The Met.Office

Unified Model User Guide

Document Version:	2
Date of Issue:	December 22, 1998
Unified Model Version:	4.4

The Meteorological Office
London Road, Bracknell
Berkshire, RG12 2SZ
United Kingdom

Contents

List of Figures

List of Tables

Preface

This document is the User Guide for the Unified Model (UM). It gives basic information on how to use the model and includes references to relevant UM Documentation Papers (UMDPs) where more detailed information is required.

Chapter 1 provides the introduction. It gives an overview description of the UM and its development history. Chapter 2 contains information describing various aspects of the Unified Model. It is intended to help explain how the UM works and describe some of its features but does not explain how to use it. Chapter 3 contains information about the UM source code. This includes information on its structure, management and compilation. The main part of the User Guide is chapter 4. It contains instructions on how to use various parts of the UM and how to perform particular tasks.

The User Guide is intended for use both by users at the Met. Office and external users of the UM. It also contains additional information relating to the particular set up at the Met. Office. Where information is specific to one set of users it has been put into the appropriate appendix (appendix A for the Portable UM and appendix B for Met. Office specific information).

This document was written in LaTeX and is available in both HTML and PostScript forms. The HTML form contains links to allow easy navigation around the document and also out to external documents (such as UMDPs, source code listings, etc). External users will find that some of these links will not work since they are specific to the Met. Office or have not been configured.

If you have any contributions, corrections, or any other feedback concerning this document please contact the editor by emailing to `dmatthews@meto.gov.uk`.

Chapter 1

Introduction

1.1 The Unified Model

Author: Anette Van der Wal, Last Updated: 1 Jun 98

1.1.1 Overview

The Unified Model is the name given to the suite of atmospheric and oceanic numerical modelling software developed and used at the Met Office. The formulation of the model supports global and regional domains and is applicable to a wide range of temporal and spatial scales that allow it to be used for both numerical weather prediction and climate modelling as well as a variety of related research activities. The Unified Model was introduced into operational service in 1991. Since then, both its formulation and capabilities have been substantially enhanced.

The modelling system is designed so that it can be run in atmosphere-only, ocean-only or in coupled mode. A number of sub-models are also available which may be used in place of or in addition to the main models. These include a simplified model of the ocean, known as the slab model, and models to simulate the thermodynamic and dynamic motion of sea ice. A typical run of the system consists of an optional period of data assimilation followed by a prediction phase. Forecasts from a few hours to a few days ahead are required for numerical weather prediction while for climate modelling the prediction phase may be for tens, hundreds or even thousands of years. The choice of horizontal and vertical resolution is arbitrary and may be varied by the user. In practice, the resolution is constrained by the available computing power and a number of standard resolutions tend to be used (see table 1.1). A separate reconfiguration module allows model data to be optionally interpolated to a new grid or area prior to the start of a run. This applies to the atmosphere only.

A range of ancillary data may be incorporated into the model as a run progresses. The options range from lateral boundary forcing information for regional configurations to the periodic updating of fields such as ozone concentration to allow for the effects of seasonal variations, and surface forcing for ocean-only configurations.

Prognostic, diagnostic and derived fields of data may be output on a time-step by time-step basis for inspection and further analysis, or as forcing information for sub-models such as a sea-state prediction model. Time-meaned or accumulated quantities may also be specified.

The wide range of options available within the Unified Model system means that a large number of parameters need to be defined in order to specify and control a particular model configuration. A user-friendly X-Windows based user interface has been developed to make this task much easier. By use of interactive windows, a user may control parameters such as model resolution, the date and time of the initial data, the length of forecast, the choice of output diagnostics and so on. The user may also control the scientific content of the model by selecting from a library of alternative representations of the physical processes.

Application	Status	Version	Resolution (degrees)	Grid (E-W x N-S)	Levels	
					Atmos	**Ocean**
Climate Modelling	Production	HadCM2	2.50 x 3.75	96 x 73	19	20
Climate Modelling	Production	HadCM3 (A)	2.50 x 3.75	96 x 73	19	N/A
		HadCM3 (O)	1.25 x 1.25	288 x 144	N/A	20
Climate Modelling	Research	HadCM4	As HadCM3	As HadCM3	38	20
Middle Atmosphere	Production	N/A	2.50 x 3.75	96 x 73	42	N/A
Middle Atmosphere	Research	HadAM3	2.50 x 3.75	96 x 73	49	N/A
Weather Forecasting	Production	N/A	0.83 x 0.56	432 x 325	30	N/A
Weather Forecasting	Research (As production, plus. . .)	N/A	1.25 x 0.83	288 x 217	19	N/A

Table 1.1: Global configurations of the Unified Model used for production and research at the Met Office

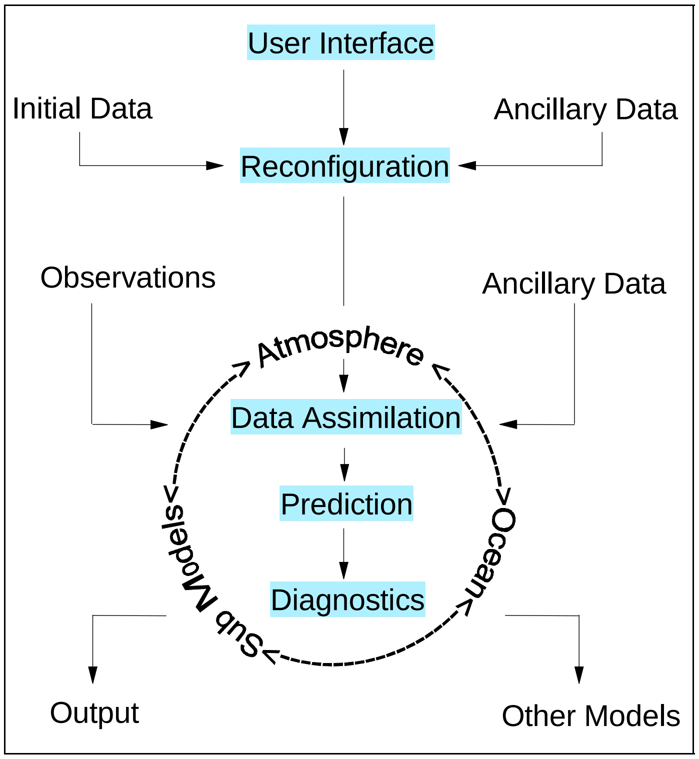


Figure 1.1: The main components of the Unified Model system

1.1.2 Formulation

The main atmospheric and oceanic components of the model are described below. In addition to these, a number of internal sub-models may be linked into the system. These include models of sea-ice and an ocean “slab” model. Development of further sub-model options is in progress and will include atmospheric chemistry and a model of the sea-state.

Atmospheric Prediction

The atmospheric prediction component uses a set of equations that describe the time-evolution of the atmosphere. The equations are solved for the motion of a fluid on a rotating, almost-spherical planet. The main variables are the zonal and meridional components of the horizontal wind, potential temperature and specific humidity. To solve the system of equations, a number of approximations or assumptions have to be made. In common with most other atmospheric models the Unified Model's atmospheric prediction scheme is based on the so-called hydrostatic primitive equations. This means that the model atmosphere is always in hydrostatic balance which means that there are no vertical accelerations. Most other atmospheric models make the shallow-atmosphere approximation which derives its name from the fact that the depth of the atmosphere is much smaller than the radius of the Earth. The Unified Model does not make this assumption. This makes the equations a little more complicated to solve by adding in extra terms which can be important when planetary-scale motions are considered.

The equations can be solved in a variety of different ways. The Unified Model uses a grid-point scheme on a regular latitude-longitude grid in the horizontal and a hybrid vertical grid, which is terrain-following near the surface but evolving to constant pressure surfaces higher up. An efficient split-explicit scheme is used to solve the equations which is designed to conserve mass, mass-weighted potential temperature and moisture and angular momentum. These are important to ensure the integrity of the solutions for long integrations required for climate-change experiments.

The physical processes represented include:

- Atmospheric radiation allowing for the effects of clouds, water vapour, ozone, carbon dioxide and a number of trace gases.
- Land surface processes including a multi-layer soil temperature and moisture prediction scheme.
- A treatment of the form drag due to the sub-grid scale variations in orography.
- Vertical turbulent transport within the boundary layer.
- Large-scale precipitation determined from the water or ice content of a cloud.
- The effects of convection through a scheme based on the initial buoyancy flux of a parcel of air. It includes entrainment, detrainment and the evaporation of falling precipitation. An explicit treatment of downdraughts is included.
- The effects of the drag caused by vertically propagating gravity waves is modelled using sub-grid scale orographic variance and known absorption properties of gravity waves.

Lateral boundary values may be output from larger scale integrations to support the running of regional configurations of the model. Regional configurations of the model are typically used to produce forecasts at much higher spatial resolutions than could be obtained within the available computing resources using global configurations of the model. A number of regional domains may be nested within each other. Regional configurations of the model use a rotated latitude-longitude grid which ensures a quasi-uniform grid length over the whole integration domain.

Oceanic Prediction

Ocean processes are represented by an ocean model that is based on the widely-used Bryan-Cox formulation. It uses an Arakawa staggered 'B' grid in the horizontal with variable spaced levels in the vertical. The main variables used by the ocean model are: currents, potential temperature and salinity, with forcing provided by surface heat, freshwater and momentum fluxes and wind mixing. In addition to the dynamical processes, the model includes representations of the following physical processes:

- Mixing in the upper ocean, and the interior.

- Surface fluxes of heat, momentum and freshwater, some of which can penetrate below the surface.
- Diffusion along isopycnal surfaces.
- Mesoscale eddy parametrisation.

Ocean properties evolve over many centuries, so the model includes modifications to the equations of motion that allow it to run for long periods to attain equilibrium solutions.

Biological and chemical processes are important on the longer timescales used in climate modelling. The ocean model contains representations of biological life cycles and their effects on the carbon budget of the ocean. Currently, this code is at the developmental stage.

Sea ice has a major impact on the climate of both the atmosphere and ocean. Within the ocean model there are several options for simulating sea ice, ranging from a scheme that assumes the ice to be stationary through to one in which the ice moves in response to winds, currents and the internal stresses of the ice.

Data Assimilation

Atmospheric and oceanic data assimilation is performed using an analysis correction scheme. This is an iterative analysis, with each variable being analyzed in turn. Appropriate constraints are included so as to improve balance and speed of convergence. In the atmosphere these include hydrostatic temperature increments derived from the analyzed surface pressure increments; geostrophic wind increments derived from the analyzed mass increments; and surface pressure and temperature increments derived from the analyzed wind increments using a linear balance relationship. Each iteration is interleaved with a timestep of the full forecast model to nudge the forecast model solution towards the observations. Observations are introduced into this continuous assimilation over a period around their validity time, with a peak weighting at their validity time. The extent of horizontal data spreading is determined by a continuous background error correlation function which has a time dependence such that progressively smaller scales are fitted.

1.1.3 Applications

The model is used by the Met Office for operational numerical weather prediction, ocean analysis and prediction, and (at the Hadley Centre for Climate Prediction and Research) to simulate and predict the Earth's climate. The model is also used by a number of meteorological centres and research organisations worldwide.

Numerical Weather Prediction

The Unified Model is used operationally at the Met Office to provide numerical forecasts of the atmospheric state for periods of a few hours to several days ahead. Both global and regional configurations are used, with the larger scale model providing lateral boundary values for higher resolution regional models. Several runs are made throughout each day against a fixed schedule and tight deadlines.

A number of major improvements to the scientific formulation of the Unified Model have been made during its lifetime (see section 1.2. These have led to significant gains in the quality of operational forecasts (see figure 1.2).

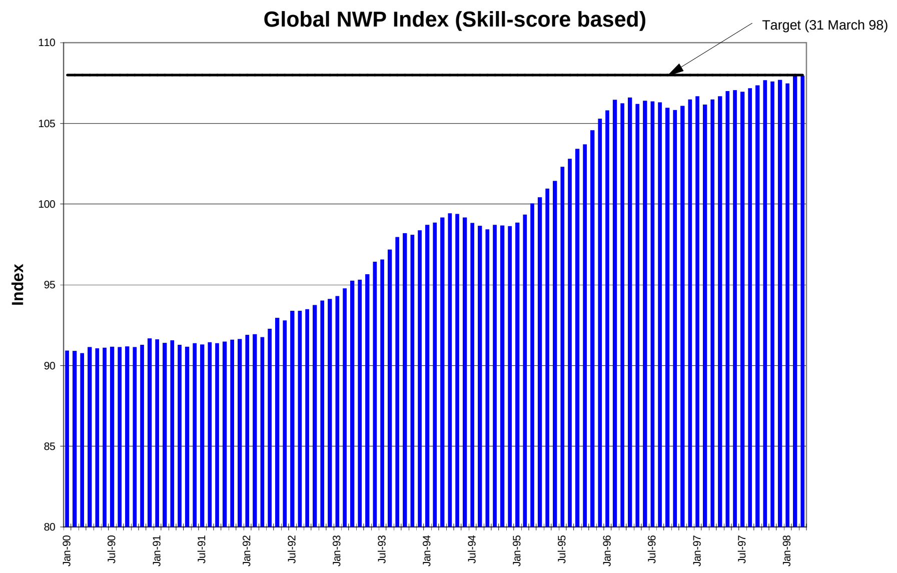


Figure 1.2: The skill of Met Office operational global forecasts as measured by the NWP index. The index increases as the skill of the forecasts improves. It is calculated by normalising the root mean square (rms) error of the model against persistence and forming a weighted average of key forecast fields. The Unified Model was introduced into operational service in June 1991.

Limited-area, regional configurations of the Unified Model are used to provide detailed guidance of the weather in the region of the United Kingdom (see figure 1.3). The current operational resolution of the UK Mesoscale Model is ˜13km, which replaced a 17km configuration with a smaller domain which was in use up to June 1998.

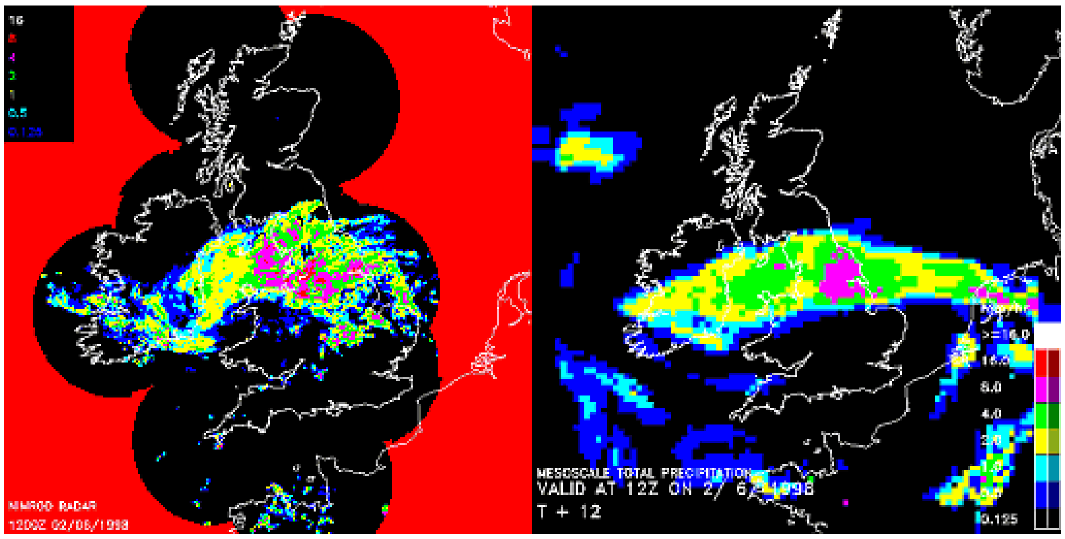


Figure 1.3: A 12 hour forecast of precipitation intensity obtained using a high resolution (17Km) regional configuration of the Unified Model (right). The observed precipitation as measured by the UK radar network is also shown (left).

Oceanographic Forecasting

FOAM, the Forecasting Ocean Atmosphere Model, produces analyses and short-range forecasts (up to 5 days ahead) of the ocean's temperature, salinity and current structure, and of sea-ice concentration and thickness. It is based on the oceanic component of the Unified Model and is driven by surface fluxes of heat, fresh water, and momentum that are calculated from operational runs of the atmospheric component of the Unified Model.

Because FOAM is intended for short-range integrations, the oceanic initial state is very important. The success of FOAM is therefore very dependent on the quality of the Unified Model's data assimilation scheme.

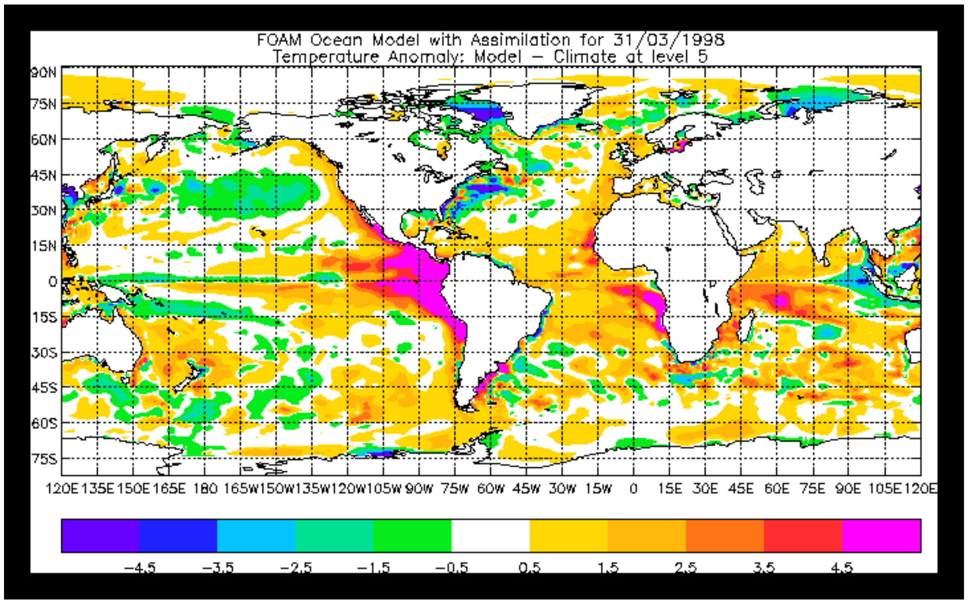

FOAM Ocean Model with Assimilation for 31/03/1998
Temperature Anomaly: Model – Climate at level 5
90N
75N
60N
45N
30N
15N
0
15S
30S
45S
60S
75S
120E 135E 150E 165E 180 165W 150W 135W 120W 105W 90W 75W 60W 45W 30W 15W 0 15E 30E 45E 60E 75E 90E 105E 120E
−4.5
−3.5
−2.5
−1.5
−0.5
0.5
1.5
2.5
3.5
4.5

Figure 1.4: Temperature anomaly at level 5 as analysed by FOAM

Climate Prediction

The prospect of climate change due to man's activities is a high profile scientific and political issue. The threat of global warming due to the steady increase in atmospheric concentrations of greenhouse gases has led to renewed efforts to understand the factors influencing the Earth's climate and to predict the extent and impact of climate change over the next century.

The Hadley Centre uses the Unified Model both to simulate the present climate in order to understand its natural variability and to predict global and regional climate change up to the end of the 21st century. Climate prediction requires models of the atmosphere, ocean, land surface and sea ice to be run in coupled mode so that the interactions between them are represented. It also requires details of changing anthropogenic and natural forcings of the climate system to be specified including, for example, greenhouse gases, sulphate aerosols, ozone, volcanic eruptions and variations in solar output.

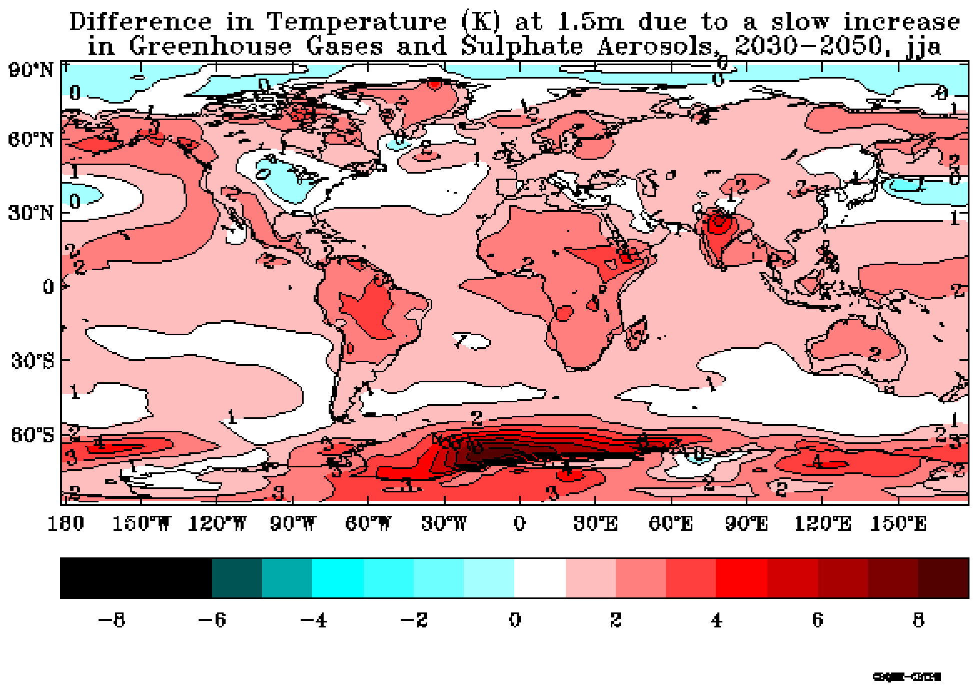

Difference in Temperature (K) at 1.5m due to a slow increase
in Greenhouse Gases and Sulphate Aerosols, 2030−2050, jja
90°N
60°N
30°N
0
30°S
60°S
180
150°W
120°W
90°W
60°W
30°W
0
30°E
60°E
90°E
120°E
150°E
−8
−6
−4
−2
0
2
4
6
8

Figure 1.5: A prediction of global temperature change in summer for the middle of next century, using the HadCM2 coupled model

Middle Atmosphere Research

Daily analyses of the stratosphere are carried out at the Met Office using a version of the Unified Model's data assimilation scheme. The analyses are used in a variety of applications which are aimed at a better understanding of the dynamics and chemistry of the stratosphere. In particular, the analyses have proved invaluable in the interpretation of measurements from the Upper Atmosphere Research Satellite. In this configuration, the model goes up to the lower mesosphere (˜60km) and has 42 levels.

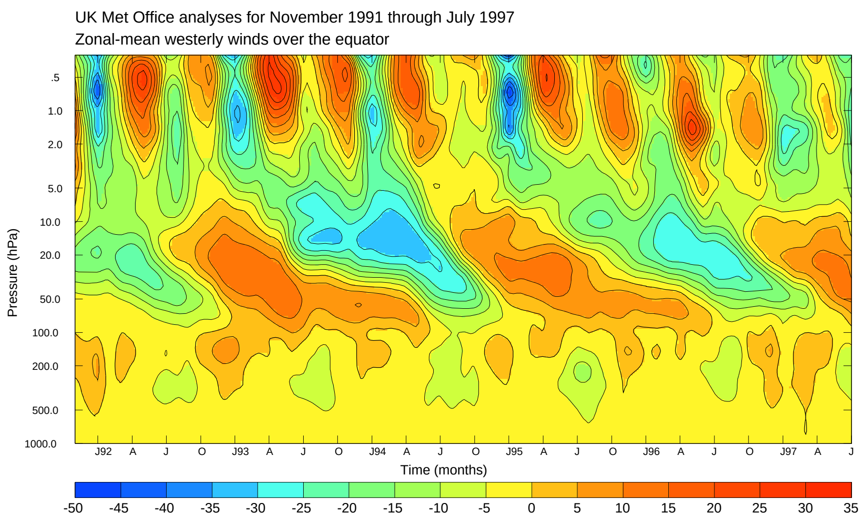


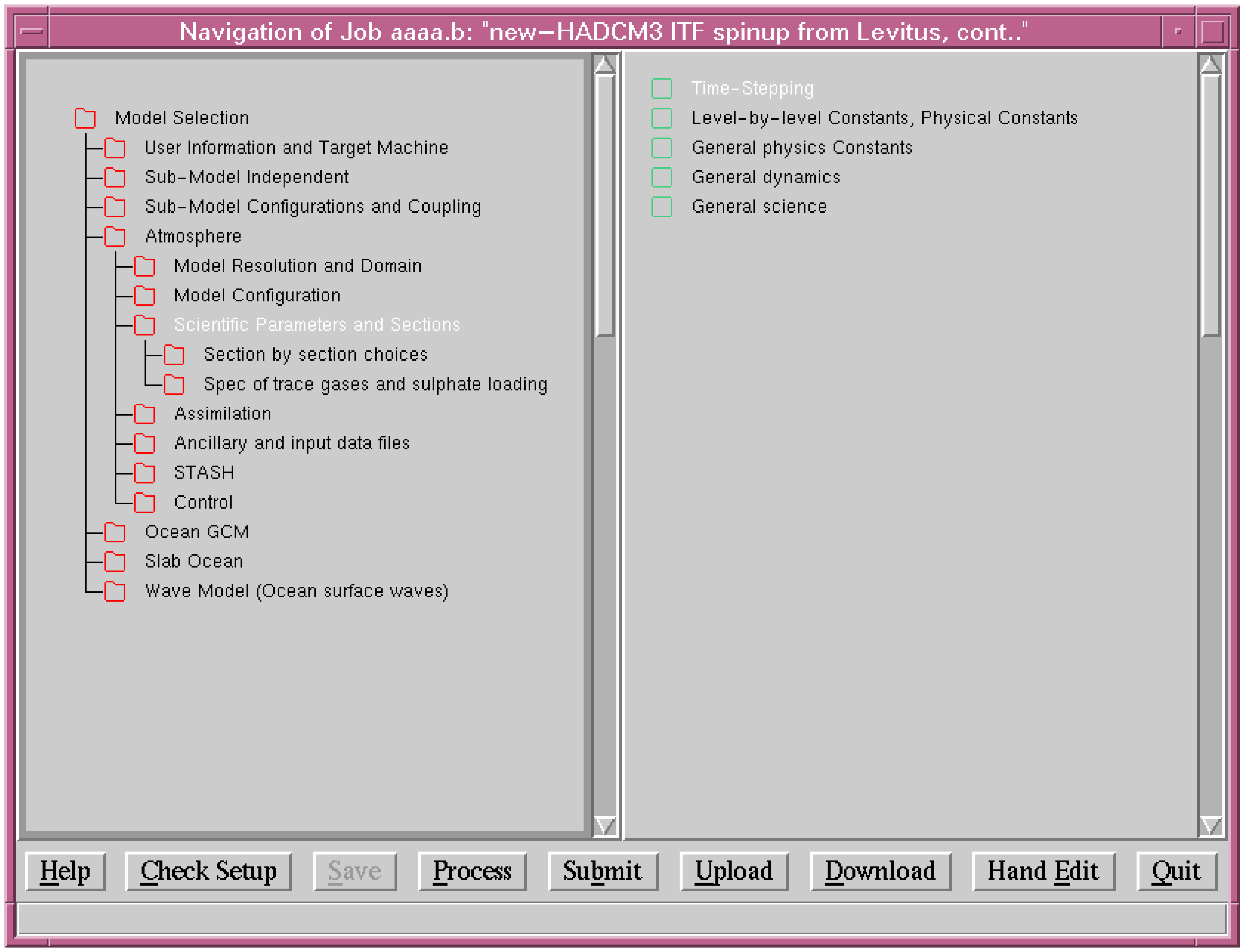
Figure 1.6: Time series of winds over the equator, analysed using the stratospheric configuration of the Unified Model. The winds show a clear Quasi-Biennial Oscillation in the stratosphere and a Semi-Annual Oscillation near the stratopause.

A 49-level troposphere/stratosphere model is run in climate mode for time periods of up to 60 years in length.

1.1.4 Features and Capabilities

User Interface

The Unified Model is a very flexible system allowing a vast number of choices. To help the user make the most of this flexibility, a graphical user interface, known as the UMUI (Unified Model User Interface), has been developed. The UMUI is X-Windows based and runs in the familiar environment of the user's own local workstation (see figure 1.7). It is designed to give access to the full functionality of the Unified Model whilst speeding up the learning process.



Navigation of Job aaaa.b: "new–HADCM3 ITF spinup from Levitus, cont.."

- Model Selection
  - User Information and Target Machine
  - Sub-Model Independent
  - Sub-Model Configurations and Coupling
  - Atmosphere
    - Model Resolution and Domain
    - Model Configuration
    - Scientific Parameters and Sections
      - Section by section choices
      - Spec of trace gases and sulphate loading
    - Assimilation
    - Ancillary and input data files
    - STASH
    - Control
  - Ocean GCM
  - Slab Ocean
  - Wave Model (Ocean surface waves)

- Time-Stepping
- Level-by-level Constants, Physical Constants
- General physics Constants
- General dynamics
- General science

Figure 1.7: The main navigation window of the UMUI. Nodes in the tree (left) may be expanded and collapsed. In this example, the atmospheric model node is expanded. Selected nodes are opened up to display windows (right).

Using a central database that works in a distributed environment, the UMUI provides an efficient method of defining model runs and managing those definitions. A run definition is known as a job. Users can search for and copy jobs belonging to colleagues, and definitions can even be sent by email from one site to another. This allows colleagues and collaborators at remote sites to increase their efficiency by pooling their effort and knowledge. The job management system also provides help for users wanting to upgrade from one version of the Unified Model to another.

In addition to the job management system, the UMUI contains a comprehensive job definition system. This is an intelligent editor that allows modifications to be made to existing jobs. It is made up of a large number of windows, each with visual cues to ensure that users can quickly see when options have been disabled due to other responses. To further aid the user all windows have a help button that displays information relating to the options available on the window.

Once a job has been defined, the responses are translated into control files that can be read by the Unified Model. On request, the UMUI will then copy these to the target machine and submit the job. The control files are text based and so a user can easily over-ride the UMUI by editing the control files. This facilitates the prototyping of new features.

Reconfiguration

The reconfiguration module processes the model's initial data. For the atmosphere mode, it allows the horizontal or vertical resolution of the data to be changed or the creation of a limited area region (see figure 1.8). Ancillary data may also be incorporated into the initial data. Fields such as the specification of the height of orography or the land-sea mask, for example, may be overwritten by more appropriate values if the horizontal resolution has been changed. Other facilities available within the reconfiguration module include the ability to incorporate ensemble perturbations, tracer fields and the transplanting of prognostic data.

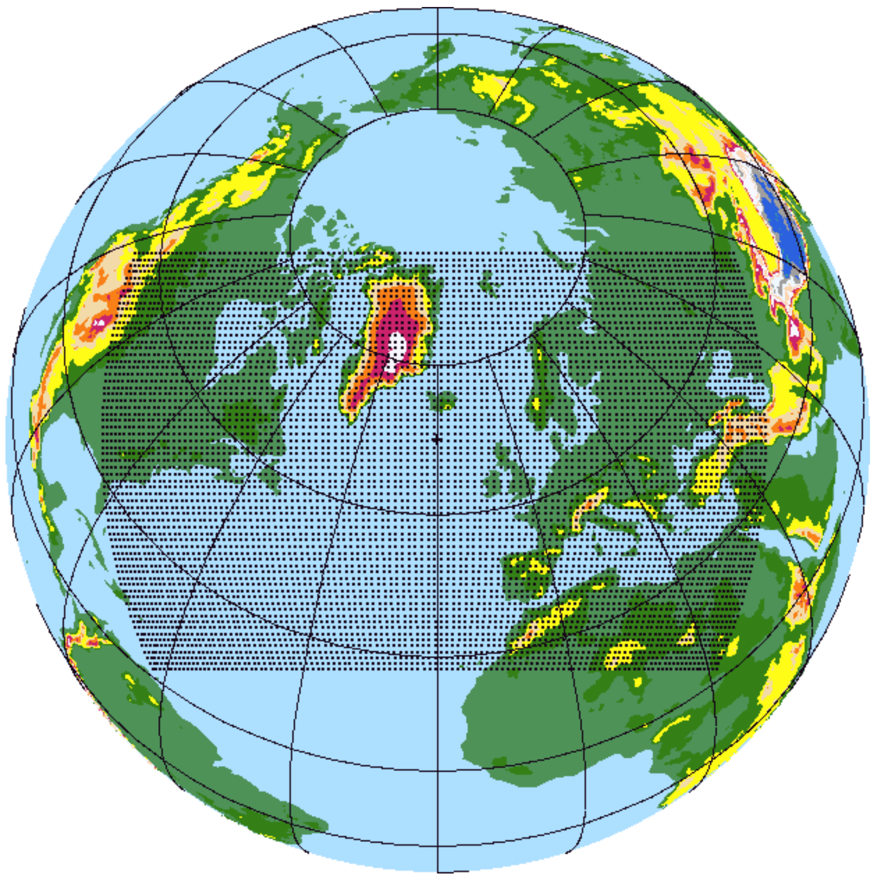


Figure 1.8: A regional integration domain and grid. Initial data may be interpolated onto such domains via the reconfiguration module.

Tracers

The atmospheric component supports the advection of passive tracers using a shape preserving advection scheme. The methods are designed to ensure that sharp gradients are preserved while keeping all concentrations greater than or equal to zero. A similar advection scheme is under development for the ocean model and will be available at UM vn4.5. This facility may be used to examine dispersion from point sources or as the basis for more involved chemical modelling by the addition of extra code. For example, atmospheric forecast configurations of the model include an aerosol variable which is advected using the tracer scheme, and may optionally have sources and sinks applied. This is used to estimate visibility, and may be adjusted by the assimilation of visibility data.

Ancillary Data

New ancillary data may be incorporated as an integration proceeds to allow for the effects of temporally varying quantities such as surface vegetation, snow cover or wind-stress. The choice of whether to include ancillary data will depend on the application. For example, the system supports daily updating of sea-surface temperatures based on climatological or analysed data - but this facility is not needed in ocean-only or coupled ocean-atmosphere runs of the model. In fact options exist to allow most boundary forcing variables to be predicted. However, it is often necessary to constrain such fields to observed or climatological values for reasons of computational economy or scientific expedience.

OASIS Coupler

The aim of the OASIS (Ocean Atmosphere Sea Ice Soil) coupler, developed at CERFACS, is to provide a tool for coupling independent GCMs of atmosphere and ocean as well as other climate modules. It is possible for the models to run on different platforms.

Diagnostic Processing

Diagnostic data output from the model may be defined and output on a timestep by timestep basis. Horizontal fields, sub-areas or timeseries may be be requested, along with the accumulation, time meaning or spatial meaning of fields. The output may also be split across a number of files in order to facilitate the separate post-processing of classes of diagnostics.

Output

The results of an integration are stored in a format compatible with other file types used within the Unified Model system. Several utilities exist which will convert them into a format suitable for further processing. These formats include GRIB - the WMO standard for storing fields of meteorological data in a machine independent format. Sadly, this is currently unavailable in the Portable UM since it requires the use of third party software. Utilities to print out and compare the contents of Unified Model files are available. These, coupled with the bit reproducibility facility (see below), offer a powerful development and debugging aid.

Computer Requirements

The code was originally written in FORTRAN 77, but now uses a number of the extensions to the language available in Fortran 90. A small number of low-level service routines are written in ANSI C to facilitate portability. At the control level, the model runs under the Korn shell of the Unix operating system. The model will normally require a powerful supercomputer for time critical production work. It is written to be efficient on distributed memory massively parallel processing (MPP) computers such as the Cray T3E, but will also run on a single node of a parallel machine or a variety of non-MPP platforms. The code is bit reproducible independently of the number of processing elements used. The code may also be run on workstation systems using the IEEE standard for representing floating point numbers. Workstations are particularly suited to development work or research activities requiring the use of low resolution configurations of the model.

1.1.5 The Future

Improvements to the scientific formulation and software capabilities of the Unified Model are always being sought and considered. A number of important developments are planned for the next few years, to include:

- Major changes to the atmospheric prediction scheme which involve less approximation and improved methods of solution are under development. The non-hydrostatic form of the governing equations will be adopted, enabling the model to be used for very high resolution forecasting.
- The Unified Model system will move towards the use of variational methods of data assimilation. In this, a model state is constructed that is the best statistical fit to the observed data. A system like this requires a method of blending the observations with the background field provided by a simple linearized model which is integrated forwards and backwards in time to fit the observations to an evolving model state.

1.2 The History of the Unified Model

Author: Rick Rawlins, Last Updated: 02 Feb 98

1.2.1 Background to the Unified Model

By the end of the 1980's the Met.Office had developed separate numerical prediction models to sustain climate research and operational forecast capabilities. The climate model consisted of 11 vertical layers with a global domain at 250km horizontal resolution and incorporated a relatively sophisticated representation of physical processes. The operational model had 15 vertical layers and ran in two versions: a 'coarse mesh' global model at 150km resolution and a 'fine mesh' limited area model covering Europe and the N.Atlantic at 75km resolution. These models had simpler calculations of physical processes and employed a different dynamical integration scheme to that of the climate model. In addition a mesoscale model with 15km resolution provided some operational products for the UK area, and comprised a completely distinct formulation for both physics and dynamics. There were also new requirements arising from the need to include more complex interactions into long climate integrations, in particular the introduction of an ocean model, to be coupled explicitly with the atmosphere model.

Each model had its own control, file and output structure, as well as separate scientific formulations, with separate teams responsible for development. The models ran on a CDC CYBER 205 which was due to be replaced by an ETA systems platform during 1989. Significant effort was expended in adapting the code of each model to run on the new platform, but this work was suspended when the manufacturing company was wound up and a new vendor had to be found. It was decided to promote a longer term project to achieve a unification of model systems to fill the gap, and reduce the duplication of effort required to implement separate models on new hardware. The lifetime of the CYBER was extended and development of the CYBER models frozen; a Cray Systems YMP with 8 vector processors was procured with a Unix based operating system and the machine arrived in January 1990. A formal Unified Model project was started in July 1989 and continued until October 1991 when it was judged that the main components were ready and the Unified Model was born.

A number of key strategic decisions has shaped the structure and development of the UM:

- The system would share a common control and file structure for all types of models, so that individual models running in separate environments would be replaced by different model configurations of the UM.
- Model set-up would be achieved via a graphical user interface, divorcing the detail of control file information from decisions made by the user. Thus presentation of model options to the user is simplified, the process of model submission is streamlined and the occurrence of inconsistencies or errors during set-up is reduced.
- The UM system would be upgraded periodically through change control management techniques, which is an essential safeguard for code changes made by developers that may affect other model configurations. Each upgrade constitutes a UM version.
- Model software would be coded in Fortran, with a minimum of machine specific or low-level language additions. This is required to loosen the dependency on a particular hardware platform, and allow more rapid development of new scientific routines by researchers. This decision was later extended to include the objective of providing explicit portability. Hence model configurations which were developed on supercomputer platforms should be readily ported to other systems such as workstations, with a minimum of changes.
- Models would depend on a common output diagnostics system, which was capable of intercepting transient fields, or calculations based on such fields, as well as primary fields available through periodic model dumps.

- Separate choices of scientific schemes would be readily available from the user interface, and different physics schemes would be 'plug compatible', ie it should be easy to import or export schemes to other organisations' models with a minimum of effort.

1.2.2 Version upgrades and hardware platforms

Table 1.2 lists release dates for each UM version and the date at which the operational forecast model was upgraded to the new release. Some indication of the more major changes to the hardware platform is given.

UM Version	**Release Date**	**Operational Release**	**Hardware Platform: Operating System**	**Compiler Release**
2.0	16.05.91	05.06.91	Cray YMP-8: Unicos5.1	4.0.2
2.1	27.06.91	16.07.91		
2.2	15.08.91	17.09.91		
2.3	23.10.91	12.11.91		
2.4	07.01.92	11.02.91	Unicos6.0	
2.5	26.03.92	12.05.92		
2.6	04.06.92	11.08.92		5.0.2
2.7	01.09.92	27.10.92		5.0.3
2.8	22.12.92			
3.0	15.01.93	23.02.93		
3.1	30.03.93	25.05.93	Unicos7.0	
3.2	06.09.93	19.10.93		
3.3	03.06.94	12.07.94	Cray C90:	6.0.3.2
3.4	24.01.95	07.03.95		
3.5	03.08.95			
4.0	12.01.96	19.03.96		
4.1	20.08.96		Unicos8.0	
4.2	15.01.97		Cray T3E: mk1.4.1	
4.3	09.06.97	29.01.98	mk1.6.1	2.0.3.4
4.4	22.12.97	15.04.98	mk2.0.0	

Table 1.2: Unified Model Versions

The following sections illustrate the more significant changes that have contributed to version upgrades. Brief descriptions of system enhancements are given, with some indication of new scientific features, concentrating on the atmosphere model. In reality each UM version consists of a hundred or more individual changes, ranging from minor corrections to major improvements of infra-structure facilities, plus work required to take advantage of successive enhancements to the available computing resources.

1.2.3 Early versions 2.0-2.6

Rapid development continued to consolidate the performance of the UM. A significant number of error corrections arose from the relative immaturity of the code. Changes included:

- Introduction of C I/O routines to replace Cray specific unblocked data files and replacement of reference to file names by assign statements to being passed by environment variables.

- Source code modification of code by users for compilation was changed from “quick” to “normal” modes of Cray proprietary update package, to ensure that a change to a COMDECK was propagated to all affected routines (see section 3.1).
- Vertical interpolation of boundary files was enabled, allowing generation of boundary conditions for a model on a different vertical grid.
- A facility to enable timeseries to be output from the diagnostic system was introduced.
- The conversion of potential temperature to temperature ($\theta \rightarrow T$) was modified to account for the dynamical formulation of the hydrostatic relation; definitions of standard vertical hybrid co-ordinates changed.
- Dynamics calculations at the top model level could now be switched to run with a halved timestep, preventing model instabilities causing model failures without the cost of running the entire model with a reduced timestep.

1.2.4 Version 2.7

Changes already available through the UM system were co-ordinated to produce a UM mesoscale configuration with 30 vertical levels. This replaced the earlier mesoscale model and went operational in December 1992.

- Several enhancements of physics were consolidated in the system, including the components contributing to the first climate version (1CV, now labelled HADAM1):
  1. Convection with an explicit downdraught;
  2. Rapidly mixed boundary layer code.
- Introduction of a separate version of the longwave radiation scheme based on transmissivity values derived at ECMWF.

1.2.5 Version 2.8

- Upgrade of source code management tool from Cray update to Cray nupdate.
- A new packing code was introduced into look-up headers of data fields to cater for new combinations of packing and data compression. This required an explicit utility to convert model dumps at 2.7 and earlier to the new format at 2.8.
- (Outstanding 1CV science change): grouping of model levels into 3 explicit cloud layers within the shortwave radiation scheme.
- Time-smoothing during the adjustment timestep within the dynamics was introduced to alleviate the problem of noisy fields at upper levels in the operational global model.

1.2.6 Version 3.0

The active code of Version 3.0 was identical in content to that at 2.8, but line numbers of code were re-sequenced and some old comments were removed. DECK names were changed systematically to reflect the concept of multiple version-releases of a code section (see section 4.7).

1.2.7 Version 3.1

- A new option was provided to allow more than one boundary file to be produced by the model, so that multiple limited area models could be driven from the same generating run.
- Functionality of the reconfiguration program was extended to include initialisation of model analyses with ECMWF perturbations, simplifying running of this category of ensemble forecasts. An option to transplant fields from a different model dump into a given domain within the model analysis was also developed.

1.2.8 Version 3.2

The first version of the portable model was made to work at 3.2 on HP workstations, but with a large number of local modifications, and relied on fixed experiment libraries generated by the user interface on the IBM mainframe computer.

- Dynamic allocation of primary fields was introduced and required large scale changes to the top level control structure of the model. Sizes of arrays depending on the model configuration were now passed at run-time, removing the necessity to re-compile when changes in diagnostic output were required.
- The value of the real missing data indicator was changed from -32768.0 to -2**30 to reduce the possibility of valid data being assigned as missing.

1.2.9 Version 3.3

- A facility for users to introduce their own prognostic variables without requiring a change to the user interface was implemented, giving code developers freedom to make and test a greater range of modifications locally.
- GRIB format for output fields was provided as an option. The file structure was of hybrid form, with lookup headers retained, but data packed as GRIB.
- A halving of the dynamics timestep could be initiated on an automatic basis, depending on thresholds of wind speed or divergence in the top level being exceeded.
- Calling convection routines less frequently than every timestep was enabled, providing a method of reducing cpu costs of the operational mesoscale model without significant degradation in performance.
- The provision of source terms for aerosols, as part of a tracer advection technique for dealing with passive variables, provided essential functionality for both climate studies and later mesoscale visibility calculations.

1.2.10 Version 3.4

- Unix scripts in the UM system were brought within code management by nupdate as part of change control for new versions. This replaced having single frozen copies and simplified the action of modifying scripts by different people.
- Macrotasking techniques were introduced into dynamics, convection and radiation schemes to facilitate more efficient use of the multi-tasked C90 environment.

- New options for atmosphere science routines, constituting the main components of the second climate version (2CV, now labelled HADAM2a):

  1. A more complex treatment of orographic roughness within the boundary layer code, including the representation of silhouette orographic roughness;
  2. A new gravity wave drag scheme incorporating anisotropic directional orographic variances and 'hydraulic jump' terms;
  3. A new version of the convection routine with excess parcel buoyancy generated from the surface level fluctuations of temperature and humidity;
  4. A new large scale precipitation scheme with all snow falling out in a layer in the same manner as for rain;
  5. Inclusion of primitive cloud microphysics within the shortwave radiation scheme, allowing a variation in cloud effective radius;
  6. An option to re-set negative humidity values locally rather than globally.

- A new multi-layer soil hydrology scheme.
- Inclusion of trace gases in a new version of the longwave radiation.
- Assimilation of humidity from MOPS (Moisture Observation Pre-processing System) was introduced, providing a method of feeding moisture information back into the hydrology cycle of the operational mesoscale model.
- A dynamical sea-ice model was included in the Slab model (see section 4.11).
- An ice section was added to the ocean model and assimilation within the ocean model enabled.
- This version was released to a number of external users, now with the capability of running on a Dec Alpha workstation.

1.2.11 Version 3.5/4.0

3.5 was an interim version required to stage the large changes planned for 4.0, and only had significance for the core development and testing team. This version spanned the transfer of central UM system facilities, including the user interface, from the IBM mainframe to a more portable Unix environment. At the Met. Office this consists of a combination of HP workstation local area networks with the Cray C90. Modifications to code for 3.5 relative to the previous version were located on the mainframe. The repository for modifications to code against 3.5 and later versions is the Cray C90. Development of a new graphical user interface, the UMUI, was partially completed at 3.5, and fully implemented for 4.0. Submission of model jobs to the Cray C90, and inspection of subsequent output listings, was made independent of the IBM mainframe.

- An X-windows user interface (UMUI) based on public domain TCL/TK building tools was produced, a completely new program and structure from the mainframe interface designed at the beginning of the UM. The UMUI now resides on HP workstation networks, and is capable of being ported to other machines, such as the Cray C90 itself. This provided a more flexible platform for initiating experiment set-up within the Met. Office and created the capability to package up the entire UM system for implementation by external users.
- The first stage of work to facilitate the introduction and coupling of new sub-models into the system was completed. This included the transfer of primary addressing calculations, which had been performed pre-3.5 by the user interface on the mainframe, into the model initialisation. In addition a new paradigm of sub-models and internal models was introduced for dealing with different combinations of sub-model types.

- Provision of the capability of running with a long physics timestep enabled significant reductions in cpu costs to be achieved by reducing the number of physics calculations while maintaining model numerical stability.
- Adoption of the UMUI running on HP workstations led to a loss of bit reproducibility for models set up at 3.4 and 4.0 due to the change in precision of floating point calculations now performed on the 64 bit Cray machine rather than the 32 bit IBM; and also due to the difference in number representation from IBM to HP. Other minor changes were made that also affected bit reproducibility:
    1. Correction in the value of π;
    2. Corrections to filtering within the dynamics.
- Addition of science routines to constitute third climate physics (3CV, now labelled HADAM2b):
    1. Precipitation changes: Improved mixed phase and evaporation calculations; explicit flux divergence and flux equilibrium schemes for ice fallout dependent on timestep;
    2. Re-appraisal of saturated specific humidity formulation.
- A new radiation scheme (Slingo-Edwards) was introduced, covering both longwave and shortwave regions, providing a greater flexibility of choice of radiative spectral intervals.
- A new hydrology routine incorporating the Penman-Monteith scheme was added, with consistent changes made to a new version of the boundary layer routine.
- Optional removal of horizontal diffusion for sloping model levels was included.
- Latent heat nudging in the assimilation was provided for investigations with the operational mesoscale model.
- Copyright notices were added to each item of source code.

1.2.12 Version 4.1

Version 4.1 rationalised a number of changes introduced earlier, including some structural changes associated with the forthcoming move to a distributed memory (MPP) platform and was the last version to be released on the Cray C90. HADCM3, the emergent climate model for the next set of major production runs was mostly developed with modifications against version 4.0, being upgraded to 4.1 in early 1997. The operational forecast model continued at 4.0 during the lifetime of the Cray C90.

The repository for nupdate modifications to source code was changed, providing an RCS capability for lodging successive corrections. This preserves a history of local changes during the period testing a new version, after the initial build but before the final release.

- Extensive addition of routines and code for running on a distributed memory platform. Existing source code was protected by placing most new code within a *DEF MPP compile time switch. Changes included the rationalisation of DO loop variable limits passed down to lower level physics and dynamics routines.
- The WAVE model was introduced into the UM system, with WAVE source code converted to nupdate form. Control level changes allowed a WAVE dump to be read and interfaced into lower level routines. This required changes to STASH to deal with an extra dimension for the prognostic variables, since WAVE model variables are a function of (x,y,frequency,direction).

- Further development of the submodel paradigm, partly associated with changes for the WAVE model. ppxref file functions were replaced by a newly formatted set of STASHmaster files, with a separate file for each internal model providing a description of each prognostic and diagnostic variable known by STASH for that model.
- Addition of science routines as a preliminary to, but not defining fully, HADCM3: MOSES (Met.Office Surface Exchange Scheme) incorporating:
  1. a Penman-Monteith boundary layer and hydrology scheme;
  2. an improved multi-layer soil moisture and temperature model;
  3. an interactive vegetation canopy.
- Introduce code for calculations of the sulphur cycle, with:
  1. A new chemistry section;
  2. Wet scavenging;
  3. Boundary layer and convective transport;
  4. Tracer advection of aerosol.

1.2.13 Version 4.2/4.3

Version 4.2 was a development version, the first to be released on the Cray T3E, giving an MPP capability, with no science changes. Version 4.3 consolidated corrections for both MPP and HADCM3 science, providing full functionality for operational and HADCM3 climate models.

- Most changes were to provide an MPP capability:
  1. Introduction of message passing libraries, realised through a generalised software interface (GC/GCOM);
  2. Domain decomposition of model grids: 2-dimensional sub-areas for the atmosphere model; 1-dimensional (rows) for the ocean model;
  3. Model dump read/write and STASH output routines re-arranged to cater for gathering and scattering data for distributed processors.
- Single processor optimisation of code tailored for a cache based rather than vector based architecture.
- The change in number representation from C90 to T3E (Cray specific to IEEE) gave greater precision but a smaller range; conversion utilities were provided for using Cray C90 files.
- A new compilation system was introduced, based on Unix ’make’ operating on nupdate decks, providing consistency beween pre-built and locally generated compiled code, with flexibility for quick compilations.

1.2.14 Version 4.4

The second UM version (after 4.0) to be released as a complete portable package, tested on a variety of platforms.

- OASIS coupler code, allowing external models to be coupled with UM models with a minimum of changes to the external model.

- Well-formed i/o, with data files re-structured to start at sector boundaries, improving i/o performance to disk.
- New science sections (precipitation, convection, vertical diffusion, dynamics adjustment and filtering), optimised for T3E processors but not bit reproducible with equivalent sections.
- Introduction of mixed phase cloud and precipitation scheme.
- New options for science routines, constituting the main components of the planned new climate experiment, HADCM4:
    1. MOSES II: enhancement of MOSES, with surface hydrology and boundary layer scheme modifications;
    2. Radiationally active anvils in convection;
    3. A new ice scheme in radiation.
- Introduction of a scheme for interactive vegetation.
- Climate meaning for a Gregorian calendar (enabled only for a 360 day calendar previously).
- Faster energy correction: accumulate fluxes over a period rather than summing every timestep.
- New utilities for generating boundary condition files: bcreconf (reconfigure from another resolution) and makebc (generate from dumps).

