Technical Report No. 19

Description of the Parallel Isopycnal Primitive Equation OGCM

PIPE

with models for

Sea Ice and Snow, Mixed Layer and Tides

coupled to an

wet atmospheric energy balance model

with

river runoff, soil and glaciers

Josef Maximilian Oberhuber
Deutsches Klimarechenzentrum GmbH
Bundestraße 55, D-20146 Hamburg

Edited by:
Deutsches Klimarechenzentrum GmbH
Modellbetreuungsgruppe

Hamburg, April 1999
Revision No. 1

ISSN 0940-9327

Contents

List of Tables

List of Figures

Summary Page

Short description of the model

The name PIPE is derived from **P**arallel **I**sopycnal **P**rimitive **E**quation model. Its origin is the OPYC model. It has been renamed because it now is fully portable between parallel shared and distributed memory computers as well as between vector and scalar ones. In addition it experienced significant changes in numerics and physical complexity.

The concept of using isopycnals as the vertical coordinate system for an OGCM is based on the observation that the interior ocean behaves rather like a conservative fluid. Even over long distances the origin of water masses can be traced back by considering the distribution of active or passive tracers. Isopycnal coordinates have the advantage that diapycnal mixing by numerical discretization errors is explicitly excluded, which may cause a too diffusive thermocline in conventional z-coordinate models that don't use advanced correction techniques.

Treating the ocean as a conservative fluid fails in areas of significant turbulence activity such as the surface boundary layer. Therefore, a surface mixed layer is coupled to the isopycnic interior ocean in order to represent near-surface vertical mixing and to improve the response time scales to atmospheric forcing which is controlled by the mixed layer thickness. In addition, parameterizations for vertical mixing in the interior ocean and a convection mechanism completes the model.

Since the model is designed for climate studies on large scales, a snow and sea ice model with rheology is included and serves the purpose of decoupling the ocean from extreme high-latitude winter conditions and promoting a realistic treatment of the salinity forcing due to melting or freezing of sea ice.

An atmospheric energy balance model (EBM) is forced by atmospheric winds and cloudiness, and predicts some vertical mean temperature and humidity. Horizontal eddy diffusivity is parameterized. Further contributions to the heat budgets as radiative fluxes, diabatic and adiabatic heating, and turbulent surface fluxes are diagnostic variables. In order to close the global heat budget evaporation and precipitation are linked with the sensible heat budget. Simple models of river runoff and glaciers on land close the global heat and water budget.

A barotropic tide model coupled to the 3-dimensional ocean via bottom drag and residual currents improves the circulation in regional application.

The open boundary formulation allows to feed the model with temperature and salinity from 3-dimensional ocean data. By assuming some barotropic mode along the open boundaries multiple disconnected boundaries can be used to compute the ocean state in a fraction of an ocean basin.

Author of the model

Josef Maximilian Oberhuber, Deutsches Klimarechenzentrum GmbH

Person responsible for model support at the DKRZ

Josef Maximilian Oberhuber [1], DKRZ Model Support Group

[1] valid till 31 July 1999

Changes relative to OPYC2.17, see DKRZ Rep. No. 7

Changes in Physics

- The parameterizations for the mixed layer and internal mixing have been altered to avoid too deep mixed layers in the subtropics
- The assignment of water masses to specific isopycnal layers has been reformulated for all vertical exchange processes such as for mixed layer, internal mixing and convection
- The radiation parameterizations are improved in order to obtain a more realistic climate sensitivity

Additional Physic Modules

- An atmospheric energy balance model (EBM) has been included, which considers a closed budget of sensible and latent heat and contains models for soil, river runoff and glaciers
- A tide model is coupled via bottom drag and residual currents to the 3-dimension ocean model

Changes in Numerics

- Alternating direction line-relaxation scheme for momentum and mass instead of line-relaxation in x-direction only for better convergence
- Exact mass conservation provided that the wave equation has converged

Additional Technical Tools

- Open boundaries now allow to specify multiple disconnected vertical sections, on which ocean data like temperature and salinity can be prescribed
- A tracer model can be run in an online or offline mode in order to compute further tracers

Adjustment to specific computer architectures

- The code now runs efficiently on vector and RISC processors
- The code now runs on parallel shared and distributed memory architectures

Changes in Pre- and Postprocessing

- The installation is highly simplified
- The data postprocessing now is supported by a big number of usefull tools

Changes in Supplied Data

- The ECMWF reanalysis data are available now

Part I

Model Description

Chapter 1

Model Physics and Dynamics

Remark:
This manual is based on the model initially published by Oberhuber(1990,1993a,1993b) henceforth called JMO. An updated version was documented by Oberhuber (1993c). Updated versions have of OPYC have been used for a couple of key studies like those published by Miller et al. (1992), Miller et al. (1994a,b), Aukrust et al. (1995), Holland et al. (1996a,b), Lunkeit et al. (1996), Cherniawsky and Oberhuber (1996), Roeckner et al. (1996), Bacher et al. (1998), Zhang et al. (1998), Timmermann et al. (1998,1999) or Oberhuber et al. (1998), or have produced significant results during a BMBF-project (Nies et al., 1998) for the problem of radioactive tracers in the Arctic oceans.

This manual describes the full source code of OPYC Cycle 3.10, which is called the **P**arallel **I**sopycnal **P**rimitive **E**quation (**PIPE**) model.

In the first chapter the model is described, in the second chapter the model numerics is discussed, in the third chapter flow diagrams show how the model is organized, in the fourth chapter, the model code is described, and finally the fifth chapter explains how PIPE is to be installed and applied to self-defined applications. Part of this last chapter also is the usage of the postprocessing software.

Notation: The subsequent descriptions reference the code in the following way:

- a variable is written in italic type, e.g, *TEMP* for potential temperature
- a SUBROUTINE is referenced by enclosing the name within '<' and '>', e.g., <SOLVERX> for the solution of the block-tridiagonal system in x-direction
- a BLOCK DATA program unit is abbreviated by enclosing the name by '|', e.g., |SCREW| for defining the time step
- a COMMON block name is enclosed by '/', e.g., /JMOBAS/ which contains the prognostic variables of the model
- a file or directory name is enclosed by '[' and ']', e.g., [ocestep.f] for the kernel of PIPE

1.1 The Interior Ocean Model

PIPE is a general circulation model based on the following ideas: Isopycnals are used as Lagrangian vertical coordinates, a realistic equation of state is included, the primitive equations together with the hydrostatic approximation are applied and a surface mixed layer, and a snow and sea ice model are coupled to the interior ocean. The equation of state together with the

sea ice model allows the model to be used on global scales. No approximation is made (except the hydrostatic approximation) that prohibits its application down to eddy-resolving scales.

Ignoring for the moment dissipative processes, the basic quantities which should be conserved are momentum, energy, mass, potential vorticity, heat and salt content. Momentum, mass, heat, and salt content are easily conserved if the flux form of the equations is chosen. Use of the flux form is crucial, otherwise conservation is hard to maintain in a moving coordinate system. Potential vorticity and energy can also be conserved after some manipulation of the momentum transport and Coriolis terms. See further discussion in JMO.

In brief, the philosophy of PIPE is to

1. not assume that the model a priori uses isopycnals. Therefore, the model is allowed to deviate from isopycnal coordinates if it is necessary to avoid an otherwise occuring conflict in the concept.

2. use two densities, which is the in situ density allowing for mass conservation in conjunction with a proper continuity equation, and the potential density, which is needed to formulate pressure gradients in the curved isopycnal coordinates and to control the coordinate system through mixing.

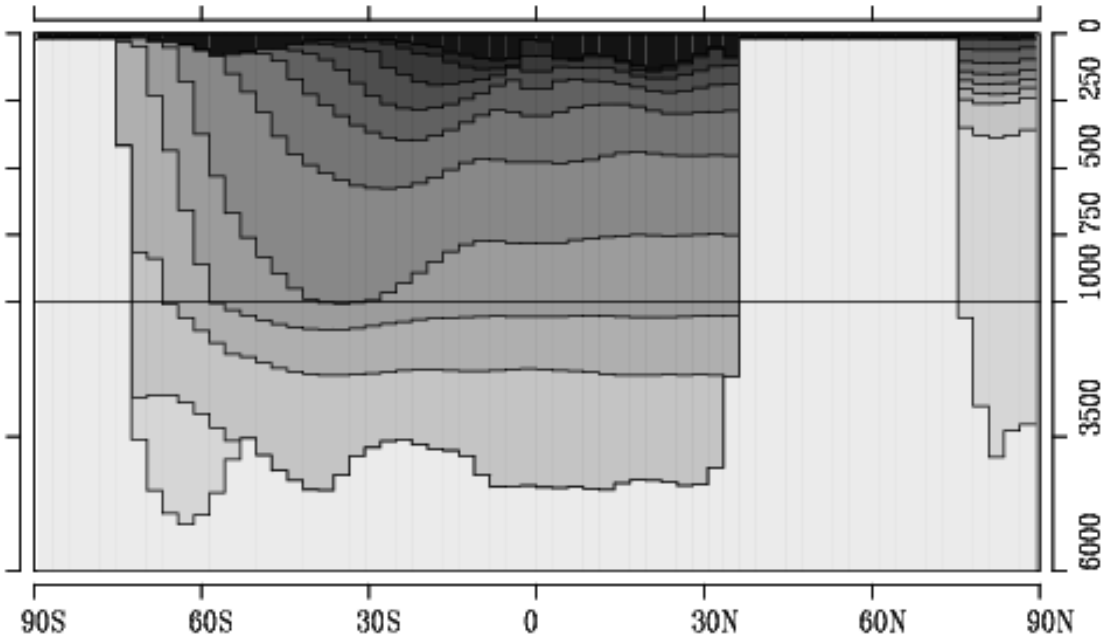

Figure 1.1: Meridional cross-section of interface height for the example of a T42 global ocean simulation. The section is located at 120W. The interfaces are plotted as steps in order to show, where grid cells are located. Each step denotes the location of a vector point. The brightest shades denote sections through Antarctica and North America.

1.1.1 Ocean Model Equations

The basic equations are formulated in flux form as conservation equations for the vertical mean of the mass flux $(\rho\vec{v}h)_k$, the mass content $(\rho h)_k$, the heat content $(\theta\rho h)_k$ and the salt content $(S\rho h)_k$ in a column of the k-th layer:

$$\begin{aligned}\frac{\partial}{\partial t}(\rho\vec{v}h)_k = & - \vec{\nabla}(\vec{v}_k(\rho\vec{v}h)_k) - h_k\vec{\nabla}p_k - \vec{f}\times(\rho\vec{v}h)_k + \vec{\nabla}\cdot((A^v\rho h)_k\vec{\nabla}(\vec{v}_k - \vec{v}_G)) \\ & + (w\vec{\rho v})_k^{k+} + (w\vec{\rho v})_k^{k-} - (w\vec{\rho v})_{k+}^{k} - (w\vec{\rho v})_{k-}^{k} + \vec{\tau}_k^{k-} + \vec{\tau}_k^{k+} \qquad (1.1)\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial t}(\rho h)_k &= -\vec{\nabla}\cdot(\rho\vec{v}h)_k + \vec{\nabla}\cdot((A^h\rho)_k\vec{\nabla}h_k) + R_k^h \\
&+ (w\rho)_k^{k+} + (w\rho)_k^{k-} - (w\rho)_{k+}^k - (w\rho)_{k-}^k \qquad (1.2)\\
\frac{\partial}{\partial t}(\theta\rho h)_k &= -\vec{\nabla}\cdot((\theta\rho h)_k(\vec{v}_k + \vec{v}_{res})) + \vec{\nabla}\cdot((A^s\rho h)_k\vec{\nabla}\theta_k) + \frac{Q_k}{c_p} \\
&+ (w\rho\theta)_k^{k-} + (w\rho\theta)_k^{k+} - (w\rho\theta)_{k-}^k - (w\rho\theta)_{k+}^k \qquad (1.3)\\
\frac{\partial}{\partial t}(S\rho h)_k &= -\vec{\nabla}\cdot((S\rho h)_k(\vec{v}_k + \vec{v}_{res})) + \vec{\nabla}\cdot((A^s\rho h)_k\vec{\nabla}S_k) + R_k^S \\
&+ (w\rho S)_k^{k-} + (w\rho S)_k^{k+} - (w\rho S)_{k-}^k - (w\rho S)_{k+}^k \qquad (1.4)
\end{aligned}$$

The terms $(w\vec{\rho v})$, $(w\rho)$, $(w\rho\theta)$ and $(w\rho S)$ describe exchange processes of momentum fluxes, mass itself, heat and salt content between neighbouring layers. $\vec{v}_G$ are barotropic velocities due to Greg Holloway's Neptun effect and $\vec{v}_{res}$ is the time-averaged residual flow computed by the barotropic tide model. The different kinds of exchange processes are entrainment/detrainment, vertical exchange as cross-isopycnal mixing, and convection. The terminology $(...)_k^l$ indicates a transfer from the l-th layer into the k-th layer, where l=k- represents the next upper and l=k+ the next lower, physically present layer. N is the number of layers, the index k starting with k=1 in the uppermost layer, which always is the mixed layer. All terms in which 1- or N+ occur are set to zero, except for the term $\vec{\tau}_1^{1-}$ which represents the surface wind stress and the term $\vec{\tau}_N^{N+}$ which represents the bottom stress. The forcing function Q represents the heat flux, R^h the fresh water forcing and R^S the forcing due to the sea ice-ocean coupling. c_p is the specific heat capacity of water.

The parameter f in Eqn. (1.1) is the Coriolis parameter, defined as

$$f = 2\Omega\sin(\varphi) \qquad \text{with} \qquad \Omega = \frac{2\pi}{86164} \qquad (1.5)$$

where Ω is the angular velocity of the rotating earth.

The stress $\vec{\tau} = (\tau_x, \tau_y)$ between neighbouring layers is set to zero. Note that vertical diffusion causes stress between layers due to vertical mixing of momentum. Since observed surface stresses are used, the surface drag coefficient does not need to be specified, however, because vector wind is not provided through observed data, however, required for a few operation, stress is converted to wind and vice versa assuming some surface drag coefficient. Bottom stress is parameterized also using some drag coefficient.

In order to complete the equations, the in situ density ρ and the potential density σ_θ are related with temperature T, salinity S and pressure P through the equation of state for sea water (UNESCO,1981) written symbolically as

$$\rho_k = \rho(T_k, S_k, P_k) \qquad and \qquad \sigma_{\theta k} = \rho(T_k, S_k, P_{ref}) \qquad (1.6)$$

where P_{ref} is the reference pressure. After discretization, the in situ pressure in the first layer is given as a vertical mean by

$$P_1 = \frac{g}{2}(\rho h)_1 + P_{atm} \qquad (1.7)$$

where g is the gravitational constant and P_{atm} is the atmospheric pressure, which optionally can be added to the oceanic pressure field. In all deeper layers the in situ pressure P_k is recursively defined by

$$P_k = P_{k-1} + \frac{g}{2}((\rho h)_{k-1} + (\rho h)_k) \qquad (1.8)$$

The pressure gradient in a layer k is computed from the sum of gradients due to the sea level gradient, the gradients of the interface heights above the layer k and the potential density

gradients integrated from the surface down into the center of layer k. The diagnostic equation for the pressure gradient is:

$$\vec{\nabla} p_k = g\rho_k(\sum_{l=k}^{N} \vec{\nabla} h_l + \vec{\nabla} D) + g\frac{\rho_k}{\sigma_{\theta k}}(\sum_{l=1}^{k-1} \vec{\nabla}(h\sigma_\theta)_l + \frac{h_k}{2}\vec{\nabla}\sigma_{\theta k}) \tag{1.9}$$

where D is the height of the topography above some reference level. The concept is to let the pressure gradient be valid for an arbitrary distribution of density and potential density in each layer. As explained in JMO this results in keeping the model dynamics valid in situations when isopycnal coordinates cannot be maintained. Therefore, gradients of potential density σ_θ are kept, which is not necessary if one a priori assumes that the vertical coordinates always are isopycnal. However, the latter a priori assumption leads to conceptual difficulties in the formulation of vertical mixing processes.

A_k^v, A_k^h and A_k^s represent the spatial and time-dependent diffusion coefficients for vector, layer and tracer quantities. Following Smagorinsky (1963), the horizontal diffusion coefficient for temperature and salinity is parameterized by being related to the deformation of the flow field:

$$A_k^v = A^{s,1}\sqrt{(\frac{\partial u_k}{\partial x} - \frac{\partial v_k}{\partial y})^2 + (\frac{\partial u_k}{\partial y} + \frac{\partial v_k}{\partial x})^2} + A^{v,0} \tag{1.10}$$

$$A_k^h = A^{h,1}\sqrt{(\frac{\partial u_k}{\partial x} - \frac{\partial v_k}{\partial y})^2 + (\frac{\partial u_k}{\partial y} + \frac{\partial v_k}{\partial x})^2} + A^{h,0} \tag{1.11}$$

$$A_k^s = A^{s,1}\sqrt{(\frac{\partial u_k}{\partial x} - \frac{\partial v_k}{\partial y})^2 + (\frac{\partial u_k}{\partial y} + \frac{\partial v_k}{\partial x})^2} + A^{s,0} \tag{1.12}$$

The values for $A^{s,1}$, $A^{h,1}$ and $A^{v,1}$ are constants and have to be chosen according to the grid resolution (see also table 4.5). The diffusion coefficients $A^{v,0}$, $A^{h,0}$ and $A^{s,0}$ depend on the grid spacing which are variable in longitude and latitude. For momentum, the underlying idea is to adjust the momentum diffusion such that the resulting frictional boundary layer is thick enough to be resolved by the grid. This is achieved by relating the actual grid distance with Kelvin and Rossby deformation radii. In the limit-case of infinite resolution diffusion for vector and scalar quantities becomes comparable. For scalar quantities such an automatic adjustment is not necessary. The approach is:

$$A_x^{v,0} = A_v(1 + \frac{\Delta x^2\beta}{c})(1 + \frac{(\Delta x f)^2}{c^2}) \tag{1.13}$$

$$A_y^{v,0} = A_v(1 + \frac{\Delta y^2\beta}{c})(1 + \frac{(\Delta y f)^2}{c^2}) \tag{1.14}$$

$$A^{s,0} = A_s \tag{1.15}$$

where $A^{v,0} = (A_x^{v,0}, A_y^{v,0})$. Here Δx and Δy are the grid distances in the spherical coordinate system, c is some phase velocity and β is the planetary vorticity. This formulation ensures a reasonable and automatic choice of diffusion coefficients in low and high resolution areas as well as in low and high latitudes. By relating the grid distance with the deformation radii for Kelvin and Rossby waves a factor is found that enlarges the diffusion in areas where the Kelvin or Rossby deformation radius is smaller than the grid resolution. According to Munk's theory the diffusion coefficient is responsible for the width of the western boundary currents. If the width is enlarged to a value comparable to the grid resolution a discrete model then is able to resolve western boundary currents. Therefore, if A_v once is tuned it is an universal constant of the model and thus there is no need to retune it if the model uses another resolution. A wellcome side-effect of momentum diffusion being bigger than scalar diffusion is that the two

separate solutions on the B-grid are coupled to each other. A similar null-mode does not exist for the variables $(\theta\rho h)$ and $(S\rho h)$.

In order to use the correct diffusion operators, these have been modified such that

- for momentum, $\vec{\nabla} \cdot A_k^v \vec{\nabla}(\rho \vec{v} h)_k$ has been exchanged by $\vec{\nabla} \cdot (A^v \rho h)_k \vec{\nabla} \vec{v}_k$

while for heat and salt, the respective operators $\vec{\nabla} \cdot (A^s \rho h)_k \vec{\nabla} \theta_k$ and $\vec{\nabla} \cdot (A^s \rho h)_k \vec{\nabla} S_k$ have already been used all the time, however, not been correctly documented in JMO. The diffusion operator for momentum is still available as option to the newer momentum conserving velocity diffusion (see Eqn. 1.1). The new one is recommended because it gives a significant improvement of eddy physics.

In case the barotropic tide model is activated, the bottom stress is computed from $\vec{\tau} = \rho_w c_d \sqrt{(u + u_{tide})^2 + (v + v_{tide})^2} (\vec{v} + \vec{v_{tide}})$.

1.1.2 Equation of State

The potential temperature θ is related with in situ temperature T, in situ pressure P and salinity S through the formula by Bryden (1973):

$$\begin{aligned} \theta = T \quad & - \quad P' \, (a_{100} + a_{101} \, T' + a_{102} \, T'^2 + a_{103} \, T'^3 + a_{110} \, S' + a_{111} \, S' \, T') \\ & - \quad P'^2 \, (a_{200} + a_{210} \, S' + a_{201} \, T' + a_{202} \, T'^2) \\ & - \quad P'^3 \, (a_{300} + a_{301} \, T') \end{aligned} \tag{1.16}$$

where $P' = \frac{P}{100}$, $S' = S - 35$ and $T' = T - 273.16$. All coefficients a with subscripts are defined in Appendix C. The temperature T is computed from the prognostic potential temperature θ by inverting the formula by Bryden using a Newtonian technique.

The in situ density is computed using the UNESCO formula for sea water, which is defined through the following relations:

$$\begin{aligned} \rho_w &= a_0 + T'(a_1 + T'(a2 + T'(a_3 + T'(a_4 + T'a_5)))) \\ k_w &= e_0 + T'(e_1 + T'(e_2 + T'(e_3 + T'e_4))) \\ a_w &= h_0 + T'(h_1 + T'(h_2 + T'h_3)) \\ b_w &= k_0 + T'(k_1 + T'k_2) \\ b &= b_w + S(m_0 + T'(m_1 + T'm_2)) \\ a &= a_w + S(i_0 + T'(i_1 + T'i_2)) + j_0 S' \\ r_0 &= k_w + S(f_0 + T'(f_1 + T'(f_2 + T'f_3))) + S'(g_0 + T'(g_1 + T'g_2)) \\ r_p &= r_0 + P'(a + P'b) \\ \rho_0 &= \rho_w + S(b_0 + T'(b_1 + T'(b_2 + T'(b_3 + T'b_4)))) + d_0 S^2 + S'(c_0 T'(c_1 + T'c_2)) \\ \rho &= \frac{\rho_0}{1 - \frac{P'}{r_p}} \end{aligned} \tag{1.17}$$

where now $P' = \frac{P}{100000}$, $S' = \sqrt{S^3}$ and $T' = T - 273.16$. All unexplained coefficients are defined in Appendix C.

In order to compute the potential density σ_θ the in situ density ρ is computed using the UNESCO formula, however, on the reference pressure level $REFPRES$. In order to use the correct temperature T, also the formula by Bryden is inverted using the same reference pressure level.

Whenever a heat and fresh water flux needs to be converted to a buoyancy flux derivatives of in situ density relative to temperature or salinity are required. These derivatives are computed analytically as both equations for potential temperature and in situ density are simple polynomials.

1.2 The Mixed Layer Model

An important aspect of the model implementation is that a large variety of mixed layer models can be invoked. JMO described only one possible choice. Various models, such as Kraus-Turner (1967), Niiler (1975), Garwoord (1977), Niiler and Kraus (1977), Garwood et al. (1985a,b) or Gaspar (1988), can be approximately obtained by an appropriate choice of numerous parameters.

1.2.1 Physical Background

A mixed layer (ML) is the result of turbulence produced by wind stirring and surface buoyancy fluxes. Temperature, salinity, velocities, and tracers are then uniformly distributed in the vertical (Kraus and Turner, 1967). The turbulent kinetic energy (TKE) is partly converted into mean potential energy, MPE, and partly dissipated. One of the standard problems is that typical mixed layer models are tuned to particular local conditions (e.g., ocean station Papa; see Martin, 1985). There is at present no general formulation, with a satisfying theoretical justification that is able to treat simultaneously the significantly different mixed layer regimes in different parts of the ocean. A summary of some difficulties concerning earlier ML-models is given in Gaspar (1988).

Since the mixed layer depth MLD is not only influenced by local mixing but also by horizontal convergence of mass or heat, the ML model invokes the full dynamics of equations (1.1) to (1.4) combined with a parameterization for the vertical transfer of mass and related quantities across the ML base. The ML model always has nonzero thickness and carries an arbitrary instantaneous potential density σ_θ.

1.2.2 The Mixed Layer Model Equations

Entrainment and detrainment are treated differently. While entrainment enters the continuity equation, i.e. the prognostic equation for the layer thickness, as a transfer rate between two adjacent layers, detrainment is treated diagnostically. The equation for the entrainment rate w is:

$$\begin{aligned} wh_1 max(g', g'_o) &- wRi_{crit}((u_1-u_{1+})^2+(v_1-v_{1+})^2)+m_6 = 2m_o m_1 u_*^3 \\ &+ (1-\Gamma)hm_2m_5(B-\gamma B_s)+\Gamma m_2 m_5 B_i \\ &+ (1-\Gamma)\gamma m_2 m_5 B_s[h(1+exp(-h/h_B))-2h_B(1-exp(-h/h_B))] \\ &- m_3\frac{12}{7}m_o\tau_x\Omega_y - m_4 h \end{aligned} \tag{1.18}$$

where

$$g' = g\frac{\sigma_{\theta 1+}-\sigma_{\theta 1}}{\sigma_{\theta 1}}(1-\Gamma)+g'_i\Gamma \tag{1.19}$$

$$B = \frac{g}{c_p\rho^2}(\alpha Q_1 - \beta\frac{c_p\rho}{S}(P-E+R)) \tag{1.20}$$

$$B_s = \frac{\alpha g}{c_p\rho^2}Q_{s,bot} \tag{1.21}$$

g' is the reduced gravity between the mixed layer and the next layer below, g'_o a threshold for g', Ri_{crit} the critical Richardson number, B the total buoyancy flux through the surface, Q_1 the total heat flux into the mixed layer, $P-E+R$ denotes precipitation minus evaporation plus river runoff, B_s the buoyancy flux due to the solar radiative heat flux $Q_{s,bot}$, α and β are the analytically determined expansion coefficients with respect to temperature and salinity, τ_x the

zonal wind stress, Ω_y the y-component of the earth's angular velocity, and $\Gamma = 1$ in the presence of ice otherwise $\Gamma = 0$. The variables m_1, m_2, m_3, m_4, m_5 and m_6 are tuning coefficients, which in the code are called $CMIX$ with an extension. The favorite parameter choice of JMO is $m_3 = m_4 = m_6 = 0$ and $m_5 = 1$. The parameter g'_o limits the entrainment rate to a finite value when $g' = 0$.

The entrainment Eqn. (1.18) contains numerous parameters. These allow the use of many different types of mixed layer models that have been used in literature. As an example, the simplest possible one from Kraus and Turner (1967) requires the following parameters: *CMIX0=CMIX2=CMIX3=CMIX5=CMIX7=0*, *CMIX9=1*, *CMIX1=CMIX6=1.E6*, *TURBEN=TURBREL=0*. This parameter set reduces the entrainment equation to $wg'h = 2m_o u_*^3 + hB$.

The entrainment/detrainment rate w is related to the transfer rates w_k^l in the Eqns. (1.1) to (1.4) by

$$w = \begin{cases} w_1^{+1} & \text{if } w > 0 \\ w_{+1}^1 & \text{if } w < 0 \end{cases} \tag{1.22}$$

The sea ice model described subsequently has some influence on the relation for entrainment. If sea ice is present, B_i is taken as the buoyancy flux and g'_i as reduced gravity. The buoyancy flux B_i from the atmosphere into the mixed layer is expressed as

$$B_i = (qQ_i + (1-q)Q)\frac{g(S_1 - S_i)}{c_{p,m}\rho\sigma_\theta}\frac{\partial\sigma_\theta}{\partial S}|_{\theta,p} \tag{1.23}$$

where S_i is the ice salinity, Q_i is the heat flux from the ice into the mixed layer (see Eqn. 1.94), Q_1 is the heat flux from the atmosphere into the mixed layer (see Eqn. 1.34), q_i is the sea ice concentration (see Eqn. 1.78) and $\partial\sigma_\theta/\partial S$ is the expansion coefficient with respect to salinity. This expression implies that the heat flux through the ice is associated with a fresh water flux, because ice keeps the mixed layer temperature at the freezing point. Furthermore, with $B_s = 0$ it is assumed that no solar radiation penetrates through the ice. In the presence of ice, i.e. $\Gamma = 1$, the definition of the reduced gravity g' through Eqn. (1.19) used in the entrainment Eqn. (1.18) is corrected by

$$g'_i = g(\frac{\sigma_{\theta 1+} - \sigma_{\theta 1}}{\sigma_{\theta 1}} + (\frac{\partial\sigma_\theta}{\partial\theta}|_{S,p} + \frac{c_p(S_1 - S_i)}{c_{p,m}\sigma_\theta}\frac{\partial\sigma_\theta}{\partial S}|_{\theta,p})(\theta_{1+} - \theta_1)) \tag{1.24}$$

Compared with the entrainment equation without sea ice, additional mechanisms have to be considered. Entrainment provides some heat flux into the mixed layer. Under the presence of sea ice this flux is exactly balanced by a heat flux due to melting or freezing of ice which keeps the mixed layer temperature at the freezing point. The connected fresh water flux induces a buoyancy flux that changes the amount of turbulence available for mixing.

The first term on the left hand side of Eqn. (1.18) describes the production of mean potential energy MPE, and the second describes the production of mean kinetic energy MKE by entrainment. On the right hand side, the first term represents the TKE production due to wind stirring (u_* denotes the friction velocity), and the following treats the influence of the surface buoyancy fluxes for the ice covered and ice free conditions, respectively. The last term, which is only non-zero for ice free cases, represents the influence of penetrating solar radiation on the total buoyancy flux (Denman and Miyake; 1973). The free parameter m_o represents the efficiency of how turbulence available for mixing is produced by the mean wind stress. Following Paulson and Simpson (1977), γ is the fraction of solar radiation that penetrates through the ocean surface and h_B is the depth at which the penetrating radiation has decayed to $1/e$. In

the relation for the entrainment rate the weighting coefficients a, b are defined as exponential decay functions:

$$m_1 = exp(-hf\ /(\kappa u_*)) \tag{1.25}$$

$$m_2 = \begin{cases} exp(-hf/(\kappa u_*)) & \text{if } B < 0 \\ exp(-hf/(\mu u_*)) & \text{if } B > 0 \end{cases} \tag{1.26}$$

Two different length scales are chosen, a larger scale for those terms that create deepening and a smaller one for those that reduce the MLD. This follows from the heuristic argument that wind stirring generates only $\overline{u'w'}$ terms at the surface, whereas $\overline{w'T'}$ terms are also produced as a result of positive buoyancy fluxes. They act as a source of TKE due to the generation of unstable stratification. This kind of turbulence penetrates into the ocean and becomes a source of $\overline{u'w'}$ turbulence. Thus positive buoyancy fluxes are assumed to be more efficient than wind induced turbulence or negative buoyancy fluxes.

An alternative is to specify a dissipation length scale which only depends on the buoyancy flux:

$$m_1 = exp(-h/h_{tke}) \tag{1.27}$$

$$m_2 = \begin{cases} exp(-h/h_{tke}) & \text{if } B < 0 \\ exp(-h/h_{buo}) & \text{if } B > 0 \end{cases} \tag{1.28}$$

where h_{tke} and h_{buo} are the dissipation length scales for negative and positive buoyancy fluxes.

In the retreat phase of the mixed layer the depth is determined by setting $w = 0$ in Eqn. (1.18) and solving for h. The resulting equation for the Monin-Obukhov length h_M for the case $m_3 = m_4 = m_6 = 0$ and $m_5 = 1$ is:

$$2m_o u_*^3 + h_M(B - \gamma B_s) + \gamma B_s(h_M(1 + exp(-h_M/h_B) - 2h_B(1 - exp(-h_M/h_B)) = 0 \tag{1.29}$$

The solution for h_M is determined iteratively with Newton's method. A second diagnostic calculation is carried out as soon as the flow becomes unstable due to excessive vertical shear. In this case a minimum depth h_{Ri} is defined by

$$h_{Ri} = Ri_{crit}\frac{(u_1 - u_{1+})^2 + (v_1 - v_{1+})^2}{g'} \tag{1.30}$$

As a result there are two constraints that limit the MLD. First the MLD cannot be deeper than the Monin-Obukhov-length and second it cannot be shallower that h_{Ri}. If the two constraints contradict each other, h_{Ri} is taken as criterion. In addition, the mixed layer is not allowed to be shallower than some threshold value h_{min} so that finally

$$h_1 = \max(h_{min}, h_{Ri}, \min(h_M, h_1^*)) \tag{1.31}$$

where h_1^* is the mixed layer depth computed prognostically through Eqn. (1.18). This means that as long as the MLD is thinner than h_M and thicker than h_{Ri} the MLD is not altered by these diagnostic calculations, but as soon as the MLD becomes thinner than h_{Ri} or thicker than h_M, the corresponding diagnostic quantity is taken.

The detrainment procedure is implemented so that during the predictor step a detrainment rate is computed from

$$w = \frac{h_M^{n+1} - h^n}{\Delta t} \tag{1.32}$$

and is used to force the continuity equation, while in the corrector step it is ensured that $h_1^{n+1} = h_M^{N+1}$, from which an effective entrainment/detrainment rate is determined.

1.2.3 Coupling of the Mixed Layer to the Interior Ocean

Entrainment is a process that does not conflict with the maintenance of the isopycnal coordinate system. The conceptual problem, however, is to select one of the layers below the ML into which the detrained water is pumped. The strategy is described in JMO and can be found in <MIXEXP>, <MIXIMP> and <CONVECT>. In detail, the strategy has changed in that mixed layer water with some potential density $\sigma_{\theta 1}$ is detrained into that layer such that $\sigma^*_{\theta k-\frac{1}{2}} < \sigma_{\theta_1} < \sigma^*_{\theta k+\frac{1}{2}}$ (see Eqn. 1.66 and 1.67). However, first it is tried to reduce the potential density error $\sigma_{\theta k} - \sigma^*_{\theta k}$ in the first physically existing layer below the mixed layer.

1.3 Parameterization of the Surface Fluxes

The surface fluxes required by the model are the fluxes of momentum, heat, fresh water, turbulent kinetic energy and buoyancy. The required data sets are surface air temperature, sea surface temperature, relative humidity, cloudiness, the time-averaged absolute wind speed and its standard deviation, wind stress, precipitation, air surface pressure and surface salinity. The flux of momentum is taken from Hellermann and Rosenstein (1983) or from ECMWF as global stresses. Based on the COADS (Comprehensive Ocean- Atmosphere Data Set; Woodruff et al., 1987), Wright (1988) prepared data on a $2^o \times 2^o$ grid, which is sufficiently fine for forcing a coarse-resolution OGCM. These fields represent a 30-year climatology from 1950 to 1979 and have been extended to 1986. Alternatively, the same quantities are also made available from the ECMWF Reanalysis Project (Gibson et al., 1997), which are available on a T106 Gaussian grid.

1.3.1 Parameterization of the Surface Heat Fluxes

The total heat flux into the mixed layer Q_1 is given by

$$Q_1 = Q_H + Q_L + Q_{l,bot} + Q_{s,bot}(1 - \gamma exp(-h_1/h_B)) \tag{1.33}$$

and consists of the sensible heat flux Q_H, the latent heat flux Q_L, the net heat flux by longwave radiation $Q_{l,bot}$ and the heat flux $Q_{s,bot}$ due to insolation. γ defines the fraction of the insolation that is not immediately absorbed at the surface but penetrates into the ocean. Following Paulson and Simpson (1977) h_B is the decay length scale for the absorption of solar radiation. This means that when all upper layers are shallow enough, deeper layers gain heat due to insolation. The heating rates of all deeper layers are defined by

$$Q_k = Q_{s,bot}(exp(-\sum_{l=1}^{k-1} h_l/h_B) - exp(-\sum_{l=1}^{k} h_l/h_B))\gamma \tag{1.34}$$

The turbulent surface heat fluxes, namely the sensible and latent heat fluxes, are estimated by the bulk formulae

$$Q_H = \rho_a c_{p,air} c_H V (T_a - T_s) \tag{1.35}$$
$$Q_L = \rho_a L_w c_L V (q_a - q_s) \tag{1.36}$$

where T_a is the air temperature, T_s the sea surface temperature, q_a the air specific humidity, q_s the specific humidity close to the surface, which is assumed to be the saturated value, ρ_a is the surface air density and V is the scalar wind (see also Eqn. 1.64). $c_{p,air}$ is the specific heat of air and L_w the latent heat of evaporation. The bulk coefficients c_H and c_L are calculated as proposed by Large and Pond (1982). The specific humidity q is given by the water vapor

pressure e and the atmospheric surface pressure p. In the following equations (1.37) - (1.44), e and p are given in Pascal and T is in K.

$$e = 611 \times 10^{7.5(T-273.16)/(T-35.86)} \tag{1.37}$$

$$q_s = \frac{0.622e}{p-0.378e} \qquad and \qquad q = \frac{0.622re}{p-0.378re} \tag{1.38}$$

where r is the relative humidity. The respective formulae are given in the next section.

Radiative fluxes have been rewritten in order to achieve a realistic climate sensitivity. This work has been done as response to greenhouse experiments with the atmospheric energy balance model (EBM, see later). Basically, it was found that factors of the type $(1-an)$ where a is some coefficient and n is a function of cloudiness and humidity, yields to unrealistic fluxes if humidity is higher than in the present climate. However, if the former factor is considered as Taylor expansion of $\frac{1}{1+bn}$ then a reversal of the factor's sign is avoided. Then, a super-greenhouse effect is not simulated. In the following, all radiative fluxes have been rewritten.

The net longwave radiation at the surface is parameterized as

$$Q_{l,bot} = -\varepsilon\sigma(T_s^4 - T_{eff}^4) \tag{1.39}$$

$$T_{eff} = T_a + \frac{0.4}{1.0+0.0021*ren}(223.15 - T_a) \tag{1.40}$$

where ε is the emissivity of water, σ the Stefan-Boltzmann constant and n the relative cloud cover.

The insolation is calculated from the daily averaged heat flux at the top of the atmosphere and is then corrected after Zillmann (1972) for relative humidity and inclination. The insolation is reduced by a combined cloudiness and humidity correction. The resulting relations to compute the daily mean downward shortwave radiative flux $Q_{s,bot}$ are

$$\cos\eta = \sin\delta\sin\varphi + \cos\delta\cos\varphi\cos t \tag{1.41}$$

$$\sin\eta_{noon} = \sin\delta\sin\varphi + \cos\delta\cos\varphi \tag{1.42}$$

$$\kappa_b = \frac{1.0+0.0019\eta_{noon}}{1.0+0.0019\eta_{equa}}\frac{1}{1.0+0.79(1.0+(0.0004re)^2)n^2} \tag{1.43}$$

$$Q_{s,bot} = \frac{\kappa_b(1-\omega)}{2\pi}\int_{t_1}^{t_2}\frac{S_o\cos^2\eta}{((\cos\eta+2.7)r\frac{e}{p}+1.085\cos\eta+0.1)}(\frac{\bar{d}}{d})^2 dt \tag{1.44}$$

where S_o is the solar constant, η the solar elevation, $\eta_{equa} = 90$ deg and ω the albedo. d denotes the distance between the sun and earth and $\bar{d}$ its annual average. Following Paltridge and Platt (1976), the ratio $(\bar{d}/d)^2$ is estimated in terms of the Julian Day β for the present day orbit:

$$\begin{aligned}(\frac{\bar{d}}{d})^2 = 1.00011 &+ 0.00128\ \sin(\beta) + 0.034221\ \cos(\beta) \\ &+ 0.000077\ \sin(2\beta) + 0.000719\ \cos(2\beta)\end{aligned} \tag{1.45}$$

The declination δ (in radians), which is needed to compute η, is given by

$$\begin{aligned}\delta = 0.006918 &+ 0.070257\ \sin(\beta) - 0.399912\ \cos(\beta) \\ &+ 0.000907\ \sin(2\beta) - 0.006758\ \cos(2\beta) \\ &+ 0.00148\ \sin(3\beta) - 0.002697\ \cos(3\beta)\end{aligned} \tag{1.46}$$

Variations in the distance between sun and earth account for slightly more than 3 % of the variations in the net global solar radiation.

Eqn. (1.43) for κ_b is used in connection with the COADS cloudiness. However, because the statistics of the ECMWF reanalysis cloudiness differs significantly from that of the COADS data, an alternative parameterization for κ_b is needed if cloudiness is taken from the ECMWF reanalysis project. The relation was found by tuning and is

$$\kappa_b = \frac{1.0 + 0.0019\eta_{noon}}{1.0 + 0.0019\eta_{equa}} \frac{1}{1.0 + 2.8(1.0 + (0.0001re)^2)n^4} \tag{1.47}$$

The net longwave radiation at the surface stays independent on the use of COADS or ECMWF reanalysis cloudiness.

1.3.2 Transfer Coefficients

The bulk coefficients were taken from Large and Pond (1981,1982):

$$c_H = \frac{c_{HN}\sqrt{\frac{c_M}{c_{MN}}}}{1 - \frac{c_{HN}}{\kappa\sqrt{c_{MN}}}\psi_H} \tag{1.48}$$

$$c_L = \frac{c_{LN}\sqrt{\frac{c_M}{c_{MN}}}}{1 - \frac{c_{LN}}{\kappa\sqrt{c_{MN}}}\psi_L} \tag{1.49}$$

$$\sqrt{\frac{c_M}{c_{MN}}} = \frac{1}{1 - \frac{\sqrt{c_{MN}}}{\kappa}\psi_M} \tag{1.50}$$

$$c_{MN} = \frac{\kappa^2}{ln^2(\frac{Z}{Z_o})} \tag{1.51}$$

$$c_{HN} = 0.0327\frac{\kappa}{ln(\frac{Z}{Z_o})} \tag{1.52}$$

$$c_{LN} = 0.0346\frac{\kappa}{ln(\frac{Z}{Z_o})} \tag{1.53}$$

$$Z_o = c_{char}\frac{u_*^2}{g} \tag{1.54}$$

$$u_*^2 = c_M u^2 \tag{1.55}$$

$$T_o = T(1 + 1.7 \times 10^{-6} Tq) \tag{1.56}$$

Here c_M, c_H and c_L are the transfer coefficients for momentum, sensible and latent heat, respectively. The subscript N denotes the transfer coefficient for neutral conditions. For stable conditions it is used:

$$\psi_M = \psi_H = \psi_L = -7(\frac{Z}{L}) \tag{1.57}$$

$$(\frac{Z}{L}) = -\frac{70Z}{u^2 T_o}(\Delta\theta + 2.5 \times 10^{-6} T_o^2 \Delta q) \tag{1.58}$$

while for unstable conditions it is used:

$$\psi_M = 2ln[(1+X)/2] + ln[(1+X^2)/2] - 2arctanX + \pi/2 \tag{1.59}$$

$$\psi_H = \psi_L = 2ln[(1+X^2)/2] \tag{1.60}$$

$$X = (1 - 16(\frac{Z}{L}))^{1/4} \tag{1.61}$$

$$(\frac{Z}{L}) = -\frac{100Z}{u^2 T_o}(\Delta\theta + 1.7 \times 10^{-6} T_o^2 \Delta q) \tag{1.62}$$

where Δq is the difference between the specific humidity of air and the sea surface, and $\Delta\theta$ is the potential temperature difference.

The only change from Large and Pond's work is that c_{MN} is not fitted against data using *ad hoc* chosen curves, but by tuning the Charnock constant. The equations (1.51), (1.54) and (1.55) describe the dependence of the neutral drag coefficient c_{MN} on the friction velocity u_*, the Karman constant κ, the height of the measurement Z and Charnock's constant c_{char}. In order to obtain a drag coefficient of about 1.15×10^{-3} for neutral conditions at 10 m/s, c_{char} should be set to 0.0064. However, as outlined by Oberhuber (1988), c_{char} in fact is set to 0.032 to compensate for the underestimation of the transfer coefficient resulting from the application of monthly mean values instead of instantaneous values. Note that the standard forcing of PIPE is based on monthly mean winds.

1.3.3 Evaluation of the Net Fresh Water Flux

A Newtonian relaxation is used to force the salinity equations with fresh water fluxes (see also JMO). The simplest way is to use relaxation towards the observed salinity. Optionally, data for precipitation, evaporation, which are computed from latent heat, and river runoff may be used, too. The relation is

$$F_1 = \rho_1 \delta \frac{S_{obs} - S_1}{S_1} + (P - E + R) \qquad with \qquad E = \frac{Q_L}{\rho_w c_{p,m}} \tag{1.63}$$

where δ is the time constant with which the actual salinity relaxes towards the observed salinity S_{obs}, P is the precipitation, E is the evaporation deduced from the latent heat flux Q_L, $c_{p,m}$ the heat of fusion, ρ_w the density of fresh water and R the river runoff. The precipitation data are taken from Legates and Willmott (1990).

1.3.4 Estimate of Turbulent Kinetic Energy Input

The monthly mean absolute wind $\overline{V}$ and its standard deviation $\sigma(V)$ are available from the COADS by Wright (1988) or from the ECMWF analyses. The standard deviation is required to accurately evaluate the time-averaged third power of the friction velocity u_*, which occurs in the relation for the entrainment rate. Since $\overline{u_*}^3$ determined from the monthly mean absolute wind $\overline{V}$ only is much smaller than the required $\overline{u_*^3}$, an effective u_*^3 must be determined by the additional use of the monthly mean standard deviation of the absolute wind. By assuming that the amplitude of the fluctuations is not too large compared with the mean wind, the effective u_*^3 can be approximated by assuming a sinusoidal fluctuation around the mean value (see JMO):

$$u_*^3 = \sqrt{\frac{c_d \rho_a}{\rho}}^3 \overline{V}(\overline{V}^2 + 3\sigma^2(V)) \tag{1.64}$$

A similar relation is required to compute u_*^2 for the neutral drag coefficient:

$$u_*^2 = \sqrt{\frac{c_d \rho_a}{\rho}}^2 \overline{V}(\overline{V} + 2\sigma(V)) \tag{1.65}$$

1.4 Parameterizations of Internal Diffusion

A parameterization for vertical diffusion in the interior ocean is introduced by allowing mass to be transferred between neighbouring layers. An explicit procedure for mixing by convection is also included. A first discussion of how the vertical mixing processes are connected with the

problem of maintaining the coordinates in an isopycnal OGCM was given by JMO, however, the strategy has changed slightly in that the potential density in each layer is allowed to vary between the lower threshold $\sigma^*_{\theta k-\frac{1}{2}}$ and the upper threshold $\sigma^*_{\theta k+\frac{1}{2}}$

$$\sigma^*_{\theta k-\frac{1}{2}} = \frac{\sigma^*_{\theta k-1}+\sigma^*_{\theta k}}{2} \tag{1.66}$$

$$\sigma^*_{\theta k+\frac{1}{2}} = \frac{\sigma^*_{\theta k+1}+\sigma^*_{\theta k}}{2} \tag{1.67}$$

As long as the potential density $\sigma_{\theta k}$ varies between these threshold values, only vertical mixing tries to attract the coordinate system towards $\sigma^*_{\theta k}$. If the potential density $\sigma_{\theta k}$ exceeds one of these thresholds, then the convection procedure assigns water of that particular layer to a neigbouring layer such that the newly organized layer fullfills the limits. Basically, a layer is allowed to have a potential density error in a range between $\sigma_{\theta k}-\sigma^*_{\theta k-\frac{1}{2}}$ and $\sigma_{\theta k}-\sigma^*_{\theta k+\frac{1}{2}}$. However, vertical diffusion is efficient enough to keep the error nearby zero, unless a layer is very thin so that the assumption of linearity of the mixing process leads to some potential density error.

1.4.1 Vertical Mixing / Coordinate Maintenance

Following the mixed layer parameterization, it is assumed that turbulent kinetic energy is converted into mean potential energy via some u^3_* term. If water is entrained only from above or only from below, the equations for the entrainment rates w^{k-*}_k and w^{k+*}_k are

$$w^{k-*}_k = \frac{2m_o u^3_{*V}}{g'_{k-}h_k - Ri_{crit}((u_k-u_{k-})^2+(v_k-v_{k-})^2)}(1+W) \tag{1.68}$$

$$w^{k+*}_k = \frac{2m_o u^3_{*V}}{g'_{k+}h_k - Ri_{crit}((u_k-u_{k+})^2+(v_k-v_{k+})^2)}(1+W) \tag{1.69}$$

$$W = \frac{g'_{k-}(h_k+h_{k-})+g'_{k+}(h_k+h_{k+})}{h_w(g'_{k-}+g'_{k+})} \tag{1.70}$$

where g'_{k-} and g'_{k+} are the stabilities across the upper and lower interfaces and h_w is a tuning coefficient. Compared with JMO, the weight W has been introduced to make the vertical diffusivity less dependent on the stability. The expression for W enhances vertical mixing in the deep ocean while mixing is unchanged in the upper ocean. The available turbulent energy $2m_0u^3_{*V}$ used for internal mixing is defined as

$$u^3_{*V} = u^3_{*V_0}(1-\frac{3q_i}{4}) + u^3_*\frac{1-q_i}{1+\frac{\sigma_{\theta k}-\sigma_{\theta 1}}{2}} \tag{1.71}$$

where $u^3_{*V_0}$ is a constant background energy, u^3_* is the turbulent energy from Eqn. (1.64) and q_i the sea ice concentration. This parameterization for TKE available for internal mixing is made dependent on the ice concentration in order to express that TKE cannot be produced if ice decouples the ocean from the energy source which is the wind. The factor $\frac{3}{4}$ in Eqn. (1.71) parameterizes the effect that an internal energy source also might exist. The 2nd term in Eqn. (1.71) parameterizes the effect that wind available from data and used by the mixed layer model also causes TKE below the mixed layer through braking internal waves. This energy available below the mixed layer partly reduces the effective entrainment rate into the mixed layer. This parameterization was added in order to avoid unrealistic deep mixed layers in the subtropics, i.e. under conditions of strong wind and strong stratification.

