal of Geophysical Research*, *86*(C2), 1205–1232.

FFTW (2005a), benchFFT, URL `http://www.fftw.org/benchfft/`, viewed 23 April 2005.

FFTW (2005b), FFTW, URL `http://fftw.org/`, viewed 23 April 2005.

Fissel, D. B., J. R. Birch, H. Melling, and R. A. Lake (1988), Non-tidal flows in the Northwest Passage, Canadian Technical Report of Hydrography and Ocean Sciences 98, Institute of Ocean Sciences, Sidney, British Columbia.

Flato, G. M., and W. D. Hibler III (1990), On a simple sea-ice dynamics model for climate studies, *Annals of Glaciology*, *14*, 72–77.

Flato, G. M., and W. D. Hibler III (1992), Modeling pack ice as a cavitating fluid, *Journal of Physical Oceanography*, *22*, 626–651.

Ford, R. W., and D. Snelling (1997), Vector and cache performance of OCCAM, in G.-R. Hoffmann and N. Kreitz, editors, *Making Its Mark: Proceedings of the Seventh ECMWF Workshop on the Use of Parallel Processors in Meteorology*, pages 198–208, World Scientific.

Gargett, A. E. (1984), Vertical eddy diffusivity in the ocean interior, *Journal of Marine Research*, *42*(2), 359–393.

Gates, W. L., and A. B. Nelson (1975a), A new (revised) tabulation of the Scripps topography on a 1 degree global grid: Part I, Terrain heights, Technical Report R-1276-1-ARPA, The Rand Corporation.

Gates, W. L., and A. B. Nelson (1975b), A new (revised) tabulation of the Scripps topography on a 1 degree global grid: Part II, Ocean depths, Technical Report R-1277-1-ARPA, The Rand Corporation.

Gent, P. R., and J. C. McWilliams (1990), Isopycnal mixing in ocean circulation models, *Journal of Physical Oceanography*, *20*, 150–155.

Gent, P. R., J. Willebrand, T. J. McDougall, and J. C. McWilliams (1995), Parameterizing eddy-induced tracer transports in ocean circulation models, *Journal of Physical Oceanography*, *25*, 463–474.

Goddard Institute for Space Studies (2001), Earth's Insolation, URL `http://aom.giss.nasa.gov/solar.html`, viewed 14 September 2001.

Gordon, H. B. (1981), A flux formulation of the spectral atmospheric equations suitable for use in long-term climate modeling, *Monthly Weather Review*, *109*, 56–64.

Gordon H. B., pers. comm., CSIRO Marine and Atmospheric Research.

Gordon, H. B., and S. P. O'Farrell (1997), Transient climate change in the CSIRO coupled model with dynamic sea ice, *Monthly Weather Review*, *125*(5), 875–907.

Gordon, H. B., L. D. Rotstayn, J. L. McGregor, M. R. Dix, E. A. Kowalczyk, S. P. O'Farrell, L. J. Waterman, A. C. Hirst, S. G. Wilson, M. A. Collier, I. G. Watterson, and T. I. Elliott (2002), The CSIRO Mk3 climate system model, Technical Report 60, CSIRO Atmospheric Research.

Gregory, D., and P. R. Rowntree (1990), A mass flux convection scheme with representation of cloud ensemble characteristics and stability-dependent closure, *Monthly Weather Review*, *118*, 1483–1506.

Hewlett-Packard Company (2005), Compaq Math Libraries, URL `http://h18000.www1.hp.com/math/`, viewed 23 April 2005.

Hewlett-Packard Development Company (2005a), HP Fortran for Tru64 UNIX Alpha Systems, URL `http://h21007.www2.hp.com/dspp/tech/tech_TechDocumentDetailPage_IDX/1,1701,7129,00.html`, viewed 24 April 2005.

Hewlett-Packard Development Company (2005b), KAP Optimizers, URL `http://www.hp.com/techservers/software/kap.html`, viewed 18 May 2005.

Hirst, A. C., pers. comm, CSIRO Marine and Atmospheric Research.

Hirst, A. C., S. P. O'Farrell, and H. B. Gordon (2000), Comparison of a coupled ocean-atmosphere model with and without oceanic eddy-induced advection. Part I: Ocean spinup and control integration, *Journal of Climate*, *13*, 139–163.

Intel Corporation (2005), Intel Fortran Compiler 8.1 for Linux, URL `http://www.intel.com/software/products/compilers/flin/`, viewed 24 April 2005.

Johns, W. E., F. Yao, D. B. Olson, S. A. Josey, J. P. Grist, and D. A. Smeed (2003), Observations of seasonal exchange through the Straits of Hormuz and the inferred heat and freshwater budgets of the Persian Gulf, *Journal of Geophysical Research*, *108*(C12), 3391, doi:10.1029/2003JC001881.

Kanamitsu, M., W. Ebisuzaki, J. Woollen, S.-K. Yang, J. J. Hnilo, M. Fiorino, and G. L. Potter (2002), NCEP-DOE AMIP-II Reanalysis (R-2), *Bulletin of the American Meteorological Society*, *83*(11), 1631–1643.