Chapter 2

Understanding the Unified Model

2.1 Atmospheric Physics

When the UM is run at the standard climate resolution grid-points are about 300 km apart and even in a high resolution forecast run grid-points are still 60 km apart. Many atmospheric processes operate on much smaller length-scales and therefore cannot properly be resolved in the UM. Such processes are *parametrized*: it is assumed that the gross effects of these processes on resolved scales can be predicted using a modelling scheme which itself operates on the large-scale fields in the model. This assumption seems to be justified in practice.

The following sections describe the physical processes which are represented in the model.

2.1.1 Cloud Scheme

Author: A.C. Bushell, Last Updated: 29 Apr 98

Cloud scheme: The Case.

In the earth's atmosphere, regions in which the local supersaturation of water vapour exceeds more than a few percent rapidly become filled with cloud in liquid or ice phase (depending on temperature). Cloud can have a significant effect upon the radiative heat balance in the atmospheric column within which it appears, leading to direct changes in forecast variables, such as surface temperatures, as well as changing the simulation via its impact upon *e. g.* the boundary layer structure. The function of cloud as a water reservoir in the hydrological cycle is generally considered to be secondary because on average more water is stored as vapour or ejected as precipitation.

Cloud scheme: The Specification.

In models with horizontal gridscales of the order 100 m, cloud cover can reasonably be set to either 0 or 1 for subsaturated or supersaturated gridboxes respectively. In GCMs, however, the resolutions encountered are such that partial cloud coverage is likely, with the possibility of cloud occurring even when the mean water content of a gridbox is below saturation. The UM Large-Scale Cloud parametrization scheme has thus been developed to fulfil two distinct functions. Firstly, it returns for each gridbox an instantaneous fractional cloud cover, which can be more accurately thought of as a 'non-clear sky fraction'. Secondly, it partitions the total water content in a grid-box between water vapour content and mean cloud water content. This function of the cloud scheme provides a strong link with the large-scale precipitation scheme which subsequently acts to remove some or all of the cloud water generated. As a further complication, the same code is also used to transform between conserved variables of liquid water temperature and total water content, which are used for advection in the UM, and the standard temperature and moisture variables: cloud fractions are not always updated in calls for this purpose. Note that the scheme is specifically dealing with macroscopic processes responsible for the general location of cloud and not with localized, microscopic physical processes such as nucleation and droplet growth that are more normally associated with 'cloud physics'.

Cloud scheme: The Description.

The cloud scheme, which is described in the open literature ([58]), uses a statistical parametrization method: full details can be obtained from [15]. In brief, fluctuations about the grid-box mean total water content are parametrized via a probability density function (PDF) with a width specified through the critical relative humidity parameter ($\mathrm{RH_{crit}}$). Care is needed when choosing the PDF shape in order to make sure that cloud fraction, gridbox mean cloud water contents and in-cloud water contents all behave monotonically as the cloud fraction rises from 0 to 1. In practice this has forced the use of symmetric PDFs, currently triangular. At RH below $\mathrm{RH_{crit}}$, no cloud is formed. When total water contents reach the saturation specific humidity with respect to liquid water temperature, the cloud cover is 0.5 and this rises to 1 for total water contents a factor $(1 + (1 - \mathrm{RH_{crit}}))$ above saturation. Significantly, there is a close relationship between cloud fraction, cloud water contents and relative humidity over this range, which future development of the cloud scheme is seeking to weaken. The scheme is generally classed as semi-prognostic because, although the cloud variables are overwritten each timestep and not updated as in a fully prognostic scheme, they do interact with schemes called later in the UM timestep sequence and thus influence the model evolution.

Cloud scheme: The Practicalities.

There are currently two variants of the Large-scale cloud scheme. *1A*, for use with Large-scale precipitation schemes up to *2E*, and *2A* which is for use with the prognostic ice microphysics (*3A*) precipitation scheme. In the *1A* version, a single cloud fraction is associated with the total cloud water content, which is subdivided into ice and water phases based upon temperature. The *2A* version does not update ice phase cloud water at all but uses it to diagnose an ice cloud fraction. The liquid cloud water and vapour contents are treated as for the *1A* scheme and produce an associated liquid cloud fraction. A total cloud fraction is obtained by assuming a minimum overlap between the ice and liquid cloud. Note that there is no longer a direct link between the ice cloud variables and relative humidity as ice can be advected away from the high RH regions in which it originally forms.

Cloud scheme: The Parameter.

Finally, a note on $\mathrm{RH_{crit}}$. This is currently specified as a level dependent constant (on a separate UMUI panel from the Large-scale Cloud section) which varies from values close to 1, on near-surface levels, dropping to a constant value in the free troposphere and above. The values chosen are somewhat arbitrary, but $\mathrm{RH_{crit}} = 1$ formally forces cloud to flip between either 0 or 1 and is the expected value in the limit of high resolution. Values of $\mathrm{RH_{crit}} < 0.7$ are usually associated with excessive cloud amounts and top of atmosphere albedo, especially when used in the boundary layer. Production of locally-diagnosed $\mathrm{RH_{crit}}$ is an area of ongoing development.

2.1.2 Radiation

Author: J.M. Edwards, Last Updated: 25 Mar 98

Shortwave radiation from the Sun is absorbed in the atmosphere and at the Earth's surface and provides the energy to drive the climate system. Longwave radiation is emitted from the planet to space and also redistributes heat within the atmosphere. To represent these processes a GCM must contain a radiation scheme.

Various radiative processes occur in the atmosphere: gases, aerosols, cloud droplets and ice crystals absorb radiation and and emit thermal radiation. Aerosols, cloud particles and air molecules scatter radiation. The surface absorbs, reflects and emits radiation. The radiative properties of atmospheric constituents may vary rapidly with frequency and the geometrical arrangement of clouds is an important influence on the radiation budget. The modelling of atmospheric radiation is therefore potentially very complicated and a radiation scheme suitable for a GCM must involve some approximations: different approaches lead to different schemes which are discussed in more detail in [9].

The original longwave scheme appears in three variants, *1A*, *1B* and *1C* which differ in their treatment of gaseous absorption. All schemes include absorption by water vapour, carbon dioxide and ozone, but version *1C* also includes absorption by methane, nitrous oxide, CFC11 and CFC12. The radiative properties of clouds are taken to depend only on the liquid or ice water path, not on the sizes of particles. Clouds may be maximally or randomly overlapped.

The original shortwave scheme (version *1A*) includes absorption by water vapour, carbon dioxide and ozone. Rayleigh scattering by air molecules is treated simply. The scattering of radiation by clouds is more difficult to treat in a computationally efficient manner and version *1B* allows faster more approximate calculations. Cloud is assumed to be randomly overlapped in the vertical and this can lead to excessive cloud cover: under version *2A* cloud is reduced to single layers of high, medium and low cloud of composite optical depths, and this version is more commonly used. Version *2B* is specifically intended for the climate model HadCM2 and is not otherwise used.

Version *3A* in the shortwave and longwave regions is a more recent scheme, designed to treat both spectral regions as far as possible within a common flexible framework. The use of external *spectral files* allows freedom in the choice of gases, aerosols and cloud parametrizations to be included in the run. The generation of such files is not a trivial matter but standard files are provided with the UM. Gaseous absorption by a subset of the gases included in the spectral file may be included in the calculation. A Climatological background aerosol model is available and the radiative effects of sulphate aerosols may be included. Convective and stratiform clouds are treated separately and may be overlapped in different ways in the vertical, though normally one should use the option of vertically coherent convective cloud which maintains the vertical coherence of stratiform and convective cloud in adjacent layers, but treats cloud in non-adjacent layers as being randomly overlapped. Within clouds water droplets and ice crystals are treated separately with optical properties depending on their sizes.

Version *3A* in both the LW and the SW and versions *1* and *2* of the SW include an optional microphysical parametrization of the size of cloud water droplets in clouds as a function of the liquid water content. If this is not selected it assumed that water droplets have an effective radius of 7 μm.

2.1.3 Boundary Layer

Author: R.N.B. Smith, Last Updated: 22 May 98

Because of the interaction of the earth's surface with the atmosphere the bottom kilometer or two of the atmosphere (the "*boundary layer*") is often turbulent. The turbulent motions are not resolved by models with the resolution used for NWP or climate research. They are important to parametrize because they transport momentum, heat, moisture, aerosol and pollutants mainly in the vertical direction.

The scheme in the Unified Model has two main parts:

1. The surface exchange scheme determines the turbulent transports from the surface to/from the atmosphere in terms of the surface variables and the atmospheric variables at the bottom model level (typically between 10 and 30 metres above the surface). The surface exchange coefficients are functions of the stability of the surface layer and quantities which characterise the underlying surface such as its roughness and wetness. The 3A version of the scheme uses the bulk Richardson number of the surface layer as its stability parameter whereas the 6A scheme uses the Monin-Obukhov length. The latter allows the direct use of empirically determined stability functions but requires iteration.

   An important part of the surface momentum flux over land comes from form drag due to the turbulence induced by unresolved orography. This is represented via an effective roughness length for momentum. The effective roughness can be an order of magnitude or so greater than that due to vegetation.

2. The boundary layer turbulent mixing scheme determines the turbulent transports above the surface layer. The 3A scheme is essentially a “local” parametrization. It uses the vertical profile of the locally determined Richardson number to determine the top of the turbulent boundary layer (and hence the length scale of the turbulence) and also the stability modulation of the turbulent mixing coefficients. This scheme additionally has the option of ”rapid mixing” of scalar quantities from the surface throughout the boundary layer in unstable conditions.

   The 6A scheme takes a different approach and classifies the boundary layer into six types:

   (i) stable boundary layer
   (ii) boundary layer with stratocumulus over a stable surface layer
   (iii) well mixed boundary layer
   (iv) boundary layer with a decoupled statocumulus layer not over cumulus
   (v) boundary layer with a decoupled stratocumulus over cumulus
   (vi) cumulus capped boundary layer

   The scheme has an explicit parametrization of entrainment into turbulent layers. This entrainment may occur at the top of a decoupled cloud layer and/or at the top of a surface-based mixed layer. The turbulent mixing coefficients below the entrainment level(s) are parametrized non-locally in terms of scaling velocities characterizing the surface or cloud top turbulence forcing mechanisms, the depth of mixing and specified shape functions. The depths of the mixing layers are determined by testing the buoyancy of plumes rising from the surface or descending from cloud top. When a cumulus capped boundary layer is diagnosed the surface-based turbulent mixing is not applied above the lifting condensation level. The mixing coefficients in stable boundary layers are determined from the local Richardson number as in the 3A scheme.

2.1.4 Sulphur Cycle

Author: M. Woodage, Last Updated: 5 Jun 98

The Sulphur Cycle aims to represent the mixing, transport, dry and wet scavenging, and simplified chemistry of Sulphur interactively in the UM. It was first made available in vn4.1, and is under continuing development, so any potential users are advised to contact the aerosol parameterisation group at the Hadley Centre before attempting to run it. The system allows for natural emissions of oceanic dimethyl sulphide and volcanic sulphur, and anthropogenic sulphur dioxide to be inserted via ancillary files supplied by the user. The Sulphur fields generated in the model from these are subject to wet and dry oxidation from various oxidants (which again need to be supplied by the user), to produce 3 modes of sulphate aerosol: Aitken, accumulation and dissolved. SO_2 and all modes of SO_4 are subject to wet and dry deposition in the relevant physics sections of the model. Only one version of the chemistry is so far available, but care must be taken to choose compatible versions of other physics routines when running the Sulphur Cycle.

2.1.5 Precipitation

Author: D.R. Wilson, Last Updated: 25 Mar 98

Precipitation is a vital part of the earth's water cycle. It removes moisture from the atmosphere. The Unified Model can produce precipitation from two schemes. The convection scheme removes moisture generated in sub grid scale convective systems. The large scale precipitation scheme, described here, removes cloud water that is resolved on the grid scale, such as that produced by frontal systems.

There are five options of large scale precipitation scheme. Options 2B, 2C, 2D and 2E are all variants of a simple bulk parameterization which converts water content into precipitation. The transfer of water content to rain is specified by two parameterizations. Firstly, there is a conversion rate which transfers water content to rain independent of rainrate. There is also an accretion term which models the sweepout of water content by precipitation. This is proportional to the precipitation rate. The ice phase is parameterized using a simple temperature dependent function to describe the ratio of liquid mixing ratio to total mixing ratio of liquid and ice. The diagnosed ice is allowed to fall at a parameterized fall speed. Precipitation will freeze or melt as it crosses the zero degree isotherm. Rain may evaporate if it falls into a sub saturated region.

The mixed phase precipitation scheme, 3A, is completely different. It predicts the amount of ice present by growing it from and decaying it to liquid, vapour and rain. Transfer terms convert one phase of water to another, modelling the processes of ice nucleation, depositional growth or decay of ice, riming, melting, capture of raindrops by ice particles, evaporation of rain, accretion of rain and autoconversion of rain. These terms are based on cloud physics processes. The partitioning between liquid water and ice is predicted in this scheme and not diagnosed using a temperature function. This is a significant improvement over the other large scale precipitation schemes.

The choice of cloud scheme depends upon the choice of precipitation scheme. See [12] for further details of the large scale precipitation scheme.

2.1.6 Convection

Author: A. Grant, Last Updated: 28 Apr 98

The UM convection scheme represents the transports of heat, moisture and momentum associated with cumulus convection occuring in a model gridbox. The same scheme is used to represent both precipitating and non-precipitating convection. The convection scheme models an ensemble of cumulus clouds within a gridbox as a single entraining-detraining plume. Entrainment and detrainment represent interactions that occur between the clouds and the environment through which they ascend. The scheme is described detail in [13].

The scheme can be broken into four components,

1. Triggering, i.e. determining whether convection will occur in a gridbox.
2. Cloudbase closure, i.e. determining how much convection will occur. The amount of convection is determined by the mass transported through cloudbase.
3. The transport model determines the changes to the model temperature, moisture and wind fields due to convection and the precipitation associated with the convection.
4. The convective cloud scheme which calculates the amount of cloud associated with the convection. The convective cloud amount is passed to the radiation scheme.

There are currently two variants of the convection *3A* and *3B* which differ only in the treatment of the freezing and melting of precipitation. Version *3C* is the same as *3A* but is optimized for the T3E.

Several options for the convection scheme can be selected through the UMUI.

1. CAPE Closure. This is an alternative for the cloudbase closure. The standard closure is based on an emipirical relationship between mass flux and the buoyancy of the plume at cloud base. The CAPE closure relates the cloudbase mass flux to the *Convective Available Potential Energy* of the environmental profile over the depth of the convection.
2. Convective Momentum Transports. Determines whether the convection scheme represents the transport of momentum by cumulus convection.
3. The Anvil Cloud Scheme. The standard scheme assumes that the convective cloud amount is constant with height, between cloudbase and cloud top. In the anvil scheme the convective cloud amount is assumed to vary with height in a way that is intended to represent the cloud associated with Mesoscale Convective Systems.

2.1.7 Land Surface and Vegetation

Author: Peter Cox, Last Updated: 21 Oct 98

The land surface scheme calculates the fluxes of heat, moisture and momentum at the land-atmosphere interface, and updates the surface and sub-surface prognostic variables which affect these fluxes. In the current UM, the surface-atmosphere fluxes are calculated within the surface exchange section of the boundary layer code, whilst the sub-surface fluxes and variables are updated in the hydrology section. There are 2 distinct versions of the land surface scheme available at UM version 4.4, and each of these is described briefly below.

The UKMO Scheme

This scheme was developed in the mid to late eighties ([61]), and is still in use in the NWP configurations of the UM. In the UKMO scheme the state of the land surface is defined by 3 hydrological variables (available soil moisture, canopy water and lying snow) and 4 soil temperatures. The surface energy partitioning is calculated using a resistance formulation. The sensible heat and the various evaporation components are subject to an aerodynamic resistance which is dependent on atmospheric stability and the surface roughness (see the boundary layer description for further details). Evapotranspiration from the soil moisture store is also limited by a surface resistance which is dependent on surface type but is fixed in time. The hydrology section calculates the interception of rainfall by the plant canopy, and the “surface” and “drainage” components of the runoff, before updating the soil moisture in the rootzone and the lying snow mass. In total the scheme has 7 soil dependent and 7 vegetation dependent parameters. Geographically varying fields of each of these parameters are derived offline using the “Central Ancillaries Program”, by assigning typical values to the vegetation and soil classes specified in the Wilson and Henderson-Sellers dataset ([62]). The scheme simulates just a single surface type within each gridbox so effective parameters must be derived for each point. These are taken as the area-weighted mean for all parameters but the roughness length, which is aggregated using a blending height approach. The UKMO scheme corresponds to the 3A boundary layer section and the 1A hydrology.

MOSES I

A new scheme, called “MOSES” (Met Office Surface Exchange Scheme) was designed to eliminate some of the most obvious limitations of the UKMO scheme, and has been adopted for use in climate configurations of the UM from HadCM3 onwards. MOSES utilises a 4 layer soil hydrology model based on a discretised version of the Richards’ equation, and includes the effects of soil water phase changes on the permeability and heat balance of the soil. The fixed surface resistance values in the UKMO scheme have been replaced by a “bulk stomatal” resistance model, which simulates how vegetation controls evapotranspiration as a function of surface temperature, solar radiation, vapour pressure deficit, soil moisture stress and atmospheric CO_2 concentration. Other changes in MOSES include deeper plant rootdepths, and a Penman-Monteith flux formulation in which the surface temperature becomes a diagnostic “skin temperature”. The major differences between the UKMO scheme and MOSES are summarised in table 2.1, and further details are given in [63]. The MOSES scheme corresponds to the 5A hydrology coupled to either the 5A or 6A boundary layer sections.

	UKMO Scheme	MOSES I
Surface Temperature	Prognostic - top soil layer	Diagnostic - from surface energy balance
Soil Hydrology	single layer	4 layer Richards' equation
Soil Hydraulic Parameters	area means	from area mean particle size distribution
Soil Thermal Parameters	constants	vary with soil water and ice content
Soil Water Phase Change	not included	by apparent heat capacity approach
Canopy Conductance	non-interactive	interactive and dependent on CO_2
Rootdepths	$\leq$1.5 metres, $\geq$0.5 metres	$\leq$3.0 metres, $\geq$1.0 metres
Surface CO_2 fluxes	not simulated	simulated

Table 2.1: Principle differences between the UKMO land surface scheme and MOSES I

2.1.8 Gravity Wave Drag

Author: S. Webster, Last Updated: 19 May 98

The gravity wave drag parametrization aims to represent the transport of momentum by unresolved gravity waves. Currently, only the impact of those gravity waves forced by orography is represented, although a parametrization of convectively forced gravity waves is being developed.

There are three GWD schemes to choose from in the UM. The first (*1A*) scheme is what might be described as the classical first generation GWD scheme. This scheme essentially assumes the surface stress is orientated in the same direction as the low level wind and is proportional to the sub-grid orographic standard deviation. The parametrization assumes that linear hydrostatic waves are generated and so the stress is deposited according to a simple saturation hypothesis. This scheme exerts most of the drag in the lower stratosphere.

The surface stress calculation for the second (*2A*) scheme is identical to that of the first. The *2A* scheme has a different stress deposition algorithm; the stress is linearly deposited between the surface and the top of the atmosphere, unless a critical level is encountered in which case all the remaining stress is deposited in the current level. This scheme usually, therefore, exerts a uniform drag through the full depth of the atmosphere.

Both the *1A* and *2A* schemes permit the Froude number option may be chosen. This option limits the amplitude of the surface stress by ensuring that the Froude number always exceeds unity.

The most widely used GWD scheme is the *3A* version. This scheme builds on the original *1A* scheme and accounts for the anisotropy of the sub-grid orography in the calculation of the surface stress. As well as the linear hydrostatic regime, a regime based on the hydraulic jump response is also modelled. Trapped lee waves are also represented in this scheme. The *3A* scheme exerts more drag at low levels and less drag in the stratosphere than the *1A* scheme.

2.2 Atmospheric Dynamics

Author: Terry Davies, Last Updated: 3 Aug 98

In atmospheric modelling the dynamics typically is the equation set without the forcing terms due to the parametrized physical processes. These are the equations of motion, the conservation of mass or continuity equation, an energy equation and an equation of state i.e. six equations for the three wind components, temperature, pressure and density. The equations of motion include Coriolis terms due to rotation of the Earth. Each tracer variable in the system requires a transport equation. In particular there is a transport equation for moisture. In the UM the advected moisture variable is total water content and the thermodynamic variable used is liquid/frozen water potential temperature. For large-scale flows, the hydrostatic approximation is usually made. This eliminates vertical accelerations and leads to a diagnostic equation for the vertical velocity.

The equations are written in a spherical coordinate system and discretized horizontally on a regular latitude-longitude grid. In global configurations the grid poles are coincident with the geographical poles. In limited-area configurations the grid is rotated so that the coordinate equator crosses the middle of the limited-area domain thus giving an almost uniform grid over the limited-area.

In the vertical a pressure-based hybrid coordinate is used. The coordinate is terrain following sigma (normalised by the surface pressure) near the surface (thus at sigma=1) and pressure in the top-most layers and therefore are almost horizontal. Between the terrain following region and the top, the coordinate levels flatten gradually. The terrain following system has the advantage of not having surfaces intersecting the surface over orography and having a simpler lower boundary condition (the vertical velocity in the sigma system is zero at both the surface and at the top of the model). The flattening of the surfaces reduces the problem that the sigma system has in that the pressure-gradient force splits into two terms of opposite sign which are each large over steep slopes which can lead to large errors. It also helps in the treatment of clouds since cloud bases tend to be horizontal. Most hydrostatic, pressure-based models make a shallow-atmosphere approximation and include only the vertical component of the Coriolis terms and also neglect certain metric terms arising from the spherical coordinate system. The radius in the metric coefficients of the derivative operators in the spherical system can then be assumed to be constant, the radius of the Earth. The UM does not make these approximations, in particular it retains all three components of the Coriolis force and the metric terms.

The equations are solved with a split-explicit technique. The splitting refers to the separation of the terms associated with the gravity-wave adjustment and the advection by the winds. A longer timestep can be used in the advection step since horizontal winds are typically a third that of the fastest gravity waves. Normally, three adjustemnt steps are taken for each advection step. A forward-backward scheme is used in the adjustment step which updates the pressure and horizontal winds. The horizontal winds and diagnosed vertical velocity over the three adjustment steps are averaged and then used as the advecting winds in the advection step which includes all the transport equations. The advection step uses a two-stage Heun advection scheme with an option to include fourth order terms for greater accuracy but slightly increased cost. Using the winds averaged from the adjustment step enables certain conservation properties to be maintained by the advection. Transport of scalars can also be done by using a monotone tracer advection scheme. The montone scheme (actually a total variation diminishing, TVD scheme) prevents overshoots and undershoots developing which would otherwise lead to negative values being generated in fields that should be positive.

In global configurations, the convergence of the meridians of the latitude-longitude grid as the poles are approached leads to an increase in the Courant number as the east-west gridlength decreases. This would lead to instability but is controlled by the use of Fourier filtering along latitude lines. The fields are Fourier transformed and the unstable components (those whose short wavelength imply a Courant number greater than unity for a given timestep) are set to zero or smoothed. The components are then transformed back into real space.

The dynamics also includes horizontal diffusion to prevent the accumulation of energy at the smallest scales. This energy arises due to forcing at the smallest scales and noise excited by errors in the numerical scheme, either from truncation errors, computational modes or lack of balance. The diffusion is performed by applying a conservative del-squared operator one or more times. The more times it is applied the more its effect is confined to the smallest scales without damaging the longer scales of interest but at increased cost. The diffusion is also designed to switch off over steeply sloping surfaces thus avoiding false transport of heat and moisture over steep terrain.

Further efficiencies in the scheme are made by allowing the use of longer timesteps for the physical processes and making multiple dynamics steps for each physics step. Multiple dynamics timesteps are also used to reduce the dynamics timestep whenever exceptionally strong winds occur, for example in the stratospheric polar-night jet, or if noise is detected in the top-level divergence. This increases the robustness of the operational model.

Precise details of how the dynamics is formulated can be found in UMDP 10 [5].

2.3 Ocean Dynamics and Physics

Author: R.A. Wood, Last Updated: 24 Sep 98

The UM ocean model is a primitive equation model based on the widely-used Bryan-Cox formulation [44]. Therefore the numerical formulation of the model is similar to the widely used MOM model and its variants, although there are some differences in the code structure. The UM code was originally based on the version written for the Cyber 205 by Cox [46], but has evolved over the years to accomodate both new computer architectures and much additional physics. The text below describes the general features of the model and the options available. References to the open literature and to UMDPs are used to direct the reader to more detailed descriptions of the model components.

2.3.1 Basic Dynamical Formulation

For a more detailed descrition of the basic formulation of the model, see [17], [46] and [44].

The model is based on the hydrostatic, Boussinesq primitive equations. The equations are discretised onto an Arakawa staggered ’B’ grid in the horizontal, and onto a number of vertical levels or gridboxes of variable spacing. All variables except vertical velocity are calculated at the level of the gridbox centre; vertical velocity is calculated at the level of the gridbox boundaries (’Lorenz’ vertical grid). Global and limited area domains can be set up, and from UM version 4.5 open boundary conditions will be available [21].

A polynomial approximation to the equation of state is used, driven by the model’s basic thermodynamic variables, potential temperature and salinity. Additional (dynamically passive) tracers may also be prescribed and are timestepped by the model using the same advection, diffusion and mixing as for potential temperature and salinity. In what follows, such additional tracers are referred to as ’extra tracers’, and the term ’tracers’ refers to all tracers in the model, including potential temperature and salinity.

A rigid lid formulation is used to filter out fast surface gravity waves (inclusion of a free surface is under consideration, but not available at UM version 4.5). The rigid lid condition allows a streamfunction to be defined for the vertically integrated (barotropic) flow, and the model velocity calculation is split into barotropic and baroclinic parts. The barotropic streamfunction is calculated at each timestep and involves solution of an elliptic partial differential equation using a relaxation technique. The model land-sea mask may include both ’islands’ and ’continents’, the latter being islands on which the streamfunction is constrained to zero. The use of continents can substantially reduce the cost of the streamfunction calculation where there are large land masses in the model domain.

A centred difference advection scheme is used for tracers and momentum at v 4.4. From v 4.5 a third order upwind scheme (’QUICK’,[52]) is available as an alternative. The Coriolis term may be treated explicitly or semi-implicitly.

Fourier filtering of model fields at each timestep is available as an option. The normal use of this is in global models, to reduce gridscale noise and instability at high latitudes where the meridians converge to give short zonal grid spacing.

Timestepping is by a leapfrog method which is second order in time. In order to prevent time splitting of the solution on odd and even timesteps, a Robert time filter is applied to the solution at each timestep. The use of a periodic forward timestep, as in the original Bryan-Cox scheme, is therefore no longer required for numerical stability. However the periodic forward timestep is retained in order to reduce the size of restart dumps. The forward timestep requires the storage of only one time level, and restart dumps are therefore always written out on forward timesteps. The frequency of forward timesteps and of restart dump production are controlled (independently of each other) through the UMUI. A model option allows application of the ’distorted physics’ technique [45], [60] to allow longer timesteps to be taken in long spinup integrations.

2.3.2 Surface Forcing

For ocean-only runs, surface forcing fields are supplied through ancillary files or, if no time variation is required, in the initial dump. For coupled atmosphere-ocean runs some or all of the forcing fields are passed from the atmosphere submodel by the coupling routines. The following forcing fields are available:

Surface heat flux: a non-penetrative surface heat flux is required to drive the model [22]. Additionally, a penetrative short wave flux can optionally be provided; this is distributed over the top few model levels according to a simple radiation model [19]. It is also possible to specify a so-called 'Haney' term, i.e. an additional surface heat flux which acts to relax the sea surface temperature (SST) to specified values on a timescale defined by the user in the UMUI. If the 'Haney forcing' is chosen, the reference SST field must be defined in the initial dump or (in the case of a time-varying field) through an ancillary file. An additonal 'flux correction' term can also be supplied through an ancillary file (normally only of use in a coupled atmosphere-ocean run).

Surface freshwater flux: a surface freshwater flux is required to drive the model [22]. Optionally a river runoff field can also be provided; this is simply an additional surface freshwater flux and is most often used in coupled models. It is also possible to specify a so-called 'Haney' term, i.e. an additional surface freshwater flux which acts to relax the sea surface salinity (SSS) to specified values on a timescale defined by the user in the UMUI. If the 'Haney forcing' is chosen, the reference SSS field must be defined in the initial dump or (in the case of a time-varying field) through an ancillary file. An additonal 'flux correction' term can also be supplied through an ancillary file (normally only of use in a coupled atmosphere-ocean run).

Surface momentum flux (wind stress): x and y components are required to drive the model.

Wind mixing power: a surface turbulent kinetic enrgy flux or wind mixing power field is required to drive the Kraus-Turner and Large et al. mixed layer models (see 'Surface Mixed Layer' below).

2.3.3 Vertical/Diapycnal Mixing

Surface Mixed Layer

Three options are available to model the surface mixed layer:

a. Kraus-Turner ([50], [18]) mixed layer model

b. Large et al. [51] 'KPP' mixing scheme (from v 4.5)

c. A simplified version of *b* ('Quadratic Large')

One only (or neither) of options *a* and *b* may be chosen through the UMUI. Option *a* is a bulk mixing scheme which completely homogenises all tracers in the mixed layer; it does not act on momentum. Option *b* is a diffusion based scheme which acts on tracers and momentum. Option *c* is designed for optional use in association with *a*, and acts on tracers and momentum. Both schemes *b* and *c* are designed for use in conjunction with either the Pacanowski and Philander or the Peters et al schemes (see 'Subsurface Mixing' below), and produce a vertical diffusivity profile near the surface which joins smoothly with the subsurface profile. All the mixed layer schemes assume sufficient vertical resolution to resolve the summer mixed layer; in most UKMO applications at the time of writing the vertical resolution is 10m near the surface.

Subsurface Mixing

Subsurface vertical mixing is mainly achieved by diffusion. The following options are available for choice of diffusivity, controlled by the UMUI:

a. Constant diffusivities for tracers and momentum *b.* Pacanowski and Philander [53] 'K-theory' scheme for tracers and momentum

c. Peters et al. [54] scheme for tracers and momentum (from v 4.5)

One and only one of these options must be chosen through the UMUI. The vertical diffusion equation may be solved either explicitly or implicity. The explicit solver may only be used if option *a* is chosen and isopycnal diffusion (see under 'Horizontal/Isopycnal Mixing' below) is *not* chosen; otherwise the implicit scheme must be used (generally this is necessary for numerical stbility). Schemes *b* and *c* allow specification of a background diffusivity which is allowed a linear variation with depth. Both schemes generally revert to this background value in the deep ocean.

In addition to the above, a convective adjustment scheme is run at the end of the tracer timestep. The scheme used [55] mixes sections of the water column which are statically unstable, to leave a profile with no residual instability.

2.3.4 Horizontal/Isopycnal Mixing

Momentum

Momentum mixing is achieved by horizontal diffusion. The diffusivity is set in the UMUI and is allowed to vary with latitude in order to allow midlatitude western boundary currents to be resolved while not imposing too great numerical stability constraints at high latitudes. From v 4.5 it is also possible to include a biharmonic momentum mixing term in addition to the Laplacian diffusion.

Tracers

The following options are available for tracer mixing (diffusion), controlled by the UMUI:

a. Constant horizontal diffusivity

b. Redi ([56], [23]) isopycnal diffusion

c. Griffies et al. [48] isopycnal diffusion (an improved numerical formulation of *b* (from v4.5))

One and only one of the above must be chosen through the UMUI.

2.3.5 Mesoscale Eddy Parameterisation

The eddy parameterisation of Gent and McWilliams [47] ('GM') is available as an option thorugh the UMUI. Either constant eddy transfer coefficients or variable coefficients can be chosen, the latter being based on the scheme of Visbeck et al. [59]

The GM scheme can only be chosen when isopycnal diffusion (of either the Redi or Griffies type) is chosen. The numerics of the GM scheme are also different depending on which isopycnal diffusion scheme is chosen (at vn4.5). If the GM scheme is chosen, then a more scale-selective, biharmonic form of GM is also available [57]; this is useful for removing grid-scale noise from the model.

From v 4.5 there is an additional option to include the GM scheme using a skew-diffusive formulation [49]. Although somewhat faster than the advective approach, this option precludes being able to output the GM 'bolus' velocities and does not allow use of the more scale-selective biharmonic form of GM.

When the GM scheme is selected, one must also select isopycnal tapering of diffusivities in the UMUI - this reduces the isopycnal diffusion and GM coefficients depending on the slope of isopycnals, to ensure that numerical stability is retained.

2.3.6 Sea Ice

A sea ice model is available as an option through the UMUI. Thermodynamics are based on the Semtner zero layer model, with a leads parameterisation included. A number of dynamics options are available. See [20] for further details.

2.3.7 Ocean Data Assimilation

Author: M.J. Bell, Last Updated: 27 Nov 98

The ocean data assimilation code uses the filtered increment version (Lorenc 1992, see [69]) of the analysis correction scheme developed for NWP by Lorenc, Bell and Macpherson (1991, see [68]). Observations are used in four groups: altimeter data, surface temperature data and profiles of temperature and salinity. For each group a number of two dimensional analyses are performed to find increments to model fields. The surface height increments based on altimeter data are applied to the model's potential temperature and salinity fields using the technique of Cooper and Haines (1996, see [65]). Surface temperature increments are combined with increments from the temperature profile data throughout the model's mixed layer. Geostrophic baroclinic current increments are made to balance the temperature and salinity increments from the profile data.

The observation files use the UM ancillary/dump file format. They can be prepared using the new Observation Processing System (OPS) and are specified in a namelist by the user. The forecast error correlation scales can vary (slowly) with latitude and longitude and be different north-south from east-west and again are specified by the user in a namelist. The observation error ratios (relative to the forecast field) are usually obtained from the observation file. Work is in progress to enable the correlation scales and error ratios to be based more closely on statistics from FOAM assimilations. Statistics on differences between the observations and analyses, background fields or climate fields can be calculated using the old Data Preparation System (DPS) or the NWP Verification System.

The assimilation code can be used in limited area or global models with uniform or stretched grids and on grids with rotated poles. It has also been adapted to work efficiently on the CRAY T3E. Technical reports describing the use of the scheme include Bell (1994) [64], Forbes (1995) [66] and Forbes (1996) [67].

Chapter 3

Code Management and Structure

3.1 Managing and Modifying Source Code

Author: Rick Rawlins, Last Updated: 13 Mar 98

3.1.1 How is the source code stored?

Source code is held in 2 files, one for Fortran plus C code, and one for Unix scripts, which are managed by the nupdate utility and are each described as an “nupdate program library”. Nupdate originated as a Cray proprietary package - “update”, later upgraded to “nupdate” - used for source code management, which now has public domain status. An nupdate program library is a binary format file, consisting of a number of DECKs with information on previous modifications. Each DECK normally contains a single subroutine or script, with every line of code labelled - for new DECKs this is the DECK name followed by consecutive line numbers.

3.1.2 Configuration management

The Unified Model system is organised on a principle of baseline code, with aggregated sets of modifications against a current UM version being consolidated as permanent changes in the next UM version. This requires extended testing of standard UM model configurations before acceptance criteria can be met. Once achieved, all source code is frozen, the new version is deemed to be live and forms a fixed baseline for subsequent code development, normally at 6-12 months intervals. Nupdate allows source code to evolve in a structured fashion, where temporary changes can be tested and developed privately before optionally forming permanent upgrades. Formal change control mechanisms for source code are described in UMDP 2: Change Control Management of the UM System[2].

The concept of baseline code is also extended to pre-compiled object code, executable, control and data files, including the UMUI, such that all files are assumed fixed for a specific release. This requires a selection of science options (see section 4.7 (Science Options)) and functionality to be supported, as well as the capability of making temporary modifications to scripts and model source code described here. Modifying source code for the model is intimately related to the compilation procedure, which is described in section 3.3 (Compilation System).

3.1.3 More about nupdate

Two versions of nupdate exist, (1) Cray nupdate and (2) portable nupdate:

- Cray nupdate is not supported by Cray (now Silicon Graphics) but has been enabled on the T3E as part of the Met. Office C90 to T3E migration. It has the full functionality shown by Cray man pages but has not been made available by them on any other platform. The source code released into the public domain is mostly in Fortran, with some C routines.
- Portable nupdate has been written in C for the portable model and is described in UMDP X2: Nupdate Source Code Manager [39] which also includes details of nupdate commands. Portable nupdate operates on a native format program library that can be browsed directly. When the source code originates from Cray nupdate program libraries, it is necessary first to convert to the native format. The main lack of functionality currently lies in its inability to reproduce nupdate source code used to make program libraries (including supporting the -s option).

Nupdate features used in UM source code:

- Compile switches. The UMUI sets up a list of *DEF switches for Fortran/C code in the UPDEFS file of the job library to select routines, and blocks of code within routines, at compile-time. An example is given by the use of *IF DEF,ATMOS;*ENDIF line pairs to isolate atmosphere-specific model code.

  For script code the only *DEFs required are those in the SCRDEFS environment variable in the set-globalvars script, which is set up as part of the build process when the UM is installed on a system.
- Insertion of common code. The *CALL facility allows a common section of code (a COMDECK, typically the declaration and definition of a constant or group of control variables), to be inserted globally throughout the program. Nupdate features several sweeps through the source code, so that a modification to a COMDECK is carried through to each DECK containing the COMDECK call and a full set of DECKs is extracted which reflect the total change.
- Labelling. Each changed line of code is visible in source code files and are labelled by the name of individual modification sets. A record of previous modifications is held within the nupdate program library so that earlier versions of UM code can be retraced.

It is recommended that nupdate output options in UMUI panels are set to in,ed,um (except with portable nupdate which does not have these options). This sends to standard output an echo of your modification set, a summary of the changed lines and a list of any modifications that remain unprocessed.

Nupdate can be invoked in several modes: full, normal or quick. Full updates are used to generate new source code libraries at build time for a new version. Normal updating is used in the extraction step preceeding compilation for individual jobs to ensure that changes in COMDECKs are propagated. The quick mode allows specific DECKs to be modified but demands that a *COMPILE directive is supplied for each DECK, which places the onus on the user and is therefore not employed in the UM system.

3.1.4 Viewing source code

Master copies of each UM version $VN (= 4.4, 4.5, etc) are held on the Met. Office Cray T3E at $UMDIR/vn$VN/source/umpl for Fortran/C code and at $UMDIR/vn$VN/source/umsl for Unix scripts. These nupdate program libraries cannot be viewed directly on the Cray and require extraction through Cray nupdate. This task is performed centrally for Met. Office users and listings can be browsed via the source code listing page of the UM Documentation page of the Met. Office Web. Extra processing has added html links so that calling sequences for routines can be easily explored. On other systems than Cray, portable model users can inspect source code directly within the $UMDIR/vn$VN/source/umpl or umsl directories.

3.1.5 Making a change: Fortran or C

A simple example of a modification set, or “modset”, is given below which replaces lines labelled ATMSTEP1.77 to ATMSTEP1.79 in DECK ATMSTEP1 by a new line and introduces a new line after ATMSTEP1.96:

```
*ID ARRZF404                   */ identifier name
*/ Include test write statements comments for mod set
*DECLARE ATMSTEP1              */ declare DECK to be modified
*D ATMSTEP1.77,ATMSTEP1.79     */ delete between these 2 lines inclusive
      WRITE(6,*) 'TEST',A_STEP */ replace by new line(s)
*I ATMSTEP1.96                 */ insert after this line
      WRITE(6,*) 'REACHED HERE'
```

which can be held in a personal file. This file can contain one or more modsets and must be located on the target machine for running the model.

To include this modset in your model run go into the UMUI and select window **subindep_Compile_Mods**

```
Sub-Model Independent
-> Compilation and modifications
--> Modifications to the model
```

and enter the file name(s) containing the modification set. Execution of the atmosphere model will include your change as a temporary modification of the source code, compiled and linked to create a new executable file.

Similarly, the reconfiguration module can be re-compiled after source code modification by selection of the adjacent UMUI window **subindep_CompRecon**

```
--> Modifications for the reconfiguration
```

3.1.6 Making a change: Unix scripts

Accessing the UMUI window **subindep_ScriptMod**

```
Sub-Model Independent
-> Compilation and modifications
--> Script inserts and Modifications
```

allows modification of scripts controlling the execution of UM jobs by 3 methods:

1. The user can supply nupdate modification sets for scripts, as for the previous section. The UM system recreates selected UM scripts in a prior step to be executed in the job.
2. Scripts can be supplied by the user to be inserted either before or after the model run.
3. Environment variables can be defined explicitly to be associated with the job, giving greater flexibility in defining file locations, customised for the user.

3.1.7 Viewing changed DECKs

The updated DECKs (eg ATMSTEP1, ATMDYN1) resulting from a modification set “mymodset” can be listed with the new sequence numbers (using Cray nupdate only) by:

```
VN=4.4                           #  Version 4.4
PL=$UMDIR/vn$VN/source/umpl       #  UM program library
DECKS="ATMSTEP1,ATMDYN1"          #  Changed decks
nupdate -p $PL -i mymodset -q $DECKS -o "ed,in,um,sq" \
        -s newdecks
cat newdecks                     #  List changed decks
```

With portable nupdate it is only possible to view the modified deck without sequence numbers. This can be done by using the following nupdate command (which will also work on the Cray) instead of the one above:

```
nupdate -p $PL -i mymodset -q $DECKS -D
```

This will create individual files with the same name as the decks that have been modified.

3.1.8 Setting up modification sets

A practical method of obtaining a modification set when more than a few lines of code are involved is to edit a copy of a DECK directly. The Cray utility nmodex can then be applied to form a difference of library and edited DECKs as follows:

```
cat > listdecks <<EOF         # list of decks to be changed eg. ATMSTEP1
*C ATMSTEP1
EOF
VN=4.4                        # Version 4.4
PL=$UMDIR/vn$VN/source/umpl    # UM program library (umpl/umsl)
EDITDECKS=editdecks           # file which will contain edited decks
nupdate -p $PL -s $EDITDECKS -i listdecks       # extract source without
                                               # sequence numbers
[ edit decks in file editdecks ]
nupdate -p $PL -s temp.s -o sq -i listdecks     # extract source with
                                               # sequence numbers
nmodex -e $EDITDECKS -s temp.s -h \
            -i GUUZF404 -m new.modset           # get difference
```

creates a new modification set ’new.modset’ from an edited copy ’editdecks’ referred against the program library. [Note, Cray nmodex creates a lengthy header which should normally be deleted by the modset author to prevent contention with other conventions for labelling modsets (DC= and duplicate *ID can cause problems).]

There is a portable version of nmodex with limited functionality that has been adapted to provide a simple application for generating modsets for Met. Office users on HP workstations and for users of the portable model. To use portable nmodex with portable nupdate you need to do the following replacing deckname with the name of the deck you are working on:

```
VN=4.4                                   # Version 4.4
PL=$UMDIR/vn$VN/source/umpl              # UM program library (umpl/umsl)
cp $PL/deckname.dk ./deckname.tmp        # create temporary copy of deck
cut -c 1-72 < deckname.tmp > deckname.ed # remove line sequence numbers
                                         # use 1-80 on Met. Office HPs
                                         # only (see note below **)
rm deckname.tmp                          # remove temporary copy
[ edit deck in file deckname.ed ]
cp $PL/deckname.dk ./deckname.src        # Get copy of deck with
                                         # sequence numbers
nmodex -e deckname.ed -s deckname.src \
       -i GUUZF404 -h -m new.modset      # get difference
```

This will create a modset with id GUUZF404 in file new.modset.

You should not:

- Use tab characters - these may give different spacing when ported to different machines;
- Continue into column 73. Portable nupdate will truncate lines, which may cause errors for script DECKs.
- Use *COPY. A bug can introduce duplicate line identifiers.

(see UMDP 3: Software standards for the UM: Fortran and Unix [3]).

Naturally, if the correction modset is very minor, it may be simpler to key in the nupdate lines directly - this will be quicker for modsets of only a few lines. Note that it is preferable to keep the order of changed nupdate line commands the same as the order of the source in the DECK. Although nupdate will cope with processing a DECK in a non-consecutive manner, the resulting modset identifier numbers for new included code will not be monotonic increasing through the DECK, which can be confusing when inspecting the updated source.

3.1.9 Pitfalls

- Overlapping changes. Message: ”inactive line”. The main problem occurs when more than one modification set is applied and overlap may occur. If an attempt is made to change a previously deleted line then nupdate will fail. If multiple modifications are targeted for the same location, nupdate will continue, but with warning messages. Error messages do not always point to the problem DECK but may indicate the next DECK to be processed.
- Non-ascending deletion ranges. These can cause difficulties for nupdate to interpret. Avoid commands such as *D MYIDENT.9,2, even if they should appear to be valid, as when line MYIDENT.9 precedes MYIDENT.2.
- Missing *DECLARE or *COMPILE. Message: ”modification not declared” or ”unknown identifier”. A default output option for the application of Cray nupdate enabled for Met. Office users is -o dc, which requires that each modset needs to include a *DECLARE statement for the DECK being modified -

otherwise nupdate will fail. This is a useful discipline to ensure that modifications are passed through to the program. If this option is disabled, explicit *COMPILE statements are required for each DECK to be extracted for the compile step. If a statement is omitted it is possible for a program to complete successfully, seemingly include the correct modifications but not produce the desired result.

- Modification not included. Beware that code modifications may not be included in your compiled executable file. In particular, you may change code in a version of a science routine that has not been selected through the UMUI when defining the model job. For example a modification may be created to alter subroutine SWMAST in DECK SWMAST1A but this will have no effect on a job where version 2A of the SW radiation has been chosen, since the effective code selected for compilation lies in deck SWMAST2A. This may not be immediately obvious from inspection of the output, which will show that nupdate has correctly modified deck SWMAST1A, but no substitution of compile file contents will take place. Conditional *DEF statements at the head of each DECK describe the scope for different versions of sections, and a list of DECKs comprising each section version for 4.4 is held in an obj_xref file within the $UMDIR/vn$VN/source directory. More discussion on versions, releases and sections of science code can be found in 4.7 (Science Options).

3.2 UM Routines

Author: Mike Hatton, Last Updated: 1 Jul 98

An abbeviated top-down calling sequence for the Unified Model is shown below. This list is greatly simplified and excludes repeated calls, data conversions, parallel computing operations and details of I/O. There are more than 1000 subroutines, including alternative versions. Source code is controlled by the nupdate code management utility discussed in section 3.1 which also describes how more detailed listings are obtained.

```
UM_SHELL     Main program to call U_MODEL subroutine
  |
  |--STASH_PROC  Initialise STASH processing for all model variables
  |
  |--U_MODEL     Timesteps loops, call submodels
       |
       |--INITIAL
       |    |
       |    |--INITDUMP     Read in initial dump
       |    |
       |    |--INANCCTL     Initialise ancillary fields
       |    |
       |    |--IN_ACCTL     Initialise assimilations
       |    |
       |    |--INITDIAG     Initial atmospheric diagnostics
       |    |
       |    |--UP_ANCIL     Update ancillary fields
       |    |
       |    |--INITDIAGO    Initial ocean diagnostics
       |    |
       |    |--UP_BOUND     Update boundary data
       |
       |--OASIS_STEP   Ocean Atmosphere Sea Ice Soil coupler
       |
       |--ATM_STEP     Perform an atmosphere timestep
       |    |
       |    |--ATM_DYN      Perform atmosphere dynamics
       |    |    |
       |    |    |--ADJ_CTL     Atmosphere dynamics adjustment
       |    |    |
       |    |    |--TRAC_ADV    Horzontal tracer advection
       |    |    |
       |    |    |--TRAC_VERT_ADV  Vertical tracer advection
       |    |    |
       |    |    |--ADV_CTL     Atmosphere dynamics advection
       |    |
       |    |--ATM_PHYS     Perform atmosphere physics
       |    |    |
       |    |    |--CLD_CTL     Large scale cloud
       |    |    |
       |    |    |--RAD_CTL     Radiation calculations
       |    |    |
       |    |    |--BL_CTL      Boundary layer calculations
|    |    |
|    |    |--CHEM_CTL    Perform chemistry processes
|    |    |
|    |    |--LSPP_CTL    Large scale precipitation
|    |    |
|    |    |--CONV_CTL    Convection
|    |    |
|    |    |--HYDR_CTL    Surface hydrology
|    |    |
|    |    |--VDF_CTL     Vertical diffusion
|    |    |
|    |    |--GWAV_CTL    Gravity wave drag
|    |    |
|    |    |--VEG_CTL     Vegetation
|    |
|    |--AC_CTL       Assimilate atmosphere data
|    |
|    |--ST_DIAG1     Atmospheric diagnostics: dynamics
|    |
|    |--ST_DIAG2     Atmospheric diagnostics : physics
|    |
|    |--MMPP_CTL     Point print
|
|--OCN_STEP     Ocean model forecast and analysis
|
|--SLABSTEP     SLAB ocean model
|
|--WAV_STEP     Wave model forecast and analysis
|
|--UP_ANCIL     Update ancillary fields
|
|--UP_BOUND     Update boundary data
```

3.3 Compilation System

Author: K. Rogers, Last Updated: 1 May 98

3.3.1 Introduction

This document aims to give a quick introduction to the Unified Model compile system for users doing runs at vn4.4 and beyond. It contains information about the system and practical examples of how to use some of the main features.

The sections describe the main sequence of actions performed by the compile system in a user run, how to understand messages in model output, the main variables a user needs to know about, how to set up compile override files, miscellaneous information on how to do various common tasks and finally how to build your own small executables.

Please see UMDP Z5 [41] for more detailed information about the build system, which overlaps with some aspects of the compile system.

3.3.2 Actions performed in a User Run

The main functions of the compile system are listed below in the order in which they happen:

Reconfiguration

- Central and user compile options merged into one file
- Directory $URECONDIR created with subdirectory exec_build/qxrecon_dump
- nupdate used to extract source code for modified Fortran/C decks into a $TMPDIR directory
- In the reconfiguration compile directory ($URECONDIR/exec_build/qxrecon_dump) .f and .c source files are created for decks that have changed since the previous run or for all decks on the initial run
- Makefile automatically generated for all decks required
- *make* command is run to compile all uncompiled source code and link to make the executable
- Executable moved to $LOADRECON
- Label added to executable (UK Met. Office only)

Model compilation

- Central and user compile options merged into one file $UCOMPVARS
- nupdate used to extract source code for modified Fortran/C decks into a $TMPDIR directory. This includes:
  - All modified decks
  - Control decks from all relevant parts of $UMDIR/vn$VN/source/cdecks file
  - All decks required for section-version DEF combinations not pre-built.

  All deck source files are renamed as lower-case and .f/.c (Fortran/C) suffixes added.

- In compile directory $UCOMPDIR (the name of this directory should be unique to a runid eg. $DATAW/compile_$RUNID):
  - file links are set up in a previous compile of the same runid deleted
  - Source code (.f/.c) files in $TMPDIR nupdate directory compared with existing copies of the same files in compile directory. If different, newly-extracted deck replaces original deck (or used directly in initial run).
  - If using a section-version DEF combination not prebuilt, directory $REBUILD_DIR created. Relevant source files copied to subdirectories under here. Links created from compile directory to these files.
  - For other decks links are created to central copies of source code (.f/.c) files and object code (.o) files for precompiled code stored on a system-wide basis under the relevant $UMDIR/vn$VN/src/section/def-combination directory. The nupdate *DEF combinations are specified by codes eg. A1G1 for *DEFs ATMOS (A1) and MPP (G1). See top of $UMDIR/vn$VN/source/obj_xref file for *DEF code letters.
  - For decks whose compile options have changed since the previous run, old versions of the .o files are deleted to force recompilation.
- A makefile (called $UCOMPDIR/Makefile) is automatically generated for all decks required in run (unless previous version of makefile still valid)
- file links are set up in a previous compile of the same runid deleted
- *make* command is run to compile all source decks that have not been compiled or have been modified since they were last compiled and also to run link step to make executable.
- Executable is run.

In second or subsequent runs it is possible to miss out the nupdate extraction and other script processing steps and go straight to the *make* step by setting SKIP_SRC_EXTRACT to true (see 'How To' section for more details). This is useful in conjunction with hand-editing the source files to speed up debugging.

The *make* utility works in the following way. To create an executable from a number of fortran files, the fortran files are compiled to create .o object files which are then linked to create the executable. To recreate the executable after just one or two fortran files are changed, only those altered files need to be recompiled before the executable is linked. In the UM compile system, the *make* utility works by comparing the dates of the files to determine what work needs to be done. So if one fortran file is changed, *make* sees that the .f file is newer than its .o file and so recompiles it. Then it will find that the .o file is newer than the executable and so it will relink the executable.