The problem of vertical mixing is underdetermined, since one is free to choose the amount of water to be entrained from above or from below. If a free parameter α for this unknown ratio is introduced the equation for the total entrainment rate w_k is given by

$$w_k = (1-\alpha)w_k^{k-*} + \alpha w_k^{k+*} \tag{1.72}$$

with individual entrainment rates

$$w_k^{k-} = (1-\alpha)w_k^{k-*} \tag{1.73}$$
$$w_k^{k+} = \alpha w_k^{k+*} \tag{1.74}$$

After substituting these entrainment rates into the momentum, mass, heat, and salt equations, the free parameter α can now be chosen to maintain the potential density within some layer at or nearby a prescribed value σ_θ^*, while at the same time compensating some potential density drift due to discretization errors in the advection and diffusion formulation. In order to balance these errors, α must be made space and time-dependent. The relevant balance equation is given by the approximate equation for the potential density at the new time level:

$$(h_k + \Delta t(w_k^{k-} + w_k^{k+}))\sigma_{\theta k}^* = h_k\sigma_{\theta k} + \Delta t w_k^{k-}\sigma_{\theta k-} + \Delta w_k^{k+}\sigma_{\theta k+} \tag{1.75}$$

where the potential density at the new time level should be equal the prescribed value $\sigma_{\theta k}^*$. This equation was derived under the assumption that the potential density of a mixed water mass is the layer thickness weighted average of the potential densities of the unmixed water masses. This yields the final equation for α:

$$\alpha = \frac{h_k(\sigma_{\theta k}^* - \sigma_{\theta k}) + \Delta t w_k^{k-*}(\sigma_{\theta k}^* - \sigma_{\theta k-})}{\Delta t w_k^{k-*}(\sigma_{\theta k}^* - \sigma_{\theta k-}) - \Delta t w_k^{k+*}(\sigma_{\theta k}^* - \sigma_{\theta k+})} \tag{1.76}$$

1.4.2 Convection

If the stratification becomes unstable it is removed by vertical mixing. All quantities are set to their vertical average over the ML and the underlying layer UL. The procedure how the coordinate system is reorganized when an isopycnal layer becomes too heavy is described in JMO. A well-known problem is the determination of the stability, when potential densities are defined with respect to some reference level. If the reference pressure is too low, the model might analyse a weakly stable stratification, while a potential density defined with respect to the considered depth yields an unstable stratification. To be consistent in the model, only potential densities with respect to one common pressure level are used, e.g. for horizontal pressure gradients as well as for convection and vertical mixing.

In the isopycnal part of the model, layers are initially stably stratified. If in rare cases unstable stratification arises through a combination of vertical mixing and a combined effect of the nonlinearity of the equation of state and temperature/salinity transport, then the concerning water mass is assigned to one of the neighbouring layers such that the isopycnal coordinate system becomes valid again.

1.5 The Snow - Sea Ice Model

Sea ice is an important boundary condition for the ocean at high latitudes. The seasonal cycle of ice thickness and ice extent influences the heat budget at the ocean surface and the internal stratification. During cold periods, freezing ice ejects salt into the mixed layer and thereby contributes to the production of heavy water. During warm periods, melting ice decreases the

salinity in the mixed layer and therefore contributes to a stabilization of the upper ocean. Basically, sea ice has the task to convert an atmospheric heat flux into an oceanic fresh water flux. While some heat flux results in a vanashing buoyancy flux due to the low thermal expansion of sea water around the freezing point, a fresh water flux induces a high buoyancy flux as result of brine rejection. Snow cover modifies the heat fluxes that occur in the presence of an ice cover. This is because snow has both a lower conductivity and a higher albedo than ice. Furthermore, the albedo depends on the snow type and for a thin snow cover, also on its thickness.

1.5.1 The Dynamic Equations

Hibler (1979) proposed a rheology for a dynamical sea ice model that can be used for a wide range of space and time scales. For a number of technical reasons such as the introduction of spherical coordinates and of the momentum and mass conserving flux form of the equations, (which permits an easier treatment of the ice edge) the model has been completely rewritten from scratch, but based on the same physics as in Hibler's model. A snow model has been added, based on a continuity equation for snow and a parameterization for the aging process of snow. The heat capacity of snow and ice is included via two prognostic temperatures for the skin temperature over snow and ice.

The basic equations for the cell averages of the ice flux $(\vec{vh})_i$, of the ice thickness h_i, of the ice concentration q_i and of the snow depth s_i are (with the latter taken as equivalent ice thickness)

$$\frac{\partial}{\partial t}(\vec{vh})_i = \vec{\nabla}\cdot A_i\vec{\nabla}(\vec{vh})_i - \vec{f}\times(\vec{vh})_i - gh_i\vec{\nabla}(\sum_{k=1}^{N} h_k + D) + \frac{\vec{\tau}_a}{\rho_i} + \frac{\vec{\tau}_o}{\rho_i} + \vec{F}_v \tag{1.77}$$

$$\frac{\partial}{\partial t}h_i = \vec{\nabla}\cdot A_i\vec{\nabla}h_i - \vec{\nabla}\cdot(\vec{vh})_i + F_h + F_a \tag{1.78}$$

$$\frac{\partial}{\partial t}q_i = \vec{\nabla}\cdot A_i\vec{\nabla}q_i - \vec{\nabla}\cdot(\vec{vq})_i + F_q \tag{1.79}$$

$$\frac{\partial}{\partial t}s_i = \vec{\nabla}\cdot A_i\vec{\nabla}s_i - \vec{\nabla}\cdot(\vec{vs})_i + F_s - F_a \tag{1.80}$$

where $\vec{\tau}_a$ and $\vec{\tau}_o$ are the surface wind stress and the stress at the bottom of the ice and $\sum_{k=1}^{N}(h_k + D)$ is the sea surface elevation. In these equations the transport of momentum has been neglected. A_i is a constant diffusion coefficient. $\vec{f}$ is the Coriolis parameter. $\vec{F}_v$, F_h, F_q, F_s and F_a are the forcing functions for the ice momentum, ice mass, ice concentration, snow mass and the aging process, respectively. In more detail, $\vec{F}_v$ represents the internal ice stress, determined by a viscous-plastic ice rheology, F_h the ice thickness change due to freezing or melting, F_q the change of the ice concentration due to external heat fluxes, F_s the change of the snow mass due to snowfall or melting, and F_a the conversion rate from snow to ice that describes the aging process of snow. The sea ice rheology reads:

$$F_{vx} = \frac{\partial}{\partial x}[(\eta+\zeta)\frac{\partial u_i}{\partial x} + (\zeta-\eta)\frac{\partial v_i}{\partial y} - \frac{P_0}{2}] + \frac{\partial}{\partial y}[\eta(\frac{\partial u_i}{\partial y} + \frac{\partial v_i}{\partial x})] \tag{1.81}$$

$$F_{vy} = \frac{\partial}{\partial y}[(\eta+\zeta)\frac{\partial v_i}{\partial y} + (\zeta-\eta)\frac{\partial u_i}{\partial x} - \frac{P_0}{2}] + \frac{\partial}{\partial x}[\eta(\frac{\partial u_i}{\partial y} + \frac{\partial v_i}{\partial x})] \tag{1.82}$$

$$\Delta = ((\frac{\partial u_i}{\partial x})^2 + (\frac{\partial v_i}{\partial y})^2)\frac{1+e^2}{e^2} + (\frac{\partial u_i}{\partial y} + \frac{\partial v_i}{\partial x})^2\frac{1}{e^2} + 2\frac{\partial u_i}{\partial x}\frac{\partial v_i}{\partial y}\frac{e^2-1}{e^2} \tag{1.83}$$

$$P_0 = \frac{P_i\ h_i}{\rho_i} exp[-\epsilon_1(1-q_i)] \tag{1.84}$$

$$\zeta = \frac{P_0}{max[2\sqrt{\Delta}, \epsilon_o]} \quad (1.85)$$

$$\eta = \frac{\zeta}{e^2} \quad (1.86)$$

P_i is a proportionality coefficient, e is the excentricity of the viscous-plastic rheology, ε_0 is a threshold for the computed bulk viscosity, ε_1 is a decay parameter and ρ_i is a constant ice density. Parts of the operators on the right hand side can be identified as a diffusion operator, with two diffusion coefficients, the bulk viscosity ζ and the shear viscosity η, which are highly dependent on the flow field. However, additional terms occur that cross-couple the velocity components.

1.5.2 The Thermodynamic Equations

The model is forced by atmospheric heat fluxes as well as fresh water fluxes given by precipitation and evaporation. Two prognostic variables for the heat content of snow and ice are introduced, the temperature between snow and ice T_h, and the temperature between snow and the atmosphere T_s. The internal temperature profile within the ice and the snow is idealized to be linear. The underlying approach is to convert the difference of the heat flux through the ice and snow layer into an accumulation of the heat content of both the snow and ice layers. Furthermore, the heat flux at the snow surface is set equal to the heat flux through the snow layer in the case that no heat is used to melt snow. The system of equations that results is as follows:

$$\frac{k_i q_i}{h_i}(T_h - T_1) + \frac{k_s q_i \rho_s}{s_i \rho_i}(T_h - T_s) + c_{p,i}[\frac{\rho_i h_i}{2}\frac{\partial}{\partial t}T_h + \frac{\rho_i s_i}{2}\frac{\partial}{\partial t}(T_h + T_s)] = 0 \quad (1.87)$$

$$\frac{k_s q_i \rho_s}{s_i \rho_i}(T_h - T_s) + Q(T_s) = 0 \quad (1.88)$$

Here, T_1 is the mixed layer temperature, k_s the conductivity of snow, k_i the conductivity of ice and $Q(T_s)$ the heat flux through the snow or at the ice surface, $c_{p,i}$ the specific heat capacity of ice and ρ_s is the density of snow, and h_i/q_i is the effective thickness of an ice floe and $(\rho_i s_i)/(\rho_s q_i)$ is the effective snow depth. Note that s_i is a cell-averaged snow depth at the ice density. The equation for the heat accumulation (1.87) contains two time derivatives. The first involves the ice surface temperature T_h assuming that the time derivative of the sea surface temperature T_1 is negligible. The second time derivative term measures the vertically averaged heat content change of the snow layer, where T_s and T_h are used as surface and bottom temperature of the snow layer. In order to treat the special cases of no snow and no ice, the equations are rewritten to give $T_h = T_s$ for no snow and to give $T_h = T_s = T_1$ for no snow and ice. For that (1.88) is used to modify (1.87) and the resulting equations are multiplied by the ice or snow mass. This finally yields:

$$k_i q_i(T_h - T_1) - h_i Q(T_s) + h_i c_{p,i}[\frac{\rho_i h_i}{2}\frac{\partial}{\partial t}T_h + \frac{\rho_i s_i}{2}\frac{\partial}{\partial t}(T_h + T_s)] = 0 \quad (1.89)$$

$$\frac{k_s q_i \rho_s}{\rho_i}(T_h - T_s) + s_i Q(T_s) = 0 \quad (1.90)$$

In the case that T_s exceeds the melting temperature $T_m = 273.16$ of ice and snow, $T_s = T_m$ is used in (1.87) to compute the ice surface temperature T_h. The cell-averaged skin temperature T_a is defined by

$$T_a = T_s q_i + T_1(1 - q_i) \quad (1.91)$$

This yields the snow surface temperature T_s or the ice surface temperature T_h by simplifying the total heat flux Q from (1.33) through a Taylor expansion around T_a:

$$Q(T) = Q(T_a) + \frac{\partial Q}{\partial T}(T - T_a) \tag{1.92}$$

This equation is used to formulate the two equations for T_s and T_h as two linear equations that can be solved easily. Since the skin temperatures influence the transfer coefficients via the stability-dependent Large and Pond coefficients, the final solution for the skin temperature is obtained by iteration. The resulting T_h and T_s are used to compute the heat flux Q_s that is converted into melting snow and the heat flux Q_i that is converted into melting or freezing sea ice:

$$Q_s = Q(T_m) + \frac{k_s q_i \rho_s}{s_i \rho_i}(T_h - T_m) \tag{1.93}$$

$$Q_i = (1 - q_i)Q(T_1) + q_i \frac{k_i q_i}{h_i}(T_h - T_1) \tag{1.94}$$

Thus, the heat flux Q_s used for melting snow is the difference between the atmospheric heat flux and the conductive heat flux through the snow, while the heat flux Q_i used for melting or freezing sea ice is the total heat flux $Q(T_1)$ into the ice free ocean plus the conducted heat flux through the ice floe. Note that if $T_s < T_m$ then $Q_s = 0$. This is guaranteed by (1.88), however, this equation is not used when (1.89) and (1.90) yield $T_s > T_m$. In this case the error in (1.88) is interpreted as heat flux used for melting snow. The linearity of (1.87) and (1.88) ensures that this flux is always downward. The resulting change in the local ice thickness F_h due to thermodynamics is then given by the heat flux Q_i and the heat flux induced by the entrainment rate w:

$$F_h = \frac{|w| + w}{2}\frac{c_p}{c_{p,m}}(T_1 - T_{1+}) - \frac{Q_i}{\rho_i c_{p,m}} \tag{1.95}$$

where $c_{p,m}$ is the latent heat of fusion.

The thermodynamic part of the ice concentration equation differs slightly from Hibler's formulation. The change of the ice concentration F_q is evaluated through

$$F_q = \begin{cases} F_h(1 - q_i)/h_0 & \text{if } F_h > 0 \\ F_h q_i / 2h_i & \text{if } F_h < 0 \end{cases} \tag{1.96}$$

where h_0 is interpreted as ice thickness that immediately builds up within leads during freezing conditions. The aging process of snow is expressed as a rate F_a at which snow is converted into ice. The first term parameterizes the metamorphosis of snow cristalls to ice by assuming a simple snow depth and time scale dependent γ. The second term adjusts the snow depth when snow suppresses the ice surface below the sea level. Thus, this mechanism represents an upper threshold for the snow depth at a given ice thickness. The change in snow thickness F_a is

$$F_a = \gamma s_i + \frac{1}{\Delta t}\frac{\rho_s}{\rho_i} max(0, \frac{h_s \rho_s - h_i(\rho_1 - \rho_i)}{\rho_1}) \tag{1.97}$$

where γ reflects mean values for the conversion rate. The forcing of the snow depth equation F_s depends on precipitation minus evaporation, melting and loss of snow mass due to reduction of the ice concentration:

$$F_s = q_i R_1^h + \frac{q_i Q_s}{c_{p,m}} + \frac{s_i}{q_i} min[F_q, 0] \tag{1.98}$$

Following Millero (1978), the equation used for the salinity dependent freezing point of sea water is

$$T_f = T_m - 0.0575\ S + 0.001710523\ S^{3/2} - 0.0002154996\ S^2 \tag{1.99}$$

which is used to set a threshold to the SST.

1.5.3 Coupling Sea Ice with Ocean Salinity

During freezing, salt is ejected from the ice but is not confined to the ML alone. It is assumed that a fraction of this ejected salt penetrates into the subsurface ocean with a length scale that depends on the stratification in the surface layers. Therefore, a higher stability should give a shorter length scale. The following *ad hoc* parameterization for salt transfer out of the mixed layer into all deeper layers ($k = 2, ..., N$) is chosen as

$$R_k^S = max[F_h, 0]\rho_1(S_1 - S_i)(exp[-\frac{\sigma_{\theta k} - \sigma_{\theta 1}}{h_p}\sum_{l=1}^{k-1} h_l] - exp[-\frac{\sigma_{\theta k+} - \sigma_{\theta 1}}{h_p}\sum_{l=1}^{k} h_l]) \qquad (1.100)$$

where $h_p = 20kgm^{-2}$ is a free parameter tuned to obtain a reasonable model response. The salinity budget in the ML is the result of the salinity gain due to freezing ice and the salinity loss due to the downward transfer of a fraction of the salinity gain. The salinity change in the ML is given by

$$R_1^S = \rho_1(S_1 - S_i)(F_h - \frac{|F_h| + F_h}{2}exp[-\frac{\sigma_{\theta 1+} - \sigma_{\theta 1}}{h_p}h_1]) \qquad (1.101)$$

Thus water formed from melting ice ($F_h < 0$) is completely mixed within the ML since $R_k^S = 0$ for $k = 2, ..., N$, but freezing ice injects a fraction of the salt into deeper layers. This allows the model to build up a salinity stratification in the Arctic basin although in the annual mean there is a net transport of salt from the sea ice through the ML into the deeper ocean. The balance between downward transport and upward diffusion yields an equilibrium state for the salinity stratification in areas covered by sea ice.

1.6 Atmospheric Energy Balance Model

The atmospheric energy balance model represents a cheap replacement of todays full-blown atmospheric GCMs. The neglect of dynamical feedbacks between pressure and flow fields as well as the resignment on detailed vertical resolution yields a one-layer atmospheric model that predicts the horizontal distribution of temperature and humidity only. In order to keep such a model sufficiently realistic the EBM is data-driven by prescribing wind and cloudiness. All other fields required to close the global energy budget as due to radiation, precipitation, sensible and latent heat exchange between ocean and atmosphere, and diabatic and adiabatic heating are diagnosed. In order to conserve energy soil and river runoff models predict temperature, water content in the soil, and water flow towards the oceans. At high latitudes the variables of the sea ice model are used to simulate glaciers with snow on top.

So far various approaches have been realized to obtain a simple atmosphere model without dynamics but with more realistic feedbacks than possible with a Newtonian relaxation (North et al., 1983; Kleeman and Power, 1995), which also don't contain the surprising sensitivity of mixed boundary conditions to arbitrary parameter choices as discussed by Mikolajewicz and Maier-Reimer (1994).

The goal is to develop an EBM that has a closed hydrological cycle and conserves the sum of sensible and latent heat. Some of the parameterizations thus are constructed in such a way that they automatically conserve heat and water independent on a proper parameter tuning.

1.6.1 Transport Equations for Temperature and Humidity

The EBM's vertically integrated temperature T_{ebm} and water vapour contents V_{ebm} are predicted through the following equations:

$$\frac{\partial T_{ebm}}{\partial t} = -\epsilon_1 \vec{v} \cdot \vec{\nabla} T_{ebm} + \vec{\nabla} \cdot (A_{ebm} \vec{\nabla} T_{ebm}) + \frac{Q_{ebm}}{c_{p,a} \rho_a H_T} \tag{1.102}$$

$$\frac{\partial V_{ebm}}{\partial t} = -\epsilon_2 \vec{\nabla} \cdot V_{ebm} + \vec{\nabla} \cdot (A_{ebm} \vec{\nabla} V_{ebm}) + E - P \tag{1.103}$$

where Q_{ebm} is the sum of all the heat fluxes at the top and the bottom of the atmosphere, and the internal heat fluxes:

$$Q_{ebm} = Q_{s,top} + Q_{l,top} - Q_{s,bot} - Q_{l,bot} - Q_H + Q_{atm} \tag{1.104}$$

where $Q_{s,top}$ is the top solar radiation (see Eqn. 1.111), $Q_{l,top}$ the top longwave radiation (see Eqn. 1.109), $Q_{s,bot}$ the solar radiation at the surface (see Eqn. 1.44), $Q_{l,bot}$ the longwave radiation at the surface (see Eqn. 1.39), Q_H the sensible heat flux into the ocean or soil (see Eqn. 1.35), Q_{atm} is the sum of latent heat release due to condensation/precipitation and adiabatic cooling due to vertical motion under the presence of some assumed atmospheric vertical stratification. E and P are the evaporation and precipitation at the earth's surface, respectively. A_{ebm} is a turbulent horizontal eddy diffusion coefficient. H_T is the height of the atmosphere.

1.6.2 Diagnostic Relations

The atmospheric eddy diffusion coefficient A_{ebm} is parameterized as:

$$A_{ebm} = \epsilon_3 \overline{V} \sigma(V) \tag{1.105}$$

with $\overline{V}$ the observed scalar wind and $\sigma(V)$ its standard deviation. This equation can be compared with Kolmogorov's idea to relate a diffusion coefficient with eddy length scale and turbulent energy. Here we may identify the mean speed $\overline{V}$ with the eddy length scale and the wind's standard deviation $\sigma(V)$ with the turbulent energy.

The EBM's internal forcing is the result of latent heat release due to condensation, which is assumed to fall out immediately as rain. Like in reality the heat release is compensated by cooling due to upward motion which is an unresolved dynamical effect of heating. The internal forcing is parameterized as:

$$Q_{atm} = \epsilon_4 L_w (P + \epsilon_5 \vec{\nabla} \cdot \vec{v}) \tag{1.106}$$

In order to parameterize the precipitation P the idea is to consider it as sum of convective and large-scale precipitation. While the first one is typical for low latitudes where near-surface convergence of moisture is driving cloud formation, the second one is asssumed to be created by large-scale variability due to low-pressure systems. Therefore, it is the concept to parameterize rain with the help of wind convergence for low latitudes and standard deviation of the wind for higher latitudes. The relation is:

$$P = V_{ebm}(\epsilon_6 \sigma(V) - \epsilon_7 min[0, \vec{\nabla} \cdot \vec{v}]) max[0, r - 0.6] \tag{1.107}$$

Assuming a typical thickness of the humid boundary layer H_v the relative humidity r is computed from:

$$r = \frac{\rho_w V}{\rho_a q H_v} \tag{1.108}$$

where q is the specific humidity (see Eqn. 1.38). The evaporation E is computed from the latent heat flux (see Eqn. 1.36). An example for a simulated precipitation is shown in Fig. 1.2.

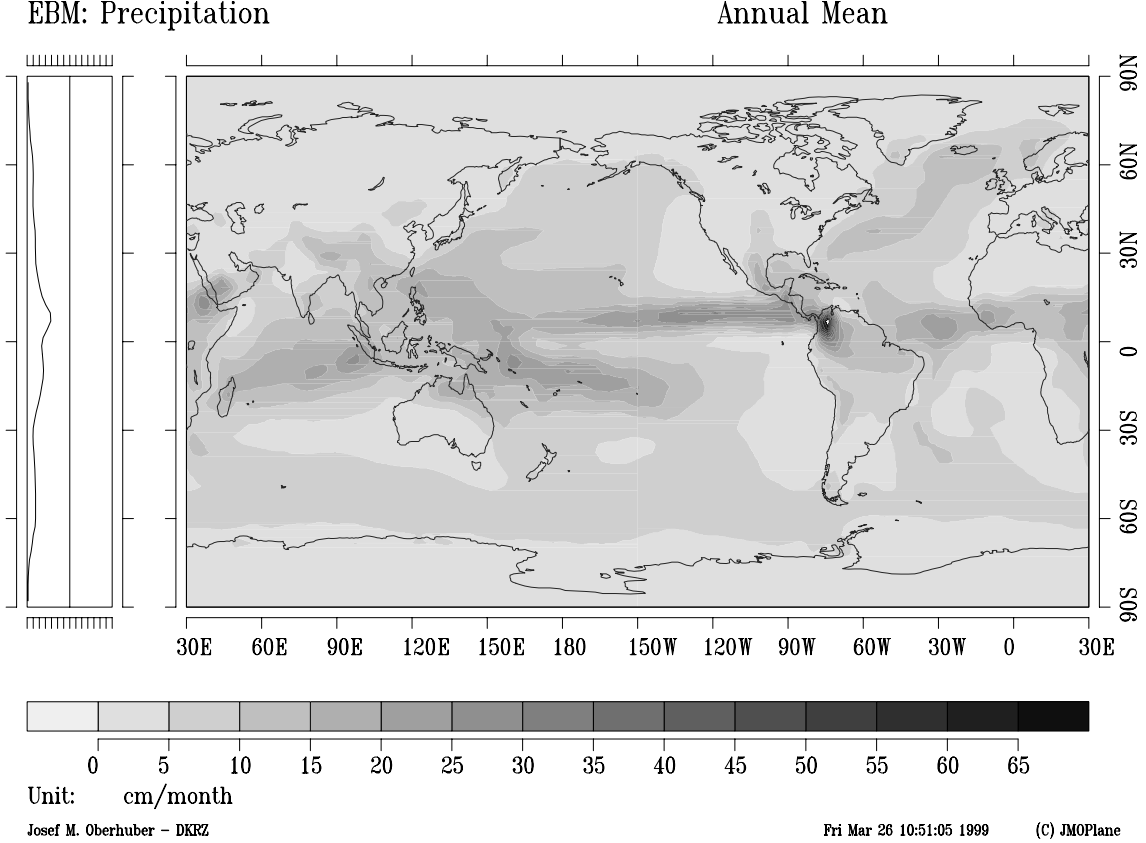

Figure 1.2: Annual mean precipitation as simulated by the EBM in a T42-resolution.

1.6.3 Radiation

Radiation is a key process to control the EBM's temperature distribution. The difference of the short- and longwave radiation at the top and the bottom of the atmosphere is the most important heat source. Bulk parameterizations for the radiation at the surface are known to give reasonable spatial distributions. However, sensitivity experiments showed that their climate sensitivity is unrealistic. Therefore, new formulae have been developed that give similar realistic spatial distributions than earlier bulk formulae, however, behave more realistic in climates of higher water vapour contents (see also Eqns. 1.39 and 1.44). The formulae for the top longwave radiation is:

$$Q_{l,top} = -\varepsilon\sigma T_{eff}^4 \tag{1.109}$$

$$T_{eff} = 223.15 + \frac{0.82}{1.0 + 0.03 * \sqrt{ren^3}}(T_{ebm} - 223.15) \tag{1.110}$$

where T_{eff} is considered to be some effective temperature at which the atmosphere radiates into space. The idea is to let the atmosphere radiate at some weighted average between the rather constant temperature at tropopause level, which here is assumed to be -50^oC and some near-surface temperature. Thus, with increasing humidity and/or cloudiness the effective radiative temperature becomes lower. The top shortwave radiation is parameterized by:

$$Q_{s,top} = \frac{\kappa_t}{2\pi}\int_{t_1}^{t_2} \frac{S_o \cos^2\eta}{((\cos\eta + 2.7)r\frac{e}{p} + 1.085\cos\eta + 0.1)}(\frac{\bar{d}}{d})^2 dt \tag{1.111}$$

$$\kappa_t = \frac{1.0 + 0.0019\eta_{noon}}{1.0 + 0.0019\eta_{equa}}\frac{1}{1.0 + 0.59(1.0 + (0.03e)^2)n^2} \tag{1.112}$$

where $\eta_{equa} = 90$ deg.

1.6.4 River Runoff Model Equations

The river runoff model follows the simple concept of a downhill flow whose speed is dependent on the topography gradient and the amount of water. The prognostic equation for the water height W_{ebm} on land is:

$$\frac{\partial W_{ebm}}{\partial t} = \vec{\nabla} \cdot (\vec{v} W_{ebm}) + P - E + \frac{Q_i}{\rho_i c_{p,m}} \tag{1.113}$$

where P is the precipitation, E the evaporation and the last term being the melting/freezing rate (compare with Eqn. 1.95) which converts ice or snow to water or vice versa. The evaporation on land assumes that the surface is saturated with water for some W_{ebm} with a linear reduction of evaporation with decreasing W_{ebm}. The water's flow speed is defined through the following approximation:

$$\vec{v} = -\epsilon_8 W_{ebm} |\vec{\nabla} O|^{0.25} \frac{\vec{\nabla} O}{|\vec{\nabla} O|} \tag{1.114}$$

where O is the orography height.

1.6.5 Glacier Model Equations

So far the glacier model is realized by extending the functionality of the sea ice model. While its dynamics is disabled on land, the ice and snow mass as well as their thermodynamics are calculated on land as well. In the present version, however, calving of glaciers or glacier flow is not yet considered. This could be included easily based on the same idea as applied for the river runoff model, however, with some diagnostic equation for the ice flow which replaces that for the water flow.

1.6.6 Soil Model Equations

The equation for the soil's temperature T_{soil} is:

$$\frac{\partial T_{soil}}{\partial t} = \frac{Q_{ebm}}{c_{p,s} \rho_s H_s + c_{p,w} \rho_w W_{ebm}} \tag{1.115}$$

where $c_{p,s}$ is the soil's heat capacity, ρ_s the soil's density and H_s the soil thickness. Thus the soil includes the amount of water heated or cooled, however, the heat transport of water is ignored.

1.6.7 Tuning of the EBM

All parameters $\epsilon_1 ... \epsilon_8$ are tuned in such a way that annual mean temperatures fit well with observations. The tuning strategy was to

1. adjust the parameters for short- and longwave radiation to ERBE-data and ECHAM4-AMIP experiments

2. tune the annual zonal mean surface temperature to the ECMWF-analysis by choosing a proper eddy diffusivity that exports enough heat from low into high latitudes

3. tune the effective height used for moisture in order to obtain a realistic vertically integrated water content

4. tune convective precipitation so that enough water is left to be exported into higher latitudes

5. tune adiabatic heating in such a way that the sum of latent heat release due to condensation and adiabatic heating due to vertical motion yields a spatially smooth field

Because parameters depend on each other a hand-based multi-variational approach was used to find an optimal set of parameters. Only annual mean fields have been used to optimize the parameters. Well-known drawbacks of the EBM are that

1. due to a missing surface boundary layer heat and fresh water fluxes are too strong in the subtropical gyres

2. due to missing upper atmosphere northward heat transport due to the Icelandic Low the heat fluxes into the Greenland-Iceland-Norwegian Sea are heavily underestimated

3. due to missing vegetation soil becomes dry and hot during sommer

4. due to missing winds above the atmospheric boundary layer the heat and water transport over larger distances might be underestimated

1.7 Tide Model

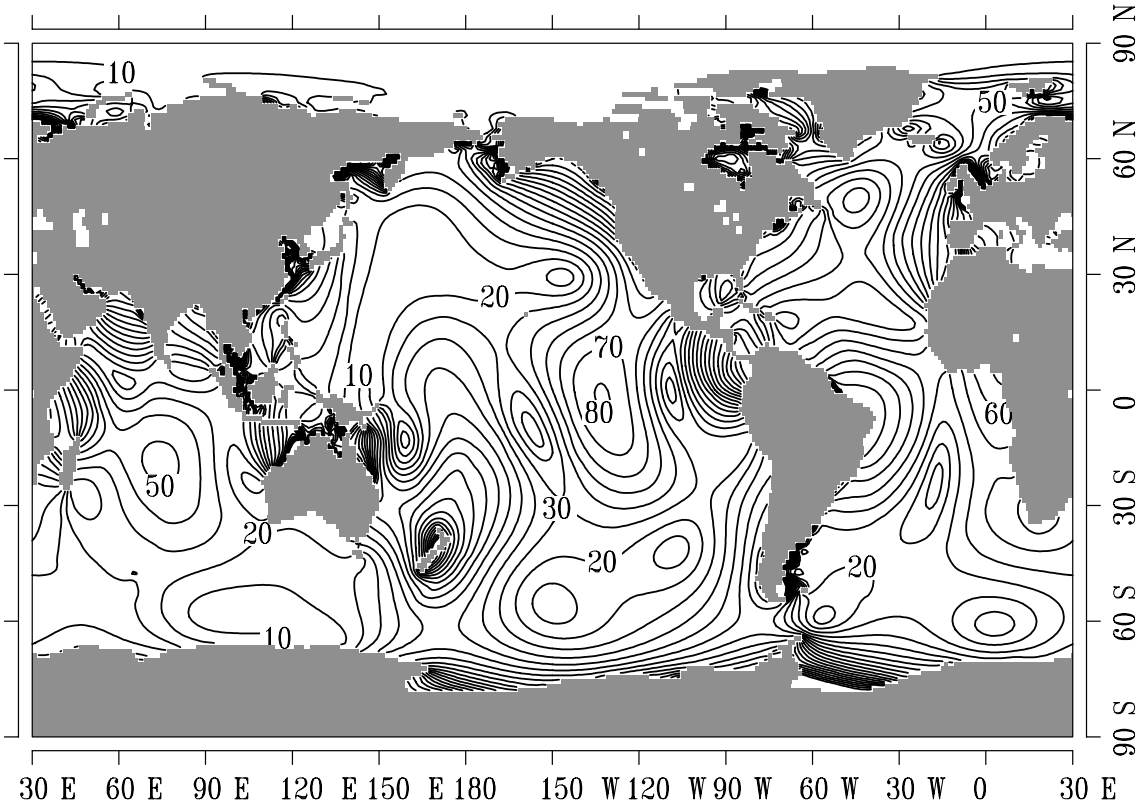

Figure 1.3: Amplitudes of the M2-Tide simulated with a T106 equivalent version of the barotropic tide model coupled to the 3-dimensional PIPE model.

Tides represent high frequency variability that might have important impacts on regional scales. This is because tides become more important with decreasing water depth due to increased interaction between the large-scale mean flow, tidal flow and topography. Thus they might

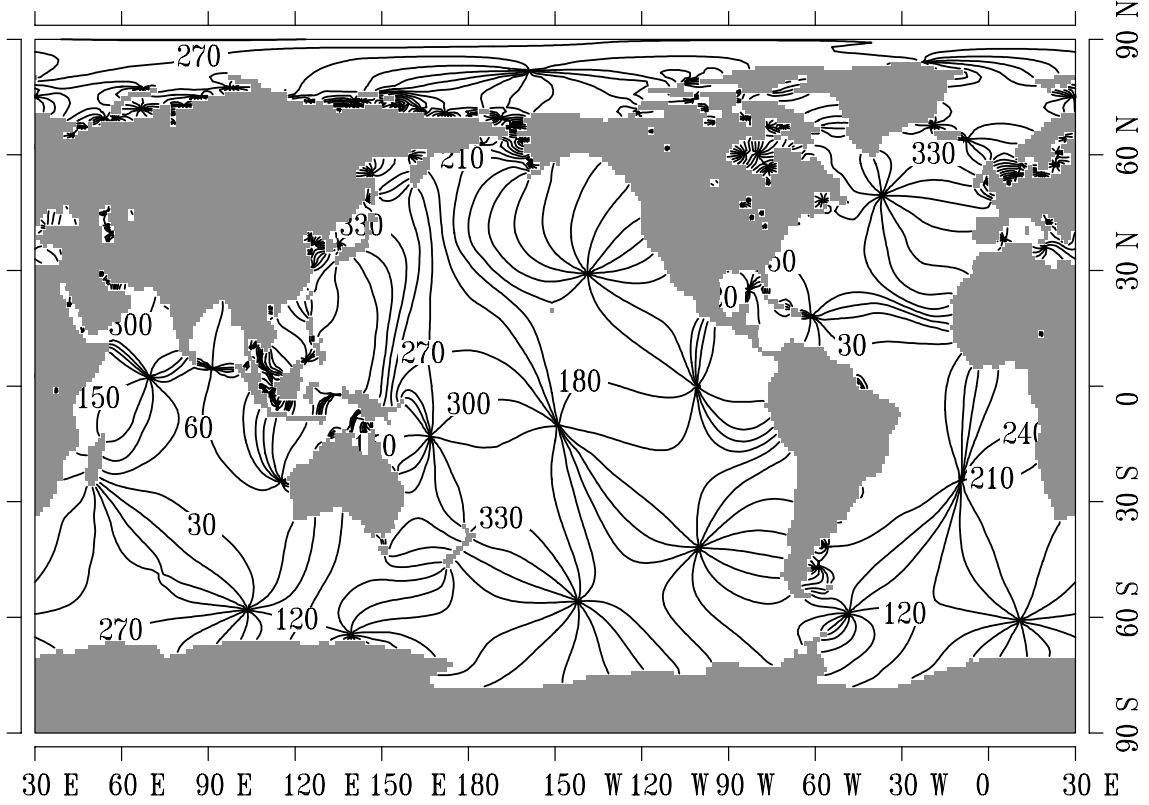

Figure 1.4: Phases of the M2-Tide simulated with a T106 equivalent version of the barotropic tide model coupled to the 3-dimensional PIPE model.

help to improve regional simulations. In order to simulate mean flows that are significantly modified as result of nonlinear interaction with tides a barotropic tide model was developed. If ζ_{tide} is the anomalous sea surface elevation, u_{tide} and v_{tide} the respective zonal and meridional flow anomalies, $H = \sum_k h_k$ the mean ocean depth as simulated by the 3-dimensional model and E_{tide} the gravitational forcing, then the equations are written in flux form:

$$\begin{aligned}
\frac{\partial u_{tide}(H+\zeta_{tide})}{\partial t} = & - g(H+\zeta_{tide})\frac{\partial}{\partial x}(E_{tide}+\zeta_{tide}) + f v_{tide}(H+\zeta_{tide}) \\
& - c_D u_{tide}\sqrt{(u_{tide}+u_N)^2+(v_{tide}+v_N)^2} \qquad (1.116) \\
\frac{\partial v_{tide}(H+\zeta_{tide})}{\partial t} = & - g(H+\zeta_{tide})\frac{\partial}{\partial y}(E_{tide}+\zeta_{tide}) - f u_{tide}(H+\zeta_{tide}) \\
& - c_D v_{tide}\sqrt{(u_{tide}+u_N)^2+(v_{tide}+v_N)^2} \qquad (1.117) \\
\frac{\partial}{\partial t}\zeta_{tide} = & - \frac{\partial}{\partial x}(u_{tide}(H+\zeta_{tide})) - \frac{\partial}{\partial y}(v_{tide}(H+\zeta_{tide})) \qquad (1.118)
\end{aligned}$$

where u_N and v_N are the respective velocities in the lowermost present layer of the isopycnal model. Because the tide model typically uses a much smaller time step than the 3-dimensional model, the tidal flow anomalies are averaged in time and added as additional flow to the bottom friction parameterization. This residual flow is also used in all transport equations in the 3-dimensional ocean model as for temperature, salinity and momentum. Because typical tidal frequencies are in the range of a time step of the 3-dimensional ocean, sampling problems are avoided through further averaging in time in order to obtain smooth residual currents.

Tides are forced through a gravitational anomaly E following Schwiderski (1979). If h_0, s_0 and p_0 are the mean longitudes of sun, moon and lunar perigee at Greenwich midnight, respectively, then these are defined through:

$$T = 27392.500528 + 1.0000000356 \quad D \qquad (1.119)$$

$$h_0 = 279.696680 + 36000.768930485\,T + 0.000303\,T^2 \quad (1.120)$$

$$s_0 = 270.434358 + 481267.88314137\,T - 0.001133\,T^2 + 0.0000019\,T^3 \quad (1.121)$$

$$p_0 = 334.329653 + 4069.0340329575\,T - 0.010325\,T^2 - 0.0000120\,T^3 \quad (1.122)$$

where D is the number of days starting with $D = 0$ at 01/01/1975. This results in an equation for the gravitational anomaly induced by the sun and moon as equivalent sea level anomaly:

$$\begin{aligned} E_{tide} = 0.69 \quad \{ \quad & \cos(\varphi)^2 \sum_{n=1}^{4} K_n \cos(\sigma_n t + 2\lambda + \chi_n) \\ + \quad & \sin(2\varphi) \sum_{n=5}^{8} K_n \cos(\sigma_n t + \lambda + \chi_n) \\ + \quad & (3\cos(\varphi)^2 - 2) \sum_{n=9}^{11} K_n \cos(\sigma_n t + \chi_n)\} \end{aligned} \quad (1.123)$$

where φ is the latitude, λ the longitude and t the time in $10^5 s$. The partial tide's amplitudes K, frequencies σ and astronomical arguments χ are defined in table (1.1).

Tidal Mode	K	σ	χ
M_2 = Principal Lunar	0.242334	1.40519	$2(h_0 - s_0)$
S_2 = Principal Solar	0.112841	1.45444	0
N_2 = Elliptical Lunar	0.046398	1.37880	$2h_0 - 3s_0 + p_0$
K_2 = Declination Luni-Solar	0.030704	1.45842	$2h_0$
K_1 = Declination Luni-Solar	0.141565	0.72921	$h_0 + 90$
O_1 = Principal Lunar	0.100574	0.67598	$h_0 - 2s_0 - 90$
P_1 = Principal Solar	0.046843	0.72523	$-h_0 - 90$
Q_1 = Elliptical Lunar	0.019256	0.64959	$h_0 - 3s_0 - 90$
Mf = Fortnightly Lunar	0.041742	0.053234	$2s_0$
Mm = Monthly Lunar	0.022026	0.026392	$s_0 - p_0$
Ssa = Semiannual Solar	0.019446	0.0038921	$2h_0$

Table 1.1: **Constants of Major Tidal Modes**

As an example of the quality of PIPE's tide submodel Fig. 1.3 shows the simulated amplitudes of the M2 tide and Fig. 1.4 the M2 phases. The model used the 5 minutes NOAA topography without any further tuning. The tide submodel used a time step of 10 minutes, while the 3-dimensional ocean model used a time step of 6 hours.

The residual current required by the bottom stress formulation of the 3-dimensional ocean model and by the transport equation for acyive and passive tracers is computed by time-averaging the tidal flow through $\vec{v}_{res}\Delta t = \int_{\Delta t} \vec{v}_{tide} dt$.

1.8 Tracer Model

In order to be able to in addition simulate passive tracers PIPE has a separate module that can be run in an online or offline mode. This means that the tracer model can be run synchroneously with the ocean model by using those model variables that are used in parallel for the transport of temperature and salinity. In the offline mode, however, it is assumed that the forcing data are saved on a file. This is created while running the ocean model once. Then these data are used to run the tracer model while the rest of the ocean model is disabled. Thus there is a considerable saving in CPU time. Because the ocean model is disabled in that mode, the time

step can be chosen significantly higher because of less restrictive CFL numbers for the tracer transport equation. This contributes to a further significant saving effect. The equations are:

$$\begin{aligned}\frac{\partial}{\partial t}(C\rho h)_k &= -\vec{\nabla}\cdot((C\rho h)_k(\vec{v}_k+\vec{v}_{res}))+\vec{\nabla}\cdot((A^s\rho h)_k\vec{\nabla}C_k)+R_k^C \\ &+ (w\rho C)_k^{k-}+(w\rho C)_k^{k+}-(w\rho C)_{k-}^k-(w\rho C)_{k+}^k \end{aligned} \tag{1.124}$$

where $(C\rho h)$ is the tracer mass, R^C is the tracer forcing and $\vec{v}_{res}$ is the time-averaged residual flow of the barotropic tide model. All other variables are already explained in section (1.1.1).

By default, an early version of the tracer model is part of the collection of codes. This version is not conservative! A newer conservative version is available, however, as it is delicate to use it will be distributed only upon a request and guided with advice.

Chapter 2

Model Numerics

2.1 The Interior Ocean and the Mixed Layer Model

2.1.1 Discretization in Space on the Arakawa B-Grid

The notation is that K is the 1st index, I the 2nd and J the 3rd. A grid point with the indices (1,1,1) is located in the top layer, the mixed layer, and in the south-west corner of the model grid. The B-grid distinguishes between scalar and vector points as shown in Fig. 2.1. In the subsequent formulae scalar quantities have a subscript *S(K,I,J)*, while a vector point is

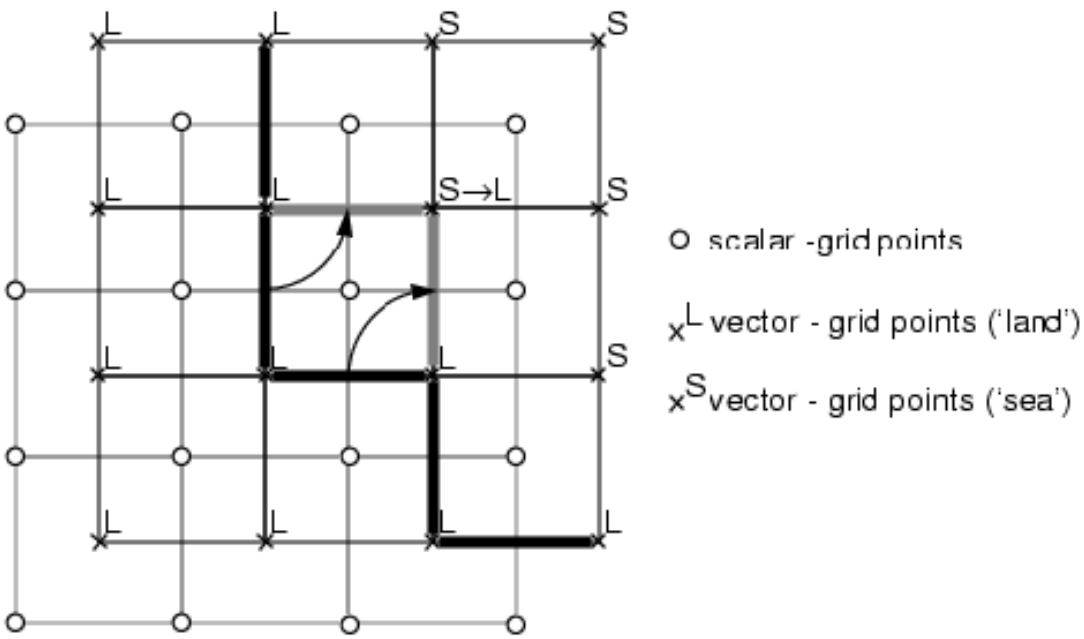

Figure 2.1: Schematic layout of the B-grid. 'L' represents land points, 'S' represents sea points. Circles denote scalar points, crosses denote vector points. The thick line marks the boundary at a certain time level. The dashed line shows an example of how the boundary changes if a vector point is switched from a sea to a land point at the edge of a zero-layer.

marked with the subscript *V(K,I,J)* for a 3-d array. 2-d arrays are similarly marked by *S(I,J)* or *V(I,J)*. Vector quantities are defined on vector points while scalar quantities are defined on scalar points. If a scalar quantity is marked with the subscript *V(K,I,J)* it means that the value is obtained by averaging the surrounding scalar values onto the vector point. Thus, the location of the vector point is located in the center of the cell that is surrounded by the scalar

points *S(I,J)*, *S(I+1,J)*, *S(I,J+1)* and *S(I+1,J+1)*. This means that if a scalar quantity P is needed on a vector point it is computed by

$$P_{V(I,J)} = \frac{P_{S(I,J)} + P_{S(I+1,J)} + P_{S(I,J+1)} + P_{S(I+1,J+1)}}{4} \tag{2.1}$$

The analog for a vector quantity needed on a scalar point is:

$$P_{S(I,J)} = \frac{P_{V(I,J)} + P_{V(I-1,J)} + P_{V(I,J-1)} + P_{V(I-1,J-1)}}{4} \tag{2.2}$$

The model coordinates (*X(NX)* for the x-direction and *Y(NY)* for the y-direction) are defined on vector points. Thus the array X contains the x-values on the equator. The local grid distance on the sphere is obtained by multiplying Δx on the equator by $cos(\varphi)$ (see array *SCALEX(NY)*) which also is defined on vector points.

Δx and Δy values are needed both on vector and scalar points. Since X and Y are defined on vector points, Δx on a scalar point is obtained by

$$\Delta x_{S(I,J)} = cos(\varphi)_{V(J)}(X_{V(I)} - X_{V(I-1)}) \tag{2.3}$$

while Δx on a vector point is:

$$\Delta x_{V(I,J)} = cos(\varphi)_{S(J)} \frac{X_{V(I+1)} - X_{V(I-1)}}{2} \tag{2.4}$$

Δy is obtained on scalar points by

$$\Delta y_{S(I,J)} = (Y_{V(J)} - Y_{V(J-1)}) \tag{2.5}$$

and for Δy on vector points

$$\Delta y_{V(I,J)} = \frac{Y_{V(J+1)} - Y_{V(J-1)}}{2} \tag{2.6}$$

For better vectorization, values of Δx are precalculated and stored into the arrays *HDXZX* and *HDXX* for differences at scalar points and into *VDXZX* and *VDXX* for differences at vector points.

2.1.1.1 Boundary Conditions

Basically, all horizontal boundary conditions are derived from the array *IFLG*, which is defined on vector points. This array contains the information about the coastline geometry. However, it also carries the locations where a physically existing layer with non-zero thickness continues as a zero layer. At these locations a boundary condition has to be computed (see JMO). Mathematically, these time variable boundary conditions are introduced in the same way as the time constant coastline geometry. The only generalization is that *IFLG* contains a 3rd index for the layers. While *IFLG* is constant for the 1st layer, *IFLG* is computed for all deeper layers through the algorithm that sets correct boundary conditions near vanishing or reappearing layers. *IFLG* contains 0 for land and 1 for ocean. Boundary conditions are set at vector points by multiplying each term with *IFLG*, while for scalar points four surrounding values of *IFLG* have to be considered thus requiring detailed knowledge about the origin of each expression.

The interfaces in the isopycnic part of the model physically disappear either into the mixed layer (where the isopycnals are vertical) or into the topography, e.g. at the continental shelf or at the sea floor. Formally, a layer that has physically disappeared is defined in the model as a layer with zero thickness. Grid points in this layer do not contain mass, but still hold dummy values for temperature, salinity and other quantities. An appropriate boundary condition is the key to decouple a zero-layer from the ocean. Furthermore, physical processes that change the

location where an isopycnal disappears/reappears or which create water masses with a not yet existing potential density must have their counterpart in a technique that allows the shifting of boundaries or the flooding of zero-layers. This is explained in JMO.

Simply explained, a finite mass cell remains part of the ocean as long as it contains mass, and is switched off from the ocean only if it has already lost its entire mass and is still losing mass, thus avoiding a negative mass content. Massless cells become part of the ocean if convergence is predicted and remain massless if the flow is divergent.

