Report No. 7

The OPYC Ocean General Circulation Model

The Description of a Coupled Snow, Sea-Ice, Mixed Layer and Isopycnal Ocean Model

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

Edited by:
Deutsches Klimarechenzentrum GmbH
Modellbetreuungsgruppe
Hamburg, October 1993
Revision 1

Table of Contents

DKRZ OPYC Model Documentation

List of Tables

List of Figures

DKRZ OPYC Model Documentation

1. SUMMARY PAGE

1.1 SHORT DESCRIPTION OF THE MODEL

The concept to use isopycnals as the vertical coordinate system for an OGCM is based on the observation that the interior ocean behaves as a rather 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 explictly excluded.

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 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.

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

The name OPYC is derived from Ocean and isoPYCnal coordinates.

1.2 AUTHOR OF THE MODEL

- Josef Maximilian Oberhuber.
  Developed at the Max-Planck-Institut für Meteorologie, Hamburg, FRG,
  continued at the Meteorologisches Institut der Universität Hamburg, FRG

1.3 PERSON RESPONSIBLE FOR MODEL SUPPORT AT THE DKRZ

- Josef Maximilian Oberhuber,
  DKRZ Model Support Group, Hamburg, FRG

2. MODEL PHYSICS AND DYNAMICS

Remark:

This manual is based on the model published in Oberhuber (1990, 1993a, 1993b) henceforth called JMO. However, in JMO only one of many available options for selecting the physics and the layout of the model geometry is discussed. This manual describes Cycle 2.17 of OPYC. All parameter settings are listed in chapter 5 and referenced in chapter 7.

Notation:

The subsequent physical description references the code in the following way:

- a variable is written in italic type, e.g. *TEMP* for temperature,
- a SUBROUTINE is referenced by enclosing the name within '<'and'>', e.g. <*SOLVER*> for the solution of the block-tridiagonal system,
- 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 name is enclosed by '['and']' as e.g. [*ocestep.f*] which is the kernel of OPYC.

2.1 THE INTERIOR OCEAN MODEL

2.1.1 Physical Background

OPYC is a general ocean circulation model based on the following ideas: to use isopycnals as Lagrangian vertical coordinates, to include a realistic equation of state, to solve the primitive equations and to fully couple a surface mixed layer and snow & sea ice model to the interior ocean. The equation of state together with the sea ice model allows the full model to be used on global scales. However, no approximation is made (except the hydrostatic approximation) that prohibits an application down to the 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 and possibly tracer concentrations. 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 the moving coordinate system. Potential vorticity and energy can also be conserved after some manipulation of the momentum transport and Coriolis terms.

2.1.2 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 $(\theta\rho h)_k$, the salt content $(S\rho h)_k$ and a tracer content $(C\rho h)_k$ in a column of the k-th layer

$$\frac{\partial}{\partial t}(\rho\vec{v}h)_k = -\vec{\nabla}(v_k(\rho\vec{v}h)_k) - h_k\vec{\nabla}p_k - \vec{f}\times(\rho\vec{v}h)_k + \vec{\nabla}\cdot A_k^v\vec{\nabla}(\rho\vec{v}h)_k + (w\rho\vec{v})_k^{k+} + (w\rho\vec{v})_k^{k-} + (-(w\rho\vec{v})_{k+}^{k}) - (w\rho\vec{v})_{k-}^{k} + \vec{\tau}_k^{k-} + \vec{\tau}_k^{k+} \tag{2.1.2.1}$$

$$\frac{\partial}{\partial t}(\rho h)_k = -\vec{\nabla}\cdot(\rho\vec{v}h)_k + (w\rho)_k^{k+} + (w\rho)_k^{k-} - (w\rho)_{k+}^{k} - (w\rho)_{k-}^{k} + R_k^h + \vec{\nabla}\cdot A_k^h\nabla(\rho h)_k \tag{2.1.2.2}$$