3.3.3 Browsing Compile Output and Error Messages

Order of information in job output files

When looking for information from the compile system first look at the script output from qsprelim and qsmain at the top of the model output file for messages as to whether the compile script processing and *make* steps completed successfully.

If not, then the detailed error messages can be found further down in the output. Search on the string **%QSMNCOMPILE** for the start of the output from the pre-processing compile scripts and on **%MAKE**. for the output from the *make* step itself (but note that these strings will also show up in the echo statements in SCRIPT if the job asks for a copy of SCRIPT to be listed in the output).

See the troubleshooting section of the User Guide for a description of the types of output in the model output file and the order they appear in.

See also the later section in this document giving an example of the output that appears after %QSMNCOMPILE and %MAKE, and an in-depth explanation of what all the messages and codes mean.

Note that at vn4.4 the reconfiguration compile output still appears in full near the top of the output file. At vn4.5 this will be moved further down the output file.

Understanding ERROR or WARNING messages

A run may fail during compilation. The message output from qsmain is.

```
qsmain(934): ***  Make has failed. Exiting
```

The actual error will be found near the bottom of the *make* output after %MAKE in the model output file. This should give an indication why it has failed. The most likely reason *make* might fail is due to a compilation error, but it can also fail for system reasons such as because there is not enough memory available. The error message may not be the last message in the output if there are several compilation processes running in parallel (due to NPROC being greater than 1) since *make* waits for the other current compilations to complete once an error has been detected.

Upon successful completion of the *make* command there may still be WARNING messages in the output. Often these can be ignored but sometimes they are significant. See the next few sections for some examples.

On Cray MPP machines the use of "-Dpermok=yes" in the load line allows the run executable to be created even if WARNING messages have been generated. This is necessary because there are always parts of the model not being used by a configuration for which the top routine calls are still there but the code is not included.

Missing externals

The main kind of WARNING, a T3E example of which is shown below, sometimes leads to failures in the model with missing externals.

e.g.

```
cld-404 cld: WARNING
The symbol ‘CHK_LOOK_BOUNDA’ referenced in relocatable object
‘inbound1.o:IN_BOUND’ is not defined.
```

It is worthwhile, if the model fails in this way after starting running, checking the relevant .f and .o files exist in the user compile directory and are not empty. (However, normally the compile system cleans up empty files caused by a system failure part way through a model compilation and this causes those routines to be recompiled.)

Occasionally when a sequence of changes are made and there are also system problems, a routine may be missing from the makefile when it is generated and hence is missing from the executable when made. This can be rectified by editing the Makefile manually and using SKIP_SRC_EXTRACT=true (see 'How to' section) or by deleting all the files in the compile directory and recompiling again to get a clean compile.

Vector directives (Cray MPP only)

The UM when running on Cray MPP machines gives the following type of warnings:

```
CMIC$ DO ALL VECTOR SHARED(U_POINTS, ROW_LENGTH, U1, SU10, SV10, U_FIELD
      ^
cf90-801 f90: WARNING IMPL_CAL, File=implca2c.f, Line=1491, Column=7
  Unsupported compiler directive.
```

These kinds of warnings can be ignored. They arise because because compiler directives exist in the code to enable the model to run on a vector parallel machine (e.g. an SGI/Cray C90).

Forcing *make* to continue through compilation errors

USE WITH CAUTION
By default *make* stops during compilation if there has been an error. An error code is returned and the run terminated. It is possible to force *make* to continue compilation after there has been an error. This means all errors can be identified in a single pass and object code will be created for decks which do not have any errors.

This is useful for debugging but unsafe in general because a success code is now returned from *make* regardless of whether there has been an error. It appears to *make* that compilation has been successful and the model run would be started after calling *make* in this way even though there were errors.

To force *make* to carry on past compile errors change the call from ’make’ to ’make -k’ in qsmain using a script mod and run with STEP=0 in the SUBMIT job library file (or set job to ’compile only’ in the UMUI). At the UK Met. Office this change is available as fix mod $UMDIR/vn4.4/mods/fixes/fkr4u404 which contains:

```
*IDENT FKR4U404
*DECLARE qsmain
*D gex3u405.24
  make -k >>$OUTPUT 2>&1
```

3.3.4 Output Messages in more detail

The following is selected output from the compilation part of a UM run. This is not totally realistic as some of the messages will not appear under some conditions but is representative of what will generally appear. The first field indicates the script that is giving the message. A numeric value in parenthesis is used for system timing and indicates the number of seconds the current shell (ie UM run) has been in existance for. The information is split into eleven discrete sections. An explanation of each of these sections follows.

```
qsmncompile(191): ***   Compile vars diff information.
qsmncompile(193): ***     System differences.
qsmncompile(194): ***   End of compile vars info.

qsmncompile(222): ***   Removing symlinks in compilation dir.
qsmncompile(401): ***   Finished removing symlinks

qsmncompile(401): ***   Removing zero length files
qsmncompile(409): ***   Finished removing files

qsmncompile(414): ***   Removing previously modified files
qsmncompile(414): ***     This looks new. No files to remove
qsmncompile(414): ***   Finished removing modified files

qsmncompile(414): ***   STARTING control source comparison
qsmncompile(416): ***     abcalc1       (Same)  (F)
qsmncompile(416): ***     ac_ctl1       (Same)  (F)
qsmncompile(338): ***     chemctl1      (New)   (F)
qsmncompile(339): ***     cldctl1       (New)   (F)
qsmncompile(341): ***     conv_ct1      (New)   (F)
qsmncompile(351): ***     divcal1a      (Diff)  (F)
qsmncompile(352): ***     dosums1       (Diff)  (F)
qsmncompile(494): ***     st_dia11      (Same)  (F)     (Opts:new)
qsmncompile(518): ***     zonmctl1      (Same)  (F)
qsmncompile(518): ***   FINISHED control source comparison

qsmncompile(518): ***   Symlink-ing pre-compiled code
qsmncompile(576): ***     Linking /u/um1/vn4.4/obj/a01_1b/Gp/NORMAL
qsmncompile(577): ***       ftsa1a      (F:ok)  (O:ok)
qsmncompile(578): ***       solang1a    (F:ok)  (O:ok)
qsmncompile(579): ***       solpos1a    (F:ok)  (O:ok)
qsmncompile(579): ***       swclop1a    (F:ok)  (O:ok)
qsmncompile(584): ***     Linking /u/um1/vn4.4/obj/a02_1b/Gp/NORMAL
qsmncompile(584): ***       lwcld1a     (F:ok)  (O:ok)
qsmncompile(584): ***       lwdcsf1a    (F:ok)  (O:ok)
qsmncompile(657): ***     Linking /u/um1/vn4.4/obj/a15_1a/G1G3/NORMAL
qsmncompile(658): ***       calcp21a    (F:ok)  (O:ok)
qsmncompile(659): ***       calcpv1a    (F:ok)  (O:ok)
qsmncompile(660): ***       cat1a       (F:ok)  (O:ok)
qsmncompile(661): ***       dthdp1a     (F:ok)  (O:ok)
qsmncompile(662): ***       dyndia1a    (F:ok)  (O:com) (Opts:ok)
qsmncompile(676): ***       omegdi1a    (F:ok)  (O:ok)
qsmncompile(677): ***       pvpin1a     (F:ok)  (O:ok)
qsmncompile(678): ***       pvthin1a    (F:ok)  (O:ok)
qsmncompile(697): ***     Linking /u/um1/vn4.4/obj/c95_2a/G1GnP4/NORMAL
qsmncompile(697): ***       portio2a    (C:ok)  (O:ok)
qsmncompile(813): ***   Finished symlink-ing object code

qsmncompile(813): ***   Generating sed script
qsmncompile(814): ***   Sed script was generated okay
qsmncompile(814): ***   Running sed to create Makefile
qsmncompile(815): ***   Finished sed

qsmain(819): ***        Running make

%MAKE
        f90  -Oaggress -Oscalar3  -c abcalc1.f
        f90  -Oaggress -Oscalar3  -c ac_ctl1.f
        f90  -Oaggress -Oscalar3  -c acumps1.f
        f90  -Oaggress -Oscalar3  -c addrchk1.f
        f90   -c st_dia11.f
        f90   -c dyndia1a.f
        f90 -Wl"-Dpermok=yes;streams=on" -lgcom1m1s4x1_shmemnam
            -l_vect -lgrib -L. -L/u/um1/gcom/rel_1m1s4x1
            -o /mydir/me/aaxid/aaxid ac_ctl1.o acumps1.o \
            addres1.o addrln1.o atmdyn1.o atmphy1.o atmstep1.o \
            bl_ctl1.o boundva1.o chemctl1.o cldctl1.o \
            conv_ct1.o dervsize.o diag3a.o
.......etc..

cld-404 cld: WARNING
The symbol ‘AC’ referenced in relocatable object ‘ac_ctl1.o:AC_CTL’
is not defined.
cld-404 cld: WARNING
The symbol ‘STOCGT’ referenced in relocatable object ‘stwork1a.o:STWORK’
is not defined.
cld-404 cld: WARNING
  etc

qsmain(934): ***        Completed make
```

1. Extraction nupdate step to COMPDIR. All decks that are either specified in the cdecks file or modified by mods directly or indirectly, via comdecks, included in the UM configuration are extracted.

2. Rename extracted decks in COMPDIR. The decks are stored in the UMPL as uppercase without file type suffixes.

3. Remove any previous symlinks in UCOMPDIR. The symlinks are removed every time to enable decks that are newly modified to be moved into place.

4. Remove any zero length files in UCOMPDIR. A precaution. Some decks from previous compilations may have no effective source code or may not have produced complete object files due to compilation errors.

5. Remove any decks that are no longer modified wrt previous run, if necessary. This is done so that files can be symlinked in the symlink step.

6. Compare source in COMPDIR with any source in UCOMPDIR. Move it in if different from previous run code or new, leave it intact in UCOMPDIR if there has been no code change since previous run. Extract by-deck compiler options for later inclusion in Makefile. The following messages can occur under different conditions:

   (Same) This is a secondary run. The code for the deck found in the UCOMPDIR is the same as that extracted. The source is not updated. Re-compilation will not be needed, unless the compiler options have changed since the last run, for this deck.

   (New) This is either a primary run and no decks exist in the UCOMPDIR directory at all or a secondary run into which a new deck (proabably from a user mod) is being added.

   (Diff) A secondary run which has changed mods wrt to the previous run. The deck that has been extracted from the UMPL (including mod) into UCOMPDIR is different from the previous version in UCOMPDIR. The new version is moved into UCOMPDIR. This will force make(1) to re-compile at a later stage.

   (F) The deck has been identified as Fortran source code.

   (C) The deck has been identified as C source code.

   (Opts:new) This deck has specific compiler options defined in the UCOMPVARS file. These will be used instead of the default compiler options.

7. Produce sym-links to required pre-compiled source code and object files. Extract by-deck compiler options for later inclusion in Makefile, if necessary. Some decks exist in more than one section that may be required simultaneously. These decks contain identical code and are compiled in the same way, so the code and object from the first occurance of the deck are included. Subsequent occurances are skipped. If the relevent section-def combination is not found in the UM system pre-built directory and BUILDALL is set to "true" then script qsconf is called to extract the relevent code in to BUILDALL_DIR and then make(1) used to compile it. The files are then linked in the same way as for system pre-built code. The following messages can occur during this stage.

Only one of the following messages should exist for each deck.

(F:ok)	The Fortran source for the section-def combination has been linked into the UCOMPDIR directory from the system pre-built source directories.
(F:src)	The Fortran source for this deck already exists in the UCOMPDIR directory. This is because the deck has been modified and extracted by the previous nupdate extract step in the current user run.
(F:Sec)	The Fortran source for this deck already exists in the UCOMPDIR directory. This is because the deck was symlinked in a previous section-def combination during the current user run.
(F:OSN)	"Outside System, New" - The Fortran source for this deck already exists in the UCOMPDIR directory. This is because the user has removed a symlink file and created a "real" file. The deck will gain a "rule" in the Makefile with compiler options from the UCOMPVARS file. NOTE: this source deck is completely under the users control. It will not be deleted by the UM build system. The user must delete it (and it's object file) directly if the system default is to be used.
(F:OSO)	"Outside System, Old" - The deck was in a previous section-def combination and has been labelled as an OSN file already. See previous item.
(C:ok)	The C source for the section-def combination has been linked into the UCOMPDIR directory from the system pre-built source directories. Note: only the first source-object pair encountered in the pre-built section-def combinations is linked to.
(C:src)	The C source for this deck already exists in the UCOMPDIR directory. This is because the deck has been modified and extracted by the previous nupdate extract step in the current user run.
(C:Sec)	The C source for this deck already exists in the UCOMPDIR directory. This is because the deck was symlinked in a previous section def combination during the current user run.
(C:OSN)	"Outside System, New" - The C source for this deck already exists in the UCOMPDIR directory. This is because the user has removed a symlink file and created a "real" file. The deck will gain an "rule" in the Makefile with compiler options from the UCOMPVARS file. NOTE: this source deck is completely under the users control. It will not be deleted by the UM build system. The user must delete it (and it's object file) directly if the system default is to be used.
(C:OSO)	"Outside System, Old" - The deck was in a previous section and has been labelled as an OSN file already. See previous item.

Only one of the following messages should exist for each deck.

(O:ok)	The object file for the section-def-compile combination has been sym-linked into the UCOMPDIR directory from the system pre-built source directories. Note: only the first source-object pair encountered in the pre-built section-def combinations is linked to.
(O:rm/ok)	An object file for the deck already exists in the UCOMPDIR directory from a previous run. This is removed and a sym-link into the UCOMPDIR from the section-def-compile combination object created.
(O:rm)	The object already exists in UCOMPDIR, but needs re-compiling as the entry in UCOMPVARS has been changed, thus it is removed.

If the first of the following messages exists, then the second message may or may not exist, depending whether or not the UCOMPVARS file has changed.

(O:com)	An entry in the makefile wil be generated so that the deck can be compiled.
(Opts:ok)	This deck has specific compiler options defined in the UCOMPVARS file. These will be used instead of the default compiler options.

If these messages are given then the above messages will not be given.

WARNING	The section-def combination that has been requested ahs not been pre-built by the UM system team.
BUILDALL	The section-def will be extracted and built in BUILDALL_DIR.

8. Generate sed script then Makefile. The sed script contains information on how information on compilation of decks should be parsed into the Makefile. The Makefile contains rules that indicate what needs to be compiled/linked and dependencies indicating which files depend on other files. This enables object files to remain up to date wrt to source files.

9. Run make(1) to generate any required object files and link the object files into a binary executable. The make(1) step compiles decks that have no object, recompiles decks that have changed since their object was created and leaves objects that are older than their source (ie the source has not changed). You can see here exactly how each object is being compiled.

3.3.5 Compile System Variables

Set in UMUI

Variable Name	Typical Setting	UMUI panel	Description
UCOMPDIR	$DATAW/compile_$RUNID	subindep_FileDir	Directory in which compilation will take place. Must be unique to a run identifier.
URECONDIR	$DATAW/recon_$RUNID	subindep_FileDir	Directory below which reconfiguration compilation will take place. Must be unique to a run identifier.
REBUILD_DIR	$DATAW/rebuild_$RUNID	subindep_FileDir	Directory below which compilations of section-version combinations that are not held centrally are done. Must be unique to a run identifier.
NPROC	5	subindep_Compile	Number of compilation processes that run in parallel under *make* (not applicable on all machines). If unset uses version-wide default in setglobalvars.

Hand-edit in job library

Variable Name	Typical Setting	Description
SKIP_SRC_EXTRACT	false	Set to true in SUBMIT to hand-edit .f files and run *make* directly.
BUILDSECT	false	Set to true in SUBMIT to allow section-version DEF combinations not prebuilt centrally to be built within run.
SKIP_COMPILE_TAR	false	(UK Met. Office only). Set to true in SUBMIT to prevent tar/gzip being used on compile files.

3.3.6 Overriding compiler options

The options used to compile source code are supplied in the central file $UMDIR/vn$VN/source/compile_vars for each version. Overrides to the options in this file can be specified in one or more compile override files.

Note that compile override files are only used by the compile system when it does the script pre-processing steps when SKIP_SRC_EXTRACT is false. When SKIP_SRC_EXTRACT is true it uses whatever compile options are specified in the Makefile.

Step by step guide to using overrides

Below are the steps you need to go through to specify your own compile options:

1. Create a directory on the machine where you are running the UM to hold compile override files e.g. $HOME/comp_override.

2. Look at the file $UMDIR/vn$VN/source/compile_vars to see the format of lines. The keywords (@fort, @load etc) and tags (FCOM_OPTS and LOAD_OPTS etc) are case-sensitive. The amount of white space around options does not matter.

   Note that apart from FCOM_CMD, CCOM_CMD and LOAD_CMD, which are used to specify the Fortran compiler, C compiler and link/loader to be used, other options are just concatenated together in each type. For example on a T3E f90 platform, all the options specified on the FCOM_OPTS, FCOM_ENABLE, FCOM_DISABLE, FCOM_OPTIM and FCOM_ROPTS lines are just concatenated together by the compile system in a run. Having more than one identifier just makes it easy to specify related options together if preferred. New categories will also be recognised provided they start with FCOM_. (At present there is no method for continuing options onto a second line with any of these identifiers. For changes to compile_vars for the next version of the UM there is a line length limit of 72 characters on these lines due to portable nupdate limitations. However this limit does not apply to lines within user compile override files.)

   All options should be specified after the '=' exactly as they would be on the compile line (such as in a f90 command on the T3E) with no need for extra quotes.

3. Any line in the compile_vars file can be overridden. Default lines are those starting with @ eg. @fort, @load. Be careful if you change a compiler default line (@fort) as this will recompile THE WHOLE OF your model code with this new default. Also large numbers of single-deck compile options will slow down the scripts.

   When overriding load options remember to include the standard default options from compile_vars as well as additional options unless you definitely don't want to use the standard defaults.

4. Create a file or a number of files to hold the lines which are being overriden. These override files must then be made available to the UM scripts via the UMUI in window **subindep_Compile_User**
   ```
   Sub-Model Independent
   -> Compilation and Modification
   --> User-defined compile option overrides
   ```
   or by hand-editing job library file COMP_OPTS. The scripts do the work to include the overrides in the model run.

   If a particular mod requires special compile options a compile override file can be used in conjunction with that mod. This is a convenient way for a run owner to organise groups of changes and associated compile options, but it is up to that person to keep track of which compile override files go with which mods (there is no link within the UMUI).

   Comments can be included in compile overide files using a # in the first column.

Some examples of compiler overrides

The first set of examples show how the global defaults, i.e. those applicable to the whole model, might be overriden.

- Set the f90 -g (debug) and -ei (enable i) options.

  Provide an override file with the line,
  ```
  @fort FCOM_OPTS=-g -ei
  ```
  to replace the default line "@fort FCOM_OPTS=" (remembering that this recompiles all routines including precompiled ones)

- Force a recompile of the entire model without changing compile options.

  Provide an override file with the line,
  ```
  @fort FCOM_ENABLE=-eq
  ```
  to replace the default line "@fort FCOM_ENABLE=" (the -q option is already enabled but not specified in the compile_vars file so having the effect that all routines including precompiled ones are recompiled but with null effect.) This can be useful for recompiling with a different version of the compiler to that used for the prebuilt code.

- Use load libraries "-lgrib -lgcom_pvm_11 -lconv"

  Provide an override file with the line,
  ```
  @load  LOAD_LIBS=-lgrib -lgcom_pvm_11 -l_conv
  ```
  to replace the default line "@load LOAD_LIBS=-lgrib -lgcom_shmnam_13 -lvect"

As you can see from browsing the compile_vars file it is possible to apply compiler options to individual decks.

Given below are some examples which show how to override existing decks specific options and how to add further compiler options for a given deck.

- Set f90 optimisation option "aggress" in deck ADVCTL1A

  Provide an override file with the line,
  ```
  ADVCTL1A  FCOM_OPTIM=-O aggress
  ```

- Use machine default compile optimisation for deck HORINF1A

  Provide an override file with the line,
  ```
  HORINF1A  FCOM_OPTIM=
  ```
  to replace the default line "HORINF1A FCOM_OPTIM=-Ounroll2 -Oaggress -Oscalar3"

Applying overrides to the reconfiguration

These work in exactly the same way as the model compile overrides but may be different overrides and have a separate list of override files.

3.3.7 Miscellaneous "How to ..." questions

1. **Hand-edit source files and just rerun *make* in a run.**

   It is well worth becoming familiar with this way of speeding up debugging since it will save many hours of time compared to running the full compile system BUT it is only suitable for making small non-permanent changes (for example adding print statements) since mods cannot be generated from the changes. It involves making alterations to copies of the actual fortran code kept in the compile directory of your job rather than writing mods. It avoids the significant overhead in time of extracting all the relevant decks, applying mods and *DEFs, expanding comdecks and recompiling many decks. Only those decks that have been changed will be recompiled before linking to produce the executable. This method does not apply to the reconfiguration.

The job must have been compiled in the normal way first. After the run has compiled, all the relevant source and object code is left in a compile directory as specified in the UMUI (typically $HOME/$RUNID/compile_$RUNID). On typing ”ls -l” in this directory, a list of source (.f and .c) and object (.o) files or symbolic links to such files will be seen. (Follow the instructions in the later section on how to get at .f files bundled into tar files (UK Met. Office only).

Files without symbolic links are generally control code, ocean decks and modified decks. Many decks, however, have been precompiled a number of times with different combinations of *DEFs. The source and object code for these decks are held centrally; unmodified decks are thus included by making symbolic links to these files. See the warnings below before attempting to edit linked files.

Make alterations to copies of the actual Fortran code kept in the compile directory of your job rather than writing mods. Only the .f files that have been changed will be recompiled.

By using the following hand edit, the source extraction and checking of which decks/compile options have changed can be avoided and the files in the compile directory will be reused:

In SUBMIT, hand edit SKIP_SRC_EXTRACT from ’false’ to ’true’ (where SUBMIT is one of the job library files produced by the UMUI).

Then resubmit the run as normal.

WARNINGS:

This system cannot be used for editing the contents of comdecks. These are already included inline in the appropriate routines. Another consequence of this is that the edited decks cannot be used with nmodex to create a modset.

Many of the files in the compile directories are, in fact, links to the libraries; these cannot be edited. Before editing a deck find out whether it is linked:

ls -l windmx1a.f

Linked files are shown as:
`windmx1a.f -> /u/um1/vn4.4/src/a15_1a/A1G1G3/NORMAL/../windmx1a.f`

If you need to edit such a deck, make a note of the path to the file to which the link is pointing and delete the links to the .f and .o files. Then copy the .f file into your compile directory. For example:

```
cd $DATAW/compile_$RUNID
ls -l windmx1a.f  # Note path
rm windmx1a.f
cp /u/um1/vn4.4/src/a15_1a/A1G1G3/NORMAL/../windmx1a.f .
```

It is important that the file is copied from the correct directory since there will be a number of alternative directories containing a deck but only the file to which the link is pointing will have had the correct *DEFS applied to it.

Additionally, the system will not know how to compile such decks since previously it just picked up the precompiled .o file. Before running with SKIP_SRC_EXTRACT you will need to either:

Compile the deck by hand:

f90 -c -Oaggress -Oscalar3 windmx1a

Or edit the Makefile to add compile instructions. eg, near the top add the line:

FFLAGS = -c -Oaggress -Oscalar3

which sets the default compile options for any FORTRAN files that do not have options listed.

Another warning: If you do handedit a linked deck and then subsequently do a normal full recompilation (ie. set SKIP_SRC_EXTRACT=false again) then your edited deck will NOT be replaced with the linked deck unless you have now written a mod for it. The compile system will assume that you want to retain your edited version. To return to the original version, delete the edited file. It is therefore advisable to delete the contents of the compile directory after finishing a piece of work that involves this type of alteration otherwise the altered deck will remain to affect subsequent runs in the future long after you forget that you carried out such edits.

2. **Run without compiling from an existing executable.**

   Set **STEP=4** in the SUBMIT job library file on the system where you run the UMUI.

3. **Change just load options.**

   It is possible to change the load options using a compile override file, for example to use different libraries. However if the code has already been compiled and you just want to change the load options or pick up updated libraries you will end up doing alot of unnecessary script pre-processing work if you put the run back in with an added compile override file.

   Therefore it is more efficient to run with SKIP_SRC_EXTRACT set to true (see section on this below) which will just run the *make* step when the run is put back in.

   Note that *make* will not detect an updated library or load options changed by handediting the makefile. Therefore to force the makefile to run the load step you need to delete or move the model executable before running with SKIP_SRC_EXTRACT.

   It is also possible to run a load step interactively outside a run by doing:

   - Stoprun the run
   - Move the executable to a different name
   - Go into your compile directory eg. for run aaumf:
     ```
     cd ~userid/aaumf/Compile
     ```
   - Hand-edit the line similar to the following one in 'Makefile' to include the new library name if necessary:
     ```
     LOADOPTS = -Wl"-Dpermok=yes;streams=on" -lgrib
                -lgcom1m1s4x1_shmemnam -l_vect -L.
     ```
   - Type 'make'
   - Resubmit job

4. **Recompile the whole model.**

   Changing a default Fortran compile option eg. “@fort FCOM_OPTS” will cause all the decks that your model uses to be recompiled. eg. on a T3E machine if the default is “@fort FCOM_OPTS=” then it could be changed to
   ```
   @fort  FCOM_OPTS=-ea
   ```
   This aborts compilation on finding the first error so is harmless when used with code that already compiles.

5. **Use a combination of *DEFs that hasn't been prebuilt for a section version.**

   Set the BUILDSECT variable in the SUBMIT job library file to true. Any DEF combinations which cannot be found will be built during the model run. The default, BUILDSECT=false, simply flags the missing section def combinations and causes the model to fail in the compile script pre-processing stage.

   Please report any commonly-used combinations that appear to be missing from the pre-built code (at the UK Met. Office only).

6. **Use a new *DEF in the UM system.**

   Set the BUILDSECT variable in the SUBMIT job library file to true. Make a copy of the obj_xref file defining a DEF_SWITCH for the new DEF at the top and adding identifier associated with the DEF to the 'DEFS' line of the relevant sections, and hand-edit your SCRIPT file to use this in your run (script mods to obj_xref do not work at present). Do not add anything to the 'BUILD hostname' lines. All the relevant sections will be recompiled with the new DEF. (A DEF_SWITCH is of the form identifier:defname e.g. A1:ATMOS.)

7. **Untar compile files to edit .f files (UK Met. Office only)**

   At vn4.4 all compile directory (mainly .f and .o) files are placed in a single tar file called $DATAW/comp_$RUNID.tar at the end of each compile session and are unpacked at the beginning of the next run that compiles that experiment. This enables data migration on the T3E to work more efficiently in migrating one file rather than hundreds of files each time. The tar file is also compressed using gzip to save space.

   The code is controlled by a variable called SKIP_TAR_COMPDIR, which may be handedited to *true* in the SUBMIT job library file to avoid the 'tar's being done. The default is *false*.

   If 'tar's are turned on, it first checks whether there are any .f files in the model compile directory. If there are it assumes that unpacking the tar file is not necessary and does not attempt to uncompress or untar.

   At the end of the run it creates the tar file, compresses it and then deletes all the files in the Compile directory. Each time a tar/gzip is done a README file is created in the $DATAW directory giving the commands required to manually untar files and edit them eg:

```
To unpack tar file:
cd $HOME/abaod/Compile # compile directory
gzip -d $HOME/abaod/comp_abaod.tar.gz # unzip tar file
tar -xf $HOME/abaod/comp_abaod.tar # extract all files
or
tar -xf $HOME/abaod/comp_abaod.tar atmstep1.f  # extract
                                               # individual
                                               # file(s)
To repack tar file:
cd $HOME/abaod/Compile # compile directory
tar -cf $HOME/abaod/comp_abaod.tar *  # tar whole directory
or
tar -uf $HOME/abaod/comp_abaod.tar atmstep1.f # update
                                              # individual
                                              # file(s) in
                                              # tar file
gzip $HOME/abaod/comp_abaod.tar # zip tar file
rm * # remove .f, .c files etc
```

   In general it is recommended that people use the tar system in case their compile files get migrated at some point in the future. However it adds about 5 minutes to a compile to untar, tar and remove all the compile files so when debugging compile errors by editing .f files using SKIP_SRC_EXTRACT=true it is more productive to turn off the use of tar files by also setting SKIP_TAR_COMPDIR=true.

   The reconfiguration files are 'tar'ed up in the same way into a file called $DATAW/recon_$RUNID.tar, with similar instructions being placed in the README file.

   Note that if you manually unpack and unzip a tar file using the commands in your README file to hand-edit .f files you need to manually do the reverse process of packing them up before resubmitting the run.

3.3.8 Building a model section or small executable

Building personal copies of the small executables

To build one or more small execs with a mod:

- Set UM version:

  ```
  export VN=4.4
  ```

- Copy $UMDIR/vn$VN/scripts/Install/configure_execs to $HOME
- Set environment variable:

  ```
  export TOPDIR=$HOME
  ```

  in either in configure_execs before it is used or before calling the script.
- Alter line:

  ```
  for exec in ‘cat $EXECXREF| awk ’{print $1}’|\
  egrep -v ’#’ | sort -u | flower‘
  ```

  to (for example):

  ```
  for exec in "qxcombine qxhistreport qxhistreset \
  qxpickup qxsetup"
  ```

  where you just list the small execs you want to build.
- To add in a Fortran mod change:

  ```
  $UMDIR/vn$VN/scripts/qsconf \\
  -outdir ‘dirname $SRCDIR‘ \\
  -execs $exec 1> $OUTDIR/$exec.out 2>&1
  ```

  By adding 1 extra line with the -femod option to:

  ```
  $UMDIR/vn$VN/scripts/qsconf \\
      -outdir ‘dirname $SRCDIR‘ \\
      -femod $HOME/mods/mymodname \\
      -execs $exec 1> $OUTDIR/$exec.out 2>&1
  ```

  using your mod path instead of $HOME/mods/mymodname.

  Use -cemod for C mods (Do ’$UMDIR/vn$VN/scripts/qsconf -help’ for more information on options).
- On Cray machines only: Edit the QSUB time, memory and queuename if appropriate. This will be used as a separate job for each small exec.
- Run your version of configure_execs:

  ```
  cd $HOME
  ./configure_execs
  ```

This will create a directory called something like build_execs_010897_1513 containing a script to build each small exec which it will submit as QSUB jobs (on Cray machines).

- The code will be placed under a directory called exec_build with a subdirectory for each small exec. Each executable then needs to be moved to the directory where it will be used eg:

```
mkdir $HOME/exec
mv $HOME/exec_build/qxcombine_dir/qxcombine $HOME/exec
mv $HOME/exec_build/qxhistreport_dir/qxhistreport $HOME/exec
mv $HOME/exec_build/qxpickup_dir/qxpickup $HOME/exec
mv $HOME/exec_build/qxhistreset_dir/qxhistreset $HOME/exec
mv $HOME/exec_build/qxsetup_dir/qxsetup $HOME/exec
```

Using personal copies of the small executables within a model run

- Select the "Using test small executables" option via the UMUI in:
  `Sub-Model Independent`
  `-> Configuration and Control`
- Hand edit the variable EXECN in SCRIPT to make the path of your directory containing the test executables available e.g. `EXECN=~username/exec`

3.4 MPP Capability

Author: Paul Burton, Last Updated: 29 June 98

3.4.1 Overview of Computing Platforms

Until the early 1990's, classic supercomputer design was based around the concept of using a small number of specially designed and manufactured powerful processors, which would all access a single shared memory. The compilers on such machines were capable of automatically adding extra instructions to the code, allowing it to efficiently run in parallel over a small number of processors. The code itself could be written as if it was just to be run on a traditional single processor computer.

However, the limitations of this architecture were beginning to show. The shared memory meant that there was a bottleneck between processors and memory as more processors were utilised, so limiting the parallel performance. Also, the cost of the proprietary processors limited the feasibility of using a large number of them.

The alternative to this approach is the Massively Parallel Processor (MPP) supercomputer. This typically uses a large number of less powerful, but commodity (ie. cheaper) processors, each having its own local memory. In order that a processor can access memory belonging to another processor, the processors are interconnected using a high performance network of some kind. Although each processor may be less powerful than the traditional supercomputer processor, the large number of them working together in parallel can provide a significant performance gain.

This increase in performance was realised by the Met. Office in 1997, when it replaced its Cray C90 (a 16 processor, parallel vector processor) by a Cray T3E, containing 880 Dec Alpha RISC processors. The T3E gives around a five times increase in performance over the C90.

However, compilers are not yet capable of automatically making a traditional single processor code (as was run on the shared memory computers) parallelise efficiently. It is therefore necessary for the parallelism to be explictly coded if programs are to be run efficiently on MPP machines.

3.4.2 Overview of the MPP UM

The UM has had the necessary explicit parallelisation code added to allow it to run on a range of MPP platforms.

The basic principle used in the MPP mode of the UM is horizontal domain decomposition. This means the model grid-point space is divided up into subdomains, each containing a complete set of vertical levels but only a rectangular latitude-longitude horizontal subsection. Each processor is then reponsible for one of these subdomains, and contains all the data for this subdomain in its own local memory. Regular inter-procesor communication is required so that information can be transferred between neighbouring subdomains. This is achieved by extending the local subdomain by a number (usually just one) of rows or columns in each dimension, to form a "halo" around the data. This halo region holds data which is communicated from adjacent processors, so allowing each processor to have access to neighbouring processors edge conditions.

The atmosphere and ocean models can have different decompositions. The atmosphere model has a user-definable two dimension decomposition (that is, the data is decomposed in both latitude and longitude), while the ocean model is currently restricted to a one dimensional decomposition, over latitude.

When a coupled model is run, each sub-model is run in turn on the same set of processors. Therefore, this requires the total number of processors used in the atmosphere decompostion to be equal to the total number of processors used in the ocean decomposition.

Communication Libraries

The most fundamental part of any MPP code is the communication between processors. This can also be the most non-portable part because there are many different communication libraries available. Some, such as PVM and MPI are recognised standards, and reasonably portable, while others, such as Cray SHMEM are only available on particular platforms, but provide very high levels of performance.

To enable both portability and high performance, the UM uses an interface library called GCOM, which can be compiled to use many different communications libraries, while leaving the UM code undisturbed. For example, on the Met. Office's Cray T3E, we use the Cray SHMEM version of GCOM, while on a network of workstations we can use the PVM or MPI versions of GCOM. The UM code itself remains unchanged, just requiring a different version of the GCOM library to be linked.

MPP Code Management

The MPP code is almost all contained within existing code, and is automatically selected when nupdate (see section 3.1) *DEF MPP is selected. A small number of subroutines have been completely rewritten for MPP so that they can run efficiently, such subroutines will be inside a nupdate *DEF MPP, and possibly as part of an MPP specific section/version (see below).

There is also some T3E specific code in the UM, which has been written to maximise the performance of the code on the Met. Office's Cray T3E. Code such as this will always be enclosed in a nupdate *DEF T3E, with a portable alternative available.

The portable, single processor code can still be run if both nupdate *DEF MPP and *DEF T3E are not applied. This code will still run efficiently on both shared memory parallel and vector machines such as the Cray C90.

3.4.3 Running the MPP code

Only a small number of changes need to be made in the UMUI to convert an experiment to work on an MPP machine. The following panels will need to be updated:

1. Window: **subindep_Target_Machine**
   ```
   User Information and Target Machine
   -> Target Machine
   ```
   - Select "Machine type" as "Distributed-Memory Parallel (eg T3E or VPP)"
   - Fill in the boxes defining the number of processors in the North-South / East-West directions (or just the North-South direction for the ocean model). Where there is a two dimensional decomposition, some experimentation may be required to find the arrangement giving optimium performance (although the UM will run correctly with any arrangement). Commonly, it is resonable to set the number of processors in the East-West and North-South directions to be roughly equal (obviously, they can only be exactly equal if the total number of processors is a square number).

2. Window: **subindep_Section_Misc3**
   ```
   Sub-Model Independent
   -> Sub-model Independent Section Options
   --> Miscellaneous Sections
   ---> Panel 3
   ```
   - Select "Choose version of section 96" as "1A For those on general MPP machines." or "1B For faster running on CRAY T3E only."

- Select “Choose version of section 97 (Timer)” as “3A Advanced UM timer (suitable for MPP).” or “4A Dummy timer code if you are running without timer.”

3. Window: **atmos_Science_Section_Advec**
```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
---> Primary field advection
```
   - Select “Choose version” as “⟨ 1C ⟩ MPP version of 1B” or “⟨ 1E ⟩ Optimised equivalent of 1C. Does not bit reproduce.” (see next section for more details on the optimised version)

Once these changes have been made, it is recommended that a ”Check Setup” is performed, which will inform you if you have any inconsistent options set in the UMUI.

3.4.4 Maximising performance

Some sections of the UM have alternative versions for MPP optimised code, and possibly code optimised specifically for the Cray T3E. These versions can be selected to improve the runtime of the model, but it should be recognized that so doing will almost invariabaly change the numerical results produced by the model. The following sections have MPP/T3E optimisations associated with them:

1. Window: **atmos_Science_Section_LSRain**
```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
---> Large Scale Precipitation
```
   - If the “2D” version of the section is selected, this can be changed to “2E”. This includes optimisations primarily added to benefit the Cray T3E, but which may also improve performance on other cache based machines. The “2E” version of the section can produce different numerical results from the “2D” version.

2. Window: **atmos_Science_Section_Conv**
```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
---> Convection
```
   - If the “3A” version of the section is selected, this can be changed to “3C”. This includes optimisations primarily added to benefit the Cray T3E, but which may also improve performance on other cache based machines. The “3C” version of the section can produce different numerical results from the “3A” version.
   - The convection scheme contains code to load balance the convection calculations over all processors. The load balancing can be controlled by changing the “Number of segments” value. A large number will result in good load balance, but at the expense of a large dynamic memory allocation overhead within the convection calculations. It has been found that the best results are to be gained from using between 3-5 segments. Experimentation will be required (using the timer output) to find the optimum setting for any particular configuration of the UM.

3. Window: **atmos_Science_Section_VDiff**
   ```
   Atmosphere
   -> Scientific Parameters and Sections
   --> Section by section choices
   ---> Vertical Diffusion
   ```
   - If the “1A” version of the section is selected, this can be changed to “1B”. This includes optimisations primarily added to benefit the Cray T3E, but which may also improve performance on other cache based machines. The “1B” version of the section can produce different numerical results from the “1A” version.

4. Window: **atmos_Science_Section_Adj**
   ```
   Atmosphere
   -> Scientific Parameters and Sections
   --> Section by section choices
   ---> Dynamical Adjustment
   ```
   - If the “1A” version of the section is selected, this can be changed to “1C”. This includes optimisations primarily added to benefit the Cray T3E, but which may also improve performance on other cache based machines. The “1C” version of the section can produce different numerical results from the “1A” version.

5. Window: **atmos_Science_Section_Advec**
   ```
   Atmosphere
   -> Scientific Parameters and Sections
   --> Section by section choices
   ---> Primary Field Advection
   ```
   - If the “1C” version of the section is selected, this can be changed to “1E”. This includes optimisations primarily added to benefit the Cray T3E, but which may also improve performance on other cache based machines. The “1E” version of the section can produce different numerical results from the “1C” version.

6. Window: **atmos_Science_Section_DiffFilt**
   ```
   Atmosphere
   -> Scientific Parameters and Sections
   --> Section by section choices
   ---> Diffusion and Filtering
   ```
   - If the “1A” version of the section is selected, this can be changed to “1C”. This includes optimisations primarily added to benefit the Cray T3E, but which may also improve performance on other cache based machines. The “1C” version of the section can produce different numerical results from the “1A” version.

7. Window: **atmos_Science_Section_Energy**
   ```
   Atmosphere
   -> Scientific Parameters and Sections
   --> Section by section choices
   ---> Energy Adjustment
   ```
   - If the “1A” version of the section is selected, this can be changed to “1B”. This version contains optimisations which will benefit most MPP machines. The “1B” version of the section can produce different numerical results from the “1A” version.

8. Window: **atmos_Science_Section_RadShared**
```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
---> Section 70 Shared Radiation Routines
```
   - If the “1A” version of the section is selected, this can be changed to “1B”. This includes optimisations primarily added to benefit the Cray T3E, but which may also improve performance on other cache based machines. The “1B” version of the section can produce different numerical results from the “1A” version.

9. Window: **atmos_Science_Section_QSAT**
```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
---> Section 90 including QSAT
```
   - If the “2A” version of the section is selected, this can be changed to “2B”. This includes optimisations primarily added to benefit the Cray T3E, but which may also improve performance on other cache based machines. The “2B” version of the section can produce different numerical results from the “2A” version.

Again it is advised that once these changes have been made, a ”Check Setup” is performed, which will inform you if you have any inconsistent options set in the UMUI.

3.5 Atmosphere Model Control Code

Author: Richard T.H. Barnes, Last Updated: 4 Feb 98

3.5.1 Introduction

The atmosphere model control code is necessarily quite complex to allow the users great flexibility in their use of the Unified Model without resort to direct code modification. Most of the complexity is hidden from users by the UMUI. This section goes some way to explaining how choices made by the user in the UMUI are communicated to, and acted on in, the Unified Model itself.

3.5.2 Control Files

Model control information is output from the UMUI in the form of a number of files containing one or more namelists. These files and brief descriptions of their area of concern are:-

- CNTLALL - overall control aspects
- CNTLATM - atmosphere sub-model specific control
- CNTLGEN - generic control affecting each sub-model
- INITHIS - history file (for restartability)
- RECONA - control of the reconfiguration
- SIZES - control of addressing and dynamic allocation of memory
- STASHC - control of STASH diagnostic output system

For ocean, wave and coupled model running, other sub-model specific control files CNTLOCN etc. are produced.

CONTCNTL is a concatenation of the CNTL... files which is used by continuation runs (CRUN) as a means of overriding values.

The scripts that run the Unified Model do little more with the control files than transfer them to the host supercomputer and concatenate the SIZES and CNTL... files into a single file containing many namelists for convenience of reading. One important thing they do do is set the I/O environment variable names and corresponding physical file names by using the contents of the last namelist in INITHIS.

3.5.3 The Main Program UM_SHELL

It may be useful to read the rest of this section in conjunction with a calling tree of Unified Model routines, such as can be found at section 3.2. The main program routine is known as UM_SHELL and has the following tasks.

Very early in a Unified Model run the namelists are read in and used or stored by the following routines:-

- READSIZE which reads the namelist &NLSIZES from the file SIZES,
- UM_SETUP which calls READHIST to read the history file and READCNTL to read from the CNTL... files.

3.5.4 The Difference between History and Control Files

A brief note on the difference between the history file and other control files may be useful at this point.

The history file contains only those control variables needed for model restartability, and apart from the I/O filename data and resubmit information in INITHIS is initialised in qxsetup rather than being user set in the UMUI. The history file is processed by the small executables qxsetup, qxpickup, qxcombine, qxhistreset and qxhistreport as well as by the model, whereas other control files are simply generated by the UMUI and read by the model.

This has implications for model development - a new control variable can readily be added to the relevant model comdeck and its value set with a hand-edit to the appropriate CNTL... file. However if it is necessary to add a variable to the history file (much less likely) the history small executables will have to be remade too.

3.5.5 The Main Program UM_SHELL continued and Routine U_MODEL

Domain decomposition (for MPP machines only) is dealt with next followed by the routine DERVSIZE which derives further sizes needed for dynamic allocation of memory.

Calls to HDPPXRF and STASH_PROC and its subsidiary routines deal with decisions about which primary variables will be given space in the general D1 array and which diagnostics will be generated by the output processing sections of the model.

The call to UM_INDEX and its appropriate sub-model routines sets up the lengths of all the main storage arrays, allowing them to be dynamically allocated at runtime.

All these preliminaries lead up to the main call to the model itself in routine U_MODEL.

Within routine U_MODEL, the first comdecks ARGSZSP etc., TYPSZSP etc., and SPINDEX do the dynamic allocation of memory, along with TYPSPD1 and the following dozen or so comdecks a little further down the routine. See UMDP C1 [27] for more details on Dynamic Allocation of Primary Fields.

The comdecks CHISTORY and CCONTROL and their component comdecks are the way in which the variables in the history and control namelists can be accessed at the top levels within the model. Each basically consists of declarations, common block and namelist for each class of control variables.

Ignoring certain T3E and STASH specific code, the first real work is done in the routine INITIAL, which is called just once at the start of each run, whether a normal run (NRUN) or a continuation run (CRUN). Its functions include initialising STASH control arrays, reading the initial model dump and setting pointers for the D1 array from STASH/addressing information, initialising the model time and checking its consistency, setting up timestep control switches and opening output files required at the start of the run.

Thereafter in U_MODEL we come to the major loop over model timesteps (and over groups of timestep for coupled runs).

Routine INCRTIME increments the model time by one timestep and routine SETTSCTL sets the timestep control switches appropriately. Eg. LINTERFACE indicates whether lateral boundary condition files will need to be generated this timestep, and LANCILLARY whether any ancillary fields will need updated this timestep. When the values of these control switches are false, certain parts of the code can be safely skipped for this timestep.

Opening and initialisation of output files, and possible re-initialisation, is controlled by a call to PPCTL.

The routine OASIS_STEP refers to the OASIS system for coupled model running which was introduced at vn4.4.

3.5.6 The Routine ATM_STEP

The heart of the time integration of the atmosphere model is the routine ATM_STEP. Apart from STASH diagnostic and lateral boundary condition aspects, this basically calls the 3 major section of the atmosphere model, viz. the dynamics, physics and (optionally) assimilation. If lateral boundary values are to be generated or updated, this is done between the dynamics and the physics, so that the conserved variables thetal and qt are available. This discussion covers control aspects of the dynamics, physics and lateral boundary values from ATM_STEP down to plug-compatible routine level. Assimilation is documented elsewhere, see UMDP 30 [16].

Routine ATM_DYN performs a number (normally 3) of adjustment steps followed by an advection step. Optional tracer advection of passive tracers, aerosols, sulphur variables, and cloud ice for the mixed phase precipitation scheme can be performed. Tracer advection of the thermodynamic variables thetal and qt can also be used as an alternative to the standard advection. The whole dynamics process may be executed more than once per physics timestep, and the number of dynamics sweeps may also be automatically doubled (effectively halving the dynamics timestep) if there are signs of model instability.

Routine BOUNDVAL performs updating of lateral boundary values for the main prognostic variables in Limited Area Models, using values generated by a driving model which may be global or a larger limited area. Boundary updating may be done over a zone with variable weighting across the zone. Usually the zone is 4 gridpoints with the weighting decreasing linearly towards the interior. It is only the primary prognostic variables that have their lateral boundary values updated. These are surface pressure (Pstar), the multi-level wind components (u & v), the total water potential temperature (thetal) and the total water (qt). i.e. for the thermodynamic variables, the conserved variables thetal & qt are updated. If the run contains tracers as prognostic variables, the model will try to update these too. If the run is using the mixed phase cloud/precipitation scheme, the thermodynamic variables to be updated are liquid water potential temperature, total liquid water and cloud ice.

Routines GEN_INTF/GEN_INTF_A generate the lookup headers and interpolate boundary data for particular times from a global or limited area model for one or more interface files to provide lateral boundary conditions for other limited area models. Again it is only the primary prognostic variables that are included in the output lateral boundary conditions. These are surface pressure (Pstar), the multi-level wind components (u & v), the total water potential temperature (thetal) and the total water (qt). i.e. for the thermodynamic variables, the conserved variables thetal & qt are output. If the run contains tracers as prognostic variables, the model will output these too. If the run is using the mixed phase cloud/precipitation scheme, the thermodynamic variables that are output are liquid water potential temperature, total liquid water and cloud ice.

Routine ATM_PHYS performs a single physics timestep, which involves some or all of cloud diagnosis, solar astronomy, short- and longwave radiation, boundary layer, sulphur chemistry, large-scale precipitation, convection, surface hydrology, vertical diffusion, gravity wave drag, and vegetation. Diagnostics can be generated and output via the STASH system for all of these processes, but there is no actual call to STASH itself within ATM_PHYS.

At the end of ATM_STEP, end of timestep diagnostics are generated in calls to routines ST_DIAG1 and ST_DIAG2, and a call to STASH for section 0 (prognostic variables).

3.5.7 The Routine U_MODEL continued

Meanwhile, back at the level of U_MODEL, ATM_STEP is followed by equivalent routines controlling the other sub-models (currently ocean, slab and wave), but these will not be considered here.

The next event in an atmosphere model integration is the possibility of a model dump, performed, if LDUMP is true, by the routine DUMPCTL. To allow bit-reproducibility of model runs restarting from any of these model dumps (which may be compressed from 64-bit to 32-bit real numbers to save space), the routine RESETATM is called to recalculate the thermodynamic/cloud values from those of the conserved variables in the dump.

Following this, if it is a suitable timestep (LMEAN true), climate mean diagnostics are created using routine MEANCTL. Whenever dumps are made, either restart or climate mean, the history file values are updated and written to the temporary history file using routine TEMPHIST.

Routine EXITCHEK checks whether the model run has completed the required number of timesteps and initiates the model completion process if it has.

Finally, at the end of the timestep, if the control logical LANCILLARY or LBOUNDARY is true, then the corresponding routine UP_ANCIL or UP_BOUND is called. UP_ANCIL calls REPLANCA to do any updating of model fields by ancillary field values, according to instructions passed from the UMUI in the namelists & & & in the file CNTLATM. UP_BOUND does not actually update the lateral boundary values - BOUNDVAL called in the middle of ATM_STEP does this. Rather UP_BOUND reads the next set of boundary value data from the lateral boundary conditions file and computes new lateral boundary tendencies.

If the model run is not ending this timestep, we reach the statement GOTO 1 and loop back to 1 CONTINUE to carry on with the next timestep of the run.

As they are the heart of the atmosphere model, it is worth looking at the constituent parts of the dynamics (ATM_DYN) and physics (ATM_PHYS) control routines in more detail.

3.5.8 The Routine ATM_DYN

ATM_DYN - DATA STRUCTURES - The ”ARG” comdecks are used extensively to provide the argument list with its arrays which are declared and dimensioned via the corresponding ”TYP” comdecks. A fair amount of local workspace storage is dimensioned by lengths passed through the argument list, eg. P_FIELDDA.

ATM_DYN - CONTROL VARIABLES - these are mainly held in the common block of comdeck CNTLATM, included in the code via the comdeck CCONTROL. The key control variables used in ATM_DYN are:-

- A_SWEEPS_DYN - no.of sweeps of dynamics routines per physics timestep.
- L_HALF_TIMESTEP_DYN - whether to check for windspeeds $>$ WIND_LIMIT as a sign of model instability requiring halved dynamics timestep.
- L_HALF_TIMESTEP_DIV - whether to check for top level divergence $>$ DIV_LIMIT indicating instability needing halved dynamics timestep.
- L_LSPICE - indicates mixed phase large scale precipitation scheme with cloud ice, so thermodynamic transformation need to be treated differently. Also indicates that cloud ice is to be advected using tracer advection.
- L_TRACER_THETAL_QT - indicates that thetal & qt will be advected by the tracer advection scheme, instead of the standard scheme.
- L_SET_FILTER - indicates that filtering constants need to be computed.
- L_FIELD_FLT - indicates that, during assimilation, fields of theta & q will be filtered before the analysis dump is written out.
- L_MURK - indicates aerosol is present requiring tracer advection.
- L_SULPC_SO2 - indicates sulphur is present requiring tracer advection.

- L_MURK_SOURCE - indicates a source function is to be applied to the murk aerosol variable from an ancillary file.
- L_MURK_ADVECT - confirms whether the murk aerosol is to be advected.
- L_MURK_BDRY - indicates that boundary values are to be applied to the murk aerosol. N.B. this is suitable only for UK mesoscale area.
- L_SO2_NATEM - indicates use of a sulphate emissions ancillary file.
- LASSIMILATION - indicates whether to use divergence damping coefficients for assimilation or forecast.

ATM_DYN - CODE STRUCTURE - Ignoring calls to diagnostics routines, the basic structure is as follows:-

The loop ”DO I_LOOP=1,A_SWEEPS_DYN” allows more than one sweep of the dynamics code per model (physics) timestep. This is so that the model can be kept dynamically stable without having to resort to an unnecessarily short model timestep. Typically we use A_SWEEPS_DYN values in the range 2 - 4.

For second and subsequent sweeps (I_LOOP $>$ 1), we call routine THLQT2THQ to convert the conserved variables thetal, qt back to theta, q, qcl, etc.