2.1.1.2 Operators

The equations for momentum, mass, heat, salinity and tracer concentration are formulated in spherical coordinates. The differential operators are taken along the interfaces. They run essentially in the horizontal. The divergence operator $\vec{\nabla}\cdot$ is given by

$$\vec{\nabla}\cdot = (\frac{\partial}{r\cos\varphi\partial\lambda}, \frac{\partial}{r\cos\varphi\partial\varphi}\cos\varphi) \tag{2.7}$$

where λ is the longitude, φ the latitude and r the earth's radius. Similarly, the gradient $\vec{\nabla}$ is

$$\vec{\nabla} = (\frac{\partial}{r\cos\varphi\partial\lambda}, \frac{\partial}{r\partial\varphi}) \tag{2.8}$$

Due to the use of Lagrangian coordinates, vertical derivatives do not appear. The Laplacian operator for scalar quantities is:

$$\nabla\cdot\vec{\nabla} = \frac{1}{r^2cos^2(\varphi)}\frac{\partial^2}{\partial\lambda^2} + \frac{1}{r^2cos(\varphi)}\frac{\partial}{\partial\varphi}(cos(\varphi)\frac{\partial}{\partial\varphi}) \tag{2.9}$$

The diffusion operator of a vector quantity (u,v) on the sphere is:

$$\begin{aligned}\nabla\cdot\vec{\nabla}u &= \frac{1}{r^2cos^2(\varphi)}\frac{\partial^2}{\partial\lambda^2}u + \frac{1}{r^2cos(\varphi)}\frac{\partial}{\partial\varphi}(cos(\varphi)\frac{\partial}{\partial\varphi}u) \\ &- \frac{(1-\tan^2\varphi)u}{r^2} - \frac{2sin(\varphi)}{r^2cos^2(\varphi)}\frac{\partial}{\partial\lambda}v \qquad (2.10)\\ \nabla\cdot\vec{\nabla}v &= \frac{1}{r^2cos^2(\varphi)}\frac{\partial^2}{\partial\lambda^2}v + \frac{1}{r^2cos(\varphi)}\frac{\partial}{\partial\varphi}(cos(\varphi)\frac{\partial}{\partial\varphi}v) \\ &- \frac{(1-\tan^2\varphi)v}{r^2} + \frac{2sin(\varphi)}{r^2cos^2(\varphi)}\frac{\partial}{\partial\lambda}u \qquad (2.11)\end{aligned}$$

2.1.1.3 The Pressure Gradient

$$\frac{\partial}{\partial x}h_{V(I,J)} = \frac{h_{S(I+1,J)} + h_{S(I+1,J+1)} - h_{S(I,J)} - h_{S(I,J+1)}}{2\Delta x_{V(I,J)}} \tag{2.12}$$

$$\frac{\partial}{\partial y}h_{V(I,J)} = \frac{h_{S(I,J+1)} + h_{S(I+1,J+1)} - h_{S(I,J)} - h_{S(I+1,J)}}{2\Delta y_{V(I,J)}} \tag{2.13}$$

The same discretization is used for σ_θ. Note that the pressure gradient within each layer is the sum of layer thickness gradients, topography gradients, and potential density gradients.

2.1.1.4 The Flux Divergence

$$\begin{aligned}
\frac{\partial}{\partial x}(\rho uh)_{S(I,J)} &= \frac{(\rho uh)_{V(I,J-1)} + (\rho uh)_{V(I,J)}}{2\Delta x_{S(J)}} \\
&- \frac{(\rho uh)_{V(I-1,J-1)} + (\rho uh)_{V(I-1,J)}}{2\Delta x_{S(J)}} \qquad (2.14) \\
\frac{\partial}{\partial y}(\rho vh)_{S(I,J)} &= \frac{\cos(\varphi)_{V(J)}((\rho vh)_{V(I-1,J)} + (\rho vh)_{V(I,J)})}{(cos(\varphi)_{V(J-1)} + cos(\varphi)_{V(J)})\Delta y_{S(J)}} \\
&- \frac{cos(\varphi)_{V(J-1)}((\rho vh)_{V(I-1,J-1)} + (\rho vh)_{V(I,J-1)})}{(cos(\varphi)_{V(J-1)} + cos(\varphi)_{V(J)})\Delta y_{S(J)}} \qquad (2.15)
\end{aligned}$$

2.1.1.5 Horizontal Diffusion of Momentum

$$\begin{aligned}
\frac{\partial}{\partial x}(A\rho h)\frac{\partial}{\partial x}\vec{v}_{V(I,J)} &= \frac{((A\rho h)_{S(I+1,J)} + (A\rho h)_{S(I+1,J+1)})(\vec{v}_{V(I+1,J)} - \vec{v}_{V(I,J)})}{2\Delta x_{S(I+1)}\Delta x_{V(I)}} \\
&- \frac{((A\rho h)_{S(I,J)} + (A\rho h)_{S(I,J+1)})(\vec{v}_{V(I,J)} - \vec{v}_{V(I-1,J)})}{2\Delta x_{S(I)}\Delta x_{V(I)}} \qquad (2.16) \\
\frac{\partial}{\partial y}(A\rho h)\frac{\partial}{\partial y}\vec{v}_{V(I,J)} &= \frac{((A\rho h)_{S(I,J+1)} + (A\rho h)_{S(I+1,J+1)})cos(\varphi)_{S(J+1)}(\vec{v}_{V(I,J+1)} - \vec{v}_{V(I,J)})}{\Delta y_{S(J+1)}\Delta y_{V(I)}(cos(\varphi)_{S(J)} + cos(\varphi)_{S(J+1)})} \\
&- \frac{((A\rho h)_{S(I,J)} + (A\rho h)_{S(I+1,J)})cos(\varphi)_{S(J)}(\vec{v}_{V(I,J)} - \vec{v}_{V(I,J-1)})}{\Delta y_{S(J)}\Delta y_{V(J)}(cos(\varphi)_{S(J)} + cos(\varphi)_{S(J-1)})} \qquad (2.17)
\end{aligned}$$

If the older version for diffusing momentum instead of velocity is used, then $(A\rho h)$ is replaced by A and $\vec{v}$ by $(v\vec{\rho h})$.

2.1.1.6 Horizontal Advection of Momentum

Depending on the value of the switch *ISW32* in /JMOFLAG/, one of the following advection schemes is chosen:

2.1.1.6.1 The Crowley Scheme: The scheme by Crowley (1968) has the following form:

$$\begin{aligned}
\frac{\partial}{\partial x}u(\rho uh)_{V(I,J)} &= (u_{V(I,J)} + u_{V(I+1,J)})(1 - (u_{V(I,J)} + u_{V(I+1,J)})\frac{\Delta t}{2\Delta x_{S(I+1)}}) \qquad (2.18) \\
&* \frac{((\rho uh)_{V(I+1,J)} - (\rho uh)_{V(I,J)})}{2\Delta x_{S(I+1)}} \\
&+ (u_{V(I-1,J)} + u_{V(I,J)})(1 + (u_{V(I-1,J)} + u_{V(I,J)})\frac{\Delta t}{2\Delta x_{S(I)}}) \\
&* \frac{(\rho uh)_{V(I,J)} - (\rho uh)_{V(I-1,J)}}{2\Delta x_{S(I)}} \\
\frac{\partial}{\partial y}v(\rho uh)_{V(I,J)} &= (v_{V(I,J)} + v_{V(I,J+1)})(1 - (v_{V(I,J)} + v_{V(I,J+1)})\frac{\Delta t}{2\Delta y_{S(J+1)}}) \qquad (2.19) \\
&* \frac{cos(\varphi)_{V(J+1)}(\rho uh)_{V(I,J+1)} - cos(\varphi)_{V(J)}(\rho uh)_{V(I,J)}}{2\Delta y_{S(J+1)}(cos(\varphi)_{V(J+1)} + cos(\varphi)_{V(J)})} \\
&+ (v_{V(I,J-1)} + v_{V(I,J)})(1 + (v_{V(I,J-1)} + v_{V(I,J)})\frac{\Delta t}{2\Delta y_{S(J)}}) \\
&* \frac{cos(\varphi)_{V(J)}(\rho uh)_{V(I,J)} - cos(\varphi)_{V(J-1)}(\rho uh)_{V(I,J-1)}}{2\Delta y_{S(J)}(cos(\varphi)_{V(J+1)} + cos(\varphi)_{V(J)})}
\end{aligned}$$

2.1.1.6.2 The Potential Vorticity and Energy Conserving Scheme: The basic idea used to obtain a potential vorticity and energy conserving scheme for momentum transport (Bleck and Boudra, 1981) is to rearrange the momentum advection and Coriolis terms so that the terms on the right hand side of the momentum Eqn. (1.1) can be rewritten as, for the x-component:

$$\begin{aligned} - & \frac{\partial}{\partial x}(uu\rho h) - \frac{\partial}{\partial y}(vu\rho h) + fv\rho h + v\rho h\frac{u\tan\varphi}{r} = \\ - & \frac{\rho h}{2}(\frac{\partial u^2}{\partial x} + \frac{\partial v^2}{\partial x}) - u(\frac{\partial u\rho h}{\partial x} + \frac{\partial v\rho h}{\partial y}) + v\rho h\frac{u\tan\varphi}{r} + (f+\zeta)v\rho h \end{aligned} \tag{2.20}$$

and for the y-component:

$$\begin{aligned} - & \frac{\partial}{\partial x}(uv\rho h) - \frac{\partial}{\partial y}(vv\rho h) - fu\rho h - u\rho h\frac{u\tan\varphi}{r} = \\ - & \frac{\rho h}{2}(\frac{\partial u^2}{\partial y} + \frac{\partial v^2}{\partial y}) - v(\frac{\partial u\rho h}{\partial x} + \frac{\partial v\rho h}{\partial y}) - u\rho h\frac{u\tan\varphi}{r} - (f+\zeta)u\rho h \end{aligned} \tag{2.21}$$

where ζ is the relative vorticity. The right hand sides consist of terms representing the energy gradient, momentum convergence, and curvature, and of an altered Coriolis term that now contains the absolute vorticity instead of only the Coriolis parameter. This term is included in the implicit formulation of the layer Eqn. (2.40) which is the predictor step. The residual terms are treated implicitly in the corrector step which also contains the momentum diffusion terms. Since the momentum components are cross-coupled in the Eqns. (2.33) and (2.34), a linear equation in x is formulated simultaneously for both velocity components. Alternatively, a linear equation for the y-direction may improve performance. The discretizations for the flux divergence are those in (2.1.1.4), the energy gradient depending on u are approximated by (the same formulae are valid for v) :

$$\begin{aligned} \frac{\rho h}{2}\frac{\partial u^2}{\partial x} &= \frac{(\rho hu)_{V(I,J)} + (\rho hu)_{V(I+1,J)}}{4}\frac{u_{V(I+1,J)} - u_{V(I,J)}}{\Delta x_{S(I+1)}} \\ &+ \frac{(\rho hu)_{V(I-1,J)} + (\rho hu)_{V(I,J)}}{4}\frac{u_{V(I,J)} - u_{V(I-1,J)}}{\Delta x_{S(I)}} \end{aligned} \tag{2.22}$$

$$\begin{aligned} \frac{\rho h}{2}\frac{\partial u^2}{\partial y} &= \frac{(\rho hu)_{V(I,J+1)} + (\rho hu)_{V(I,J)}}{2}\frac{cos(\varphi)_{V(J+1)}u_{V(I,J+1)} - cos(\varphi)_{V(J)}u_{V(I,J)}}{(cos(\varphi)_{V(J)} + cos(\varphi)_{V(J+1)})\Delta y_{S(J+1)}} \\ &+ \frac{(\rho hu)_{V(I,J)} + (\rho hu)_{V(I,J-1)}}{2}\frac{cos(\varphi)_{V(J)}u_{V(I,J)} - cos(\varphi)_{V(J-1)}u_{V(I,J-1)}}{(cos(\varphi)_{V(J-1)} + cos(\varphi)_{V(J)})\Delta y_{S(J)}} \end{aligned} \tag{2.23}$$

The local vorticity ζ is discretized by

$$\begin{aligned} \zeta_{V(I,J)} &= \frac{v_{S(I,J-1)} + v_{S(I,J)} - v_{S(I-1,J-1)} - v_{S(I-1,J)}}{2\Delta x_{V(I)}} \\ &- \frac{u_{S(I-1,J)} + u_{S(I,J)} - u_{S(I-1,J-1)} - u_{S(I,J-1)}}{2\Delta y_{V(J)}} \end{aligned} \tag{2.24}$$

This involves a horizontal filter that removes efficiently B-grid computational modes from the local vorticity ζ.

2.1.1.7 Horizontal Diffusion of Scalar Variables

$$\frac{\partial}{\partial x}A\frac{\partial}{\partial x}\theta_{S(I,J)} = \frac{(A_{S(I,J)} + A_{S(I+1,J)})(\theta_{S(I+1,J)} - \theta_{S(I,J)})}{2\Delta x_{V(I)}\Delta x_{S(I)}} \tag{2.25}$$

$$
\begin{aligned}
&+ \quad \frac{(A_{S(I-1,J)} + A_{S(I,J)})(\theta_{S(I,J)} - \theta_{S(I-1,J)})}{2\Delta x_{V(I-1)} \Delta x_{S(I)}} \\
\frac{\partial}{\partial y} A \frac{\partial}{\partial y} \theta_{S(I,J)} &= \frac{(A_{S(I,J+1)} + A_{S(I,J)})(\theta_{S(I,J+1)} - \theta_{S(I,J)}) cos(\varphi)_{V(J)}}{\Delta y_{S(J)} \Delta y_{V(J)} (cos(\varphi)_{V(J-1)} + cos(\varphi)_{V(J)})} \qquad (2.26) \\
&+ \quad \frac{(A_{S(I,J)} + A_{S(I,J-1)})(\theta_{S(I,J)} - \theta_{S(I,J-1)}) cos(\varphi)_{V(J-1)}}{\Delta y_{S(J)} \Delta y_{V(J-1)} (cos(\varphi)_{V(J-1)} + cos(\varphi)_{V(J)})}
\end{aligned}
$$

The same discretization is used for the salinity S.

2.1.1.8 Horizontal Advection of Scalar Variables: The Crowley Scheme

The advection scheme is similar to that discussed by Smolarkiewicz (1982). It is an implicit version of the scheme presented by Crowley (1968). The scheme is less diffusive than the upstream scheme and reduces the tendency of centered difference schemes to overshoot. It was modified so that it converges towards the upstream scheme for $CFL > 1$. This has to be done as otherwise the artificial diffusion introduced by the Crowley scheme dominates the physics for $CFL > 1$. It can be understood as a quasi-Lagrangian scheme in which higher-order terms are neglected in the Taylor expansion of the transport equation.

$$
\begin{aligned}
\frac{\partial}{\partial x}(\theta \rho u h)_{S(I,J)} &= ((\rho u h)_{V(I,J)} + (\rho u h)_{V(I,J-1)})(1 - (u_{V(I,J-1)} + u_{V(I,J)}) \frac{\Delta t}{2\Delta x_{V(I)}}) \qquad (2.27) \\
&* \quad \frac{\theta_{S(I+1,J)} - \theta_{S(I,J)}}{4\Delta x_{V(I)}} \\
&+ \quad ((\rho u h)_{V(I-1,J-1)} + (\rho u h)_{V(I-1,J)})(1 + (u_{V(I-1,J-1)} + u_{V(I-1,J)}) \frac{\Delta t}{2\Delta x_{V(I-1)}}) \\
&* \quad \frac{\theta_{S(I,J)} - \theta_{S(I-1,J)}}{4\Delta x_{V(I-1)}} \\
\frac{\partial}{\partial y}(\theta \rho v h)_{S(I,J)} &= ((\rho u h)_{V(I-1,J)} + (\rho u h)_{V(I,J)})(1 - (u_{V(I-1,J)} + u_{V(I,J)}) \frac{\Delta t}{2\Delta y_{V(J)}}) \qquad (2.28) \\
&* \quad \frac{\theta_{S(I,J+1)} - \theta_{S(I,J)}}{2\Delta y_{V(J)}} \frac{cos(\varphi)_{V(J)}}{cos(\varphi)_{V(J-1)} + cos(\varphi)_{V(J)}} \\
&+ \quad ((\rho u h)_{V(I-1,J-1)} + (\rho u h)_{V(I,J-1)})(1 + (u_{V(I-1,J-1)} + u_{V(I,J-1)}) \frac{\Delta t}{2\Delta y_{V(J-1)}}) \\
&* \quad \frac{\theta_{S(I,J)} - \theta_{S(I,J-1)}}{2\Delta y_{V(J-1)}} \frac{cos(\varphi)_{V(J-1)}}{cos(\varphi)_{V(J-1)} + cos(\varphi)_{V(J)}}
\end{aligned}
$$

2.1.2 Discretization in Time

2.1.2.1 Predictor Step

In order to allow larger time steps a predictor-corrector technique is adopted. Each of these steps is based on a semi-implicit method. For details see Robert et al. (1972). The method yields an unconditionally stable time integration scheme with respect to all external and internal waves, advection, diffusion and mixed layer physics. Coupling of the semi-implicit steps has been described in Oberhuber (1986). The predictor step can be written as

$$
\begin{aligned}
(\rho \vec{v} h)^{*n+1} - \frac{\Delta t}{2} \vec{G}^{*n+1}_{(\rho \vec{v} h)} &= (\rho \vec{v} h)^{n} + \frac{\Delta t}{2} \vec{G}^{*n}_{(\rho \vec{v} h)} + \Delta t \vec{F}^{*n}_{(\rho \vec{v} h)} \qquad (2.29) \\
(\rho h)^{*n+1} - \frac{\Delta t}{2} G^{*n+1}_{(\rho h)} &= (\rho h)^{n} + \frac{\Delta t}{2} G^{*n}_{(\rho h)} + \Delta t F^{*n}_{(\rho h)} \qquad (2.30)
\end{aligned}
$$

$$(\rho h\theta)^{*n+1} - \frac{\Delta t}{2} G^{*n+1}_{(\rho h\theta)} = (\rho h\theta)^n + \frac{\Delta t}{2} G^{*n}_{(\rho h\theta)} + \Delta t F^{*n}_{(\rho h\theta)} \quad (2.31)$$

$$(\rho hS)^{*n+1} - \frac{\Delta t}{2} G^{*n+1}_{(\rho hS)} = (\rho hS)^n + \frac{\Delta t}{2} G^{*n}_{(\rho hS)} + \Delta t F^{*n}_{(\rho hS)} \quad (2.32)$$

where $(\rho\vec{v}h)$ represents the mass flux, (ρh) the mass content, $(\rho h\theta)$ the heat content and (ρhS) the salt content. All equations are discretized in time using an Euler scheme. G represents all those terms that are treated implicitly and F those terms that are explicitly computed with forward steps. In the predictor step the pressure gradient, the Coriolis term and the vorticity part of the advection in the momentum equation, the flux divergence in the continuity equation, and the advection and diffusion terms in the equation for potential temperature and salinity are treated implicitly. All remaining terms are collected in F.

In order to demonstrate how the technique works it is explained here in detail how to derive the wave equation. A simplified derivation of the semi-implicit scheme can be found in Appendix A. In the first step, the momentum equation is discretized in time by using a centered Euler scheme. However, the code allows for an arbitrary choice for the forward and backward weights. The x- and y-components of the momentum Eqn. (1.1) then read:

$$(\rho uh)^{n+1} = (\rho uh)^n - \frac{\Delta t}{2} h^{n+1} \frac{\partial}{\partial x} p^{n+1} + \frac{\Delta t}{2} (f+\zeta)(\rho vh)^{n+1} + F^n_{\rho uh} \quad (2.33)$$

$$(\rho vh)^{n+1} = (\rho vh)^n - \frac{\Delta t}{2} h^{n+1} \frac{\partial}{\partial y} p^{n+1} - \frac{\Delta t}{2} (f+\zeta)(\rho uh)^{n+1} + F^n_{\rho vh} \quad (2.34)$$

where $F^n_{\rho uh}$ and $F^n_{\rho vh}$ represent all the remaining terms taken at time level n. In the next step the equations are solved for $(\rho uh)^{n+1}$ or $(\rho vh)^{n+1}$ by eliminating the cross-referenced flux components. This yields:

$$(\rho uh)^{n+1} = F^{*n}_{\rho uh} - \frac{\frac{\Delta t}{2} h^{n+1} \frac{\partial}{\partial x} p^{n+1} + \frac{\Delta t^2}{4} (f+\zeta) h^{n+1} \frac{\partial}{\partial y} p^{n+1}}{1 + \frac{\Delta t^2}{4} (f+\zeta)^2} \quad (2.35)$$

$$(\rho vh)^{n+1} = F^{*n}_{\rho vh} - \frac{\frac{\Delta t}{2} h^{n+1} \frac{\partial}{\partial y} p^{n+1} - \frac{\Delta t^2}{4} (f+\zeta) h^{n+1} \frac{\partial}{\partial x} p^{n+1}}{1 + \frac{\Delta t^2}{4} (f+\zeta)^2} \quad (2.36)$$

where $F^{*n}_{\rho uh}$ and $F^{*n}_{\rho vh}$ are the abbreviations of

$$F^{*n}_{\rho uh} = \frac{(\rho uh)^n + F^n_{\rho uh} + \frac{\Delta t}{2} (f+\zeta)((\rho vh)^n + F^n_{\rho vh})}{1 + \frac{\Delta t^2}{4} (f+\zeta)^2} \quad (2.37)$$

$$F^{*n}_{\rho vh} = \frac{(\rho vh)^n + F^n_{\rho vh} - \frac{\Delta t}{2} (f+\zeta)((\rho uh)^n + F^n_{\rho uh})}{1 + \frac{\Delta t^2}{4} (f+\zeta)^2} \quad (2.38)$$

If the continuity equation is discretized in the same manner as the momentum equation this yields:

$$(\rho h)^{n+1} = (\rho h)^n - \frac{\Delta t}{2} \left(\frac{\partial}{\partial x} (\rho uh)^{n+1} + \frac{\partial}{\partial y} (\rho vh)^{n+1} \right) + F^n_{\rho h} \quad (2.39)$$

where $F^n_{\rho h}$ represents all the remaining terms taken at time level n. If the flux divergence now is eliminated by using (2.35) and (2.36) an equation for the layer thickness h only is obtained:

$$\begin{aligned} (h^{n+1}_k - h^n_k) \frac{\rho^n_k + \rho^{n+1}_k}{2} &= F^{*n}_{\rho h} - (\rho^{n+1}_k - \rho^n_k) \frac{h^n_k + h^{n+1}_k}{2} \quad (2.40) \\ + \frac{\Delta t^2}{4} \frac{\partial}{\partial x} \frac{h^{n+1}_k}{1 + \Delta t^2 (f+\zeta)^2/4} \frac{\partial}{\partial x} p^{n+1}_k &+ \frac{\Delta t^2}{4} \frac{\partial}{\partial y} \frac{h^{n+1}_k}{1 + \Delta t^2 (f+\zeta)^2/4} \frac{\partial}{\partial y} p^{n+1}_k \\ + \frac{\Delta t^2}{4} \frac{\partial}{\partial x} \frac{h^{n+1}_k \Delta t (f+\zeta)/2}{1 + \Delta t^2 (f+\zeta)^2/4} \frac{\partial}{\partial y} p^{n+1}_k &- \frac{\Delta t^2}{4} \frac{\partial}{\partial y} \frac{h^{n+1}_k \Delta t (f+\zeta)/2}{1 + \Delta t^2 (f+\zeta)^2/4} \frac{\partial}{\partial x} p^{n+1}_k \end{aligned}$$

The following identity is used to separate ρ and h from the product ρh:

$$(\rho h)^{n+1} - (\rho h)^n = (\rho^n + \rho^{n+1})\frac{h^{n+1} - h^n}{2} + (h^n + h^{n+1})\frac{\rho^{n+1} - \rho^n}{2} \tag{2.41}$$

This relation is also used to split the heat and salt content up into their basic quantities which are potential temperature and salinity. The quantity $F^{*n}_{\rho h}$ denotes the explicit part of the continuity equation in the semi-implicit technique, defined by

$$F^{*n}_{\rho h} = F^n_{\rho h} - \frac{\Delta t}{2}(\frac{\partial}{\partial x}F^{*n}_{\rho uh} + \frac{\partial}{\partial y}F^{*n}_{\rho vh}) \tag{2.42}$$

Together with the equation of state (1.6) and the relation for the in situ pressure (1.7) and (1.8), Eqn. (2.40) determines h^{n+1}_k. The density at the new time level is updated during the iteration by using the last guess for the layer thickness and the previously determined potential temperature and salinity. Finally, after having found the solution for layer thickness and density, the mass fluxes are obtained from Eqns. (2.35) through (2.36).

The problem is formulated as a linear equation with frozen matrix coefficients during one iteration step. Between iteration steps the matrix coefficients are updated with a method that avoids oscillations around the solution. The layer equation is formulated as a system of linear equations in the x- and z-direction and iterated in y, which is an application of the *line-relaxation* method. Optionally, the equations are formulated directly in y- and z-direction in addition in order to improve convergence. The layer equation consists of quadratic terms, which represent the inertia-gravity waves, and mixed derivatives, which correspond to Rossby waves. Because of the large time steps used in this model ($\Delta t(f+\zeta)/2 > 1$), the mixed derivative terms dominate the quadratic terms, so that simple iteration methods used for Laplacian-type equations cannot be used. Based on the fact that an iteration converges if the mean diagonal elements dominate over the neighbouring diagonals, a method has been developed which calculates an optimal relaxation coefficient for every grid point by changing the mean diagonal element artificially without changing the final solution.

An issue worth noting is that there exists a concept to derive the correct boundary conditions used for solving the pressure Eqn. (2.40). The concept is explained in Appendix B with the help of Appendix A. The same concept is used as part of the tide model, which is an application that demonstrates that external gravity waves are realistically reflected along coasts (see also Fig. (1.1) and (1.2)).

2.1.2.2 Corrector Step

The entrainment and detrainment rate and the resulting changes in the mass fluxes are treated implicitly in the corrector step. The solution for the mixed layer is obtained with Newton's method to find zeros of the resulting nonlinear equation. Because no spatial derivatives occur the following equations can be solved pointwise:

$$(\rho\vec{v}h)^{**n+1} - \frac{\Delta t}{2}\vec{G}^{**n+1}_{\rho\vec{v}h} = (\rho\vec{v}h)^{*n+1} - \frac{\Delta t}{2}\vec{G}^{**n}_{\rho\vec{v}h} \tag{2.43}$$

$$(\rho h)^{**n+1} - \frac{\Delta t}{2}G^{**n+1}_{\rho h} = (\rho h)^{*n+1} - \frac{\Delta t}{2}G^{**n}_{\rho h} \tag{2.44}$$

$$(\rho h\theta)^{**n+1} - \frac{\Delta t}{2}G^{**n+1}_{\rho h\theta} = (\rho h\theta)^{*n+1} - \frac{\Delta t}{2}G^{**n}_{\rho h\theta} \tag{2.45}$$

$$(\rho hS)^{**n+1} - \frac{\Delta t}{2}G^{**n+1}_{\rho hS} = (\rho hS)^{*n+1} - \frac{\Delta t}{2}G^{**n}_{\rho hS} \tag{2.46}$$

$(\rho\vec{v}h)^{**n+1}$, $(\rho h)^{**n+1}$, $(\rho h\theta)^{**n+1}$ and $(\rho hS)^{**n+1}$ are the corrected guesses for the new time level n+1.

2.2 The Sea Ice Model

2.2.1 Discretization in Space

The sea ice model works on the same grid and with the same time step as the ocean model. It is written in spherical coordinates. This means that the momentum diffusion is discretized according to (2.1.1.5), the sea level pressure gradient according to (2.1.1.3), the ice mass and compactness diffusion according to (2.1.1.7) and the mass and compactness convergence according to (2.1.1.4). The discretization of the quadratic and mixed derivatives of the rheology terms is based on 9-point formulae. Thus, the derivatives of the velocity components are carried out on scalar points according to (2.1.1.4) and the result is differentiated on vector points according to (2.1.1.3). Note that the 9-point formulae do not diffuse B-grid computational waves which appear as checker-board pattern. However, these spurious waves do not appear in the solutions anyway, because the formulae are consistent with the divergence terms and thus permit an explicit Eulerforward step for the mass and compactness convergence.

2.2.2 Discretization in Time

A predictor-corrector method in connection with the semi-implicit technique is applied. First, diffusion of ice thickness and ice concentration are determined implicitly. In this step the flux divergence of ice thickness and ice concentration are still taken explicitly forward. In the next step, the stress and Coriolis term are treated implicitly. In the final correction step the sea ice rheology in flux form is determined implicitly. The coupling of the predictor and two corrector steps has already been outlined in the contest of the ocean model. All iterations are carried out in the y-direction only, since the system of linear equations is solved directly in x. The matrix coefficients in the rheology part are updated during the iteration to account for the extreme nonlinearities in the viscous-plastic rheology. This means that bulk and shear viscosities are taken partly at the new time step. For consistency with the continuity equation, 9-point formulae are taken for the rheology. The formulation of the model ensures that the small diffusion coefficient A in equations (1.77) to (1.80) is sufficient to guarantee computational stability.

2.3 The Atmospheric Energy Balance Model

The energy balance model works on the same grid and with the same time step as the ocean model. As transport scheme the Crowley-scheme is used for temperature transport. Humidity is transported through flux-form equations. The according divergence operator is implicitly formulated as part of a matrix system.

2.4 Tide Model

The tide model is formulated on an Arakawa-C grid, however, scalar points coincide with those of the ocean model. In order to allow tides to be resolved in time an internally defined small time step is used. The tide model is subcycled to synchronize it with the ocean model. The iterative scheme is an alternating direction line-relaxation (ADLR) scheme, which gives good results even at surprising long time steps of e.g. 10 minutes at 1 degree resolution. The derivation of the wave equation differs to that of the layer thickness equation such that the Coriolis-terms are not eliminated from the pressure equation. Instead, the final equations are derived as follows.

If the tide equations for simplicity written in non-dimensional form are discretized in time by

$$u^{n+1} = F_u - \frac{\Delta t}{2}(\frac{\partial}{\partial x}p^{n+1} - fv^{n+1}) \tag{2.47}$$

$$v^{n+1} = F_v - \frac{\Delta t}{2}(\frac{\partial}{\partial y}p^{n+1} + fu^{n+1}) \tag{2.48}$$

$$p^{n+1} = F_p - \frac{\Delta t}{2}(\frac{\partial}{\partial x}u^{n+1} + \frac{\partial}{\partial y}v^{n+1}) \tag{2.49}$$

where F_u, F_v and F_p are defined by

$$F_u = u^n - \frac{\Delta t}{2}(\frac{\partial}{\partial x}p^n - fv^n) \tag{2.50}$$

$$F_v = v^n - \frac{\Delta t}{2}(\frac{\partial}{\partial y}p^n + fu^n) \tag{2.51}$$

$$F_p = p^n - \frac{\Delta t}{2}(\frac{\partial}{\partial x}u^n + \frac{\partial}{\partial y}v^n) \tag{2.52}$$

and if F_p^* is defined as

$$F_p^* = F_p - \frac{\Delta t}{2}(\frac{\partial}{\partial x}F_u + \frac{\partial}{\partial y}F_v) \tag{2.53}$$

then the equations for the pressure and the velocities at the new time level become

$$p^{n+1} - \frac{\Delta t^2}{4}(\frac{\partial^2}{\partial x^2}p^{n+1} + \frac{\partial^2}{\partial y^2}p^{n+1}) = F_p^* - \frac{\Delta t^2}{4}f(\frac{\partial}{\partial x}v^{n+1} - \frac{\partial}{\partial y}u^{n+1}) \tag{2.54}$$

$$u^{n+1} = \frac{F_u + \frac{\Delta t}{2}fF_v - \frac{\Delta t}{2}\frac{\partial}{\partial x}p^{n+1} + \frac{\Delta t^2}{4}f\frac{\partial}{\partial y}p^{n+1}}{1 + \frac{\Delta t^2}{4}f^2} \tag{2.55}$$

$$v^{n+1} = \frac{F_v - \frac{\Delta t}{2}fF_u - \frac{\Delta t}{2}\frac{\partial}{\partial y}p^{n+1} - \frac{\Delta t^2}{4}f\frac{\partial}{\partial x}p^{n+1}}{1 + \frac{\Delta t^2}{4}f^2} \tag{2.56}$$

The latter three equations form a system of equations for the three unknowns at the new time level, i.e. the pressure and the two velocity components. Because the factor $\frac{\Delta t}{2}f < 1$ for tidal applications the vorticity term in Eqn. (2.54) can be updated during the iteration for the pressure by using Eqns. (2.55) and (2.56) for the velocity components at the new time step. Convergence is guaranteed if the time step is shorter than the inertial period. The time step of the tide model *DTIDE* is hardcoded in routine <POTTIDE> and currently is set to 900 seconds.

2.5 Open Boundary Formulation

The basic philosophy to install open boundary conditions into PIPE is to feed the model with sections of temperature and salinity, i.e. with the pressure field, while leaving it to the physics of the model to generate any flow at and near the open boundaries. This strategy avoids an overdetermined system. Therefore, one may resign on a swamp near the open boundaries as required in many other ocean and atmosphere models when the pressure and the flow fields are prescribed on and near the open boundaries.

Because 3-dimensional ocean data do not provide an information on the barotropic mode the computation of the full pressure is performed in the following order:

1. transform a temperature and salinity profile into layer thicknesses with integrated temperature and salinity as property

2. compute the baroclinic pressure, i.e. the internal isopycnal interfaces and the according sea level elevation in order to obtain a zero pressure gradient at the bottom

3. take the barotropic mode from a global higher-resolution ocean simulation and add its barotropic sea level gradient onto the sea level induced by the baroclinic modes. The sea level gradient induced by the barotropic mode is assumed to be in geostrophic balance with the barotropic flow, which is the data source

2.6 Computer Performance

The model concept is to use isopycnal coordinates for the interior ocean. The decision to use the flux form of the equations in order to be able to conserve mass, which is the spatial integral of in situ density, and to conserve momentum results in nonlinear equations. The disadvantageous consequence is that the matrix coefficients need to be updated during each iteration step in order to find the solution of the nonlinear equations. However, ocean models are not made to spend CPU-time only, but should produce good oceanographic results. PIPE has demonstrated it by a number of key publications (see reference list).

The full source code of PIPE contains code sections that are either commonly or alternatively used by PVP-, MPP- or RISC versions of the model. These versions are created using preprocessing directives. System commands like *cpp* or *fpp* extract the model version from the full source code.

The strategy to achieve a high Mflop-rates on all these architectures is as follows:

- 3-d arrays are always dimensioned as *(NZ,NX,NY)* where *NZ* is the vertical index, *NX* is the zonal index and *NY* the meridional index. Since there are no vertical derivatives to compute, the 1st index always runs over its full dimension. The 2nd index typically runs between *2* and *NX-1* which allows to write many statements with vector length *NZ*(NX-2)*. If the 2nd index runs over its full dimension then the 3rd index allows even longer vectors, for instance, of length *NZ*NX*NY*.

- If a 2-d operation is carried out with a 3-d array, these operations always run over *NX* and *NY*. This means that vector elements are not contiguous but have an increment of *NZ*. Therefore, on a CRAY, *NZ* must be odd otherwise bank conflicts occur. In some cases 2-d arrays are extracted out of a 3-d array and scattered back if required.

- Equations that must be solved either iteratively or directly are treated such that direct solutions are obtained by an alternating direction line-relaxation ADLR scheme. It consists of two steps, first a direct solution on the zx-plane and an iteration in y-direction, and a second direct solution on the yz-plane and an iteration in x-direction.

2.6.1 PVP Architecture

If the code is setup for a PVP architecture, iterations are performed such that a parallelization is performed over each second latitude or longitude, respectively, for the ADLR scheme. For instance, a coarse resolution T42 global ocean version of the model achieves 450 Mflops on a CRAY C90, and requires only 6MWords of memory.

Parallelization works best when granularity is coarse, i.e. a lot of work is carried out in parallel regions in order to avoid latency time of frequent initializations of them. Therefore, the code contains micro-tasking directives which tell the compiler that a loop should be executed in parallel. Routines which cause more overhead than speedup due to parallelization as well

as those which are called inside a parallel region are excluded from parallelization. Due to fine granularity of the code the speedup is more moderate than excellent. The code achieves, for instance, a speedup of 3.2 on 4 processors or 5.4 on 8 processors for a coarse-resolution T42-version.

2.6.2 RISC Architecture

The RISC benefits from the optimization for the MPP architecture, because PIPE has been developed for a CRAY T3D, which is based on RISC chips from DEC. There exist many code sections that are alternatively formulated for cache-based processors. The key word used by the *cpp* or *fpp* preprocessor is *RISC*. Basically, RISC processors have a memory bandwidth problem, i.e. memory is slow. Because PIPE is a memory intensive code the slow memory does not allow a high speedup when the code is compiled as parallel code on a workstation. Therefore, it does not make sense to use more than 4 processors.

2.6.3 MPP Architecture

The model is parallalized for MPP computers (Oberhuber and Ketelsen, 1995). In order to execute the model in parallel on many processors like possible on a CRAY T3D, the code is split into four executables (see title page of this report). These executables have the task to

1. initialize the data structure on the server to be able to efficiently supply the client with data, which are read from the files [OCDAT], [FORCES] and [OBNDSEC]

2. perform the model integration by running two executables

   - one running on the server is supplying the client with data if required
   - the other one is running on the client, performs the time integration and returns data to the server for e.g. later analysis

3. clean up the intermediate results and return a data structure identical to a model run on either a PVP or RISC machine, i.e. the file[OCDAT] and all PPSF-files appear in the same style

The data transfer between server and client is carried out using *parallel pipes* (see Fig. 2.2). If the domain is decomposed into *(pe_x,pe_y)* subdomains each executed on one processor, the horizontal grid is separated such that each data piece contains only those latitudes which *(1,pe_y)* is responsible for. The data piece still has full dimension in x-direction. If these data pieces are sent to the according processor *(1,pe_y)*, this processor only needs to further split the data in x-direction, such that these data pieces exactly for with the grid on the processors *(pe_x,pe_y)*. There does not exist severe bottlenecks on the network. Therefore, using the concept of *parallel pipes* allows an efficient data transfer between server and client and does not cause pronounced latency times till communication is finished.

On the MPP each processor holds one extra longitude to the west and to the east, and an extra latitude to the south and to the north. Whenever there is a *call* to one of the boundary routines in the PVP- or RISC-version of the code, these are the locations where data have to exchange in order to provide updated boundary values for computing gradients or divergences. The southernmost latitude on the southernmost processor row and the northernmost latitude on the northernmost processor row are treated differently. Fig. 2.3 shows a sketch for how data are exchanged.

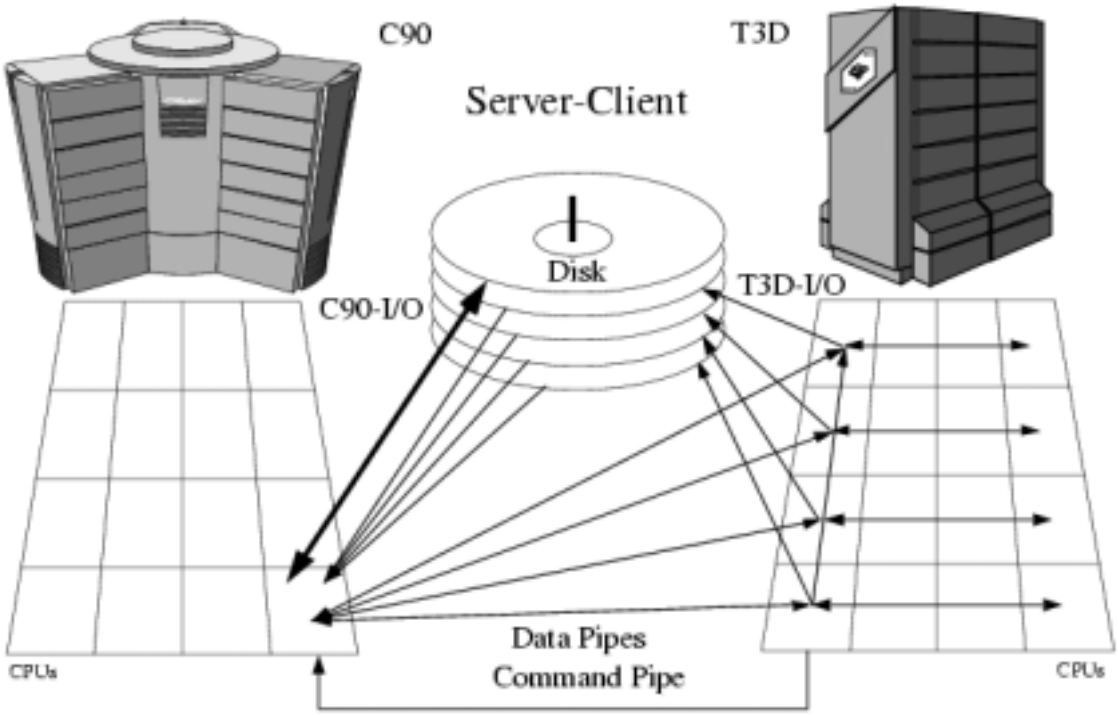

Figure 2.2: Concept of Server-Client Interaction.

In order to improve convergence of various iteratively solved equations the starting latitude is interchanged on subsequent processors in longitude. A further technique to speedup convergence is to shift the data relative to the processor address space. This data shifting is setup such that the amount of sent data is minimized. The latter technique allows to resign on the solution of linear equation systems across processors. Therefore, there exist no sequential operations. Because the data transfer is minimized the code scales well up to many processors. Even in high-resolution experiments there has not been observed a significant reduction of convergence speed. A further trick is to perform the iterations on alternating latitudes with increasing *pe_x* as shown in Fig. 2.4.

In order to speedup the data postprocessing, i.e. the transfer of data to disk, the data are gathered in x-direction on *(1,pe_y)* and then written onto files. A signal is sent to the server which contains the amount of data already written to disk. Then the server reads these distributed data and merges them onto one file in PPSF-format. Data that finally are saved on file [OCDAT] are returned to the servier via the *parallel pipes* and written to disk by the server. Finally, there are no data files left on disk which are needed later.

Due to the small network time and the negligible amount of code sections that have to run sequentially on an MPP, PIPE scales excellently. If overall 64 processors are used even a low-resolution version of PIPE like the T42 with 128×64 grid points on a CRAY T3D achieves a nearly theoretical speedup when comparing with a one-processor version. A higher-resolution version with 512×256 grid points e.g. achieves a speedup of 436 on 512 processors with about a *CRAY C90*-equivalent Mflop-rate of 24. The reasons for this excellent behaviour are the use of an implicit scheme which performs computational intensive work and requires little communication.

Currently, there exists the restriction that the MPP-version of PIPE does not allow for open boundaries, and does not allow to use the tracer module. These pieces of the code will be ported to the MPP architecture upon request.

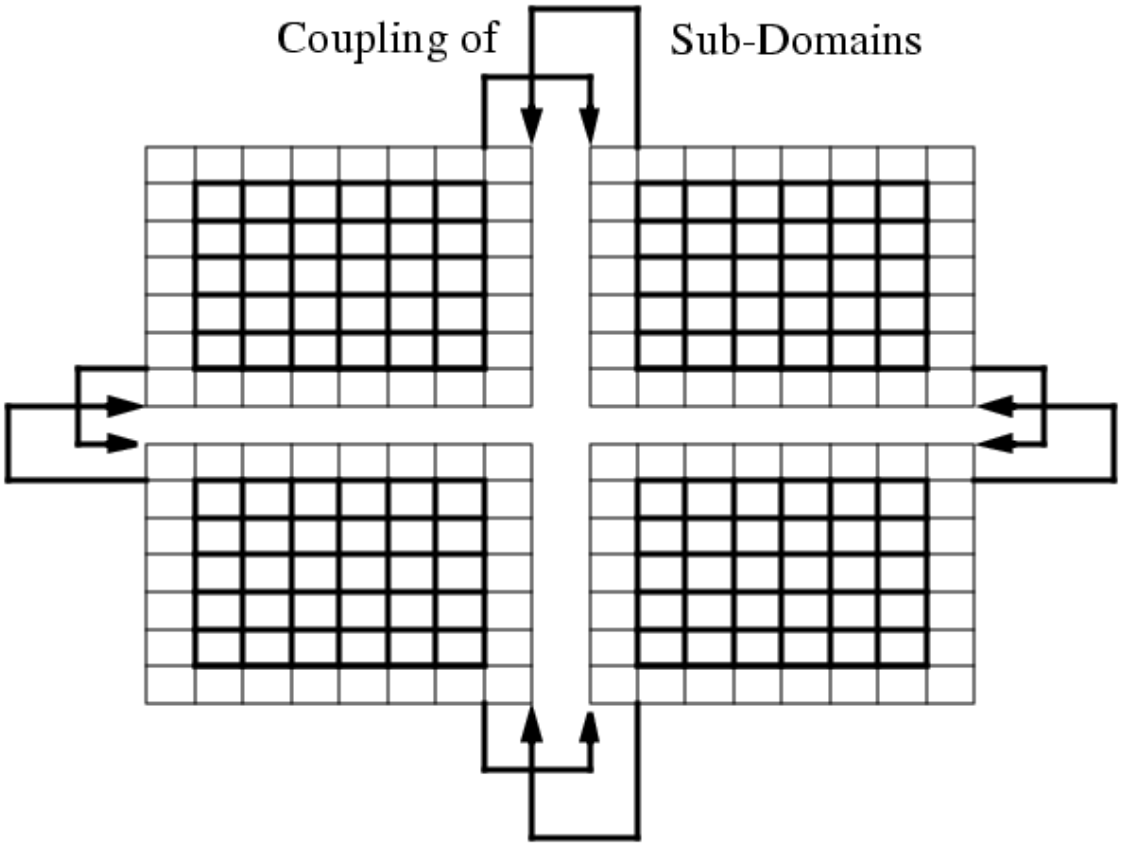

Figure 2.3: Concept for Exchanging Data between Processors.

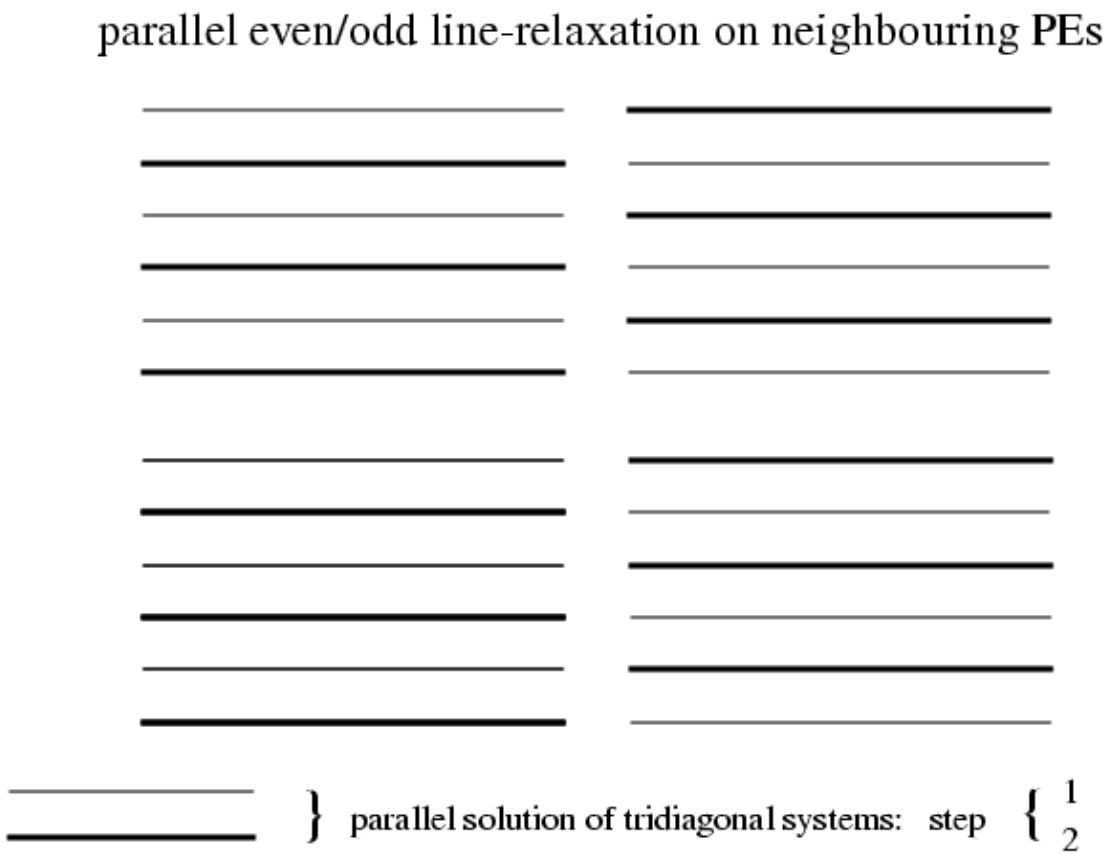

Figure 2.4: Concept for Iterating on Neighbouring Processors.

Part II

The PIPE System Components

Chapter 3

Flow Diagrams

The **PIPE** model has a pretty simple and easy to understand program flow. It to some extend is controlled by switches consisting of *ISW* and an appended number. Either these switches have to do with initialization, postprossing, the physics or the numerical scheme. The driving routines are <OCEINIT>, <OCESTEP>, <OCEPOST> and <OCESTOP>. The computational work is pushed into lower level routines. Thus the driving routines widely consist only of *calls* to these lower level routines. Nevertheless, <OCEINIT> and <OCESTEP> are pretty long.