Knudsen, M. (1900), Ein hydrographischer Lehrsatz, *Annalen der Hydrographie und Maritimen Meteorologie*, *28*, 316–320.

Kowalczyk, E. A., J. R. Garratt, and P. B. Krummel (1991), A soil-canopy scheme for use in a numerical model of the atmosphere - 1D stand-alone model, Technical Report 23, CSIRO Division of Atmospheric Research.

Kowalczyk, E. A., J. R. Garratt, and P. B. Krummel (1994), Implementation of a soil-canopy scheme into the CSIRO GCM - regional aspects of the model response, Technical Report 32, CSIRO Division of Atmospheric Research.

Lacis, A. A., and J. E. Hansen (1974), A parameterization for the absorption of solar radiation in the earth's atmosphere, *Journal of the Atmospheric Sciences*, *31*, 118–133.

Lide, D. R., editor, (1999), *CRC Handbook of Chemistry and Physics*, CRC Press, 80th edition.

McGregor, J. L. (1993), Economical determination of departure points for semi-Lagrangian models, *Monthly Weather Review*, *121*, 221–230.

Michalakes, J. G., M. McAtee, and J. Wegiel (2002), Software infrastructure for the Weather Research and Forecast Model, in *Proceedings of UGC 2002*, Austin, Texas.

MPI Forum (2005), Message Passing Interface (MPI) Forum Home Page, URL `http://www.mpi-forum.org/`, viewed 27 May 2005.

National Oceanographic Data Center (2002), World Ocean Atlas 1998, URL `http://www.nodc.noaa.gov/OC5/pr_woa.html`, viewed 13 September 2005.

NOAA-CIRES (2005), NOAA-CIRES Climate Diagnostics Center, URL `http://www.cdc.noaa.gov/`, viewed 15 October 2005.

O'Farrell, S. P. (1998), Investigation of the dynamic sea ice component of a coupled atmosphere-sea ice general circulation model, *Journal of Geophysical Research*, *103*(C8), 15,751–15,782.

Ooura, T. (2005), General Purpose FFT (Fast Fourier/Cosine/Sine Transform) Package, URL `http://momonga.t.u-tokyo.ac.jp/~ooura/fft.html`, viewed 2 October 2005.

OpenMP Architecture Review Board (2005), OpenMP, URL `http://www.openmp.org/drupal/`, viewed 23 April 2005.

Pacific Marine Environmental Laboratory (2005), Ferret, URL `http://ferret.wrc.noaa.gov/Ferret/`, viewed 15 October 2005.

Paleoclimate Modelling Intercomparison Project (2005), Paleoclimate Modelling Intercomparison Project Phase II, URL `http://www-lsce.cea.fr/pmip2/`, viewed 19 September 2005.

Phipps, S. J. (2006), *On Long-Term Climate Studies Using a Coupled General Circulation Model*, PhD thesis, University of Tasmania.

Research Systems Inc. (2005), IDL, URL `http://www.rsinc.com/idl/`, viewed 15 October 2005.

Robert, A. J. (1966), The integration of a low order spectral form of the primitive meteorological equations, *Journal of the Meteorological Society of Japan*, *44*(5), 237–244.

Rotstayn, L. D. (1997), A physically based scheme for the treatment of stratiform clouds and precipitation in large-scale models. I: Description and evaluation of the microphysical processes, *Quarterly Journal of the Royal Meteorological Society*, *123*, 1227–1282.

Rotstayn, L. D. (1998), A physically based scheme for the treatment of stratiform clouds and precipitation in large-scale models. II: Comparison of modelled and observed climatological fields, *Quarterly Journal of the Royal Meteorological Society*, *124*, 389–415.

Rotstayn, L. D. (2000), On the "tuning" of autoconversion parameterizations in climate models, *Journal of Geophysical Research*, *105*(D12), 15,495–15,507.

Rotstayn, L. D., pers. comm., CSIRO Marine and Atmospheric Research.

Rotstayn, L. D., and M. R. Dix (1992), Parallelization of a spectral general circulation model, *Supercomputer*, *47*, 33–42.

Sadler, H. E. (1982), Water flow into Foxe Basin through Fury and Hecla Strait, *Naturaliste Canadien*, *109*, 701–707.

Saucier, F. J., S. Senneville, S. Prinsenberg, F. Roy, G. Smith, P. Gachon, D. Caya, and R. Laprise (2004), Modelling the sea ice-ocean seasonal cycle in Hudson Bay, Foxe Basin and Hudson Strait, Canada, *Climate Dynamics*, *23*, 303–326.

Schwarzkopf, M. D., and S. B. Fels (1985), Improvements to the algorithm for computing CO_2 transmissivities and cooling rates, *Journal of Geophysical Research*, *90*(D6), 10,541–10,550.