If requested via L_HALF_TIMESTEP_DYN or L_HALF_TIMESTEP_DIV, tests are made to check for windspeed or top level divergence greater than limits set in the UMUI. These may indicate that the model is becoming dynamically unstable, in which case the number of dynamics sweeps can be doubled for this timestep to help preserve stability.

If total water or cloud ice are going to be advected by the tracer advection scheme, checks are made to remove any negative values.

For global runs with assimilation, fields can be filtered before the analysis dump is made. If this is done in an MPP run, SWAPBOUNDS is called to reset the halo values on each processor.

With the incorporation of the mixed phase cloud/precipitation scheme into the Unified Model at vn4.4, the are now several places with alternative calls to cloud and thermodynamic conversion routines depending on the value of L_LSPICE.

Also at vn4.4 a set of code sections optimised for faster running on the Cray T3E has been introduced, at the cost of slight lost of bit-comparison compared with previous versions. These are available for the adjustment, advection and diffusion sections, selectable from the UMUI.

The adjustment part of the dynamics (for which there are usually 3 adjustment steps) is done by routine ADJ_CTL. For details of this process see UMDP 10 [5].

We then need to call THETL_QT to convert THETA and Q back to THETAL and QT.

There are several situations where we might want to use the tracer advection scheme. These include passive or active tracers, aerosol for mesoscale visibility, the sulphur cycle, prognostic cloud ice for the mixed phase cloud/precipitation scheme, and the option to advect the thermodynamic variables with the tracer advection scheme. If any of these is true, we first call SET_TRAC to calculate the number of East-West horizontal sweeps required at each level for Courant number stability. Next the horizontal tracer advection routine TRAC_ADV is called, performing a number of East-West sweeps followed by one North-South sweep. Finally the vertical component of tracer advection is performed by routine TRAC_VERT_ADV. For some variables source functions are applied using routine TRSRCE, and for the UK mesoscale model only aerosol boundary conditions are applied by routine TRBDRY. In MPP runs a SWAPBOUNDS is done for each advected variable.

The next major call is to routine ADV_CTL to do the advection of the main prognostic variables. The only difference in the call for the mixed phase scheme (L_LSPICE true) is that cloud ice (D1(JQCF(1))) is replaced by a dummy zero field. For details of the advection scheme, again see UMDP 10 [5].

Following this the advected fields thetal, qt, u & v need to be ”mass-unweighted”. This is done by routine MASS_UWT, or, if only u & v require this because thetal & qt were advected by the tracer scheme, by routine MASS_UWT_UV.

For MPP runs, the wind components u & v are put through SWAPBOUNDS to update their halo values; for thetal & qt this is done inside the ”mass-unweighting” routine MASS_UWT.

Towards the end of the dynamics code, routine DIF_CTL performs any divergence damping and diffusion requested in the UMUI. Routine QT_POS_CTL checks for and removes any negative humidity that the advection scheme may have generated; choices for how to do this are available in the UMUI.

Finally, for global models, the wind components are filtered in the polar regions to remove grid point noise.

3.5.9 The Routine ATM_PHYS

Looking now just briefly at the control of the many physical parametrisations included in the Unified Model, we note the following points.

ATM_PHYS - DATA STRUCTURES - The ”ARG” comdecks are used extensively to provided the argument list with its arrays which are declared and dimensioned via the corresponding ”TYP” comdecks. A large amount of local workspace storage is dimensioned by lengths passed through the argument list, eg. P_FIELDDA. This workspace is given general names like WORK1. A number of key variables are passed from routine to routine via these workspace arrays, but some space is reused for more than one physical variable. Comments in the source code make this clear.

ATM_PHYS - CONTROL VARIABLES - these are mainly held in the common block of comdeck CNTLATM, included in the code via the comdeck CCONTROL. The key control variables used in ATM_PHYS are:-

- L_SW_RADIATE - indicates whether this is a SW radiation timestep.
- LEMCORR - indicates whether energy correction is to be calculated.
- L_LW_RADIATE - indicates whether this is a LW radiation timestep.
- L_USE_SULPC_DIRECT, etc. - indicates whether sulphur cycle is in use and hence if additional workspace is needed.
- L_SNOW_ALBEDO - indicates whether prognostic snow albedo is in use and hence if additional workspace is needed.
- L_SULPC_SO2 - indicates whether to call CHEM_CTL for the sulphur cycle.
- L_SULPC_DMS - indicates whether space is needed for a DMS variable.
- A_CONV_STEP - indicates whether convection is to be called less frequently than every timestep ($>$1) or not at all (=0).
- L_MOM - indicates whether convective momentum transport is in use.
- *IF -DEF,A07_0A - used to skip vertical diffusion code.
- VERTICAL_DIFFUSION - used to skip vertical diffusion routine if =0.
- *IF -DEF,A06_0A - used to skip gravity wave drag code.
- KAY_GWAVE - used to skip gravity wave drag routine if =0.
- *IF -DEF,A19_0A - used to skip vegetation code.

- L_PHENOL & L_TRIFFID - indicate whether vegetation is to be called.
- PHENOL_CALL & TRIFFID_CALL - indicate whether vegetation is to be called this timestep.
- L_LSPICE - indicates whether mixed phase precipitation is in use - N.B. if so polar updating of temperature, humidity and cloud prognostics is skipped.

ATM_PHYS - CODE STRUCTURE - Ignoring calls to diagnostics routines, the basic structure is as follows:-

The first few physics routines work with temperature rather than potential temperature, so straightaway we convert from potential temperature to temperature, and call the cloud scheme via CLD_CTL to recover the physical variables humidity q, cloud water qcl, cloud ice qcf and temperature T from the conserved variables total water qt and water temperature Tl. See UMDP 29 [15] for details.

In runs where the energy correction procedure is being applied, the routines ADD_ENG_CORR and ENG_CTL are called. See UMDP 28 [14] for details.

If it is a full radiation timestep (typically every 3 hours in global runs and every hour in mesoscale runs) then RAD_CTL is called which controls both short wave and long wave radiation schemes and associated astronomical calculations. There are several choices for each radiation scheme, selected using the UMUI. See UMDP 23 [9] for details.

Next comes routine BL_CTL to control the boundary layer scheme. See UMDP 24 [10] for details.

Looking at RAD_CTL and BL_CTL in more detail illustrates the 2 different ways in which different versions of a model section can be accommodated. In RAD_CTL, selection of which version to use is made at run time using IF tests on the appropriate element of variable H_SECT held in the CNTLATM comdeck. In BL_CTL, code selection is made at compile time where one of many possible versions of the interfacing or ”glue” routine BL_INTCT is chosen by the compile system using the information in the UPDEFS file for the job. Each version of BL_INTCT has the same input argument list but calls a different version of routine BDY_LAYR with different argument list depending on which of the 5 versions of the boundary layer scheme has been selected in the UMUI.

If you are running with the sulphur chemistry scheme, it is called next through routine CHEM_CTL.

Like the boundary layer scheme, stratiform or large scale precipitation is always called every timestep via routine LSPP_CTL. Likewise it uses an interface routine GLUE_LSPP to accommodate the choice of one of several different versions. See UMDP 26 [12] for details.

Following that, temperature is converted to potential temperature, since that is the variable used by the convection scheme. Routine CONV_CTL proceeds via an interface routine GLUE_CONV to invoke one of many versions of the convection scheme. The UMUI contains an option to call the convection scheme less frequently than every timestep, but note that if convective momentum transport is selected the convection scheme must be called every timestep. By setting the calling frequency to zero, the convection scheme can be safely skipped altogether, should this be desired. See UMDP 27 [13] for details.

Next the surface hydrology scheme is called via HYDR_CTL which again allows a choice of versions through an interface routine HYD_INTCTL. In MPP running some processors may have no land points. In this case the call to HYD_INTCTL is skipped for those PEs and an informative message issued. See UMDP 25 [11] for details.

The next few physical parametrisations are optional, and the *IF DEF facility of the source code manager nupdate is used to include or exclude the code for these processes, depending on values found in the UPDEFS file for the job.

The first of these is the vertical diffusion routine VDF_CTL, which is rather confusingly considered to be a physics routine, but in any case is only normally used in the mid troposphere for global runs to improve the vertical structure. See UMDP 21 [7] for details.

The second optional parametrisation scheme is gravity wave drag, which is used in all models except the mesoscale. Again GWAV_CTL calls an interface routine GWAV_INTCTL to allow access to a variety of schemes. See UMDP 22 [8] for details.

The third optional scheme, new at vn4.4, is a vegetation dynamics scheme called by VEG_CTL.

Further routines ENG_MASS_DIAG and CAL_ENG_MASS_CORR are called in runs using the energy correction procedure.

In general the physics routines do not update the prognostic variables on the polar rows (for global runs) or North- and Southmost rows (for limited area runs). In the latter case this is fine as the values are specified by the lateral boundary conditions. For global runs there is some code at the end of ATM_PHYS, with calls to the routine POLAR, to update the polar row values of the thermodynamic variables on the basis of fluxes calculated using the mean values on the row adjacent the pole. Note, however, that this polar updating has for the time being been hard-wired off for the mixed phase cloud/precipitation scheme by its developers because it resulted in negative cloud ice values at the poles.

Finally SWAPBOUNDS is applied to all the prognostic variables that have been updated during ATM_PHYS.

For further information on the atmosphere model physics and dynamics at the level below the control level, i.e. at the level of the plug-compatible science routines, you are recommended to refer to sections 2.1 & 2.2 of this User Guide. For greater detail, the appropriate Unified Model Documentation Papers should be consulted.

Chapter 4

Using the Unified Model

4.1 The Unified Model User Interface (UMUI)

Author: Steve Mullerworth, Last Updated: 2 Jul 98

4.1.1 Introduction

This document is aimed at new users of the UMUI. However, the GHUI, which is the design package that underlies the UMUI, has also been used to create other user interfaces such as the VARUI, and this document will be useful for these applications as well. In addition to the guidance in this document, any GHUI-based application should contain plenty of online help. Users should get into the habit of reading this help in order to avoid common pitfalls.

The best way of starting to learn about the application is to sit down with an experienced colleague for half an hour and work through the basic functions of the application. However, this document should also be read as it is a source of further information to help with dealing with future problems.

A preliminary description of the GHUI system is followed by a basic guide for using GHUI-based user interfaces and for dealing with problems and a step-by-step guide to getting started with an application. Finally, a list of frequently asked questions and their answers is included.

4.1.2 What is the GHUI

GHUI stands for Generic Hierarchical User Interface. It is a user interface design package that has been used to create the Unified Model User Interface (UMUI) and the VAR user interface (VARUI) among others. The GHUI is written in a language called Tcl/Tk. All the GHUI-based user interfaces have a similar look and feel. Therefore, although the following introductory description often refers to the UMUI, it is applicable to other applications.

The GHUI is particularly suitable for creating interfaces to large scientific packages such as the Unified Model. The flexibility of such packages means that a lot of parameters need to be set up in order to control a model run. A GHUI-based UI breaks down this process into a set of questions on a series of logically ordered windows. The user sets up a run by using the keyboard and the mouse to access each of the relevant windows and to input responses to each question.

If an application contains 200 windows and 1000 questions (as the UMUI does) then it would be a difficult job to set up a run from scratch. Therefore the GHUI provides systems for saving work to a central database and copying of jobs between users. Typically, then, a user begins by copying a standard job that is similar to the requirements of his or her particular project and then alters it to fulfil the new requirements.

4.1.3 Starting the UMUI

To start up the UMUI on the workstation systems, type:

```
$ umui
```

Some systems may have different startup methods - seek local advice if the above does not work.

When the UMUI starts, the rather large Entry Window appears, comprising an experiment table and a set of menus. There is a comprehensive help menu on this window which should be read.

To summarise, each separate UM setup is called a job. Jobs are grouped into experiments. When the experiment table is first opened, it will display a list of the experiments owned by the user. Each experiment has a unique four-letter identifier, and contains up to 26 jobs, each of which has a single letter identifying it.

Experiments are initially displayed in the “closed” state; that is, without displaying the list of jobs that it contains. If the folder icon to the left of the experiment identifier is selected with the mouse, the experiment folder will open; a list of of the jobs contained within the experiment will appear.

Experiments and jobs can be selected or unselected by clicking on the description text with the mouse. Note the difference between clicking on the folder icon, which opens and closes the folder, and clicking on the text of the experiment description which selects, by highlighting, or unselects the experiment. In general, the operations in the experiment or job menu operate on all highlighted items displayed in the window.

Jobs are typically described by both their experiment and job identifiers in one of the formats “aaik#b” or “aaikb” to specify job “b” in experiment “aaik”. All jobs owned by all users are kept together in a central database. Therefore it is possible to look at and copy other users’ jobs.

The help menu of the entry system contains information about all the menu options in the Entry Window. Before any useful work can be done a new UMUI user must first create a new experiment and copy a job into it. The help provides all the information needed to do this, but as an additional aid, the following is provided as a step-by-step guide to this first operation.

4.1.4 Creating the first job

The first time the UMUI is started, the experiment table should be blank because the user will not own any experiments or jobs. The following is a guide to creating an experiment and the first job.

- Use the mouse to select “New” from the Experiment menu. A dialog box will request a short description of the experiment.
- It is advisable to begin with a copy of a standard job rather than starting with a brand new one so take local advice to find the identifier of a suitable job.
- Select “Filter” from the Find menu. A panel will appear containing a set of options for filtering the jobs in the database.
- When starting the UMUI, the filter is automatically set to select the user’s own experiments using the “Owner” substring. Unselect this and select the experiment “Identifier” substring. Type in the four-letter id of your experiment and the experiment which holds the job you want to copy. eg “aaik,aamy”. Do NOT use spaces.
- Select the filter button - the two named experiment folders should be displayed in the table.

- Open the folder containing the job that is to be copied and highlight the job or jobs that are wanted by clicking over the text with the mouse.
- Highlight your own experiment, again by clicking over the text (not the folder icon) with the mouse.
- Select “Copy” from the Job menu. All highlighted jobs will be copied to the highlighted experiment. For each job you will be prompted for an alternative title and a job identifier letter.
- The new list of jobs can now be displayed by selecting the folder icon of the highlighted experiment.

4.1.5 The Constituent Parts of the UMUI

The Entry System

Having started the UMUI and created your first job you should now have gained some familiarity with the entry system. You should now read the introductory help reached from the entry window menu.

Try making a copy of your own job. Then create another experiment and try copying both jobs to the new experiment at the same time. What happens if you try copying one job to two experiments at the same time? Use the “Difference” function from the Job menu to compare two jobs - note that the speed of this function is dependent on the number of differences and it can be very slow for radically different jobs. Try deleting some jobs. “Delete” is a significantly underused function! Remember that each job uses up diskspace resources. Additionally, the more jobs there are in the system, the slower it is.

Experiment with the Filter system. What happens if you turn off both Experiment and Job filters? How do you get the Entry System to display your jobs and the jobs of a colleague at the same time? What happens if you try to copy one of your jobs to your colleague’s experiment? Finally, try opening a job. Opening a job will take you into the navigation system, described next.

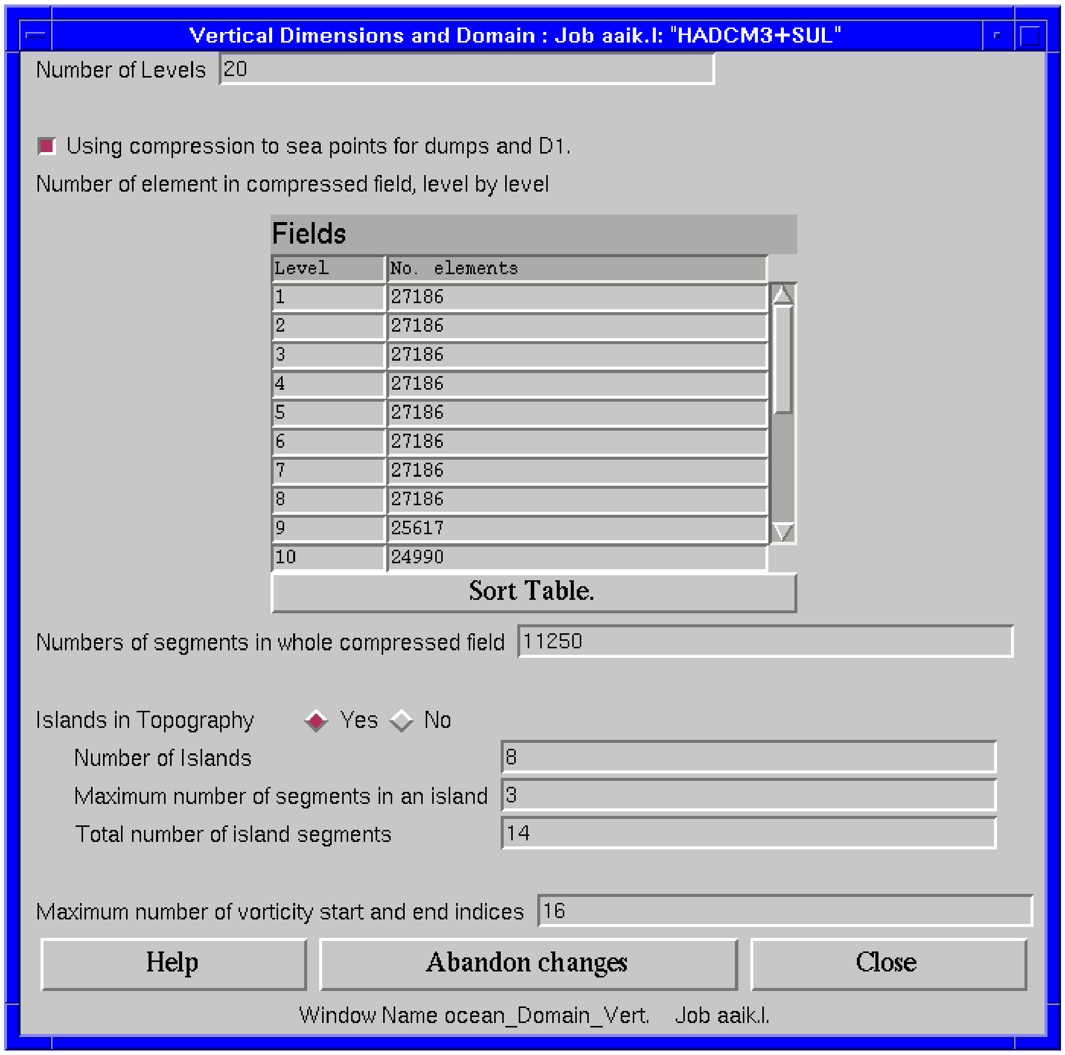
The Navigation System

The navigation window is the main job edit window which stays on the screen throughout a job editing session. All input panels (windows containing questions) are reached through this window by using the “navigation tree” in the upper part of the window. The buttons in the lower part provide functions for such things as saving changes, processing jobs and quitting the editing session.

You should now read the help windows on the navigation system which are accessed from a menu that appears when the “Help” button is selected.

Input panels

The navigation tree provides access to what are called “Input Panels ”; windows that contain the questions for setting up a job. The navigation tree provides a structure for grouping input panels into different categories and subcategories, which makes it easier to find a particular panel.



Vertical Dimensions and Domain : Job aaik.l: "HADCM3+SUL"

Number of Levels: 20

☑ Using compression to sea points for dumps and D1.

Number of element in compressed field, level by level

Fields

Level	No. elements
1	27186
2	27186
3	27186
4	27186
5	27186
6	27186
7	27186
8	27186
9	25617
10	24990

Numbers of segments in whole compressed field: 11250

Islands in Topography [✓ Yes] No

Number of Islands: 8

Maximum number of segments in an island: 3

Total number of island segments: 14

Maximum number of vorticity start and end indices: 16

Window Name ocean_Domain_Vert. Job aaik.l.

Figure 4.1: A simple example input panel from the UMUI showing entry boxes, a check box, a table widget and a pair of radiobuttons. The three buttons at the bottom of the window are common to all panels. ‘Help’ brings up local help for the window, ‘Close’ and ‘Abandon’ close the window with or without saving changes.

The standard GHUI style input panels comprise a series of questions and inputs in the form of buttons, entry boxes and tables. At the base of the panel is a row of buttons. “Help” provides specific help for the panel - you are strongly advised to get into the habit of reading it. “Abandon” closes the window and discards any changes. “Close” validates all the responses in the window. If an error is found, an error dialog box appears and the panel cannot be closed. Otherwise, changes are kept and the window is closed (Note: such changes are not permanently changed until the “Save” button is used to save changes to the central database). Some panels have further buttons next to the Save, Abandon and Close buttons that, when selected, lead to other panels.

Some applications may include some non-standard panels designed for the specific needs of that application. These should be documented locally and, if properly designed, should have their own online help. The UM STASH facility is managed by a non-standard UMUI window.

4.1.6 The Standard Working Procedure

You should follow this sequence of events when editing and running jobs. The help will give a fuller description of each of the processes.

Define the job. The definition of the job should have been finalised through the application's input panels.

Check Setup. Helps check that everything is consistent. Major errors such as unset questions and inconsistent responses will be highlighted, but a successful check does not guarantee that the job will work.

Save. Saves changes to jobs back to the central database. As with any system, it is advisable to save periodically during a long editing session.

Process. Processes the job definition into control files suitable for use by the UM.

Hand Edit your job. Optional. The process step will create a job directory containing the control files and will report the name of the directory. You may wish to hand edit the control files. This is NOT the purpose of the Hand Edit button on the navigation window:

Submit. Run the job. In the UMUI, the Submit button will do this.

4.1.7 Additional Guidance for the UMUI

Submitting UM Jobs

Processing a UMUI job creates a job library of 20-30 files in a directory called $HOME/umui_jobs/abcde where abcde stands for the five-letter job identifier. Each time the job is processed, the old directory is completely cleared out, so other important files should not be kept there.

Processed jobs can be submitted using the UMUI Submit button. On some systems a umsubmit script is installed which can be used. Once submitted, the files in the UMUI job library are copied to the target machine. The job library on the workstation can then be edited, or the job can be reprocessed, without affecting the running job.

When the first job is submitted under a particular username, a umui_runs directory is created in the user's home directory on the target machine. The job library of each subsequently submitted job is then copied to a subdirectory within umui_runs. The name of the subdirectory is created from the job id plus a time stamp (e.g. umui_runs/aaaka_23452345). These subdirectories are removed after a successful run, but umui_runs should be cleared out periodically because a large number of job libraries from killed runs can gradually build up.

Browsing Output from UM Runs

The output from UM runs is to be found in the output directory $MY_OUTPUT on the target machine with name $JOBNAME[$COUNT].$RUNID.$DATE.$TIME.$ACTION.

$JOBNAME is the job-name defined through the UMUI by the user.

$COUNT. This number is incremented to differentiate the output from a sequence of automatically resubmitted runs.

$RUNID is the RUNID ($EXPID$JOBID) of the job.

$DATE is a date stamp d{%y%j}.

$TIME is a time stamp t{%H%M%S}.

$ACTION is the optional action defined by the user. At the Met. Office a server is set up that performs the action defined, such as send to fiche, or archive.

Typically, UM output files are thousands of lines long, containing details of the applied modifications, the compilation, the reconfiguration, much of the job library details in addition to the output from the run. Some experience is required to gain a full understanding of the output. Help from experienced colleagues in diagnosing problems is very useful and other information is available in the Unified Model User Guide.

The amount of output can make it quite cumbersome to view. Utilities such as pg and more are useful, as are editors such as emacs. The vi editor can have problems due to the length of some of the lines in the output. On the Cray there is a browser called xbrowse.

```
sn4023$ xbrowse H20MKE01.aaaee.d96011.t111020.leave
```

This is an x-windows browser so the DISPLAY environment variable needs to be set appropriately and exported first. It contains online help.

Other people's output can be viewed by looking at the target machine directory $JOB_OUTPUT. This directory also contains a README file defining how to set up .redirects files for some of the action options.

Submitting a run A more detailed description of the job submission procedure provides insight that is helpful when dealing with job submission problems caused by broken machine connections or when interactive job execution is required.

The machine to which a run is submitted is called the target machine. When the user submits a job to the target machine the following operations take place:

- The job library is copied to a subdirectory in $HOME/umui_runs on the target machine.
- A unique number, the SUBMITID, is computed and substituted into the top of the SUBMIT script.
- The SUBMIT script is executed. The SUBMIT script creates another script called umuisubmit, which consists of the following items concatenated:
    1. A list of QSUB options (Cray supercomputers only).
    2. A list of variables and values computed by the SUBMIT script.
    3. A copy of the SCRIPT job library file.
- The umuisubmit script is then run by SUBMIT either by submitting to the batch queue on Cray supercomputers or using 'at' on other platforms.

On some systems a further processing step may take place due to differences between the local system and the system for which the submit program was written. However, the basic steps described above should be common to all systems.

The umuisubmit script is the top level script for the whole of a UM run. It sets up a temporary directory into which it copies all the relevant job library files needed to control a run, and it calls the lower level UM scripts which control such things as the compilation and execution of the UM program, the post-processing of dumps and PP files, and the management of output.

An Introduction to the Job Library

This section is intended to be a basic description of some of the more important job library files - those files produced when a job is processed.

The following are short descriptions of the job library files that would form part of a typical coupled atmosphere and ocean job. Files with names that terminate with A or ATM relate to the atmosphere component and those that terminate with O or OCN relate to the ocean part.

SUBMIT creates the top-level umuisubmit script.

SCRIPT one of the components of the umuisubmit script.

CNTLALL includes various general parameters that apply to all submodels such as run length and archiving selections.

CNTLGEN includes other general parameters that are dimensioned by submodel. For example, in an atmosphere-ocean coupled job,

```
STEPS_PER_PERIODim=48,24,0,0
```

shows that there are 48 steps per period in the atmosphere component, 24 in the ocean component, and that there are two other unused submodels (slab and wave).

CONTCNTL A particular experiment is first run as TYPE=NRUN, or "New Run" in the SUBMIT script, for perhaps one or two dumping periods. This enables confidence to be established that the run is working. Files left in the data directories at the completion of an NRUN allow a run to be restarted from the point it finished after hand-editing TYPE=NRUN to TYPE=CRUN, or "Continuation Run". The parameters in the CONTCNTL file control the continuation run. It contains copies of CNTLALL, CNTLGEN, CNTLATM and CNTLOCN. The CRUN technique is generally used only for long integrations.

RECONA and RECONO control the reconfiguration program.

CNTLATM and CNTLOCN hold namelists containing scientific parameters that control the atmosphere and ocean models.

PPCTL determines how the various output data files are dealt with.

STASHC contains a list of STASH requests. STASH is the system selecting which fields are to be output from a model run as diagnostics.

PRESMA and PRESMO Existing STASHmaster records can be overwritten and new record can be added by specifying user STASH files. Copies of these records are output to PRESMA and PRESMO.

INITHIS For initialising the HISTORY arrays used for passing filename information around the model.

SIZES contains parameters relating to the sizes of dynamically allocated arrays; it needs to be read at an early stage of a run to set up arrays that are used later.

A number of files deal with the compilation of the model and reconfiguration executables.

UPDEFS holds all the DEFS that are applied to the extraction of source code from the UM Program Library (UMPL). For example, if option 3A of the convection (section 5) is requested, the UPDEFS file will contain the DEF A05_3A.

MODS_UM etc A number of files, prefixed with MODS hold lists of files containing modifications to code. MODS_UM modifies UM Fortran code, MODS_C modifies UM C code, MODS_RECF and MODS_RECC do likewise for reconfiguration Fortran and C code. MODS_SCRIPT modifies the UM scripts.

COMP_OPTS and RECON_COMP_OPTS Each model or reconfiguration deck has a standard set of compile options defined in the build system. To modify the compile options for one or more decks, a compile override file is used. The COMP_OPTS files contain lists of these override files.

SECT_MODE When the UM is built on a system, code for many of the sections is extracted from the UMPL with a range of predefined DEFS and precompiled with a range of options (typically standard options and debug options). This means that subsequent model compilations are faster because the object code for many decks already exists. The SECT_MODE file tells the compile system which sections should be included in this way, and also which options and which set of compile options. The options for precompiled sections in this file need to be consistent with the DEFS in the UPDEFS file.

Hand-Editing the Job Library The questions in the panels of the UMUI relate to parameters in the job library; switching a particular checkbox on or off might result in a related variable being set to .TRUE. or .FALSE. when the job is processed. Therefore, once an understanding of the various parameters is obtained it is common for users to bypass the UMUI when making small changes to jobs. In fact, for certain operations, hand-editing is necessary. A typical example is when converting a run from an NRUN to a CRUN.

Typically, a job is set up to compile and run the UM executable. When it is processed, the STEP variable in SUBMIT is set to 1 or 2 to indicate that the compile step is to be undertaken. When a run has completed, say, one dump period successfully, the job can be set up to run further on by:

1. Hand-editing NRUN (new run) to CRUN (continuation run) in SUBMIT.
2. Hand-editing STEP=1 or 2 to STEP=4 to indicate that the compile step does not need to be done again.
3. Altering the RUN_TARGET_END variable in CONTCNTL. If the NRUN was for 1 day then this variable would be set to

   ```
   RUN_TARGET_END= 0 , 0 , 1 , 0 , 0 , 0 ,
   ```

   which means 0 years, 0 months, 1 day and 0 hours, minutes and seconds. If it is altered to

   ```
   RUN_TARGET_END= 0 , 0 , 10 , 0 , 0 , 0 ,
   ```

   then the CRUN will run for a further 9 days.
4. Resubmitting the job. The phist file left by the previous run contains all the details of the current position of the run.

4.1.8 Frequently Asked Questions

This section includes a list of some common questions and answers relating to the administration and use of the UMUI.

The Questions

1. Bus error when trying to start UMUI.
2. Why does cut-and-paste not work properly.
3. No more colours left in colourmap; changing screen's colour model to monochrome.
4. I get a "SYSTEM WARNING: Internal inconsistency" error box when I close a particular window.
5. I get an "Access denied at local host" failure when I try to submit a job.
6. I get an error message about the server when I try to Save, Load, Quit...

7. When I open a job I get an error saying something about a variable not being in the basis database or being in the wrong segment.
8. I get an error when I try to process my job. How do I find out its cause?
9. I got an error when I processed my job. I have fixed it but now the “Process” button is stuck down ?
10. I cannot start the server from the umui_admin client.

1. Bus error when trying to start UMUI.

   This occurs on some x-terminals when they are overloaded. Closing another application usually helps.

2. Why does cut-and-paste not work properly.

   This is a limitation of the version of Tcl/Tk in use. The problem will be rectified when the GHUI system is upgraded to use the latest version of Tcl/Tk (expected mid-1998).

3. No more colours left in colourmap; changing screen’s colour model to monochrome.

   Another application (commonly Netscape) has grabbed all the colours. Close the application and restart the UMUI. Consider running Netscape in one of the following modes: netscape -mono, (black and white); netscape -ncols 20 (reduced colour map); or netscape -install (private colour map).

4. I get a “SYSTEM WARNING: Internal inconsistency” error box when I close a particular window.

   Do not worry about it too much - it is a minor internal inconsistency. But please report it to UMUI administration.

5. I get an “Access denied at local host” failure when I try to submit.

   The word “local” as opposed to “remote” informs you that the job has been rejected by the machine to which you have submitted it. You have probably submitted the job to the wrong queue. Check your selection in the “Personal Details” window.

6. I get an error message about the server when I try to Save, Load, Quit etc.

   Read the message very carefully. It may be an intermittent fault so you could “Retry”. If retry does not work and the error message asks you to report the problem then please do. If you have made a lot of changes since you last saved then use the “Download” button to save your job to a local file. Later, when the server is working again, “Upload” it back into the same job and save it properly.

7. When I open a job I get an error saying something about a variable not being in the basis database or being in the wrong segment.

   The UMUI has changed slightly since you last used it (common with development versions). You do not need to take any action. Once you save the job, the message will no longer appear when you reopen it.

8. I get an error when I try to process my job. How do I find out its cause ?

   Try the “Check Setup” button which checks all your settings and outputs warning messages. If you still get processing errors after fixing any “Check Setup” errors then it is probably a system problem. Please make a copy of the job as it is and report the problem to UMUI admin.

9. I got an error when I processed my job. I have fixed it but now the “Process” button is stuck down ?

   Ignore the appearance of the button and just press it again.

10. I get. “Pid 24065 killed due to text modification or page I/O error” when I try to open a job.

    An operating system problem occasionally results in such a failure when running from certain servers. The solution is to mv the executable to a temporary location and then cp it back. Do not immediately delete the moved version since some users will be successfully using it. The GHUI executables are under the GHUI installation directories and are called client.exe, server.exe and ghui.exe.

11. I cannot start the server from the umui_admin client.

    This is a problem that may affect the UMUI database adminstrator, not the user. You have successfully installed the server routines on one host and the client routines on another host. When you start the umui_admin client from the client host and attempt to “Start Primary Server” you get an error message “remshd: Login incorrect”.

    This is happening because you are trying to start the server from a different account on a different host. The umui_admin client that you have started is attempting to execute the following shell command to start the server process on the remote host:

    ```
    remsh <server_host> -n <server_umui_base>/bin/umui_server \
      <server_umui_base> <server_umui_dbse> PRIMARY
    ```

    The way we get around this “problem” is to create a shell script on the local client host called “umui_admin” which supplants the one you have installed. This shell script calls the umui_admin script on the server with the correct userid. You will also need to have a correct .rhosts on the server account to enable remote access from the client account.

    ```
    #!/bin/ksh

    remsh <server_host> -l <userid> -n \
      "export DISPLAY=$DISPLAY ; <server_umui_base>/bin/umui_admin"
    ```

4.2 Getting Started

Author: Shaun de Witt, Last Updated: 5 Feb 98

4.2.1 Introduction

OK, so you have managed to start up the UMUI and are faced with a blank screen, with the words *File Find Experiment Job*. These are the names associated with pull down menus; click on them with the mouse and you will get a whole list of things you could do if only you had an experiment/job. This section is designed to take you through this in as easy and painless a way as possible.

Later, we will also deal with the correct way to modify source files and make permanent and configured changes to the UM source code. This isn't necessary if all you want to do is run the model, but as you become more experienced, you will almost certainly find section of code which you will want to modify or even correct (the code is thoroughly tested for all normal situations, but there may be odd features lying in rarely accessed code which may have been overlooked).

Before we go any further, a quick word on nomenclature and pictures. This is really for people who are used to using Windows or other Interfaces. The concept of *Experiment* is really a folder or directory (which is pretty easy given that the chosen icon is typical of directory icons used by other interfaces) while a *Job* is more akin to a file. Indeed, a file can be written out which describes the Job. The reason for the name is for scientific convenience.

4.2.2 Before you start

There are a number of things which you must do before you start. These are outlined below. In what follows, within the Met. Office, the remote machine is the Cray T3E while the host machine will normally be a workstation. Other users must interpret these names appropriately to their computing system.

- Make sure you have a valid account on the host and remote machine, and any other machines which you may require access to (including COSMOS within the Met. Office)
- Ensure you have a *.rhosts* file set up in your home directory on the remote machine, which should look like the example below:
  ```
  local_host_name          local_host_user_name
  ```
- Ensure that you have a *.rhosts* file on your local machine, containing a line of the form given below:
  ```
  remote_host_name                remote_host_user_name
  ```
- If you want job output files automatically sent to you local machine from the remote machine, you will need to set up a *.redirects* file in the output directory on the remote machine. The instructions given below are specific to the Met. Office and will need to be modified for other users.
  - Ensure that there is a directory under /u/output with your user-name on the Cray T3E. If not, you will need to create one.
  - In this directory create a .redirects file containing the following line:
    ```
    workstat: HP_userid@fr0200:/home/string/HP_userid/HP_output
    ```

After this you should now be ready to start using the Unified model and UMUI.

4.2.3 Creating an Experiment

The first thing you have to do when you start is create yourself an experiment. Click on the *Experiment* label and the *New* menu item and you will get a pop-up box asking you for some text. This is used to say roughly what the jobs contained in the experiment are related to; for example you could type "Global model using different resolutions" (the quote marks are not needed). Then just click on OK and the UMUI will make an experiment for you. It will be given a four letter identifier, the meaning of which is not important but is documented elsewhere. All subsequent jobs created in this experiment will start with these four letters. And that is all there is to it. You now have an experiment, owned by you.

4.2.4 Standard Experiments

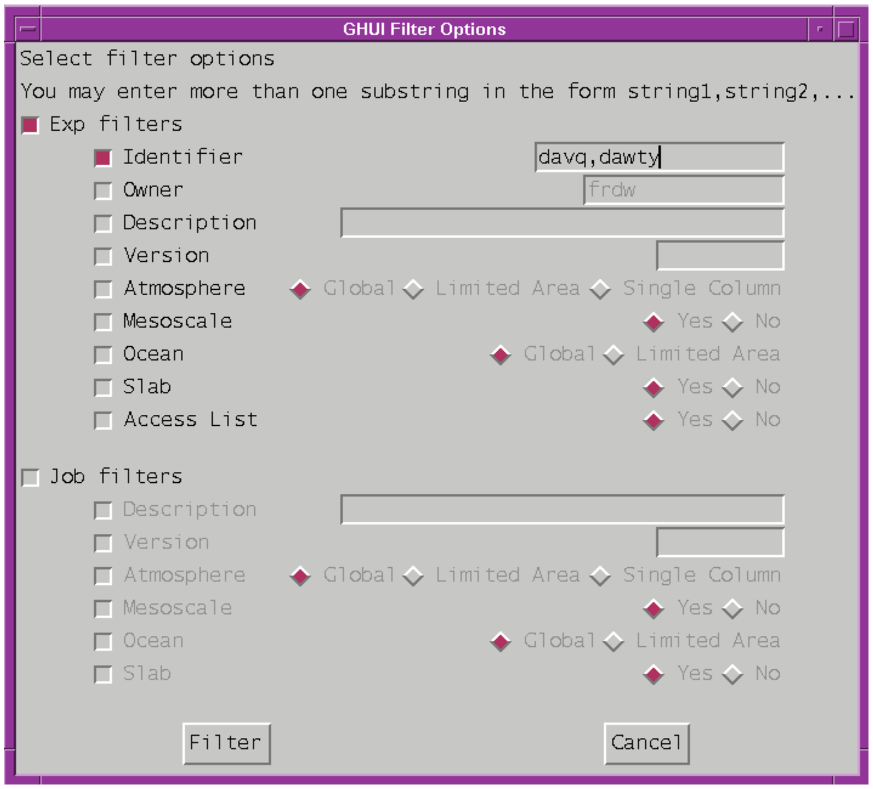
Before going on to the mechanics of how to, it is worthwhile mentioning standard experiments. These, as there name suggests, are common jobs which have had most of the UMUI inputs filled in for you. If you are using the Portable Unified Model, this should have come supplied with a set of standard experiments, one for each of global climate, LAM and mesoscale model. If you are not sure where to find them, you will need to ask the person who installed the Unified Model and UMUI.

Standard experiments for Met. Office Users can be found at the URL http://fr0800/umdoc/hestanexpt.html. External users will have standard experiments packaged with the software. These experiments have already been largely set up for you and only a few changes need to be made in order to start your job running. These changes are outlined in the next section.

4.2.5 Copying a Job

Jobs are where your real run is defined. You could create one from scratch, but since there are well over 250 windows to go through with almost no defaults, this would not only be very time consuming and error prone but also pretty wasteful. A much better way to create a job is to copy someone else's, since they will have done much of the work for you. It's best to either copy from a colleague who is doing similar work or use one of the "Standard Experiments" provided.

Copying the job is actually fairly easy. The first thing to do is to get the UMUI to list the experiment into which you want to copy the job and the experiment containing the job to be copied. This is all done using the menu command *Find* → *Filter*. This brings up a pop-up window which allows you to do all sorts of filtering, as shown below.


GHUI Filter Options

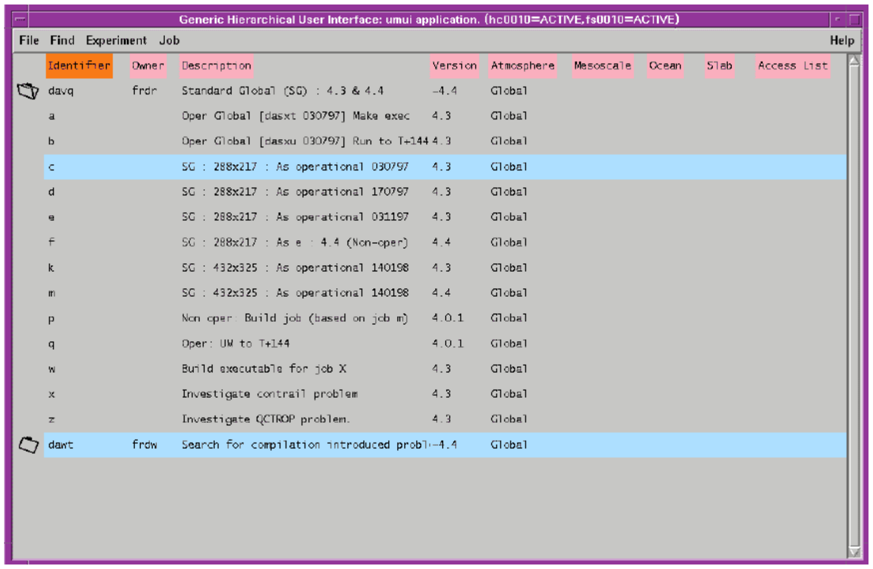
Select filter options

You may enter more than one substring in the form string1,string2,...

- ✓ Exp filters
  - ✓ Identifier: davq,dawty
  - ☐ Owner: frdw
  - ☐ Description:
  - ☐ Version:
  - ☐ Atmosphere: [✓ Global] Limited Area Single Column
  - ☐ Mesoscale: [✓ Yes] No
  - ☐ Ocean: [✓ Global] Limited Area
  - ☐ Slab: [✓ Yes] No
  - ☐ Access List: [✓ Yes] No
- ☐ Job filters
  - ☐ Description:
  - ☐ Version:
  - ☐ Atmosphere: [✓ Global] Limited Area Single Column
  - ☐ Mesoscale: [✓ Yes] No
  - ☐ Ocean: [✓ Global] Limited Area
  - ☐ Slab: [✓ Yes] No

Figure 4.2: Filter Pop-Up Window

Having got the two experiments required, you now need to select the job to be copied. If you click on the icon of the experiment containing the job to be copied, it will open up and show the contents of the folder; i.e. the jobs contained therein. Then click on the line of the job to be copied and this should highlight it. You then click on the line of the experiment where you want your job copied to, and you are ready. At this point the UMUI should look as below.


Generic Hierarchical User Interface: umui application. (hc0010=ACTIVE,fs0010=ACTIVE)

Identifier	Owner	Description	Version	Atmosphere	Mesoscale	Ocean	Slab	Access List
davq	frdr	Standard Global (SG) : 4.3 & 4.4	-4.4	Global				
a		Oper Global [dasxt 030797] Make exec	4.3	Global				
b		Oper Global [dasxu 030797] Run to T+144	4.3	Global				
c		SG : 288x217 : As operational 030797	4.3	Global				
d		SG : 288x217 : As operational 170797	4.3	Global				
e		SG : 288x217 : As operational 031197	4.3	Global				
f		SG : 288x217 : As e : 4.4 (Non-oper)	4.4	Global				
k		SG : 432x325 : As operational 140198	4.3	Global				
m		SG : 432x325 : As operational 140198	4.4	Global				
p		Non oper: Build job (based on job m)	4.0.1	Global				
q		Oper: UM to T+144	4.0.1	Global				
w		Build executable for job X	4.3	Global				
x		Investigate contrail problem	4.3	Global				
z		Investigate QCTROP problem.	4.3	Global				
dawt	frdw	Search for compilation introduced probl	-4.4	Global				

Figure 4.3: Copy Window

All you need to do then is use the menus again, clicking on *Job* → *Copy...*. This will then perform the copy and ask you to provide a one letter job identifier for your copy.

4.2.6 Before you run your new job…

When you copy a job, it is copied almost verbatim and there are a few changes which you must make before running it to ensure things like the output goes where you want it to, not where the original job wanted it put. We now go through each of the windows which you must, or are very likely to want to modify.

An important point to note when defining the fields outlined below is that they refer to information on the target machine, not necessarily the host machine which is running the UMUI. So, for example, any input data should reside on the target machine, not the host.

First, you must change all the user information, such as user-name and e-mail address. These can be found on the window
personal_gen
```
User Information and Target Machine
-> General Details
```
on which you will need to change the fields *Target Machine user-id, Mail-id for notification and job-name*. If you are running from within the Met. Office, you may also need to change the mysterious field *Tic Code*, the value of which you should find out from either you line manager or a work colleague. If you then click on the "SUBMIT" button, another window will appear, on which you might want to modify the *Output Class* field. After this it should be safe to close this window.

The next window which you almost certainly will need to modify is the one containing the list of environment variables to be used during your run. These are contained in the window
subindep_FileDir
```
Sub-Model Independent
-> File & Directory Naming. Time Convention & Envirnmnt Vars.
```
On this window, there will probably need to redefine the fields *DATAM* and *DATAW* It is safe to use the system environment variable $HOME, which is the location of your home directory, within this window without the need to define it. You may also need to change some of the entries in the table *Defined Environment Variables for Directories*. In particular, references to $HOME within this should be looked at carefully since they will almost certainly refer to the home directory of the person from whom you copied the job. You should either take a copy of these directories or define them explicity (e.g. /home/t11zx/data).

It is also worth checking the window
subindep_Compile
```
Sub-Model Independent
-> Compilation and Modification
--> Compile options for the model
```
and ensure that the option *Run from existing executable* is not selected.

If you are using different input data, you will also need to change the window
atmos_InFiles_Start
```
Atmosphere
-> Ancillary and input data files
--> Start Dump
```
which can be used not only to give the name of the input file, but also the start date. Note that for the start date and time, setting all of the fields to zero will disable date consistency checking. Similary, if you are also using assimilation, you may need to specify new input ACOBS files (files containing observation data) in the window // **atmos_InFiles_ObsData_ACObs**
```
Atmosphere
-> Ancillary and input data files
--> Observational data
---> ACOBS and IAU data
```

And thats all that you should need to do to define you job.

4.2.7 Submitting your job

Having defined your job and made the necessary modification to the appropriate windows, you are almost ready to start you job running. Before doing so, it is worth running *Check Setup*; one of the buttons along the bottom of the UMUI main panel. This will take a few minutes but will perform some consistency checking and ensure that all of the elements which need to be filled are filled.

After correcting any errors found, you should then *Save* you job. This will save you job to an internal database to ensure that the job is archived and recoverable. Strictly speaking this is not necessary; if you are putting in a temporary modification which you do not want saved, this step can be omitted. Processing the information will use the fields you have defined in the UMUI, but any modified fields will be lost when you quit the job window, unless you have saved it first.

The next step is to press the *Process* button. This writes various files into a directory on your local machine which are used during a run to define internal environment variables, array sizes and other parameters required during execution of the model. In fact, once you get used to these files, it is often quicker to modify these directly rather than going through the User Interface, although if you are new to using the system, this is not recommended. These files can be quite large and complex, and processing the files will take a few seconds, so don't worry if nothing seems to be happening.

Finally, clicking the *Submit* button will copy all of the necessary files to the machine on which the job will run, and start the actual run for you.

One final important note about closing down you job window. It is important that you close it down using the *Quit* button, not any other method. If you do accidentally close the window using some other method, next time you start the UMUI you may find that you can not open your job again since the database thinks it is still open. If this does happen then you will need to use the *Force Close...* option under the *Job* menu. This will force the database to close your job, but it means that any changes made since the last save will be lost.

4.2.8 Making Changes

As you become more familiar with using the Unified Model, you will probably want to tailor your job to suit your needs. This section will go through a number of windows which are useful and can speed up your development and testing time. These will be limited to windows under "Sub-Model Independent" and "Atmoshere", since the other models have basically similar windows.

Window: **subindep_OutputMan**

```
Sub-Model Independent
-> Output Management
```

This allows you some degree of control over the location of outputs from your job. The field *Output class for first job* lists a number of possibilities, only one of which may be selected. Note that the standard directory for output is system dependent; on the Met. Office Cray T3E, this is defined as "/u/output/*username*", where *username* is your user id on the supercomputer.

This window may also be used to specify whether or not you want to be mailed when your job is completed.

Window: **subindep_Compile**

```
Sub-Model Independent
-> Compilation and Modification
--> Compile options for the model
```

For your first run you will almost certainly need to compile and build an executable before actually running. If you create this execuatable in a permanent directory (not under /tmp), then you can use this window to chose to run later jobs from this, or any other, execuatable. This means that you do not have to waste time contiually re-compiling the same source code.

The section

```
Atmosphere
-> Model Resolution and Domain
```

should be set up OK for standard jobs. If you need to alter these, read the sections on changing resolution (4.4) and selecting a new LAM area (4.6).

The various windows under

```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
```

allow you access to various ’flavours’ of code sections. The “Help” associated with each window should be read before confirming any changes, since changing a section may result in you not being able to run another section, or changing input ancillary files.

At some point you may want to run with assimilation of observations. The options for this are to be found under the window
atmos_Assim_General

```
Atmosphere
-> Assimilation
--> Parameters, Control
```

where, for most standard jobs, no assimilation is used. Currently, the only valid choices are “No assimilation” and “Analysis correction assimilation<1A>”. The 3DVAR options will be available in later model releases. If you choose to turn assimilation on, then you must select both “<1A> AC scheme” and “Analysis correction assimilation <1A>”. In addition, you will probably need to fill in the time information on this window. Suitable times would be Start=0, End=10 and Nominal analysis time=6. To run with assimilation you will need suitable observation (ACObs) files. For more information, see the separate section 4.8.

The section

```
Atmosphere
-> Ancillary and input data files
```

is a large section with several further sub-sections to explore, though standard jobs should have everything set correctly if the pathnames of your start dump and any ancillary files agree with those specified in the appropriate windows. There is a separate section giving details of ancillary and boundary files (4.9).

The section

```
Atmosphere
-> STASH
```

which controls diagnostic output requests, is probably the biggest and most complicated section to set up. See section 4.14 on diagnostic output (STASH) for details. The standard jobs contain just a few typical diagnostic requests to give a flavour of what is possible. Unless you really need to, it is not worth changing this until you have quite a bit of experience with the Unified model. Even the windows under this section behave differently to other windows!

The final atmospheric section

```
Atmosphere
-> Control
```

contains many interesting and useful windows. The more commonly altered ones are described below, but it is worth looking through all the windows in this section.

Window:**atmos_Control_PostProc_DumpMean**

```
Atmosphere
-> Control
--> Postprocessing, Dumping & Meaning
```

allows you to alter how often you want output dumps to be produced and how you want the data packed. For climate jobs you can also define a meaning sequence (time over which data is meaned).

The windows under the subsection

```
Atmosphere
-> Control
--> Diagnostic (Point) writing
```

are very useful as an aid to debugging problems. The window **atmos_Control_Output_Writes_MaxMin** allows you to switch on the writing of maximum and minimum values of prognostic variables, which is useful for finding on which timestep problem variables go out of range.

The window **atmos_Control_Output_Writes_D1** allows you to write out the information in the D1 super-array at various points. This is useful when trying to find the routine in which a prognostic variable starts to blow up.

If you want to insert source code, the proper way to do it is to use a file in "Update" format (see section 3.1), normally called a modset. These are then included in the compilation of the model by including the modsets in the window
subindep_Compile_Mods

```
Sub-Model Independent
-> Compilation and Modifications
--> Modifications for the model
```

which allows you to specify both Fortran and C modsets. Remember that if you do add a modset, you will need to ensure that the model is recompiled. Handily, the facility to do this is included on this window.

4.3 Reconfiguration

Author: D.M. Goddard, Last Updated: 28 Jan 98

4.3.1 Introduction

The reconfiguration is a standalone program which allows the user to modify dump files. A dump file contains the necessary data to define the initial conditions for a model run in the format described in UMDP F3 (see [34]). The reconfiguration program will process both atmospheric and oceanic data, however its functionality for ocean dumps is limited.

The following facilities are available for atmospheric and oceanic data:

- Upgrading a dump from earlier releases of the model. Backward compatibility is generally not supported, but may work in some circumstances.
- Adding fields to or subtracting fields from the dump.
- Adding user defined fields to the dump.
- Adding tracers to the dump

The following facilities are also available only for atmospheric data:

- Interpolating to a new resolution or domain.
- Initializing fields with ancillary data.
- Incorporating UARS upper air analyses into the dump.
- The incorporation of ensemble perturbation: whereby ECMWF analysis increments are used to adjust prognostic fields in the dump file .
- A transplant option: A facility to replace prognostic data over part of the domain with prognostic data from another dump.