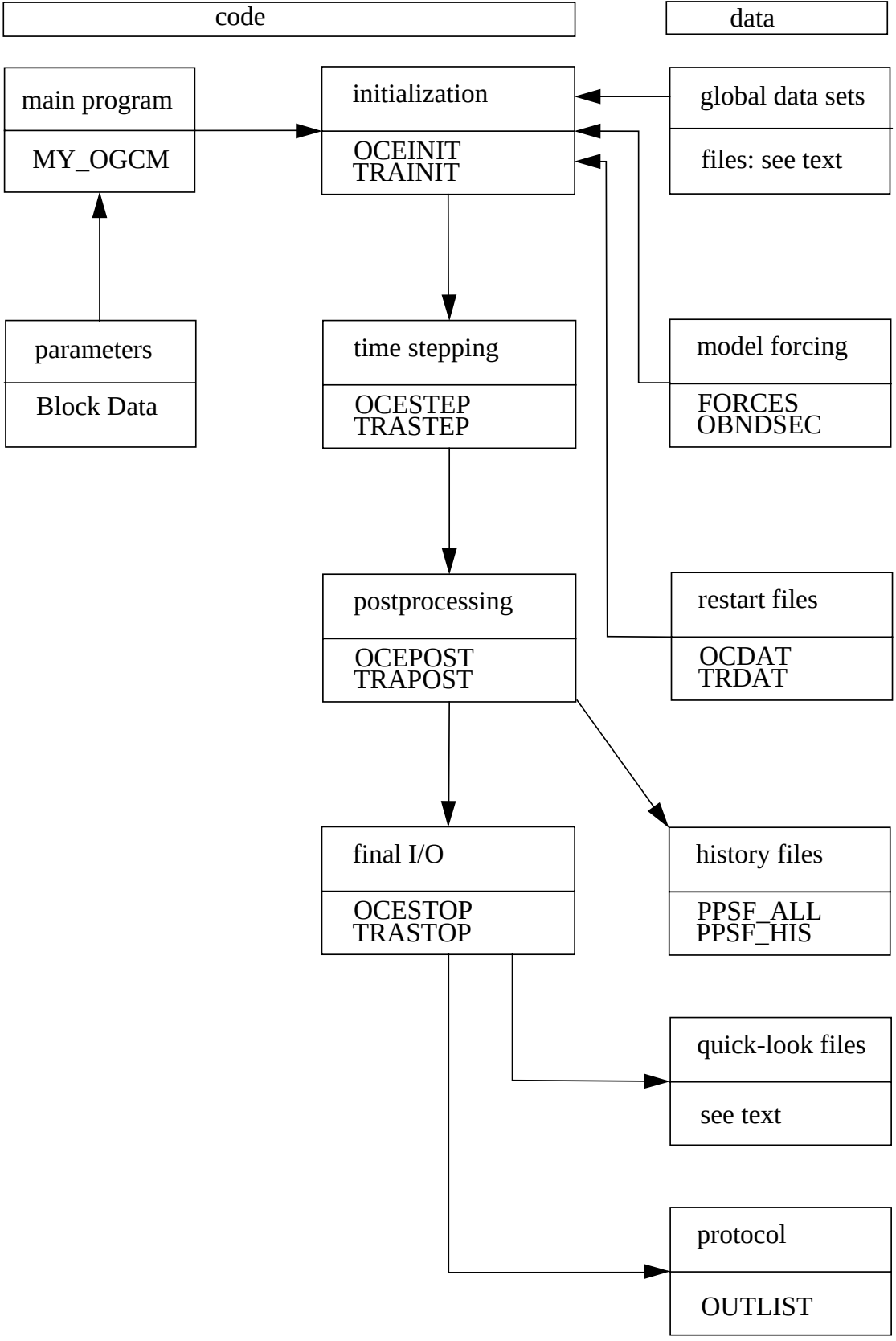

Figure 3.1: Main structure of main program <MY_OGCM> which describes the interaction between the code and the data base.

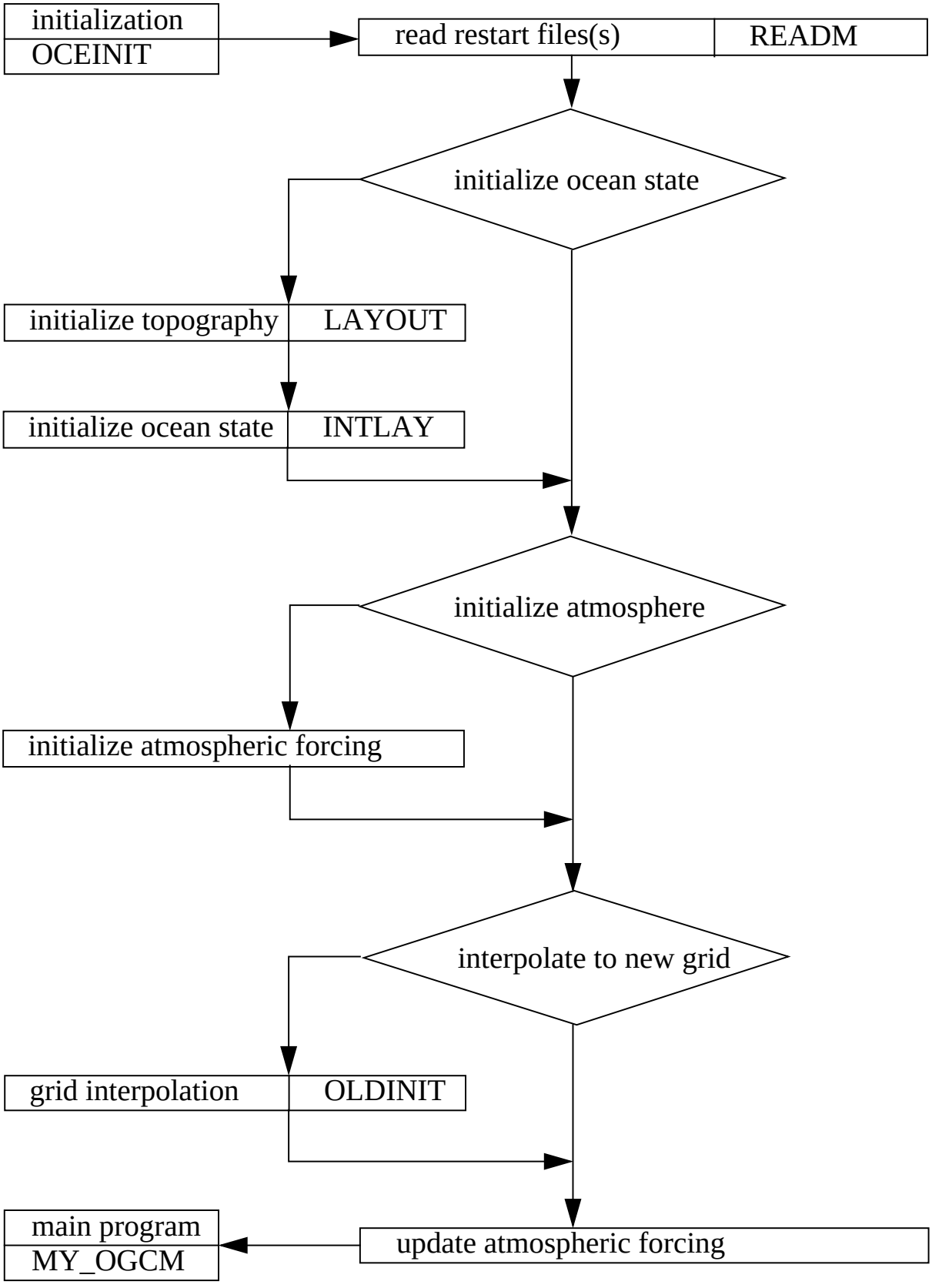

Figure 3.2: Main structure of <OCEINIT> which is the driving routine for initialization.

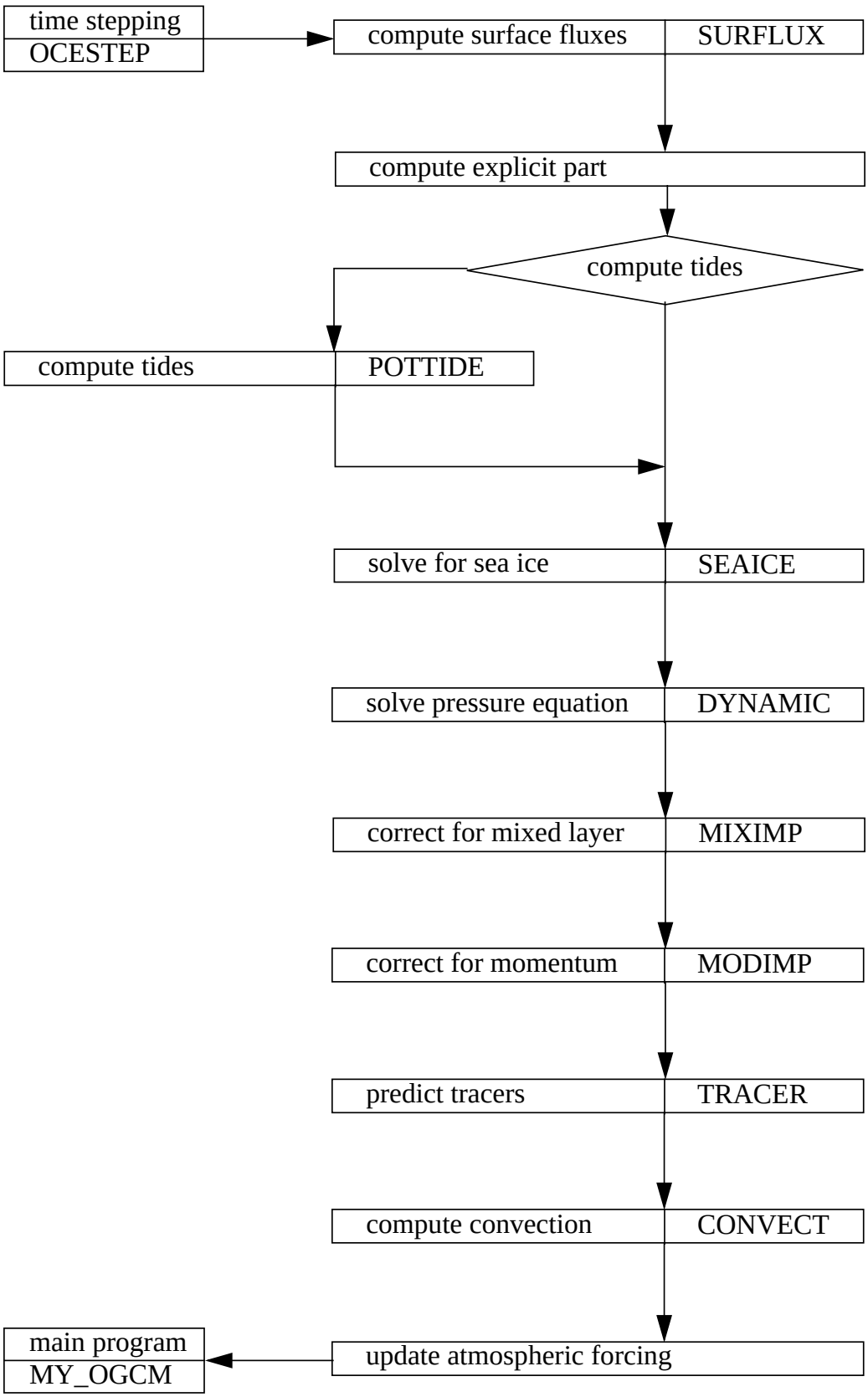

Figure 3.3: Main structure of <OCESTEP> which is the driving routine for carrying out time steps.

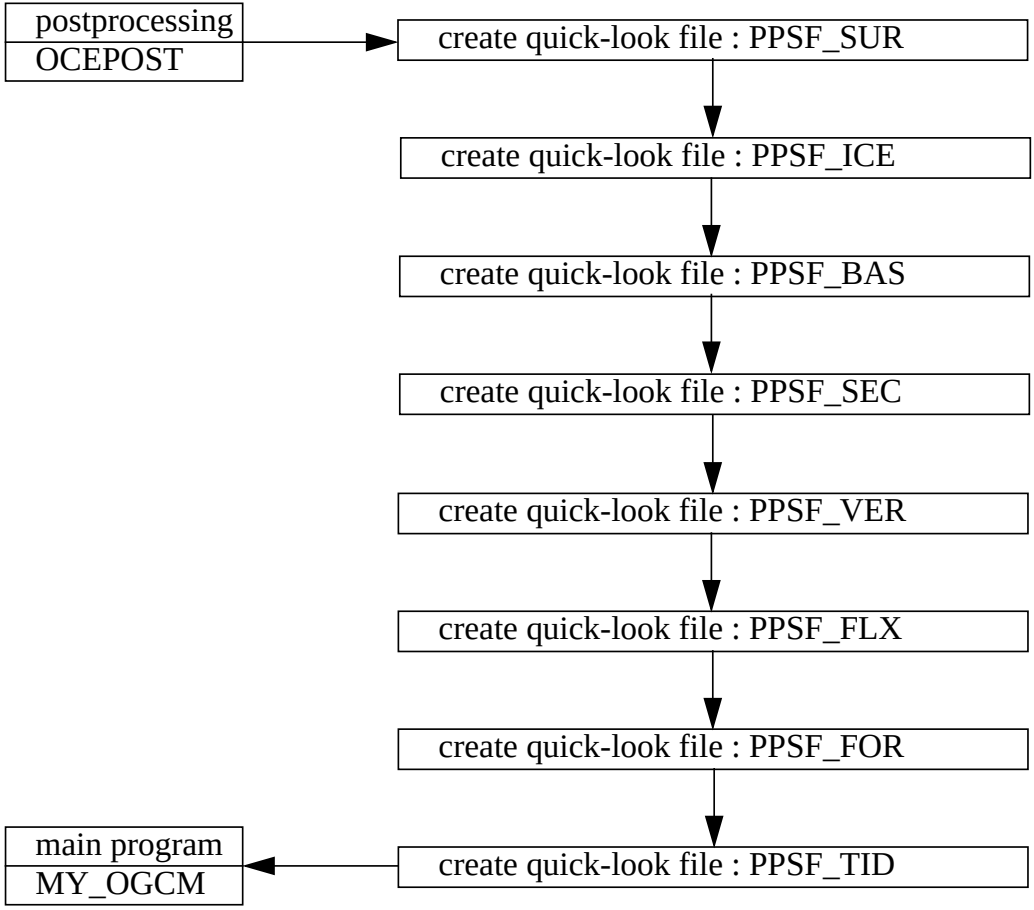

Figure 3.4: Main structure of <OCEPOST> which is the driving routine for data postprocessing and generation of quick-look files.

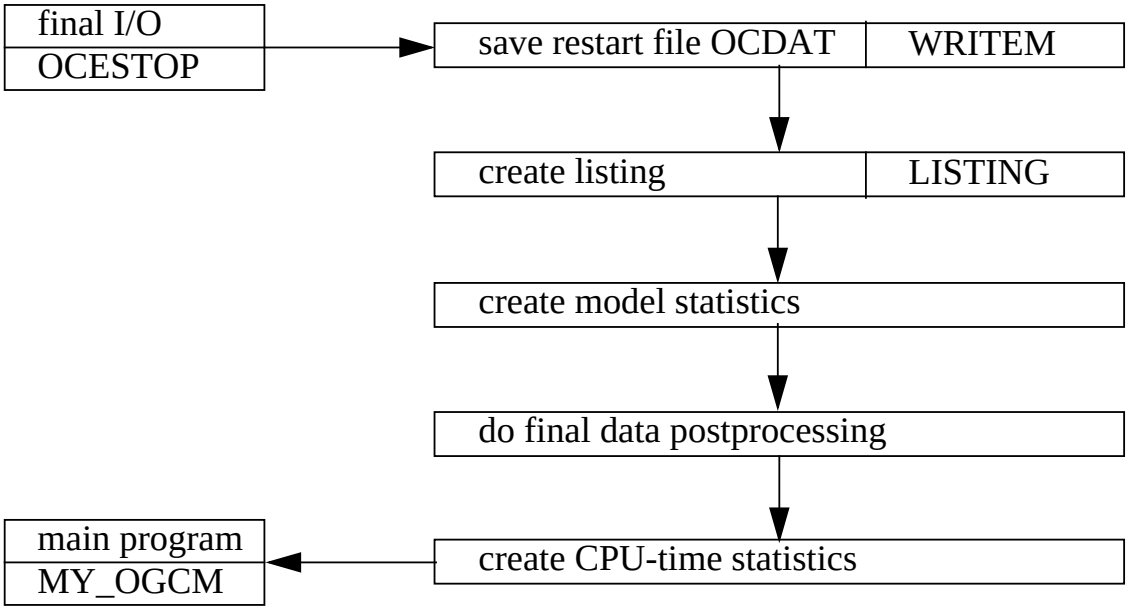

Figure 3.5: Main structure of <OCESTOP> which is the driving routine for final calculations.

Chapter 4

Model Code Description

4.1 General Remarks

The ocean code is divided up into 5 files which are [ocemain.f], [ocestep.f], [oceproc.f], [ocepost.f] and [ocemods.f]. [ocemain.f] consists of a simple main program and all parameter definitions via BLOCK DATA statements. [ocestep.f] contains all routines to carry out the time steps, [oceproc.f] those routines required to initialize the model and the final operations of the model run and [ocepost.f] contains all routines for the postprocessing. [ocemods.f] optionally contains routines of [ocestep.f], [oceproc.f] or [ocepost.f] that require some specific changes and overwrite the original version of those routines while the code is loaded. In general, [ocestep.f], [oceproc.f] and [ocepost.f] contain only the dimensions as specification for the model setup. All parameters are defined in one of the BLOCK DATA statements. This has the advantage that as long as the model dimensions are unchanged, all executable routines have to be compiled only once. Any change to one of the executable routines can be put into [ocemods.f]. Any change of switches or physical parameters are implemented in [ocemain.f] as it contains all BLOCK DATA statements. Therefore, only [ocemain.f] and optionally [ocemods.f] have to be compiled before each model run.

4.2 Contents of [ocemain.f]

4.2.1 Main Program MY_OGCM

The main program consists of four calls to the routines <OCEINIT>, <OCESTEP>, <OCEPOST> and <OCESTOP>. When the model is coupled to an AGCM, the main program also contains calls to the atmosphere and to routines that carry out the communication between both systems. From the view point of the main program, the ocean appears as a black box. Any data exchange with the black box is accomplished with the help of the coupling interface routines. If tracers are computed then the routines <TRAINIT>, <TRASTEP>, <TRAPOST> and <TRASTOP> are called as well.

4.2.2 Definition of Control Parameters

The meaning of the control parameters is listed in each of the BLOCK DATAs. Henceforth, it is distinguished between default values and constants. Default values are marked with a '$\sim$' and should be understood as recommended and well-tuned values for most purposes. If 'any' appears in the column for *Value*, then there is no limitation except when further restrictions are listed. Constants such as the specific heat capacity of water are listed without a '$\sim$' and

never should be changed. Values such as *0,1,2* represent lists of allowed values for switches. Other choices lead to unpredictable results. In most cases routines are referenced; however, if a variable is used at many locations of the code, the COMMON block that encloses the variable is referenced. In general, the code strictly follows the FORTRAN convention that variables starting with *I,J,K,L,M or N* are *INTEGER* variables, while all other variables are *REAL* except for a few cases of *CHARACTER* definitions.

The subsequent tables are not complete. For a complete overview see the descriptions in [ocemain.f], and don't be too lazy to simply look where some variable, switch or whatever appears in the code. Use an ASCII-editor to understand PIPE.

4.2.2.1 Block Data which concern Flow Control

4.2.2.1.1 Block Data SCREW defines the time step and its limits. The maximal number of time steps carried out during one call of <OCESTEP> is *MAXREP*, unless *CALCMAX* initiates an earlier exit (see also table 4.1 in which CPUs denotes seconds CPU-time and MODs denotes seconds of model time). Switches control the coupling interface. *NHEAT*, *NFRESH*, *NSTRS*, *NMIX* and *NICE* are active only if *NINIT=1*. Depending on how the switches *NHEAT* through *NICE* are set, the fluxes are either interpreted as total fields or as anomalies (see table 4.2). Also, there exists the possibility to feed the model with atmospheric raw data which then are used to calculate fluxes in the same style as performed with the data available on the file [FORCES]. Finally, the variables *KHEAT* through *LENAVER* determine whether a bias correction is applied (see table 4.3).

Variable	Value	Unit	Meaning	Reference
MAXREP	any >0		maximal number of time steps	<OCESTEP>
CALCMAX	any >0	CPUs	CPU-time after which model stops	<OCESTEP>
DTMIN	any $\leq$DTMAX	MODs	smallest allowed time step	<OCESTEP>
DTMAX	any >0	MODs	largest allowed time step	<OCESTEP>
DTSTOP	any $\leq$DTMAX	MODs	time step at which model exits	<OCESTEP>
NHOUR	any $\geq$0	MODh	completed hours for model stop	<OCESTEP>
NDAY	any $\geq$0	MODd	completed days for model stop	<OCESTEP>
NMONTH	any $\geq$0	MODm	completed months for model stop	<OCESTEP>
NYEAR	any $\geq$0	MODy	completed years for model stop	<OCESTEP>

Table 4.1: **Time Step Control Parameters**

4.2.2.2 Block Data which concern Physical Parameters

4.2.2.2.1 Block Data HEATFLX defines all variables required to compute heat and fresh water fluxes (see table 4.4).

4.2.2.2.2 Block Data HORIMIX defines parameters required for computing the horizontal diffusion coefficients for scalar and vector quantities (see table 4.5). These parameters MUST be adjusted to the requirements of numerical stability, which depends on the grid resolution. Use Eqns. (1.13) and (1.14) to determine which diffusion coefficients the model computes internally. See also <ISODIFF>, <MODEXP>, <MODIMP> and <TRACTIV>. Note that too large values for *DIADIF0* and *DIADIF1* will result in physical rather than numerical instability ($A^h < 500 m^2 s^{-1}$).

Variable	Value	Default	Meaning	Reference
NINIT	0,1,2	0	switch for coupling interface	/JMOGLOB/
NDRUCK	0,1	0	switch for print	/JMOGLOB/
NHEAT	0,1,2	0	switch for heat coupling	<SURFLUX>
NSOL	0,1,2	0	switch for solar radiation coupling	<SURFLUX>
NFRESH	0,1,2	0	switch for salt coupling	<SURFLUX>
NSTRS	0,1,2	0	switch for stress coupling	<STRESS>
NMIX	0,1,2	0	switch for mixing energy coupling	<SURFLUX>
NICE	0,1,2	0	switch for snow & sea ice coupling	<SURFLUX>
NAUVA	0,1	0	switch for scalar wind	<SURFLUX>
NAUVS	0,1	0	switch for standard deviation	<SURFLUX>
NAIRT	0,1	0	switch for surface air temperature	<SURFLUX>
NAHUM	0,1	0	switch for surface air humidity	<SURFLUX>
NACLD	0,1	0	switch for cloudiness	<SURFLUX>
NOCET	0,1	0	switch for sea surface temperature	<SURFLUX>
NOCES	0,1	0	switch for sea surface salinity	<SURFLUX>
NPRES	0,1	0	switch for surface pressure	<SURFLUX>

Table 4.2: **Coupling Interface Parameters**

Variable	Value	Default	Meaning	Reference
KHEAT	0,1,2	0	switch for heat fluxes	<SURFLUX>
KFRESH	0,1,2	0	switch for fresh water fluxes	<SURFLUX>
KSTRESS	0,1,2	0	switch for wind stress	<STRESS>
LENAVER	any ≥ 0	100	switch for averaging period	<SURFLUX>

Table 4.3: **Parameters for Annual Mean Flux Adjustment**

4.2.2.2.3 Block Data ICE defines parameters for the snow and sea ice model (see table 4.6). Note that *CMELT* and *CFREEZ* are tuning coefficients that allow changes in the rate at which heating causes the opening of leads or heat loss causes ice to form within leads. Note that it is recommended to guarantee *EDDYS=EDDYUV* (see table 4.6).

4.2.2.2.4 Block Data PHYSICS defines numerous parameters. *KTOTAL* defines the model time when initialized. *TEMMEAN* and *SALMEAN* determine the prescribed potential density that is used as potential density coordinate. The layer properties if not initialized using Levitus data are chosen in <CREATIV> by assuming water with salinity of *SALMEAN* and a parabolic temperature distribution where *TEMMEAN* is used as surface temperature and 273.16 *K* is used as bottom temperature (see table 4.17). The potential density coordinates are stored onto *POTSOLL*. Note that *DRAGS* is used to convert the stress data into wind vectors (*ATU* and *ATV*). The same drag coefficient is used to convert the wind vectors back to stress values (see table 4.18). The definition of *TURBLEV* allows to tune the upper ocean thermocline, while *HMIX* is more responsible for the deep ocean (see table 4.13). The mixed layer parameters are well-tuned (see table 4.9).

4.2.2.3 Block Data which concern Initialization

4.2.2.3.1 Block Data LAYOUT defines the model boundaries and the parameters used to create a focus, i.e., an area with enhanced resolution. Furthermore, the Eulerian angles allow one to rotate the model longitudes and latitudes on the sphere. This is useful for rotating a

Variable	Value	Unit	Meaning	Reference
HEATLAT	2500800	$Wskg^{-1}$	heat of fusion	<SURFLUX>
SIGMA	5.67E-8	WsK^{-1}	Stefan Boltzmann constant	<SURFLUX>
SOLAR	1367	Wm^{-2}	solar constant	<SURFLUX>
GAS	287	$m^2K^{-1}s^{-2}$	gas constant of dry air	<SURFLUX>
CPAIR	1005	$Wskg^{-1}K^{-1}$	specific heat capacity of air	<SURFLUX>
SLPRESS	100000.	Nm^{-2}	mean air surface pressure	<SURFLUX>
CONSTS	0.0327		constant for sensible transfer coefficient	<VARDRAG>
CONSTL	0.0346		constant for latent tranfer coefficient	<VARDRAG>
CHARNCK	0.016		Charnock constant	<VARDRAG>
REFHGT	10	m	reference height for transfer coefficient	<VARDRAG>
RKARMAN	0.4		Karman constant	<VARDRAG>
ALBEDO	~0.07		albedo over water	<SURFLUX>
ABSORB	~0.97		emissivity for longwave radiation	<SURFLUX>
ALBICE	~0.50		albedo over snow free dry sea ice	<SURFLUX>
ALBSNOW	~0.60		albedo over dry snow	<SURFLUX>
ALBJUMP	~0.30		albedo jump from dry to wet ice	<SURFLUX>

Table 4.4: **Heat Flux Parameters**

Variable	Value	Unit	Meaning	Reference
DIFFUSS	~1.E-9	m	proportionality coefficient	<ISODIFF>
DIFFUSV	~1000.	s	scaling coefficient for momentum	<MODEXP/IMP>
DIFMINS	~2000.	s	smallest scalar diffusion	<TRACER>
DIFMINV	~5000.	m^2s^{-1}	smallest vector diffusion	<MODEXP/IMP>
DIADIF0	~200	m^2s^{-1}	layer diffusion coefficient	<DIADIFF>
DIADIF1	~1.E-10	m	proportionality coefficient	<DIADIFF>
SLBGRID	~0.2	m	sea level smoothing parameter	<SOLVERX/Y>
BAROINS	~0.	m	barotropic instability parameter	<ISODIFF>

Table 4.5: **Parameters for Horizontal Diffusion**

coordinate pole out of the model area to avoid the coordinate convergence and the resulting deteriorating performance near the pole (see table 4.11). Surface and bottom temperatures and salinities (see table 4.13) define the vertical, i.e. isopycnal, coordinates. Furthermore, eliminating points allow one to remove lakes from the model area. They are disabled if set to zero (see table 4.14). Switches allow for standard problems to add channels or land bridges across topography and/or coastline geometry. They are disabled if set to zero (see table 4.15).

4.2.2.3.2 Block Data ROTATE defines the three Eulerian angles which are responsible for a rotation in longitude, then in latitude and finally in longitude. The rotation becomes active if *ISWITCH=1* (see table 4.12).

4.2.2.4 Block Data which concern Data Postprocessing

4.2.2.4.1 Block Data POSTPRO defines parameters that allow one to control which vertical or horizontal sections appear on the quick-look files. Up to 10 different horizontal or vertical sections can be selected (see table 4.16).

Variable	Value	Unit	Meaning	Reference
TMELT	273.16	K	freezing temperature of fresh water	<TFREEZ>
CPMELT	334000	$Wskg^{-1}$	heat of freezing	<SEAICE>
CPICE	2090	$Wskg^{-1}K^{-1}$	specific heat capacity of ice	<SEAICE>
SALTICE	5.0	gkg^{-1}	salinity of sea ice	<SEAICE>
CMELT	~1.0		tuning coefficient for melting	<SEAICE>
CFREEZ	~2.0		tuning coefficient for freezing	<SEAICE>
EXCENT	~2.0		excentricity of ice rheology	<SEAICE>
PRESICE	~25.0	$Wskg^{-1}$	proportionality for ice pressure	<SEAICE>
PDECAY	~2.0		decay coefficient for ice pressure	<SEAICE>
HNULL	~0.5	m	tuning coefficient for freezing	<SEAICE>
EPSNULL	~1.E-6	ms^{-1}	coefficient that limits ice pressure	<SEAICE>
AGING	~1.E-7	s^{-1}	snow to ice conversion rate	<SEAICE>
CICE	~2.0	$WK^{-1}m^{-1}$	heat conductivity for ice	<SURFLUX>
CSNOW	~0.3	$WK^{-1}m^{-1}$	heat conductivity for snow	<SURFLUX>
EDDYS	~2000.	m^2s^{-1}	scalar diffusion coefficient	<SEAICE>
EDDYUV	~2000.	m^2s^{-1}	vector diffusion coefficient	<SEAICE>
RHOICE	~900	kgm^{-3}	ice density	<SURFLUX>
RHOSNOW	~300	kgm^{-3}	snow density	<SURFLUX>
COMPMAX	~0.99		maximal ice concentration	<SEAICE>

Table 4.6: **Snow and Sea Ice Parameters**

Variable	Value	Unit	Meaning	Reference
KTOTAL	any	s	initial model time	<OCEINIT>
KTSHIFT	1296000	s	length of half a month	<OCEINIT>
SALMEAN	~35.0	g/kg	mean salinity	<DEFORCE>
TEMMEAN	~298.15	K	mean temperature	<DEFORCE>
HTOTAL	~6000	m	reference depth of ocean	<OCEINIT>
UMFANG	40030174	m	earth's circumference	<MAKEXY>

Table 4.7: **Parameters for Coordinate Generation**

4.2.2.4.2 Block Data PRINTER defines switches that control which quantities are written to [OUTLIST]. A focus allows to print every grid point within the some area (see table 4.17).

4.2.2.4.3 Block Data QBDGT defines the vertical layer index *KBDGT* used to bookkeep the budgets of various terms and the frequency at which the budgets are written out. The latter is done each *DTBDGT* seconds model time.

4.2.2.4.4 Block Data DEFINE defines all coefficients required for the equation of state (UNESCO,1981) and for the equation of Bryden (1973) that converts temperature into potential temperature (see Appendix C).

4.2.2.5 Block Data which concern Model Layout

4.2.2.5.1 Block Data SWITCH defines switches that control how the forcing is computed and whether observed or artificial data should be used (see 4.18). The switch *ISW48* is noteworthy. With *ISW48=1* the model computes the running mean fluxes of heat, fresh water

Variable	Value	Unit	Meaning	Reference
CP	4180	$Wskg^{-1}K^{-1}$	specific heat capacity of water	/JMOPAR/
GRAV	9.81	ms^{-2}	earth accelleration	/JMOPAR/
DRAGT	~5.E-6	s^{-1}	time constant for salinity forcing	<SURFLUX>
DRAGS	~1.5E-6		drag coefficient at surface	<STRESS>
DRAGB	~1.E-4		drag coefficient at bottom	<STRESS>

Table 4.8: **Some Physical Parameters**

Variable	Value	Unit	Meaning	Reference
EPSILON	~0.10	m^2s^{-2}	vertical mixing damping coef.	<CROSMIX>
RICRIT	.25		critical Richardson Number	<CROSMIX>
HVMIX	~500	m	tuning coefficient	<CROSMIX>
TURBLEV	~4.E-7	m^3s^{-3}	tuning coefficient	<CROSMIX>

Table 4.9: **Parameters for Diapycnal Diffusion**

and stress, if *NINIT=1*. If the length of the run is a multiple of a year the arrays *HEATQ* will contain the mean heat flux, *PMEQ* the mean fresh water flux, and *TAUXQ* and *TAUYQ* the mean stresses. If the user switches to $ISW48 = 2$ the model will continue to compute the running mean, however, it will also use this mean value for forcing the surface temperature and salinity. After every finished year, these running means are stored onto *HEATM*, *PMEM*, *TAUXM* and *TAUYM*. If the run with *ISW48=2* is long enough and again a multiple of a year, one can start to use *ISW48=3*. In this mode the *HEATM*, *PMEM*, *TAUXM* and *TAUYM* is used as ocean forcing, however, the annual mean is not updated any more. This procedure allows to the model to find that fresh water flux which is required for a stable integration but avoids the negative feedback due to the Newton relaxation. Other switches control the physical content (see table 4.19), select numerical schemes (see 4.20), control the output (see table 4.21) and select the data source used to create [FORCES] (see table 4.22). The usual choice is *ISW32=2* which means that the potential vorticity conserving scheme is taken instead of the Crowley scheme for momentum (*ISW32=1*). Note that it is possible to choose an open pole for the sea ice model only with *ISW28=1* while *ISW28=2* means that an open pole is used for sea ice and ocean. This choice is meaningful, since currently the sea ice model is numerically more stable than the ocean model with open pole boundary conditions. *ISW42* determines the initialization mode. If *ISW42=0* then the model tries to initialize [OCDAT] and [FORCES] when an empty [OCDAT] is detected. If *ISW42=1* then the model creates [FORCES] independent of whether an [OCDAT] exists. In case of *ISW42=2* the model initializes the ocean state by delivering a new [OCDAT] but it assumes that a [FORCES] is already available. These choices are useful during the model setup to save CPU-time.

4.2.2.6 Block Data which concern Numerics

4.2.2.6.1 Block Data TUNING contains the weighting coefficients that define how the old and new time level is weighted within the implicit algorithm. Note that these parameters must be between 0.5 and 1.0 to ensure stability (see table 4.23). The over- and underrelaxation coefficients are of empirical origin. *NITMIN* defines the number of iterations that are used under all conditions. In fact twice the number of iterations is taken everywhere. *NITMAX* defines the iteration count that enables the model to detect insufficient convergence of one of the equations and, dependent on *ISW35*, causes an error exit (see table 4.24). Threshold parameters finally allow one to limit certain variables mainly for stability purposes (see table 4.25).

Variable	Value	Unit	Meaning	Reference
CMIX0	~0	$m^2 s^{-3}$	linear damping term of ML	<MIXEXP>
CMIX1	~1.E6	m	constant TKE decay length scale	<MIXEXP>
CMIX2	0.25		critical Richardson number	<MIXEXP>
CMIX3	~2.5		Ekman dissipation for TKE	<MIXEXP>
CMIX4	~1.25		wave energy conversion factor	<MIXEXP>
CMIX5	~0		Garwood-term tuning coefficient	<MIXEXP>
CMIX6	~1.E6	m	constant buoyancy decay length scale	<MIXEXP>
CMIX7	~0.5		Ekman dissipation for buoyancy flux	<MIXEXP>
CMIX8	~0.01	$m^2 s^{-2}$	threshold for hg'	<MIXEXP>
CMIX9	~1.0		tuning coefficient for buoyancy flux	<MIXEXP>
DETIME	~0	s	time scale for detrainment	<MIXEXP>
TURBEN	~0	$m^3 s^{-3}$	tuning parameter	<MIXEXP>
TURBID	~15	m	penetration depth of solar radiation	<DECAY>
TURBREL	~0.42		transmission coefficient	<DECAY>
PENET	~20	$m^3 kg^{-1}$	parameter for salt ejection	<BUBBLES>

Table 4.10: **Parameters for Mixed Layer**

Variable	Value	Default	Unit	Meaning	Reference
XGL	any	0	deg	left margin of model domain	<MAKEXY>
XGR	any	360	deg	right margin of model domain	<MAKEXY>
YGU	any	-90	deg	lower margin of model domain	<MAKEXY>
YGO	any	90	deg	upper margin of model domain	<MAKEXY>
IDENS	any	0		number of extra grid points in x-dir.	<MAKEXY>
JDENS	any	0		number of extra grid points in y-dir.	<MAKEXY>
ICENTER	any	0		x-index location of focus centre	<MAKEXY>
JCENTER	any	0		y-index location of focus centre	<MAKEXY>
IBAND	any	0		index width used for focus in x-dir.	<MAKEXY>
JBAND	any	0		index width used for focus in y-dir.	<MAKEXY>
XCOM	any	1		refinement factor in x-direction	<MAKEXY>
YCOM	any	1		refinement factor in y-direction	<MAKEXY>
IORIGIN	any	1		x-index origin of focus	<MAKEXY>
JORIGIN	any	1		y-index origin of focus	<MAKEXY>
LEAVOUT	any ≥ 0	0		number of ignored latitudes	<MAKEXY>

Table 4.11: **Parameters for Horizontal Grid Definition**

Variable	Value	Unit	Meaning	Reference
ALPHA	any	deg	first longitudinal rotation angle	<ROMATIN>
BETA	any	deg	second latitudinal rotation angle	<ROMATIN>
GAMMA	any	deg	third longitudinal rotation angle	<ROMATIN>
ISWITCH	0,1		switch for activating rotation	<ROMATIN>

Table 4.12: **Threshold Variables**

Variable	Value	Default	Unit	Meaning	Reference
TEMPTOP	any	299.16	Kelvin	temperature for top layer	<MAKEVER>
TEMPBOT	any	273.16	Kelvin	temperature for bottom layer	<MAKEVER>
SALTTOP	any	35.00	psu	salinity for top layer	<MAKEVER>
SALTBOT	any	34.90	psu	salinity for bottom layer	<MAKEVER>

Table 4.13: **Parameters for Vertical Grid Definition**

Variable	Value	Default	Meaning	Reference
KILL1I	any index	0	x-index for eliminating point #1	<ELIMIN>
KILL1J	any index	0	y-index for eliminating point #1	<ELIMIN>
KILL2I	any index	0	x-index for eliminating point #2	<ELIMIN>
KILL2J	any index	0	y-index for eliminating point #2	<ELIMIN>
KILL3I	any index	0	x-index for eliminating point #3	<ELIMIN>
KILL3J	any index	0	y-index for eliminating point #3	<ELIMIN>
KILL4I	any index	0	x-index for eliminating point #4	<ELIMIN>
KILL4J	any index	0	y-index for eliminating point #4	<ELIMIN>
KILL5I	any index	0	x-index for eliminating point #5	<ELIMIN>
KILL5J	any index	0	y-index for eliminating point #5	<ELIMIN>
KILL6I	any index	0	x-index for eliminating point #6	<ELIMIN>
KILL6J	any index	0	y-index for eliminating point #6	<ELIMIN>

Table 4.14: **Definition of Eliminating Points**

Variable	Value	Default	Meaning	Reference
KSW1	0,1	0	switch for Panama bridge	<GRITUNE>
KSW2	0,1	0	switch for Florida Peninsula	<GRITUNE>
KSW3	0,1,2	0	switch for Thule land bridge or channels	<GRITUNE>
KSW4	0,1,2	0	switch for Hudson Bay closed or channel	<GRITUNE>
KSW5	0,1	0	switch for cut off of Ochotskic Sea	<GRITUNE>
KSW6	0,1	0	switch for bridge Spitzbergen-Norway	<GRITUNE>
KSW7	0,1,2	0	switch for bridge Alaska/Russia or chan.	<GRITUNE>
KSW8	0,1,2	0	switch for bridge Gibraltar or channel	<GRITUNE>
KSW9	0,1	0	switch for Weddell Sea Peninsula	<GRITUNE>
KSW10	0,1,2	0	switch for Cuba+Thaiti or wipeout	<GRITUNE>
KSW11	0,1	0	switch for bridge Malaysia and Indonesia	<GRITUNE>
KSW12	0,1	0	switch for channel Met. Sea and Black Sea	<GRITUNE>
KSW13	0,1	0	switch for bridge: Australia-Indonesia	<GRITUNE>
KSW14	0,1	0	switch for bridge: Greenland-Canada	<GRITUNE>
KSW15	0,1	0	switch for island: Novaja Semlja	<GRITUNE>
KSW16	0,1	0	switch for island: Severnaja Semlja	<GRITUNE>
KSW17	0,1	0	switch for island: Japan	<GRITUNE>
KSW18	0,1	0	switch for island: New Zealand	<GRITUNE>
KSW19	0,1,2	0	switch for bridge: Baltic-North Sea	<GRITUNE>
KSW20	0,1,2	0	switch for bridge: Red Sea and Indic	<GRITUNE>

Table 4.15: **Grid Tuning Switches**

Variable	Value	Default	Meaning	Reference
ISWPOST(1)	0,1	1	generate file [PPSF_SUR]	<OCEPOST>
ISWPOST(2)	0,1	1	generate file [PPSF_ICE]	<OCEPOST>
ISWPOST(3)	0,1	1	generate file [PPSF_BAS]	<OCEPOST>
ISWPOST(4)	0,1	1	generate file [PPSF_SEC]	<OCEPOST>
ISWPOST(5)	0,1	1	generate file [PPSF_VER]	<OCEPOST>
ISWPOST(6)	0,1	1	generate file [PPSF_FLX]	<OCEPOST>
ISWPOST(7)	0,1	1	generate file [PPSF_FOR]	<OCEPOST>
ISWPOST(8)	0,1	1	generate file [PPSF_INI]	<OCEPOST>
ISWPOST(9)	0,1	1	generate file [PPSF_TID]	<OCEPOST>
ISEC(10)	$1 \leq$ ISEC $\leq$ NX	0	x-indices for cross sections	<OCEPOST>
JSEC(10)	$1 \leq$ JSEC $\leq$ NY	0	y-indices for cross sections	<OCEPOST>
VSEC(10)	any in m	0	depth of horizontal sections	<OCEPOST>
KSEC	$1 \leq$ KSEC $\leq$ NZ	0	z-index for layer sections	<OCEPOST>
AVERTIM	any ≥ 1	2592000	averaging period	<OCEPOST>

Table 4.16: **Parameters for Quick-Look System**

Variable	Value	Default	Meaning	Reference
KSW	0,1	0	focus used for listing	<DRUCK..>
MAXIM	$1 \leq$ MAXIM $\leq$ NZ	1	number of printed layer	<LISTING>
ILEFT	any $<$ IRIGHT	1	left index used for focus	<LISTING>
IRIGHT	any $>$ ILEFT	1	right index used for focus	<LISTING>
JLOW	any $<$ JUPP	1	lower index used for focus	<LISTING>
JUPP	any $>$ JLOW	1	upper index used for focus	<LISTING>
JSW1	0,1	0	horizontal fluxes	<LISTING>
JSW2	0,1	0	surface and interfaces	<LISTING>
JSW3	0,1	0	layer thickness	<LISTING>
JSW4	0,1	1	temperature	<LISTING>
JSW5	0,1	1	salinity	<LISTING>
JSW9	0,1	0	diffusion coefficient	<LISTING>
JSW11	0,1	0	potential vorticity	<LISTING>
JSW12	0,1	1	velocity	<LISTING>
JSW13	0,1	0	potential density	<LISTING>
JSW14	0,1	1	SST error	<LISTING>
JSW15	0,1	1	sea ice and snow	<LISTING>
JSW16	0,1	0	topography	<LISTING>
JSW17	0,1	1	index field	<LISTING>

Table 4.17: **Switches for Listing**

Variable	Value	Default	Meaning	Reference
ISW1	0,1	0	switch from artificial to observed forcing	<OCEINIT>
ISW5	0,1	0	switch from artificial to observed initial state	<OCEINIT>
ISW8	0,1	0	switch from artificial to observed topography	<OCEINIT>
ISW13	0,1	0	switch from linear to parameterized heat fluxes	<SURFLUX>
ISW15	0,1	0	switch from closed, cyclic or open boundaries	<OCEINIT>
ISW18	0,1	0	switch from linear to parameterized salt fluxes	<SURFLUX>
ISW19	0,1	0	switch from diurnal cycle	<SURFLUX>
ISW28	0,1,2	0	switch for closed/open North Pole	<BND...>
ISW41	0,1	0	switch from artificial to observed atmosphere	<OCEINIT>
ISW48	0,1,2,3	0	switch for mode of heat and fresh water forcing	<OCESTEP>

Table 4.18: **Switches for Type of Forcing**

Variable	Value	Default	Meaning	Reference
ISW2	0,1	1	switch for heat fluxes	<OCESTEP>
ISW3	0,1	1	switch for layer diffusion	<OCESTEP>
ISW4	0,1	1	switch for mixed layer physics	<OCESTEP>
ISW6	0,1	1	switch for river runoff when EBM absent	<OCESTEP>
ISW7	0,1,2	1	switch for Coriolis force	<OCEINIT>
ISW9	0,1	1	switch for spherical coordinates	<OCEINIT>
ISW19	0,1	1	switch for diurnal cycle	<OCEINIT>
ISW10	0,1	1	switch for updating isopycnal coordinates	<OCEINIT>
ISW14	0,1	1	switch for convection	<OCESTEP>
ISW22	0,1	1	switch for forcing through sea level pressure	<OCESTEP>
ISW23	0,1,2	1	switch for tracer model	<EBM>
ISW24	0,1	1	switch for EBM	<EBM>
ISW25	0,1	1	switch for salt calculation	<OCESTEP>
ISW26	0,1	1	switch for tides	<OCESTEP>
ISW33	0,1	0	switch for Greg Holloway's Neptun effect	<OCESTEP>
ISW34	0,1	1	switch for vertical diffusion	<OCESTEP>
ISW39	0,1,2	1	switch for mass budget $->$ sea level	<OCESTEP>
ISW40	0,1,2,3	1	switch for mass conservation	<OCESTEP>
ISW44	0,1	1	switch for sea ice calculation	<OCESTEP>

Table 4.19: **Switches for Model Physics**

Variable	Value	Default	Meaning	Reference
ISW21	0,1	1	switch for ADLR scheme	<OCESTEP>
ISW27	0,1	1	switch for diapycnal pressure gradient	<OCESTEP>
ISW32	0,1,2	2	switch for choice of transport scheme	<MODEXP>
ISW37	0,1,2	0	switch for momentum diffusion	<OCESTEP>

Table 4.20: **Switches for Numerical Schemes**

Variable	Value	Default	Meaning	Reference
ISW11	0,1	0	switch for exit after topography generation	<OCEINIT>
ISW17	any$\geq$ 0	0	switch for generation of history file	<OCESTEP>
ISW21	0,1	1	switch for generation of vertical velocity	<OCESTOP>
ISW31	0,1	0	switch from binary to formatted [OCDAT]	<WRITEM>
ISW35	0,1	1	switch for emergency exit	<OCESTEP>
ISW46	0,1	0	switch for output of monthly data	<OCESTEP>

Table 4.21: **Switches for Output**

Variable	Value	Default	Meaning	Reference
ISW12	0,1	0	restart from other [OCDAT]	<OCEINIT>
ISW16	0,1	1	H&R or ECMWF stresses	<OCEINIT>
ISW30	0,1,2	1	COADS, ECMWF or Levitus SST data	<OCEINIT>
ISW38	0,1	1	Scripps or NOAA topography	<OCEINIT>
ISW42	0,1,2	0	initialization mode for [FORCES] and [OBNDSEC]	<OCEINIT>
ISW43	0,1	1	COADS or ECMWF air temperature	<OCEINIT>
ISW45	0,1,2	0	import/export of land-sea mask	<OCEINIT>
ISW47	0,1	1	COADS or ECMWF winds	<OCEINIT>

Table 4.22: **Switches for Data Sources**

Variable	Value	Meaning	Reference
ALPHA	$\sim$0.75	time weight for flux divergence	/JMOBACK/
BETA	$\sim$0.75	time weight for pressure gradient	/JMOBACK/
GAMMA	$\sim$0.75	time weight for Coriolis force	/JMOBACK/
DELTA	$\sim$0.75	time weight for momentum advection	/JMOBACK/
ETA	$\sim$0.75	time weight for momentum diffusion	/JMOBACK/

Table 4.23: **Time Weights for Implicit Scheme**

Variable	Value	Unit	Meaning	Reference
OVER	$\sim$1.25		overrelaxation coefficient	/JMOTUNE/
UNDER1	$\sim$1.00		underrelaxation coefficient except for ice	/JMOTUNE/
UNDER2	$\sim$1.00		underrelaxation coefficient for ice	/JMOTUNE/
NITMIN	$\sim$3		minimal number of iterations	/JMOITER/
NITMAX	$\sim$10		maximal number of iterations	/JMOITER/
ACCURU	$\sim$0.1	$kgs^{-1}m^{-1}$	required accuracy for momentum	<MODIMP>
ACCURH	$\sim$0.01	m	required accuracy for thickness	<SOLVER>
ACCURT	$\sim$0.00002	K	required accuracy for temperature	<TRACER>
ACCURS	$\sim$0.00001	gkg^{-1}	required accuracy for salinity	<TRACER>
ACCURF	$\sim$0.00001		required accuracy for tracer	<TRACER>

Table 4.24: **Numerical Tuning Variables**

Variable	Value	Unit	Meaning	Reference
STABMIN	∼0.01	kgm^{-3}	threshold for density difference	<STABIL>
STABDIF	∼0.005	ms^{-2}	threshold for g' in mixed layer	<MIXEXP>
HMIXMIN	∼5	m	threshold for mixed layer depth	<MIXEXP>
UVMAX	∼5	ms^{-1}	threshold for velocity	<LIMVEL>
TOPOMIN	∼20	m	threshold for ocean depth	<OCEINIT>

Table 4.25: **Threshold Variables**

4.3 The Ocean Model: [ocestep.f]

4.3.1 Interior Ocean Model Subroutines

- <**OCESTEP**> is the driving routine to carry out the time step. At each call of it *MAXREP* time steps are carried out unless other thresholds as defined by the maximal CPU-time *CALCMAX* or the final model time *NYEAR* etc. are reached. It can be called several times from the main program.

- <**CONVECT**> computes the exchange of momentum, heat and salt concentration between two adjacent layers in case of unstable stratification. In the first part the mixed layer exchanges information with the next adjacent layer and in the second part two layers in the deep ocean exchange their properties. In the latter case the coordinates are rearranged if the coordinate error exceeds a certain threshold.