Schwarzkopf, M. D., and S. B. Fels (1991), The simplified exchange method revisited: An accurate, rapid method for computation of infrared cooling rates and fluxes, *Journal of Geophysical Research*, *96*(D5), 9075–9096.

Semtner, Jr, A. J. (1976), A model for the thermodynamic growth of sea ice in numerical investigations of climate, *Journal of Physical Oceanography*, *6*, 379–389.

Standards Association of Australia (1983), *Australian Standard 1486-1983: Programming Language FORTRAN*, Standards Association of Australia.

Teuler, J.-M. (2005), JMFFT, URL `http://www.idris.fr/data/publications/JMFFT/`, viewed 2 October 2005.

Trenberth, K. E. (1997), The definition of El Niño, *Bulletin of the American Meteorological Society*, *78*(12), 2771–2777.

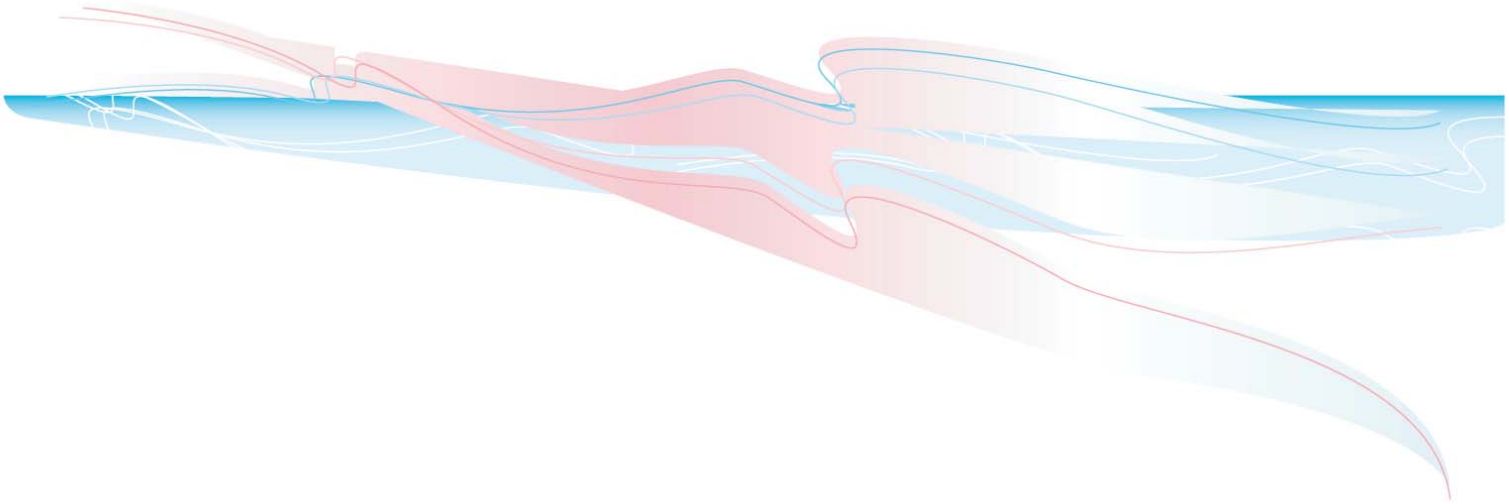
Tsimplis, M. N., and H. L. Bryden (2000), Estimation of the transports through the Strait of Gibraltar, *Deep-Sea Research Part I: Oceanographic Research Papers*, *47*(12), 2219–2242.

Unidata Program Center (2005), NetCDF, URL `http://my.unidata.ucar.edu/content/software/netcdf/index.html`, viewed 23 April 2005.

Wang, W.-C., X.-Z. Liang, M. P. Dudek, D. Pollard, and S. L. Thompson (1995), Atmospheric ozone as a climate gas, *Atmospheric Research*, *37*, 247–256.

Washington, W. M., and C. L. Parkinson (1986), *An Introduction to Three-Dimensional Climate Modeling*, Oxford University Press.

Wulff, F., L. Rahm, A. K. Hallin, and J. Sandberg (2001), A nutrient budget model of the Baltic Sea, in F. Wulff, L. Rahm, and P. Larsson, editors, *A systems analysis of the Baltic Sea*, volume 148 of *Ecological Studies*, chapter 13, pages 353–372, Springer Verlag.



Core Participants

Australian Antarctic Division
University of Tasmania
CSIRO Marine & Atmospheric Research
Australian Bureau of Meteorology

Supporting Participants

Alfred Wegener Institute for Polar and Marine Research
Australian Greenhouse Office
Australian National University
National Institute of Water and Atmospheric Research
Silicon Graphics International
Tasmanian Department of Economic Development

Address

ACE CRC
Private Bag 80
Hobart, Tasmania Australia 7001
P +61 3 6226 7888
F +61 3 6226 2440
E enquiries@acecrc.org.au
www.acecrc.org.au

Established and supported under the
Australian Government's Cooperative
Research Centres Programme