Note that any data from sources external to the model *eg ancillary files* must be at the same resolution and on the same domain as the output dump.

The reconfiguration is run as part of a Unified Model job submitted from the UMUI. The initial conditions are input through the UMUI and processed into a series of namelists in the RECONA (atmosphere) and RECONO (ocean) files in the umui_jobs directory. The reconfiguration may be switched on or off through the UMUI.

4.3.2 When does a dump file need reconfiguring?

A file needs reconfiguring if:

1. There is a change of resolution or area. The dump needs to be interpolated onto the new domain.
2. There is a request for configuration from an ancillary file. The ancillary data may overwrite existing data or may form a new field.

3. Upgrading to a new model release. This will ensure that the dump corresponds to the format for the new model release. The reconfiguration will convert any dumps as required. See UMDP F3 [34] for details of format. *Note downgrading to an earlier release of the model is not guaranteed to work.*

4. New user prognostics, ensemble perturbation, tracers or transplanted prognostic data are requested. Then the reconfiguration reserves space in the dump for these fields and initializes them. See UMDP S1 [37] for details.

5. Initial dump in GRIB format. Currently only GRIB data from ECMWF are supported.

4.3.3 Defining a reconfiguration using the UMUI

Define for all configurations

- Select window **subindep_Control**
  ```
  Sub-Model Independent
  -> General Configuration and Control
  ```
  The user can specify a reconfiguration only run by clicking on the box next to the statement "perform the reconfiguration step only".

- Select window **subindep_FileDir**
  ```
  Sub-Model Independent
  -> File and Directory Naming
  ```
  The user should define the URECONDIR environment variable if rebuilding the reconfiguration. It is advisable to use temporary space here unless the compile files produced will be required later. Also the user may define here environment variables for directories which are to be used later to point to file names. *eg in the next section the executable can be written to directory **$EXEC** where **EXEC** was defined in this section as mydir/exec*.

- Select window **subindep_CompRecon**
  ```
  Submodel Independent
  -> Compilation and Modifications
  --> Modifications for the reconfiguration
  ```
  The user may select to use the standard reconfiguration executable, use another executable or build a new executable. If a new executable is to be built, the user must specify the file name and directory to which the new executable is to be written and any modsets and compiler options to be used in the build.

- The windows referenced in the following are for the atmosphere submodel but similar windows can be found for the ocean and slab submodels by following the ocean or slab branches instead of the atmosphere branch.

  – The contents of the dump file depends on which scientific sections are requested. *For example* **Soil moisture in layer (STASH code 9)** *will only be incorporated into the dump if* **multi-layer hydrology** *sections are used.* So the user must specify what sections are required for the model run even if the model is run in reconfiguration only mode. In such circumstances the user should look at how the reconfigured dump is to be used. This will determine what needs to be in the reconfigured dump.

    To view and change the scientific sections selected look in the UMUI section
    ```
    Atmosphere
    -> Scientific Parameters and Sections
    --> Section by section choices
    ```

  - Select window **atmos_STASH_UserDiags**
    ```
    Atmosphere
    -> STASH
    --> User-STASHmaster files, Diags, Progs & Ancills
    ```
    In this panel the user should specify any user preSTASHmaster containing user prognostics which are to be included in the dump. *Note that this file must be local to the machine from which the UMUI is being run.*
  - Select window **atmos_STASH_UserProgs**
    ```
    Atmosphere
    -> STASH
    --> Initialization of User Prognostics
    ```
    In this panel the user selects the method of initialization of the user prognostics added to the dump in the previous panel.

Define for atmosphere submodel

- Select window **atmos_Domain_Horiz**
  ```
  Atmosphere
  -> Model Resolution and Domain
  --> Horizontal
  ```
  In this window the user defines atmospheric model horizontal resolution, domain and number of land points. For land points see section 4.3.4.
- Select window **atmos_Domain_Vert**
  ```
  Atmosphere
  -> Model Resolution and Domain
  --> Vertical
  ```
  In this window the user defines atmospheric model levels.
- Select window **atmos_Config_Prognos**
  ```
  Atmosphere
  -> Model configuration
  --> Prognostic variable choice
  ```
  In this window the user may add SST anomalies and aerosol fields to the dump.
- Select window **atmos_Science_Section_SulphChem**
  ```
  Atmosphere
  -> Model configuration
  --> Chemistry (Sulphur cycle)
  ```
  In this window the user may add sulphur cycle fields to the dump.
- Select window **atmos_Config_Tracer**
  ```
  Atmosphere
  -> Model configuration
  --> Tracers
  ```
  In this window the user may add atmospheric tracers to the dump.
- Select window **atmos_InFiles_Start**
  ```
  Atmosphere
  -> Ancillary and input data files
  --> Start dump
  ---> Start dump
  ```

In this window the user specifies whether or not to reconfigure the dump (see section 4.3.2). If so the user is then asked to specify the name of the initial dump, the name of the reconfigured dump, the type of horizontal interpolation and the type of coastal adjustment. Horizontal interpolation and coastal adjustment are discussed in section 4.3.5. If reconfiguration is not required, the user need only specify the initial dump. If the initial dump is in GRIB format, the user **must** reconfigure the dump and switch on the ECMWF GRIB switch.

- The user should then select which fields are required from ancillary files.

  - Some of these are climatological datasets and are accessed from the section
    ```
    Atmosphere
    -> Ancillary and input data files
    --> Climatologies and Potential climatologies
    ```
  - The remainder including *Land-sea mask* and *Orography* can be accessed from the section
    ```
    Atmosphere
    -> Ancillary and input data files
    --> Other ancillary files and Lateral Boundary files
    ```

See section 4.9 for more information about ancillary files and section 4.9.4 for more details on how to fill in the panels.

Define for ocean submodel

- Select window **ocean_Domain_Horiz**
  ```
  Ocean GCM
  -> Model Resolution and Domain
  --> Horizontal
  ```
  In this window the user defines ocean model horizontal resolution and domain *Note this must be the same as the input dump, as interpolation is not available for Ocean dumps.*

- Select window **ocean_Domain_Vert**
  ```
  Ocean GCM
  -> Model Resolution and Domain
  --> Vertical
  ```
  In this window the user defines ocean model vertical resolution and domain. *Note this must be the same as the input dump, as interpolation is not available for Ocean dumps.*

- Select window **ocean_Tracer_UserDef**
  ```
  Ocean GCM
  -> Tracers
  --> User Defined Tracers
  ```
  In this window the user may add oceanic tracers to the dump.

- Select window **ocean_InFiles_Start**
  ```
  Ocean GCM
  -> Input files
  --> Start dump
  ```
  In this window the user specifies whether or not to reconfigure the dump. If so the user is then asked to specify the name of the initial dump and the name of the reconfigured dump. If reconfiguration is not required, the user need only specify the initial dump.

- The user should then select which fields require initialisation from other input files.
  - Some of these are climatological datasets and are accessed from the section
    ```
    Ocean GCM
    -> Input files
    --> Climatologies and Potential climatologies
    ```
  - If extra tracers were requested these may need to be initialized from ancillary data. Select **ocean_InFiles_OtherAncil_Tracer**
    ```
    Ocean GCM
    -> Input files
    --> Other ancillary/input files
    ---> User Defined Tracer fields. Extra Tracers
    ```

See section 4.9 for more information about ancillary files and section 4.9.4 for more details on how to fill in the panels.

Define for slab submodel

- The user should select what fields are required from ancillary files. This is done in a series of windows in the section
  ```
  Slab Ocean
  -> Ancillary files
  ```

See section 4.9 for more information about ancillary files and section 4.9.4 for more details on how to fill in the panels.

4.3.4 Changing the resolution of the dump

The minimum requirement when reconfiguring to a new resolution is to state the number of land points in the interpolated land-sea mask at the new resolution. Then all the fields will be automatically interpolated to the new resolution from the original dump by the reconfiguration program. If the number of land points is unknown, it can be determined by running the reconfiguration with an arbitrary number of land points. This reconfiguration will abort with a message stating the number of land points expected. The UMUI job should then be reprocessed with the correct number of land points and reconfiguration rerun.

For a more realistic representation where more accurate simulations are needed, it is strongly recommended to use dedicated ancillary files at the new resolution which have been derived from high resolution base data. See section 4.9 for more details about ancillary files. Note that importing an ancillary land-sea mask may change the number of land points.

Warning: *Care must be taken with special fields such as orography and the land-sea mask. Requesting either of these from an ancillary file will cause interpolation, Interpolation can negate bit comparison as it affects other fields.*

4.3.5 Interpolation

Interpolation is the mechanism by which data is transformed from one resolution to another. The reconfiguration offers a choice of a bilinear technique or an area weighted technique to interpolate horizontally and assumes a linear relationship with log pressure to interpolate vertically. See UMDP S1 [37] for more details.

For general use the bilinear technique is recommended, however if interpolating onto a coarser grid the area weighted technique is preferred.

The reconfiguration also has a coastal adjustment step, which adjusts land specific fields at coastal points where horizontal interpolation uses a mixture of land and sea points. The spiral coastal adjustment scheme is recommended here as the other scheme may generate inappropriate results when the nearest land points are far removed geographically from the coastal points. This is explained in UMDP S1.

4.4 Changing Resolution

Author: Richard T.H. Barnes / Terry Davies, Last Updated: 5 Feb 98

4.4.1 Introduction

The Unified Model is designed to be run at any horizontal resolution reasonable for a hydrostatic model, from coarse climate resolution with just 10's of grid-points in each direction to high resolution limited area models with grid-lengths down to 5km (0.05 degrees), provided sensible parameters are chosen. Section 4.4.2 covers the basic windows involved in changing horizontal resolution.

Changing vertical resolution, however, needs to be done with greater care. In particular, avoid too great a difference between the thicknesses of nearby levels and make sure there is no oscillation of thicknesses of adjacent levels. That is, the half-level eta values that specify the vertical coordinate should be smoothly varying. There are now 4 sets of vertical levels that have been tried and tested in standard jobs (19 for global and LAM, 30 for high resolution global, and 31 and 38 for mesoscale). In addition the stratosphere model group have experience of running with around 50 levels, the extra levels all being in the stratosphere. Section 4.4.4 lists the windows involved if you do have to change vertical resolution.

4.4.2 Windows to Change Horizontal Resolution

1. Window: **atmos_Domain_Horiz**
   ```
   Atmosphere
   -> Model Resolution and Domain
   --> Horizontal
   ```
   As well as the Numbers of Columns and Rows, you will need to change the Number of Land Points - you may need to run your job once with a wrong value and let it fail to determine this value. If you are reconfiguring, search for 'No of land points =' in the output, near the land sea masks. The model's failure message near the end of the output will be 'SETCONA: Wrong number of land points', with just above it 'I=' also giving the correct number of land points and 'LAND_FIELD=' giving the value that is wrongly set. Unfortunately, the domain decomposition aspect of MPP running on the Cray T3E may make it more difficult to diagnose the correct number of land points from a model run. Yet another way is to use the utility pumf on your start dump (after reconfiguration) and look at the value of integer header item 25 in the pumf_head output.

2. Window: **atmos_Science_Tstep**
   ```
   Atmosphere
   -> Scientific Parameters and Sections
   --> Time-Stepping
   ```
   An increase in resolution normally needs to be accompanied by a reduction in timestep. If the 'period' is kept as 1 day, this means increasing the Number of timesteps per 'period'. A useful alternative strategy is to use more than one dynamics sweep per model timestep. This allows dynamical stability to be maintained without resorting to unnecessarily small timesteps for the physics parts of the model. Values of 2 - 4 dynamics sweep per model timestep have been used successfully.

3. Window: **atmos_Science_Onlevel**
   ```
   Atmosphere
   -> Scientific Parameters and Sections
   --> Level-by-level Constants, Physical Constants
   ```
   If model resolution has been significantly changed, the run is likely to fail within a few timesteps if the diffusion coefficients have not been scaled to be suitable for the new grid-length and dynamics timestep. See section 4.4.3 on changing horizontal diffusion for the issues involved here.

4. Window: **atmos_Science_Section_DiffFilt**
   ```
   Atmosphere
   -> Scientific Parameters and Sections
   --> Section by section choices
   ---> Diffusion and Filtering
   ```
   Depending on the height of the highest orography in your domain, you may be able to increase the pressure altitude (i.e. lower the model level) up to which horizontal diffusion is switched off over steep slopes. Doing this should give a small reduction in computer time. See section 4.4.3 for details.

5. Windows: **atmos_Science_Section_SW**, **atmos_Science_Section_LW** & **atmos_Science_Section_Conv**
   ```
   Atmosphere
   -> Scientific Parameters and Sections
   --> Section by section choices
   ---> SW Radiation OR LW Radiation OR Convection
   ```
   Increasing resolution will increase the memory requirements of the model - you may want to increase the "Number of segments" particularly for Long-wave and Short-wave Radiation, and possibly for Convection, to alleviate this. Alternatively, if running in MPP mode it may be necessary to increase the number of processors used by the job. To do this use:-

6. Window: **subindep_Target_Machine**
   ```
   User Information and Target Machine
   -> Target Machine
   ```
   Choose suitable new values for the number of processors in the North-South and East-West directions.

7. Window: **subindep_JobRes**
   ```
   Sub-Model Independent
   -> Job resources and re-submission pattern
   ```
   Choose suitable new values for the job resources, particularly memory and time. As a rough guide, scale the memory limit in proportion to the change in number of grid points (though increasing the 'segments' for the large physics sections may alleviate this). Scale the time limit in proportion to the change in number of grid points and number of timesteps required to complete the forecast (though the cost of compilation will remain fairly constant). Check the actual amounts used and refine your choices if necessary.

8. Windows in the section:
   ```
   Atmosphere
   -> Ancillary and input data files
   ```
   The reconfiguration program will interpolate dumps to other resolutions. However, the ancillary files cannot be used in model runs at other resolutions, so for these you must ensure that windows relating to ancillary files have "Not used" set for all ancillary fields. The model will fail with suitable error messages if you try to "Update" ancillary fields from climatological ancillary files at the wrong resolution. The reconfiguration program will fail if you try to "Configure" from ancillary files at the wrong resolution. See section 4.9.3 for notes on creating new ancillary files for new grids.

4.4.3 Changing Horizontal Diffusion

The formula for horizontal diffusion is given by equations 47 and 48 of UMDP 10 [5]. Essentially this is a conservative form of a Laplacian-type operator and repeated application gives higher-order diffusion. Higher-order diffusion is considered beneficial because it is more scale selective i.e. there is more damping on the less-accurate shorter scales than on the meteorologically important longer scales. However, the higher the order of diffusion the more expensive it is to use. In the integration scheme used by the UM, extra damping is required for high-resolution limited-area modelling as a means of controlling grid-splitting in extreme cases.

The diffusion coefficient used depends upon resolution and timestep and has to be chosen carefully to avoid having either too little or too much damping overall. The behaviour of the diffusion operator can be analysed by considering its behaviour on a wave-like structure. In practice, a one-dimensional analysis is adequate to estimate the diffusion coefficient. Once chosen, the final value can be refined by trial and error.

The effect of applying the diffusion to a 2 grid-length wave is given by:

$$D = 1 - \frac{2^{2k} K^k \Delta t}{\Delta x^{2k}}$$

where K is the diffusion coefficient, k is the order of diffusion, Δt is the timestep in seconds and Δx is the grid-length in metres. We choose the diffusion coefficient K as the value which reduces the amplitude of the 2 grid-length wave by $1/e$ after N applications of the horizontal diffusion. Thus, we require $D = e^{-1/N}$. Therefore the expression for the diffusion coefficient K is

$$K = \frac{1}{4} \Delta x^2 \left(\frac{1 - e^{-1/N}}{\Delta t} \right)^{1/k}$$

This expression can be evaluated on a calculator or a simple program may be written to output various values of K given the input values which are repeated here

- Δx = the approximate horizontal grid-length in metres
- Δt = the dynamics timestep in seconds
- N = the number of dynamics steps over which the 2 grid-length wave amplitude reduces by $1/e$ (the damping time is then $N * \Delta t$, e.g. say 6 hours is required for the climate model using a 30 minute dynamics timestep, then N = 6 hours / 30 minutes = 12). As can be seen in the examples below, the higher the resolution (smaller grid-length) then the shorter the damping time.
- k = the order of diffusion (k = 1 for ∇^2, 2 for ∇^4, 3 for ∇^6, etc).

The value of K returned need not be used exactly since the above formula is only a guide in one-dimension. Usually just one or two significant figures are sufficiently accurate. When used in the dynamics, the diffusion coefficient in the east-west (longitudinal) direction is scaled with the latitude so that the diffusion operates on the same length scales everywhere regardless of the convergence of the meridians. Below are some examples produced by the formula (using Fortran on an HP workstation), followed by the values actually used in the models.

- Climate Model: Δx = 280km, Δt = 1800 seconds, N = 24 (12 hour damping) K = 5.54742E+08 for third order (∇^6) diffusion. K = 5.47E+08 is used in climate versions.
- Old Global Model: Δx = 100km, Δt = 600 seconds, N = 6 (1 hour damping) K = 3.99894E+07 for second order (∇^4) diffusion. Use K = 4.0E+07 as was used operationally.
- New Global Model: Δx = 60km, Δt = 400 seconds, N = 4 (1600 second damping) K = 2.116E+07 for second order (∇^4) diffusion. Use K = 2.0E+07 as used operationally.
- Old UKMO LAM: Δx = 50km, Δt = 300 seconds, N = 4 (20 minutes damping) K = 1.69711E+07 for second order (∇^4) diffusion. Use K = 1.7E+07 as was used operationally.
- Old UK Mesoscale: Δx = 17km, Δt = 100 seconds, N = 5 (500 seconds damping) K = 3.076096E+06 for second order (∇^4) diffusion. Use K = 3.0E+06 as was used operationally.

- New Mesoscale: Δx = 12km, Δt = 75 seconds, N = 4 (300 seconds damping) K = 1.955E+06 for second order (∇^4) diffusion. Use K = 1.9E+06 as used operationally.

Note that at the top level, extra diffusion is normally applied since extra damping is required at least for winds and thetal. The suggested values are:

- Climate Model: Δx = 280km, Δt = 1800 seconds, N = 2.5 (1.25 hour damping) K = 3.59E+06 for first order (∇^2) diffusion. Use K = 4.0E+06.
- Old Global Model: Δx = 100km, Δt = 600 seconds, N = 3 (1/2 hour damping) K = 1.181119E+06 for first order (∇^2) diffusion. Use K = 1.0E+06 as was used operationally.
- New Global Model: Δx = 60km, Δt = 400 seconds, N = 3 (20 minute damping) K = 6.378E+05 for first order (∇^2) diffusion. Use K = 7.0E+05 as used operationally.
- Old UKMO LAM: Δx = 50km, Δt = 300 seconds, N = 4 (20 minutes damping) K = 4.608316E+05 for first order (∇^2) diffusion. Use K = 4.25E+05 as was used operationally.
- Operational Mesoscale Model: uses the same values at all levels.

The other diffusion parameter over which the user has control is the level at which the horizontal diffusion is switched off over steeply sloping model coordinate surfaces. This is specified in:
Window: **atmos_Science_Section_DiffFilt**

```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
---> Diffusion and Filtering
```

Typically, these surfaces are near the ground over mountains but the highest mountains in the UM make the model coordinate surfaces slope significantly even above 300hPa over the Himalayas and the Andes. Since this modification to the diffusion adds to the running cost of the model there is no need to apply the slope test where the model surfaces are horizontal, which is at least true for the levels that are pure pressure levels. The *pressure_altitude* used in global configurations is best set to 20000Pa (200hPa) if we wish to test for all sloping coordinate surfaces. Limited-area configurations can be changed according to the typical pressure just above the top of the highest orography in the region e.g. the UK Mesoscale model could use 70000Pa (700hPa). A large pressure value e.g. 200000Pa (2000hPa) can be used to switch off the slope test in the diffusion but this leads to noisy and sometimes unrealistic rainfall over mountains and in the Mesoscale model reduces the accuracy of fog forecasts.

4.4.4 Windows to Change Vertical Resolution

1. Window: **atmos_Domain_Vert**

   ```
   Atmosphere
   -> Model Resolution and Domain
   --> Vertical
   ```

   “Pressure weighted mean” is the recommended way of calculating model levels. As well as choosing Number of levels and ’wet’ levels (those at which moisture calculations are included), you have to specify the ’half-levels’ up to which the vertical coordinate is purely ’sigma’ and at and above which it is purely pressure. In between these a ’hybrid’ vertical coordinate is used.

   Given these constraints, the positioning of levels is specified by a set of values of ’eta’ for the n+1 half-levels, decreasing monotonically from 1.0 for the Earth’s surface to a small number (typically 0.0005) for the ’top’ of the atmosphere. These values should be smoothly varying as the sudden introduction of very thin or thick levels (or oscillations in level thicknesses) can cause unrealistic model behaviour, particularly in the physics routines. It is also best to leave the top 4 half-levels as set in the standard jobs, since model stability at the upper boundary has been found to be sensitive to these values.

Number of boundary layer levels is normally the number with eta values up to about 0.8, approximately equivalent to 800hPa. Leave number of deep soil levels unchanged here. Set number of cloud levels used in radiation to minimum of number of wet levels and number of levels-1. The other level parameters for gravity wave drag and vertical diffusion are best set by reference to the "Help" given in their respective windows in the section

```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
```

2. Window: **atmos_Science_Onlevel**

```
Atmosphere
-> Scientific Parameters and Sections
--> Level-by-level Constants, Physical Constants
```

If you have added levels, the values in these tables will need extending. The mesoscale model uses a different set of low level critical humidity ratio values from the other standard configurations.

3. Windows: **atmos_Science_Section_SW**, **atmos_Science_Section_LW** & **atmos_Science_Section_Conv**

```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
---> SW Radiation OR LW Radiation OR Convection
```

Increasing vertical resolution will also increase the memory requirements of the model - you may want to increase the "Number of segments" particularly for Long-wave and Short-wave Radiation, and possibly for Convection, to alleviate this. Alternatively, if running in MPP mode it may be necessary to increase the number of processors used by the job. To do this use:-

4. Window: **subindep_Target_Machine**

```
User Information and Target Machine
-> Target Machine
```

Choose suitable new values for the number of processors in the North-South and East-West directions.

5. Window: **subindep_JobRes**

```
Sub-Model Independent
 => Job resources and re-submission pattern
```

Choose suitable new values for the job resources, particularly memory and time. As a rough guide, scale the memory limit in proportion to the change in number of grid points (though increasing the 'segments' for the large physics sections may alleviate this). Just changing the vertical resolution does not necessarily require changing the timestep, so also scale the time limit in proportion to the change in number of grid points (though the cost of compilation will remain fairly constant). Check the actual amounts used and refine your choices if necessary.

6. Windows in the section:

```
Atmosphere
-> Ancillary and input data files
```

The reconfiguration program will interpolate dumps to other resolutions, both horizontally and vertically. However, the ancillary files cannot be used in model runs at other resolutions, so for these you must ensure that windows relating to ancillary files have "Not used" set for all ancillary fields - the model will fail with suitable error messages if you try to "Update", and the reconfiguration program will fail if you try to "Configure", with ancillary files at the wrong resolution. With changes to number and disposition of model levels this applies particularly to the ozone ancillary file. See section 4.9.3 for notes on creating new ancillary files for new grids.

4.5 Atmospheric Tracers

Author: David Robinson / Peter A. Clark, Last Updated: 18 May 98

4.5.1 Introduction

The UM contains the facility to advect a number (29/18 for atmosphere/ocean at version 4.4) of 'tracers'. These are variables that represent the mass mixing ratio (nominally kg/kg) of any quantity that the user wants to be advected around using the tracer advection scheme.

The tracer system is intended as a basis for people to test out some new scheme or for 'private' applications that would not be appropriate for general intrduction into the UM. The tracers are thus referred to sometimes as 'free use' tracers to remind potential users that they should not write code which hard wires them into some aspect of the model. The system may be used as a way of marking particular airparcels to track them, or may be (and is by various groups) used as the basis for more involved chemical modelling by the addition of extra code.

The tracers are advected by the model winds, but using a different numerical method (of which there is a choice - see UMDP 10 [5]). The methods are designed to ensure that sharp gradients are preserved while keeping all concentrations greater than or equal to zero. This is particularly important when the tracer concentration is taken to represent a real concentration, especially when the concentrations are passed to a chemical model which calculates changes in concentration by chemical reaction, which would be highly confused by negative concentrations. The scheme may be regarded as being highly accurate, at least in comparison with the scheme used to advect the main model variables, but this accuracy comes with quite a high cost.

This section will cover the UMUI aspects of using tracers, some guidelines for getting them into the model (i.e. ancillary fields) and a brief introduction to aspects of the code as a start for people wishing to build on the system.

4.5.2 UMUI aspects

Tell the system you want to use tracers

For *atmosphere* tracers select window **atmos_Config_Tracer**
```
Atmosphere
-> Model Configuration
-- => Tracers
```

For users running the ocean or the coupled (Atmosphere/Ocean) model, there is also a list of *ocean* tracers to choose from in window **ocean_Tracer_UserDef**
```
Ocean GCM
-> Tracers
--> User Defined Tracers
```
Ocean tracers are not widely used yet and this note concentrates on atmosphere tracers.

Through the first question in both windows, select whether tracers are to be included in the Atmosphere model and/or the Ocean model.

After this question, there is a table that can be scrolled to select the tracers to be used. Tracers are selected by inserting one of the following values against the relevant tracer and the values (> 0) indicate how the tracer fields are to be included.

- 0 : Do not use the tracer.
- 1 : Include from an input dump.
- 2 : Configure from an ancillary tracer file.
- 3 : Update from aerosol climatology.

Notes: The chemical identification of the tracers is purely for the benefit of the stratospheric chemists, the biggest users up to recently, and has no relevance to the code. The main identifying numbers are the section 0 STASH item codes, which are needed when creating initial concentrations.

After the Tracer table, there are two questions.

One asks for the number of model levels to have tracers. This option was first added for the benefit of the stratospheric chemists so if you select a number of levels less than the total in the model *they are assumed to be at the top of the model*.

The other question selects whether the tracers should only be advected or whether they should be mixed in the vertical by the boundary layer scheme as well. It is possible, via private mods, to select a subset to be mixed, but the UMUI currently sets either all or none.

Tell the system what sort of tracer advection to use

There is currently only one version (1A) available. To advect atmosphere tracers this version must be included through window **atmos_Science_Section_Atradv**

```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choice
---> Atmospheric tracer advection
```

There is also a choice in this window whether to use the **Van Leer** or the **Superbee** limiter in the code. A discussion of the options for the limiter can be found in UMDP 11 [6].

Finally, there is a question offering to run the tracer advection scheme for thetal and qt. This option is experimental and can be used particularly for mesoscale applications. Potential users are advised to contact NWP Division before using this option.

Tell the system where to get it's initial data from

If data are to be introduced from an external ancillary file (normally the case) the directory and filename are specified in window **atmos_InFiles_OtherAncil_TracAer**

```
Atmosphere
-> Ancillary and input data files
--> Other ancillary files and Lateral boundary files
---> Tracer and aerosol ancillary file
```

This window allows ancillary data to be provided for three tracers : Soot, Sulphuric Acid and Ammonium Sulphate.

4.5.3 Initial Condition Files

Initial data (unless, of course, they already exist in the dump) are introduced at the reconfiguration step and must therefore be available in the form of an ancillary file. Creating such a file is, in fact, relatively straightforward, but depends, of course, where the data originate and how they are generated. It is therefore impossible to outline a general method. Programs exist to generate such files from pp format files, and the user is referred to the ancillary file group in NWP Division for advice. Points to note, however, are:

- Ensure that the STASH, pp and grid codes are correct.
- The reconfiguration step checks that the verification time of the data matches that of the dump being reconfigured into. It is therefore difficult to include a general 'test blob' without recreating the ancillary field. A modset can be generated to overcome this.
- It is important that the level dependent constants for the levels and the level thicknesses match the dump.

4.5.4 Model aspects

Once inside the model dump, tracers will be advected and may be output by STASH using the appropriate variables in section 0. For those who simply wish to use tracers as markers this will suffice. For those who wish to use the tracer variables as the basis of, say, a chemical or dispersion model, hooking in your own routines is straightforward.

The tracer data are stored in the dump array D1 on the p grid, the number of levels being defined by TR_LEVELS and the number of tracers by TR_VARS. The first data point on each level is pointed to by the pointer JTRACER(LEV,JVAR), where LEV refers to level and JVAR to the tracer variable. Each level is stored immediately after the last, so it is safe, for example, to pass tracer data to a subroutine expecting multi-level data by referring, in the CALL statement, to D1(JTRACER(1,JVAR)). It is thus very straightforward to pass tracer arrays to 'physics' routines from the control level.

Note, JVAR goes from 1 to TR_VARS and they are ordered as selected in the UMUI. The 'true' identity can be found by looking in the array A_TR_INDEX, stored in the common block ATRACER which is declared in the COMDECK CTRACERA. This contains the 'local' (JVAR) value for each tracer in use. It is thus feasible (though probably unnecessary) to write code which uses absolute tracer identities.

4.5.5 Additional remarks

A few systems have been built using the tracer variables which the potential user should be aware of, as some UMUI support exists for these which may be of use for experiments. Systems currently known about are:

- Stratospheric/middle atmosphere chemistry. From UM version 3.4 the facility exists to assimilate data from atmospheric measurements of chemical species. The user is referred to members of the appropriate group for information.
- Sulphur Cycle Chemistry. From UM version 4.1 the facility exists to run the Sulphur Cycle as a a separate chemistry section within the UM. This can be run with or without the addition of other free tracers as described in this section.
- From UM version 3.4 an aerosol variable exists which is advected using the tracer scheme, and may (optionally) have sources and sinks applied. This is used to estimate visibility, and may be adjusted by assimilation of visibility data. In limited area models the boundaries are not updated from the boundary conditions file.

4.6 Selecting a New LAM Area

Author: Richard T.H. Barnes, Last Updated: 18 Feb 98

4.6.1 Preliminaries

Setting up the Unified Model for a new domain is relatively straightforward, and will enable you to run higher resolution experiments concentrating on your area of interest. However you should at the outset consider the time and effort involved.

The time will be 2 days to a week or so of dedicated work, to get a smoothly running Limited Area Model on a new domain, depending on experience. A similar length of time may be needed to create new ancillary files for the new domain. In addition, because of the work involved in preparing acobs files, running with data assimilation on a new domain requires further work. If this is required, consult section 4.8.

Initial thoughts about what you want from your experiment should include:

- the size of domain, horizontal resolution, number and spacing of levels
- the model timestep and number of dynamics sweeps per timestep
- the choice of physics versions
- the order and coefficients of diffusion; these will depend on grid length and timestep
- the availability of initial data and boundary conditions
- the length of forecast
- whether ancillary files are available or will need to be created
- what STASH diagnostic output you will require
- what computer resources you have available

4.6.2 Overview

The work required can be split into 4 areas, some of which can be carried out in parallel. They will be described in detail below:

1. Choose new LAM domain
2. Create new ancillary files
3. Set up and run UM to generate boundary data
4. Set up and run UM to produce new LAM forecast

Tasks 2 and 3 can be done in parallel, along with the first stage of task 4.

4.6.3 Choose new LAM domain

The most convenient method is to use the “lampos” graphical utility. For UK Met. Office users, this is available on the NWP HP-system. Otherwise it may have to be done by spherical trigonometry or by trial and error.

The following 8 data items are needed to define any Unified Model domain, and they are used in the subsequent steps for generation of boundary conditions and setting up your new model domain:-

- latitude and longitude of the rotated pole
- x and y direction gridlengths (lampos assumes these will be equal)
- x and y grid dimensions in gridlengths
- latitude and longitude of the “top left corner” of the domain (relative to the rotated pole)

Conceptually it is simplest to think of situations where the equator/meridian crossing of the rotated grid (“+” on the lampos display) is roughly in the centre of the domain, as with the old UKMO LAM and new UK mesoscale models. In the Northern Hemisphere, the latitude of the rotated pole in degrees is given by (90 - latitude of rotated equator) and the longitude of the rotated pole by (180 + the Eastward longitude of the rotated meridian). Eg. for the old UKMO LAM the equator/meridian crossing is at 60 N 20 W, so the coordinates of the rotated pole are 30 N 160 E.

However it is not essential for the rotated equator/meridian point to be within the domain, and virtually any domain location and rotation can be achieved by moving the rotated meridian outside of the domain and adjusting the coordinates of the top left corner accordingly. Navigating the rotated pole to the best location can be quite tricky, and exploring the possibilities with “lampos” is a great help.

In particular, to get into the Southern Hemisphere requires specifying an “imaginary” latitude, greater than 90N, for the rotated pole. The Unified Model system (ancillary file creation and forecast model) will accept and run with such an imaginary latitude, but it has been found that some pp-package based utilities will not then display the correct map background, so beware.

An alternative approach, which gives an identical forecast and a correct map background, is to “mirror” back to a rotated latitude between 0 and 90N by subtracting from 180, eg. 110N => 70N. Then swing the rotated longitude through 180 degrees, and also add 180 degrees to the longitude of the top left corner.

When selecting the new domain, it will save time on tuning the model if you choose a tried and tested resolution for which diffusion coefficients etc. are known. These are 0.4425 degrees for the old UKMO LAM, 0.15 degrees for the old UK Mesoscale model and 0.11 degrees for the new high resolution mesoscale.

4.6.4 Create new ancillary files

This step is not essential as you could simply reconfigure (see section 4.3) from a global dump to get initial conditions for your run. However, the resulting land/sea mask, orography and other fields so interpolated will be smooth and lacking in detail. For your new domain, you will almost certainly want to create new land/sea mask and orography files at your chosen resolution. Most people also make new soil and vegetation ancillary files for their new LAM domain.

Atmosphere ancillary files for a new domain can be created using the Central Ancillary File Making Program which has gathered together all the currently available options into a single flexible package. For details on its use see UMDP 73 [25]. A script to run this program and generate one or more ancillary files can be obtained by taking a copy of the Cray T3E file /u/um1/vn4.4/ancil/atmos/scripts/master/ancil_top_level for version 4.4. The script is run on the Cray T3E and among its output includes simple maps of land/sea mask and orography for you to inspect. N.B: Make a note of the 'No of Land Points' as this value is required by the UMUI when setting up a forecast run.

This program will probably not be available directly to external UM users. Please seek advise and assistance from Portable Model support at the UK Met. Office.

Before using ancilary files for a new domain, it is advisable to have a careful look at the land/sea mask, which influences all the other ancillary files except ozone, to make sure it meets your requirements. In particular, while single grid point islands or sea inlets will not cause model failures, they could give rise to unsightly or unrealistic effects in surface and boundary layer fields.

Make sure you enter values to exactly the same precision as you will use in the UMUI when setting up experiments. Otherwise your runs may fail internal consistency checks in the model. Also make sure longitudes are in the range **0 - 360 degrees** as this is what the UMUI requires.

4.6.5 Set up and run UM to generate boundary data

This step is not essential as the Unified Model can run as a limited area model with fixed boundary conditions, but, except for idealised experiments, beyond a few hours the forecast will become unrealistic without properly generated boundary data.

Typically you will run a global model to provide boundary conditions for a limited area model (LAM). However, it is possible to drive higher resolution LAMs from another LAM. Eg. the old UK Mesoscale model was driven by boundary data generated by the old UKMO LAM.

To copy an existing or standard (see section 4.2.4) global experiment, use mouse clicks to highlight the job to copy and an experiment to receive it. Use the “Job” & “Copy” commands at the top of the GHUI entry window of the UMUI system. Enter a suitable description and select an unused identifier. When the copy has completed, unhighlight those 2 lines and highlight the new one. Use “File” & “Open read/write” to open your new experiment. All the windows you need to enter are listed below:

Window: **subindep_Runlen**
```
Sub-Model Independent
-> Start Date and Run Length Options
```
Set start time for this run and length of forecast to generate enough boundary data for your LAM run.

Window: **atmos_Control_OutputData_LBC1**
```
Atmosphere
-> Control
--> Output data files (LBCs etc)
---> LBCs out
```
Click on Stream 1 (or the next available stream if you wish to preserve what is already active) and click on “next” to open the data entry window **atmos_Control_OutputData_LBC2**. Set:

- whether data are to be packed (usual on Cray C90) or not (probably necessary on other platforms).
- the frequency, start and end times for boundary data generation.
- the 8 data items specifying the domain (see section 4.6.3).
- the rimwidth in gridlengths of the boundary data (usually 4) - this gives more realistic fields near the boundaries than when boundary data are applied only at the edgemost points.
- whether you want files re-initialised - usually no, except for long climate LAM runs.

Press “next” for a further window, and specify whether the vertical grid for the LBCs is to be different from the driving model - if it is, seek guidance in specifying a suitable set of “eta half levels”. Only

- 19 levels (as in the old global & old UKMO LAM)
- 30 levels (as in the new high resolution global)
- 31 levels (as in the old UK mesoscale) and
- 38 levels (as in the new higher resolution mesoscale)

have been extensively tested. See also section 4.4.4 on vertical resolution.

Boundary data will be generated in a file named *runid*.alabcou*n*, where *runid* is the 5 character code of your experiment id and job id, and *n* is the output stream selected.

If you want to start your LAM run at other than T+0 of this run, you may want a dump from this run at, say, T+6. Go to window **atmos_Control_PostProc_DumpMean**

```
Atmosphere
-> Control
--> Post-processing, Dumping & Meaning
--=> Dumping & Meaning
```

to request this, and click on “next” to open the data entry window **atmos_Control_PostProc_DumpMean2** for irregular dump times.

The rest of this job is no different from a normal global run.

Make sure you “Save” and “Process” before you “Submit” the job.

4.6.6 Set up and run UM to produce new LAM forecast

To copy an existing or standard (see section 4.2.4) LAM or mesoscale experiment, use mouse clicks to highlight the job to copy and an experiment to receive it. Use the “Job” & “Copy” commands at the top of the GHUI entry window of the UMUI system. Enter a suitable description and select an unused identifier. When the copy has completed, unhighlight those 2 lines and highlight the new one. Use “File” & “Open read/write” to open your new experiment.

Use window **subindep_Runlen**

```
Sub-Model Independent
-> Start Date and Run Length Options
```

to set your run start time and forecast length, making sure you do not go outside the timespan covered by your boundary data.

Go to window **atmos_Domain_Horiz**

```
Atmosphere
-> Model Resolution and Domain
--> Horizontal
```

and enter the 8 data items which specify your domain (see section 4.6.3). Set rimwidth the same as you set in the job generating your boundary data (usually 4). The question about “mesoscale” is irrelevant as it only affects assimilation defaults. Also enter the number of land points in the land/sea mask as given by the ancillary creation job (see section 4.6.4), or by a previous run of the reconfiguration program.

If you have chosen a non-standard set of levels, go to window **atmos_Domain_Vert**

```
Atmosphere
-> Model Resolution and Domain
```

```
--> Vertical
```
and enter their eta half levels here - they must be identical to those specified for generating your boundary data.

If you have chosen a non-standard horizontal resolution, you should revise your choice of diffusion coefficients at window **atmos_Science_Onlevel**
```
Atmosphere
-> Scientific Parameters and Sections
--> Level-by-level Constants, Physical Constants
```
See section 4.4.3 on horizontal diffusion for help with this.

Check window **atmos_Assim_General**
```
Atmosphere
-> Assimilation
--> Parameters, Control
```
and select No assimilation, unless you have produced valid acobs files for your domain.

Under the section
```
Atmosphere
-> Ancillary and input data files
```
you can specify your initial data dump, boundary data file, and any ancillary files created for your new domain, eg:

- Window: **atmos_InFiles_Start**
  ```
  Atmosphere
  -> Ancillary and input data files
  --> Start Dump
  ---> Start Dump
  ```
  Specify initial dump file, and you will usually want to use the reconfiguration.

- Window: **atmos_InFiles_PAncil_Soil**
  ```
  Atmosphere
  -> Ancillary and input data files
  --> Climatologies & potential climatologies
  ---> Soil: VSMC, hydrological/thermal conductivity etc
  ```
  Specify soil ancillary file (if you have one), with “Configured” for each variable present (but “Not used” for Saturated soil water suction, unless multi-level hydrology has been selected).

- Window: **atmos_InFiles_PAncil_Veg**
  ```
  Atmosphere
  -> Ancillary and input data files
  --> Climatologies & potential climatologies
  ---> Vegetation
  ```
  Specify vegetation ancillary file (if you have one), with “Configured” for each variable.

- Window: **atmos_InFiles_OtherAncil_LBC**
  ```
  Atmosphere
  -> Ancillary and input data files
  --> Other ancillary files and Lateral Boundary files
  ---> Lateral boundary conditions
  ```
  Specify boundary data file (directory and filename from generating job), also specify rimwidth (eg. 4) and weights (eg. 1.0, 0.75, 0.5, 0.25). This weighted application of boundary data over several edge rows gives more realistic fields near the boundaries and better forecasts than just applying single boundary values to the edgemost points.

- Window: **atmos_InFiles_OtherAncil_Orog**
  ```
  Atmosphere
  -> Ancillary and input data files
  --> Other ancillary files and Lateral Boundary files
  ---> Orography ancillary file and fields
  ```
  Specify orography ancillary file, and select orography to be configured plus standard deviation and orographic roughness as appropriate. Operationally we allow the 8 edgemost boundary gridpoints to retain values interpolated from the initial data, to assist in the smooth application of boundary data.

- Window: **atmos_InFiles_OtherAncil_LSMask**
  ```
  Atmosphere
  -> Ancillary and input data files
  --> Other ancillary files and Lateral Boundary files
  ---> Land-Sea-Mask ancillary file and fields
  ```
  Specify land/sea mask file and to be configured.

All other windows should be as for an ordinary forecast run.

Make sure you “Save” and “Process” before you “Submit” the job.

4.6.7 Advice

Having set up your new experiment, it is best to test it with a short run (but long enough to cover all important features, eg. STASH output or ancillary updating) and check the output, before committing yourself to using large amounts of computer time for your full runs.

4.7 Science Options

Author: Rick Rawlins, Last Updated: 19 Mar 98

4.7.1 Fundamentals

The key to the UM system is that it supports use in many configurations, with different choices of functionality and science code all available via the UMUI, as well as being a development test-bed for prospective new schemes. Selection of different science options is enabled in 3 ways:

1. Run-time variables. Logical switches and numerical parameters are initialised as Fortran NAMELIST variables. This is the preferred method since it reduces extra compilation needs and requires less testing.
2. Compile-time switches controlled by nupdate *DEF directives (see section 3.1). This is easier to program initially and ring fence from pre-existing code but tends to lead to code complexities over the longer term and has shortcomings for testing.
3. Section Version-Releases. Only adopted for atmosphere model code. See next items. More work is needed to interface a new scheme but the structure is more easily maintained over longer periods.

4.7.2 Sub-models, Internal Models and Sections

From a systems point of view, a UM experiment can be described by an hierarchy of Fortran components forming a composite structure that defines the model as a whole. The model may be composed of one or more sub-models, such as atmosphere, ocean or wave sub-models, which each have their own data space, holding respective state variables. Within a sub-model there may be one or more internal models, such as atmosphere and slab internal models within the atmosphere sub-model, which are distinct entities with their own time-stepping sequence.

The organisation of code is projected onto the idealised picture above such that each sub-model allocates its own state variables within separate definitions of the main data array, ie D1 in the Fortran code, and generates separate dumps of data - this distinguishes sub-models from internal models. For the simplest case of a single atmosphere or ocean model, the extra complexity of sub- and internal model identities is often redundant.

Sections

Sub-dividing an internal model further, a number of separate sets of calculations, such as for radiation or boundary layer processes, are included lower down the calling tree and these are termed “sections”. This label is used in 2 ways:

1. To group related routines together in practical sets, for source code management and ownership, and for partitioning pre-compiled object code.
2. To specify groups of diagnostic variables calculated in these routines, to be referenced by the internal addressing and diagnostic system: STASH (see section 4.14). Each prognostic and diagnostic variable is indexed by (internal model, section, item) in a master file.

The term has been extended to include specific sets of routines for calculating diagnostics, such as atmospheric dynamics diagnostics (section 15); and also sets independent of internal models, eg for I/O operations (section 95), which have no application for labelling indices of diagnostic variables.

Scientific sections for the atmosphere internal model can be accessed from the UMUI via

```
Atmosphere
-> Scientific Parameters and Sections
--> Section by section choices
```

A list of sections current at vn4.4 is given in Table 4.1.

Section label	Description
A01	Short-wave radiation
A02	Long-wave radiation
A03	Boundary layer
A04	Large scale (stratiform) precipitation
A05	Convection
A06	Gravity wave drag
A07	Vertical diffusion
A08	Surface hydrology
A09	Cloud scheme - stratiform cloud amount
A10	Dynamics adjustment
A11	Tracer advection
A12	Dynamics advection
A13	Dynamics filtering and diffusion
A14	Energy budget and energy adjustment
A15	Dynamics diagnostics
A16	Physics diagnostics
A17	Sulphur cycle chemistry
A18	Atmosphere assimilation of observations
A19	Vegetation scheme
A70	Radiation service routines
A71	Control level routines for atmosphere
A87	Zonal mean print routine
C70	Control level routines for all sub-models
C72	Control level routines for atmosphere/ocean coupled models
C80	Model dump file I/O routines
C82	Ancillary field initialisation and updating
C84	STASH diagnostics service routines
C90	General configuration-specific service routines
C91	Fourier filtering and workstation specific routines
C92	Vertical, horizontal (including LAM) and time interpolation
C93	Mathematical service routines
C94	Miscellaneous service routines
C95	All C code: low level I/O and portable alternative routines
C96	MPP service routines
C97	UM timer routines
C99	OASIS coupler routines
O35	Ocean assimilation of observations
S40	Slab model scheme

Table 4.1: Unified Model Sections

Section Version-Releases (atmosphere only)

From the UMUI, you will note that some sections have several alternatives, defined by version and release, abbreviated to v<r>. This is to allow groups of users to have access to different choices for these sections. For example some users need to be able to access the latest developments in gravity wave drag schemes, others need to be consistent with work started much earlier. This system also allows externally written versions of the sections to be imported for testing.

For the UM as a whole, the terms 'version' and 'release' are used synonymously. However for labelling alternative code within the same section, their meaning is more specialised. Every allocated section contains at least one version of the code, normally version 1A (i.e. version 1, release A). When there are significant conceptual differences, these are denoted by a new version number (1,2,3 etc). Smaller differences would be denoted by a change in the release letter (2A, 2B, etc).

A section v<r> is defined by a series of DECKs in the source code, but note that some DECKs are shared across several v<r>s. The naming convention for the *DEFs define internal model, section and it's v<r>: A01_2B would be version 2B of section 1 (SW radiation) in the atmospheric internal model. A v<r> *DEF switch can be seen in the nupdate source code at the head of each science routine, providing the mechanism for automatic inclusion of the correct modules at compile time.

Ideally, each section of code would form an independent module and different v<r>s could be selected with any combination of choices for other sections. However in practice the assumptions behind some calculations for different sections are related, so that it only makes sense to use certain combinations of v<r>s. Guidance is given in help panels of the UMUI when a choice of different schemes is presented.

Plug compatability and Glue routines

It is a requirement of the UM system that science schemes should be readily interchangeable with those from other models through the concept of 'plug compatability‘, such that there is no dependency on data structure within the schemes. In practice this requires data to be passed down into science subroutines by argument. This is achieved within the UM atmosphere for science sections, below an interface known as a glue routine. The glue routine separates control and science layers, and allows alternative schemes - potentiallyt with different argument lists - to be accessed for the same section. More details are given in an Appendix to UMDP 4: Maintenance of frozen versions of the UM for climate users [4].

4.7.3 Frozen and Kept Versions for Met. Office users

One v<r> for each section is denoted a frozen version, defined as the default climate version in the UMUI. This v<r> must give the same results from one version of the model to the next. Even if an error is found, code changes to this v<r> can only be made as long as they do not affect the results of prognostic variables. This rule ensures that long running climate models can be repeated.

Periodically, the definitions of the frozen v<r>s for all sections of the model are reconsidered, thus defining a new frozen model (although modifications may be found to be necessary during the testing period). New v<r>s are chosen for the new frozen model, other v<r>s are kept but corrected. This topic is covered further in UMDP 4: Maintenance of frozen versions of the UM for climate users [4].

One version of the UM is designated the kept version, for each frozen model. This UM version is kept on line at the Met. Office and maintained for a long period of time (as space permits).

Portable versions of the UM will map onto the definitions of UM versions given above, but not every version released for Met. Office users would necessarily be packaged as a portable model for external release.

4.8 Assimilation in the Atmosphere Model

Author: Stuart Bell, Last Updated: 6 Jul 98

4.8.1 Introduction

The data assimilation scheme adjusts the model state towards observations, providing initial fields for forecasts. Observational data is assimilated by an iterative or "nudging" procedure which is carried out at each timestep of a separate model integration preceding the analysis time. This technique is referred to as the Analysis Correction scheme. The analysis increments required to nudge the model state towards the observations are computed from the observational increments, weighted by a factor depending on their displacement in space and time, and on the observational density and error. Additional balancing increments are also calculated.

In brief, we start with an analysis and one or more observation files valid for a period beyond the starting analysis and we end with an analysis valid at the time of the last batch of observation. The UM may then continue to run beyond the end of the assimilation period in forecast only mode.

For those who need more than just the basic knowledge of the analysis correction assimilation process, I direct you to the UM documentation papers:

- UMDP 30: Analysis Correction Data Assimilation scheme [16]
- UMDP P3: Technical Details of the Unified Model Data Assimilation System [36]

This technical documentation is essential if you need to run anything more than the default configuration.

4.8.2 The Observations

Observations are obtained from the Observation Processing system (OPS). These are often called ACOBS files. They are tailored to the required UM configuration. Each ACOBS file contains all the observations within a time window (usually 6 hours for global and 3 hours for limited area configurations) which are required for the assimilation. The observations within an ACOBS file are grouped into types. These AC_OBS_TYPES (see section 4.8.3) can be selected via the appropriate UMUI panel as desired.

ACOBS files are created by the observation processing system (OPS) which takes as input, the unprocessed observations from the MetDB archive and a background QC fieldsfile from a UM run at the desired configuration. Alternatively, the operational ACOBS files are available in the operational archive. However users of the archive should be aware that ACOBS files are configured to the target resolution and also some observation processing (eg for GLOSS) makes use of the model background and significant changes to the model formulation will influence the processed observations.

For internal Met Office Users, OPS scientific documentation and OPS technical documentation is available online.

4.8.3 AC Observation Type Numbers

P*

- 101: Pressure at model surface

Potential Temperature

- 201: Radiosondes, on model levels
- 202: Surface Temperature from Ships
- 203: Single, upper level temperatures (aireps)
- 204: Surface Temperature from Synops
- 205: GLOSS Temperature (satellite soundings processed in-house)
- 206: Layer mean temperatures from 500km Satems
- 207: Layer mean temperatures from 120km Satems
- 208: GLOSS Temperature with first guess temperatures
- 209: CFO Bogus thickness (1000-500mb) observations (processed as layer mean virtual potential temperature - only available in ACOBS files processed using the old OPS pre-Jan1998)
- 211: UARS Temperatures

Winds

- 301: Radiosondes, on model levels
- 302: Surface Winds from Ships
- 303: Single, upper level winds (aireps, satobs and CFO bogus)
- 304: Surface Winds from Synops
- 305: Surface Winds from the ERS-1 Scatterometer (SCATWINDS)
- 306: Surface Winds from Buoys
- 311: UARS Winds

Relative Humidity (RH)

- 401: Radiosondes, on model (wet) levels
- 402: Surface RH from Ships
- 403: Single, upper level RH (CFO Bogus observations)
- 404: Surface RH from Synops
- 405: GLOSS RH (satellite soundings processed in-house)
- 406: RH from MOPS
- 407: AVHRR cloud histograms

MOPS

- 506: MOPS rain

UARS Chemical Data

AC Observation Types 601-629 are reserved for various chemical tracers observed by UARS.

Surface Visibility

- 901: Surface Visibility

4.8.4 Relevant UMUI Details

At build 4.4, the panels relating to the assimilation contain references to the variational analysis options. Please ignore all these references. We don't anticipate releasing the 3DVAR system until Autumn 1998. The following panels are the ones relating to assimilation with the most convenient route given. There are full help options within the UMUI so only a brief commentary is given here.

1. Window: **subindep_Runlen**
   ```
   Sub-Model Independent
   -> Start Date and Run Length Options
   ```
   The run time should be the SUM of the assimilation period and any subsequent forecast period. When starting in assimilation mode, the start date/time may be set to 0.