- <**CORFORC**> computes the Coriolis force and adds its contribution to the momentum change to the arrays *RESTU* and *RESTV*.
  Entry <**RALEIGH**> adds on a Raleigh friction term. In the case of open boundaries this routine might be used to define a swamp for momentum. A *stop*-statement remembers the user to adjust this entry to the user's application.

- <**CROSMIX**> determines the upward and downward mass flux due to diapycnal mixing. The result is added to *VERTUP* and *VERTDN*. These arrays are later used in <TRACTIV> to compute the related changes in temperature and salinity concentration. The contribution to the momentum budget is added to *RESTU* and *RESTV*.

- <**ISODIFF**> computes the diffusion coefficient from the shear flow, with a specified threshold minimum value.

- <**DIADIFF**> computes the diffusion coefficient for the layer diffusion.

- <**DIVADD**> adds the flux divergence of the continuity equation to *RESTH*.

- <**DIVFOR**> computes parts of the right hand side of the wave Eqn. (2.40) and adds the contribution to *RESTH*. Compare with Eqn. (2.37) and (2.38).

- <**FLUXES**> computes the new fluxes from the pressure gradient at the new time level and the already determined right hand side of the momentum equations that are not treated implicitly. Compare with Eqn. (2.35) and (2.36).

- <**VELOC**> computes the velocities from fluxes.

- <**LIMVEL**> limits velocities to *UVMAX*.
  Entry <**LIMFLX**> limits the according mass flux.

- <**SETGRADX**> computes (as differences on two latitudes) the part of the pressure gradient that results from layer thickness gradients. The differences are used to compute the right hand side of the linear equation system of the wave equation. It also determines the weighting coefficients that are required to compute the matrix for the wave equation. This routine is used by <SOLVERX> only.

- <**SETGRADY**> computes (as differences on two latitudes) the part of the pressure gradient that results from layer thickness gradients. The differences are used to compute the right hand side of the linear equation system of the wave equation. It also determines the weighting coefficients that are required to compute the matrix for the wave equation. This routine is used by <SOLVERY> only.

- <**GRDADD**> computes explicitly that part of the pressure gradient that is due to layer thickness and density gradients. The result is added to *RESTU* and *RESTV*.

- <**LAPRADD**> computes the contribution to the right hand side of the wave equation that contains the gradients of density as a Laplacian operator. The routine updates its contribution for the new layer thicknesses during the iteration.

- <**VORABS**> updates the potential vorticity.

- <**LAYDIFF**> determines explicitly the layer thickness changes that result from the layer thickness diffusion and adds the result to *RESTH*. The same term is treated implicitly as part of the matrix within <SOLVERX> and <SOLVERY>.

- <**LAYFLUX**> corrects the horizontal transports by those contributions that result from layer thickness diffusion. This routine is needed to provide horizontal transport fields UH and VH containing diffusive mass fluxes for later postprocessing.

- <**MODEXP**> determines the momentum advection and diffusion explicitly and adds their contribution to *RESTU* and *RESTV*.
  Entry <**MODIMP**> performs the same calculation, however, in implicit formulation. It uses the line-relaxation method to treat these terms directly in x- and z-direction and iterates in y-direction only or optionally solves these terms also in y- and z-direction and iterates in x-direction.

- <**MOLVERX**> performs an direct solution in x-direction and an iterative one in y-direction. It is called by <MODIMP>.

- <**MOLVERY**> performs an direct solution in y-direction and an iterative one in x-direction. It is called by <MODIMP>.

- <**DYNAMIC**> organizes the solution of the wave equation, computes the contribution of expansion, updates potential vorticity and calls <SOLVERX> and <SOLVERY> dependent on whether the alternating direction line relaxation scheme is used or the solution is performed directly only in xz-direction

- <**SOLVERX**> is one of the major routines of the entire model. It performs one complete iteration of the wave equation and yields the layer thickness as the result. It uses a line-relaxation method that solves a linear equation system directly on xz-slices and iterates in the y-direction.

- <**SOLVERY**> is one of the major routines of the entire model. It performs one complete iteration of the wave equation and yields the layer thickness as the result. It uses a line-relaxation method that solves a linear equation system directly on yz-slices and iterates in the x-direction.

- <**STRESS**> computes the contribution of the surface stress, the stresses between the layers and the bottom stress to the right hand side of the momentum equation which is added to *RESTU* and *RESTV*.

- <**VTOTAU**> converts the vector surface wind to stress.
  Entry <**TAUTOV**> performs the inverse operation.

- <**TRHTOU**> computes mean layer thickness on velocity grid points with additional weights. The output variable *HQLIN* is a key variable needed for the flux form of the dynamical equations.

- <**STABIL**> limits the stability to a marginally positive value.

- <**HTOZETA**> converts layer thickness to interface height above reference level.
  Entry <**ZETATOH**> performs the inverse operation.

4.3.2 Mixed Layer Model Subroutines

- <**MIXEXP**> computes the mixed layer physics explicitly forward. It computes the entrainment and detrainment rates diagnostically and delivers changes of layer thickness and momentum to <OCESTEP>.
  Entry <**MIXIMP**> performs the same calculations. It treats the mixed layer implicitly by an iterative procedure to find the new layer thickness for the corrector step.

- <**EQUILIB**> computes the Monin-Obukhov length by Newton's method to find zeros. Values are saved on *HEQU* and used either in <MIXEXP> or <MIXIMP>.

- <**MECHAN**> computes the turbulent kinetic energy *ENERGY*, the energy source due to the Garwood-term *TAUX* and the inverse Ekman length scale *EKMAN*.

- <**SHEAR**> computes the shear of the flow between mixed layer and the adjacent layer. The result is used to calculate the shear instability.

4.3.3 Active Tracer Subroutines

- <**TRACER**> organizes the time step for temperature and salinity.

- <**TRACTIV**> computes the new distribution of temperature, salinity or tracer concentration. Since the underlying equations for all these quantities are identical and only differ in their forcing, the routine is used for all different type of tracers. The only difference is how the forcing is computed for the various tracers. This is organized within <TRACER>.

- <**BUBBLES**> computes how the ejected salt from freezing ice is distributed among the near surface layers. The result is used to predict the new salinity via <TRACER>.

- <**DECAY**> determines the heat flux into each layer that is the result of the penetrating solar radiation. The result is used when <TRACTIV> is called to determine the new temperature distribution.

4.3.4 The Sea Ice Model

- <**SEAICE**> computes the ice flow as well as the snow and ice cover from forcings due to stress, heat flux and fresh water flux. Note that the heat flux stored in *QFLUX* is already adjusted to the ice conditions by <SURFLUX>. The routine subtracts from *QFLUX* that part which is used for melting or freezing. The same applies for the salt flux *SFLUX*.

- <**DIADICE**> computes the thermodynamic forcing of the sea ice model.

- <**ENSURE**> makes ice extent consistent with SST distribution.

- <**PFREEZ**> computes the dependence of the freezing point of sea water on salinity.

4.3.5 Atmospheric Energy Balance Model

- <**EBM**> computes the horizontal transport of temperature and moisture. It also predicts the soil temperature over land, and below a glacier if existing. The radiation calculation and that of the turbulent fluxes is performed in <SURFLUX>.

- <**RIVERS**> performs the river runoff on land.

4.3.6 Tide Model

- <**POTTIDE**> is the driver for the tide model. It organizes the timestepping, i.e. the subcycling of the tide model. It also bookkeeps the residual currents and performs the data postprocessing.

- <**TIDES**> predicts the anomalous tidal flow and sea surface elevation.

- <**EQUIPOT**> computes the gravitational forcing of the tide model.

4.3.7 The Equation of State

- <**EXPANDT**> computes the expansion coefficients with respect to temperature. The UNESCO formula is differentiated analytically.
  Entry <**EXPANDS**> performs the same calculation for salinity.
  Entry <**EXPANDP**> performs the same calculation for in situ pressure.
  Entry <**EXPANDT**> performs the same calculation for potential temperature.

- <**NEWDENS**> computes the in situ density, the potential density and the in situ pressure from potential temperature and salinity.

- <**REPRESS**> computes the in situ pressure.

- <**STATETR**> computes the in situ density from temperature, salinity and in situ pressure for one layer only.
  Entry <**STATETP**> performs the same calculation for potential density.

- <**STATEVR**> computes the in situ density from temperature, salinity and in situ pressure for all layers.
  Entry <**STATEVP**> performs the same calculation for potential density.

- <**THERMO**> computes the layer thickness changes at a given mass content, temperature and salinity. This routine is required to update the expansion effects while solving the wave equation.

- <**TTOTET**> computes the potential temperature from temperature for one layer.

- <**TETATOT**> computes the temperature from potential temperature by inverting the formula of Bryden for all layers.

4.3.8 Model Forcing

- <**SURFLUX**> computes the heat fluxes, fresh water fluxes and buoyancy fluxes from the atmospheric data, the model SST and model sea surface salinity. Depending on the switches either simple Newtonian type parameterizations or more detailed one are taken. It also contains the calculation of the fluxes through ice and snow and computes by iteration the snow skin temperature as well as the snow-ice interface temperature as prognostic variable. Note that this routine is responsible for predicting the thermodynamics of the snow & sea ice model while <SEAICE> is responsible for the dynamical part. < SURFLUX> also computes radiative and turbulent fluxes for the EBM.

- <**VARDRAG**> computes the drag coefficient and the transfer coefficients for sensible and latent heat from wind speed, boundary layer stability and humidity.

4.4 Preprocessing: [oceproc.f]

4.4.1 Driving Routines

- <**OCEINIT**> interpolates global data sets onto the model grid and stores them in [FORCES] via <WRITEF>. The file names used as data source from the following routines are the same names as those of the permanent files, and not those used as local file names. Furthermore, it reads file [OCDAT] and initializes the 3-dimensional ocean state and it allows to initialize the ocean by interpolating data of an existing [OCDAT] onto data on a different grid.

- <**OCEPOST**> is the driving routine for postprocessing performed after the model run. **Entry** <**OCEPST4**> writes out data at the end of the model run. This routine is called from <OCESTOP>.

- <**OCESTOP**> cleans up the model run finally.

4.4.2 Grid Initialization

- <**MAKEXY**> initializes the x-and y-coordinates of the model using either the individual definitions for the model boundary (*XGL*, *XGR*, *YGU* and *YGO*) or the Gaussian grid for the T21, T32, T42, T63, T84 or T106 resolutions. To obtain one of the latter grids, the preprocessor [precomp.f] has to be used to activate the relevant parts of the code.

- <**ELIMIN**> deletes sea points in the land-sea mask $IFLG$ which are connected to the defined eliminating point indices (*KILLI* , *KILLJ*) by sea points. This routine is useful for deleting undesired lakes or sea points that are not connected to the ocean region of interest. However, if an index is defined which by mistake sits inside the desired model

domain, all active ocean points may be erased. Even a floating-point exception may appear when dividing through a zero number of active grid points.

- <**GRITUNE**> calls <ELIMIN>, <CHANNEL> and <SHUTOFF>, respectively, to delete undesired parts of the ocean after they have been cut off by a land bridge, to introduce channels to ensure that the ocean is deep enough (a narrow strait is left in the model grid) and to build desired land bridges.

- <**FLAGMOD**> allows to change the land-sea mask definition after it has been derived from the topography data set.

- <**YCORT21**> computes the y-coordinates of the Gaussian grid for the T21-resolution.

- <**YCORT32**> computes the y-coordinates of the Gaussian grid for the T32-resolution.

- <**YCORT42**> computes the y-coordinates of the Gaussian grid for the T42-resolution.

- <**YCORT63**> computes the y-coordinates of the Gaussian grid for the T63-resolution.

- <**YCORT84**> computes the y-coordinates of the Gaussian grid for the T84-resolution.

- <**YCORT16**> computes the y-coordinates of the Gaussian grid for the T106-resolution.

- <**SPHERIC**> computes the weights of the Gaussian latitudes. This routine is copy of the ECHAM model.

- <**BESSEL**> computes the Bessel coefficients for <SPHERIC>. This routine is a copy of the ECHAM model.

- <**SHUTOFF**> switches off sea points between two points in geographical coordinates and thus creates a land-brige between these two points.

- <**CHANNEL**> opens a channel between two points in geographical coordinates and ensures a minimal depth along the line that connects these two points.

- <**DEFMASK**> corrects the land-sea mask for various boundary conditions.

- <**GEOTOMO**> transforms a pair of (x,y)-coordinates from geographical into model coordinates on the sphere. These routines are activated if the grid rotation is used. They are needed to transform the forcing data onto the rotated model grid, to compute the local Coriolis parameter and the daily mean insolation for each grid point.
  Entry <**MOTOGEO**> performs the inverse operation.

- <**ROMATIN**> computes the rotation matrix.

- <**MO2GEOV**> performs a rotation from model to geographical coordinates for vector quantities.
  Entry <**MO2GEOS**> performs a rotation from model to geographical coordinates for scalar quantities.

4.4.3 Initialization of Atmospheric Forcing

Each atmospheric quantity used to compute fluxes into the ocean is initialized by its own routine.

- <**DEFORCE**> is used to define an artificial forcing when *ISW1=0*. In this case no atmospheric data are used by PIPE.
- <**ABSOL**> reads one month of the COADS surface wind speed from [UVABS] and interpolates it onto the model grid.
- <**VARIAN**> reads one month of the COADS standard deviation of the absolute wind speed from [UVDEV] and interpolates it onto the model grid.
- <**OCTEMP**> reads one month of the COADS sea surface temperature from [SSTEMP] and interpolates it onto the model grid.
- <**ATTEMP**> reads one month of the COADS air temperature from [AIRGLOB] and interpolates it onto the model grid.
- <**CUMULUS**> reads one month of the COADS cloud cover from [COVER] and interpolates it onto the model grid.
- <**VAPOR**> reads one month of the COADS surface air relative humidity from [WETNESS] and interpolates it onto the model grid.
- <**ECUSC**> reads one month of the ECMWF absolute wind speed from [USC] and iterpolates it onto the model grid.
- <**ECSTD**> reads one month of the ECMWF standard deviation of the wind from [STDV_USC] and interpolates it onto the model grid.
- <**ECTAU**> reads one month of the ECMWF surface wind stress from [TAUXY] and interpolates it onto the model grid.
- <**ECTSC**> reads one month of the ECMWF surface air temperature from [TSC] and interpolates it onto the model grid.
- <**ECSST**> reads one month of the ECMWF surface temperature from [SST] and interpolates it onto the model grid.
- <**ECCLD**> reads one month of the ECMWF cloudiness from [CLD] and interpolates it onto the model grid.
- <**ECSLP**> reads one month of the ECMWF surface pressure from [SLP] and interpolates it onto the model grid.
- <**ECHUM**> reads one month of the ECMWF surface air humidity from [HUM] and interpolates it onto the model grid.
- <**LEGATES**> reads one month of the Legates & Willmott precipitation data set from [RAIN1DEG] and interpolates it onto the model grid.
- <**SEATEMP**> reads the Levitus monthly mean surface temperature from [TEMP.MON] from the local file [SSTEMP] and interpolates it onto the model grid.

- <**SEASALT**> reads the Levitus annual mean sea surface salinity from [SSSALT] or from the monthly mean sea surface salinity from [SALT.MON] from the loacl file [SSSALT]. and interpolates it onto the model grid.

- <**WIND**> reads one month of the Hellermann and Rosenstein surface wind stress from [UVGLOB] and interpolates it onto the model grid.

- <**UPDATE**> computes the new current forcing of all the interpolated forcing data from the current forcing and the next stored monthly mean.

- <**WRITEF**> saves the interpolated atmospheric forcing on [FORCES].
  Entry <**READF**> reads the forcing from [FORCES].

4.4.4 Initialization of Ocean State

- <**LAYOUT1**> initializes the x- and y-coordinates, reads the topography from [TOP1DEG], interpolates it onto the model grid and arranges the land-sea mask. It also creates the EBM's orography.

- <**READTOP**> reads the file [TOP1DEG].

- <**LAYOUT2**> initializes the x- and y-coordinates, reads the topography from [TOP5MIN], interpolates it onto the model by using an envelope algorithm to take care of the variance of the bottom topography and arranges the land-sea mask. It also creates the EBM's orography.

- <**READTOF**> reads the file [TOP5MIN].

- <**DEFTOPO**> allows the user to construct his own topography and land-sea distribution. It is called by choosing $ISW8 = 0$.

- <**REMEDY**> fills up isolated depressions in the orography, which is essential to avoid lakes in the soil model of the EBM.

- <**INTLAY**> is the underlying routine to interpolate the observed ocean temperature and salinity onto the model grid and to generate the layer thicknesses and temperature and salinity for each layer.

- <**INTERPO**> reads the Levitus ocean temperature and salinity from [SALTEMP] and interpolates it onto the model grid.

- <**LAYDEPT**> optimizes the layer thicknesses and their respective temperature and salinity in order to yield the prescribed potential density within each layer.

- <**INTEGRA**> computes the vertically averaged potential density between two given depth levels from the Levitus data set.

- <**CREATIV**> initializes the vertical coordinates that are stored in *POTSOLL* and also initializes the whole density field from the initial temperature and salinity profile.

4.4.5 Generation of Isopycnal Coordinates

- <**MAKENEW**> organizes the generation of the model's isopycnal coordinates.
- <**MAKEVER**> computes the vertical coordinate system.
- <**MAKETET**> computes potential density for a single profile.
- <**MAKETET**> computes in situ density for a single profile.
- <**OLDINIT**> is the driver for generating a 3-dimensional ocean state for initialization from data imported from some [OCDAT], however, whose data use a different horizontal and vertical grid, i.e., number of layers *NZ*.
- <**REFORMH**> interpolates scalar model quantities between two different grids.
  Entry <**REFORMU**> interpolates vector quantities between two different grids.

4.4.6 Conservation Properties

- <**MASSCON**> computes the mass content of the ocean.
- <**HEATCON**> computes the heat content of the ocean.
- <**SALTCON**> computes the salt content of the ocean.
- <**HGTCOR**> removes residual negative layer thicknesses.
- <**MEAN2D**> computes the 2-dimensional area mean.
- <**MEAN3D**> computes the 3-dimensional area mean.
- <**ADJUST**> optionally corrects for the mass, heat and salt content when one of the budgets is not closed.
- <**ZEROLAY**> fills values in cells that have physically closed and have zero thickness. This routine is only needed for cosmetics to provide some numbers for a zero-layer.

4.4.7 Boundary Conditions

The basic idea behind the model's structure is that boundary conditions never appear explicitly in the model. All procedures assume that *I=1*, *I=NX*, *J=1* and *J=NY* contain the correct values for any differentiation. The subsequent routines are called whenever required to store the proper values on the grid points along the model margin. The land-sea mask $IFLG$ allows the model to deduce the proper boundary conditions inside the model area.

- <**BNDH1**> There are three different versions of the routines that provide the boundary conditions along the model margin. <BNDH1> is used for scalar quantities on scalar grid points. Note that the extension '1' is taken to express that only arrays containing one layer are treated by these routines.
  Entry <**BNDU1**> is used for scalar quantities on vector grid points.
  Entry <**BNDV1**> is used for vector quantities on vector grid points.
- <**BNDHNZ**> The following routines are similar to the above ones, however, full model arrays are treated with these routines.
  Entry <**BNDUNZ**> works like <BNDU1> but for all *NZ* layers.
  Entry <**BNDHNZ**> works like <BNDH1> but for all *NZ* layers.

- **<UVPOLE1>** These routines are called from <BND...> and compute the northern boundary values on vector points in case that the open North Pole is switched on. The 'UV' marks the routine used for vector points and 'S' the one for scalar points. The '1' at the end is used since these routines treat one layer only.
  Entry <SPOLE1> is used for scalar quantities.

- **<UVPOLEN>** Similar to above but used for full model arrays.
  Entry <SPOLEN> is used for scalar quantities.

- **<BNDFLGS>** performs the boundary conditions for a land-sea mask on scalar points.

- **<BNDFLGV>** performs the boundary conditions for a land-sea mask on vector points.

- **<OBH_MOD>** overwrites boundary data of layer thickness in case open boundaries are used.

- **<OBT_MOD>** overwrites boundary data of potential temperature in case open boundaries are used.

- **<OBS_MOD>** overwrites boundary data of salinity in case open boundaries are used.

- **<OBZH_MOD>** overwrites boundary data of tidal sea level anomalies in case open boundaries are used.

- **<SAVOPEN>** saves open boundary data for later use.

- **<SEALAND>** sets vector quantities to zero on land.

4.4.8 Routines for Listing

- **<LISTING>** generates parts of the protocol that appears on [OUTLIST]. Depending on the switches various fields are printed out mainly to provide a simple preview system.

- **<DRUCKV>** These routines print vector fields on the protocol as integer arrays with land set to blank fields. If the model dimensions are too large for the width of the output, only grid points with some increment are printed.
  Entry <DRUCKH> is used for scalar quantities.

- **<EXTRAV>** Same as before, but able to print all grid points within a portion of the model domain.
  Entry <EXTRAH> is used for scalar quantities.

- **<DRUCKF>** prints out the land-sea mask.

4.4.9 Routines for Model I/O

- **<WRITEM>** writes all the model fields onto the restart file [OCDAT] that are required to restart the model for continuing a model run. Optionally, the output can be saved as an ASCII file.
  Entry <READM> reads the first part of [OCDAT]. In case the file is an ASCII version of [OCDAT] <READM> uses formatted I/O automatically.
  Entry <READMF> reads the second part of the data from [OCDAT] that have been created by <WRITEM>. If the file is an ASCII version of [OCDAT] <READMF> also uses formatted I/O automatically.

- <**FILOPEN**> opens various files.
  Entry <**FILCLOS**> closes various files.

- <**PPSFAST**> writes data onto quick-look files. The data format is called PPSF (Post Processing System Format).

4.4.10 Routines for Model and CPU Time

- <**COUNTER**> converts the variable $KTOTAL$ into seconds, hours, days, months and years.

- <**TIMECON**> converts the variable $KTOTAL$ into seconds, minutes, hours, days, months and years.

- <**PRINCPU**> prints out the bookkeeped CPU-times.

4.4.11 Routines for Vertical Flag Calculations

- <**TOUCHLW**> computes the indices of the next deeper physical layer for each horizontal grid box within a given layer. It uses a critical number $EPSHGT$ to detect a physical layer.
  Entry <**TOUCHUP**> computes indices for next upper physical layer.

- <**CONTLW**> works similarly to <TOUCHLW>, but uses the flag field $IFLG$ to detect a physical layer.
  Entry <**CONTUP**> works similarly to <TOUCHUP>, but uses the flag field $IFLG$ to detect a physical layer.

4.4.12 Routines for Data Manipulation

- <**SELECT**> extracts a 2-dimensional real array out of a 3-dimensional one.

- <**KELECT**> extracts a 2-dimensional integer array out of a 3-dimensional one.

- <**SAMMELN**> extracts a 2-dimensional real array out of a 3-dimensional one with the vertical dimension of the number of data levels, i.e. from Levitus.
  Entry <**VERTEIL**> performs the inverse operation.

- <**COLLECT**> extracts a 2-dimensional real array out of a 3-dimensional one for a variable vertical index.

- <**VECMAX**> sets the 2-dimensional input field to the maximum of itself and a specified constant.
  Entry <**VECMIN**> uses the minimum instead.

- <**FELDMAX**> sets the 3-dimensional input field to the maximum of itself and a specified constant.
  Entry <**FELDMIN**> uses the minimum instead.

- <**STOREXZ**> stores xz-section onto another array, required for open boundaries.

- <**STOREYZ**> stores yz-section onto another array, required for open boundaries.

- <**STOREC1X**> stores xz-section onto southern boundary of 3-dimensional field, required for open boundaries.

- <**STORECNX**> stores xz-section onto northern boundary of 3-dimensional field, required for open boundaries.

- <**STOREC1Y**> stores yz-section onto western boundary of 3-dimensional field, required for open boundaries.

- <**STORECNY**> stores yz-section onto eastern boundary of 3-dimensional field, required for open boundaries.

4.4.13 Routines for Filtering

- >**RUBBS1**> filters 2-dimensional scalar quantities.
  Entry <**RUBBV1**> filters 2-dimensional vetcor quantities.

- >**RUBBSNZ**> filters 3-dimensional scalar quantities.
  Entry <**RUBBVNZ**> filters 3-dimensional vector quantities.

4.4.14 Routines for Filling Up

- <**FILLUP**> fills up arrays containing undefined data.

- <**FILLUP1**> fills up arrays containing undefined data.

4.4.15 Routines for Error Detection

- <**INFORM**> outputs an information that certain data exceeded a specified range.

- <**CAUTION**> outputs an information that certain data exceeded a specified range.

- <**CHECK**> outputs an information that certain data exceeded a specified range.

- <**SUICIDE**> causes an error exit through floating-point exception upon request.

4.5 Postprocessing: [ocepost.f]

The vertical coordinates in the model are defined by *HTOTAL*. This is the depth on which the origin of the vertical coordinate is located. Therefore, the topography depth is zero at a depth of *HTOTAL*. This determines how the depth values are stored in the 2d-array *DEPTH*. The sea level will vary around *HTOTAL* which is important to know if one converts the layer thicknesses and topography depth into interface depth values via <HTOZETA> and performs the reverse transformation via <ZETATOH>. However, for the output onto the postprocessing files, the vertical coordinate starts at the mean sea level and is positive downward (see various definitions of default levels by *COORDIN* in [ocepost.f]).

4.5.1 Postprocessing

- <**OCEPST1**> writes time averaged data onto the history file. This routine is called at every time step.
  Entry <**OCEPST2**> writes data onto the history file. It is called at each 15th of a month.
  Entry <**OCEPST3**> writes data onto the history file at each n'th time step, where n is defined by *ISW17*.
  Entry <**OCEPST5**> writes data onto the history file for *ISW17¡0*. This routine is used to create e.g. movies by writing data at each time step.

4.5.2 General Output Routine

The output of data is not straightforward in a model that uses Lagrangian coordinates. Since the data postprocessing is made easier if the quantities are available on levels instead as layers, a routine is offered for each of the standard problems that transforms quantities from Lagrangian coordinates to levels of constant depth. The available routines are listed below.

- <**PPSFOUT**> These routines write data in well-defined format (see 5.5.2) onto the history file. The data format henceforth is called PPSF (Post Processing System Format).

4.5.3 Output of Averaged Quantities

For many purposes it is sufficient to provide data on a time-averaged basis, e.g., when quantities do not vary much in time. Therefore, routines are offered that average a quantity and write out the average at a chosen rate. For the subsequent routines it is essential that they are called at each time step as they use some internal bookkeeping. The variable *DTMONTH* is used to determine the averaging period. The default setting is 1 month which means that the subsequently described routines compute and write monthly means onto the history file.

- <**QENTXY**> writes out the time-averaged entrainment and detrainment.
- <**QFLUXY**> writes out the time-averaged heat flux, fresh water flux, and wind stress components.
- <**QICEXY**> writes out time-averaged ice flow components, ice thickness and ice compactness.
- <**QMLDXY**> writes out the time-averaged mixed layer depth onto the history file.
- <**QSSTXY**> writes out the time-averaged sea surface temperature onto the history file.
- <**QSALTXY**> writes out the time-averaged sea surface salinity onto the history file.
- <**QSLDXY**> writes out the time-averaged sea level onto the history file.
- <**QUVXY**> writes out the time-averaged sea surface velocity components onto the history file.
- <**QCONVXY**> bookkeeps and writes out maximal convection depth.
- <**QEBM**> writes out a number of quantities characterizing the EBM's physics.
- <**QTIDES**> writes out the anomalous sea level due to tides.

- <**QFTTID**> performs a modal analysis and writes out the results.
- <**QTIDST**> bookkeeps and writes out residual currents.

4.5.4 Output of Instantaneous Fields

The following routines can be called to write out model fields after any transformation. Some of them already compute spatial averages and thus write out fields with reduced dimension. The first character of the name denotes whether a vector or a scalar is treated. The quantity code number must be specified, however, the routines specify and store the correct section code internally.

- <**VALL**> writes out all layers of a model array without any transformation and should be used if one wants to reconstruct model quantities during the data postprocessing.
- <**SALL**> writes out scalar quantities.
- <**VXYALL**> writes out all levels of vertically interpolated vector quantities.
  Entry <**SXYALL**> writes out all levels of vertically interpolated scalar quantities.
- <**VXYONE**> writes out a precalculated xy-section at a given depth.
  Entry <**SXYONE**> prints out scalar field.
- <**VXONE**> writes out a precalculated one-dimensional vector in x-direction.
  Entry <**SXONE**> prints out scalar field.
- <**VYONE**> writes out a precalculated one-dimensional vector in y-direction.
  Entry <**SYONE**> prints out scalar field.
- <**VXSEC**>write outs a one-dimensional vector in x-direction.
  Entry <**SXSEC**> prints out scalar field.
- <**VYSEC**> writes out a one-dimensional vector in y-direction.
  Entry <**SYSEC**> prints out scalar field.
- <**VZSEC**> writes out a one-dimensional vector in z-direction for certain level.
  Entry <**SZSEC**> prints out scalar field.
- <**VXYSEC**> writes out an internally computed xy-section at a given depth.
  Entry <**SXYSEC**> prints out scalar field.
- <**VXZSEC**> writes out an internally computed xz-section at a given latitude.
  Entry <**SXZSEC**> prints out scalar field.
- <**VYZSEC**> writes out an internally computed yz-section at a given longitude.
  Entry <**SYZSEC**> prints out scalar field.

4.5.5 Tridiagonal Solvers

- <**TRIDIAX**> computes the solution of a single linear equation of tridiagonal type for NX number of equations. It is used by <TIDES>.
- <**TRIDIAY**> computes the solution of a single linear equation of tridiagonal type for NY number of equations. It is used by <TIDES>.

- <**TRIDIA2**> computes the solution of two independent linear equations of tridiagonal type. It is used by <EBM> to compute the solution of the two tridiagonal systems for temperature and moisture.

- <**TRIDIA3**> computes the solution of three independent linear equations of tridiagonal type. It is used by <SEAICE> to compute the solution of the three tridiagonal systems for ice thickness, ice concentration and snow cover.

- <**TRIDIAV**> computes the solution of *NV*NZ* linear equations of tridiagonal type for *NX* number of equations. It is used by <TRACTIV>.

- <**TRIONE**> computes the solution of *NV* independent pairs of coupled linear equations of block-tridiagonal type with *NX* number of equations. It is used be <SEAICE> to find the solutions for the rheology terms in the momentum equation. Similar to <TRITWO> the pair of equations couple the x- and y-components of the ice flux, in order to obtain also a fast convergence for the bulk and shear viscosity terms.

- <**TRITWOX**> computes the solution of *NV*NZ* independent pairs of coupled linear equations of block-tridiagonal type with *NX* number of equations. It is used by <MOLVERX> and is required to couple x- and y-components of the advection and curvature terms since flux components are cross-referenced in the momentum equations.

- <**TRITWOY**> computes the solution of *NV*NZ* independent pairs of coupled linear equations of block-tridiagonal type with *NY* number of equations. It is used by <MOLVERY> and is required to couple x- and y-components of the advection and curvature terms since flux components are cross-referenced in the momentum equations.

- <**TRIBLCKX**> computes the solution of *NV* independent sets of linear equations of block-tridiagonal type with *NX*NZ* number of equations. Each linear equation system solves the matrix on a xz-section. *NV=(NY-1)/2* is the result of the used line-relaxation method.

- <**TRIBLCKY**> computes the solution of *NV* independent sets of linear equations of block-tridiagonal type with $NY * NZ$ number of equations. Each linear equation system solves the matrix on a yz-section. *NV=(NX-1)/2* is the result of the used line-relaxation method.

4.5.6 Coupling Interface

The following routines allow easy use of the model either in coupled mode or with simple forcing from atmospheric data that are not provided by the model's data bank. The idea is to define a common data area for all fluxes required to force the model and for those quantities another model needs to analyse the ocean state. By calling these routines the user does not need to know where the data are stored. Thus, the model can be used as black box. The further advantage is that the code ensures that the forcing is correctly read from this common data area whenever needed. Note also that the fluxes can be interpreted either as absolute values or as anomalies. In the latter case the climatology is taken from the model.

4.5.6.1 Data Transfer into the Ocean Model

The data passed into the subsequent routines must already be defined on the ocean model grid and must have the same dimensions as the model grid. Each routine also contains an entry that performs the opposite operation, i.e. to export data from the ocean model common blocks.

- **<AOPME>** writes the fresh water flux into the common data area.
  Entry <OAPME> exports the model computed fresh water flux.

- **<AOQFLX>** writes the heat flux and the solar radiation into the common data area.
  Entry <OAQFLX> exports the model computed heat fluxes.

- **<AOTAU>** writes the wind stress components into the common data area.
  Entry <OATAU> exports the model computed wind stress.

- **<AOUSTAR>** writes the turbulent kinetic energy u_*^3 into the common data area.
  Entry <OAUSTAR> exports the model computed turbulent kinetic energy.

- **<AOTICE>** writes the snow surface and snow-ice interface temperature into the common data area.
  Entry <OATICE> exports the model computed snow surface and snow-ice interface temperature.

- **<AOAIRT>** writes the surface air temperature into the common data area.
  Entry <OAAIRT> exports the model computed surface air temperature.

- **<AOOCET>** writes the ocean surface temperature into the common data area.
  Entry <OAOCET> exports the model computed ocean surface temperature.

- **<AOOCES>** writes the ocean surface salinity into the common data area.
  Entry <OAOCES> exports the model computed ocean surface salinity.

- **<AOAHUM>** writes the surface air humidity into the common data area.
  Entry <OAHUM> exports the model computed surface air humidity.

- **<AOACLD>** writes the atmospheric cloudiness into the common data area.
  Entry <OACLD> exports the model computed atmospheric cloudiness.

- **<AOAUVA>** writes the atmospheric scalar surface wind into the common data area.
  Entry <OAAUVA> exports the model computed atmospheric scalar surface wind.

- **<AOAUVS>** writes the atmospheric standard deviation of the scalar surface wind into the common data area.
  Entry <OAAUVS> exports the model computed atmospheric standard deviation of the scalar wind.

- **<AOSURP>** writes the atmospheric surface pressure into the common data area.
  Entry <OASURP> exports the model computed surface pressure.

- **<SFOR2MOD>** interpolates scalar data from an arbitrary grid onto the model grid. It is required if data defined on another grid have to be prepared before one of the routines <AOPME> to <AOTICE> can be called. <SFOR2MOD> also is able to fill gaps in the data source using a Laplacian filter.

- **<VFOR2MOD>** interpolates vector data from an arbitrary grid onto the model grid. It is required if data defined on another grid have to be prepared before one of the routines <AOPME> to <AOICE> can be called. <VFOR2MOD> also is able to fill gaps in the data source using a Laplacian filter.

4.5.6.2 Data Transfer out of the Ocean Model

The output data of the following routines are defined on the ocean model grid. This routines typically are required to supply an atmospheric model with data that are required for either grid-interpolation or flux calculations.

- <**OAGRID**> provides the land-sea mask and the x- and y-coordinates of the ocean model to be used, e.g., for interpolation between two different model or data grids.
- <**OASST**> reads the model sea surface temperature and the $\partial Q/\partial T$ out of the common data area.
- <**OADICE**> reads the ice thickness, snow thickness and ice compactness out of the common data area, e.g., for later heat flux computations in an atmospheric model.

4.5.6.3 Internal Data Transfer

The following routines supply the model with data from the common data area, which is /JMOCOUP/ for the fluxes and /JMOBASE/ for the underlying observational data, however, they should not be used explicitly to feed the model with data. Instead, use the following routines:

- <**HEATMOD**> extracts the total surface heat flux from the common data area. See <SURFLUX>.
- <**SOLMOD**> extracts the solar radiation from the common data area. See <SURFLUX>.
- <**PMEMOD**> extracts the fresh water flux from the common data area. See <SURFLUX>.
- <**TAUMOD**> extracts the wind stress components from the common data area. See <STRESS>.
- <**MIXMOD**> extracts the turbulent kinetic energy for the mixed layer from the common data area. See <MIXEXP> and <MIXIMP>.
- <**ICEMOD**> extracts the snow surface and snow-ice interface temperature from the common data area. See <SURFLUX>.
- <**SNOMOD**> extracts the heat fluxes that concern ice and snow out of the common block area. See <DIADICE>.
- <**TKEMOD**> extracts the turbulent kinetic energy out of the common data area. See <MECHAN>.
- <**AIRTMOD**> extracts the surface air temperature from the common data area.
- <**OCETMOD**> extracts the sea surface temperature from the common data area.
- <**OCESMOD**> extracts the sea surface salinity from the common data area.
- <**AHUMMOD**> extracts the surface air humidity from the common data area.
- <**ACLDMOD**> extracts the atmospheric cloudiness from the common data area.
- <**AUVAMOD**> extracts the scalar surface wind from the common data area.

- <**AUVSMOD**> extracts the standard deviation of the scalar surface wind from the common data area.
- <**ASURMOD**> extracts the atmospheric surface pressure from the common data area.

4.5.7 Coupling with Tracers

- <**TWRIT**> generates time-averaged data required by the offline version of the tracer model.

4.5.8 Routines for Vertical Interpolation

- <**TRASXY**> computes the horizontal section of some scalar quantity at some depth.
- <**TRASZX**> computes the zx-section of some scalar quantity at some depth.
- <**TRASZY**> computes the zy-section of some scalar quantity at some depth.
- <**TRAVXY**> computes the horizontal section of some vector quantity at some depth.
- <**TRAVZX**> computes the zx-section of some vector quantity at some depth.
- <**TRAVZY**> computes the zy-section of some vector quantity at some depth.
- <**TRAVONE**> computes a vertical profile of a vector quantity at one xy-grid point. **Entry** <**TRASONE**> computes a vertical profile of a scalar quantity at one xy-grid point.

4.5.9 Routines for Data Manipulation

- <**STORE1**> copies a 2-dimensional real array onto another one.
- <**STOREN**> copies a 3-dimensional real array onto another one.
- <**KTORE1**> copies a 2-dimensional integer array onto another one.
- <**KTOREN**> copies a 3-dimensional integer array onto another one.
- <**VALADD1**> adds a 2-dimensional real array onto another one.
- <**VALADDN**> adds a 3-dimensional real array onto another one.
- <**VALSHFT**> adds a constant onto a 2-dimensional real array.
- <**VALMUL1**> multiplies a 2-dimensional real array with some factor.
- <**VALMULN**> multiplies a 3-dimensional real array with some factor.
- <**SETVAL1**> sets a 2-dimensional real array to a constant value.
- <**SETVALN**> sets a 3-dimensional real array to a constant value.
- <**KETVAL1**> sets a 2-dimensional integer array to a constant value.
- <**KETVALN**> sets a 3-dimensional integer array to a constant value.
- <**SELECTIDE**> extracts a specific tidal mode from an array containing all tides.

4.5.10 Routines to Shift Data

- <**ROLLZXY**> shifts 3-dimensional real data in x-direction in order to improve convergence.
- <**KOLLZXY**> shifts 3-dimensional integer data in x-direction in order to improve convergence.
- <**ROLLXY**> shifts 2-dimensional real data in x-direction in order to improve convergence.
- <**KOLLXY**> shifts 2-dimensional integer data in x-direction in order to improve convergence.
- <**ROLLX**> shifts 1-dimensional real data in x-direction in order to improve convergence.
- <**ROLLZX**> shifts 2-dimensional real data on a xz-section in x-direction in order to improve convergence.

4.5.11 Routines to Bookkeep Budgets

- <**QBDGT_INIT**> initializes arrays that boodkeep the budget fields.
  Entry <**QBDGT_ADD**> adds current budget terms onto already existing integrals.
  Entry <**QBDGT_PPSF**> writes out budget terms at a specified frequency.

4.6 Passive Tracer Model: [ocetrac.f]

- <**TRAINIT**> reads initialization data from [TRDAT]. If it is empty the tracer fields are initialized.
- <**TRAFORC**> determines the kind of forcing. Please adjust to your needs.
- <**TRASTEP**> organizes initialization, time stepping and postprocessing.
- <**TRAPOST**> performs the postprocessing.
- <**TRASTOP**> cleans up and saves the data in [TRDAT].
- <**TRACPAS**> performs the tracer forecast based on a scheme similar to <TRACTIV> for the active tracers.
- <**TRAVECT**> performs the vertical convection when stratification becomes instable.
- <**TREAD**> prepares the ocean data like flow, layer thickness, density, vertical exchange rates and others to provide data for the time stepping. Dependent on whether the tracer model is running in online or offline mode, its forcing data either are taken from the ocean model's common blocks or are read from [TRAFORCE].
- <**CSOURCE**> is an empty routine that allows to add an arbitrary forcing to the model without the need to understand the time stepping.

4.7 Open Boundaries: [oceopen.f]

4.7.1 Routines for Preprocessing Data

- <**INTCUT**> is the main driving routine to create the boundary data. It is called by <OCEINIT>.
- <**STREAM_T106**> reads the barotropic transport from data and interpolates it onto the model grid.
- <**SELECOBND**> reads monthly 3-dimensional ocean data for temperature and salinity from Levitus, reads the barotropic transports that have been created by a T106 version of PIPE, interpolates them onto the model grid and saves them on file [OBNDSEC]. If the tide model is switched on data are read, interpolated and saved onto [OBNDSEC]. Therefore, a file [OBNDSEC] is not interchangable between models versions that don't use and those which use a tide model although the model geometry and grid is identical.
- <**WRITEOBND**> writes temperature, salinity and transports along the southern, northern, western and eastern boundary of the model domain onto [OBNDSEC].
- <**INTIDES**> reads tidal data and interpolates them onto the model grid.
- <**WRITETIDE**> writes amplitude and phases of various tidal modes onto [OBNSEC].
- <**READOBND**> reads current boundary data.
- <**READOBNDOLD**> reads boundary data of last month.
- <**STRDAT1X**> writes southern boundary of observed ocean data on a 3-dimensional array onto a 2-dimensional array containing an xz-section.
- <**STRDATNX**> writes northern boundary of observed ocean data on a 3-dimensional array onto a 2-dimensional array containing an xz-section.
- <**STRDAT1Y**> writes western boundary of observed ocean data on a 3-dimensional array onto a 2-dimensional array containing an yz-section.
- <**STRDATNY**> writes eastern boundary of observed ocean data on a 3-dimensional array onto a 2-dimensional array containing an yz-section.
- <**BAROTROP**> merges the baroclinic and the barotropic sea level anomaly and constructs a sea level, which is connected around corners of the model domain. This routine contains a *stop*-statement in order to remember a user to adjust this routine to the user's application.

4.7.2 Routines for Coordinate Conversion

- <**CONVDX**> converts the definition of the vertical coordinate's origin for an array with dimension NX.
- <**CONVDY**> converts the definition of the vertical coordinate's origin for an array with dimension NY.
- <**LAYDEPTX**< performs the data conversion from level to layer coordinates, i.e. computes layer thicknesses from ocean data and the vertical integrals for temperature and salinity, for an xz-section.

- <**LAYDEPTY**< performs the data conversion from level to layer coordinates, i.e. computes layer thicknesses from ocean data and the vertical integrals for temperature and salinity, for an yz-section.

- <**INTEGRAX**< performs the vertical integration of potential density, potential temperature and salinity for an xz-section.

- <**INTEGRAY**< performs the vertical integration of potential density, potential temperature and salinity for an yz-section.

- <**ROTVEC**< rotates a vector from the geographical coordinates onto a rotated grid.

4.7.3 Routines for Pressure Calculation

- <**STATETRX**> computes the in situ density for an array with dimension *NX*.
  Entry <**STATETPX**> computes the potential density for an array with dimension *NX*.

- <**STATETRY**> computes the in situ density for an array with dimension *NY*.
  Entry <**STATETPY**> computes the potential density for an array with dimension *NY*.

- <**STATEVRX**> computes the in situ density for an array with dimension *(NZ,NX)*.
  Entry <**STATEVPX**> computes the potential density for an array with dimension *(NZ,NX)*.

- <**STATEVRY**> computes the in situ density for an array with dimension *(NZ,NY)*.
  Entry <**STATEVPY**> computes the potential density for an array with dimension *(NZ,NY)*.

- <**TTOTETX**> computes the potential temperature for an array with dimension *NX*.

- <**TTOTETY**> computes the potential temperature for an array with dimension *NY*.

- <**REPRESSX**> computes the pressure for an array with dimension *(NZ,NX)*.

- <**REPRESSY**> computes the pressure for an array with dimension *(NZ,NY)*.

- <**NEWDENSX**> computes iterativly in situ density from potential temperature and salinity for an array with dimension *(NZ,NX)*.

- <**NEWDENSY**> computes iterativly in situ density from potential temperature and salinity for an array with dimension *(NZ,NY)*.

- <**GRDX**> computes the pressure gradient in x-direction.

- <**GRDY**> computes the pressure gradient in y-direction.

4.7.4 Routines for Preparing Data

- <**UPDATEOBND**> organizes the updating of the boundary data in time.

- <**UPDATEX**> updates an array with dimension *NX*.

- <**UPDATEY**> updates an array with dimension *NY*.

- <**UPDATEXZ**> updates an array with dimension *(NZ,NX)*.

- <**UPDATEYZ**> updates an array with dimension *(NZ,NY)*.
- <**STRLAYCUT**> organizes the computation of layer thicknesses and vertically integrated temperatures and salinities from the profiles of temperature and salinity in z-coordinates, and finally computes sea level from the baroclinic and barotropic contributions.
- <**LAYX**> computes the layer thickness and vertically integrated temperatures and salinities for arrays with dimension *(NZ,NX)*.
- <**LAYY**> computes the layer thickness and vertically integrated temperatures and salinities for arrays with dimension *(NZ,NY)*.
- <**WRITEC1X**> writes southern boundary data of a 3-dimensional array onto a 2-dimensional array containing an xz-section.
- <**WRITECNX**> writes northern boundary data of a 3-dimensional array onto a 2-dimensional array containing an xz-section.
- <**WRITEC1Y**> writes western boundary data of a 3-dimensional array onto a 2-dimensional array containing an yz-section.
- <**WRITECNY**> writes eastern boundary data of a 3-dimensional array onto a 2-dimensional array containing an yz-section.
- <**STROBND**> organizes the copy of boundary data.
- <**STRDATX**> copies an array with dimension *NX*.
- <**STRDATY**> copies an array with dimension *NY*.
- <**STRDATY**> copies an array with dimension *NY*.
- <**STRDATXZ**> copies an array with dimension *(NZ,NX)*.
- <**STRDATYZ**> copies an array with dimension *(NZ,NY)*.
- <**SAMMELNX**> extracts an array with dimension *NX* from a 2-dimensional zx-section. **Entry** <**VERTEILX**> performs the reverse operation.
- <**SAMMELNY**> extracts an array with dimension *NY* from a 2-dimensional zy-section. **Entry** <**VERTEILY**> performs the reverse operation.
- <**SETVAL1X**> sets an array with dimension *NX* to a specified constant.
- <**SETVAL1Y**> sets an array with dimension *NY* to a specified constant.
- <**SETVALNX**> sets an array with dimension *(NZ,NX)* to a specified constant.
- <**SETVALNY**> sets an array with dimension *(NZ,NY)* to a specified constant.
- <**STORENX**> copies an array with dimension *(NZ,NX)*.
- <**STORENY**> copies an array with dimension *(NZ,NY)*.
- <**SELHMLX**> extract the mixed layer thickness on an x-section from the 3-dimensional layer thickness array.
- <**SELHMLY**> extract the mixed layer thickness on an y-section from the 3-dimensional layer thickness array.

4.7.5 Routines for Listing

- <**DRUCKHX**> prints out a scalar quantity defined on a xz-section.
- <**DRUCKHY**> prints out a scalar quantity defined on a yz-section.
- <**DRUDATX**> prints out observed ocean data defined on a xz-section.
- <**DRUDATY**> prints out observed ocean data defined on a yz-section.

4.8 Parallel Code: [oceutil.f]

4.8.1 Pre- and Postprocessing

- <**MPPSEND**> is the main program that preprocesses the input data and organizes the preparation of the subsequent execution of the parallel code.
- <**MPPSTEP**> is the main program that serves to provide the parallel job with data while it is running.
- <**MPPSTOP**> is the main program that postprocesses the data left by the parallel job on the disk.
- <**t3d_main**> is the main program of the parallel job.
- <**t3d_setup**> sets up the topology of the parallel job.
- <**RECEIVE**> as part of the parallel job has the task to receive the data from the server and to distribute the data to the respective processors.
- <**SEND_C90**> as part of the parallel job has the task to collect the data and return them to the server.

4.8.2 Data Transfer

4.8.2.1 Routines on the Server-Side

- <**SPIPE_OPN**> opens the command pipe. The command pipe is used to synchronize the tasks on the server and the client.
  Entry <**SPIPE_CLS**> closes the command pipe.
- <**SERVER_OPN**> opens the data pipes.
  Entry <**SERVER_CLS**> closes the data pipes.
- <**SERVER_GET**> receives real data from the client.
  Entry <**SERVER_PUT**> sends real data to the client.
- <**KERVER_GET**> receives integer data from the client.
  Entry <**KERVER_PUT**> sends integer data to the client.

4.8.2.2 Routines on the Client-Side

- <**cpipe_opn**> opens the command pipe. The command pipe is used to synchronize the tasks on the server and the client.
  Entry <**cpipe_cls**> closes the command pipe.

- <**client_opn**> opens the data pipes.
  Entry <**client_cls**> closes the data pipes.

- <**client_get**> receives real data from the server.
  Entry <**client_put**> sends real data to the server.

- <**klient_get**> receives integer data from the server.
  Entry <**client_put**> sends integer data to the server.