$$\frac{\partial}{\partial t}(\theta\rho h)_k = \vec{\nabla}\cdot((\theta\rho h)\vec{v})_k + \vec{\nabla}\cdot A_k^s\vec{\nabla}(\theta\rho h)_k + \frac{Q_k}{c_p} + (w\rho\theta)_k^{k-} + (w\rho\theta)_k^{k+} - (w\rho\theta)_{k-}^{k} - (w\rho\theta)_{k+}^{k} \tag{2.1.2.3}$$

$$\frac{\partial}{\partial t}(S\rho h)_k = \vec{\nabla}\cdot((S\rho h)\vec{v})_k + \vec{\nabla}\cdot A_k^s\vec{\nabla}(S\rho h)_k + R_k^S + (w\rho S)_k^{k-} + (w\rho S)_k^{k+} - (w\rho S)_{k-}^{k} - (w\rho S)_{k+}^{k} \tag{2.1.2.4}$$

$$\frac{\partial}{\partial t}(C\rho h)_k = \vec{\nabla}\cdot((C\rho h)\vec{v})_k + \vec{\nabla}\cdot A_k^s\vec{\nabla}(C\rho h)_k + R_k^C + (w\rho C)_k^{k-} + (w\rho C)_k^{k+} - (w\rho C)_{k-}^{k} - (w\rho C)_{k+}^{k} \tag{2.1.2.5}$$

The terms $(w\rho\vec{v})$, $(w\rho)$, $(w\rho\theta)$, $(w\rho S)$ and $(w\rho C)$ describe exchange processes of mass fluxes, mass itself, heat, salt and tracer content between neighbouring layers. 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 (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, R^S the forcing due to the sea - ice ocean coupling and R^C the forcing of the tracer as surface input and e.g. radioactive decay in the interior ocean. c_p is the specific heat capacity of the water.

The parameter f in equation (2.1.2.1) is the Coriolis parameter:

$$f = 2\Omega\sin\varphi \quad \text{with} \quad \Omega = \frac{2\pi}{86164} \tag{2.1.2.6}$$

where Ω is the angular velocity of the rotating earth. The stress $\vec{\tau} = (\tau_x, \tau_y)$ between neighbouring layers is parameterized as

$$\vec{\tau}_k^l = c_d \rho_a \sqrt{\Delta\vec{v}^2}\,\Delta\vec{v} \quad \text{with} \quad \Delta\vec{v} = (\Delta u, \Delta v) = (u_l - u_k, v_l - v_k) \tag{2.1.2.7}$$

where ρ_a is the surface air density. Since observed surface stresses are used, the surface drag coefficient does not need to be specified. For the internal stresses different drag coefficients can be chosen for interfacial and bottom friction.

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

$$\rho_k = \rho_k(T_k, S_k, P_k) \tag{2.1.2.8}$$

After discretization, the in situ pressure in the first layer is given as a vertical mean by

$$P_1 = \frac{g}{2}(\rho h)_1 \tag{2.1.2.9}$$

where *g* is the gravitational constant. In all deeper layers the in situ pressure P_k is given by

$$P_k = P_{k-1} + \frac{g}{2}((\rho h)_{k-1} + (\rho h)_k) \tag{2.1.2.10}$$

The pressure gradient in a layer *k* is computed by the sum of gradients due to the sea level gradient, the gradients in 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:

$$\vec{\nabla} p_k = g\rho_k\left(\sum_{l=k}^{N}\vec{\nabla}h_l + \vec{\nabla}D\right) + g\frac{\rho_k}{\sigma_{\theta k}}\left(\sum_{l=1}^{k-1}\vec{\nabla}(h\sigma_\theta)_l + \frac{h_k}{2}\nabla\sigma_{\theta k}\right) \tag{2.1.2.11}$$

where *D* is the height of the topography above some reference level.