2. Window: **atmos_Assim_General**
   ```
   Atmosphere
   -> Assimilation
   --> Parameters, Control
   ```
   Select version 1A and run in Analysis Correction mode with Standard macro. The assimilation time details (hours after basis time) are specified as follows:

Start	= x,	usually x=0
End	= y,	y=z+4 for global (z+2 for MES)
Nominal analysis time	= z,	z=(assim cycle length)*(no. of assim cycles beyond start)

   A few general options about the assimilation characteristics may be specified, particularly those dependent on the configuration. For the experts, many detailed tuning option can be made available by providing an additional namelist details, taking guidance from UMDP P3 (see [36]).

3. Window: **atmos_Assim_Obs**
   ```
   Atmosphere
   -> Assimilation
   --> Observation choices
   ```
   This panel allows the user to choose the AC_OBS_TYPES (see section 4.8.3) required (presuming of course that they were included in the ACOBS file when the OPS was run). The observations are grouped together during the assimilation stage. This grouping is flexible but it is anticipated that nearly all users will wish to group as in the operational configuration. The crib in column 2 (GLMS) indicates which observation types are contained within each of the operational groupings (a '.' implies the observations are not used in a particular configuration; G=global,L=limited area, M=mesoscale and S=stratospheric). The crib in column 3 identifies which potential groupings allowed.

4. Window: **atmos_Assim_DivDamp**
   ```
   Atmosphere
   -> Assimilation
   --> Divergence damping in the assimilation
   ```
   Appropriate divergence damping coefficients should be set for assimilation mode (resolution dependent values usually mirror those used operationally which are given in the UMUI help panel).

5. Window: **atmos_Science_Section_Dassim**
   ```
   Atmosphere
   -> Assimilation
   --> Data assimilation, section choice
   ```
   This is also set within window **atmos_Assim_General**. Use version 1A of the assimilation code.

6. Window: **atmos_InFiles_ObsData_ACObs**
   ```
   Atmosphere
   -> Assimilation
   --> ACOBS data (and IAU files for the 3DVAR alternative option)
   ```
   Specify the path of the observation files, up to a maximum of 10. Usually one more file is required than the number of observation windows up to the analysis time (the extra file covering the period around the start analysis time. There is a new option to specify directories containing sets of ACOBS files created from the new OPS rather than individual files. These files (one per observation group) follow a pre-defined naming convention.

7. Window: **atmos_STASH_Macros_BackG**
   ```
   Atmosphere
   -> STASH
   --> STASH Macros
   ---> Background quality control macro
   ```
   Click standard macro to specify that a background fieldsfile is required for use by the OPS in a subsequent cycle. It is not relevant for single cycle runs. The global standard STASH Macro is appropriate to the new OPS. The standard macro for other configurations may become inappropriate for use with operational ACOBS files during the lifetime of the UM4.4 build when the use of the new OPS is extended to those other configurations. In this event the development macro should be used.

8. Window: **atmos_STASH_Macros_Assim**
   ```
   Atmosphere
   -> STASH
   --> STASH Macros
   ---> Internal assimilation macro
   ```
   This is also set within window **atmos_Assim_General**. Click standard macro if running an assimilation to make non prognostic variables available to the assimilation during the model run.

9. Window: **atmos_STASH_Macros_MOPS**
   ```
   Atmosphere
   -> STASH
   --> STASH Macros
   ---> MOPS output macro
   ```
   Click standard macro if you intend to process MOPS data in a follow-on cycle. This is unlikely outside the operational environment.

4.9 Ancillary and Boundary Files

Author: Richard T.H. Barnes / David Robinson, Last Updated: 18 Feb 98

4.9.1 Introduction

What is an ancillary field? This is not as easy a question as it seems, because whether a particular field is an ancillary field depends on the context in which it is used in a given configuration of the Unified Model. One suitable definition is:

> A prescribed "prognostic" variable held in a model dump, which may be updated during a UM run (with appropriate time interpolation) by values read from an external ancillary file.

For example, snow depth is an ancillary field if it is imposed and either held constant or updated from an ancillary file, but if (as is also possible) its value is allowed to be predicted by the model's precipitation and hydrology schemes it is not an ancillary field.

Some fields, such as orography and the parameters which classify soil and vegetation characteristics, are at present ancillaries in all circumstances. However, introduction of, for instance, a scheme for interactive vegetation could change this.

How is an ancillary field got into a UM run? There are 3 basic ways:

- It may be in the initial dump already. Most data fields found in ancillary files are generally already available in initial dumps. Note that if the dump is reconfigured to a new resolution and/or domain (see section 4.3) then the ancillary fields will also be interpolated to the new grid.
- It may be replaced in the initial dump by configuring in from an ancillary file on the correct domain and resolution.
- It may be added to the initial dump as a new ancillary field by configuring in from an ancillary file on the correct domain and resolution. This is usually necessary when trying out a new parameterisation scheme such as moving from single-level hydrology to MOSES hydrology.

Details of the above three procedures can be found in section 4.9.4.

Details on creating new ancillary files can be found in section 4.9.3.

The user should also be aware of the following:

- There are advantages of using ancillary files for the correct resolution and/or domain. The user has more control of the data used and the interpolation done within the ancillary creation program is better as certain fields are checked against one another for consistency (vegetation and soil parameters for instance). Also, some fields especially the land-sea mask are edited by hand to reduce the amount of noise in the lower boundary layer.
- For initial experiments of new areas the interpolation performed within the reconfiguration is usually sufficient and it only becomes necessary to use ancillary files when the new area is finalised.

Some comments on boundary updating files, which are essentially a special form of ancillary file, will be given in section 4.9.6.

4.9.2 Location of standard ancillary files

Ancillary files exist for each standard Unified Model domain and are available for general use. Most of these files have been interpolated from raw data at 1 degree resolution, though those for orography and land-sea mask are from global 10 minute data or higher resolution data for certain parts on the world. For a full list of files and data sources see UMDP 70 [24]. At the UK Met. Office all the files are stored on the Cray T3E supercomputer in a directory structure of the form:

$UMDIR/vn*n.m*/ancil/*submodel/domain*/qr*type.data.suffix*

- $UMDIR (= /u/um1 on the Cray T3E) is the Unified Model 'home' directory
- *n.m* is Unified Model version number
- *submodel* is one of atmos, ocean, etc.
- *domain* is one of the standard model domains, thus:
  - cl_9673 for 96x73 grid points climate model,
  - gl_432325 for 432x325 grid points high resolution operational global model,
  - gl_288217 for 288x217 grid points former operational global model,
  - la_229132 for 229x132 grid points former operational Limited Area Model,
  - ms_9292_uk for 92x92 grid points former operational UK mesoscale model,
  - ms_146182_uk for 146x182 grid points high resolution operational UK mesoscale model,
- *type* indicates if the data have any time dependence, thus:
  - parm for constant fields, eg. orography,
  - clim for climatologies, eg. sea surface temperature,
  - m if valid for one particular month,
- *data* indicates the file contents, eg. mask, orog, sst, veg, soil
- *suffix* is optional

N.B: for climate users - 'clim' files in the cl_9673 directory have headers set for the standard climate 360 day year, whereas files in the cl365_9673 directory have headers set for the Gregorian 365 day year. All other domains only have files for the 365 day year.

4.9.3 Generation of new ancillary files

Atmosphere ancillary files for a new domain can be created using the Central Ancillary File Making Program which has gathered together all the currently available options into a single flexible package. For details on its use and examples to run the program see UMDP 73 [25]. The script is run on the Cray T3E and among its output includes simple maps of land/sea mask and orography for you to inspect. N.B: Make a note of the 'No of Land Points' as this value is required by the UMUI when setting up a forecast run.

This program is not made directly available to external UM users. Please seek advise and assistance from Portable Model support at the UK Met. Office.

For ocean ancillaries the usual procedure is to convert a pp-file into UM ancillary file format using the utility program PPTOANC.

Before using ancillary files for a new domain, it is advisable to make a careful scrutiny of the land/sea mask, which influences all the other ancillary files except ozone, to make sure it meets your requirements. In particular, while single grid point islands or sea inlets will not cause model failures, they could give rise to unsightly or unrealistic effects in surface and boundary layer fields. The facility exists to manually change the objective land sea mask.

Make sure you enter values to exactly the same precision as you will use in the UMUI when setting up experiments. Otherwise your runs may fail internal consistency checks in the model. Also make sure longitudes are in the range **0 - 360 degrees** as this is what the UMUI requires.

4.9.4 UMUI aspects

The information in this section will make more sense if you read it in conjunction with an active UMUI session - if necessary copy a standard experiment (see section 4.2.4) for the purpose. Window titles given are those under the section

```
Atmosphere
```

Similar ones exist for the Ocean Model in the section

```
Ocean GCM
```

The section

```
Atmosphere
-> Ancillary and input data files
```

is where ancillary files (if any) are specified. There is a window for each possible ancillary file, most of which contain several fields. These windows control how each field is to be used, both in the reconfiguration step and during the model run. The reconfiguration program is only executed if reconfiguration is selected in window **atmos_InFiles_Start**

```
Atmosphere
-> Ancillary and input data files
--> Start Dump
---> Start Dump
```

The section

```
Atmosphere
-> Ancillary and input data files
--> Climatologies & potential climatologies
```

covers most ancillary files, but the section

```
Atmosphere
-> Ancillary and input data files
--> Other ancillary files and Lateral Boundary files
```

contains orography and land/sea mask files.

Each window has a similar pattern:

- Specify ancillary file as directory and filename. Directory may be a full pathname or an environment variable specified in the window **subindep_FileDir**
  ```
  Sub-Model independent
  -> File and Directory Naming
  ```
- For each field in the file specify whether or not it is to be:
  - Configured, i.e. read in from the ancillary file in the reconfiguration step. If not, values from the initial dump will be used. Note that these will be interpolated if the initial dump is for a different domain and resolution from the model, but only if reconfiguration is selected in window **atmos_InFiles_Start**.

  - Updated, i.e. set to values time interpolated from the ancillary file at a specified interval. If not, values may be computed by the model as it runs (eg. snowdepth) or may be fixed (eg. ozone) - this depends on what is possible for each field - see the particular UMUI panel for details.
  - Not used. In fact this means that, if the field is required by the model, it will be taken from the original dump.
- For some fields, notably orography and land/sea mask, updating is not a valid operation, so the only choice is to configure or not.

Warnings

Before asking for ancillary files to be used, you should consider whether it is necessary and what the implications are.

If the model run you are setting up is on the same domain and resolution as your initial dump, then many or all of the ancillary fields may already be there. If they are not and you wish to read in data from an external file, then ensure the file was made for your domain and resolution. Be aware that, up to and including version 3.3, if reconfiguration of orography and/or land-sea mask was requested, then interpolation routines were be called for the prognostic variables, thus negating bit-comparability, even if domain and resolution are not changed. From version 3.4 onwards, checks are included in the reconfiguration to suppress horizontal interpolation if land-sea mask is identical to that in the dump, and to suppress vertical interpolation if orography and model levels are identical.

If you wish to run with a domain and/or resolution different from your initial dump, ancillary fields in your dump (and prognostic variables too) will be interpolated. Is the resultant loss of detail acceptable to you? If not, then you must create and configure from ancillary files on your domain and resolution.

Window: **atmos_InFiles_Options_Headers**
```
Atmosphere
-> Ancillary and input data files
--> In file related options
---> Header record sizes
```
allows you to “specify the total number of lookup headers on the ancillary files used for updating” for atmosphere (and similarly for ocean). This is typically set to 300 in the standard experiments, which is usually plenty, but if you are using a multi-level, multi-time ancillary file (eg. an aerosol climatology) you may need to increase this value.

Window: **subindep_AncilRef**
```
Atmosphere
-> Ancillary and input data files
--> In file related options
---> Ancillary reference time
```
OR
```
Sub-Model Independent
-> Ancillary reference time
```
allows you to specify a reference time from which all ancillary field updating times will be computed. Setting all zeroes allows the dump basis time to be used, but climate users will often want to specify a convenient reference time. Specifying a fixed ancillary reference time ensure bit comparison when starting a new run from a dump made part way through a previous run.

4.9.5 Model aspects

This is difficult to document simply, but fortunately a new user does not normally need to know the details.

The interpolation, replacement or addition of ancillary fields by the stand-alone reconfiguration program is covered separately (see section 4.3).

Within the model itself, control of ancillary fields is supposedly by namelists in the umui_jobs directory files CNTLATM. &ANCILCTA and optionally one or more &UPANCA namelist, depending on which fields (if any) are to be updated, are used for this purpose. However, the only thing it really controls is whether to update a specific field and, if so, at what frequency. The equivalent for the ocean model is &UPANCO in CNTLOCN.

Whether the data are single-time or multi-time, and whether updating is regular/irregular, periodic/non-periodic or time-series, are controlled by the ancillary file header. See UMDP F3 [34] for details.

Whether the calendar is Gregorian or 360 day is controlled by the logical LCAL360 set in window **subindep_Control**
```
Sub-Model Independent
-> General Configuration and Control
```
(Climate users must be careful to use ancillary files created for the right year-length.)

All other controls, such as:

- inter-field dependencies, eg. SST and sea-ice,
- time interpolation using simple linear interpolation, but controlled linear interpolation for snow depth and SST & sea-ice fields,
- whether to update land points, sea points or all points,

are coded within the subroutine REPLANCA (and REPLANCO). For full details see UMDP C7 [31].

4.9.6 Boundary updating files

Being an external file which updates model prognostic values during a model run, albeit only at a small number of lateral boundary points, a boundary data file can reasonably be termed an ancillary file.

It is specified in the UMUI window **atmos_InFiles_OtherAncil_LBC**
```
Atmosphere
-> Ancillary and input data files
--> Other ancillary files and Lateral Boundary files
---> Lateral boundary conditions
```
in the same section as orography and land/sea mask.

However, they are generated, not by the central ancillary program, but in a run of a UM experiment covering a larger domain, specifications being set in panels under
```
Atmosphere
-> Control
--> Output data files (LBCs etc)
---> LBC's out
```

The new mixed phase cloud/precipitation scheme introduced as an option at vn4.4 has implications for lateral boundary conditions as follows:-
Runs with the original cloud/precipitation scheme continue to use and generate lateral boundary files containing data for the five main prognostic variables, viz. P*, u, v, thetal & qt.
Runs with the new mixed phase cloud/precipitation scheme will generate LBC files containing data for 6 variables, the standard 5 as above plus qcf - the new cloud ice prognostic variable. Limited area runs with the

new mixed phase cloud/precipitation scheme will expect to receive lateral boundary data with all 6 prognostic variables. However for upward compatibility the window

atmos_InFiles_OtherAncil_LBC
```
Atmosphere
-> Ancillary and input data files
--> Other ancillary files and Lateral Boundary files
---> Lateral boundary conditions
```
contains an additional option ”Define mixed phase cloud compliance”. Choosing the second option, ”Input LBCs forced to be from a non-mixed-phase LS-Rain scheme.”, enables lateral boundary data generated by a run with the original scheme (and therefore lacking cloud ice) to drive a model with the new scheme. In this case, zero cloud ice boundary conditions are implicitly applied. There is no provision for backward compatibility of LBC files from the new to the original scheme.

For a detailed description of the structure and the contents of a boundary update file see UMDP C7 [31], Appendix 2.

4.9.7 User Ancillaries

Introduction

The UM allows users to add new prognostic fields (ie. fields that define the atmospheric state and are required to perform the integration) and new ancillary fields (ie. fields that act as boundary conditions to the integration) - ‘User prognostics’ and ‘User Ancillaries’ respectively. User prognostics may be set up using any item number and can be initialised in the reconfiguration. Setting up user ancillaries is more restrictive and the user must choose from a list of reserved item numbers for single and multi-level ancillary fields.

The UM code has been set up to add user ancillaries through four ancillary files recognised in the UM system by their names as follows:

- USRANCIL : Atmosphere - Single level user ancillaries
- USRMULTI : Atmosphere - Multi-level user ancillaries
- OUSRANCL : Ocean - Single level user ancillaries
- OUSRMULT : Ocean - Multi-level user ancillaries

This set up simplifies the process of adding ancillary fields which normally requires extensive changes to the UM code. Ancillary fields provided through these four files will be recognised and read in; the user then has to provide code changes to use these fields within the model.

How to set up User Ancillaries

The user will need to take the following steps:

- Create a User Stashmaster file (USMF).

  Stashcodes (in section 0) have been reserved for the user ancillaries as follows:

  – USRANCIL : 20 fields, Stashcodes 301-320
  – USRMULTI : 4 fields, Stashcodes 321-324

  - OUSRANCL : 10 fields, Stashcodes 331-340
  - OUSRMULT : 4 fields, Stashcodes 351-354

  Detailed information on setting up a USMF can be found in UMDP C4 [29] (page 13).

- Attach the USMF to the UMUI in windows **atmos_STASH_UserDiags** or **ocean_STASH_UserDiags**
  ```
  Atmosphere OR Ocean
  -> Stash
  --> User-STASHmaster files, Diags, Progs \& Ancills.
  ```
  Several USMFs may be attached. They must be set up before entering STASH and be on the same platform as the UMUI client.

- Set up to initialise the User Ancillaries from data in the ancillary files. This is done in window **atmos_STASH_UserProgs** or **ocean_STASH_UserProgs**
  ```
  Atmosphere OR Ocean
  -> Stash
  --> Initialisation of User Prognostics
  ```
  The user needs to consider whether or not the ancillary field is being updated in the model run.

  - If the ancillary field is NOT to be updated, select option 7 and attach the user ancillary file in the right hand column. The field will be initialised in the reconfiguration step.
  - If the ancillary field IS to be updated, select option 2. The field will be initialised with the ancillary data at timestep 0.

- For ancillary fields to be updated,

  - specify the location(s) of the ancillary files
  - specify an update frequency.

  through the following windows:

  - For atmosphere user ancillaries: windows **atmos_InFiles_PAncil_UserM** and/or **atmos_InFiles_PAncil_UserS**
    ```
    Atmosphere
    -> Ancillary and input data files
    --> Climatologies & potential climatologies
    ---> User multi-level ancillary file and fields
    ```
    and/or
    ```
    ---> User single-level ancillary file and fields
    ```
  - For ocean user ancillaries: windows **ocean_InFiles_PAncil_UserM** and/or **ocean_InFiles_PAncil_UserS**
    ```
    Ocean GCM
    -> Input Files
    --> Climatologies \& potential climatologies
    ---> User multi-level ancillary file and fields
    ```
    and/or
    ```
    ---> User single-level ancillary file and fields
    ```

WARNINGS on setting up user ancillaries

- Note that the UMUI allows one update frequency per ancillary **file** and not **field**. The same updating frequency applies to any fields being updated in the ancillary file. However, if necessary, the updating frequency could be over-written (by experienced users) by hand-editing the ‘PERIOD’ and ‘INTERVAL’ values in the UPANCA/O namelists in the CNTLATM/CNTLOCN files for your processed job.
- Code changes will be required to the REPLANCA subroutine depending on whether the field is on all points, land points or sea points. It is assumed that
  - the atmosphere single level ancillaries will be **for land points only**.
  - the atmosphere multi-level ancillaries will be for all points and the full set of model levels (P_LEVELS).

4.10 Ocean Model

Author: Richard S.R. Hill, Last Updated: 26 March 1998

4.10.1 Introduction

The Ocean Model is one of the sub-models available within the Met. Office Unified Model.

This section is intended to act as a general guide to setting up and running ocean models within the Unified Model system. Ocean models may be run alone, or in coupled mode with a suitable atmosphere configuration. The Ocean model may be run in MPP mode on suitable platforms (eg: Cray T3E) or in non-MPP mode. Whilst mainly aimed at MPP use, much of the ocean model code lends itself to optimisation on PVP platforms. See the section 2.3 for information about how to set up ocean model physics and dynamics parameters.

4.10.2 Setting Things Up

Setting up an ocean model is usually best done by taking a copy of an existing working model and modifying it. Many details will be common to models of similar type. This approach avoids the need to visit every relevant UMUI window. See the sub-section on standard experiments in section 4.2 for more information about copying existing models. Ocean models may be set up from copies of existing ocean only configurations or coupled atmosphere-ocean configurations.

Before you can run

Before the model can be run, there are a number of things which need to be set up and in place and a number of further items which may be required under certain model configurations.

1. Directories containing input files - In the case of the Met Office T3E, there are often existing standard input files and directories which can be referenced directly.
2. Start dump file - A file containing the initial conditions of your model.
3. Ancillary file(s) - Optional files containing information about values of certain fields which may be used to force your model. See the umui at:
   ```
   Ocean GCM
   -> Input Files
   --> Climatologies and potential climatologies
   ```
4. Compiler override file(s) - Optional file defining any special compiler and optimisation options which you require.
5. Fortran and/or C modification(s) - Optional files containing modifications (eg bug fixes) to the base code can apply either to the main model code or to the reconfiguration.
6. Script modification(s) - Optional files containing modifications (eg bug fixes) to the scripts used to run the UM.
7. Lateral boundary condition file - only required in limited area models when updating lateral boundary conditions. See the umui at:
   ocean_InFiles_OtherAncil_LBCs
   ```
   Ocean GCM
   -> Input Files
   ```

```
--> Other ancillary/input files
---> Lateral boundary conditions
```

8. Basin indices file - required when running with MEAD diagnostics enabled. See the umui at:
ocean_Diag_MEAD
```
Ocean GCM
-> Input Files
--> Other ancillary/input files
---> Basin indices file for MEAD Diagnostics
```

9. Observational data files - only required when running with ocean assimilation enabled. See the umui at:
ocean_InFiles_ObsData_ACObs
```
Ocean GCM
-> Input Files
--> Observational data
---> ACOBS data
```

10. User STASHmaster file - required if you are introducing new diagnostic, prognostic or ancillary fields to the model. See section 4.14 for more information. See the umui at:
ocean_STASH_UserDiags
```
Ocean GCM
-> STASH
--> User-STASHmaster files Diags, Progs & Ancils
```

See section 4.2 for more information about getting started with the Unified Model.

The user interface for the ocean model

When setting up the parameters to run an ocean model in the UMUI, there are three main panels which will need to be considered.

They are:

- `User Information and Target Machine`
- `Sub-Model Independent`
- `Ocean GCM`

If running in coupled mode with the atmosphere model, additional panels will need to be considered - see section 4.12 for more information on coupled models.

Most items in the User Information and Target Machine and Sub-Model Independent panels will be common to all sub-models. See section 4.1 for general details on how to fill in the panels of the UMUI.

See section 2.3 for information about ocean model physics and dynamics settings.

When setting up an ocean only model based on a copy of an existing coupled model, you will need to switch off the atmosphere component in umui window:

smcc_Model
```
Sub-Model Configurations and Coupling
->Sub-Model Inclusion switches
```

Visit the Target_Machine window to set the number of PEs you wish to run on **AFTER** switching off the atmosphere component. If you do not do this, the total number of PEs will be set to the number used by the atmosphere in the North-South direction only.

You may also have to specify ancillary forcing fields (in place of values which would otherwise have been available from the atmospheric part of the model) as mentioned earlier.

4.10.3 Computing Resource Considerations

Running in MPP mode

If you are running in MPP mode, you need to decide how many PEs to run on. The MPP ocean model uses a one-dimensional row-wise MPP decomposition. That is, work is distributed by dividing the domain into a number of sub-domains in the south-north (zonal) direction.

Your choice of the number of PEs will be influenced by the memory requirement of your job, how quickly you need results, available resources etc. However, there is a maximum limit to the number of processors which any given ocean model can use.

This number must be ¡= Number of Rows (South-North)/2. This effectively means that each processor must be allocated at least 2 rows for the computation.

For example, if your model consists of 173 rows in the South-North direction, then the maximum number of processors you could use would be arrived at from 173/2 = 86.5 ie: 86 Processors. (In this case PE 0 would be assigned three rows. All other PEs would be given two rows to deal with).

For all but development work, it is worth trying to ensure as even a load distribution as possible. In practical terms, this means trying to ensure that each PE is assigned approximately the same number of rows.

The effect of an imbalance in the number of rows assigned to each PE becomes more significant as fewer rows are assigned to each PE. If 45 PEs are used to run a model with 96 rows, the first 6 PEs will have 3 rows each and the remaining 39 will have 2 rows. Thus, the first six PEs will have 50% more work than the rest. A more efficient use of resources would be to use eg: 32 PEs (3 rows each) or 48 PEs (2 rows each). This ideal situation may not always be attainable where the number of model rows is an awkward (eg: prime) number. When considering coupled models, what works best for the atmosphere may not be so good for the ocean and vice versa.

Bear in mind that since the MPP ocean code uses a regular 1-d decomposition, your flexibility in defining a well load balanced model will be limited. This is partly due to the fact that some rows will unavoidably contain more sea points (fewer land points) than others. This will in turn result in more work for some PEs than others. The atmosphere model does not suffer from this land point problem. (Background note: Use of a regular 2-d decomposition by the ocean model could make this load balance situation even worse. This is because some PEs would be assigned areas containing few or even no sea points. So a 2-d decomposition would allow more PEs to be used and could bring faster run times, but could result in less efficient use of those PEs. Irregular domains would need to be defined to exclude land points).

See section 3.4 for more information on the MPP capability of the UM system.

Three dimensional primary variables

The three dimensional primary variables may be stored in compressed form or uncompressed form. Compressed form means that all data relating to land points is removed from the relevant arrays. When values are required for particular rows, data is accessed by reference to pointers indicating where the data belongs on the model grid.

Uncompressed form means that land point data (effectively dummy values) is retained in the relevant arrays. Uncompressed data consumes more memory and disk space (in dumps) than compressed data. Compression of fields is only an option when running the model in non-MPP mode. So in most cases on the T3E, uncompressed data is used.

Storing in compressed form saves about 50% of the storage required by the 3-d fields in a global ocean model dump. The variables which may be stored this way are: potential temperature, salinity, and currents. Additional three dimensional tracers may also be compressed.

Storage in memory

Two time levels of three dimensional prognostic data are retained for use by the leapfrog timestep scheme. This is achieved by storing the additional values relating to the second time level in memory after the copies from the dump. Pointers switch between these ”past” and ”present” values in memory. Thus, on one timestep the ”present” value might be that stored first. On the next timestep the ”present” values would be stored second.

4.10.4 Ocean Model Control Code

The ocean model control code structure closely mirrors that of the atmosphere model. Indeed, the main high level control routines UM_SHELL and U_MODEL are common to all sub-models. Much of the description of the atmosphere model control code (section3.5) applies equally to the ocean. The following are the major differences.

Control Files

Most control files are relevant to all submodels. The following are the Ocean versions of their atmosphere equivalents.

- CNTLOCN - ocean sub-model specific control
- RECONO - control of the reconfiguration

UM_SHELL

The UM_SHELL program contains the call to DERVSIZE which sets up dimensions used in dynamic allocation of arrays. For the ocean model, many of these dimensions are controlled by logical switches. The switches are set depending on the model configuration being run (eg: L_OBIOLOGY is set to true when the biology model is included). Generally, if a particular switch is set, then the dimensions used to define the size of arrays associated with that scheme will be given their full size. If a switch is set to false, then dimensions are usually set to one. This saves reserving space in memory for arrays which will never be used by the configuration.

In MPP mode, DERVSIZE also contains a call to a routine called OCEAN_SIZES. This routine sets up various dimensions and indices associated with the ocean MPP domain decomposition.

U_MODEL

U_MODEL is called from UM_SHELL. Of particular interest are INITIAL and OCN_STEP.

INITIAL calls INITDUMP which calls a number of ocean model specific control routines:

- SET_OCN_POINTERS - sets up a number of pointers to allow addressing of the various fields in the main D1 data array.
- READNLST_OCN - reads namelist data set up by the umui which mainly concerns parameters associated with various scientific schemes used by the ocean model.
- SET_CONSTANTS_OCEAN - sets up most of the constants used by an ocean model.

OCN_STEP, as with ATM_STEP in the atmosphere, contains the three major sections of the ocean model. That is, it contains the dynamics, physics and (optionally) ocean assimilation.

OCN_STEP controls calls to the ocean assimilation and to OCN_CTL. OCN_CTL controls calls to the three main areas of computation in the ocean model:

- ROW_CTL - the area of code containing the main computations associated with model evolution, Temperatures, salinities velocities, mixed layer depth etc.
- TROP_CTL - the area of code concerned with any barotropic solutions used by the model.
- ICE_CTL - the area of code controlling any ice model calculations.

Unfortunately, not all code relevant to a particular control area is wholly contained within these discrete sections. E.g. certain aspects of the ice model are dealt with in the code controlled by ROW_CTL. This can make development and debugging slightly awkward when dealing with such situations.

4.10.5 Development and Debugging of Ocean Model Code

Many of the procedures pertinent to code development and debugging are equally applicable to all UM submodels. See section 4.18 for general information on debugging UM code.

However, there are some instances where a special approach may be of use when dealing which the ocean model.

Code development

Row 1 of the ocean will be the most southerly row of the model. Operations over rows typically go from row 1 to row JMT. JMT will therefore refer to the northernmost row of your horizontal domain. This information can be useful if you base ocean code developments on the atmosphere code structure. The atmosphere model works in the opposite direction, from north to south so row 1 will be the most northerly row in the atmosphere domain. Advection of a property from, say, row 5 to row 6 means something different in each model.

This detail may also be of use when developing MPP message passing code. If a PE needs to send data to its northern neighbour, care must be taken to ensure that is what happens. In particular, code and comments in the widely used SWAPBOUNDS routine are incorrect when applied to the ocean model. The terms **southern**

neighbour and **northern neighbour** as used in this routine actually refer to **northern** and **southern** respectively when the routine is applied to the ocean code. (SWAPBOUNDS works correctly but operates in the opposite order from the way the documentation would have you believe).

Many UM control routines were written with the atmosphere in mind. Consequently, documentation and coding styles can sometimes be rather atmosphere model orientated.

Much of the atmosphere model and control code is partitioned into what are known as sections of code. Subroutines in these sections are usually precompiled on the Met Office T3E. This saves the need to compile every routine when a model is compiled. Object files from pre-compiled sections can be linked during the creation of the executable file. The ocean model code, whilst making use of a number of control code sections, is not generally partitioned into sections.

Most ocean specific routines are contained within an *IF DEF,OCEAN control to ensure that they are only included in ocean model compilations. There are additional *IF DEF controls in ocean code which limit what code gets included in a compilation.

For example:

- MPP - Controls code which is only relevant to MPP configurations - widely used throughout all parts of the UM.
- OBIOLOGY - Controls ocean biology model code
- OCARBON - Controls ocean carbon cycle model code
- OCNASSM - Controls ocean data assimilation code
- OISOPYC - Controls ocean isopycnal diffusion scheme code
- OISOPYCGM - Controls ocean Gent & McWilliams isopycnal diffusion scheme code
- RIVERS - Controls River run-off code associated with coupled ocean-atmosphere models.
- SEAICE - Controls ocean ice model code

See section 3.1 for more information on how to modify UM source code. See 3.3 for more on the UM compilation system and controls.

Debugging code

Most of the standard UM procedures for debugging code apply equally well to the Ocean model as to any other part of the UM. See section 4.18 for more information.

There are some further ocean model specific items which may help you to debug ocean model code.

Generally for ocean model code on all platforms:

- Several ocean model subroutines are very large. (Particularly ROWCALC, CLINIC and TRACER). In the past, compilers have sometimes failed to cope with processing such large routines. This has typically occurred when an attempt is made to use certain optimisation options. The code becomes too complex to be analysed and the compilation fails. On Cray systems, a message of at least vague help may be issued. Behaviour will vary from platform to platform. This is not thought to be a current problem on the Met Office Cray T3E.

- The ice model routine ICE_UPDATE_SNOW often traps errors when something is wrong with velocities. This usually points to a blow up of the model velocities or streamfunction at a much earlier stage. (ICE_UPDATE_SNOW merely acts as a trap for these conditions).
- Continuation lines limit. Many lists of arguments passed between subroutines in the model are very long. Some of these are approaching the 99 line limit imposed by Cray Fortran. Addition of even one or two extra lines (particularly in coupled models, where more data is passed around) may lead to this limit being reached. Subroutines particularly at risk are OCN_CTL, ROW_CTL and BLOKCALC.
- Many standard ocean models include large numbers of diagnostic requests. These can consume very large amounts of memory. When running on 1 PE (either in MPP or non-MPP mode), the PE memory limit may well be reached for such jobs. In such cases, either switch to running on a high memory PE (on the Cray T3E) or cut down the number of diagnostics requested.

Specifically for MPP ocean model code:

- If your model is failing after one or more timesteps, try producing a dump file at each timestep. Re-run the job with a different number of PEs and compare the output with the first job. If differences occur, a parallisation problem may well be to blame (eg missing halo swapping). If the results are the same, then something rather more fundamental may be wrong with the code formulation.
- If your model fails early on, try using totalview to examine variable values at the failure point. You could also step through the code with two jobs side by side. E.g. a 1 CPU version and a multi-cpu version. This would allow comparison of values in which you are interested.
- Be careful about insterting barriers in the code, particularly in the subroutine ROWCALC and below. Not all PEs will go through the same code path. With particular reference to those PEs who own the first and last ocean row, the bulk of the code in CLINIC and TRACER may not be executed for these rows. Consequently any attempt at synchronisation may fail unless you are careful about where the barrier call is made.
- Most of the standard generic message passing subroutines have the functionality to deal with the ocean's 1-d decomposition. However, there are instances where a specific routine is needed for the ocean. The routine O_SMARTPASS allows an array which has been zonally decomposed to be gathered into a global copy on all PEs. This acts in similar fashion to the GATHER_FIELD routine. However, there are several important points to note:
    1. it will only work for a 1-d zonal decomposition
    2. it allows 1-dimensional (zonal) and 2-dimensional arrays to be gathered (GATHER_FIELD requires a second dimension and will therefore not work for 1-d arrays).
    3. it provides a complete copy of the array to all PEs (rather than to one specific PE).
    4. wherever possible, GATHER_FIELD should be used in preference to O_SMARTPASS.
- Generally and particularly when including write statements in the code, be sure to specify that you want to keep output from all PEs via umui window:

  subindep_OutputMan
  ```
   Sub-Model Independent
   -> Output management
  ```
  If you include write statements to unit 6 (standard output), be aware that output will go to the appropriate file for the PE concerned. Eg: If writing values of an array for a particular row, the output will only appear in the standard output file for the PE which deals with the row in question.
- Use of print statements causes output from all PEs to go to the PE 0 standard output file. However, the order in which data appears is unpredictable and can be difficult to interpret.

Code Performance

A number of tools are available on Cray platforms to analyse code performance. See the documentation on performance analysis tools for your platform. The easiest method of performance analysis in the UM is to make use of the UM's own timer diagnostics. These may be enabled via the UMUI. See the TIMER section of 4.18 for more information.

Analysis of CPU and wallclock times from each subroutine will allow identification of the most costly subroutines. This will help to identify which areas of code are best targeted in any attempt at improving performance.

A number of points should be born in mind when analysing MPP timing data for the ocean model.

- In many configurations, no computations are performed for the very first and last ocean rows. This means that timing data from the first or last PE can be misleading. Time may be attributed to routines where little or no work is being done. This is because the PE is waiting (in a particular routine) to synchronise with other PEs which are performing work elsewhere. Check output timings from a selection of PEs to understand where the time is consumed.
- Check the timing information about minimum and maximum CPU and wallclock use (this appears under the PE 0 timer output). This can help to indicate which routines have a significant load imbalance.
- Separate timing data only appears for those areas of code which specifically have a call to the UM TIMER subroutine. Many of the subroutines called by (for example) TRACER do not have separate TIMER calls. This results in all the time consumed by these subroutines being attributed to TRACER. More detailed timing of particular areas of code, may be obtained by inclusion of extra TIMER calls of your own. (See existing code for the syntax relevant in calling TIMER).
- Use a run long enough for your timing data to be realistic. Certain areas of code (eg: the Barotropic streamfunction solver) may take longer than average during the first few timesteps (eg: to converge during model spinup). Use of a very short run in such a case could be misleading. It may suggest that a routine is more (or less) significant than is really the case.

  Some parts of the code are only executed at certain times. E.g. dump/STASH file output, ancillary file updating, Visbeck scheme (if included). You should make your run long enough to include such routines or bear in mind that they may have an additional effect on the performance.
- System load can affect overall performance. In Cray T3E tests, the CPU time taken by a subroutine is usually very nearly reproducible from run to run, provided the routine contains no message passing or I/O. Routines which contain a high amount of communication or I/O tend to vary to a small extent in their CPU use. That is, they appear to be rather more dependent on the system load. Any such effects will vary from platform to platform.

4.11 Slab Model

Author: C.D. Hewitt / C.A. Senior, Last Updated: 20 May 98

4.11.1 Introduction

The slab model comprises a mixed-layer ocean model, together with a simple ice model. Details of the physics will not be given here, but will be described in UM Documentation Paper 58: The Slab Ocean Model (not yet available).

The slab model is run coupled to the atmospheric model. At sea points, the atmosphere model requires the sea surface temperature, together with an ice depth and concentration when sea-ice is present. In an atmosphere only run, the data for these surface conditions is provided from ancillary files. When the slab model is included, the SST and ice parameters are computed interactively during the run. The atmosphere model is run for one coupling period, typically 1 day, during which the driving fluxes for the slab model are averaged. Then the slab model is called (typically with a 1 day timestep) to update the SST, ice depth and concentration, which are passed back to the atmosphere model for use over the next coupling period. The surface temperature of the sea-ice is computed by the atmosphere model rather than the slab model, as the atmosphere model timestep is short enough to represent the diurnal cycle of melting and freezing of the ice surface. At open ocean points, the slab model includes the thermodynamics of the mixed-layer part of the ocean but with no representation of ocean currents. At sea-ice points, the model can include both thermodynamics and a simple free drift parameterization of sea-ice dynamics (a more detailed representation of sea-ice dynamics has been coded but is not recommended to be chosen as it still contains errors).

The information and instructions given here refer to vn4.4 of the model.

4.11.2 Heat Convergence Corrections

In order to obtain a realistic representation of SST and sea-ice, a corrective heat flux called the heat convergence must be included to account for the lack of ocean dynamics and errors in the surface fluxes. The data is supplied from an ancillary file and a slab calibration run will usually be required to generate this data. In a calibration run, the slab model is coupled with the atmosphere model, and SST and ice parameters are computed, but after each slab timestep the SSTs are reset to climatological values provided by an ancillary file, and the corrective heat flux required to do this is stored. At sea-ice points the SST under the ice is reset but no correction is applied to the ice depth. The heat convergence ancillary file is made from these corrective heat fluxes. The stash diagnostic is called the anomalous heat convergence (see diagnostics section).

4.11.3 Setting up a slab model run

These instructions are for converting an atmosphere only experiment to a slab experiment, and assume some familiarity with the structure of the UMUI. There are two types of slab experiment; A calibration experiment and a control experiment. A calibration experiment is specifically to derive the anomalous heat convergences described above, and a reasonable length for the run is 5-10 years, of which the first 3 months of data are usually ignored while the model spins up. A control run is the name for any experiment in which the slab model predicts SSTs and sea-ice and encompasses experiments with any range of scenarios of external forcing. The UMUI panels that need to be altered are described below with the recommended settings for a calibration or control run described.

1. Sub-Model sections

   The slab model is one of the sub-models of the UM system. It can only be run in conjuntion with the atmospheric model. These panels control the sub-model switches and coupling betwen the sub-models.

   a) Window: **smcc_Model**
   ```
   Sub-Model Configuration and Coupling
   -> Sub-Model inclusion switches
   ```
   Choose Atmospheric GCM and Slab Ocean ON

   b) Window: **smcc_AS_Coupling**
   ```
   Sub-Model Configuration and Coupling
   -> Atmosphere and Slab-ocean coupling
   ```
   Choose standard coupling Macro with coupling every 1 day

2. Slab model science

   a) Window: **slab_Science_Timestep**
   ```
   Slab Ocean
   -> Science and scientific sections
   --> Timestep
   ```
   Choose Coupling every 1 day with 1 timestep per period and 1 day in period

   b) Window: **slab_Science_Section**
   ```
   Slab Ocean
   -> Science and scientific sections
   --> Slab section 40 configuration choice
   ```
   Choose version 1A. DO NOT choose simple ocean dynamics.

   For *calibration* run choose 'defining this to be a calibration run'.

   For a *control* run DO NOT set this switch

   c) Window: **slab_Science_Physical**
   ```
   Slab Ocean
   -> Science and scientific sections
   --> Physical parameters
   ```
   Set:

Thickness of mixed layer (m):	50
Minimum ice concentration:	0.001
Maximum ice concentration, northern hemisphere:	0.995
Maximum ice concentration, southern hemisphere:	0.980
Minimum grid-box average ice depth (m):	0.01
Minimum local ice depth (m):	0.5
Limit on heat convergence under sea-ice (W/m**2):	-40.0

   This panel sets the parameters in the namelist &SLABLIST, and the recommended values are shown above. The first parameter is the mixed layer depth and a constant value is used over the whole ocean, usually 50m. The ice concentration is simply the fraction of a gridbox covered by sea-ice. A minimum value is needed for this to avoid numerical problems, the recommended value being 0.001. Since unbroken ice cover is very rare, the ice concentration is prevented from reaching 1.0 by setting a maximum value. This upper limit is different for the northern and southern hemispheres as Antarctic sea-ice has been observed to be less compact than that in the the Arctic, and the recommended values are 0.98 and 0.995 respectively. The minimum grid-box-average ice depth specifies the smallest amount of ice that is allowed to exist in any grid box. This is required to avoid problems with very thin layers of ice, and is set to 0.01m. The minimum local ice depth is the thickness of newly formed ice over the icy portion of the grid box. This is used to determine ice concentration of newly formed ice. For example, suppose the minimum amount of ice forms with a grid-box mean of 0.01m. It is then assumed that the actual depth of this ice is 0.5m, and so the ice concentration must be 0.02; i.e. 2% of the grid-box is covered by ice. If the heat convergence under sea-ice is large and negative, it can lead to a run-away build up of ice. To avoid this a limit is set on the value (-40Wm-2) and lower values are reduced to this limit. Conservancy is maintained by redistributing the difference over all open ocean points. In practice, this rarely happens with the present configuration as there is no specific correction to ice depth in the calibration experiment (as has previously been the case) and large values of heat convergence under sea-ice are avoided.

d) Window: **slab_Science_Ice**
```
Slab Ocean
-> Science and scientific sections
--> Slab-Ice model
```
Choose:
coupled model ice thermodynamics INCLUDED
Using Eddy Diffusion Coefficient for ocean-ice heat flux: 3.750e-04
Choose:
Simple ice advection and diffusion Using Diffusion Coefficient for ice (m**2/s): 0.0

This panel sets the options for the ice dynamics part of the sea-ice code. You will also see the options set in the previous panel for the thermodynamic part of the ice code. The coupled model thermodynamics brings the slab model in-line with the sea-ice thermodynamics used in the coupled model. The ocean-ice heat flux eddy diffusion coefficent has been specially chosen in conjunction with a specially constructed SST ancillary file to achieve a reasonable present-day sea-ice climatology. It is recommended that this value is chosen if the ancillary file $UMDIR/vn4.4/ancil/atmos/cl_9673/qrclim.newsst5 is chosen. The diffusion of ice is currently switched OFF. This is done by using a diffusion coefficient of 0.

3. Slab model control code

a) Window: **slab_Control_Section_ClimMean**
```
Slab Ocean
-> Control sections
--> Define climate meaning section choice
```
Choose: Climate meaning not included

4. Slab model ancillary files

b) Window: **slab_InFiles_PAncil_HeatCon**
```
Slab Ocean
-> Ancillary Files
--> Heat convergence
```
Calibration run: Choose Heat Convergence Ancillary file not used

Control Run : Choose Heat Convergence Ancillary file Updated every 5 days and specify the directory and filename. This file will be made in the calibration experiment

c) Window: **slab_InFiles_PAncil_Calib**
```
Slab Ocean
-> Ancillary Files
--> Calibration fields
```
Calibration run:

Choose sea surface temperature file: Updated every 5 days
Choose sea-ice thickness field: Updated every 5 days.
The standard fields for 96x73 are:
$UMDIR/vn4.4/ancil/atmos/cl_9673/qrclim.newsst5
$UMDIR/vn4.4/ancil/slab/cl_9673/qrclim.icedp.old
This data goes into the reference SST and the reference ice depth in the slab model. Note that the ice depth file is NOT the standard file used for climate runs. It has the same ice extents as the

standard file, but different ice depths. For low resolution runs (5 x 7.5), the equivalent ancillary files in the cl_4837 directories should be used.

The standard atmospheric SST and ice updating MUST NOT be updated, although they can be configured. This can be done here by pushing the SST and ICE buttons. If these are updated they will overwrite the slab generated fields. If they are configured they MUST be identical to the ancillary files described above. The SST anomalies should NOT be included.

Control run :

Choose sea surface temperature file: Not used.
Choose sea-ice thickness field: Not used.

If the model dump used for starting the slab model does not contain the prognostic slab temperature then it should be initialised from the surface temperature. If the model is starting from a pre-existing slab model dump which contains the prognostics slab temperature (as is usual) then this should NOT be switched on.

d) Window: **atmos_InFiles_PAncil_Seasurf**

```
Atmosphere
-> Ancillary amd input data files
--> Climatologies and potential climatologies
---> sea surface currents
```

Calibration and *Control* run: Choose sea surface currents file: Updated evey 5 days

The datasets for 96x73 are: $UMDIR/vn4.4/ancil/slab/cl_9673/qrclim.uvcurr

5. Modsets

   At vn4.4 of the model, two mods to the slab code are required. These are changes to the control code only and do not affect the science. These are:

$UMDIR/vn4.5/mods/source/sch0f405	enables the model to run in an mpp-configuration
$UMDIR/vn4.5/mods/source/gie0f405	deletes superceded restart dumps

6. Diagnostics

```
Slab Ocean
-> STASH
--> STASH. Specification of diagnostic requirements
```

Slab model diagnostics are in stash section 40. These can be accessed via ’Load New Diag’ Some of the diagnostics are only available for certain configurations of the slab model. In particular, many are limited to the choice of ’cavitating fluid ice dynamics’ or the choice of ’simple ocean dynamics’ (see above). It is not recommended that either of these options are chosen at the moment, so these diagnostcis will not be available. There are also diagnostics that are only available in calibration runs or only in control runs. These are;

Calibration only;

40 201 ANOMALOUS HEAT CONVERGENCE (SLAB) W/M2

Control only;

40 177 HEAT CONVERGENCE AFTER SLAB W/M2 A
40 202 REDISTRIBUTED HEAT CONVERGENCE SLAB

The anomalous heat convergence is diagnosed during a calibration run and this diagnostic is only available then. It is needed to make the heat convergence ancillary file and it is therefore VITAL that this is switched on during a calibration experiment. The two other heat convergence diagnostics are not available in a calibration run. ’Heat convergence (slab model)’ is the data input to the model from the ancillary file, ’Redistributed heat convergence’ is the heat convergence actually used during the run (which is modified from the input values). It is recommended that all other available diagnostics are switched ON. If restart dumps are output every day then all the diagnostics should have a time profile such as that described below;

```
Time profile name                  TALL
Specify time processing required  'No time processing. Field valid
                                   at output timesteps.'
Specify the output times for the diagnostic
   Specification type              'Regular intervals'
   Time units                      'Timesteps'
   Starting                        1
   Ending                          -1
   Frequency (every)               1
Set ending to -1 for the whole run
```

These are usually sent to the dump store for climate meaning for all mean periods and are all single level fields.

For the anomalous heat convergence diagnostics in a calibration run it is advantageous to additionally save this to a seperate output stream with a time profile like;

```
Time profile name                  TSLBMN
Specify time processing required  'Time mean, specify meaning period
                                   and sampling frequency below.'
Define the meaning period:
   Time units                      'Timesteps'
   Sampling period                 30
Define the sampling frequency to make up the above:
   Time units                      'Timesteps'
   Frequency (every)               1
   Sampling offset                 0
Specify the output times for the diagnostic
   Specification type              'Regular intervals'
   Time units                      'Timesteps'
   Starting                        30
   Ending                          -1
   Frequency (every)               30
Set ending to -1 for the whole run
```

A separate USAGE profile should be set up that sends the diagnostic to an ouput stream which is reinitialised every 360 days. Chose the starting point someway into the run (eg 3 months) to allow the model to spin-up before the heat convergences are saved.

If restart dumps are output less frequently than every day, i.e. every 10 days as is suggested for long climate runs then the following time profiles should be used

```
Time profile name                  TALL
Specify time processing required  'Time mean, specify meaning period
                                   and sampling...'
Define the meaning period:
   Time units                      'Dump periods'
   Sampling period                 1
```

```
Define the sampling frequency to make up the above:
   Time units                      'Days'
   Frequency (every)               1
   Sampling offset                 0
Specify the output times for the diagnostic
   Specification type              'Regular intervals'
   Time units                      'Dump periods
   Starting                        1
   Ending                          -1
   Frequency (every)               1
Set ending to -1 for the whole run

Time profile name                  TSLBMN
Specify time processing required   'Time mean, specify meaning period
                                    and sampling frequency below.'
Define the meaning period:
   Time units                      'Dump periods'
   Sampling period                 3
Define the sampling frequency to make up the above:
   Time units                      'Days'
   Frequency (every)               1
   Sampling offset                 0
Specify the output times for the diagnostic
   Specification type              'Regular intervals'
   Time units                      'Dump periods'
   Starting                        3
   Ending                          -1
   Frequency (every)               3
Set ending to -1 for the whole run
```

4.11.4 Running the slab model

The actual running of a slab model is no different to running an atmosphere only model; just process the job and submit as normal.

4.11.5 Making a heat convergence ancillary file

Once the calibration run is completed, the heat convergence ancillary file can be made from its output. From your model output, you need to construct a pp-file containing 12 fields, each a multi-year monthly mean of the anomalous heat convergence, starting with January, and ending with December. The easiest way to do this is to use the fields saved in the separate output stream for the anomalous heat convergences. However, it is also straightforward if your data is just in the standard monthly mean files. If you have access to the HC workstation system, there are some programs available to manipulate the data into the required form. Instructions are given in ~hadca/slab/heatconv/README. Once the heat convergence file is created, it must be converted to an ancillary file. This can be done using the UM Utility PPTOANC via the method described on the web pages (http://fr0500/~frdr/pptoanc/pptoanc.html). An example namelist is at /u/m20/cdev/mef/t20ch/ancil/namelist/heat_convergence.

4.12 Atmosphere-Ocean Coupled Models

Author: Rick Rawlins, Last Updated: 29 Jan 98

4.12.1 Overview

The present design of the model control levels supports two-way coupling between:

(a) Atmospheric submodel and full oceanic submodel (including seaice)

(b) Atmospheric submodel and simple slab oceanic submodel

(c) An internal UM submodel (eg. atmosphere) and an external submodel, via the OASIS coupling system (see 4.13 OASIS)

where (a) and (b) lie within the standard UM library code, and (c) requires the external model to have an OASIS interface.

Future plans for the development of the UM include an expanded coupling concept based on the idea of multiple submodels (eg. atmosphere, ocean, seaice, wave, atmospheric chemistry, land/ocean biology, etc) with greater flexibility and modularity of coupling between them.

In order to achieve coupling in general, information is passed periodically from one submodel to another. This information may be in the form of instantaneous values (eg. presently predicted SST from the ocean model), or time-averaged over the period used for coupling (eg. the daily average solar energy flux into the ocean surface computed by the atmospheric submodel).

Extra technical details are given in UMDP C2 (see [28]).

4.12.2 Timestepping in Coupled Models

A well-defined information-passing sequence is necessary to carry a coupled model forward in time, since each submodel is dependent on other submodels for forcing or boundary conditions. In the case of an atmosphere-ocean coupled model, the convention in the UM is that the atmospheric submodel is integrated first, atmosphere-ocean fluxes are computed and passed to the ocean/seaice submodel, the ocean/seaice submodel is integrated next, and new ocean/seaice surface variables are passed back to the atmospheric submodel to enable the cycle to be repeated. The period over which the cycle repeats is called the *coupling period*, and this approach is called *asynchronous coupling* because only one component of the model is being integrated at a time. There are alternative ways of coupling in which the submodels might be integrated in parallel, which might be more suited to highly parallel computer hardware. This *parallel coupling* approach requires additional synchronisation calls in the code and may be explored in later stages of the *Submodels* project.

Note: It is common practice to use a 1-day coupling period in the coupled atmosphere-ocean configuration, but this can easily be changed if desired.