4.8.3 Josef's Private Interface

Josef's **P**rivate **I**nterface is a collection of routines that perform a data transfer either via the SHMEM- or the MPI-library. The routines do the hard work for communication between the processors of an MPP. Subsequently, lx is some arbitrary vertical dimension, nx_tot is the model's full dimension in x-direction, ny_tot is the model's full dimension in y-direction, and nx and ny are the x- and y-dimensions on each node.

- <**jpi_col_y**> collects data with dimension ny onto processor (1,1) and constructs data with dimension ny_tot.
  Entry <**jpi_col_y_1**> collects data with dimension ny onto processor (1,pe_y)

- <**jpi_col_x**> collects data with dimension nx onto processor (1,1) and constructs data with dimension nx_tot.

- <**jpi_col_xy**> collects data with dimension lx, nx, ny onto processor (1,1) and constructs data with dimension lx, nx_tot, ny_tot.
  Entry <**jpi_col_xy_1**> collects data with dimension lx, nx, ny onto processor (1,pe_y) and constructs data with dimension lx, nx_tot, ny_tot.

- <**jpi_dis_y**> distributes data with dimension ny_tot from processor (1,1) and constructs data with dimension ny.
  Entry <**jpi_dis_y_1**> distributes data with dimension ny_tot from processor (1,pe_y)

- <**jpi_dis_x**> distributes data with dimension nx_tot from processor (1,1) and constructs data with dimension nx.

- <**jpi_dis_xy**> distribute data with dimension lx, nx_tot, ny_tot from processor (1,1) and constructs data with dimension lx, nx, ny.
  Entry <**jpi_dis_xy_1**> distributes data with dimension lx, nx_tot, ny_tot from processor (1,pe_y) and constructs data with dimension lx, nx, ny.

- <**jpi_zon**> performs a zonal data exchange.

- <**jpi_mer**> performs a meridional data exchange.

- <**jpi_mer_s**> performs a meridional data exchange for scalar quantities.

- <**jpi_mer_v**> performs a meridional data exchange for vector quantities.

- <**jpi_shift**> performs a zonal data shift.
- <**jpi_sum**> performs a global sum based on a binary tree algorithm.
- <**jpi_max**> performs a global minimum based on a binary tree algorithm.
- <**jpi_flood**> performs a broadcast from processor (1,1) to all others.
- <**jpi_merge**> collects data.
- <**jpi_hold**> performs a barrier.

Part III

User's Manual

Chapter 5

How to Use PIPE

5.1 Installing PIPE

The **PIPE** model comes along with a tar-file containing codes, scripts and data. Assuming that the file is untarred in a directory *PIPE* then the directory structure becomes as follows:

```
PIPE/Source                  -> full source codes
   ./Source/Manual           -> PostScript file for Techn. Rep. No. 7
   ./Source/PVP              -> preprocessed source code for global T42
          ./PVP/Jobs         -> scripts for compiling and running the code
          ./PVP/Model        -> example directory for specific model version
   ./Source/RISC             -> preprocessed source code for global T42
          ./RISC/Jobs        -> scripts for compiling and running the code
          ./RISC/Model       -> example directory for specific model version
   ./Source/MPP              -> unpreprocessed source code
          ./MPP/Jobs         -> scripts for compiling and running the code
          ./MPP/Model        -> example directory for specific model version
              ./Model/bin    -> directory containing makefiles and executables
              ./Model/lib    -> directory containing libraries
              ./Model/code   -> directory containing fsplit'd code
   ./Source/DOS              -> preprocessed code for DOS
PIPE/PostPro                 -> home directory of postprocessing software
   ./PostPro/Codes           -> codes for modules
   ./PostPro/Scripts         -> scripts for modules
   ./PostPro/Help            -> documentation files for modules
   ./PostPro/Objects         -> object files for modules
PIPE/Graphic                 -> graphic library
   ./Graphic/Manual          -> manual for graphic library
   ./Graphic/GrADS           -> GrADS alternative for graphics
PIPE/Tools                   -> a few tiny helpfull codes
PIPE/Data                    -> data for initializing PIPE
PIPE/bin                     -> usefull scripts
```

There exist 3 tested installations for a

1. Parallel Vector Processor machine like a CRAY C90
2. Massively Parallel Processor machine like CRAY T3D
3. RISC Processor based workstations like from DEC, HP, IBM or SUN. The code also works on a PC under Linux and has been successfully tested, excluding scripts and postprocessing, on a PC using DOS6.22 and the GCC/G77-compilers from the Gnu-Project.

5.1.1 Installation: Step No. 1

In order to arrive at a ready-to-compile model there exist the

- Script [./Source/setup.pvp] to create a PVP-version on the directory [./Source/PVP]
- Script [./Source/setup.rsc] to create a RISC-version on the directory [./Source/RISC]
- Script [./Source/setup.mpp] to create a MPP-version on the directory [./Source/MPP]
- Script [./Source/setup.bat] to create a DOS-version on the directory [./Source/DOS]

The model source code contains preprocessing directives that are used by the scripts [setup.pvp], [setup.rsc] and [setup.bat]. These extract a PVP or a RISC version of the code. The result is copied to [./Source/PVP] and [./Source/RISC], respectively. The MPP-version, however, ist not preprocessed by [setup.mpp]. It is only copied to the directory [./Source/MPP]. The MPP-codes are preprocessed at compile-time, later. The installation of a DOS-version is not complete. All relevant scripts are missing. However, the RISC-version worked well on a DOS-machine if compiled and loaded by hand.

5.1.2 Installation: Step No. 2

In each of the directories [./Source/PVP], [./Source/RISC] and [./Source/MPP] there exists a subdirectory [Model] where a further script [setup] is located. These scripts install by default a T42-model with 130×66 horizontal grid points and 11 layers. In the directories for the PVP-, RISC- and MPP-version files [ocemain.f], [ocestep.f], [oceproc.f], [ocepost.f], [ocemods.f] and [ocetrac.f] appear. These files need to be changed according the rules as described in the next section in order to setup a different model version than the T42. For the MPP-version a further step is to split these FORTRAN-files into files containing one subroutine each only. These are copied to [./Source/MPP/Model/code] and are used by the makefiles to create libraries and finally executables for a CRAY T3D. The compile jobs can be found in [./Source/PVP/Jobs], [./Source/RISC/Jobs] and [./Source/MPP/Jobs].

During a first installation of the T42-model it is recommended to leave the script [./Source/RISC/Model/setup] unchanged, also leave unchanged the subsequently installed file [./Source/RISC/Model/ocemain.f] and test whether you succeed in running the T42-model. If this step is successfull, start to setup your own model application.

5.1.2.1 Installation of a RISC-version

1. Adjust [./Source/RISC/Model/setup] in order to obtain FORTRAN codes that use the proper dimensions, and other options as documented in [setup]
2. Run [./Source/RISC/Model/setup]
3. Adjust code, essentially [ocemain.f], to obtain a model domain and proper choice of the model parameters
4. Run [./Source/RISC/Model/make.sun] in order to create object files, which by default are copied to the subdirectory [./Source/RISC/Model/Run]
5. Adjust [./Source/RISC/Model/model.sun] to perform a model initialization
6. Run [./Source/RISC/Model/model.sun]

7. Adjust [./Source/RISC/Model/model.sun] to perform a continuation run

8. Run [./Source/RISC/Model/model.sun]

In order to setup an arbitrary version for a RISC computer the file [./Source/parampvp.h] is inserted into the code through an *#include*-statement. The dimensions *nx_tot* and *nx* for the x-direction, and the dimensions *ny_tot* and *ny* for the y-direction each have identical meaning. These dimensions are set through the script [./Source/RISC/Model/setup]. There is no need to modify [parampvp.h] during the installation.

5.1.2.2 Installation of a PVP-version

1. Adjust [./Source/PVP/Model/setup] in order to obtain FORTRAN codes that use the proper dimensions, and other options as documented in [setup]

2. Run [./Source/PVP/Model/setup]

3. Adjust code, essentially [ocemain.f], to obtain a model domain and proper choice of the model parameters

4. Run [./Source/PVP/Model/make.cri] in order to create object files, which by default are copied to the subdirectory [./Source/PVP/Model/Run]

5. Adjust [./Source/PVP/Model/model.cri] to perform a model initialization

6. Run [./Source/PVP/Model/model.cri]

7. Adjust [./Source/PVP/Model/model.cri] to perform a continuation run

8. Run [./Source/PVP/Model/model.cri]

In order to setup an arbitrary version for a PVP computer the file [./Source/parampvp.h] is inserted into the code through an *#include*-statement. The dimensions *nx_tot* and *nx* for the x-direction, and the dimensions *ny_tot* and *ny* for the y-direction each have identical meaning. These dimensions are set through the script [./Source/PVP/Model/setup]. There is no need to modify [parampvp.h] during the installation.

5.1.2.3 Installation of a MPP-version

1. Adjust [./Source/MPP/Model/setup] in order to obtain FORTRAN codes that use the proper dimensions, and other options as documented in [setup]

2. Run [./Source/MPP/Model/setup]

3. Adjust code, essentially [ocemain.f], to obtain a model domain and proper choice of the model parameters

4. Run [./Source/MPP/Model/setup] again. The files for compilation now are located on [./Source/MPP/Model/code]. Transfer these files if the [setup] was not executed on the main-frame of the MPP.

5. Run [./Source/MPP/Model/make_all.job] in order to create libraries, which by default are copied to the subdirectory [./Source/RISC/Model/lib] and to create an executable copied to the subdirectory [./Source/RISC/Model/bin]

6. Adjust [./Source/RISC/Model/model.mpp] to perform a model initialization
7. Run [./Source/RISC/Model/model.mpp]
8. Adjust [./Source/RISC/Model/model.mpp] to perform a continuation run
9. Run [./Source/RISC/Model/model.mpp]

In order to setup an arbitrary version for a MPP computer there are rules to setup the dimensions. During the installation the file [./Source/parammpp.h] is inserted into the code through an *#include*-statement. The dimensions used on each node are the result of the parameters

1. *nx_tot*, which is the full dimension in x-direction of the model domain
2. *npe_x*, which is the number of processors used in x-direction
3. *ny_tot*, which is the full dimension in y-direction of the model domain
4. *npe_y*, which is the number of processors used in y-direction

The dimensions nx and ny used for each node are then computed through $nx = (nx\_tot - 2)/npe\_x + 2$ and $ny = (ny\_tot - 2)/npe\_y + 2$ inside [parammpp.h]. It must be guaranteed that $(nx\_tot - 2)/npe\_x$ and $ny = (ny\_tot - 2)/npe\_y$ has no remainder. In addition it must be guaranteed that the variable *NPIPE* in the script [pipe] is equal *npe_y*. That's because each processor in y-direction must be connected via UNIX-pipes with the server. In general *npe_x* and *npe_y* with the product *npe_x npe_y* equal the number of used processors should be chosen in such a way that *npe_x*=*npe_y* in order to minimize data traffic. If this is not possible choose $npe\_x < npe\_y$ which results in better convergence in x-direction because of a smaller partitioning and because the data traffic between server and clients is splitted over a higher number of *npe_y* UNIX-pipes. For more details please also read [./README].

5.1.3 Installation: Step No. 3

5.1.3.1 Installing the Plot-System

In order to create pictures from model output the graphic library [jmoplane.f] has to be installed. For this execute the scripts [libgen.sun] or [libgen.cri] as examples for a SUN-workstation or a CRAY on the directory [./Graphic]. Basically, the script [plot] expects a file [jmoplane.o] and [utility.o] on the *Graphic*-subdirectory. Furthermore, the files [COAST.LIN] and [COAST.COL], which contain the coastline geometry, must be located on the same directory. To be able to access these data adjust the *open*-statement in [jmoplane.f]. Finally, don't forget to adjust the file [plot.f] in such a way that model geometry appears properly on the pictures.

5.1.3.2 Installing the Postprocessing

In order to install the postprocessing the scripts [modgen.sun] or [modgen.cri] must be executed separately for each utility. The task of these scripts is to create object-files which are saved. Later, when some postprocessing command is executed, these object-files are loaded and an executable is generated. If the postprocessing is to be installed on a remote computer the script [modgen.cri] gives an example on how scripts, object files and documentation files are installed. On the local machine the script [modgen.sun] generates and saves only the object files, while the script and the domumentation files are already installed after the command: *tar xvf pipe.tar*.

5.2 Setting up PIPE

In this section it will be described how to adjust the code for a certain purpose. PIPE is not developed only for one application. It is programmed to allow fundamental changes in the application with only few adjustments of parameters. Standard FORTRAN77 is used, so the model will run on many different types of computers. The model was tested on workstations like SUN, IBM-RISC, DEC or HP, on mini-computers like MicroVax, and on main-frame computers like IBM3090, Cyber205, NEC-SX3/4, Fujitsu- or CRAY-type computers like CRAY C90 or CRAY T3D.

5.2.1 The Code Preprocessor

The code is written for scalar, vector or parallel computers with shared or distributed memory. The optimization was primarily done for CRAY-type computers. Since the same code cannot be used for all of type of computers, control statements have been included that are used by [precomp.f] to either activate or disable certain FORTRAN statements. These control words are written into columns 73-80. The precompiler [precomp.f] distinguishes 4 types of statements.

- **Single or Double Precision:** Since the code has to work with 64-bit words to ensure an accurate solution of the matrix for the wave equation, a double precision version has to be set up for a 32-bit workstation. The code optionally declares all REAL variables with 64-bit words via an IMPLICIT DOUBLE PRECISION statement. The keywords for the precompiler are '_DOUBLE_' and '_SINGLE_'.

- **Scalar or Vector:** Dependent on whether a computer uses a RISC- or Vector-processor, DO-loops are written differently to help compilers to obtain the best performance. The keywords are '_SCALAR_'and '_VECTOR_'.

- **CRAY or Not-CRAY Computer:** Depending on the type of computer it is sometimes necessary to specify statements. Therefore, PIPE distinguishes between directives used by CRAY and non-CRAY computers. If a directive is used for a CRAY then the keyword is '_CRADIR_'. Currently, other directives don't appear. However, if it is necessary to add directives for another PVP-machine then [precomp.f] already has the proper structure to allow further kind of directives.

- **Choice of the Grid:** There exists the possibility to use pre-defined grids as for T21, T32, T42, T63, T84 and T106 Gaussian grid. Alternatively, one may define a self-defined grid (see Table 4.11). The precompiler either activates or disables statements that are required for one of the choices. The keywords are '_MYGRID_' for self-defined grid, '_T21MOD_' for the T21-grid, '_T32MOD_' for the T32-grid, '_T42MOD_' for the T42-grid, '_T63MOD_' for the T63-grid, '_T84MOD_' for the T84-grid and '_T16MOD_' for the T106-grid.

- **Cleaning from Disabled Codes:** To make the code more readable it is possible to delete all disabled statements while the new code is created. Note however that all keywords vanish as well.

Further control words occur in the code that are not touched by [precomp.f]. These are '_OPTION_' to mark optional changes in the code and '_DANGER_' for statements that are crucial for the model numerics. [precomp.f] is called by the script [setup]. These UNIX-scripts have to be edited once to define the model setup (see comments in [setup]). In addition they change the dimensions of the arrays by the batch editor *sed* (see further comments on how to

specify the array dimensions). Finally, these scripts generate the code for a certain computer and with a specified dimension.

5.2.2 Defining the Grid

If one of the T21, T32, T42, T63, T84 or T106 grids is chosen, the work is already done by [precomp.f]. Only the proper dimension have to be chosen (see section 5.3.1). If the grid is self-defined, then the corresponding switch in [precomp.f] has to be set accordingly. The model margins are then defined by *XGL*, *XGR*, *YGU* and *YGO*. Note that the arrays *X* and *Y* that contain the x- and y-coordinates are defined on velocity points and that *X(2)=XGL*, *X(NX-1)=XGR*, *Y(2)=YGU* and *Y(NY-1)=YGO*.

5.2.3 Specification of the Dimensions

For the case of a constant grid spacing Δx and Δy, the following procedure is used to determine the dimensions NX and NY:

$$NX = \frac{XGR - XGL}{\Delta x} + 2 \tag{5.1}$$

$$NY = \frac{YGO - YGU}{\Delta y} + 2 \tag{5.2}$$

For the purpose of specifying the boundary conditions there is always an extra grid point to the east, west, south and north of the model domain. It also must be ensured that the focus is switched off, first! To avoid bank conflicts on the CRAY, there are two limitations:

1. The number of layers *NZ* must be an odd integer,

2. The number of latitudes *NY* must be chosen in such a way that the FORTRAN expression *NV=(NY-1)/2* yields an odd value for *NV*.

Other computers may have other restrictions. Performance on Cache-type computers suffers mainly through the large address increments of non-contiguous vectors. In most cases NZ is the address increment. If the model is implemented on a cache-based, i.e. RISC, processor then alternative formulations ensure that the address increments are contiguous. This is ensured during the installation of PIPE as a RISC-version.

5.2.4 Defining a Focus

A focus is an area of enhanced resolution. The code allows for any definition of the x- and y-coordinates as long as the values increase monotonically. For numerical stability, however, it is necessary to change the grid distance as smooth as possible. This means that the ratio of the left-hand and right-hand side grid distance should be nearly unity. This is ensured to some extent by a spline technique to compute the coordinates. The protocol [OUTLIST] provides the x- and y-coordinates and the relative changes of the grid distances when the model initializes itself. The focus parameters and the dimensions for NX and NY should be changed if the relative changes become too big at one grid point. The procedure to define a focus is the following:

1. Define the model margins and compute the required dimensions to obtain a certain grid distance (see 5.2.3). This yields a grid distance that will be used outside the focus area.

2. Choose the number of grid points that will additionally be added to the grid in the region of higher resolution (the focus area). The final dimensions are:

$$NX = \frac{XGR - XGL}{\Delta x} + 2 + 2 * IDENS \tag{5.3}$$

$$NY = \frac{YGR - YGL}{\Delta y} + 2 + 2 * JDENS \tag{5.4}$$

where now Δx and Δy are the grid distances outside the focus and *IDENS* and *JDENS* are the number of grid points used to increase the resolution on both sides of the focus center. In the next step the index for the focus center has to be determined. If *XGC* and *YGC* are the x- and y-locations of the focus center, *ICENTER* and *JCENTER* are given by

$$ICENTER = \frac{XGC - XGL}{\Delta x} + 1 + IDENS \tag{5.5}$$

$$JCENTER = \frac{YGC - YGU}{\Delta y} + 1 + JDENS \tag{5.6}$$

If ΔX and ΔY are the total widths of the focus in the x- and y-direction respectively, then the widths in terms of index range are given by

$$IBAND = \frac{2 * \Delta X}{XGR - XGL} \tag{5.7}$$

$$JBAND = \frac{2 * \Delta Y}{YGO - YGU} \tag{5.8}$$

3. Choose the refinement factors *XCOM* and *YCOM* for the x- and y-direction, respectively. These represent the ratio between the grid distance outside the focus and in the focus center.

The focus is switched off if *IDENS=JDENS=0*, *IBAND=JBAND=0* and *XCOM=YCOM=1*. Note that the choices for *IBAND* and *XCOM*, and for *JBAND* and *YCOM* depend on each other.

5.2.5 Tuning Topography and Coastline

If the model initializes itself during the first model run (see below), there will possibly be sea points that should be declared as land points and vice versa. With the switches *KSW1-KSW20* one is able to treat the standard problems like closing the Panama channel, etc. These switches activate pre-defined land bridges or channels. Undesired lakes or decoupled parts of the world ocean in a regional model can be deleted by the eliminating points that define array indices. The procedure is such that all sea points around an eliminating point are changed to a land point until all sea points within a connected area are redefined. If the eliminating points are improperly chosen it can happen that all sea points vanish. That will result in an overflow during divides by the number of sea points. Therefore, the first run to initialize the model should be performed with *KSW1-KSW20* and *KILL1I-KILL6J* set to zero. Then these switches have no effect. An explicit way to change the land-sea mask is to put statements into <FLAGMOD> that redefine certain array elements of *IFLG*, which carries the land-sea mask on vector points.

An alternative option to control the assignment of land or sea to a grid point is to save the land-sea mask during the first initialization. For this use *ISW45=1*. Then the file [LSMASK], which contains the land-sea distribution on scalar points, can be edited. After having set

ISW45=2, the file [LSMASK] is read and used to overwrite the internally initialized land-sea mask. This switch setting also must be used if one of the T21- through T106-land-sea masks are used in connection with the according pre-defined grid (see section 5.3.1).

In order to prepare the orography on land to allow the river runoff model to produce meaningfull results, lakes i.e. depressions that are not connected with the ocean, are eliminated. Furthermore, the standard deviation of the fine-resolution raw data is used to properly define rivers.

5.2.6 Selecting the Forcing Data

As explained later in detail, the model offers the choice to force the model with either the COADS data or ECMWF analysis. Two topography data sets are offered, one with 1 degree and the other with 5 minutes resolution. The choice of the topography data set depends on whether the roughest possible topography should be used in an experiment. The COADS data set can be used for regional models as for the North Atlantic. The drawback of the COADS is that they are not spatially complete. Even if the model fills up holes within a data set by artificial but physical meaningfull data, a global model should use the ECMWF data. The chosen data must be defined as switches in the code (see table 4.21) as well as switches in the batch job [model.sun] or [model.cri] that runs the model. The data format in general is '20I4'. For details see references given in 5.4.3.1.

5.2.7 Defining the Output

The model delivers the following files:

1. File [**OCDAT**] contains all data required to restart the model from an earlier state. It uses binary format, although an ASCII file can alternatively be created (see switch *ISW31*). During the start, the model is able to read the binary or the formatted version of [OCDAT]. The data format is recognized automatically.

2. File [**TRDAT**] holds the tracer distribution for restarting the tracer model like [OCDAT] holds the distribution of the dynamical quantities. [TRDAT] is generated at the end of the initialization and is used for restarting the tracer model.

3. File [**FORCES**] contains all monthly mean atmospheric forcing data as well as the monthly mean SST and SSS on the model grid. It is automatically created from the chosen global data sets of atmospheric forcing, sea surface temperature and sea surface salinity when [OCDAT] is found to be empty. However, it must be ensured that all required global data are available (see [model.sun] or [model.cri]). Depending on $ISW42$ [FORCES] also can be created if an [OCDAT] exists or it need not be created despite [OCDAT] being empty. This is useful if [FORCES] has already been created but if one wants to start the ocean state from initial conditions.

4. File [**OBNDSEC**] holds the boundary data, i.e. the temperatures, salinities and barotropic transports, along the xz- and yz-sections on the model boundaries. If the tide model is used, also tidal sea level anomalies from data are saved.

5. File [**TRAFORCE**] contains data for an offline running tracer model which is basically built around the routine <TRACPAS>. The tracer model obtains flow and isopycnal data from the file [TRAFORCE] as monthly forcing data.

6. File [**PPSF_ALL**] is the history file which is created depending on the switch *ISW17*. Other files with an initial [PPSF_...] are created if one of the switches *ISWPOST* is set (for details see Quick-Look System).

5.3 Examples for Specific Model Layouts

In this section it is demonstrated how simple it is to setup PIPE for any geometry and forcing. Starting with any model version of PIPE, the code preprocessor [precomp.f] has to be called by [setup] to ensure that the correct grid type (pre- or self-defined) is enabled. Also dimensions have to be specified. The final changes only concern [ocemain.f]. The definition of numerical parameters such as the time step, or physical parameters, such as the horizontal diffusion coefficient, have to be chosen according to the application and numerical limitations. High resolution grids require some adjustments.

5.3.1 Pre-defined Grids

In addition to the rules to set up a model version, the following parameter setting is required or recommended:

1. One of the T21-, T32-, T42-, T63-, T84- or T106-grids has to be enabled by [setup].
2. Ensure that the following dimensions have been chosen:
   - **T21-Grid:** *NX=66* and *NY=34*.
   - **T32-Grid:** *NX=98* and *NY=50*.
   - **T42-Grid:** *NX=130* and *NY=66*.
   - **T63-Grid:** *NX=194* and *NY=98*.
   - **T84-Grid:** *NX=258* and *NY=130*.
   - **T106-Grid:** *NX=322* and *NY=162*.
3. Use *ISW45=2* in order to use the according land-sea mask
4. Open cyclic boundary conditions must be enabled by *ISW15=1*.
5. It is recommended to set *ISW1=1*, *ISW5=1*, *ISW8=1*, *ISW13=1*, *ISW18=0*, *ISW19=0*, *ISW41=0*.
6. Switch full physics on.
7. Ensure *ISW48=0*.
8. Ensure that Eulerian angles are switched off.
9. Choose optionally an open North Pole with *ISW28=1* or *ISW28=2*.

Note that the variables for defining the model margins *XGL*, *XGR*, *YGU* and *YGO* are not used when one of the T21, T32, T42, T63, T84 or T106 grids is activated. However, a focus can be included. But then the dimensions for *NX* or/and *NY* must be increased according to *IDENS* or/and *JDENS*.

5.3.2 Self-defined Grid

The only difference from the pre-defined grid is that now *XGL*, *XGR*, *YGU* and *YGO* can be chosen arbitrarily. The switches *ISW15* for the cyclic boundary condition and $ISW28$ must be chosen according to the application.

5.3.3 Box-Model

A model that is driven by artificial data can be realized by setting *ISW1=0*, *ISW5=0*, *ISW8=0* and *ISW41=1*. Then the model expects the topography definition from <DEFTOPO>, the atmospheric forcing from <DEFORCE> and uses an initial state defined by *TEMMEAN* and *SALMEAN*.

5.4 Running PIPE

In order to run the model a number of batch jobs and scripts are available. All these scripts are written for UNICOS CRAY or SUNOS operating systems. Before using them adjust the directory path names to your account. The script files are:

1. **[setup]** is required to generate the code for a specific computer with the help of the precompiler [precomp.f] and changes the initial model dimensions to the desired values for *NX*, *NY*, *NZ*, *NXOLD* and *NYOLD*. For the latter two parameters see also Multi-Grid Method.

2. **[make.sun]** or **[make.cri]** generates the object files for [ocestep.f], [oceproc.f], [ocepost.f], [ocemods.f] and [oceplot.f] and saves them in the prescribed directory. The script contains switches for how the FORTRAN codes should be compiled, e.g., for autotasking, debugging or performance tests.

3. **[model.sun] or [model.cri]** runs the model. A number of switches control, e.g., which data have to be used for initialization, whether the model is started from scratch, which output files are saved, etc.

4. **[plot]** is an example for how to create plots. It plots data written onto one of the postprocessing files whose names start by default with [PPSF_...].

5.4.1 How to Generate an Executable Model Version

The grid dimensions *NX*, *NY*, *NZ*, *NXOLD* and *NYOLD* have to be determined before the final FORTRAN source code can be compiled. In the next step, [setup] creates the FORTRAN code. The batch job [model.sun] or [model.cri] requires at least object files of [ocestep.f], [oceproc.f], [ocepost.f] and optionally [ocemods.f]. These files are created by [make.sun] or [make.cri]. After having adjusted all control parameters in [ocemain.f] the entire source code is ready for compilation. In case any modification of some routine is necessary, save it onto [ocemods.f].

5.4.2 Start from Scratch

There are several steps to develop a model version that can be restarted and used for production:

1. **Grid Check:** The purpose of the first run is to verify that the grid definition was carried out properly. For that run, set $ISW11 = 1$ in [ocemain.f] when starting [model.sun] or

[model.cri] for the first time. Since the x- and y-coordinates as well as the topography are initialized first, $ISW11$ allows the model to stop after short a CPU-time. Before that, it saves [PPSF_TOP] if $SAVEPOST = 1$ in [model.sun] or [model.cri]. Ensure that $TOPFORC$ in [model.sun] or [model.cri] is consistently defined with $ISW38$. The [OUTLIST] contains the x- and y-coordinates as well as the relative changes of the grid spacing. With the help of this listing the model margins and the focus parameters can be optimized. For this run also set *AGEDATA=0*, *UPDATED=0*, *INITIAL=1* and *SAVFORC=0*.

2. **Grid Manipulation:** After having obtain a first guess for the land-sea mask it is commonly necessary to modify the land-sea mask. The earlier still available mechanism is to use the tools which are accessible through switches defined in [ocemaon.f]. Meanwhile the switch *ISW45* allows a better control of the land-sea mask, which is recommended.

3. **Model Initialization:** After setting *ISW11=0*, *SAVFORC=1*, *UPDATED=1* and having chosen *VELFORC*, *TEMFORC* and *SSTFORC* consistent with *ISW30*, *ISW43* and *ISW47*, run [model.sun] or [model.cri]. After completion of this short run it delivers [FORCES] and [OCDAT] as well as all quick-look files.

4. **Continuation Run:** After having set *INITIAL=0*, *SAVFORC=0*, *AGEDATA=1* and *UPDATED=2* the run can be continued from the previously computed state saved on [OCDAT]. Note that if [OCDAT] is empty, the model tries to initialize the topography. Also note that the initial time step $DTMIN$ should be small. The model will increase the time step according to an algorithm that uses the convergence of the wave equation as measure. The model can be stopped at a certain date by specifying *NYEAR*, *NMONTH*, etc.

5. **Production Run:** The easiest way to synchronize the data output with the time stepping is to choose *DTMIN=DTMAX*. This ensures that the model does not vary the time step according to some numerical measure. However, it must be ensured that the model works properly for the entire production run with these chosen values. With MAXREP the user can specify the integration time per job run.

5.4.3 Data Preprocessing

In order to avoid extra work for computing the forcing for individual model layouts, all forcing data are held only as global fields. Depending on the chosen grid, the model creates its forcing during initialization and saves its individual forcing on [FORCES]. Thus, no work has to be done to create a model forcing by hand, except for exotic demands.

5.4.3.1 Data Bank

Next, it is briefly outlined how to read all the data files. The $FORTRAN$ statements neither contain any interpolation to the model grid nor do they show how undefined data points are treated.

5.4.3.1.1 Ocean Initialization

- **File [TOP1DEG]** contains the so-called Scripps topography on a $1^o \times 1^o$ grid. Extra latitudes for the South and North Pole have been included. It is read by <READTOP> from the local file [TOPOG] with the following simplified statements:

```
DIMENSION IDEPTH(360),DEPTH(360,180)
DO J=1,180
  READ(*,'(1X,12(I5,1X))') (IDEPTH(I),I=1,360)
  DO I=1,360
    DEPTH(I,J)=HOTAL-IDEPTH(I)
  ENDDO
ENDDO
```

- **File [TOP5MIN]** contains the NOAA topography which is based on a 5 minutes resolution. It is read by <READTOF> from the local file [TOPOG] with the following simplified statements:

```
DIMENSION IDEPTH(4320),DEPTH(4320,2160)
DO J=1,2160
  READ(*,'(12I6)') (IDEPTH(I),I=1,4320)
  DO I=1,4320
    DEPTH(I,J,MONTH)=HOTAL-IDEPTH(I)
  ENDDO
ENDDO
```

- **File [SALTEMP]** contains the monthly mean temperature and salinity of Levitus for the global ocean on a $1^o \times 1^o$ grid. It is read by <INTERPO> with the following simplified statements:

```
DIMENSION ITEMP(32),ISALT(32),TEMP(360,180,32),SALT(360,180,32)
DO MONTH=1,12
  DO J=1,180
    DO I=1,360
      READ(*,'(32I4)') (ISALT(K),K=1,32)
      READ(*,'(32I4)') (ITEMP(K),K=1,32)
      DO K=1,32
        TEMP(I,J)=ITEMP(K)/100.+273.16
        SALT(I,J)=ISALT(K)/100.
      ENDDO
    ENDDO
  ENDDO
ENDDO
```

- **File [SSSALT]** contains the monthly mean sea surface salinity from Levitus on a $1^o \times 1^o$ grid. It is read by <SEASALT> with the following simplified statements:

```
DIMENSION ISSS(360),SSS(360,180,12)
DO MONTH=1,12
  DO J=1,180
    READ(*,'(20I4)') (ISSS(I),I=1,360)
    DO I=1,360
      SSS(I,J,MONTH)=ISSS(I)/100.+35.
    ENDDO
  ENDDO
ENDDO
```

- **File [TEMP.MONTH]** contains the monthly mean sea surface temperature from Levitus on a $1^o \times 1^o$ grid. It is read by <SEATEMP> from the local file [SSTEMP] with the following simplified statements:

```
DIMENSION ISSS(360),SSS(360,180,12)
DO MONTH=1,12
  DO J=1,180
    READ(*,'(20I4)') (ISSS(I),I=1,360)
    DO I=1,360
      SSS(I,J,MONTH)=ISSS(I)/100.+35.
    ENDDO
  ENDDO
ENDDO
```

- **File [SALT.MONTH]** contains the monthly mean sea surface salinity from Levitus on a $1^o \times 1^o$ grid. It is read by <SEASALT> with the following simplified statements:

```
DIMENSION ISSS(360),SSS(360,180,12)
DO MONTH=1,12
  DO J=1,180
    READ(*,'(20I4)') (ISSS(I),I=1,360)
    DO I=1,360
      SSS(I,J,MONTH)=ISSS(I)/100.+35.
    ENDDO
  ENDDO
ENDDO
```

5.4.3.1.2 COADS Data

- **File [SSTGLOB]** contains a merged data set from the COADS and the Reynolds data set on a $2^o \times 2^o$ grid. The Reynolds data are used only if the COADS contain undefined values. It is read by <OCTEMP> from the local file [SSTEMP] with the following simplified statements:

```
DIMENSION ISST(180),SST(180,90,12)
DO MONTH=1,12
  DO J=1,90
    READ(*,'(20I4)') (ISST(I),I=1,180)
    DO I=1,180
      SST(I,J,MONTH)=ISST(I)/100.+273.16
    ENDDO
  ENDDO
ENDDO
```

- **File [AIRGLOB]** contains the monthly mean surface air temperature on a $2^o \times 2^o$ grid. The data are merged from the COADS whenever possible, from the data of Shea (1986) for the Arctic and the Taljaard data for the southern hemisphere. It is read by <ATTEMP> from the local file [AIRTEMP] with the following simplified statements:

```
DIMENSION IAIRT(180),AIRTEMP(180,90,12)
DO MONTH=1,12
  DO J=1,90
    READ(*,'(20I4)') (IAIRT(I),I=1,180)
    DO I=1,180
      AIRTEMP(I,J,MONTH)=IAIRT(I)/100.-60.+273.16
    ENDDO
  ENDDO
ENDDO
```

- **File [COVER]** contains the monthly mean fractional cloud cover from COADS on a $2^o \times 2^o$ grid. It is read by <CUMULUS> with the following simplified statements:

```
DIMENSION ICLOUDS(180),CLOUDS(180,90,12)
DO MONTH=1,12
  DO J=1,90
    READ(*,'(20I4)') (ICLOUDS(I),I=1,180)
    DO I=1,180
      CLOUDS(I,J,MONTH)=ICLOUDS(I)/80.
    ENDDO
  ENDDO
ENDDO
```

- **File [WETNESS]** contains the monthly mean relative humidity from COADS on a $2^o \times 2^o$ grid. It is read by <VAPOR> with the following simplified statements:

```
DIMENSION IVAPOR(180),VAPOR(180,90,12)
DO MONTH=1,12
  DO J=1,90
    READ(*,'(20I4)') (IVAPOR(I),I=1,180)
    DO I=1,180
      VAPOR(I,J,MONTH)=IVAPOR(I)/1000.
    ENDDO
  ENDDO
ENDDO
```

- **File [PRESURF]** contains the monthly mean sea level pressure from COADS on a $2^o \times 2^o$ grid. This file is not used.

- **File [UVGLOB]:** It contains the monthly mean surface wind stress and its annual mean from Hellermann and Rosenstein as pseudo wind stresses on a $2^o \times 2^o$ grid. The first block of data is the annual mean, the second one starts with January. It is read by <WIND> with the following simplified statements:

```
DIMENSION ITAUX(180),ITAUY(180),TAUX(180,72,13),TAUY(180,72,13)
DO MONTH=1,13
  DO J=1,90
    READ(*,'(20I4)') (ITAUX(I),I=1,180)
    READ(*,'(20I4)') (ITAUY(I),I=1,180)
    DO I=1,180
      TAUXM=(ITAUX(I)-5000.)/5000.
      TAUYM=(ITAUY(I)-5000.)/5000.
      TAUX(I,J,MONTH)=TAUXM*SQRT(TAUXM**2+TAUYM**2)
      TAUY(I,J,MONTH)=TAUYM*SQRT(TAUXM**2+TAUYM**2)
    ENDDO
  ENDDO
ENDDO
```

- **File [UVABS]** contains the monthly mean wind speed from the COADS on a $2^o \times 2^o$ grid. It is read by <ABSOL> with the following simplified statements:

```
DIMENSION IABSOL(180),UVABSOL(180,90,12)
DO MONTH=1,12
  DO J=1,90
```

```
      READ(*,'(20I4)') (IABSOL(I),I=1,180)
      DO I=1,180
        UVABSOL(I,J,MONTH)=IABSOL(I)/10.
      ENDDO
    ENDDO
  ENDDO
```

- **File [UVDEV]** contains the monthly mean wind speed standard deviation from the COADS on a $2^o \times 2^o$ grid. It is read by <VARIAN> with the following simplified statements:

```
DIMENSION IVARIAN(180),UVARIAN(180,90,12)
DO MONTH=1,12
  DO J=1,90
    READ(*,'(20I4)') (IVARIAN(I),I=1,180)
    DO I=1,180
      UVARIAN(I,J,MONTH)=IVARIAN(I)/100.
    ENDDO
  ENDDO
ENDDO
```

5.4.3.1.3 ECMWF Data

- **File [ECSST]** contains the observed SST data from the ECMWF reanalysis Project on a T106 grid. It is read by <ECSST> from the local file [SSTEMP] with the following simplified statements:

```
DIMENSION ISST(360),SST(360,160,12)
DO MONTH=1,12
  DO J=1,160
    READ(*,'(20I4)') (ISST(I),I=1,360)
    DO I=1,360
      SST(I,J,MONTH)=ISST(I)/200.+273.16
    ENDDO
  ENDDO
ENDDO
```

- **File [ECAIR]** contains surface air temperature data from the ECMWF reanalysis Project on a T106 grid. It is read by <ECTSC> from the local file [AIRTEMP] with the following simplified statements:

```
DIMENSION IAIRT(360),AIRT(360,160,12)
DO MONTH=1,12
  DO J=1,160
    READ(*,'(20I4)') (IAIRT(I),I=1,360)
    DO I=1,360
      AIRT(I,J,MONTH)=IAIRT(I)/80.+203.15
    ENDDO
  ENDDO
ENDDO
```

- **File [ECCLD]** contains the cloudiness from the ECMWF reanalysis Project on a T106 grid. It is read by <ECCLD> from the local file [COVER] with the following simplified statements:

```
DIMENSION ICLD(360),CLD(360,160,12)
DO MONTH=1,12
  DO J=1,160
    READ(*,'(20I4)') (ICLD(I),I=1,360)
    DO I=1,360
      CLD(I,J,MONTH)=ICLD(I)/10000.
    ENDDO
  ENDDO
ENDDO
```

- **File** [**ECHUM**] contains the surface humidity from the ECMWF reanalysis Project on a T106 grid. It is read by <ECHUM> from the local file [WETNESS] with the following simplified statements:

```
DIMENSION IHUM(320),HUM(320,160,12)
DO MONTH=1,12
  DO J=1,160
    READ(*,'(20I4)') (IHUM(I),I=1,320)
    DO I=1,320
      HUM(I,J,MONTH)=IHUM(I)/10000.
    ENDDO
  ENDDO
ENDDO
```

- **File** [**ECSLP**] contains the surface pressure from the ECMWF reanalysis Project on a T106 grid. It is read by <ECSLP> from the local file [MSLP] with the following simplified statements:

```
DIMENSION ISLP(320),SLP(320,160,12)
DO MONTH=1,12
  DO J=1,160
    READ(*,'(20I4)') (ISLP(I),I=1,320)
    DO I=1,320
      SLP(I,J,MONTH)=ISLP(I)+95000.
    ENDDO
  ENDDO
ENDDO
```

- **File** [**ECTAU**] contains the surface wind stress from the ECMWF reanalysis Project on a T106 grid. It is read by <ECTAU> from the local file [UVGLOB] with the following simplified statements:

```
DIMENSION ITAUX(360),ITAUY(360),TAUX(360,160,12),TAUY(3600,160,12)
DO MONTH=1,12
  DO J=1,160
    READ(*,'(20I4)') (ITAUX(I),I=1,360)
    READ(*,'(20I4)') (ITAUY(I),I=1,360)
    DO I=1,144
      TAUX(I,J,MONTH)=(ITAUX(I)-4000.)/5000.
      TAUY(I,J,MONTH)=(ITAUY(I)-4000.)/5000.
    ENDDO
  ENDDO
ENDDO
```

- **File [ECUSC]** contains the surface scalar wind from the ECMWF reanalysis Project on a T106 grid. It is read by <ECUSC> from the local file [UVABS] with the following simplified statements:

```
DIMENSION IUSC(360),USC(360,160,12)
DO MONTH=1,12
  DO J=1,160
    READ(*,'(20I4)') (IUSC(I),I=1,360)
    DO I=1,360
      USC(I,J,MONTH)=IUSC(I)/500.
    ENDDO
  ENDDO
ENDDO
```

- **File [ECSTD]** contains the surface scalar wind from the ECMWF reanalysis Project on a T106 grid. It is read by <ECSTD> from the local file [UVDEV] with the following simplified statements:

```
DIMENSION ISTD(360),STD(360,160,12)
DO MONTH=1,12
  DO J=1,160
    READ(*,'(20I4)') (ISTD(I),I=1,360)
    DO I=1,360
      STD(I,J,MONTH)=ISTD(I)/1000.
    ENDDO
  ENDDO
ENDDO
```

5.4.3.1.4 Precipitation Data

- **File [RAIN1DEG]** contains the precipitation data set of Legates & Willmott (1990). However, it has been averaged onto a $1^o \times 1^o$ grid from the original $0.5^o \times 0.5^o$ grid. It is read by <LEGATES> from the local file [MONRAIN] with the following simplified statements:

```
DIMENSION IRAIN(360),RAIN(360,180,12)
DO MONTH=1,12
  DO J=1,180
    READ(*,'(20I4)') (IRAIN(I),I=1,360)
    DO I=1,360
      RAIN(I,J,MONTH)=IRAIN(I)/(1000.*2592000.)
    ENDDO
  ENDDO
ENDDO
```

5.4.3.1.5 Land-Sea Masks: In order to make the setup of various global models simpler, a number of readily tuned masks are offered. This is done because certain combinations of land and sea points lead to B-grid splitting. The following masks don't have this problem. Thus a tuning of the land-sea mask is not needed when these masks are taken. These files are read by <FLAGMOD> from the local file [LSMASK]. However, one has to ensure that the script [setup] is consistent with the script [model.sun] or [model.cri] which has to copy the corresponding grid-file onto the local file [LSMASK]. The available pre-defined grids are:

- **File [T21GRID]** has dimension $(64, 32)$
- **File [T32GRID]** has dimension $(96, 48)$
- **File [T42GRID]** has dimension $(128, 64)$
- **File [T63GRID]** has dimension $(192, 96)$
- **File [T84GRID]** has dimension $(256, 128)$
- **File [T106GRID]** has dimension $(320, 160)$

5.4.3.1.6 Open Boundary Forcing

- **File [SALTEMP.JAN]** is the 3-dimensional observed temperature and salinity for January. Format is identical with [SALTEMP]. The local file is [STMON01].
- **File [SALTEMP.FEB]** is the 3-dimensional observed temperature and salinity for February. Format is identical with [SALTEMP]. The local file is [STMON02].
- **File [SALTEMP.MAR]** is the 3-dimensional observed temperature and salinity for March. Format is identical with [SALTEMP]. The local file is [STMON03].
- **File [SALTEMP.APR]** is the 3-dimensional observed temperature and salinity for April. Format is identical with [SALTEMP]. The local file is [STMON04].
- **File [SALTEMP.MAY]** is the 3-dimensional observed temperature and salinity for May. Format is identical with [SALTEMP]. The local file is [STMON05].
- **File [SALTEMP.JUN]** is the 3-dimensional observed temperature and salinity for June. Format is identical with [SALTEMP]. The local file is [STMON06].
- **File [SALTEMP.JUL]** is the 3-dimensional observed temperature and salinity for July. Format is identical with [SALTEMP]. The local file is [STMON07].
- **File [SALTEMP.AUG]** is the 3-dimensional observed temperature and salinity for August. Format is identical with [SALTEMP]. The local file is [STMON08].
- **File [SALTEMP.SEP]** is the 3-dimensional observed temperature and salinity for September. Format is identical with [SALTEMP]. The local file is [STMON09].
- **File [SALTEMP.OCT]** is the 3-dimensional observed temperature and salinity for October. Format is identical with [SALTEMP]. The local file is [STMON10].
- **File [SALTEMP.NOV]** is the 3-dimensional observed temperature and salinity for November. Format is identical with [SALTEMP]. The local file is [STMON11].
- **File [SALTEMP.DEC]** is the 3-dimensional observed temperature and salinity for December. Format is identical with [SALTEMP]. The local file is [STMON12].
- **File [TIDES.ASS]** contains the observed tide data. The file is read by <INTIDES>. The local file is [TIDES].
- **File [BAROFLOW]** contains the barotropic transport computed with a T106-version of the PIPE model. The file is read by <SELECOBND> and <STREAM_T106>. The local file is [BAROFLOW].

5.4.3.2 Model Forcing

The model forcing is derived from the global data set by bilinear interpolation onto the model grid and is saved in [FORCES]. The global data sets are required only once. A continuation run needs only a restart file [OCDAT] and a forcing file [FORCES]. If other data sets should be used, there are two choices:

1. If the data are monthly means, then the responsible routine for interpolating that quantity onto the model grid can be adjusted for the other data set.

2. If the data are not monthly means, then the simplest way is to use the coupling interface routines.

If open boundary conditions are used, the file [OBNDSEC] is needed. If the tracer model is used, the file [TRDAT] is needed in addition. If the tracer model is running in offline mode then also file [TRAFORCE] is needed.

5.4.4 The Multi-Grid Method

The model allows the user to jump between different grids within a model run. For instance, this can be useful if a spin-up run is carried out with a coarse resolution, but the production experiments require higher resolution. Since the coarse resolution model can be run to a stationary state much faster than a fine resolution model, this provides a convenient way to save CPU-time. Before the model is compiled the variables $NXOLD$ and $NYOLD$ must be set to the dimensions of NX and NY, that have been used for the coarse grid. This can be defined in [setup] while creating the model source code. Note that currently there is the restriction that the vertical dimension NZ of the old and new model grid must be identical. However, widely untested modifications already exist for allowing to decrease or increase *NZ*. Simply try and if necessary debug <OLDINIT>. In order to use this method, initialize the model for the new model grid as explained above. However, ensure that the [OCDAT] of the old run is saved as [OLDFILE] on the data directory, that the parameter $JUMGRID = 1$ in [model.sun] or [model.cri] and that $ISW12 = 1$ in [ocemain.f]. This causes the code to interpolate the old grid onto the new grid and onto the new topography. If there are no data available from [OLDFILE] for interpolation, the initial data on these points are taken from Levitus. Also ensure that the initial model time defined by $KTOTAL$ agrees with the model time of the old restart file [OLDFILE]. Otherwise, a discontinuity of the forcing between the model time of [OLDFILE] and the model time during the initialization may result. Note that a proper definition of $KTOTAL$ allows the definition of the model time during initialization in general.

5.4.5 Diagnostic Output

During the model run, a number of formatted files appear that allow a quick check of the model physics. These are:

1. File [**OUTLIST**] contains the protocol of the model. After each time step, crucial numbers (mainly for monitoring stability properties) are written out. See description in [OUTLIST].

2. File [**LSMASK**] contains the land-sea mask written out during the initialization of the topography, if *ISW45=1*. This file can be read again and is used to overwrite the land-sea mask during a second topography initialization, if *ISW45=2*.

3. File [**MEANS**] contains layer averages for temperature, salinity, in situ density and potential density. These data are written out by <STATIST> at each time step. It is called by <OCESTEP>.

4. File [**PERFORM**] contains an overview of CPU-times from the most CPU-time consuming routines.

5. File [**BUDGET**] contains the heat and fresh water budget as well as conservation errors for each time step. The data are written out from <OCESTEP>.

5.4.6 How to Apply the Coupling Interface

The following example shows how to introduce data from outside the ocean model, without making any changes to the bulk of the code. The listed changes apply only for <MY_OGCM>. Assume that data with daily resolution are prepared on [FORCING] with the sequence as defined in the example below and that the data are already interpolated to the model grid. Furthermore, assume that the model run should cover one month of prediction and uses a time step of a day. In this case set $MAXREP = 2$ and $DTMAX = 43200.$, in order to sychronize the data flow from [FORCING] with the model time stepping. After having set $NINIT = 1$, the parameters $NHEAT$, $NFRESH$, $NSTRESS$ and $NMIX$ must be chosen dependent on whether the forcing is stored as absolute value or as anomaly on [FORCING]. Note that the ice coupling can only be switched on if the snow and ice surface temperature are consistently computed from the model snow and ice cover. This can only be ensured in a coupled AGCM/OGCM where the AGCM uses the same algorithm to compute the snow and ice temperature as <SURFLUX>. The example is as follows.

```
    .
    .
PARAMETER (NX=130,NY=63)
DIMENSION USTERN(NX,NY),TAUX(NX,NY)  ,TAUY(NX,NY)
DIMENSION SOLAR(NX,NY) ,FRESH(NX,NY),HEAT(NX,NY)
CALL OCEINIT(MSEC,MINUTE,MHOUR,MDAY,MONTH,MYEAR)
OPEN(90,FILE='FORCING',FORM='UNFORMATTED')
DO LREP=1,30
  READ(90) ((USTERN(I,J),I=1,NX),J=1,NY)
  READ(90) ((TAUX  (I,J),I=1,NX),J=1,NY)
  READ(90) ((TAUY  (I,J),I=1,NX),J=1,NY)
  READ(90) ((SOLAR (I,J),I=1,NX),J=1,NY)
  READ(90) ((FRESH (I,J),I=1,NX),J=1,NY)
  READ(90) ((HEAT  (I,J),I=1,NX),J=1,NY)
  CALL AOUSTAR(USTERN)
  CALL AOTAU  (TAUX,TAUY)
  CALL AOQFLX (HEAT,SOLAR)
  CALL AOPME  (FRESH)
  CALL OCESTEP(MSEC,MINUTE,MHOUR,MDAY,MONTH,MYEAR)
ENDDO
CLOSE(90)
CALL OCEPOST
CALL OCESTOP
    .
    .
```

In this style all routines can be called which start with AO (stands for atmosphere to ocean) for importing data into the ocean model as well as all those routines which start with OA (stands

for ocean to atmosphere) for exporting data out of the ocean model. The latter must be called after <OCESTEP>.

5.4.7 How to Use the Bias Correction

Bias correction is a technique that has been developed for recent coupled experiments with ECHAM4 and OPYC3. In opposite to flux correction based on monthly means bias correction means that an annual mean correction for heat, fresh water and stress is applied. In order to use this technique the model has to pass three steps. These are for

1. spinning up the model with full feedbacks for temperature and salinity. During the first period the switches *NHEAT*, *NFRESH* and *NSTRS* must be set to unity, as well as *KHEAT=1*, *KFRESH=1* and *KSTRESS=1*. This means that the model expects fluxes from e.g. data or an atmospheric model that are fed through the coupling interface. If *ISW48=1*, then the bias correction is computed as temporal average, however, because model physics is nonlinear, the resulting bias correction is not exactly that which is expected when Newtonian feedbacks are finally switched off.

2. learning what the bias correction is for zero feedbacks. Setting *ISW48=2* reduces the Newtonian feedback continuously until it is negligible.

3. freezing the bias correction and starting the production experiment.

During the first and second updating process at each time step the following recursion is used to improve the bias correction:

$$Q_{run}^{l+1} = (Q_{run}^{l}(1-l) + Q_1)/l \tag{5.9}$$
$$F_{run}^{l+1} = (F_{run}^{l}(1-l) + F_1)/l \tag{5.10}$$
$$\tau_{x,run}^{l+1} = (\tau_{x,run}^{l}(1-l) + \tau_x)/l \tag{5.11}$$
$$\tau_{y,run}^{l+1} = (\tau_{y,run}^{l}(1-l) + \tau_y)/l \tag{5.12}$$

After each completed year the variables Q_{bias}, F_{bias}, $\tau_{x,bias}$ and $\tau_{y,bias}$ are set to their respective current annual means Q_{run}, F_{run}, $\tau_{x,run}$ and $\tau_{y,run}$. From the correction terms the final fluxes are computed by

$$Q_1 = Q_1^* + Q_{bias}(1-\delta_{run}) + (T - T_{obs})\delta_T\delta_{run} \tag{5.13}$$
$$F_1 = F_1^* + F_{bias}(1-\delta_{run}) + (S - S_{obs})\delta_S\delta_{run} \tag{5.14}$$
$$\tau_x = \tau_x^* + \tau_{x,bias} \tag{5.15}$$
$$\tau_y = \tau_y^* + \tau_{y,bias} \tag{5.16}$$

where δ_{run} is a variable (see <SURFLUX>) that decays from unity to zero during the updating process, T and S are the model's surface temperature and salinity with T_{obs} and S_{obs} their respective observed fields, δ_T and δ_S are feedback coefficients that induce a relaxation towards the observed fields, at least in the annual mean, Q_1^* is the preliminary heat flux computed from radiative and turbulent fluxes, F^* is the preliminary fresh water flux as computed through relaxation and $(P - E + R)$, and τ_x^* and τ_y^* are wind stresses either from e.g. an atmospheric model or from data fed through the coupling interface.

Unfortunately, the method is not as easy to apply than it reads. The difficulty is that during the second step, the updating process for the bias correction may produce oscillations such that the bias correction does not converge monotonously to some final field. In order to save the situation it has been found that an EOF-analysis of the bias correction fields yields PCs for

the mean trend, which decay continuously, and further PCs that contain the oscillations. If the trend-EOF is used to reconstruct the time series for the bias correction, then the oscillations are cancelled. In a final step, the arrays *HEATM*, *PMEM* and *(TAUXM,TAUYM)* are overwritten in the file [OCDAT] before the production experiment starts.

Alternatively, one may use an exponential fit of the time series for the bias corrections in order to determine the stationary corrections in the limit-case of an infinite long spinup run. This method was used for the successfull experiments by Bacher et al. (1998).

5.4.8 About Numerical Stability of PIPE

The CFL-number does not provide a necessary condition for numerical stability as PIPE is a highly nonlinear model. Furthermore, e.g. stability properties of implicit schemes are based on the assumption that a solution of any either iteratively or directly solved equation is found exactly. In practice this is never the case. Any iterative algorithm leaves a residual error that can lead to instability. Based on the experience of the author a few suggestions on how to set the numerical parameters properly should be given. To visualize the models behaviour the following lines appear in the protocol [OUTLIST]:

```
+-----------> Wave Equation                  >-+
| +---------> Momentum Advection & Diffusion  |
| | +-------> Temperature Equation            |
| | | +-----> Salinity Equation                +-> Iterations
| | | | +---> Ice Thickness & Concentration    |
| | | | | +-> Ice Momentum                   >-+
| | | | | |
| | | | | |   +---> Used Time Step
| | | | | |   |     +---> Mean Explicit Acceleration
| | | | | |   |     |     +---> Computed Change of Flux
| | | | | |   |     |     |     +---> Computed Change of H
| | | | | |   |     |     |     |      +---> Total Momentum
| | | | | |   |     |     |     |      |
3 3 3 3 3 3 0.500  865.  166.  281.  1759. 174 -5.9 (11,071,006)  47  85  2976
3 3 3 3 3 3 0.500  904.  182.  282.  1760. 169 -5.9 (11,071,006)  75  55  2934
3 3 3 3 3 3 0.500  935.  179.  280.  1761. 170 -5.9 (11,071,006)  76  95  3009
3 3 3 3 3 3 0.500  924.  175.  279.  1763. 170 -5.9 (11,071,006)  79  71  2984
3 3 3 3 3 3 0.500  935.  182.  278.  1765. 168 -5.9 (11,071,006)  65  91  2918
                                          |   |   |        |    |   |     |
                 Found negative H Points  <-+   |   +----+---+    |   |     |
                 Maximal found negative H <-----+        |        |   |     |
                 (Z,X,Y) Index of Point   <--------------+        |   |     |
                                  Number of Cells being opened <-+   |     |
                                  Number of Cells being closed <-----+     |
                                  Number of total Convection Events <------+
```

Each time step writes a line of critical numbers into [OUTLIST]. These numbers are documented in [OUTLIST] as in the example that shows five time steps. The numbers appearing in rows 1-6 denote used iterations within the major iterative problems. The smallest number that the model has to take is defined by *NITMIN*. Each of the algorithms decides by itself whether more iterations are necessary. The required accuracy can be adjusted with the variables *ACCUR...*. If the maximal number of iterations defined by *NITMAX* is exceeded the model diagnoses a numerical problem. Depending on the switch *ISW35*, PIPE either creates an error exit or stops the code without saving any state. Note that the shown numbers always are half of what is really used as number of iterations. Row 7 shows the time step that the model has

chosen. If *DTMAX=DTMIN* then the model does not try to vary the time step, however, if *DTMIN<DTMAX* then the model searches that maximal time step that still ensures a sufficient convergence of all iterative algorithms. In the case that the model suddenly blows up, the model reduces the time step and retries the same time step once more in the hope that the convergence conditions are satisfied by the reduced time step. Row 8 is the average of the absolute value of the acceleration over each grid point. It is a sensitive measure for numerical noise. Any instability will immediately appear as big number. These are slightly problem-depdendent, but must never exceed values of about *5000*. The following two rows (9 and 10) show the mean of the absolute values of the computed change of the flux and layer thickness. The next row shows the total content of momentum. The rows (12-14) are important. Since the model works with time variable boundary conditions, it formally may happen that the layer thickness becomes negative due to a too strong divergence at a grid cell that has already lost its entire mass. The numbers are extracted from the predictor step. In the corrector steps negative values are cancelled. However, these first guesses are an indication of a too large time step. It is important that even residual negative layer thicknesses are listed in the sum. A rule should be that the negative layer thicknesses must never appear as '****' since this indicates fundamental stability problems. The indices in the order (z,x,y) indicate where the problems are located. Finally, the next two rows list the number of cells that have been opened or closed per time step. This is also some measure for numerical noise. The last row shows how many horizontal grid points experienced a convection event.

5.4.9 Numerical Filters

In order to suppress instabilities, the model has a number of mechanisms to reduce numerical noise. These are:

- The parameter **SMOOTH** enables filtering of all interface heights. This parameter must be small, otherwise one violates the first law of ocean modelling, namely to never touch the pressure field. One can create arbitrary nonsense if *SMOOTH* is too big. However, it should be non-zero in order to suppress B-grid waves in the pressure field.

- The parameters **DIFFUSV** and **DIFMINV** specify the strength of momentum diffusion. Note that momentum diffusion may be strong without severe consequences. It might even cause a physical improvement.

- The parameters **DIADIF0** and **DIADIF1** control the strength of the layer thickness diffusion. A small value is physically justified and like *SMOOTH* is helpfull to remove B-Grid waves from the layer thickness field.

- The parameters **DIFFUSS** and **DIFMINS** are required to smooth the temperature and salinity fields. However, physically meaningfull small values are sufficient to suppress numerical noise.

- The parameter **NITMIN** controls the quality of the iterative solutions. If *NITMIN* is too small and/or the time step too long then residual errors might accumulate over many time steps such that the model blows up finally. The parameter *NITMAX* limits the number of iterations. If the iteration count exceeds *NITMAX* for the wave equation, then an error, i.e. floating-point exception, is created. This error is generated by the routine <SUICIDE>.

5.4.10 Error Conditions

The model contains a number of stop-statements. All of them give an indication on the kind of error. It also may happen that the model exits with a floating-point exception. If the latter happens the possible reasons are:

No Convergence in Wave Equation: As a consequence the routine <SUICIDE> is called in order to interrupt the execution. In this situation the conclusion is that the model had problems in finding clean solutions. In other words the algorithms are overcharged with the combination of physical situation and layout of model grid, choice of time step, external forcing etc.

Model Time Threshold Reached: If the model time counter has reached the model time as defined by *NHOUR*, *NDAY*, *NMONTH* and *NYEAR*, then the model stops independent on the setting of *MAXREP*. If one tries to continue the run, then the routine <SUICIDE> is called, which creates a floating point exception. If you want to continue the run increase the model time thresholds.

Other Floating-Point Exceptions: So far the only known possibilities are that either the model time threshold is reached, however, <SUICIDE> was not called, or that a numerical instability appeared. In the first case, the time step *DT* becomes zero which results in a divide by zero in subsequent routines. The first routine then called is <CROSMIX>. In the second case, the routine <SURFLUX> is the first one called during a time step, which seems to respond sensitively to e.g. erroneous temperatures.

Stop-statement: If an unexpected *stop*-statement is executed follow the instructions. This is a way to educate people that it is worth to read manuals. These *stop*-statements appear in <OCEINIT> in order to issue nonsense switch or parameter settings, and in <RALEIGH> and <BAROTROP> for adjustments that are required for the open boundaries.

5.4.11 Conceptual Problems

PIPE has been developed to simulate the global ocean circulation with isopycnal coordinates. Therefore, a line-relaxation scheme has been chosen in order to deal with converging grids without the need for any spatial strong filters. The disadvantage is that the matrices for tridiagonal or block-tridiagonal linear equation systems have to be setup and solved in an iterative process at each time step. This computationally demanding concept, however, combines the excellent wave propagation properties of a layer model (see Oberhuber et al. 1998) e.g. needed for regional applications with an uncritical treatment of polar problems in a global model (see Roeckner et al. 1996). Fancy grid rotations are not needed in PIPE.

However, due to the use of an implicit time stepping scheme, one has to be aware of specific properties that are important to know. An analytic solution of a linear model solved with an implicit scheme has the following properties:

- The wave speed always is limited by the ratio between wave length and time step. Thus long waves might travel nearly at their theoretical speed while short but theoretically fast waves are slowed down significantly. In consequence, slow modes like baroclinic ones unlikely are slowed down, while fast external modes, i.e. the barotropic modes, are slowed down significantly if they are short. Long barotropic waves are less affected.

- The ratio between the theoretical speed versus the speed allowed by the implicit scheme indirectly appears as contribution to the matrices relative to the main diagonal element

which is created by the time derivative. The consequence is that the equations converge fast for slow modes while they converge much slower for fast waves. This is because for fast waves the equations have the nature of a Laplace-equation, which is known to converge much slower than a Helmholtz-equation.

- Due to the need to iteratively solve the equations for e.g. pressure, the solutions are accurate for short waves but might contain non-negligible errors on large scales. Even the recently introduced ADLR-scheme cannot remedy this problem fully.

As a result of the reduced convergence for large-scale fast modes the transport potentials, which usually is a tiny residual of the flow, might contain non-negligible values. Because errors are large-scale, the local velocity error should always be negligible. Anyway, the transport potential can be used to test the quality of the solution besides the stationarity of the ocean state. To test stationarity analyse annual means only.

5.4.12 About Code Consistency

If one of the terms should be modified which is part of the predictor-corrector scheme then one has to perform the modifications with care. That's because those terms are formulated twice, namely in an explicit and then in an implicit manner. If one changes such a term inconsistently so that its explicit does not fit with its implicit formulation the result is unpredictable. This concerns mainly the routines <MODEXP> and <MODIMP>, and <MIXEXP> and <MIXIMP>. The same is true for the ADLR scheme. If changes in <SOLVERX> are not consistent with those in <SOLVERY>, or <MOLVERX> becomes inconsistent with <MOLVERY> then the model might behave unphysical. Basically, these pairs of routines must solve the identical mathematical problem.

5.5 Postprocessing

The data postprocessing is based on the self-developed PPSF, which means Post-Processing System Format. The goal is to write all key quantities onto a block of data that allows an unique identification of each quantity. For the purpose of quick postprocessing (see quick-look files) data are written out in simple binary format. To allow transfers to other computers via a network like ethernet, a conversion to 32-bit IEEE format is possible (see script [ppsf2i3e]). Another possibility is to compress data to a 3-byte accurate portabe file format (see script [implode]). This allows the user to reduce the file size by a factor of about 3. The major underlying idea is to use several types of codes to identify a quantity, to include the land-sea mask for data analysis and plotting purposes and to include the coordinates that are required to determine the location of each grid point. Postprocessing routines such as the plot program are then independent of any other data source required to analyse the model fields.

5.5.1 Code Definitions

For a complete list of code definitions see block data statements in [ocepict.f] or execute *show*.

5.5.2 PPSF - Binary Format

The following write statements demonstrate how a data block for one quantity is generated:

Code Number	Variable	Unit	Reference
1	layer velocity components	$kgm^{-1}s^{-1}$	
2	layer thickness	m	$HEIGTH$
3	layer potential temperature	K	$TEMP$
4	layer temperature	K	
5	layer salinity	gkg^{-1}	$SALT$
7	layer in situ density	kgm^{-3}	$DENSITY$
8	layer potential density	kgm^{-3}	$POTDENS$
9	layer in situ pressure	$kgs^{-2}m^{-1}$	$PRESS$
17	sea ice velocity components	ms^{-1}	$UICE, VICE$
18	sea ice thickness	m	$HICE$
19	sea ice compactness		$COMPACT$
37	snow surface temperature	K	$TSNOW$
39	snow-ice interface temperature	K	$TICE$
60	snow cover ice equivalent	m	$HSNOW$

Table 5.1: **Definition of Quantity Code: Prognostic Variables**

Code Number	Variable	Unit	Reference
10	level potential density	kgm^{-3}	
11	level velocity components	ms^{-1}	U, V
12	level interface height	m	$HEIGTH$
13	level potential temperature	K	
14	level in situ temperature	K	
15	level salinity	gkg^{-1}	
66	sea surface temperature	K	$TEMP(1, I, J)$
67	mixed layer depth	m	$HEIGTH(1, I, J)$
68	sea surface elevation	m	$HEIGTH(1, I, J)$
69	sea surface velocity components	ms^{-1}	$U(1, I, J)$,$V(1, I, J)$
70	sea surface salinity	gkg^{-1}	$SALT(1, I, J)$

Table 5.2: **Definition of Quantity Code: Level Quantities**

```
      .
      .
 DIMENSION IFLG(N1,N2),X1(N1),X2(N2),FIELD(N1,N2,N3)
 CHARACTER*8 LDATE,LTIME
 CHARACTER*40 EXPERIM
      .
      .
 CALL DATE(LDATE)
 CALL CLOCK(LTIME)
 CALL TIMECON(KTOTAL,MSEC,MINUTE,MHOUR,MDAY,MONTH,MYEAR)
      .
      .
 WRITE(40) KODEQ,KODES,N1,N2,N3,KODEL,KTOTAL,MSEC,MINUTE,MHOUR,MDAY,
$          MONTH,MYEAR,LDATE,LTIME,EXPERIM
 WRITE(40) CHAR(0),CHAR(1),((CHAR(IFLG(I,J)),I=1,N1),J=1,N2)
 WRITE(40) (X1(I),I=1,N1)
 WRITE(40) (X2(J),J=1,N2)
 WRITE(40) (((FIELD(I,J,K),I=1,N1),J=1,N2),K=1,N3)
      .
```

Code Number	Variable	Unit	Reference
20	velocity stream function	$m^2 s^{-1}$	
30	absolute stress	$m^2 s^{-2}$	
31	entrainment/detrainment rate	ms^{-1}	$VERTIC$
33	cross isopycnal mixing	ms^{-1}	$VERTUP, VERTDN$
34	downward bubble salt flux	ms^{-1}	
38	SST error	K	
41	maximal depth of surface convection	m	<QCONV>

Table 5.3: **Definition of Quantity Code: Diagnostic Variables**

Code Number	Variable	Unit	Reference
32	observed sea level pressure	Nm^{-2}	<OCESTEP>
50	topography	m	$DEPTH$
51	observed sea surface temperature	K	SST
52	observed air temperature	K	$AIRTEMP$
53	observed salinity	gkg^{-1}	SSS
54	observed wind stress	$m^2 s^{-2}$	$TAUX, TAUY$
55	observed wind speed	ms^{-1}	$UVABSOL$
56	observed standard deviation	ms^{-1}	$UVDEV$
57	observed humidity		$HUMID$
58	observed cloudiness		$CLOUDS$
59	observed precipitation	ms^{-1}	$RAIN$

Table 5.4: **Definition of Quantity Code: Forcing Variables**

.

where the variables have the following meaning:

1. **KODEQ** is the first code number and defines which quantity is written onto, e.g., [PPSF_ALL]. See also table 5.1 through table 5.6.

2. **KODES** describes the type of section on which the quantity is written out. See also table 5.7 and 5.8.

3. **N1** is the first dimension of the array.

4. **N2** is the second dimension of the array.

5. **N3** is the third dimension of the array. Note that this allows to write out several 2d-arrays that require the same land-sea mask and have the same coordinates.

6. **KODEL** is a key number that defines the location of a section, e.g., as depth in meter or longitude/latitude in degree.

7. **IFLG** is the land-sea mask with dimension (N1,N2). In the ocean model *IFLG* is defined as INTEGER array, while on an PPSF-file it is written out as CHARACTER*1.

8. **KTOTAL** is the model time in seconds.

9. **MSEC** is the model time in seconds after the full minute.

10. **MINUTE** is the model time in minutes after the full hour.

Code Number	Variable	Unit	Reference
21	total surface heat flux	Wm^{-2}	$QFLUX$
22	solar radiation	Wm^{-2}	$QSOLAR$
23	longwave radiation	Wm^{-2}	$QLONG$
24	sensible heat flux	Wm^{-2}	$QSENSIB$
25	latent heat flux	Wm^{-2}	$QLATENT$
26	net fresh water flux as salt flux	$WK^{-2}m^{-2}$	$SFLUX$
27	surface buoyancy flux	m^2s^{-3}	$BFLUX$
28	solar buoyancy flucx	m^2s^{-3}	$BSOLAR$
29	friction velocity	ms^{-1}	$USTERN$
61	drag coefficient		$DRAG$
62	transfer coefficient for sensible heat		<VARDRAG>
63	transfer coefficient for latent heat		<VARDRAG>
64	feedback coefficient	$WK^{-2}m^{-2}$	$DQDT$
75	vertical velocity	ms^{-1}	[makeall.f]
76	layer relative vorticity	s^{-1}	[makeall.f]
77	layer absolute vorticity	s^{-1}	[makeall.f]
78	layer potential vorticity	s^{-1}	[makeall.f]
91	heat flux bias correction	Wm^{-2}	<SURFLUX>
92	fresh water flux bias correction	ms^{-1}	<SURFLUX>
93	wind stress bias correction	Nm^{-2}	<STRESS>

Table 5.5: **Definition of Quantity Code: Flux Variables**

Code Number	Variable	Unit	Reference
35	barotropic stream function	kgs^{-1}	[makeall.f]
36	meridional overturning	kgs^{-1}	[makeall.f]
47	overturning in Atlantic Ocean	kgs^{-1}	[makeall.f]
48	overturning in Indic Ocea	kgs^{-1}	[makeall.f]
49	overturning in Pacific Ocean	kgs^{-1}	[makeall.f]

Table 5.6: **Definition of Quantity Code: Transport Variables**

11. **MHOUR** is the model time in hours after the full day.

12. **MONTH** is the model time in months after the full year.

13. **MYEAR** is the model time in years after the start.

14. **LDATE** contains the date at which the data set was created.

15. **LTIME** contains the time at which the data set was created.

16. **EXPERIM:** It is a character string declared as CHARACTER*40 and contains the identification of an experiment.

17. **X1** contains the coordinates of the first index of the array $FIELD$.

18. **X2** contains the coordinates of the second index of the array $FIELD$.

19. **FIELD** contains the quantity to be saved.

Code Number	Variable	Unit	Reference
30	surface air relative humidity		<SURFLUX>
80	surface water on land	m	<SURFLUX>
81	atmospheric water content	kgm^{-3}	<SURFLUX>
82	diagnostic precipitation	ms^{-1}	<SURFLUX>
101	surface air temperature	K	<SURFLUX>
102	top shortwave radiation	Wm^{-2}	<SURFLUX>
103	bottom shortwave radiation	Wm^{-2}	<SURFLUX>
104	top longwave radiation	Wm^{-2}	<SURFLUX>
105	bottom longwave radiation	Wm^{-2}	<SURFLUX>
106	sea surface and soil temperature	K	<SURFLUX>
107	sea surface temperature	K	<SURFLUX>
108	ice and snow cover	m	<SURFLUX>
109	horizontal heat transport	Wm^{-2}	<SURFLUX>
110	vertical heat release	Wm^{-2}	<SURFLUX>
111	orography	m	<SURFLUX>
112	surface turbulent fluxes	Wm^{-2}	<SURFLUX>

Table 5.7: **Definition of Quantity Code: EBM Variables**

5.5.3 Portable Compressed Format

If PPSF-files are created as binary files, these usually are not portable to another UNIX-platform, unless the files are created as 32-bit IEEE format, which is the default on many workstations. However, if the PPSF-files are created e.g. on a CRAY one needs to convert them in order to be able to postprocess data on a workstation. There are two possibilities. The script [ppsf2i3e] can be used to convert the binary files to 32-bit IEEE, which gives a compression effect of nearly 50%. Another option is to use the script [implode]. It converts the binary format into a character string with 3-byte resolution. If the script [explode] is installed onto some system, then it converts the result of [implode] back into the local binary-format on that particular system. Because [explode] and [implode] use standard ANSI language only, the combination [implode] and [explode] allow the transfer of data that originally are in PPSF-binary format onto all computer platforms.

5.5.4 List of PPSF-Files

5.5.4.1 The Quick-Look System

The model delivers a number of quick-look files in PPSF (Post Processing System Format). They contain model data from the end of a model run. The data are sorted in the quick-look files according to their meaning. The contents of these files can be easily adjusted to any requirements. See <OCEPOST> for details.

5.5.4.1.1 File [PPSF_SUR] contains surface quantities for velocity, sea level, mixed layer depth, temperature and salinity.

5.5.4.1.2 File [PPSF_ICE] contains the ice drift, the ice thickness, the ice compactness, the snow cover, the snow surface temperature and the snow-ice interface temperature.

Code Number	Type of Section	Reference
1	section for xy-plane of instantaneous variable	<PPSFOUT>
2	section for xz-plane of instantaneous variable	<PPSFOUT>
3	section for yz-plane of instantaneous variable	<PPSFOUT>
4	integral of yz-plane over x	<PPSFOUT>
5	integral of xz-plane over y	<PPSFOUT>
6	integral of xy-plane over z	<PPSFOUT>
7	monthly means on xy-section	<PPSFOUT>
8	monthly means on xz-section	<PPSFOUT>
9	monthly means on yz-section	<PPSFOUT>
10	line section in x-direction	<PPSFOUT>
11	line section in y-direction	<PPSFOUT>
12	line section in z-direction	<PPSFOUT>
13	integral along x over yz-section	<PPSFOUT>
14	integral along y over xz-section	<PPSFOUT>
15	integral along z over xy-section	<PPSFOUT>
16	monthly means along line section in x-direction	<PPSFOUT>
17	monthly means along line section in y-direction	<PPSFOUT>
18	monthly means along line section in z-direction	<PPSFOUT>

Table 5.8: **Definition of Section Code: Part I**

5.5.4.1.3 File [PPSF_BAS] contains the momentum fluxes, layer thicknesses temperature and salinity in original model coordinates. This file can be used by [makeall.f] to compute the horizontal mass transport stream function, the vertical overturning stream function and many other diagnostic variables such as vertical velocity, potential vorticity, etc.

5.5.4.1.4 File [PPSF_SEC] contains the flow, temperature, salinity and vertical velocity on those depth levels which are defined by *VSEC* (see table 4.13).

5.5.4.1.5 File [PPSF_VER] contains xz- and yz-sections of temperature, salinity, zonal and meridional velocity components and the interface distribution on longitudes and latitudes that are defined by *ISEC* and *JSEC* as *I*- and *J*-indices of the model arrays.

5.5.4.1.6 File [PPSF_FLX] contains the fluxes into the ocean. These are the surface wind stress, the friction velocity and the fluxes of heat, fresh water and buoyancy. Dependent on *ISW48* it also can contain the long term mean heat and fresh water flux, or the bias correction terms if enabled. It also contains the EBM fluxes.

5.5.4.1.7 File [PPSF_FOR] contains the model topography, the absolute wind speed and its standard deviation, the air temperature, the sea surface temperature and the sea surface salinity at the end of the model run and some EBM output.

5.5.4.1.8 File [PPSF_TID] contains quick-look data of the tide model.

5.5.4.1.9 File [PPSF_TOP] contains the model topography if the ocean state has been initialized. Note that this file should be used to verify a correct initialization of the grid and coastline geometry.

Code Number	Type of Section	Reference
19	intergral over (x,y,z,t)	<PPSFOUT>
22	model run means on xy-section	<PPSFOUT>
23	model run means on xz-section	<PPSFOUT>
24	model run means on yz-section	<PPSFOUT>
25	section on xt-plane	<PPSFOUT>
26	section on yt-plane	<PPSFOUT>
27	section on zt-plane	<PPSFOUT>
28	integral on xy-section over t	<PPSFOUT>
29	integral on xz-section over t	<PPSFOUT>
30	integral on yz-section over t	<PPSFOUT>
31	point section along t	<PPSFOUT>
35	annual means on xy-section	<PPSFOUT>
36	annual means on xz-section	<PPSFOUT>
37	annual means on yz-section	<PPSFOUT>

Table 5.9: **Definition of Section Code: Part II**

5.5.4.1.10 File [PPSF_INI] contains all atmospheric quantities for forcing the ocean as well as the sea surface temperature and sea surface salinity. Fields are written out for all months except the salinity which appears only once as an annual mean. This file should be used to verify the correct initialization of the model forcing or to detect possible data problems. Note that this file is created only when the atmospheric forcing is created and saved in [FORCES]. Caution: The file contains a large amount of data.

5.5.4.2 The History Files

5.5.4.2.1 File [PPSF_ALL] is the history file which contains accumulated model quantities that are written out with some frequency but in unsorted order. Which quantities are written out by <OCEPST1>, <OCEPST2>, <OCEPST3>, <OCEPST4> and <OCEPST5> depends on the particular model experiment. With the help of the postprocessing output routines (see section 4.5) it can be decided which quantities are saved on the history file, and the frequency of storing them.

5.5.4.2.2 File [PPSF_HIS] is the history file of the tracer model.

5.5.5 [getf] and [putf]

The scripts [getf] and [putf] are used in numerous scripts. Both scripts perform a *ftp* to another machine, where [getf] is trying to receive a file while [putf] is trying to send a file. In both scripts, the length of the local file and the remote file are compared with each other. If both have the same size the transfer is finished. Possibly, the scripts have to be adjusted to the operating system so that the file size in bytes is correctly cut with the *awk*-command from the line created by the ftp's *dir*-command.

5.5.6 Utilities for Data Analysis

The following routines allow to manipulate and convert the contents of a PPSF-file, and to perform mathematical operations. See also the file [README] in *./PostPro* for further information. Unless noticed otherwise all codes below work as long as the dimensions used in the

PPSF-format does not exceed the dimensions used in the underlying routines. A few routines expect that the dimensions are identical with that used in the input data (see warnings). All utilities have a simple online documentation. Use extension *-h* to view the help-file instead of performing any operation. For instance, *filter -h* results in:

```
filter [ -q KODEQ ] [ -s KODES ] [ -l KODEL ] Input_File Output_file

Purpose: Extract data set from Input_file and write onto Output_File

Definition(s)        : Integer KODEQ, KODES, KODEL
                       File    Input_File, Output_File

Optional Parameter(s): -q KODEQ , where KODEQ is the Quantity Code
                       -s KODES , where KODES is the Section  Code
                       -l KODEL , where KODEL is the Location Code

Remark(s)            : If one of the KODEs matches codes in the data set,
                       these are written to Output_File
```

The following commands consist of scripts, codes and documentation files. The scripts are tested on a SUN-station and on a CRAY using *CShell*. Dependent on the operating system the script decides whether the *assign*- or the *link*-command is used to link the data file with the local file in order to avoid extensive copying of data across directories or disks. They also use different loader-commands.

5.5.6.1 Environment Variables

In order to find the subsequent scripts for execution, the search path for, in the example, *$HOME/OPYC/PostPro/Scripts* needs to be defined in [.profile], [.cshrc] or [.login] under UNIX. Furthermore, all scripts expect an environment variable *OPYC_PPS_OBJ* to be defined as e.g. *$HOME/OPYC/PostPro/Objects* in order to find the object files.

5.5.6.2 Utilities for Manipulating Data Sequence

- Script [**append**] merges a sequence of input files.
- Script [**delete**] deletes specified codes on the input file.
- Script [**extract**] extracts blocks of data from an input file by defining the first block number, the last one and an increment.
- Script [**filter**] extracts all data blocks with a prescribed quantity code *KODEQ*, section code *KODES* and location code *KODEL* from an input file.
- Script [**insert**] inserts data at a specified location of an input file and writes the result to an output file.
- Script [**cycle**] performs a cyclic move of the data from the end to the start of the input file or vice versa.
- Script [**annual**] computes the climatological seasonal cycle from an input file containing many years of data.
- Script [**seasons**] computes seasonal means like DJF, MAM, JJA and SON from an input file containing many years of data.

- Script [**area**] computes an area average for non-zero data.
- Script [**curve**] extracts a time series of means.
- Script [**blank**] extracts either the Atlantic, the Pacific or the Indic Ocean basin data from an input file containing e.g. global data.
- Script [**clean**] sets the coordinates and flag fields of all subsequent data to those of the first data record. All data must be of the same type.

5.5.6.3 Utilities for Manipulating Data Contents

- Script [**add**] adds the contents of two input files and writes the result to an output file.
- Script [**mult**] multiplies the contents of two input files and writes the result to an output file.
- Script [**divide**] divides the contents of the first input file by the contents of the second one and writes the result to an output file.
- Script [**differ**] computes the difference of the contents of a first input file with that of a second one and writes the result to an output file.
- Script [**triade**] computes the sum of the first two input files, multiplies the result with the contents of a third input file and writes the result to an output file.
- Script [**addc**] adds a constant value to all quantities of an input file and writes the result to an output file.
- Script [**multc**] multiplies a constant to all quantities of an input file and writes the result to an output file.
- Script [**curl**] computes the rotation of vector input data.
- Script [**div**] computes the divergence of vector input data.
- Script [**grad**] computes the gradient of scalar input data.
- Script [**norm**] computes the norm of vector input data, i.e. the result of $\sqrt{u^2 + v^2}$ if (u, v) is the input vector.
- Script [**abs**] computes the absolute value.
- Script [**min**] computes the minimum of subsequent data of the same type.
- Script [**max**] computes the maximum of subsequent data of the same type.
- Script [**mean**] computes the mean of subsequent data of the same type.
- Script [**int**] computes the integral of subsequent data of the same type.
- Script [**average**] computes the mean of subsequent data. In opposite to [mean] the input file now may contain different quantities. The output file contains the averages for each quantity.
- Script [**merge**] computes the mean of subsequent ocean layer raw data only, separately for each quantity. Other quantities are forwarded to the output file.

- Script [**anomaly**] computes the anomaly of subsequent data of the same type. It is a macro that calls other scripts.
- Script [**stdv**] computes the standard deviation of subsequent data of the same type.

5.5.6.4 Time Series Analysis

- Script [**xtime**] extracts data for a Hovmueller-diagram (x-direction and time) from an input file that contains a sequence of some quantity ordered in time. The location of the section is defined by one parameter.
- Script [**ytime**] extracts data for a Hovmueller-diagram (y-direction and time) from an input file that contains a sequence of some quantity ordered in time. The location of the section is defined by one parameter.
- Script [**regress**] computes the linear regression coefficient from subsequent data of the same type.
- Script [**degress**] reconstructs a time series of subsequent data of the same type by using the linear regression coefficients that have been generated by [regress].
- Script [**ppsf2eof**] computes the empirical orthogonal functions EOFs and the principal components PCs from subsequent data of the same type.
- Script [**eof2ppsf**] reconstructs the time series from empirical orthogonal functions EOFs and according principal components PCs.
- Script [**eof2cca**] performs a canonical correlation analysis CCA using EOFs and PCs as generated by [ppsf2eof].
  Warning: The dimension-statements must be adjusted to the particular settings of the current model version.
- Script [**ppsf2cca**] performs a canonical correlation analysis CCA from subsequent data of the same type. It is a macro that calls other scripts.
  Warning: The dimension-statements must be adjusted to the particular settings of the current model version.
- Script [**detrend**] detrend subsequent data of the same type using an EOF-analysis. It is a macro that calls other scripts.
  Warning: The dimension-statements must be adjusted to the particular settings of the current model version.

5.5.6.5 Frequency Analysis

- Script [**spectrum**] performs a spectral analysis and outputs data either as curves or as Hovmueller-diagram in time and either in x- or in y-direction.
- Script [**fourier**] performs a fourier analysis and outputs amplitude and phases at a specified period.
- Script [**wavelet**] performs a wavelet analysis.
- Script [**fttid**] performs a fourier analysis for tidal frequencies.
  Warning: The dimension-statements must be adjusted to the particular settings of the current model version.

5.5.6.6 Data Filtering

- Script [**smooth**] filters the data using a Helmholtz-equation. Thus, the result is widely independent on the actual grid resolution.
- Script [**rmean**] computes the running mean over a specified period.

5.5.6.7 Data Reformatting

- Script [**spr2dpr**] converts a binary PPSF-format that has been created with single precision into a binary PPSF-format that should contain double precision words. This procedure is useful if the model runs on a computer with 32bit-CPU with double precision, but the postprocessing should be done with single precision.
- Script [**dpr2spr**] converts a binary PPSF-format that has been created with double precision into a binary PPSF-format with single precision.
- Script [**implode**] converts a binary PPSF-format to a portable file with reduced accuracy, i.e. 3 byte resolution. This file can be ported to any computer system and unpacked with [explode].
- Script [**explode**] converts the portable file with reduced accuracy, i.e. 3 byte resolution, to a binary PPSF-format. Dependent on the operating system, it either will create a 32-bit IEEE format or like on PVP-computers a 64-bit binary format.
- Script [**ppsf2i3e**] converts a binary PPSF-format to a 32-bit IEEE format in order to make PPSF-files created on a PVP-computer portable to a 32-bit operating system. This script makes sense only to execute it a PVP-computer.
- Script [**i3e2ppsf**] converts a 32-bit IEEE format to a binary PPSF-format. This script makes sense only to execute it a PVP-computer.
- Script [**modify**] modifies one of the quantity, section or location codes in the file headers of the PPSF-format.
- Script [**reduce**] extracts a fraction of the model domain and creates new data in PPSF-format with adjusted dimensions in the header and the data records.
- Script [**fill**] fills in extra latitudes south of the first latitude of a data set. This makes sense only for horizontal fields, where for the sake of saving CPU-time the southernmost latitudes on Antarctica are not part of the model domain, however, are needed to adjust the data format with that of other model data.
- Script [**fit**] merges data on two input files containing data of the same type such that e.g. two scalar data sets are merged to one vector data set.

5.5.6.8 Viewing Data

- Script [**contents**] prints the contents of the headers.
- Script [**brief**] prints the statistics of each data set, i.e. the means, minima, maxima and the standard deviations.
- Script [**show**] prints the code definitions.

- Script [**list**] prints coordinates, flag fields and data.

- Script [**plot**] creates plots of all quantities from an input file. [plot.f] needs to be adjusted to the model version by choosing parameters that define model geometry as shown by the plot.
  Warning: The dimension-statements must be adjusted to the particular settings of the current model version.

5.5.6.9 Data Preprocessor

- Script [**lev2mod**] computes horizontal sections of temperature and salinity from the Levitus data file [SALTEMP] and delivers these quantities on the model grid. The output file can be used to compute, e.g., differences between the Levitus data and model data.
  Warning: The dimension-statements must be adjusted to the particular settings of the current model version. A script does not exist.

5.5.6.10 Data Postprocessor

The postprocessor consisting of the script [makeall], the code [makeall.f] and the documentation file [makeall.doc] is a general purpose tool to diagnose a variety of model quantities.

- Script [**makeall**] computes the horizonal transport stream function and the vertical overturning stream function. Optionally, the overturning stream functions for the Atlantic, the Pacific and the Indian Ocean are written onto an output file. It also computes horizontal and vertical sections of temperature, salinity and many other quantities. Use the command *makeall -h* in order to obtain an overview. As then listed *makeall* also computes quantities like electro-streamfunction or 3-dimensional magnetic potentials as part of a project with APL, Seattle, through Robert Tyler.
  Warning: The dimension-statements must be adjusted to the particular settings of the current model version.

- Script [**ppsf2grd**] converts the PPSF-format to the data structure expected by the GrADS-System. This possibility is offered but not supported.
  Warning: The dimension-statements must be adjusted to the particular settings of the current model version.

5.6 The Plot System

The plot software reads files in binary PPSF-format. The header of each data block specifies how a quantity will be plotted. The intention for the plot system is to work without additional changes to produce any chosen picture unless the subsequently listed control parameters have been specified by a new model layout.

5.6.1 The Plot Program

Similar to the ocean model, the plot program is split up into a simple main PROGRAM and several BLOCK DATA BLOCKS in [ocepict.f] and a collection of routines in [oceplot.f]. The object code is created by the command *modgen.sun plot* or by *modgen.cri plot*. Currently, the whole plot system requires the plot library [jmoplane.f], a small C-program [utility.c] and a GKS-library like provided by NCAR. Thereby, the simplest version of a GKS-ststem, the

GKS0A-level, is sufficient. If a GKS-system is not available, [jmoplane.f] can be compiled without the directive *-DNCAR*. Then the resulting executable creates a PostScript-file, which might be viewed with e.g. *ghostscript*.

5.6.1.1 Main Program PICTURE

The main program <PICTURE> consists only of one call to <OCEPLOT> which is the driving subroutine. It reads data block by data block and creates a plot that is dependent on the code numbers required to identify a quantity.

5.6.1.2 Block Data PLOTPAR

The parameters of table 5.10 control the overall behaviour of the plot program. Features as number of to be plotted pictures, plot quality, type of plotter (laser or color plotter) etc. can be chosen. The parameters of table 5.11 define in more detail the size, content and lables used for plotting. Finally, table 5.12 allows to define the Eulerian angles essentially to add lines of geographical longitude and latitude on each frame.

Variable	Value	Default	Meaning	Reference
NPIC	any $\geq$1	999	number of pictures to be plotted	/JMOPLT1/
NCOL	-1,0,1,2	-1	switch for black&white or color plots	/JMOPLT1/
NTURN	0,1	0	switch for rotation of picture on sheet	/JMOPLT1/
NSHIFT	any	0	number of grid points for x-shifting	/JMOPLT1/
NQUAL	-3,-2,-1,1,2,3	-1	switch for quality of contour plots	/JMOPLT1/
MUE	any $\geq$1	1	x-refinement factor for contours	/JMOPLT1/
NUE	any $\geq$1	1	y-refinement factor for contours	/JMOPLT1/
IPART	any $\geq$1	1	subdividing figure in x-direction	/JMOPLT1/
JPART	any $\geq$1	1	subdividing figure in y-direction	/JMOPLT1/
NFILTER	0,1	1	switch for spatial filter	/JMOPLT1/
NPRETTY	0,1	0	switch from model mask to real mask	/JMOPLT1/
NSIGN	0,1	0	switch from solid to dashed contours	/JMOPLT1/
NFILM	0,1	0	switch from sheet to movy mode	/JMOPLT1/
NOFFSET	any	0	time-offset to model time counter	/JMOPLT1/
NTRAJ	any	0	switch for trajectories instead of vectorrs	/JMOPLT1/

Table 5.10: **Definition of Main Switches**

5.6.1.3 Block Data STRINGS

<STRINGS> contains the relation of the first code number to the physical quantity. Data are transferred into <OCEPLOT> via /JMOSTRI/ and are required to define the header for each plot. This program unit contains a complete list of code definitions.

Variable	Value	Unit	Default	Meaning	Reference
XLEFT	any	deg	0	location of left margin	/JMOPLT2/
XRIGHT	any > XLEFT	deg	360	location of right margin	/JMOPLT2/
YLOW	any	deg	-90	location of lower margin	/JMOPLT2/
YUPP	any > YLOW	deg	90	location of upper margin	/JMOPLT2/
NXS	any $\geq$2		-13	number of labels for x-direction	/JMOPLT2/
LX0	any		0	first label for x-direction	/JMOPLT2/
LX	any		360	label difference for x-direction	/JMOPLT2/
NYS	any $\geq$2		-7	number of labels for y-direction	/JMOPLT2/
LY0	0,1		-90	first label for y-direction	/JMOPLT2/
LY	0,1		180	label difference for y-direction	/JMOPLT2/
BMINA	any >0		180	length of plot excluding frame	/JMOPLT2/
DMINC	any >0		90	height of plot excluding frame	/JMOPLT2/

Table 5.11: **Definition of Margins**

Variable	Value	Unit	Default	Meaning	Reference
LANGLE	0,1		0	switch for Eulerian angles	/JMOPLT6/
ALPHA	any	deg	0	1st Eulerian rotation angle in longitude	/JMOPLT6/
BETA	any	deg	0	2nd Eulerian rotation angle in latitude	/JMOPLT6/
GAMMA	any	deg	0	3rd Eulerian rotation angle in longitude	/JMOPLT6/

Table 5.12: **Definition of Eulerian Angles**

5.6.1.4 Block Data MARKS

<MARKS> contains the relation of the section code to the type of section used to write the data. Data are transferred into <OCEPLOT> via /JMOMARK/ and are required to switch between different layouts of the plots as between xy-, xz- or yz-sections.

5.6.1.5 Block Data UNITS

<UNITS> contains the definition of the unit for each quantity and a different definition in case the section code is negative. Data are transferred into <OCEPLOT> via /JMOUNIT/ and are required for the plot.

5.6.1.6 Block Data ALLIND

<ALLIND> contains the definition for reordering the sequence of colors in the color table, for the layout of the color table, for the type of projection chosen for all vector, contour or color plots, and a parameter which defines whether contours surround all equally colored areas (see also tabel 5.13).

Variable	Value	Meaning	Reference
IROT	0,1	switch for rotation of the color table	/JMOIND/
KEYCOL	1,2,3,4	switch to control the margins of color table	/JMOIND/
IPOL	0,1,2,3	switch to select a projection	/JMOIND/
ICON	any $\geq$0	switch for contours in color plots	/JMOIND/

Table 5.13: **Definition of Layout Parameters**

5.6.1.7 Block Data MINMAXS

<MINMAXS> contains the definitions for the lowest and highest level that is plotted by contours or colored areas (see also table 5.14).

Variable	Value	Meaning	Reference
SMIN	any	lowest level used for contouring	/JMOMIMA/
SMAX	any > SMIN	highest level used for contouring	/JMOMIMA/
DMIN	any	as SMIN but for negative section code	/JMOMIMA/
DMAX	any > SMAX	as SMAX but for negative code numbers	/JMOMIMA/
IMON	-2,-1,0,1,2	switch for shading in contour plots	/JMOMIMA/
IKOL	any	number of colours used for shading	/JMOMIMA/

Table 5.14: **Definition of Interval Parameters**

5.6.1.8 Block Data CVARIAB

<CVARIAB> allows to specify non equidistant contour or color intervals.

5.6.1.9 Block Data ADDFACS

<ADDFACS> contains definitions that rescale quantities e.g. to avoid long labels in a contour plot and a parameter that determines the layout of a label (see also table 5.15).

Variable	Value	Meaning	Reference
ADD	any	additive constant to scale quantity	/JMOADFA/
FACTOR	any	multiplicative constant to scale quantity	/JMOADFA/
NDEC	any	switch for selecting the label type	/JMOADFA/

Table 5.15: **Definition of Scaling Parameters**

5.6.1.10 Block Data QZONMER

<QZONMER> contains definitions for additional curves for means in x-direction or/and y-direction, and for ignoring a contour with some specified index (see also table 5.16).

Variable	Value	Meaning	Reference
NMERID	any	number of contour/colour intervals shown as x-curve	/ZONMER/
NZONAL	any	number of contour/colour intervals shown as y-curve	/ZONMER/
IGNORE	any	index of contour which is not plotted	/ZONMER/

Table 5.16: **Definition of Zonal and/or Meridional Means**

5.6.1.11 Executing Program OCEPLOT

<OCEPLOT> uses the plot library [jmoplane.f]. Basically, it analyses the header of each data block and decides how a data set has to be treated. The access to the key parameters is provided by the BLOCK DATA structures explained above.

5.6.2 The Underlying Library JMOPLANE

[jmoplane.f] is a plot library developed by the author. It is optimized for the purpose of plotting ocean quantities with land-sea masks being easily treated. For any detailed information there is an online manual available located on [./Graphic/Manual] which describes each routine. It either expects some GKS0A-level GKS-library or writes the output in PostScript-format if the option *-DNCAR* is not used at compile-time.

5.6.3 Example Plots

Fig. 5.1 shows a typical example for a vector plot. Each second arrow is taken out in order to improve the layout. Here the land-sea mask of the model is used to mark the continents. Fig. 5.2 is a standard example for a contour plot on a xy-section. Fig. 5.3 is an example for the layout of a vertical section. Vertical sections are plotted twice; for the upper *500 m* and for the whole depth. However, they are split up into two linear representations, one for the upper *1000 m* and one between *1000 m* and *6000 m*. Fig. 5.4 shows an example for a polar projection. This is achieved by setting *IPOL=3* in |ALLIND| to obtain a geometrical projection. Otherwise, a scalar quantity would be plotted as shown by Fig. 5.2. If a global field is plotted, the south- and north-polar projection are plotted on two separate sheets. The south-polar projection is not shown here. The same projection is available also for vector plots. Fig. 5.5 shows an example for a fine-resolution North Atlantic-Arctic model in rotated coordinates.

5.6.4 Usefull Tools

The directory [./Tools] contains 3 simple codes which help to define the model's coordinates in [ocemain.f]. These are

- File [**makexy.f**], which creates the horizontal grid using the same block data statements than used in [ocemain.f].
- File [**makever.f**], which generates the potential density coordinates using the same block data statements than used in [ocemain.f].
- File [**plotgeo.f**], which generates a plot that shows the coastline geometry of some rotated model domain. This code is usefull to optimize the Eulerian angles. The resulting object file must be loaded together with [jmoplane.o] and [utility.o] as shown in [./bin/paint].