4.12.3 Interpolation Between Submodels

In general, the spatial grids of two submodels connected by coupling may not be congruent, although they are congruent for some configurations in common use at the moment. Interpolations or area-averaging will need to be applied to coupling fields if the grids are not congruent. This operation is carried out as part of the internal coupling process in the model, with options as to the method used in some cases.

4.12.4 Special Ancillary Forcing for Coupled Models

In general, coupled models require fewer ancillary fields to be supplied than single-component models (eg. atmosphere-only or ocean-only), since the model predicts many of the fields which otherwise need to be specified as forcing or boundary conditions. However, atmosphere-ocean coupled models typically need to be calibrated against various reference climatologies (in so-called *calibration* runs) before they can be used to run prediction experiments. In the present atmosphere-slab-seaice configuration, these calibration fields comprise sea surface temperatures, ocean currents and sea ice distribution. In the full atmosphere-ocean-seaice GCM, calibration is performed against reference sea surface temperatures and salinities, but the model is now sufficiently good that it is increasingly being run without such an initial calibration procedure. As a result of the calibration process, ancillary fields are derived which are used as constant (seasonally-varying) forcing terms in subsequent runs to encourage the model to give a control climatology close to the reference climatology. These derived fields are referred to as *flux adjustments* or, in the case of the slab submodel heat corrections, *heat convergences*.

4.12.5 Coupled Models on an MPP computer

Atmosphere and ocean submodels run asynchronously, with a parallel version of atmosphere followed by a parallel version of ocean, and each submodel uses the same number of processors. It is important to realise that the data decomposition strategy used in atmosphere and ocean components is fundamentally different. This needs to be recognised if making changes in the coupling areas of code which are concerned with the spatial arrangements of data. Additionally, the number of processors that can be used for the coupled model is limited to the number of ocean rows, due to the row-wise decomposition employed in the ocean submodel. Interpolation calculations at each coupling period are performed on a single processor and include the change in domain composition.

The slab submodel has not yet been converted to run on an MPP platform.

4.12.6 Examples of Atmosphere-Ocean Coupled Models

Atmosphere-Ocean General Circulation Coupled Model

This version of the model, running on a global domain with a seaice submodel embedded in the ocean submodel, is the main configuration used for climate change simulations. The technique of *flux adjustment* is used in some production experiments to suppress the problem of climate drift which can occur in models of this type. An example is the HadCM2 model. Some experiments can also be run with flux adjustments omitted. The HadCM3 model is an example of the latter. [The HadCM2 and HadCM3 models were developed at the Hadley Centre for Climate Prediction and Research within the UK Met Office.]

Atmosphere-to-Ocean Coupling Fields

(i) Penetrative solar flux into ocean

(ii) Non-penetrative solar flux plus other heat fluxes into ocean surface

(iii) Precipitation minus evaporation net fresh water flux into ocean surface

(iv) River runoff net fresh water flux into coastal ocean outflow points

(v) Atmospheric wind stress into ocean surface

(vi) Wind mixing energy flux into ocean near-surface mixed layer

(vii) Net heat flux from atmosphere into seaice

(viii) Net snowfall minus sublimation onto seaice

Note: (vii) and (viii) are only passed if the seaice submodel is included.

Ocean-to-Atmosphere Coupling Fields

(i) Ocean sea surface temperature and surface currents

(ii) Seaice fractional coverage within gridbox and mean seaice thickness

Atmosphere-Slab Coupled Model

This version of the model always uses a global domain, and usually includes a seaice submodel as part of the slab ocean. It is primarily used for short climate sensitivity experiments, and for examination of equilibrium climate change responses in comparison to transient responses from the full model.

Atmosphere-to-Slab Coupling Fields

(i) Net solar flux plus other heat fluxes into slab ocean

(ii) Net heat flux from atmosphere into seaice

(iii) Net snowfall minus sublimation onto seaice

Slab-to-Atmosphere Coupling Fields

(i) Slab ocean temperature

(ii) Seaice fractional coverage within gridbox and mean seaice thickness

4.12.7 Coupling Information Required on UMUI Panels

This falls logically into 3 parts.

1. Which submodels are included in the configuration?
   Window: **smcc_Model**
   ```
   Sub-Model Configurations and Coupling
   -> Sub-Model Inclusion Switches
   ```

2. *a)* What is the coupling period and interpolation method for atmos-ocean GCM?
   Window: **smcc_AO_Coupling**
   ```
   Sub-Model Configurations and Coupling
   -> Atmosphere & Ocean GCM coupling
   ```
   b) What is the coupling period for atmos-slab GCM?
   Window: **smcc_AS_Coupling**
   ```
   Sub-Model Configurations and Coupling
   -> Atmosphere & Slab-ocean coupling
   ```

3. Other detailed information (ocean and slab-related) which determines the flavour of the coupled configuration (eg. flux-adjustments included/excluded, calibration/control run, frequency of updating related ancillary fields, etc.).
   Window: **ocean_Science_SurfForce_Haney**
   ```
   Ocean GCM
   -> Scientific parameters and sections
   --> Surface Forcing
   ---> Haney Forcing
   ```
   Window: **ocean_Science_SurfForce_Flux**
   ```
   Ocean GCM
   -> Scientific parameters and sections
   --> Surface Forcing
   ---> Flux Correction
   ```
   Window: **ocean_InFiles_PAncil_SstSss**
   ```
   Ocean GCM
   -> Input Files
   --> Climatologies and potential climatologies
   ---> Reference SST, SSS, Air-Temp & Ice-Depth
   ```
   Window: **ocean_InFiles_PAncil_CorrHeatSal**
   ```
   Ocean GCM
   -> Input Files
   --> Climatologies and potential climatologies
   ---> Flux Correction fields
   ```
   Window: **slab_Science_Section**
   ```
   Slab Ocean
   -> Science and scientific sections
   --> Slab section 40. Configuration choices
   ```
   Window: **slab_InFiles_PAncil_HeatCon**
   ```
   Slab Ocean
   -> Ancillary files
   --> Heat convergence
   ```
   Window: **slab_InFiles_PAncil_Calib**
   ```
   Slab Ocean
   -> Ancillary files
   --> Calibration fields
   ```

4.13 OASIS Coupling Between Models

Author: J-C Thil (jcthil@meto.gov.uk), Last Updated: 27 April 1998

4.13.1 Overview

A new method for coupling the Unified Model to external modelling components has been introduced at version 4.4. The strategy is to provide a mechanism for independently developed sub-models to interact with the UM with minimal programming development costs. The system is based on a distributed external coupler called OASIS (Ocean Atmosphere Sea Ice Soil) developed at CERFACS by Laurent Terray which has been integrated into the UM infrastructure.

External components need their code to be slightly adapted to specify which physical quantities should be exported or imported to the UM, along with description files for the grid layout and land-sea masks. Once this preliminary work has been accomplished, the OASIS system within the UM allows external and UM sub-model components to execute in parallel as two independent processes.

Each of the executable model communicates with the OASIS executable through Unix named pipes and files until the full coupled model integration completes. In the event of a failure in one of the components, a sibling mechanism between the spawned Unix processes provokes the failure of other components, which would otherwise remain in a waiting state.

Various interpolation methods and masking techniques are made available to users, allowing them to specify external components of different grid layout and resolution from the UM. These are described in the original documentation of OASIS [42].

Each of the atmosphere and ocean components of the UM have been adapted for the use of OASIS, either separately with another external component, or with both UM ocean and atmosphere sub-models being coupled via OASIS. In the former case, the user will need to request - via the UMUI - UM fields required for exchange by the UM component. In the latter case, the fields exchanged will normally comply with the list described in section 4.12 (Coupled Models).

Please refer to UMDP C8 [32] for more technically involved details on the issue of OASIS coupling.

4.13.2 Information Required on the UMUI Panels

1. Selecting the UM section to enable OASIS coupling in the UM :
   Window: **smcc_Coupling_Sections**
   ```
   Sub-Model Configurations and Coupling
   -> Pre-compiled coupling sections
   ```
2. General set up of OASIS
   Window: **smcc_OASIS_Coupling**
   ```
   Sub-Model Configurations and Coupling
   -> OASIS Coupling
   ```
3. Compile window for the OASIS coupler
   Window: **smcc_OASIS_Compile**
   ```
   Sub-Model Configurations and Coupling
   -> OASIS Coupling
   --> Compile button
   ```
4. Definition of the external slave Model from the Master UM model when coupling with OASIS
   Window: **smcc_OASIS_Coupling2**
   ```
   Sub-Model Configurations and Coupling
   -> OASIS Coupling
   --> Slave button
   ```

4.14 Diagnostic Output (STASH)

Author: David Robinson, Last Updated: 18 May 98

4.14.1 Introduction

The output of diagnostics from the UM is set up using the Spatial and Temporal Averaging and Storage Handling (STASH) system. STASH is a very flexible system that allows users to obtain diagnostics on a timestep to timestep basis and to process diagnostics in both time and space. Diagnostics include intercepted fields generated as the model runs and derived quantities. Users can chose from a wide range of diagnostics available or interface new diagnostics into the system.

Further details on the STASH system can be found in UMDP C4 [29].

4.14.2 STASH in the UMUI

In the UMUI, there is a main STASH panel available for each internal model currently available - Atmosphere, Ocean and Slab. The path to this panel is:

```
Atmosphere OR Ocean GCM OR Slab Ocean
-> STASH
--> STASH, Specification of Diagnostic requirements
```

This panel is where the user will select the diagnostics required in a model run. The bottom part of the panel lists the diagnostics required. Each diagnostic will have three profiles attached to them and the profiles appear in the top half of the panel. If there are no diagnostics or profiles then all the boxes in this panel will be empty. It is recommended that the user copies an UMUI experiment/job with some diagnostics and profiles already set up.

In the middle of the panel, there is a HELP button. Clicking here will bring up a menu leading to help pages on STASH, Diagnostics and Profiles. Users should refer to the help page on STASH before doing any STASH processing, especially those entering STASH for the first time. All the windows involved with STASH have HELP buttons which provide detailed information for each window.

It is recommended that the user defines the model configuration in the UMUI before entering STASH. This is because the availability of some diagnostics depend on answers to questions in other parts of the UMUI.

4.14.3 Setting up the Diagnostic List

The first task for most users will be to sort out what diagnostics are required in the list. Detailed information on the diagnostic list and how to manage it through action buttons can be found through the help page on diagnostics. Basically, the user chooses to add or delete diagnostics from one of the available sections. A useful feature is the option to disable (rather than delete) various diagnostics so that they can be used later if required.

4.14.4 Profiles

After setting up the diagnostic list, the user needs to consider the three profiles - Time, Domain and Usage - to be attached to each diagnostic. The profiles are identified by their names which normally begin with ‘T’, ‘D’ and ‘U’ to help recognise the type of profile. This naming convention is not compulsory.

Detailed information on the profiles, how to manage them through action buttons and how to attach them to the diagnostics in the diagnostic list can be found through the help page on profiles.

Time Profiles

They determine when a diagnostic will be output and the period over which time processing on any diagnostics is done (eg, accumulations or means).

- For diagnostics that involve time processing (Time Accumulations, Time Means, Time Series or Max/Min value in a period), two times must be entered. The *first* is the sampling frequency and the *second* is the period for the time processing. As an example, if an accumulation period of 6 hours with a sampling frequency of one timestep, the output will be the sum of the value at every timestep over a period of six hours. If an accumulation is to take place throughout the run then the accumulation period should be set to -1.
- Diagnostics can be output at any time ; Regular or irregular output times can be selected. For *regular* output a start time, end time and a frequency are required. If output is required at regular intervals for the entire length of the run the end time should be set to -1. For *irregular* output, the times are entered in the window.
- For time-processed diagnostics, ensure that the frequency of output times is correctly specified and in agreement with the sampling and period of the time processing. For regular output, there is a calculation done on the start time to ensure that the output times coincide with the end of any time-processing period. Take a six hour accumulation period for example, the output times can be at the end of any accumulation period, ie. after 6,12,18,24... hours. So it is possible to output a six hour accumulation every 24 hours; the output will contain accumulated data for the periods 18-24 hours, 42-48 hours and so on. The frequency of output times must not be shorter, but not need be the same as, the time processing period.
- For time series diagnostics, time and domain profiles for time series must be set up. A time series holds a number of records in one PP field. Each record contains the average for a sub-region of the model domain at periodic intervals for a single model variable. All the records have the same time period and contain values from the same time-steps. They only differ in the sub-region from which they are computed.
- Care needs to be taken when requesting diagnostics from sections that are not called every timestep, such as SW radiation.

Domain Profiles

They determine the geographical region and level(s) for which the diagnostic will be output. Four windows are involved in setting up a domain profile and the UMUI will start with window **atmos_STASH_Domain1**. At the bottom of each window, there are three out of the four - LEVS, HORIZ, PSEUDO, TSERIES - buttons which will take the user to one of the other three windows.

Through window **atmos_STASH_Domain1** (LEVS button), the level type is selected. Depending on the choice, the user will be prompted to enter more information, for example, choosing model levels will prompt the user to enter whether a range or a selection of levels is required.

Through window **atmos_STASH_Domain3** (HORIZ button), the following are selected:

- The geographical area. There are a number of standard areas to choose from. The user can also select an area by specifying lat-long or gridpoint rectangle. Where necessary, the area is rounded up to match full model grid boxes and diagnostics could be output for a slightly greater area than that specified - especially for Limited Area models.

- Grid points to be included (ie, all, land or sea). Points not included are set to the missing data indicator.
- Any meaning of the data (ie, in zonal, vertical, meridional or horizontal). Meaned data saves time and disk space.
- Any weighting of the data (ie, horizontally or by volume or by mass)
- For time series diagnostics, domain (and time) profiles for time series must be set up.

Through window **atmos_STASH_Domain2** (PSEUDO button), a pseudo level type is selected. This allows a fourth dimension (such as LW or SW bands) which is required with a few fields. This is not often used and is normally set to 'No pseudo level dimension'.

Through window **atmos_STASH_Domain4** (TSERIES button), the domain profile is defined for a Time Series diagnostic. A list of model grid-points and levels allows the user to specify the sub-region(s) that are meaned in order to form an average for each sub-region. Note that a domain profile can contain a number of sub-regions. If desired, just one model grid-point can be selected, e.g. to compare model data with data from individual stations.

Usage Profiles

They specify which output unit (often PP-file) the diagnostic will be stored in. Diagnostics can be sent to more than one output unit by repeating the diagnostic in the list and attaching different usage profiles.

The unit numbers for the PP files are in the range 60-69. Output written to units 60,61,…,69 are stored in files with the extension .pp0,.pp1,…,.pp9 where PP files are not reinitialised. As an example, output sent to unit 64 from experiment ABCD, job Z will go to the file abcdz.pp4. Reinitialised PP files have the extension .pa,.pb,…,.pj and include a model date calculated automatically.

Usage profiles are also used to direct output to files in connection with climate meaning. For further details, see section 4.15.

Not all diagnostic fields go into a PP file. A number of fields are stored internally to be passed between different sections of the UM during a model run. Through the Usage profile those fields are given a TAG number to enable the UM to search for these fields by their TAG number when required. The help page on Usage profiles provides more information on attaching Tag Numbers to diagnostic fields.

4.14.5 The STASHmaster file (SMF)

This file contains information on all diagnostics available to the UM. For every prognostic or diagnostic, there is a record which is described in full in UMDP C4. Typical information stored is what grid and levels (model or pressure) the data is stored on and how the data should be packed. It is the information in this file that the stash panels in the UMUI use to verify whether any diagnostics and its profiles in the diagnostic list have been set up correctly.

Any prognostic or diagnostic used in the UM must exist in this file otherwise it must be provided through an USER STASHmaster file.

The USER STASHmaster file (USMF)

This file enables an user to add new prognostics, diagnostics or ancillary fields if they do not exist in the SMF. It can also be used to correct or test changes to a record in the SMF. USMFs are specified in the window

atmos_STASH_UserDiags

```
Atmosphere
-> STASH
--> User-STASHmaster files. Diags, Progs & Ancills.
```

(similar windows exist in the `Ocean GCM` and `Slab Ocean` sections)

- Several USMFs can be attached. Typically, each file should relate to individual modsets.
- Any USMF must be set up and attached before entering STASH.
- The USMF must exist on the same platform as the UMUI client.
- UMDP C4 [29] (page 13) provides all the information that can be set in a record for each field.
- A template USMF to start from can be found in the help panel for this window.

Adding any new fields usually requires extensive changes to the UM code. A system has been set up to allow the user to add new fields and use them as ancillary fields without the need to modify any code for the UM to recognise and get these fields into the model (Code changes are still required to integrate the data in the model). These fields are also known as 'User Ancillaries'. This involves setting up a USMF using Stashcodes (under section 0) which have been reserved:

- 301-320 : For atmosphere single level ancillaries
- 321-324 : For atmosphere multi-level ancillaries
- 331-340 : For ocean single level ancillaries
- 351-354 : For ocean multi-level ancillaries

Full details on 'User Ancillaries' can be found in section 4.9.7 and the HELP panels.

4.14.6 Initialisation of User Prognostics

After the user has provided extra records through a USMF, it is necessary to initialise any *prognostic* fields for a run. This is done in the window **atmos_STASH_UserProgs**

```
Atmosphere
-> STASH
--> Initialisation of User Prognostics
```

(similar windows exist in the `Ocean GCM` and `Slab Ocean` sections)

In this window, the user has a wide choice on how to initialise their prognostic fields. See the HELP panel.

4.14.7 STASH Macros

For the atmosphere model only, there are a number of STASH macros set up. A STASH macro is an internal list of STASH diagnostic requests with profiles attached. A list of macros are available for selection in the UMUI section:

```
Atmosphere
-> STASH
--> STASH macros
```

There are other STASH macros associated with coupling or including the assimilation. These macros are selected for inclusion in different UMUI locations.

If a macro is included in your job, the macro asks STASH to generate the diagnostics either in a PP file or internally during a model run. The help page for each macro will inform the user where the diagnostics go. If users need to add extra diagnostics to those in the macro (this should be rare), this can be done through the STASH processing section and care should be taken with the Usage profile to ensure the diagnostic goes to the same destination.

The full list of diagnostic fields set by the macro can be browsed in the STASHC file after the job has been processed.

4.15 Climate Meaning

Author: K. Rogers, Last Updated: 1 May 98

4.15.1 Introduction

Climate runs are very long runs of the Unified Model that can automatically resubmit themselves, and are divided into conveniently-sized chunks (or resubmit increments) such as one or two years. They have a built-in mechanism that makes them restartable from a safe restart point linked to the meaning periods. This means that if there is a system crash the model can be restarted from the last safe restart point. The run can also be continued at a later date beyond the original run target-end date.

These runs are different from forecast runs in that the initial conditions as defined in the start dump are not that important compared to the ancillary files that provide the boundary conditions and the forcing parameters. Also in forecast runs the forecast must be produced in the minimum time and it does not matter that results are not reproducible. With climate runs it is important that runs bit compare so that parts of a long run can be repeated with extra diagnostics and to ensure an experiment is consistent with a previous one, since last bit differences can make a significant difference to the climate produced by a run. During a climate run, certain properties have to be conserved, such as the mass of the atmosphere, but strict conservation is less important for the short timescales of a forecast run.

The following sections describe the climate meaning system and how to setup diagnostics that use this system.

For more information on the Unified Climate Model see Unified Model documentation papers:

- UMDP 1: Introduction to the Unified Forecast/Climate Model [1]
- UMDP C0: The Top Level Control System [26]

4.15.2 Climate Meaning System

The climate meaning system allows the user to specify up to four ’meaning periods’ and the model automatically produces means of the selected variables over each of these periods. For example, a climate modeller could specify mean fields over 30 days, 1 season (3 months), 1 year and 10 years. The last safe restart point for the model will be at the start of the last period 1 mean eg. the start of the month in the above case.

In UM versions before version 4.4, the system assumed that model years were 360 days long for simplicity. At version 4.4 the system was extended to allow for a real 365/366-day year too. This version of the meaning system works out when there are leap years and uses the actual number of days in the month (28, 29, 30 or 31), 3-month season (89, 90, 91 or 92) or year (365 or 366) being processed. It assumes that period 1, 2 and 3 means are always monthly, seasonal and annual, and currently there is no option to produce period 4 means. The dump frequency must be daily to use the 365 day version of climate meaning. If selected, dump archiving is done on real-month boundaries and there is also the option to reinitialise PP files at the same times.

The basic unit of time of the UM is the ’dump frequency’, typically once a day. This is how often fields are written out from D1, the main data array that holds primary (ie. prognostic and ancillary), diagnostic and secondary fields, to various kinds of dump files. Fields can be processed in various ways within each day (see section 4.15.3 on diagnostics below), but all output files for periods longer than the dump frequency are means of these daily dumps.

Instantaneous dumps (also known as restart dumps) have filenames of the form $RUNID.da…, eg. cbdjia.daj34e0. When things go wrong, it is possible to continue the run from one of these instead of making a new run. It is conventional to choose to produce a restart dump every day, delete them when a new one is produced, and archive one at every period 1 mean, usually the first one of every month, using the UMUI to set this up.

Fields from D1 are also added to partial sum mean dumps, which have names of the form $RUNID.apstmp1 for atmosphere fields and $RUNID.opstmp1 for ocean fields. These files are usually deleted automatically at the end of a run, since they reside in a temporary directory. However if the run stops part way through a meaning period the files are copied to a permanent directory. At the end of every period 1 time length, the completed period 1 partial sum mean dump is copied to the period 2 partial sum mean dump, and so on for the other meaning periods. These further partial sum mean dump files have a different naming convention from the period 1 partial sum dumps, viz. $RUNIDa_s2a and $RUNIDo_s2b etc. where the letter 'a' or 'o' shows whether it is an atmosphere or ocean file, the number shows which period the mean is for and the last indicating letter shows whether the file is for writing to or reading from. These files are not deleted routinely at the end of a run and are held in permanent disk space.

The last type of dump file that may be produced is a mean dump. The filename convention for these is $RUNID.dm... where the letter m indicates that the dump is meaned over a month, t for 10 days, s for a season, y for a year etc. If meaning is done over less conventional timescales the filename convention is $RUNID.d1... where the number 1 indicates that the mean dump is for a period 1 mean; similarly for other mean periods. These mean dump files are not normally archived.

For more information on the meaning system see Unified Model documentation paper:

- UMDP C5: Control of Means Calculations [30]

4.15.3 Setting Up Mean Diagnostics

There are two ways of producing mean diagnostics in the Unified Model system. Most people use the first method (see section on General Meaning System below), from which a wide variety of diagnostics can be obtained. The second method (see section on Selected Means of Instantaneous Values below) uses diagnostics that are calculated anyway if the meaning system is switched on and so involves no extra overhead, but only provides means of instantaneous prognostic variables at the dump time eg. midnight.

Anyone setting up diagnostics for a climate run is strongly advised to copy a standard climate experiment containing diagnostics, so that suitable time and usage profiles together with most commonly used diagnostics are already set up. The notes below give an overview only and are not adequate for setting up diagnostics from scratch.

For more information on the diagnostic system see Unified Model documentation paper:

- UMDP C4: Storage Handling and Diagnostic System (STASH) [29]

General Meaning System

Any diagnostic can have a number of processes applied to it at a frequency of one timestep up to the dump frequency (usually once a day). Then the value held for the day can further be meaned over the different meaning periods. The processes that can be applied include meaning, extracting an instantaneous value, accumulating and calculating maximum/minimum values, all every nth timestep. The frequency is often every timestep but it may be more appropriate for it to be say every 3 timesteps if the model only does relevant calculations every 3 timesteps (as with certain radiation diagnostics for example).

Extra space has to be allocated within the D1 array for these diagnostics and they involve extra processing.

To set up the diagnostics:

1. Open the main stash panel in the UMUI:
   `Atmosphere`
   `-> STASH`
   `--> STASH, Specification of Diagnostic requirements`
   For ocean and slab diagnostics go to the equivalent panel within the `Ocean` or `Slab` section.

2. To set up a time profile press the 'Copy Profile' button to copy an existing time profile and then 'Edit Profile' to edit it. Specify the 'time mean' option and how often you want to take a sample of the diagnostic for meaning (the 'frequency'). The 'output times' specifies the points in the run over which the diagnostic should be produced as a range or a list. Alternatively you can select to accumulate, take an instantaneous value or max/min value over the dump period from the same window.

3. If you want to mean over a meaning period in addition to meaning over the dump frequency (eg. a day) you need to set up the usage profile as follows. Select 'Dump store with climate mean TAG' and then specify which of the 4 meaning periods the diagnostics should be meaned over. Then ensure that all the diagnostics you want meaned use this usage profile in the main stash panel.

4. Select the diagnostics you want to use from each section: Press the 'Load New Diag' button in the main stash window. Double click on a section number to look at all the available diagnostics for that section and double click on individual diagnostics to select them. All selected diagnostics are shown in the list at the bottom of the main stash panel. Then double click on the time/domain/usage profile for each diagnostic to pick up the highlighted profile of each type from the top part of the panel, changing the highlighted profiles as appropriate.

Selected Means of Instantaneous Values

These diagnostics are only used for period means of instantaneous prognostic values at each dumping period (eg. each day) or diagnostics directly calculable from these, so there is a limited list of diagnostics to select from, eg. 26 for atmosphere. The values used in these means are the field values at midnight (since that is when the dump file gets written out each model day). The contents of the dump file get added to the partial dump file each day until the period 1 mean is reached. Then the contents of the file are meaned.

To set up the diagnostics:

Go into the main stash panel and press 'Load New Diag'. Double click on sections 21-24 or 41-44 as appropriate (see below) and then select the diagnostics you want by double-clicking.

Use the following section numbers:

	Model Type		**Diagnostics**
	Atmosphere	**Ocean**	**Called**
Period 1 means	21	41	CM1:...
Period 2 means	22	42	CM2:...
Period 3 means	23	43	CM3:...
Period 4 means	24	44	CM4:...

These sections are only called at the relevant meaning periods.

4.15.4 Output to Reinitialised PP Files

If diagnostics are not to be meaned using sections 21-24 and 41-44 in the climate meaning system and output to mean PP files, they can be output to reinitialised 'daily' PP files (or the same diagnostics can be output in both ways). For example the following types of diagnostic are frequently requested:

1. Instantaneous eg. every 12 hours. Requires no memory overhead.
2. Means over any specified time length x. Stored in D1 so requires increase in memory.
3. Maximum or minimum values over a time length x. Requires space in D1.
4. Accumulation over a time length x. Requires space in D1.

If the same diagnostics are requested as mean diagnostics for the period 1 mean and the time length x is the dump frequency, then no extra memory is required to produce the above diagnostics, in addition to that required to hold the climate mean values.

It is important to check how often these files are reinitialised in window **subindep_PostProc_PPInit**

```
Sub-Model Independent
-> Post Processing
--> Initialization and processing of mean & standard PP files
```

Typically they are reinitialised every period 1 mean, but may need to be reinitialised more frequently if the files are large or there is a shortage of disk space. Daily PP files have names of the form:

$RUNID[ao].p[a-j]date

where [ao] refers to atmosphere or ocean and the [a-j] are the different output streams corresponding to unit numbers 60-69. eg. cabcda.pbj1feb is an atmosphere PP file outputting on unit 61 for the month after 1st Feb 1991 (using the absolute time standard naming convention).

4.16 Automatic Post-Processing for Climate Runs

Author: Linda Wiles, Last Updated: 26 Jan 98

4.16.1 Introduction

Climate mode integrations carried out on powerful computers can generate output at rates which make it necessary for data to be archived and housekeeping tasks carried out as the model is running. Failure to do this can lead to disk space shortages.

The model initiates output post processing by sending output processing requests direct to an independent 'slave' process, which acts as a server for the model. A pipe connects the model to the server, and requests are read by the server down the pipe in the form:

```
%%% filename ARCHIVE/DELETE file_type
```

where '%%%' is a pattern the server can recognise as indicating that the following information is a request to either archive or delete the data. The four file types are dump, pp, mean pp and boundary data.

4.16.2 Archiving system at vn4.4

These scripts are mainly used at the Met. Office, but potentially could be used on other sites by inserting site-dependent copy commands and variables in the parts where $UMMACHINE is not set to "METOCRAY". This would enable superceded dumps or PP files to be deleted and data files to be copied (togther with lists of files to be archived) to another directory or machine while the run is continuing. A separate archive server would need to be developed to move the data to more permanent storage.

Main Functions of the System

- Copy model pp files, restart and dump files to another Unix system or archive system (or to the front end system as at the Met. Office)
- Disk management of the current disk files generated by the model i.e. delete files on current system as run proceeds
- Release users own post-processing scripts

At vn4.4 the archive system generates 'trigger files' and the archiving jobs are built on the secondary system using data contained in the trigger files.

- Upon successful data file transfer, a 'trigger file' is updated with details such as the secondary system dataset name, the review date, tape type etc.
- The trigger file is transferred to the secondary system upon reaching a maximum number of file entries depending on the total size of the files, or on job completion. This number can be over-ridden in the UMUI.
- An Archive Server Function is active on the secondary system and deals with the generation and submission of archive tasks. Such tasks examine the trigger file contents, then carry out archiving procedures. At the Met. Office, the archive server runs on the IBM MVS system.

The system performs the following additional tasks

- Execute user supplied output processing scripts when required (e.g. for the operational suite at the Met. Office).
- Update model output processing history variables as requests are dealt with and recreate output processing history file on completion of each request.
- If the server dies or a transfer error occurs, a message is sent to the model requesting it to stop. This prevents files building up on the computer running the model. In this case, requests from the pipe are sent to a 'runid.failure' file in the users experiment directory. This file is read when the model is restarted, tidying up files automatically, if starting from a continuation run.
- Optionally deletes superceded files.

4.16.3 User instructions for Automatic Post-processing of Climate Runs

The trigger files detailed below are text files created by the post processing Unix scripts containing information required by the secondary system in order to archive data files that have been transferred from the primary system.

Setting up an experiment at version 4.4

The UMUI post-processing screen allows archiving to be switched on or off. This sets a variable in the SCRIPT file. There is also a choice of archive system. The climate runs use the CRACER system; others use UABRF. There is a third option for non Met. Office users. A stream option must be specified, which indicates the files which may be stored together. Five stream options are available which allow different storage combinations. Medium and short-term runs usually archive all streams together, and longer jobs use option 5, but any one of the following options may be chosen:

1. All streams separately.
2. All streams together.
3. Dump and boundary files together, others separately.
4. PP and mean PP files together, others separately.
5. Dump and boundary files together, PP and mean PP files together.

The data archived into CRACER is managed by a review system. Data is not expired automatically. Users specify the review period in the UMUI. The minimum is 60 days to allow for processing and warning periods.

In addition, the user needs to specify the storage class and tape type. There is also the option to make duplex backups of each file type.

Using the information in the completed post-processing screen, the process job will set variables required to convert such requests into information required by the archive system. Such variables are to be found in the experiment PPCNTL file.

Trigger file names and format

Example Trigger file name:

```
     CRACERU.AAWKQX00.V404.D1994335.T102759
     ^       ^    ^    ^         ^    ^
  archive runid stream version Date Time
  prefix        type
```

- CRACERU - Unprocessed trigger file
- RUNID
- Possible values for the stream type are:

  ```
  D00 - Dump
  M00 - Mean pp
  N00 - Mean pp and instantaneous (daily) pp
  P00 - Instantaneous (daily) pp
  B00 - Boundary
  C00 - Dump and boundary
  X00 - All stream types together
  ```

- Version of the UM
- Date year/julian day in the format YYYY/DDD
- Time hrs/mins/secs

Example of trigger file contents:

```
CRACD.DS.CCLBNA.PTE7BB0 FOREVER 1997/249 SNNCN re01z hadds@hc0100
CRACD.DS.CCLBNA.PTE7BL0 FOREVER 1997/249 SNNCN re01z hadds@hc0100
CRACD.DS.CCLBNA.PTE7C10 FOREVER 1997/249 SNNCN re01z hadds@hc0100
CRACD.DS.CCLBNA.PME7NOV FOREVER 1997/249 SNNCN re01z hadds@hc0100
CRACD.DS.CCLBNA.PAE7NOV FOREVER 1997/249 SNNCN re01z hadds@hc0100
```

All trigger files have a blank first line.

The Trigger File contains the following information for each file. The last three are Met. Office specific items.

- The file name for the secondary system.
- The date of expiry set to FOREVER for the review system.
- The review date, in YYYY/julian day format.
- Archiving information with alternatives

  ```
  S/N  Scratch tape  / Not scratched
  M/N  Mail user with archiving errors / no mail
  D/N  Duplex file / No duplex
  C/U  CRACER/UABRF  Archive type
  R/N  Redwood/Normal tape types
  ```

- The Accounting Code.
- The Users mailid.

4.16.4 Action to be taken on archiving failures

Users wishing to use the automatic archive tidy-up facility, after failed runs and system crashes, should have AUTOMATIC_PP set to true in the SCRIPT file. This is the default in the UMUI.

The 'autopp_tidyup' script has been written to tidy up archiving after a system or archive server failure for the previous run. The tasks that it performs depend on whether the previous run has crashed or failed.

If the model has stopped cleanly

When a run fails, because of a model or archiving error, the job closes down cleanly, leaving all unprocessed files listed in the runid.failure file. Files which have been copied to the secondary system will have entries in a trigger file.

Users should check that the runid.failure file contains all the files to be archived before re-starting as a continuation run.

If the model has crashed

A model stopping suddenly leaves behind a runid.thist file in the *$DATAM* directory. If this is present, it is possible for the system to do most of the tidying up necessary. Users should check that a runid.thist file exists. If this is missing it may not be possible to restart the run. The user will need to create a runid.failure file to archive any data files remaining on the primary system and restart the job as a continuation run. No processing is done in a new run; any runid.failure files are deleted.

Construct a $RUNID.failure file under *$DATAM*, using the format in the example below, starting each line in the first column. Each archive request is followed by a delete request that removes the now transferred file from *$DATAM*. Daily dumps that are not archived should just have a delete request if they are no longer needed, BUT

- Do not delete the last safe re-start dump.
- Do not include a delete request for the latest dump on-line, even though this may need to be archived. A delete request for these dumps will be sent by the model when the next dump of this type is created.

Example: file *aawkq.failure* Note, there should be no blank spaces at the beginning of the lines

```
** WAKEUP **
%%% aawkqa.daj16d0 ARCHIVE DUMP
%%% aawkqa.daj16d0 DELETE
%%% aawkqa.p1j16d0 ARCHIVE PPNOCHART
%%% aawkqa.p1j16d0 DELETE
```

Tasks performed by autopp_tidyup

- Automatic check for inconsistancies between files and trigger files. In a normal model run, and entry is made in the runid.requests file as soon as the request is read from the model. The last entry in the runid.requests file is normally the file which was being processed when the system crashed. This entry is checked and then if the file is still on the primary system, an entry is made to the runid.failure file. If the file has been copied to the secondary system, then an entry is made to the trigger file.

- Automatic trigger file checks. The users experiment directory *$DATAM*, is searched for trigger files for the runid. If the current job is a new run, the trigger files are deleted, otherwise the trigger files are copied to the secondary system. All empy trigger files are deleted.
- Input contents of runid.failure file (archive requests from failed run) into qsserver in order to process unarchived data files.
- The automatic tidyup system creates a trigger file for the last file copied across to the secondary system if it is not already included.

Checks before re-submitting run

- If any old trigger files from a previous run with the same $RUNID, rather than the one that has just failed, are residing in your *$DATAM* directory, then they will be copied to the secondary system; they must be DELETED.
- Finally, do not copy backup versions of Trigger Files in your *$DATAM* directory, as the scripts perform a 'grep' command to find all files containing the pattern 'CRACER'. If these files also have the correct jobid letter for the current run the files are copied to the secondary system and deleted.

4.17 File Utilities

Author: D.M. Goddard, Last Updated: 28 Jan 98

The utilities are a collection of standalone executables. Most are located both on the Cray T3E and on workstations. They require small amounts of memory and can be used interactively.

The majority are run by scripts located in directory $UMDIR/vnn.m/utils where n.m is the UM version number, eg. '4.4'. The environment variable UMDIR is automatically initialised at logon time. *bigend* is only found in $UMDIR/bin on the workstations. The syntax for all the utilities is given in UMDP F5 (see [35]) except for bcreconf, makebc and pptoanc (see table below for documentation).

Unified Model data files are binary files in a format described in UMDP F3 (see [34]). They include model dumps, fieldsfiles (.pp files output by the UM), ancillary files, boundary datasets and observation files.

Most of these utilities will manipulate files created by earlier versions of the model. Any exceptions are stated explicitly in UMDP F5.

Table 4.2 lists the utilities which are available.

Utility	**Platform**	**Description**
cumf	Cray T3E/ws	Compares two UM files. Useful in testing for bit comparison or highlighting differences between dumps
pumf	Cray T3E/ws	Prints out contents of a UM file. Useful for checking headers.
ieee	Cray T3E	(1) Converts Cray T3E UM files to 32-bit IEEE format for transfer to workstations. (2) Converts Cray C90 UM files to 64-bit IEEE format after transfer from Cray C90. (3) Converts Cray T3E UM files to Cray format for use on Cray C90
convpp	Cray T3E/ws	Converts fieldsfile into sequential PP-file for PP processing. Allows users to display dump data on workstations using PP based software.
mergeum	Cray T3E/ws	Appends a UM file onto another at a place specified by the user Useful for combining two boundary data for use in long runs of the model.
bigend	ws	Reverses the order of bytes in each word of a UM file. Needed for DEC Alpha workstations.
fieldop	Cray T3E/ws	Performs simple mathematical functions on UM dumps and fieldsfiles. e.g. subtract, add, multiply etc.
fieldmod	Cray T3E	Thin/scale/convert/select/reject fields in a UM fieldsfile
bcreconf	Cray T3E/ws	Reconfigure a boundary dataset including vertical interpolation if required. For details, see (http://fr0500/˜frdr/bcreconf/bcreconf.html)
makebc	Cray T3E/ws	Create a boundary dataset from model analyses or dumps. For details, see (http://fr0500/˜frdr/makebc/makebc.html)
pptoanc	Cray T3E/ws	Create a UM Ancillary File or Dump from PP files. For details, see (http://fr0500/˜frdr/pptoanc/pptoanc.html)

Table 4.2: File Utilities

4.18 Troubleshooting

Author: Rick Rawlins, Last Updated: 26 Jun 98

4.18.1 So you have problems with running the model?

The UM is a large and complex system with many opportunities for mistakes to occur and errors can become apparent at a number of stages. The logical sequence of a successful run of the model starts with the experiment being defined through accessing the user interface. Once experiment files have been created, a job is launched on the target machine and UM program scripts are initiated. A standard pattern for a job is first to invoke the reconfiguration program, then compile the model to generate a new executable file, and finally run the model, possibly with automatic post-processing. For any particular run, each execution stage is optional. Of course, these stages may all complete successfully but the model integration may still contain serious errors, as evident by an inspection of model output fields.

For Met. Office users, errors within the UM system for successive versions are entered in a Technical Problems Reporting system available from the UM Documentation page via Technical and Meteorological Reported Problems.

A number of topics are covered in the following sections which may give clues to the source of the problem. Where comments predominantly refer to findings with the Cray T3E as target machine to run the model, then this is indicated by the section title. Considering problems in the different stages separately:

Problems during set-up in the UMUI

There is a significant level of verification of users' entries within the UMUI. Each variable within a window has explicit ranges that are checked against, and the validity of the user's responses is checked each time a window is exited. A facility is available to pass through the entire set of variables for the model, providing cross-checking of consistency between the entries of different sections. In addition there are similar tools to inspect the list of STASH output diagnostic requests and profiles for consistency. Most inappropriate settings of UMUI entries will be captured by this method, and Help panels will generally provide guidance. The last line of defence comprises any error messages generated when the job is processed to generate its library of files for the target machine. Although the UMUI can be relied on to create the correct file content in nearly all cases, there may be occasions when a rare combination of options is processed, that wrong values are generated - which should be borne in mind for error investigations.

The most common error for users, having copied a working experiment is:

- Picking up environment variables defined personally from the previous job, eg USERR=$HOME/myancil may be valid for the previous user but not for you. Environment variables are set in window **subindep_FileDir**
  ```
  Sub-Model Independent
  -> File and Directory Naming
  ```

Problems during model initialisation: reconfiguration and compilation

A script to reconfigure the initial model dump is executed first and will be the first contact with real data, failing if initial dump or ancillary files are absent. This will be revealed by generic error messages from the I/O routines "FATAL ERROR WHEN READING/WRITING MODEL DUMP". Note that this message could refer to any dump format file, which includes ancillary and boundary files. Note also that the I/O routines will not necessarily point to an error when file opening is first attempted: a "**WARNING: FILE NOT FOUND" message is produced if the file does not exist. There will often be a number of such messages in a standard valid reconfiguration, since all potential ancillary files are opened and the logic to decide which ancillary files are needed is only introduced at a later stage, before explicitly reading the files.

Model scripts initiate an nupdate step that introduces user changes in source code from the basic release (see section 3.1: Modifying source code). Conflict can occur between competing changes - these are signalled by explicit nupdate messages. The model can fail to compile - again explicit messages point to the offending subroutine in the job output.

Failures during model execution

Can arise (most likely first) from:

1. User errors in model set-up.
   - Incorrect filenames specified;
   - Logic errors in user updates;
   - Inappropriate parameter values chosen.
2. UM system errors in control.
3. Science formulation errors.
4. Input data errors.
5. Computer operating system/hardware problems.

Methods are described for investigating failures during model execution (see section 4.18.3) and discovering sources of error.

Errors in results only apparent after successful completion of the job

This is the trickiest problem! Cause(s) could lie in one of many areas and may require much detective work to resolve. Later sections gives hints on some tools that may help attacking the problem, using run-time diagnostics (see section 4.18.4). Inspection and comparison of model dumps, boundary and ancillary files, and output pp files, can be achieved by the use of UM utilities such as pumf and cumf, described in section 4.17 (File Utilities).

4.18.2 Inspecting output listings from a model run

A standard model compilation and run can easily produce 10k lines of printout, and it is useful to have some idea of the overall structure of the information and know what to search for.

A job output listing will consist of: i) Unix messages from scripts; ii) output from Fortran and C routines - labelled with a banner “OUTPUT”; iii) Unix housekeeping and job accounting information. [On MPP machines, note that an OUTPUT file is generated for each processor and held temporarily on separate files, optionally deleted at end of job. Only the file corresponding to PE0 is included in the job output listing, with banner “PE0 OUTPUT”.]

Output will appear in the following order in job output files. The variables starting “%” can be used for searching but are also in the listing of SCRIPT if requested.

- Script output:
  - updscripts: nupdate for script modifications
  - qsprelim: reconfiguration output
  - qsmain:
    - compile system nupdate extraction

        * compile system linking to prebuilt code
        * make (compile/link) control/modified decks
        * output from executable including tracebacks (possibly from all PEs)
        * location of corefile if applicable
    - qsfinal: tidy up history files from run

- Listing of all job library files if requested in UMUI
- %UPDATES : script nupdate output
- %RECONEW : reconfiguration pre-processing and compile output
- %recona and/or %recona: atmos/ocean reconfiguration output
- %UPDATE : model nupdate output
- %QSMNCOMPILE: pre-processing required for compilation
- %MAKE: make command (compilation/link step) output
- %MODEL : output from all PEs - mainly from C I/O
- HISTORY FILE PRINTOUT : printout of all history file records
- %PE0 : all output from PE0 (except a small amount in %MODEL)
    - set-up information from STASH processing routines
    - "LIST OF USER_DEFINED DIAGNOSTICS" : STASH output requests
    - "READING UNIFIED MODEL DUMP" : details of the input starting dump
    - "ATMOS TIMESTEP" : initialisation completed, atmosphere time-stepping started
    - "FLOW TRACE SUMMARY": timing information
- %AUTOPP_TIDYUP : archiving server tidy up of old files
- %SERVER : archiving server output
- List of files in $DATAW, $DATAM and $TEMP directories
- Job accounting listing, full then summary

Sdout and sderr output streams from scripts are intermingled so that if an error occurs that initiates a Unix message, say a traceback in qsmain, this will place the Unix message after the "Starting script : qsmain" line.

4.18.3 Investigating failures during model execution

If an internal model check perceives that a gross error has occurred, an integer error code indicator is set and the model exits via a 'soft failure' in which control is passed back to the top level control routines in an ordered way. A partial error message is passed to the Unix level giving the name of the routine in which the error was trapped and an explanatory comment. A fuller error message is written to the Fortran output, and this can point immediately to the problem or offer no guidance or be downright obtuse! Bear in mind that the model is relatively robust to field values extending outside their normal range and that it may require some iterations of extreme values before an error is detected, such as by a negative absolute temperature being calculated. Also the error may be flagged in a routine some distance away in the calling tree from where the error originated. The atmosphere model will generally fail in routines LSP_FORM or SWRAD when meteorological prognostic fields become contaminated with wild values, leading to catastrophic growth of instabilities.

What to look at first

Obviously the output to see where the error occurred. The output should be scanned from the top to see what system messages are generated. The next place to look is from the bottom of the listing backwards, to determine how far the integration reached and where the error was trapped if a soft failure occurred. Note that in perverse cases with some hard failures the last buffer written to the output stream may be lost and the error point could be further on than indicated.

System messages on Cray platforms:

- "floating point exception" indicates that an attempt has been made to use an illegal value (NaN) in a floating point operation, usually because an element of an expression has not been initialised. ie this requires illegal values of A or B in:

  ```
  C=A*B        ! can generate floating point exception
  C=A          ! but a simple assign will not: no fail
  ```

- "ARLIB Maths library error", or similar, is invoked when an illegal value is input to a maths library routine such as LOG or SQRT.
- "Operand range error" normally indicates corruption or mis-alignment of argument list pointers for subroutine calls.
- "Failed %ALLOC..." states that too much memory is required. Valid increases in memory may arise from a change of science routine, or extra output diagnostics, or increased model field sizes, of course. However dynamic allocation of local arrays may pick up a corrupted dimension length, perhaps through incorrect initialisation or, more seriously, through overwriting of the address contents by an earlier error. For the atmosphere model it is possible to juggle performance and memory by changing the number of segments for physics calculations. With MPP, simply increasing the number of processors may solve the problem.

[On Cray machines a traceback using the debug utility is automatically invoked by UM scripts after a model failure; other platforms need this to be done manually. Searching for A_STEP will show which timestep the atmosphere model failed on. If this points to a failure in the first or second timestep, it is almost certain that there is an error in set-up, which could be small but serious.]

In MPP mode each processor continues independently until a barrier is reached. When a failure occurs on one processor, abort signals are sent to the remaining processors, which will usually be waiting at a barrier. The list of process states will indicate which processor initiated the abort, and will give some traceback information. Messages from other processors can be ignored.

Further investigations (for Cray users)

Tracebacks are only provided for routines compiled with the debug option z enabled with the Cray f90 compiler; relevant routines need to be explicitly re-compiled. Compiler options are set - via override files on the target machine - in window **subindep_Compile_User**

```
Sub-Model Independent
-> Compilation and Modifications
--> User defined compile overrides
```

Enabling the Cray f90 compiler option -ei sets unintialised variables to an indefinite number for every call to the routine. This provides a useful check for real variables since it forces a run-time error when an uninitialised local variable is used in a floating point or assign operation, although dummy arguments and common block variables are not included. Errors in integer variables cannot be trapped in the same way since all bit combinations are legal for integers. Other f90 compile or load options may also be relevant here.

The Cray f90 compiler run-time check option -R can also give useful information: -Rbc produces messages if array bounds are exceeded within a routine. -Rabc additionally compares the number and type of arguments passed between calling and called routines. However note that the UM produces a large number of messages pointing to inconsistencies between real and integer arguments across a routine call - these are due to the widespread practice of invoking extra arguments to access elements of mixed type arrays such as in lookup tables or the main D1 data array.

Problems still unresolved?

There is usually no alternative but to include an increasing number of write statements within the code to try to home in on the error. You may prefer to use a more sophisticated debug utility such as Cray's totalview but these need familiarisation and the size of the UM code can overwhelm some applications. Overwriting problems can be very difficult to track down. One useful technique is to include a write statement and/or calculations within routine TIMER, since this is called repeatedly throughout the control code and can sometimes identify when a problem begins. Otherwise go back to a model run that worked before you made your changes and gradually isolate components that are associated with the failure. The file comparison utility cumf may be useful in this exercise, see section 4.17.

Note that for MPP, UM code has been designed to give bit reproducible answers when a model run is repeated. This also holds true for repeated runs with different numbers of processors, with the exception of assimilation code. Comparing runs with different numbers and configurations of processors is a useful technique for exposing errors in new code.

4.18.4 Run-time diagnostics

Point prints of meteorological fields

Atmosphere model only. It is possible to obtain simple printed output of the primary meteorological variables for specified timestep intervals and located at a specific grid location, with neighbouring values also being optionally included. Maxima and minima of fields for each level of the model may also be obtained and is a useful monitoring facility. These are set up through windows **atmos_Control_Output_Writes_Patch** and **atmos_Control_Output_Writes_MaxMin**

```
Atmosphere
-> Control
--> Diagnostic (Point) writing
---> Patch writes AND Max-min prints
```

and can be switched on without re-compilation. More experienced users can insert code based on this method to obtain point prints after various stages of execution of the model timestep. For MPP, note that the diagnostics refer to the local domain of each processor separately, ie maxima and minima are not global values.

Timestep increments

Atmosphere model only. Selection of the UMUI window **atmos_Control_Output_Writes_Incr**

```
Atmosphere
-> Control
--> Diagnostic (Point) writing
---> Increment writes
```

allows the user to request a printout of statistics of field increments between a specified number of timesteps. This is useful for identifying the onset and location of the growth of instabilities. It adds significantly to the cost of execution and so should only be used in investigative mode. For MPP, results are gathered on a single processor and so the statistics refer to fields for the whole model domain.

Bit comparison of dumps

Testing for exact bit comparison between model dumps generated with and without a modification is a very stringent check on system only changes. [This test is strictest if prognostic fields written to dumps are specified as 64 bit rather than packed at 32 bit.] An option is provided to write out to disk model dumps at various stages through the model timestep, making it easier to pinpoint where a loss of comparability was first introduced. Select window **atmos_Control_Output_Writes_D1**

```
Atmosphere
-> Control
--> Diagnostic (Point) writing
---> Dumps from WRITD1
```

but beware that a large amount of data may be written out to files. The same facility is available in the ocean model from window **ocean_Control_Output_Writes_D1**. It is possible to extend this technique with user modification sets, allowing dumps to be written during execution of other model routines.

Note that bit comparison can be lost with:

- different compile optimisations (but a judicious use of brackets can force the compiler to evaluate in a particular order);
- different compiler options such as debug or runtime checking, which may affect optimisation;
- compiler upgrades;
- a change in order of any calculation;
- summations of a variable in a different order - care is needed for gathering results from processors in MPP mode;
- changing the dumping frequency of the model - conversion of moist conserved variables is not reversible.

4.18.5 Performance diagnostics (Cray users)

TIMER

A UM timing routine (TIMER) records how much cpu and elapsed time is spent in major model routines, controlled by software calls at the control level. Timings are given for the sum of calculations for called routines at a lower level and so the output refers to the cost of sections of code rather than individual routines. For MPP users, TIMER also returns statistics on all processors which gives some information on load balancing. Elapsed time is the most significant measure here, but may be distorted by contention with other jobs handled by the operating system.

Cray tools (Met. Office)

More information on Cray T3E tools for Met. Office users for general programs is available online . The most useful tool for debugging is totalview, which can show the contents of all variables at the point of failure from a core file and can also be set up to step through a program incrementally.

4.18.6 Test diagnostics

Atmosphere model only. A set of analytic fields can be calculated and produced as standard output through the STASH pp system. The diagnostic names and analytic formulae used for generated the fields are described in UMDP D7: Test diagnostics for output [33]. These special fields can be generated as individual diagnostics through STASH output panels of the user interface in the standard way. Alternatively, a macro has been set up to create a standard set of the test diagnostics and this can be initiated by accessing the window **atmos_STASH_Macros_Test**

```
Atmosphere
-> Control
--> STASH macros
---> Test diagnostic macro
```

Analytic diagnostics are useful when a system is being set up that requires a number of transfer or intermediate stages between UM runs and the final display of output.

Appendix A

The Portable UM

Author: Mussadiq Ahmed, Last Updated: 5 Aug 98

A.1 Introduction

Whenever the Met. Office upgraded their supercomputer, development of the UM was concentrated on optimising the performance of the model on that platform. In effect, the UM became hardwired to that computer. This made the UM impossible to run on other platforms, but this was not an issue as there were a very limited number of expensive computers that could run the UM.

As computers increased in power with more memory and faster processors, it was possible to run low resolution climate models and limited area models on workstations. As the cost of supercomputers decreased in real terms, more scientific organisations were purchasing such computers for research.