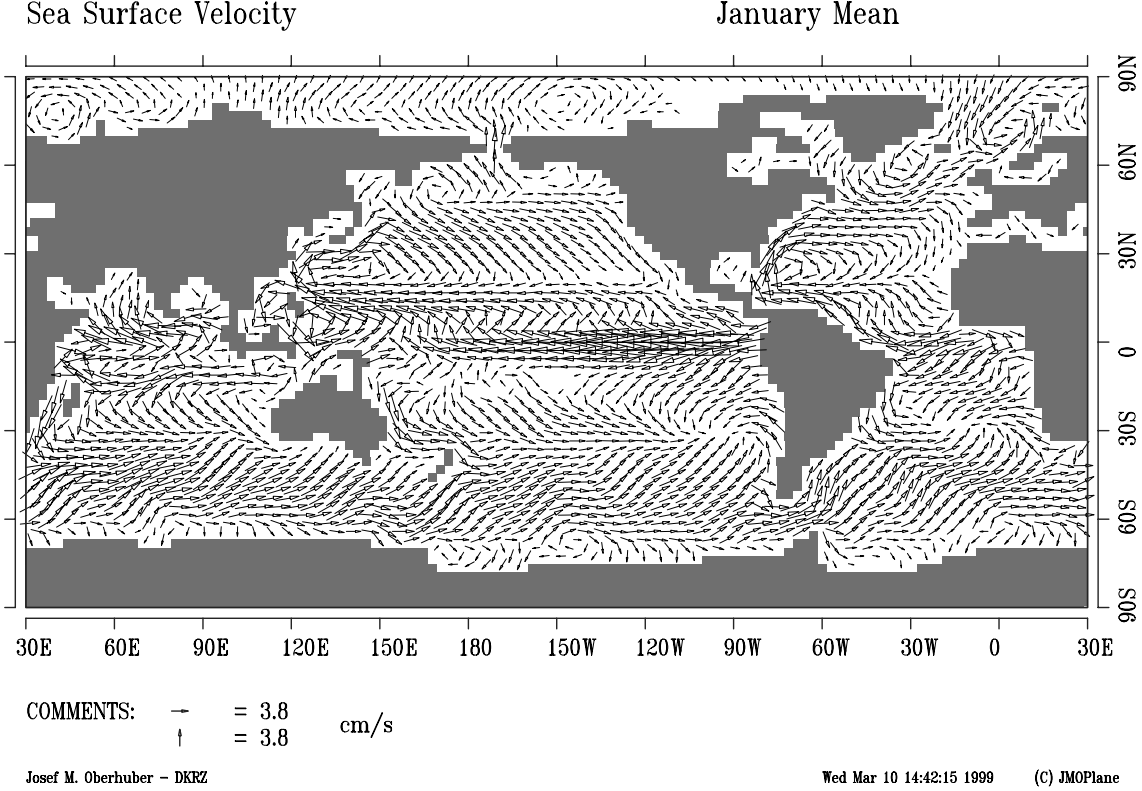

Figure 5.1: Example showing a vector plot for the surface flow of PIPE/T42. The picture is created interactively using *plot -b -e -q 69 -a 'Annual Mean' PPSF_SUR*.

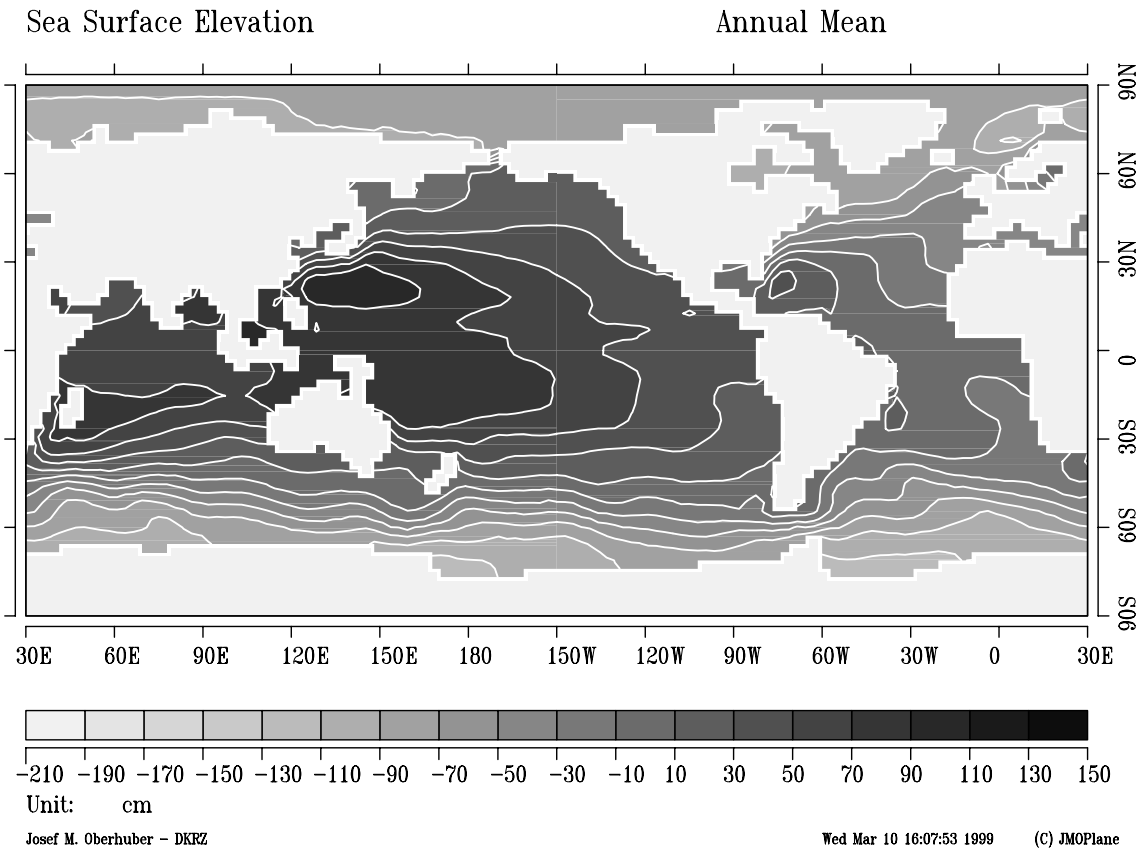

Figure 5.2: Example showing a greyscale colour plot for the sea level of the PIPE/T42. The picture is created interactively using *plot -b -p -e -q 68 -a 'Annual Mean' PPSF_SUR*.

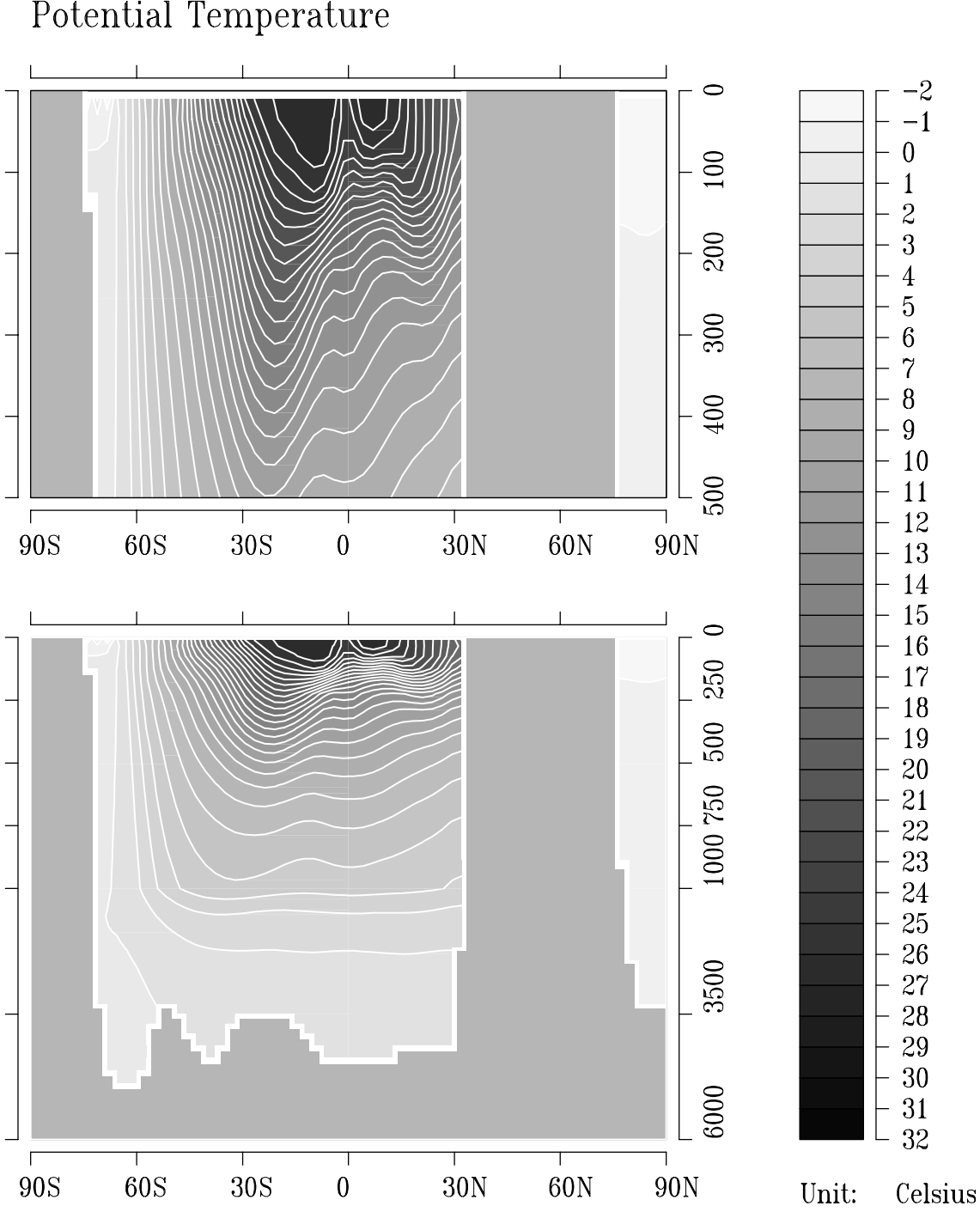

Figure 5.3: Example showing a yz-section for the potential temperature at $120W$ from the PIPE/T42. The picture is created interactively using *plot -b -p -e -q 13 -l 240 PPSF_VER*.

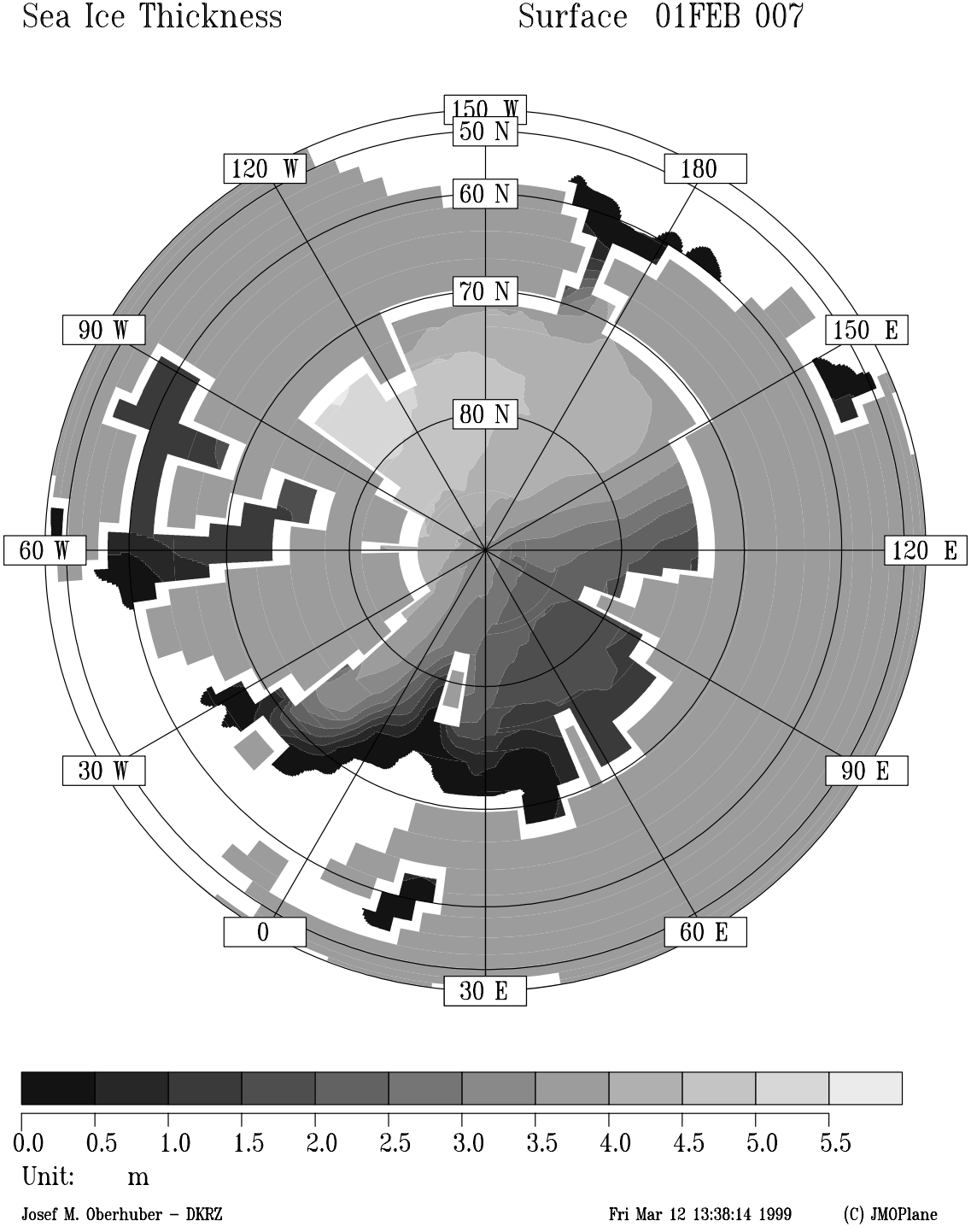

Figure 5.4: Example showing a north-polar projection for the sea ice thickness of PIPE/T42. The picture is created interactively using *plot -b -p -e -r -q 18 -N PPSF_ICE*.

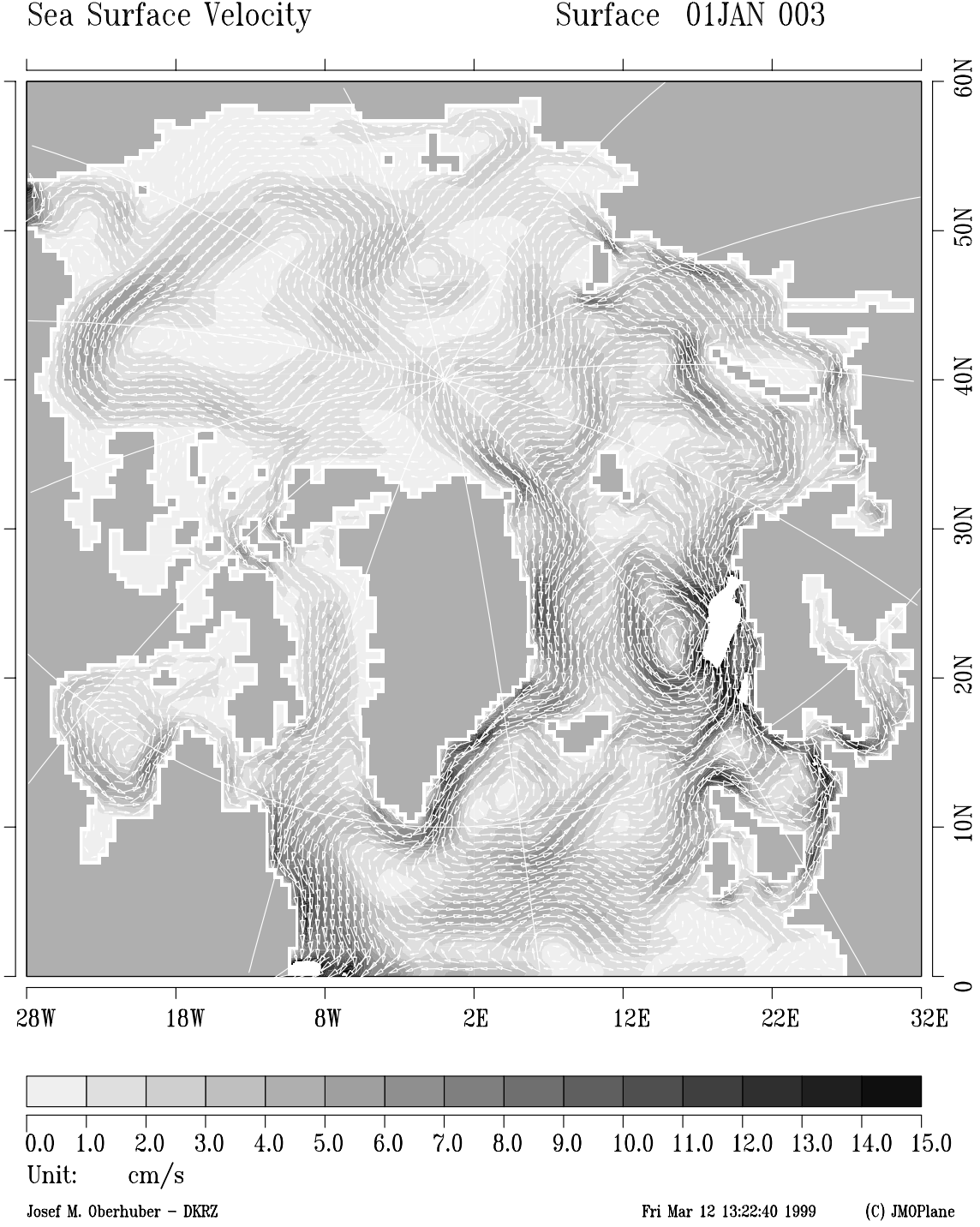

Figure 5.5: Example showing the surface flow in a North Atlantic-Arctic model using Eulerian angles (*ALPHA=-40*, *BETA=-50*, *GAMMA=0*). Geographical longitude and latitude are plotted for better orientation. The picture is created interactively using *plot -b -p -e -q 69 PPSF_SUR*.

Chapter 6

Appendices

6.1 Appendix A: Prognostic Pressure Equation

In order to present the strategy to solve the model equations with an implicit time-integration scheme, the momentum and pressure equations are written in non-dimensional form:

$$\frac{\partial}{\partial t}u = -\frac{\partial}{\partial x}p \tag{6.1}$$

$$\frac{\partial}{\partial t}p = -\frac{\partial}{\partial x}u \tag{6.2}$$

If the pressure gradient and the flux divergence are equally weighted in time, the finite-difference representation in space is:

$$u^{l+1} + \frac{\Delta t}{2}\Gamma D_x^s p^{l+1} = u^l - \frac{\Delta t}{2}\Gamma D_x^s p^l \tag{6.3}$$

$$p^{l+1} + \frac{\Delta t}{2}D_x^v u^{l+1} = p^l - \frac{\Delta t}{2}D_x^v u^l \tag{6.4}$$

where D_x^s and D_x^v denote the algebraic representations of the gradient on vector points and divergence on scalar points, respectively, Δt is the time step and the superscripts l and $l+1$ mark the old and new time levels. The boundary conditions are introduced via a flag Γ that is $\Gamma = 0$ for land and $\Gamma = 1$ for the ocean. This implies no-slip coastal boundary conditions. Because Eqn. (6.1) and (6.2) are interconnected by the pressure gradient and the flux divergence, one of the two options is to remove u^{l+1} in the pressure equation. This results in an equation for the pressure p^{l+1} only at the new time level:

$$p^{l+1} - \frac{\Delta t^2}{4}D_{x,v}(\Gamma D_{x,s}p^{l+1}) = p^l - \frac{\Delta t}{2}D_{x,v}u^l + \frac{\Delta t^2}{4}D_{x,v}(\Gamma D_{x,s}p^l) \tag{6.5}$$

After having found the solution for p^{l+1}, it is used to calculate the flux u^{l+1} at the new time level. This simple strategy to obtain a stable solution for any chosen time step is used to derive a wave equation for the model's equations for the momentum and pressure.

6.2 Appendix B: Pressure Boundary Condition

A subject of controversy is often the use of an elliptic equation for predicting the pressure. It must be stressed that the analytical derivation of Eqn. (2.40) after eliminating the velocities according to section 2.1.2. does not automatically provide the correct pressure boundary condition. In order to obtain a correct discretization for pressure points near the boundary, first

a no-slip boundary condition is applied for the momentum Eqn. (1.1) and a no-flux condition for the continuity Eqn. (1.2). So far, there is no difference to explicit time-stepping models. The discrete prognostic equations are then modified such that specific terms are labelled with the new time level index *l+1* rather than the old time level *l*. Then, the prognostic equation for the pressure p^{l+1} at the new time level is obtained using algebraic manipulations, which do not change the solution at all and thus do not introduce inconsistencies in boundary conditions. This way of deriving a discrete equation for the new pressure provides the algebraic representation of the prognostic pressure equation near the boundary. This equation replaces otherwise necessary and mostly intuitive diagnostic pressure boundary conditions that might lead to erroneous physical behaviour. In analogy to the derivation of Eqn. (6.5) the derivation of the model's pressure equation on boundary grid points yields a non-elliptic equation which can be solved nevertheless with an appropriate underrelaxation scheme.

6.3 Appendix C: Constants for Equation of State

a_{100}	0.36504	a_{110}	0.017439	a_{101}	0.083198	a_{111}	-0.00029778
a_{200}	0.89306	a_{210}	-0.0041057	a_{201}	-0.031626		
a_{102}	-0.00054065	a_{202}	0.00021987	a_{103}	0.0000040274		
a_{300}	-0.16056			a_{301}	0.0050484		

Table 6.1: **Constants for Bryden's Formula:** The subscripts denote the exponent for pressure (first index), salinity (second index) and temperature (third index). See also Eqn. (1.16).

a_0	999.842594	b_0	$8.24493\ 10^{-1}$	e_0	19652.21	f_0	54.6746
a_1	0.06793952	b_1	$-4.0899\ 10^{-3}$	e_1	148.4206	f_1	-0.603459
a_2	$-9.09529\ 10^{-3}$	b_2	$7.6438\ 10^{-5}$	e_2	-2.327105	f_2	0.0109987
a_3	$1.001685\ 10^{-4}$	b_3	$-8.2467\ 10^{-7}$	e_3	0.01360477	f_3	$-6.167\ 10^{-5}$
a_4	$-1.120083\ 10^{-6}$	b_4	$5.3875\ 10^{-9}$	e_4	$-5.155288\ 10^{-5}$		
a_5	$6.536332\ 10^{-9}$						
c_0	$-5.72466\ 10^{-3}$	g_0	$7.944\ 10^{-2}$	h_0	3.239908		
c_1	$1.0227\ 10^{-4}$	g_1	$1.6483\ 10^{-2}$	h_1	$1.43713\ 10^{-3}$		
c_2	$-1.6546\ 10^{-6}$	g_2	$-5.3009\ 10^{-4}$	h_2	$1.16092\ 10^{-4}$		
				h_3	$-5.77905\ 10^{-7}$		
d_0	$4.8314\ 10^{-4}$	j_0	$1.91075\ 10^{-4}$				
m_1	$-9.9348\ 10^{-7}$	i_0	$2.2838\ 10^{-3}$	k_0	$8.50935\ 10^{-5}$		
m_2	$2.0816\ 10^{-8}$	i_1	$-1.0981\ 10^{-5}$	k_1	$-6.12293\ 10^{-6}$		
m_3	$9.1697\ 10^{-10}$	i_2	$-1.6078\ 10^{-6}$	k_2	$5.2787\ 10^{-8}$		

Table 6.2: **Constants for UNESCO Formula:** The subscripts denote the exponent of respective variables. See also Eqn. (1.17).

Part IV

References

Chapter 7

How to Find References

7.1 Code References

Name in Code	Name in Formulae	Meaning	Unit	Equation
UH	ρhu	zonal mass flux	$kgm^{-1}s^{-1}$	(1.1)
VH	ρhv	meridional mass flux	$kgm^{-1}s^{-1}$	(1.1)
HEIGTH	h	layer thickness	m	(1.2)
TEMP	θ	potential temperature	K	(1.3)
SALT	S	salinity	gkg^{-1}	(1.4)

Table 7.1: **Names for Prognostic Ocean Quantities**

Name in Code	Name in Formulae	Meaning	Unit	Equation
U	u	zonal velocity	ms^{-1}	(1.1)
V	v	meridional velocity	ms^{-1}	(1.1)
DENSITY	ρ	in situ density	kgm^{-3}	(1.6)
POTDENS	σ_θ	potential density	kgm^{-3}	(1.6)
PRESS	P	in situ pressure	m^2s^{-2}	(1.8)
DIFFUSE	A^s	scalar diffusion coefficient	m^2s^{-1}	(1.12)
VERTUP	w_k^{k+}	upward diffusion	ms^{-1}	(1.73)
VERTDN	w_k^{k-}	downward diffusion	ms^{-1}	(1.74)
LEVUPP	$k-$	index of next upper layer		(1.73)
LEVLOW	$k+$	index of next lower layer		(1.74)
RESTU	$F^{*n}_{\rho uh}$	right side of zonal flux	$kgm^{-1}s^{-1}$	(2.37)
RESTV	$F^{*n}_{\rho vh}$	right side of meridional flux	$kgm^{-1}s^{-1}$	(2.38)
RESTH	$F^{*n}_{\rho h}$	right side of wave equation	kgm^{-2}	(2.40)
POTSOLL	σ^*_θ	potential density coordinate	kgm^{-3}	(1.6)

Table 7.2: **Names for Diagnostic Ocean Quantities**

Name in Code	Name in Formulae	Meaning	Unit	Equation
TAUX	τ_x	zonal wind stress	$kgm^{-1}s^{-2}$	(1.1)
TAUY	τ_y	meridional wind stress	$kgm^{-1}s^{-2}$	(1.1)
QFLUX	Q	net surface heat flux	Wm^{-2}	(1.33)
QSOLAR	$Q_{s,bot}$	downward solar heat flux	Wm^{-2}	(1.44)
QSENSIB	Q_H	surface sensible heat flux	Wm^{-2}	(1.35)
QLATENT	Q_L	surface latent heat flux	Wm^{-2}	(1.36)
QLONG	$Q_{l,bot}$	longwave radiation budget	Wm^{-2}	(1.39)
SFLUX	R	equivalent salt flux	$WK^{-2}m^{-2}$	(1.63)
BFLUX	B	net surface buoyancy flux	m^2s^{-3}	(1.20)
BSOLAR	B_s	solar buoyancy flux	m^2s^{-3}	(1.21)
USTERN	u_*	friction velocity	ms^{-1}	(1.64)
HEATLAT	L_w	heat of fusion	$Wskg^{-1}$	(1.36)
SIGMA	σ	Stefan Boltzmann constant	$kgm^{-1}s^{-1}$	(1.39)
SOLAR	S_0	solar constant	Wm^{-2}	(1.44)
CPAIR	$c_{p,air}$	specific heat capacity	$Wskg^{-1}K^{-1}$	(1.35)
CHARNCK	c_{char}	Charnock constant		(1.54)
RKARMAN	κ	von Karman constant		(1.51)
ALBEDO	ω	albedo		(1.44)
ABSORB	ε	emmisivity of water		(1.44)
TURBID	h_B	penetration depth of $Q_{s,bot}$	m	(1.33)
TURBREL	γ	transmission coefficient		(1.33)

Table 7.3: **Names for Surface Flux Parameters**

Name in Code	Name in Formulae	Meaning	Unit	Equation
HEATQ	Q_{run}	running mean heat flux	Wm^{-2}	(5.9)
HEATM	Q_{bias}	heat flux bias correction	Wm^{-2}	(5.13)
PMEQ	F_{run}	running mean fresh water	Wm^{-2}	(5.10)
PMEM	F_{bias}	fresh water flux bias correction	Wm^{-2}	(5.14)
TAUXQ	$\tau_{x,run}$	running mean zonal wind stress	Nm^{-2}	(5.11)
TAUXM	$\tau_{x,bias}$	zonal wind stress bias correction	Nm^{-2}	(5.15)
TAUXQ	$\tau_{y,run}$	running mean meridional wind stress	Nm^{-2}	(5.12)
PMEM	$\tau_{y,bias}$	meridional wind stress bias correction	Nm^{-2}	(5.16)
FACDEC	δ_{run}	relaxation factor		(5.13)

Table 7.4: **Names for Bias Correction Terms**

Name in Code	Name in Formulae	Meaning	Unit	Equation
UICE	$(uh)_i$	zonal flux of ice	kgm^2s^{-1}	(1.77)
VICE	$(vh)_i$	meridional flux of ice	kgm^2s^{-1}	(1.77)
HICE	h_i	ice thickness	m	(1.78)
COMPACT	q_i	ice compactness		(1.79)
HSNOW	s_i	snow depth	m	(1.80)
TICE	T_h	ice temperature	K	(1.87)
TSNOW	T_s	snow temperature	K	(1.88)
DMELT	F_h	melting of ice & snow per Δt	m	(1.95)

Table 7.5: **Names for Prognostic Snow and Sea Ice Variables**

Name in Code	Name in Formulae	Meaning	Unit	Equation
VERTIC	w	entrainment rate	ms^{-1}	(1.18)
HEQU	h_M	Monin-Obukhov length	m	(1.29)
ENERGY	u_*^3	turbulent kinetic energy	m^3s^{-3}	(1.64)
TAUX	τ_x	Garwood-term	$kgm^{-1}s^{-2}$	(1.18)

Table 7.6: **Names for Diagnostic Mixed Layer Variables**

Name in Code	Name in Formulae	Meaning	Unit	Equation
DIFFUSV	A_V	factor for momentum diffusion	m^2s^{-1}	(1.13)
DIFMINV		threshold for momentum diffusion	m^2s^{-1}	(1.10)
DIFFUSS	A^s	factor for scalar diffusion	m^2s^{-1}	(1.15)
DIFMINS		threshold value for scalar diffusion	m^2s^{-1}	(1.12)
DIADIF0	$A^{h,0}$	diffusion for layer thickness	m^2s^{-1}	(1.11)
DIADIF1	$A^{h,1}$	factor for layer thickness diffusion	m^2s^{-1}	(1.11)
TURBLEV	$2m_ou_{*v}^3$	TKE for vertical mixing	m^3s^{-3}	(1.68)
HMIXV	h_w	reference thickness	m	(1.68)
DRAGT	δ	time constant for salinity forcing	s^{-1}	(1.63)
DRAGS	c_d	drag coefficient at surface	s^{-1}	(1.1)
DRAGB	c_d	drag coefficient at bottom	s^{-1}	(1.1)

Table 7.7: **Names for Horizontal and Vertical Diffusion Parameters**

Name in Code	Name in Formulae	Meaning	Unit	Equation
CMIX0	m_4	linear damping term of ML	m^2s^{-3}	(1.18)
CMIX1	h_{tke}	constant TKE decay length scale	m	(1.27)
CMIX2	Ri_{crit}	critical Richardson number		(1.30)
CMIX3	$1/\kappa$	Ekman dissipation for TKE		(1.25)
CMIX4	m_o	wave enery conversion factor		(1.18)
CMIX5	m_3	Garwood term tuning coefficient		(1.18)
CMIX6	h_{buo}	constant buoyancy decay length scale	m	(1.28)
CMIX7	$1/\mu$	Ekman dissipation for buoyancy		(1.26)
CMIX8	hg'_o	threshold for hg'	m^2s^{-2}	(1.18)
CMIX9	m_5	tuning coefficient for buoyancy flux		(1.18)
TURBEN	m_6	tuning parameter	m^3s^{-3}	(1.18)
HMIXMIN	h_{min}	minimal mixed layer thickness	m	(1.31)

Table 7.8: **Names for Mixed Layer Parameters**

Name in Code	Name in Formulae	Meaning	Unit	Equation
EDDYUV	A_i^v	diffusion coefficient	$m^2 s^{-1}$	(1.77)
EDDYS	A_i^s	diffusion coefficient	$m^2 s^{-1}$	(1.78)
RHOICE	ρ_i	ice density	kgm^{-3}	(1.87)
RHOSNOW	ρ_s	snow density	kgm^{-3}	(1.88)
CICE	k_i	heat conductivity for ice	$WK^{-1}m^{-1}$	(1.87)
CSNOW	k_s	heat conductivity for snow	$WK^{-1}m^{-1}$	(1.88)
CPICE	$c_{p,i}$	specific heat capacity of ice	$Wskg^{-1}K^{-1}$	(1.89)
CPMELT	$c_{p,m}$	heat of fusion	$Wskg^{-1}$	(1.95)
EXCENT	e	excentricity		(1.86)
PRESICE	P_i	proportionality	$m^2 s^{-2}$	(1.84)
PDECAY	ϵ_1	proportionality	$m^2 s^{-2}$	(1.84)
HNULL	h_0	tuning coefficient	m	(1.96)
EPSNULL	ϵ_0	tuning coefficient		(1.85)
AGING	γ	snow aging parameter	s^{-1}	(1.97)
SALTICE	S_i	salinity of sea ice	gkg^{-1}	(1.24)
PENET	h_p	parameter for salt ejection	$m^3 kg^{-1}$	(1.100)

Table 7.9: **Names of Snow and Sea Ice Parameters**

Name in Code	Name in Formulae	Meaning	Unit	Equation
ATU	u	zonal surface wind component	ms^{-1}	(1.1)
ATV	v	meridional surface wind component	ms^{-1}	(1.1)
UVABSOL	V	wind speed	ms^{-1}	(1.64)
UVARIAN	$\sigma(V)$	standard deviation of wind speed	ms^{-1}	(1.64)
AIRT	T_a	surface air temperature	K	(1.35)
SST		sea surface temperature	K	
SSS	S_{obs}	sea surface salinity	gkg^{-1}	(1.63)
CLOUDS	n	cloudiness		(1.43)
HUMID	r	relative humidity		(1.38)
SURPRES	P_{atm}	atmospheric sea level pressure	Nm^{-2}	(1.7)
HUMID	r	relative humidity		(1.38)
DEPTH	D	topography depth	m	(1.9)

Table 7.10: **Names of Observed Quantities**

Name in Code	Name in Formulae	Meaning	Unit	Equation
TEBM	ϵ_1	weight for temperature transport		(1.102)
HEBM	ϵ_2	weight for moisture transport		(1.103)
DIFWGHT	ϵ_3	weight for eddy diffusion		(1.105)
	ϵ_4	weight for precipitation		(1.106)
	ϵ_5	weight for latent heat conversion		(1.106)
	ϵ_6	weight for large-scale precipitation		(1.107)
	ϵ_7	weight for convective precipitation		(1.107)
FLOWFAC	ϵ_8	weight for river runoff velocity		(1.114)
	H_s	thickness of the soil	m	(1.115)
HHUMY	H_v	effective thicknes of the moist layer	m	(1.108)
QINSOL	$Q_{s,top}$	top downward solar heat flux	Wm^{-2}	(1.111)
QINLON	$Q_{l,top}$	top longwave radiative heat flux	Wm^{-2}	(1.109)

Table 7.11: **Names of EBM Parameters**

Name in Code	Name in Formulae	Meaning	Unit	Equation
UHTIDE	$(uh)_{tide}$	zonal barotropic tidal flow	m^2s^{-1}	(1.116)
VHTIDE	$(vh)_{tide}$	meridional barotropic tidal flow	m^2s^{-1}	(1.117)
ZHTIDE	ζ_{tide}	tidal sea level anomaly	m	(1.118)
UTIDE	u_{res}	zonal residual flow due to tides	ms^{-1}	(1.124)
VTIDE	v_{res}	meridional residual flow due to tides	ms^{-1}	(1.124)
U(NZ,I,J)	u_N	zonal flow in the lowest layer	ms^{-1}	(1.117)
V(NZ,I,J)	v_N	meridional flow in the lowest layer	ms^{-1}	(1.117)
EQUTIDE	E_{tide}	gravitational tidal forcing	ms^{-2}	(1.123)

Table 7.12: **Names of Tide Parameters**

7.2 Acknowledgements

I would like to thank Trond Aukrust (Bergen Scientific Centre, Bergen) for his contributions to the grid rotation by Eulerian angles, David Holland (McGill University, Montreal) for the bookkeeping of the budget terms and Frank Kauker (GKSS, Geesthacht) for the open boundary conditions. I also appreciate the valuable comments by Arthur Miller at the initial state of the model and Andreas Bacher for his comments on numerous coupling aspects and related improvements.

7.3 Literature References

7.3.1 Text References

Bleck, R., and D.B. Boudra, 1981: Initial testing of a numerical ocean circulation model using a hybrid (quasi-isopycnic) vertical coordinate. *J. Phys. Oceanogr.*, **1**, 755-770.

Bryden, H., 1973: New polynomials for thermal expansion, adiabatic temperature gradient and potential temperature of sea water. *Deep Sea Res.*, **20**, 401-408.

Denman, K.L. and M. Miyake, 1973; Upper Layer Modification at Ocean Station Papa: Observations and Simulations. *J. Phys. Oceanogr.*, **3**, 185-196.

Garwood, R.W., 1977: An oceanic mixed layer model capable of simulating cyclic states. *J. Phys. Oceanogr.*, **7**, 455-468.

Garwood R.W., Jr., P.C. Gallacher and P. Müller, 1985a: Wind Direction and Equilibrium Mixed layer Depth: General Theory. *J. Phys. Oceanogr.*, **15**, 1525-1531.

Garwood R.W., Jr., P.C. Gallacher and P. Müller, 1985b: Wind Direction and Equilibrium Mixed Layer Depth in the Tropical Pacific Ocean. *J. Phys. Oceanogr.*, **15**, 1532-1538.

Gaspar, P., 1988: Modeling the Seasonal Cycle of the Upper Ocean. *J. Phys. Oceanogr.*, **18**, 161-180.

Gibson, J.K., O. Kallberg, S. Uppala, A. Hernandez, A. Nomura and E. Serrano, 1997: ECMWF reanalysis project report series, 1. ERA description. *European Centre for Medium-Range Weather Forecasts*, Reading/UK, 72pp.

Hellermann, S., and M. Rosenstein, 1983: Normal monthly wind stress over the world ocean with error estimates. *J. Phys. Oceanogr.*, **13**, 1093-1104.

Hibler, W.D., III, 1979: A Dynamic Thermodynamic Sea Ice Model. *J. Phys. Oceanogr.*, **9**, 815-846.

Kleeman, R., and S.B. Power, 1995: A simple atmospheric model of the surface heat flux for use in ocean modeling studies. *J. Phys. Oceanogr.*, **25**, 92-105.

Kraus E.B., and J.S. Turner, 1967: A one-dimensional model of the seasonal thermocline. *Tellus*, **1**, 88-97.

Large, W.G., and S. Pond, 1981: Open Ocean Momentum Measurements in Moderate to Strong Winds. *J. Phys. Oceanogr.*, **11**, 324-336.

Large, W.G., and S. Pond, 1982: Sensible and Latent Heat Flux Measurements over the Sea. *J. Phys. Oceanogr.*, **12**, 464-482.

Legates, D.R. and C.J. Willmott, 1990: Mean seasonal and spatial variability in gauge-corrected global precipitation. *Internat. J. Climatol.*, **9**, 111-127.

Martin, P.J., 1985: Simulation of the Mixed Layer at OWS November and Papa With Several Models. *J. Geoph. Res.*, **90**, 903-916.

Mikolajewicz, U., and E. Maier-Reimer, 1994: Mixed boundary conditions in ocean general circulation models and their influence on the stability of the model's conveyor belt. *J. Geophys. Res.*, 99:22633-22644.

Millero, F.J., 1978: Freezing point of seawater. Eighth Report of the Joint Panel on Oceanographic Tables and Standards, *Unesco Tech. Pap. in Mar. Sci.*, No. **28**, Annex 6, UNESCO, Paris.

Niiler, P.P., 1975: Deepening of the wind-mixed layer. *J. Mar. Res.*, **33**, 405-422.

Niiler, P.P., and E.B. Kraus, 1977: One-dimensional model of the seasonal thermocline. *The Sea*, Vol. **VI**, Wiley Interscience, 97-115.

North, G.R., J.G. Mengel and D.A. Short, 1983: Simple Energy Balance Model Resolving the Seas and the Continents: Application to the Astronomical Theory of Ice Ages. *J. Geophys. Res.*, 88:6576-6586.

Oberhuber J.M., 1986: About Some Numerical Methods Used in an Ocean General Circulation Model with Isopycnic Coordinates. Advanced Physical Oceanographic Numerical Modelling, NATO ASI Series, Series C: Mathematical and Physical Sciences Vol. **186**, 511-522.

Oberhuber, J.M., 1988: An atlas based on the 'COADS' data set: The budgets of heat, buoyancy and turbulent kinetic energy at the surface of the global ocean. Max-Planck-Institute for Meteorology, Hamburg, Report No. **15**, 199pp.

Oberhuber, J.M., 1990: Simulation of the Atlantic Circulation with a coupled Sea Ice - Mixed Layer - Isopycnal General Circulation Model. Max-Planck-Institute for Meteorology, Hamburg, Report No. **59**, 86pp.

Oberhuber, J.M., 1993a: Simulation of the Atlantic Circulation with a Coupled Sea Ice - Mixed Layer - Isopycnal General Circulation Model. Part I: Model Description. *J. Phys. Oceanogr.*, Vol. 23, No. 5, 808-829.

Oberhuber, J.M., 1993b: Simulation of the Atlantic Circulation with a Coupled Sea Ice - Mixed Layer - Isopycnal General Circulation Model. Part II: Model Experiment. *J. Phys. Oceanogr.*, Vol. 23, No. 5, 830-845.

Oberhuber, J.M., 1993c: The OPYC Ocean General Circulation Model. Technical Report No. 7, Deutsches Klimarechenzentrum GmbH, Hamburg.

Paulson, C.A., J.J. Simpson, 1977: Irradiance measurements in the upper ocean. *J. Phys. Oceanogr.*, **7**, 952-956.

Robert, A., J. Henderson, and C. Turnbell, 1972: An implicit time step scheme for baroclinic models of the atmosphere. *Mon. Wea. Rev.*, **100**, 329-335.

Schwiderski, E.W., 1979: Ocean Tides, Part I: Global Tidal Equations. *Marine Geodesy*, **3**, 161-179.

Shea, D.J., 1986: Climatological Atlas: 1950-1979, National Center for Atmospheric Research, Boulder, Colorado.

Smagorinsky, J.S., 1963: General circulation experiments with the primitive equations. I: The basic experiment. *Mon. Wea. Rev.*, **91**, 99-164.

Smolarkiewicz, P.K., 1982: The multi-dimensional Crowley Advection Scheme. *Mon. Wea. Rev.*, **110**, 1968-1983.

UNESCO, 1981: The Practical Salinity Scale 1978 and the International Equation of State of Seawater 1980. *Unesco Techn. Pap. in Mar. Sci.*, No. **36**, 13-21.

Woodruff, S.D., R.J. Slutz, R.L. Jenne and P.M. Steurer, 1987: A Comprehensive Ocean-Atmosphere Data Set. *Bull. Amer. Met. Soc.*, **68**, 1239-1250.

Wright, P., 1988: An Atlas based on the 'COADS' data set: Fields of mean wind, cloudiness and humidity at the surface of the global ocean. Max-Planck-Institute for Meteorology, Hamburg, Report No. **14**, 70pp.

Zillmann, J.W., 1972: A study of some aspects of the radiation and the heat budgets of the southern hemisphere oceans. *Meteorol. Stud.*, **26**, 562pp, Bur. of Meteorol., Dept. of the Interior, Canberra, Australia.

7.3.2 Key Publications with PIPE - sorted in time

Oberhuber, J.M., 1988: An atlas based on the "COADS" data set: The budgets of heat, buoyancy and turbulent kinetic energy at the surface of the gloabl ocean. Max-Planck-Institut für Meteorologie, Hamburg, Report No. **15**, 199pp.

Miller, A.J., J.M. Oberhuber, N.E. Graham and T.P. Barnett, 1992: Tropical Pacific Ocean Response to observed Winds in a Layered General Circulation Model. *J. Geophys. Res.*, Vol. 97, No. C5, 7117-7340.

Holland, D.M., L.A. Mysak, D.K. Manak and J.M. Oberhuber, 1993: Sensitivity Study of a Dynamic Thermodynamic Sea Ice Model. *J. Geophys. Res.*, Vol. 98, No. C2, 2561-2586.

Oberhuber, J.M., 1993: The OPYC Ocean General Circulation Model. Technical Report No. 7, Deutsches Klimarechenzentrum GmbH, Hamburg.

Oberhuber, J.M., 1993a: Simulation of the Atlantic Circulation with a Coupled Sea Ice - Mixed Layer - Isopycnal General Circulation Model. Part I: Model Description. *J. Phys. Oceanogr.*, Vol. 23, No. 5, 808-829.

Oberhuber, J.M., 1993b: Simulation of the Atlantic Circulation with a Coupled Sea Ice - Mixed Layer - Isopycnal General Circulation Model. Part II: Model Experiment. *J. Phys. Oceanogr.*, Vol. 23, No. 5, 830-845.

Miller, A.J., D.R. Cayan, T.P. Barnett, N.E. Graham and J.M. Oberhuber, 1994a: Interdecadal variability of the Pacific Ocean: model response to observed heat fluxes and wind stress anomalies. *Clim. Dyn.*, Vol. 9, 287-302.

Miller, A.J., D.R. Cayan, T.P. Barnett, N.E. Graham and J.M. Oberhuber, 1994b: The 1976-77 Climate Shift of the Pacific Ocean. *Oceanography*, Vol. 7, 21-26.

Oberhuber, J.M., 1995 Parallelization of an ocean general circulation model on the CRAY T3D system. *CRAY CHANNELS*, Vol. 17, No. 2, 22-25.

Holland, D.M., R.G. Ingram, L.A. Mysak and J.M. Oberhuber, 1995: A numerical simulation of the sea ice cover in the northern Greenland Sea. *J. Geophys. Res.*, Vol. 100, No. C3, 4751-4760.

Oberhuber, J.M., and K. Ketelsen, 1995: Parallelization of an OGCM on the CRAY T3D. *Proceedings of the Sixth ECMWF Workshop on the Use of Parallel Processors in Meteorology*, Title: Coming of Age. Editor: G.-R. Hoffmann and N. Kreitz, World Scientific Publishing.

Aukrust, T., and J.M. Oberhuber, 1995: Modeling of the Greenland, Iceland and Norwegian Seas with a coupled sea ice - mixed layer - isopycnal ocean model. *J. Geophys. Res.*, Vol. 100, No. C3, 4771-4789.

Duffy, P.B., D.E. Eliason, A.J. Bourgeois and C.C. Covey, 1995: Simulation of bomb radiacarbon in two global ocean general circulation models. *J. Geophys. Res.*, Vol. 100, No. C11, 22,545-22,563.

Oberhuber, J.M., 1996: Surface flux fields for ocean models. *WCRP-Report WMO/TD-No. 762*, 53-58.

Holland, D.M., L.A. Mysak and J.M. Oberhuber, 1996a: An investigation of the general circulation of the Arctic Ocean using an isopycnal model. *Tellus*, Vol. 48A, 138-157.

Holland, D.M., L.A. Mysak and J.M. Oberhuber, 1996b: Simulation of the mixed-layer circulation in the Arctic Ocean. *J. Geophys. Res.*, Vol. 101, No. C1, 1111-1128.

Cherniawsky, J.Y., and J.M. Oberhuber, 1996: The seasonal cycle of mixed layer temperatures in a global ocean general circulation model. *Clim. Dyn.*, Vol. 12, 171-183.

Lunkeit, F., R. Sausen and J.M. Oberhuber, 1996: Climate simulations with the global coupled atmosphere-ocean model ECHAM2/OPYC. *Clim. Dyn.*, Vol. 12, 195-212.

Roeckner, E., J.M. Oberhuber, A. Bacher, M. Christoph and I. Kirchner, 1996: ENSO variability and atmospheric response in a global coupled atmosphere-ocean GCM. *Clim. Dyn.*, Vol. 12, 737-754.

Stössel, A., J.M. Oberhuber and E. Maier-Reimer, 1996: On the representation of sea ice in global general circulation models. *J. Geophys. Res.*, Vol. 101, No. C8, 18,193-18,212.

Kauker, F., and J.M. Oberhuber, 1997: An Isopycnal Ocean Circulation Model of the North Sea for Dynamical Downscaling. *GKSS 97/E/47*, GKSS Forschungszentrum GmbH, Geesthacht.

Tyler, R.H., L.A. Mysak and J.M. Oberhuber, 1997: Electromagnetic fields generated by a three-dimensional global ocean circulation. *J. Geophys. Res.*, Vol. 102, No. C3, 5531-5551.

Tyler, R.H., T.B. Sanford and J.M. Oberhuber, 1997: Geophysical Challenges in Using Large-Scale Ocean-Generated EM Fields to Determine the Ocean Flow. *J. Geomag. Geoelectr.*, Vol. 49, 1351-1372.

Nies, H., I.H. Harms, M.J. Karcher, D. Dethleff, C. Bahe, G. Kuhlmann, J.M. Oberhuber, J.O. Backhaus, E. Kleine, P. Löwe, D. Matishov, A. Stephanov and O.F. Vasiliev, 1998: Anthropogenic radiaoactivity in the Nordic Seas and the Arctic Ocean - Results of a joint project. Project Report, BMBF.

Christoph, M., T.B. Barnett and E. Roeckner, 1998: The Antarctic Circumpolar Wave in a Coupled Ocean-Atmosphere GCM. *J. Climate*, Vol. 11, 1659-1672.

Cabos Narvaez, W.,M.J. OrtizBeviá and J.M. Oberhuber, 1998: The variability of the tropical Atlantic. *J. Geophys. Res.*, Vol. 103, No. C4, 7475-7489.

Bacher, A., J.M. Oberhuber and E. Roeckner, 1998: ENSO dynamics and seasonal cycle in the tropical Pacific as simulated by the ECHAM4/OPYC3 coupled general circulation model. *Clim. Dyn.*, Vol. 14, 432-450.

Zhang, X.-H., J.M. Oberhuber, A. Bacher and E. Roeckner, 1998: Interpretation of interbasin exchange in an isopycnal ocean model. *Clim. Dyn.*, Vol. 14, 725-740.

Timmermann, A., J.M. Oberhuber, A. Bacher, M. Esch, M. Latif and E. Roeckner, 1998: ENSO Response to Greenhouse Warming. Max-Planck-Institut für Meteorologie, Hamburg, Report No. **251**, *Nature, in press.*

Oberhuber, J.M., E. Roeckner, M. Christoph, M. Esch and M. Latif, 1998: Predicting the '97 El Niño event with a global climate model. *Geophys. Res. Let.*, Vol. 25, No. 13, 2273-2276.