The decision was taken to create a portable version of the UM that would run on different supercomputer platforms and workstations. The portable UM is of great benefit to the Met. Office and to scientists around the world. It's distribution to the scientific community for a small licensing fee has increased the global awareness of the Met. Office and it's major product, the UM. The portable UM has allowed the scientific community to run their own weather models for the first time and carry out important research work. For some countries, the portable UM has been the first step in the feasibility of running their own limited area forecast model. Feedback from the scientific community to the Met. Office and to each other has resulted in enhancing the performance of the UM with scope for further improvement for the benefit of all. The Met. Office also acts as a point of contact to provide support for users.

The portable UM is used all over the world such as Poland, USA, Thailand, New Zealand, South Africa and Brazil. The portable UM has run on IBM and Fujitsu supercomputers and HP, SGI and Digital workstations.

With each release of the portable UM, new developments have improved the scientific capability, functionality and user-friendliness of the UM. The first available release of the portable Unified Model was vn3.2. It was based on a given set of fixed experiments which had been previously set up on the UKMO IBM mainframe User Interface. Vn4.0 of the Unified Model was the next release with the X-Windows based Unified Model User Interface. The UMUI gave users the ease and flexibility to modify the supplied experiments or design their own personal experiments. Vn4.4 is the current release with the ability to run the UM in MPP or non-MPP mode. This version also made extensive use of the make facility for compiling and creating the UM executables. Make allows users to carry out scientific research efficiently as they modify the UM source code with minimum effort.

As personal computers become more powerful and catch up with workstations in terms of speed and memory, one day the UM will run on a pc using public domain software for the Unix operating system and language compilers. Thus making the UM truly portable.

A.2 Porting the UM

The main criteria of the portable model is for the Fortran and C code to remain the same as that of the Unified Model. In other words, the Portable Unified Model is the Unified Model as used in the Met. Office.

The UM is tested in non-MPP mode on the different workstation platforms available in the Met. Office. First the UM is compiled on a workstation and then standard experiments are used for testing the UM. Such experiments are global climate runs and limited area models. This gives the opportunity to test aspects of the UM such as reconfiguration, STASH output, dumping and meaning. After successful execution of the experiments, output is compared with the equivalent Cray T3E experiments to look for differences in model output. The UM is also created and run on the Cray T3E in MPP and non-MPP mode using public domain software (ie. message passing utilities and fast fourier transforms) instead of Met. Office and Cray proprietary code.

When porting the UM issues to consider include:

1. The Operating system of the platform
2. Fortran and C code compiler
3. The source code management tool : Portable and Cray Nupdate
4. The Fortran code / C code interface
5. The MPP mode of the UM
6. Replacing proprietary code such as message passing utilities and fast fourier transforms with public domain equivalents
7. The data format of the platform

As external users use different combinations and versions of operating systems and code compilers, it is impossible to forsee all porting issues and difficulties that may arise. A user may come across a problem not seen by the Met. Office. So it is important for a user not to treat the portable UM as a black box, instead they will develop the skills to have a good understanding in how the UM operates and solve problems that occur on their computer.

Appendix B

The UK Met. Office UM

This section contains information which is only relevant to users at the Met. Office. This information will not be of interest to external users.

B.1 File Transfer

Author: D.M. Goddard, Last Updated: 19 May 98

B.1.1 Introduction

There are several routes for transferring UM data from the Cray T3E to other machines in the UKMO. Transfers are available to workstations both directly and via COSMOS. Section B.1.2 describes interactive methods for transferring UM dump files or fieldsfiles. Section B.1.3 describes batch methods for transferring UM fieldsfiles.

B.1.2 Interactive methods for transferring UM dumps and Fieldsfiles

These procedures are described in UMDP F5 [35].

Transfers from the Cray T3E to workstations

The following will transfer a UM file *file.64* into the home directory of frxx on fr0600.

- Convert UM file from 64-bit IEEE numbers to 32-bit IEEE numbers

```
$ /u/um1/vn4.4/utils/ieee -32 file.64 file.32
```

- Send 32-bit IEEE file to workstation.

```
$ rcp dump.ieee frxx@fr0600:~frxx/workstation_filename
```

- If output is required in PP format run *convpp* see section 4.17 for details.

Transfers from the Cray T3E to COSMOS

- Convert UM file from 64-bit IEEE numbers to 32-bit IEEE numbers

```
$ /u/um1/vn4.4/utils/ieee -32 file.64 file.32
```

- Send 32-bit IEEE file to COSMOS. *Note: quotes are important.*

```
$ /u/um1/vn4.4/scripts/cray2hds file.32 'cosmos_filename'
```

Transfers from COSMOS to Workstations

On a *DECNET* server workstation:

- Copy IEEE file from COSMOS to workstation.

```
$ hds2hp cosmos_filename dump.file
```

- If output is required in PP format run *convpp* see section 4.17 for details.

In NWP, the DECNET server workstations are fr0500, fr0800 and fr1300.

B.1.3 How to Transfer UM Fieldsfiles

There are a number of useful UNICOS and JCL routines for transferring data from the Cray T3E to COSMOS and on to the workstations. All the routines are set up to transfer the file scaab.pp4 from your home directory on the Cray T3E, to MS15.XXSCAAB.PP4 on COSMOS and thence to /temp/fr0600/frxx/scaab4.pp on the workstations. It is assumed throughout that your programmer letters are xx, your workstation account is on fr0600, your COSMOS prefix is MS15, your group is M11 and your TIC is zzzzz. Some or all of these will have to be changed before the routines can be used.

Transfers from the Cray T3E to COSMOS

The following job should be submitted on COSMOS to the Cray T3E (ISPF option N.R.1) after the model run is complete.

```
# QSUB -lT 60
# QSUB -lM 7Mw
# QSUB -q express
# QSUB -eo
#
UI=ruser=t11xx,racct='"(m11,xx,zzzzz)"'
cd $TMPDIR
cat >nlist <<EOF
 &PACK UNPACK=.FALSE.
 &END
 &TYPE OPER=.TRUE.
 &END
EOF
```

```
# PROGRAM to copy UM PP output to vbs fieldsfile #####
#
#Set up default aliases.
#
RUNID=scaab
PP=4
CRAYFILE=$HOME/$RUNID.pp$PP            # Input file
IBMFILE=ms15.xx$RUNID.pp$PP            # Output file
DIAGFILE=$TMPDIR/$RUNID.pp$PP.diag
TEMPFILE=$TMPDIR/$RUNID.pp$PP.cos
echo "Cray file to copy " $CRAYFILE
echo "IBM file          " $IBMFILE
#
export UNIT07=$DIAGFILE
export UNIT10=$CRAYFILE
export UNIT11=$TEMPFILE
/u/um1/vn4.4/exec/qxfieldcos < nlist >outfc
vbstext='text="dcb=(lrecl=x,blksize=10000,recfm=vbs),unit=disk'
vbstext=$vbstext,'space=(cyl,(10,10),rlse)'
vbstext=$vbstext,'storclas=scdatprk,mgmtclas=mcsnc4"'
cat outfc
#### check fieldcos o/p in case no fields have been transferred #####
#### search on line .....THE VALUE OF NENT IS       0  (if no.=0)####
head outfc | grep NUMBER | cut -f3 -d"S" > outfc1
read <outfc1 NENTRIES
if test $NENTRIES -eq 0
then
echo "Number of fields is $NENTRIES so abort file transfer and o/p"
exit
fi
#### end of fields no. check                                    #####
putibm $TEMPFILE $IBMFILE,stat=new,df=tb,$UI,$vbstext
CC=$?
if test $CC -ne 0
then
  echo " error in copy to vbs file on IBM "
  echo $vbstext
  exit $CC
fi
rm $TEMPFILE nlist outfc*
#     ###end###############################################################
```

Transfers from COSMOS to Workstations

The following JCL should be submitted to COSMOS. It will then transfer the file MS15.XXSCAAB.PP4 to the workstations, creating the file /temp/fr0600/frxx/scaab4.pp. In the job card PROGRAMMER.XTN should be replaced with your own name and extension number, eg, ANOTHER.4829. Towards the end of the job, SER='05' should be replaced with the number of your own server.

```
//T11XXI2H JOB (M11,XX,ZZZZZ),PROGRAMMER.XTN,PRTY=??,NOTIFY=T11XX,
// MSGCLASS=Q
//PROCLIB JCLLIB ORDER=(MS11.PROCLIB)
//*
//C1  EXEC PPHP3,
//  DATASET=MS15.XXSCAAB.PP4,
//  STRING='fr0600',
//  TIMESPAN='temp',
//  FILENAME='scaab4.pp',
//  SPACE=(10,10),
//  JOBID=T11XXHP0,
//  HPUID='frxx',
//  TSOUID=T11XX,
//  PREFIX=MS15,
//  SER='05'
//*
//
```

Automatic Transfers

PP files can be transferred to COSMOS and/or the workstations automatically by inserting a script at the end of a UM run. The script is defined by going into the UMUI window **subindep_PostProc_ScrIns**

```
Sub-Model Independent
-> Post Processing
--> Top and bottom script inserts
```

Here the user must specify both a top and bottom script both of these must be resident on the machine on which the model is being run. The top script should just be a dummy script, the bottom script follows.

If transfers are required to COSMOS or an HP then IBMTR or HPTR respectively should be set to be true. This script uses getibm (see man page) to pull another script across. It is these which actually carry out the transfers. They are assumed to be in MS15.XXUM.JOBS. They are also given here, the last two of the following three scripts. As quoted here this job transfers scaab.pp4 to both the workstations and COSMOS and scaab.pp2 to COSMOS only.

```
HPTR=true
IBMTR=true
#
######### start of hp rel
if $HPTR
then
  echo "Release scripts fppthp to output pp files"
  cd $TMPDIR
  getibm "ms15.xxum.jobs(fppthp)" fpp
  CC=$?
  if test $CC -eq 0
  then
#
    PR=8
    PP=4
    ./fpp
#
  else
```

```
    echo " error in getibm fpp"
  fi
  rm fpp
fi
######## end of hp release
#
######## start of ibm release
if $IBMTR
then
  echo "Release scripts fpptibm to output pp files"
  cd $TMPDIR
  getibm "ms15.xxum.jobs(fpptibm)" fpp
  CC=$?
  if test $CC -eq 0
  then
#
    PP=4
    ./fpp
#
    PP=2
    ./fpp
#
 else
    echo " error in getibm fpp"
  fi
  rm fpp
fi
######## end of ibm release
```

The following script should be in MS15.XXUM.JOBS(FPPTHP).

```
echo "Release script fpp to output pp$PP file"
UI=ruser=t11xx,racct='"(m11,xx,zzzzz)"'
cd $TMPDIR
cat >nlist <<EOF
 &PACK UNPACK=.FALSE.
 &END
 &TYPE OPER=.FALSE.
 &END
EOF
# PROGRAM to copy UM PP output to vbs fieldsfile #####
CRAYFILE=$DATAW/$RUNID.pp$PP                 # Input file
IBMFILE=ms15.dg$RUNID.pp$PP                  # Output file
DIAGFILE=$RUNID.pp$PP.diag                   # Diagnostic file
TEMPFILE=$RUNID.pp$PP.cos                    # cos file
export UNIT07=$DIAGFILE
export UNIT10=$CRAYFILE
export UNIT11=$TEMPFILE
$EXEC/qxfieldcos < nlist >outfc
vbstext='text="dcb=(lrecl=x,blksize=10000,recfm=vbs)'
vbstext=$vbstext,'storclas=scdatprk,mgmtclas=mcsnc4'
```

```
vbstext=$vbstext,'dataclas=dcdatvbs,space=(cyl,(180,60),rlse)"'
cat outfc
#### check fieldcos o/p in case no fields have been transferred #####
#### search on line .....THE VALUE OF NENT IS       0  (if no.=0)####
head outfc | grep NUMBER | cut -f3 -d"S" > outfc1
read <outfc1 NENTRIES
if test $NENTRIES -eq 0
then
echo "Number of fields is $NENTRIES so abort file transfer and o/p"
exit
fi
#### end of fields no. check                                    #####
putibm $TEMPFILE $IBMFILE,df=tb $vbstext
CC=$?
if test $CC -ne 0
then
  echo " error in copy to vbs file on IBM "
  echo $vbstext
  exit $CC
fi
#####Get lower case runid and convert to uppercase in RUNIDUC
cat >runlc <<EOF
$RUNID
EOF
tr abcdefghijklmnopqrstuvwxyz ABCDEFGHIJKLMNOPQRSTUVWXYZ <runlc >runuc
read <runuc RUNIDUC
#####Get current second to generate IBM job name so not identical
date '+%S' > date_second
read <date_second IBMNO
#####Generate dataset datename
date '+%d%m' > date_daymon
read <date_daymon DAYMON
########## IBM job ##############
cat >jclfile <<EOF
//T11XXH$IBMNO JOB (M11,XX,ZZZZZ),PROGRAMMER.XTN,
// PRTY=$PR,TIME=1,
// NOTIFY=T11XX,MSGCLASS=Q
//*
/*
//PROCLIB JCLLIB ORDER=(MS11.PROCLIB)
//  EXEC PPHP3,
//  DATASET=MS15.XX$RUNIDUC.PP$PP,
//  FILENAME='$RUNID$DAYMON.$PP',
//  SPACE=(10,10),
//  STRING='fr0600',
//  TIMESPAN='temp',
//  JOBID=T11XXH$IBMNO,
//  HPUID='frxx',
//  TSOUID=T11XX,
//  PREFIX=MS15,
//  SER='05'
```

```
//
EOF
echo "runid is $RUNIDUC"
ibmsub jclfile
CC=$?
if test $CC -eq 0
then
  echo " T11XXH$IBMNO job sent to IBM"
fi
  echo "condition code=$CC"
rm $TEMPFILE $DIAGFILE jclfile nlist runlc runuc outfc*
####end#########################################################
```

The next file should be placed in MS15.XXUM.JOBS(FPPTIBM).

```
echo "Release script fpp to output pp$PP file"
UI=ruser=t11xx,racct='"(m11,xx,zzzzz)"'
cd $TMPDIR
cat >nlist <<EOF
 &PACK UNPACK=.FALSE.
 &END
 &TYPE OPER=.FALSE.
 &END
EOF
# PROGRAM to copy UM PP output to vbs fieldsfile #####
CRAYFILE=$DATAW/$RUNID.pp$PP                 # Input file
IBMFILE=ms15.dg$RUNID.pp$PP                  # Output file
DIAGFILE=$RUNID.pp$PP.diag                   # Diagnostic file
TEMPFILE=$RUNID.pp$PP.cos                    # cos file
export UNIT07=$DIAGFILE
export UNIT10=$CRAYFILE
export UNIT11=$TEMPFILE
$EXEC/qxfieldcos < nlist >outfc
vbstext='text="dcb=(lrecl=x,blksize=10000,recfm=vbs),unit=disk'
vbstext=$vbstext,'space=(cyl,(10,10),rlse)'
vbstext=$vbstext,'storclas=scdatprk,mgmtclas=mcsnc4"'
cat outfc
#### check fieldcos o/p in case no fields have been transferred #####
#### search on line .....THE VALUE OF NENT IS       0  (if no.=0)####
head outfc | grep NUMBER | cut -f3 -d"S" > outfc1
read <outfc1 NENTRIES
if test $NENTRIES -eq 0
then
echo "Number of fields is $NENTRIES so abort file transfer and o/p"
exit
fi
#### end of fields no. check                                    #####
putibm $TEMPFILE $IBMFILE,df=tb $vbstext
CC=$?
if test $CC -ne 0
then
```

```
  echo " error in copy to vbs file on IBM "
  echo $vbstext
  exit $CC
fi
  echo "condition code=$CC"
rm $DIAGFILE $TEMPFILE1 nlist outfc*
 ###end##########################################################
```

B.2 Operational Archives

Author: Anette Van der Wal, Last Updated: 1 Apr 98

B.2.1 What's Available

There seem to be so many different operational archives, that even the experienced user might be daunted by the task of retrieving model output. A full list is available from the COSMOS dataset cop.core.info(zarchive). Many of these are temporary (less than 1 week) and are primarily for operational purposes.

Some of the most useful archives are:

1. The WGDOS Packed Archive (see section B.2.2) -
   contains limited model output fields.
2. The WGDOS Fieldsfile Archive (see section B.2.3) -
   contains all model output fields.
3. The 18-Month Cray (or Re-run) Archive (see section B.2.4) -
   contains initial analyses, acobs and boundary conditions which can be used for model reruns.
4. The Observation Archive (see section B.2.5) -
   contains 30 days of observations which were used in operational runs.
5. The MOPS Archive (see section B.2.6) -
   contains MOPS data for use in mesoscale model runs.

B.2.2 The WGDOS Packed Archive

This archive is provided by the *Working Group on the Development of the Operational Suite (WGDOS)*. Details of the available fields are given in the following COSMOS datasets:

Model Type	**COSMOS Dataset**
UM Analysis	cop.wgdos.fields(umanal)
UM Background	cop.wgdos.fields(umback)
UM Forecast	cop.wgdos.fields(umfcst)
Main	cop.wgdos.fields(recm)
Update	cop.wgdos.fields(recu)
Fine Mesh	cop.wgdos.fields(recf)
Global Wave	cop.wgdos.fields(recg)
Lim. Wave	cop.wgdos.fields(rece)
Changes	mcf.wgdos.text(changes)

There are 3 formats which the model data can be restored to:

Fieldsfile: The file is of the same format as model diagnostic output.

PP format: This is probably the most useful form to restore archived data to, since the UKMO plotting facilities deal with files of this format.

VBS format: Try this format if you don't need a lot of data description (the header is only 16 words long). See mcf.wgdos.cntl(painfo) for more details.

Technical information on these options is available in UMDP F3 [34] or UMDP Y8 [40], or you may want to read more about diagnostic output (see section 4.14).

A Basic Example

Suppose you want to restore a week's worth of PMSL and H500 forecast fields (01/08/97 to 07/08/97, say), at T+0 and T+24, from a 00Z global forecast, and write them to a PP style dataset. The following is a step-by-step guide to do this:

1. Enter the WGDOS panels
Log on to COSMOS and type *tso wgdos* on the command line. If this doesn't work, try *tso exec 'mcf.master.clist(wgdos)'*. This will bring up the WGDOS title panel. Hit *Enter*, to give the main WGDOS menu.

2. Find the field type and level code
Choose *option 1 - WGDOS*, *option 1 - WGDOS* again, and then *option F - Fields*. Select the type of data you require ... in this example, it is *option 3 - UM Fcst*. This is a complete list of all the fields archived. In the first column, to the left of the field name, is a number. This is the *field type* and will be coded as *ITYP* in the extraction job. So, for pressure, *ITYP*=12 and for height, *ITYP*=2. Along the top of this table is a list of levels (coded as *LEV*), and an *X* indicates which levels are archived for a particular field. For height, the *A* beneath the *500* signifies that this field is also available at T+96 and T+120. Next to *pressure* there is an *A* in the column headed *8888* (indicating mean sea level), which means that there are 2 additional forecast times for this field too. We require *LEV*=8888 (mean sea level) for pressure and *LEV*=500 (500hPa) for height.

3. Check for missing data
Before you go any further, it is imperitive that you check that the fields required are not missing. To do this, go back 2 panels, and choose *option M - Missing* followed by the type of fields you are interested in (*3* in this case). This list gives the dates and times of any fields which have been omitted from the archive, but be warned, it is not comprehensive. If you discover a field which is missing, but not on the list, let the archive manager know! Neither of the fields in this example are missing.

4. Look up cartridge details
The next step is to find out which cartridges contain the dates which you want to restore. From the top level menu, choose *option 3 - Extract2*. On this panel we would enter:

AGF for global forecast, *Y* for one month only, and then fill in the details of the month (MM=08, YY=97).

You will then be presented with a list of cartidge volumes and dataset names. Since these files are catalogued, you only need to remember the name, the last part of which tells you the first date that is on the tape. For the first week in August, we require 1 tape, namely *cop.paagf.j970729*.

5. Preparing the namelists
There are 2 namelists which are used to extract the fields from the archive.

- &DT
- &REST

The first defines the data time of the restore and the second defines the fields. Each &DT can have several &REST namelists associated with it. See *option 4 - Examples* on the top level menu, and then *option 5 - Further Hints* for more details. In this example, we would set them up as follows:

```
&DT IZ=00,IU=21,PP=.TRUE.,ALL=.TRUE. &END
&REST ITYP=2,ITIME=00,24,LEV=500 &END
&REST ITYP=12,ITIME=00,24,LEV=8888,LAST=.TRUE. &END
```

There are some parameters which we haven't mentioned yet ...

- IU : the logical unit number attached to the output file in the JCL.
- ALL : specify as true if you want to restore all dates from a dataset. This is often quicker than specifying each date separately, if you want several dates from one tape.
- LAST : this should be set to true on the last *&REST* namelist.

6. Setting up the JCL

The restores are done on the IBM, so JCL is needed. Check out the example in WGDOS panel 4.2. You need a separate *EXEC* statement for each dataset (or cartridge) you are trying to access. The final syntax for our restore job looks like this:

```
//M11AVRES JOB (M11,AV,FC03Z),NAME.EXT,PRTY=5,NOTIFY=T11AV,
//  MSGCLASS=Q,REGION=8000K
//*
//A1 EXEC PGM=PAREST
//GO.STEPLIB DD DSN=COP.PRODUCT.LOADLIB,DISP=SHR
//GO.FT05F001 DD *
 &DT IZ=00,IU=21,PP=.TRUE.,ALL=.TRUE. &END
 &REST ITYP=2,ITIME=00,24,LEV=500 &END
 &REST ITYP=12,ITIME=00,24,LEV=8888,LAST=.TRUE. &END
/*
//GO.FT06F001 DD SYSOUT=Q,DCB=(LRECL=133,BLKSIZE=133,RECFM=FBA)
//GO.FT21F001 DD DSN=MS15.AVTMP1,DISP=(NEW,CATLG),
//  STORCLAS=SCDATPRK,SPACE=(CYL,(2,1),RLSE),
//  DCB=(RECFM=VBS,BLKSIZE=15476)
//GO.FT81F001 DD DSN=COP.PAAGF.J970729,DISP=SHR,LABEL=(,,,IN)
//
```

Items in bold typeface should be replaced by your own account details. Retrieving fields is quite an expensive process, so it's best to use priority 5 whenever possible.

7. Restoring from UABRF

Only the most recent years of data is kept on CA-1 cartridge, which enables convenient access. Data older than about 18 months is archived to UABRF. This means that before use, it must first be restored to disk. The disk dataset is then entered on the GO.FT81F001 step of the job above and the job is run as before.

Naming convention for UABRF'd data is the same as for CA-1 datasets except the high level qualifier changes from COP to CFPROD. In this example, COP.PAAGF.J970729 would be renamed CFPROD.PAAGF.J970729.

The JCL for restoring from UABRF can be generated by choosing *option P - Restore* from the second level menu.

B.2.3 The WGDOS Fieldsfile Archive

This archive is basically a straight dump of fieldsfiles output from the UM. It contains all fields which are output operationally. See the COSMOS dataset cop.core.fflist to see what's available.

Depending on how much and what type of data you require, you may wish to use this archive. It is good for daily restores of fields not on the packed archive, but becomes expensive and time consuming if you want to restore data to form a monthly mean, say. Only 4 years worth of output is retained (on a rolling archive basis).

A Basic Example

Imagine the same example as in section B.2.2, but for just one day . . . 01/08/97. Here's what to do:

1. Enter the WGDOS panels as in section B.2.2

2. Restore Data

Choose *option N - WGDNEW* to go into the "new" archive. Select *option 3 - Restore*, which will take you through a set of panels designed to generate the JCL required. Remember that it is the full model fieldsfile which is restored, which can use anything up to 1,150 cylinders depending on model domain and resolution. In our example, we want the global 00Z fieldsfile, so we would enter the following (where the values in bold type should be replaced by your account details):

```
OPERATIONAL RUN TIME ===> G00
DAY  ===> 01
MONTH  ===> 08
YEAR  ===> 97
RESTORE TO HI-LEV INDEX  ===> MS15
PROGRAMMER LETTERS  ===> AV
```

. . . then hit *Enter*. At the next panel, hit *Enter* again, to get a list of all fieldsfiles available, or just enter CFPROD.FFCLGM.ARCHQG00.FF970801. Type *sub* on the command line to move on. At this next panel, you should fill in your jobcard details. The JCL can then be stored and edited before submitting. It should look like this . . .

```
//M11AVR01 JOB (M11,AV,FC03Z),DSM.FDR.RESTORE,REGION=4000K,PRTY=8,
//  NOTIFY=T11AV,MSGCLASS=Q,MSGLEVEL=(1,1),TIME=3
//*INFORM PRINTDATA
//JCL   JCLLIB  ORDER=(DSM.STORAGE.JCL)
//*
//* NEW WGDOS DIALOG GENERATED RESTORE JOB FROM USER ARCHIVE UABRF
//*
//TRY1 EXEC RESTORE
//RESTORE DD *
    SELECT DSN=CFPROD.FFCLGM.ARCHQG00.FF970801,NEWI=MS15.+AV,
           STORCLAS=SCDATPRK,PRESTAGE,COPY=1,NOTIFY=T11AV
/*
//
```

Items in bold will be replaced with your own account details if you go through the panels.

3. Extract required fields

You may already have a routine which extracts fields from a fieldsfile; it probably uses FFREAD. If not, then you can find out details of FFREAD and other access routines by selecting *option 5 - Access*.

Alternatively, Clive Jones has written a program which takes namelist input, to extract any number of fields. The namelist is called &FIND, and any of the pp header codes can be used to search for a field. In our example, the job would look like this . . .

```
//M11AVR01 JOB (M11,AV,FC03Z),NAME.EXT,REGION=4000K,PRTY=8,
//  NOTIFY=T11AV,MSGCLASS=Q,MSGLEVEL=(1,1),TIME=3
//*INFORM PRINTDATA
//JCL JCLLIB ORDER=(MS11.PROCLIB)
//*
//* JOB TO EXTRACT USER SPECIFIED FIELDS FROM A FIELDSFILE
//*
//EXTRACT EXEC PPP2G,OUTPUT=Q,BLK=,GOPGM=FFTOVAX,
//  LOADLIB='MS12.CJSSTLIB.LOADLIB'
//FT05F001 DD *
 &FIND MSFC=1,MSFT=0,SLEV=500 &END
 &FIND MSFC=8,MSFT=0,MSVC=128 &END
 &FIND MSFC=1,MSFT=24,SLEV=500 &END
 &FIND MSFC=8,MSFT=24,MSVC=128 &END
/*
//FT10F001 DD DSN=MS15.AV.FFCLGM.ARCHQG00.FF970801,DISP=(SHR,PASS)
//FT11F001 DD DSN=MS15.AVNEW.DATASET,DISP=(NEW,CATLG),
//   STORCLAS=SCDATPRK,MGMTCLAS=MCSNC4
```

Unlike the &REST namelists in section B.2.2, each &FIND namelist must be for a single field only i.e. lists for any of the parameters, such as MSFT=0,24,48,..., are not acceptable.

Note: The dataset attached to unit 10 is the input dataset i.e. the one that has just been restored in step 2. The dataset MS15.AVNEW.DATASET is a new dataset which will be created, and will contain the fields you have extracted from the fieldsfile. It is important that you delete the fieldsfile as soon as you have finished with it as it uses a large amount of disk space.

B.2.4 The 18-Month Cray Archive

This is the place from which to retrieve your data if you wish to do a model re-run. The latest 18 months worth of analyses, acobs, boundary conditions and MOPS data are stored for global, LAM and mesoscale models on a rolling archive. If you are running a case which is not on the archive, i.e. older than 18 months, then you will have to bring back the ECMWF analyses (see section B.3). The easiest way to set up a job to restore the data is to use the Observations User Interface (Obs UI, see UMDP U3 [38]) on COSMOS. This JCL can be stored and ammended for subsequent retrievals, although you are strongly advised to go back through the Obs UI panels, to ensure that the code is up to date, as developments are ongoing.

A Basic Example

Suppose you require analysis for a global 00Z re-run from 01/08/97. The following steps show you how to restore such data:

1. Enter the Observations User Interface

Log on to COSMOS. If you are in FS, CR or NWP, then the Obs UI will be on your branch applications panel (as specified in UMDP U3 [38]). If you are in any other division then you will need to change a parameter on one of the COSMOS log in panels, which will enable you to access the NWP Division branch menu, and enter the Obs UI. On the 3rd log in panel (entitled: "Welcome to Time Sharing Option"), change the *PROCEDURE* to *M11LPROC*.

WARNING: By changing the *PROCEDURE* parameter, you may find that you are no longer able to access some applications specific to your branch. Once you have restored the required data, it is advisable to restart your COSMOS session, by logging off, and on again.

The screen size is important. If you are using a COSMOS emulator (from an HP for instance), you will need a screen with 43 lines or more.

2. Create the JCL

Once you have gained access to the Obs UI, it is only a matter of filling in answers to the relevant questions. Firstly, choose *option 2 - Restore Datasets from the Cray Model Archive*, and then use the operational version of the panels. Once you have set up the JCL, you will be asked if you want to save it. It is a good idea to do so, since you will then have a reference point if a problem occurs. For our example, the JCL looks like this:

```
//M11AVRES JOB (M11,AV,FC03Z),NAME.EXT,PRTY=5,MSGCLASS=Q,
//  NOTIFY=T11AV
//*
//* Front End Job to restore datasets from Cray Model
//* Archive.
//*
//*INFORM PRINTDATA
//*
// EXEC PGM=FDRABR
//SYSPRINT DD SYSOUT=Q
//SYSPRIN1 DD SYSOUT=Q
//DISK1 DD STORCLAS=SCDATPRK
//TAPE1 DD UNIT=CART,DSN=COP.ARCHIVE.AM010897,
// DISP=OLD,LABEL=1
//ABRDUMMY DD DUMMY
//ABRWORK DD UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN DD *
  RESTORE TYPE=ABR
  SELECT DSN=COP.ANALYSIS.GLM,TAPEDD=1,STORCLAS=SCDATPRK,
  MGMTCLAS=MCSNC4,NEWN=MS15.AVANAL.QG00
/*
//
```

3. Tips

(i) The *NEWN* parameter is the dataset to which the restored data will be copied. Note that the Obs UI generates the same dataset name for restores of different dates, but similar field types. For instance, global 00z analysis for 01/08/97 will be copied to *MS15.AVANAL.QG00* (for me), as will global 00z analysis for any other date. Watch out for this, as trying to create a dataset which already exists will cause a failure.

(ii) Datsets created from this archive are **large**; it is best to transfer them to the Cray T3E asap, and remember to delete the copy on the IBM.

4. Useful Contacts

If you have any questions about the Obs UI, or if you are a first-time user, please contact Adam Maycock.

B.2.5 The 30-day Observation Archive

The primary use of the observation archive is to allow exact re-running of the operational OPS in order to investigate any operational failures. It is becoming more redundant as OPS improvements allow easier retrieval of observations from the MetDB.

For more information, contact the OPS system manager.

B.2.6 The MOPS Archive

MOPS data is used in the mesoscale model, and is processed satellite and radar imagery. The processing is done by Nimrod. Forecast data from a mesoscale run is sent over to Nimrod, who use that data plus Meteosat IR imagery and UK radar data to produce a 3d cloud cover analysis and a precipitation rate analysis. These analyses are then interpolated to the mesoscale UM grid (to produce a MOPS fieldsfile), converted to an ACOBS file, and used in the assimilation.

Files that are archived for each mesoscale run are..

Cloud analysis from Nimrod

Precip analysis from Nimrod

MOPS fieldsfile (Nimrod analyses interpolated to mes grid)

MOPS ACOBS file

The MOPS data archive is similar to the 18 month rerun archive (section B.2.4), the only differences are the names of datasets which can be found in section 8 of cop.core.info(zarchive).

B.3 Obtaining Data from the ECMWF Archives

Author: Anette Van der Wal, Last Updated: 30 Mar 98

B.3.1 The ECMWF MARS Archive

ECMWF store all of their observational, analysis and forecast data on a central file store. This archive contains operational and research data going back over ten years. A software interface known as MARS (Meteorological Archival and Retrieval System) provides a facility for accessing this data. Details on the contents of the ECMWF archive and how to use the MARS interface can be found in the MARS User Guide [43].

B.3.2 Running the UM from ECMWF Analyses

This section explains how to extract data from MARS and return the resulting file to the Met. Office's Unix systems. Only those programmers who have accounts on the ECMWF computer system can access MARS data.

The following is an example of the MARS request required to extract ECMWF analysis data. It will later be referred to as **the MARS request**:

```
#!/bin/ksh
rdate=19980330
rtime=12
rgrid="1.25/1.25"
rfile="an${rtime}_${rdate}"

cd $SCRATCH

mars <<EOR
retrieve,type=an,levtype=sfc,levelist=off,
 repres=gg,date=$rdate,time=$rtime,target="$rfile",
 param=st/sp/lsm/z,format=packed,grid=$rgrid,
 accuracy=normal.
retrieve,type=an,levtype=pl,
 levelist=1000/850/700/500/400/300/250/200/150/100/70/50/30/10,
 repres=sh,date=$rdate,time=$rtime,target="$rfile",
 param=t/u/v/r,format=packed,grid=$rgrid,
 accuracy=normal.
EOR
```

For Y2K compliance, it is important that the "year" is a 4 figure value. The user is reminded that the $HOME disk space at ECMWF is small. Hence it is good practice to keep restored data in the $SCRATCH area.

The job accesses the pressure level archive at ECMWF via the Met. Office-ECMWF link. The horizontal grid resolution and the analysis time and date may be changed via the parameters grid, time and date respectively. In addition, for dates prior to 15/7/86 where the surface geopotential is unavailable, param=st/msl/lsm must be coded for the surface field extraction.

If data on ECMWF hybrid vertical coordinates are required, then levtype=ml and param=t/u/v/q should be coded for upper air fields. Note that this option will not work if surface pressure and surface geopotential are not available.

For vn4.4 of the model, the reconfiguration expects land-sea mask (lsm) to be the 3rd field in the analysis file.

Use the eccopy command as follows to send data back to the Met. Office:

```
userid=frav
ip_address=151.170.5.6

eccopy -u $userid -h $ip_address $rfile
```

where the ip_address is that of the Met. Office host machine (from hereon referred to as **the host machine**), and userid is the users ID on that machine.

Currently, the host machines running eccopy are:

```
Cray T3E  - 151.170.3.140
fr0110    - 151.170.5.6
cf0200    - 151.170.3.199
hc0020    - 151.170.1.71
```

The user must have an account on the machine they wish to use as a host. If in doubt, check with Unix support.

Setting Up

A few things need to be set up before data retrieval and transfer can begin.

1. Create an eccopy directory
Using the eccopy command, data will automatically be returned to a directory called $HOME/eccopy on the host machine. This should be created by the user.

2. Create a .echosts file
In order for eccopy to work, the user requires a file called $HOME/.echosts on the host machine which should contain the line:

```
ecgate1.ecmwf.int frc
```

where frc is the ECMWF userid.

3. Create a .nqshosts file
In order for batch submission to work, the user requires a file called $HOME/.nqshosts on the ECMWF machine. It should contain some or all of the following:

```
hades frc
styx frc
ecgate1 frc
ecgate2 frc
ecgate1.ecmwf.int frc
ecgate2.ecmwf.int frc
ecbatch           frc
ecbatch.ecmwf.int frc
vpp700            frc
```

```
vpp700-fddi-00a   frc
vpp700-fddi-00    frc
vpp700-eth-00     frc
nimbus FRC
nimbus RQS_SYSTEM
nimbus.ecmwf.int FRC
nimbus.ecmwf.int RQS_SYSTEM
cirrus.ecmwf.int FRC
cirrus.ecmwf.int RQS_SYSTEM
cirrus FRC
cirrus RQS_SYSTEM
```

The important entries are those starting “ec”. FRC or frc should be replaced by the users ECMWF userid in the correct case.

4. A Warning about Shells

The default shell at ECMWF is the C shell, whereas most users at the Met. Office are more familiar with the Korn shell. Be aware of this when submitting jobs, especially if doing so in batch.

Interactive Retrieval

1. Telnet to ecgate1

This should work from any host machine, but currently does not work from the Cray T3E. Commands (especially editors) behave more predictably using an xterm (rather than HPterm). Enter userid and passcode at the prompt.

2. Access MARS

Create a Unix script which contains the MARS request and the eccopy command; make sure it is executable (chmod 755 script_file). Run the script. The data will be sent back to the host machine automatically.

Batch Retrieval

1. Create a valid certificate

The file $HOME/.eccert, on the host machine, contains userid and encrypted passcode information which is used in the batch commands. To create a new certificate, or update an old one, type eccert at the host prompt. At the time of writing, batch submissions are not possible from the Cray T3E.

2. Send a Batch Request

The command used for batch submission is ecqsub.

A simple submission would be something like

```
ecqsub -q ecgate -s /bin/ksh script_file
```

where script_file contains the MARS request and eccopy command.

Output, including stdout and stderr go into the users $HOME/eccopy directory by default. Unless renamed, stdout and stderr are given names like aaaaaaa.onnnn and aaaaaaa.ennnn where aaaaaaa are the first seven characters of the script file name, and nnnn is a unique 4-digit identifier (not controlled by the user).

3. Check or Delete a Queued Batch Request

This is done using the ecqstat and ecqdel commands.

Using the Data

If the data is on the HP, it can be copied to the Cray T3E using the rcp command. Alternatively, the user could get it sent to the Cray T3E directly using -h 151.170.3.140 in the eccopy command.

The data is in GRIB format, and as such can be read directly into the reconfiguration. From the UMUI choose

Window: **atmos_InFiles_Start**

```
Atmosphere
-> Ancillary and input data files
--> Start dump
```

and under “Using the reconfiguration”, select “The dump is in ECMWF GRIB format”.

Acronyms

3DVAR: Three dimensional variational analysis (the next generation data assimilation system being developed to replace the analysis correction scheme)

ACOBS: Observation files used for the analysis correction data assimilation component of the UM

ADCP: Acoustic Doppler Current Profiler

AVHRR: Advanced Very High Resolution Radiometer

BUFR: Binary Universal Form for the Representation of meteorological data. It is a standard developed by WMO for the efficient storage of meteorological observations in a machine independent form, where all the information to describe the observations is contained within the data.

C90: Cray 90 Supercomputer

CERFACS: European Centre for Advanced Research and Training in Scientific Computing

CFO: Central Forecasting Office

COSMOS: Front end computing system at the Met. Office, residing on an IBM mainframe computer, and separate from the HP workstation networks and Cray C90 supercomputer.

CR: Climate Research (Hadley Centre)

ELF: Equatorial Lat-long Fine-mesh

ECMWF: European Centre for Medium-Range Weather Forecasts

ERS: European Research Satellite

FOAM: Forecasting Ocean Atmosphere Model

FR: Forecasting Research (defunct), see NWP

FS: Forecasting Systems (old), see OS

GLOSS: Global locally-processed satellite soundings

GRIB: WMO standard for GRIdded Binary data. It is designed to provide efficient storage on data on a variety of grids

HP: Hewlett Packard. Brand of workstation currently used at the Met. Office.

JCL: Job Control Language (IBM)

JCMM: Joint Centre for Mesoscale Meteorology

LAM: Limited Area Model

LASS: Local Area Sounding System

MARS: Meteorological Archival and Retrieval System

MES: Mesoscale Model

MetDB: Meteorological Data Base (of observations prior to processing for the UM)

MLD: Mixed Layer Depth

MOPS: Moisture Observation Pre-processing System

MPP: Massively Parallel Processors

MSL: Mean Sea Level

MSLP: Mean Sea Level Pressure (see PMSL)

MVS: Multi Virtual Storage IBM mainframe operating system - in use at the Met. Office since 1979

NOAA: National Oceanic and Atmospheric Administration

NWP: Numerical Weather Prediction

OASIS: Ocean Atmosphere Sea Ice Soil; a distributed O-AGCM coupler developed at CERFACS.

OPS: Observation Processing System

OS: Operational Services

PL: Programmer Letters

PMSL: Pressure at Mean Sea Level

SST: Sea Surface Temperature

STASH: Spatial and Temporal Averaging and Storage Handling (storage handling and diagnostic system)

TKE: Turbulent Kinetic Energy

TIC: Task Identification Code (for administrative accounting)

UARS: Upper Atmosphere Research Satellite

UI: User Interface

UKMO: United Kingdom Meteorological Office

UM: Unified Model

UMACS: Unified Model Analysis Correction Scheme

UMDP: Unified Model Documentation Paper

UMUI: Unified Model User Interface

WBPT: Wetbulb Potential Temperature

WGDOS: Working Group on the Development of the Operational Suite

WGDUM: Working Group on Development of the Model

WMO: World Meteorological Organisation

References

[1] **Introduction to the Unified Forecast/Climate Model** ,
M.J.P. Cullen,
Unified Model Documentation Paper 1, version 7, 30 Aug 90.

[2] **Change Control Management of the Unified Model System**,
R. Rawlins,
Unified Model Documentation Paper 2, version 1.4, 26 Jan 95.

[3] **Software Standards for the Unified Model: Fortran and Unix**,
P. Andrews,
Unified Model Documentation Paper 3, version 7.2, 5 Feb 98.

[4] **Maintenance of Frozen Versions of the UM for Climate Users**,
F. Rawlins,
Unified Model Documentation Paper 4, version 2, 2 Nov 94.

[5] **Conservative Finite Difference Schemes for a Unified Forecast / Climate Model**,
M.J.P. Cullen, T. Davies, M.H. Mawson,
Unified Model Documentation Paper 10, version 27, 29 Mar 93.

[6] **Positive Definite Advection Scheme**,
M.J.P. Cullen, R.T.H. Barnes,
Unified Model Documentation Paper 11, version 6, 24 Jun 97.

[7] **Vertical Diffusion**,
C.A. Wilson,
Unified Model Documentation Paper 21, version 4.0, 12 Nov 92.

[8] **Gravity Wave Drag**,
C.A. Wilson, R. Swinbank,
Unified Model Documentation Paper 22, version 1, 1 Sep 91.

[9] **Radiation**,
W.J. Ingram, S. Woodward and J. Edwards,
Unified Model Documentation Paper 23, version 3, 25 Feb 97.

[10] **Subsurface, Surface and Boundary Layer Processes**,
R.N.B. Smith,
Unified Model Documentation Paper 24, version 2, 31 Mar 93.

[11] **Canopy, Surface and Soil Hydrology**,
D. Gregory, R.N.B. Smith, P.M. Cox,
Unified Model Documentation Paper 25, version 3, Nov 94.

[12] **Large-scale Precipitation**,
R.N.B. Smith, D. Gregory, J.F.B. Mitchell, A.C. Bushell, D.R. Wilson,
Unified Model Documentation Paper 26, version 4, 25 Feb 98.

[13] **Convection scheme**,
D. Gregory, P. Inness,
Unified Model Documentation Paper 27, version 2, 1 Mar 96.

[14] **Energy Correction**,
D. Gregory,
Unified Model Documentation Paper 28, version 1, Nov 98.

[15] **Calculation of Saturated Specific Humidity and Large-scale Cloud**,
R.N.B. Smith, D. Gregory, C. Wilson, A.C. Bushell,
Unified Model Documentation Paper 29, version 3, 3 Nov 97.

[16] **The Analysis Correction Data Assimilation Scheme**,
R.S. Bell, A.C. Lorenc, B. Macpherson, R. Swinbank, P. Andrews,
Unified Model Documentation Paper 30, version 4, 3 Mar 93.

[17] **The Ocean Model** ,
S.J. Foreman,
Unified Model Documentation Paper 40, version 1.

[18] **Ocean Model Mixed Layer Formulation** ,
S.J. Foreman,
Unified Model Documentation Paper 41, 17 Sep 90.

[19] **Penetrative Solar Radiation** ,
S.J. Foreman,
Unified Model Documentation Paper 42, 17 Sep 90.

[20] **The Thermodynamic / Dynamic Sea Ice Model** ,
J.F. Crossley, D.L. Roberts,
Unified Model Documentation Paper 45, version 1, 2 Oct 95.

[21] **Lateral Boundary Conditions in the Ocean Model** ,
D.J. Carrington,
Unified Model Documentation Paper 48, Version 1.0, 9 Oct 90.

[22] **Application of Surface Heat and Fresh Water Fluxes** ,
S.J. Foreman,
Unified Model Documentation Paper 50, 14 Sep 90.

[23] **Ocean Model Isopycnal Diffusion Scheme** ,
D.J. Carrington,
Unified Model Documentation Paper 51, 10 Oct 90.

[24] **Specification of Ancillary Fields**,
C.P. Jones,
Unified Model Documentation Paper 70, version 4, 4 Dec 95.

[25] **Ancillary File Creation for the UM**,
D. Robinson, C.P. Jones,
Unified Model Documentation Paper 73, version 4, 24 Feb 98.

[26] **The Top Level Control System** ,
T.C. Johns,
Unified Model Documentation Paper C0, version 9, 18 Nov 92.

[27] **Dynamic Allocation of Primary Fields**,
F. Rawlins,
Unified Model Documentation Paper C1, version 1.0, 6 Aug 93.

[28] **Atmosphere - Ocean coupling: technical overview**,
R. Rawlins,
Unified Model Documentation Paper C2, version 1.0, 30 Jun 97.

[29] **Storage Handling and Diagnostic System (STASH)**,
S.J. Swarbrick, K. Rogers,
Unified Model Documentation Paper C4, version 13.1, 6 Sep 98.

[30] **Control of Means Calculations**,
N.S. Grahame, R.A. Stratton,
Unified Model Documentation Paper C5, version 2, 20 Aug 92.

[31] **Updating Ancillary and Boundary Fields in the Atmosphere and Ocean Models**,
M.J.P. Cullen, C. Wilson,
Unified Model Documentation Paper C7, version 15, 8 Sep 94.

[32] **OASIS Coupling: Technical Overview**,
J-C Thil,
Unified Model Documentation Paper C8, version 1.

[33] **Test Diagnostics for Output**,
F. Rawlins,
Unified Model Documentation Paper D7, 19 Feb 93.

[34] **Format of Atmospheric, Oceanic and Wave Dumps, Fieldsfiles, Ancillary Data Sets, Boundary Data Sets and Observation Files**,
D. Robinson,
Unified Model Documentation Paper F3, version 24, 18 May 98.

[35] **Unified Model File Utilities**,
A. Dickinson,
Unified Model Documentation Paper F5, version 9, 9 Jul 96.

[36] **Technical Details of the Unified Model Data Assimilation Scheme**,
R.S. Bell, B. Macpherson, D. Robinson, P. Andrews,
Unified Model Documentation Paper P3, version 6, 22 Jun 95.

[37] **Interpolation Techniques and Grid Transformation Used in the Unified Model**,
A. Dickinson,
Unified Model Documentation Paper S1, version 11, 18 Feb 98.

[38] **Introduction to the Observations User Interface** ,
G. Bason,
Unified Model Documentation Paper U3, version 2, 17 Jan 95.

[39] **The Nupdate Source Code Manager**,
R. Krishna,
Unified Model Documentation Paper X2, version 2, 9 Nov 93.

[40] **Obtaining Output from the Unified Model** ,
P.J. Trevellyan,
Unified Model Documentation Paper Y8, version 1, 31 Aug 90.

[41] **System Maintenance Procedures**,
G. Henderson,
Unified Model Documentation Paper Z5, version 4, 25 Apr 96.

[42] **OASIS 2.0 User's Guide and Reference Manual**,
Laurent Terray et al.,
CERFACS, 42 av. Coriolis, 31057 Toulouse Cedex

[43] **MARS User Guide**,
ECMWF Computer Bulletin 6.7/2

[44] **A numerical method for the study of the circulation of the world ocean**
K. Bryan,
J. Comp. Phys., **3**, pp. 347-376. (1969)

[45] **Accelerating the convergence to equilibrium of ocean climate models**
K. Bryan,
J. Phys. Oceanogr., **14**, pp. 666-673. (1969)

[46] **A primitive equation, three-dimensional model of the ocean**
M.D. Cox,
GFDL Ocean Group Technical Report no. 1, Princeton, NJ, USA. (1984)

[47] **Isopycnal mixiing in ocean circulation models**
P.R. Gent and J.C. McWilliams,
J. Phys. Oceanogr., **20**, pp. 150-155. (1990)

[48] **Isoneutral diffusion in a z-coordinate ocean model**
S.M. Griffies, A. Gnanadesikan, R.C. Pacanowski, V.D. Larichev, J.K. Dukowicz and R.D. Smith,
J. Phys. Oceanogr., **28**, pp. 805-830. (1998)

[49] **The Gent-McWilliams skew flux**
S.M. Griffies,
J. Phys. Oceanogr., **28**, pp. 831-841. (1998)

[50] **A one-dimensional model of the seasonal thermocline. Part II.**
E.B. Kraus and J.S. Turner,
Tellus, **19**, pp. 98-105. (1967)

[51] **Ocean vertical mixing: a review and a model with a nonlocal boundary layer parametrization**
W.G. Large, J.C. McWilliams and S.C. Doney,
Rev. Geophys., **32**, pp. 363-403. (1994)

[52] **A stable and accurate convective modelling procedure based on quadratic upstream interpolation**
B.P. Leonard,
Cmput. Methods Appl. Mech. Eng., **19**, pp. 59-98. (1979)

[53] **Parameterisation of vertical mixing in numerical models of tropical oceans**
R.C Pacanowski and S.G.H Philander,
J. Phys. Oceanogr., **11**, pp. 1443-1451. (1981)

[54] **On the parameterization of equatorial turbulence**
H. Peters, M.C. Gregg and J.M. Toole,
J. Geophys. Res., **93**, pp. 1199-1218. (1988)

[55] **A fast and complete convection scheme for ocean models**
S. Rahmstorf,
Ocean Modelling, **101**, pp. 9-11 (Unpublished Manuscript). (1993)

[56] **Ocean isopycnal mixing by coordinate rotation**
M.H. Redi,
J. Phys. Oceanogr., **12**, pp. 1154-1158. (1982)

[57] **Do we require adiabatic dissipation schemes in eddy-resolving ocean models?**
M.J. Roberts and D. Marshall,
J. Phys. Oceanogr., **28**, pp. 2050-2063. (1998)

[58] **A scheme for predicting layer clouds and their water contents in a General Circulation Model**
R.N.B. Smith,
Q.J.Roy. Meteorol. Soc., **116**, pp. 435–460. (1990)

[59] **On the specification of eddy transfer coefficients in coarse resolution ocean circulation models**
M.J. Visbeck, J. Marshall, T. Haine and M. Spall,
J. Phys. Oceanogr., **27**, pp. 381-402. (1997)

[60] **Timestep sensitivity and accelerated spinup of an ocean GCM with a complex mixing scheme**
R.A. Wood,
J. Atmos. Ocean. Technol., **15**, pp. 482-495. (1998)

[61] **Modelling of land surface processes and their influence on European climate**
D.A. Warrilow, A.B. Sangster and A. Slingo,
Dynamical Climatology Technical Note 38, 92pp, UK Met Office (1986).

[62] **A global archive of land cover and soils data for use in general circulation climate models**
M.F. Wilson and A. Henderson-Sellers,
Journal of Climatology, 5, 119–143 (1985).

[63] **The impact of new land surface physics on the GCM simulation of climate and climate sensitivity**
P. Cox, R. Betts, C. Bunton, R. Essery, P. Rowntree and J. Smith,
Climate Research Technical Note 82, Hadley Centre for Climate Prediction and Research (1998).

[64] **Experiments with the assimilation of thermal profiles into a dynamical model of the Atlantic Ocean**
M.J. Bell,
Forecasting Research Technical Report 134, UK Met Office (1994).

[65] **Altimetric assimilation with water property conservation**
M. Cooper, and K. Haines,
J. Geophys. Res., 101, C1, 1059-1077, 1996.

[66] **Experiments with the assimilation of surface temperature and thermal profile observations into a dynamical model of the Atlantic Ocean**
R.M. Forbes,
Forecasting Research Technical Report 167, UK Met Office (1995).

[67] **Initial results from experiments assimilating satellite altimeter data sea surface height data into a tropical Pacific ocean model**
R.M. Forbes,
Ocean Applications Technical Note 12, UK Met Office (1996).

[68] **The Meteorological Office analysis correction data assimilation scheme**
A.C. Lorenc, R.S. Bell and B. MacPherson,
Quart. J. Roy. Meteor. Soc., 117, 59-89, 1991.

[69] **Iterative analysis using covariance functions filter**
A.C. Lorenc,
Quart. J. Roy. Meteor. Soc., 118,569-591, 1992.