The in situ temperature T_k is calculated by inverting the formula of Bryden (1973), which relates potential temperature to temperature, salinity and pressure:

$$\theta_k = \theta_k(T_k, S_k, P_k) \tag{2.1.2.12}$$

By combining the UNESCO formula for density with the formula for potential temperature, the potential density σ_θ is defined by using the potential temperature and salinity in the same layer and reducing it to a specified reference pressure level (chosen here to be the surface).

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:

$$A_k^v = A^{v,1}\sqrt{\left(\frac{\partial u_k}{\partial x}-\frac{\partial v_k}{\partial y}\right)^2+\left(\frac{\partial u_k}{\partial y}+\frac{\partial v_k}{\partial x}\right)^2}+A^{v,0} \tag{2.1.2.13}$$

$$A_k^h = A^{h,1}\sqrt{\left(\frac{\partial u_k}{\partial x}-\frac{\partial v_k}{\partial y}\right)^2+\left(\frac{\partial u_k}{\partial y}+\frac{\partial v_k}{\partial x}\right)^2}+A^{h,0} \tag{2.1.2.14}$$

$$A_k^s = A^{s,1}\sqrt{\left(\frac{\partial u_k}{\partial x}-\frac{\partial v_k}{\partial y}\right)^2+\left(\frac{\partial u_k}{\partial y}+\frac{\partial v_k}{\partial x}\right)^2}+A^{s,0} \tag{2.1.2.15}$$

The values for $A^{s,1}$ and $A^{v,1}$ have to be chosen according to the grid resolution (see also Table 37). The diffusion coefficients $A^{v,0}$, $A^{h,0}$ and $A^{s,0}$ depend on the grid spacing, which is variable in longitude and latitude. This is useful if a grid with highly variable resolution is chosen. By defining the damping time scales T_v for a vector quantity and T_s for a scalar quantity the final diffusion coefficient automatically satisfies the local numerical stability requirements at each grid point (see also Table 42)

$$A^{v,0} = \frac{\Delta x^2+\Delta y^2}{2T_v} \tag{2.1.2.16}$$

$$A^{s,0} = \frac{\Delta x^2+\Delta y^2}{2T_s} \tag{2.1.2.17}$$

Here Δx and Δy are the grid distances in the spherical coordinate system. This formulation ensures a reasonable automatic choice of diffusion coefficients in low and high resolution areas, as well as in low and high latitudes. The fact that diffusivity is chosen to be stronger for momentum than for the scalar quantities is governed primarily by the need to couple the two separate solutions in the B-grid (this problem does not arise in the equations for $(\theta\rho h)$, $(S\rho h)$ and $(C\rho h)$.

2.2 THE MIXED LAYER MODEL

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

2.2.1 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 TKE is partly converted into mean potential energy, MPE, and partly dissipated. One of the main problems is that the different kinds of 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 of different parts of the ocean. A summary of some difficulties concerning earlier ML-models is given in Gaspar (1988).

Since the 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 (2.1.2.1) to (2.1.2.5) 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.

2.2.2 The Mixed Layer Model Equations

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

$$\begin{aligned} wh_1(g' + g'_0) - wRi_{crit}((u_1 - u_{1+})^2 + (v_1 - v_{1+})^2) + m_6 &= 2m_0 m_1 u_*^3 + \Gamma h_1 m_2 m_5 B_I \\ &+ (1-\Gamma) h_1 m_2 m_5 (B - m_3 B_s) \\ + (1-\Gamma)\gamma m_2 m_5 B_s \left[h_1 \left(1 + exp\left(-\frac{h_1}{h_B}\right)\right) - 2h_B \left(1 - exp\left(-\frac{h_1}{h_B}\right)\right)\right] & \\ &- m_3 \frac{12}{7} m_0 \tau_x \Omega_y - m_4 h_1 \end{aligned} \qquad (2.2.2.1)$$

where

$$g' = g \frac{\sigma_{\theta 1+} - \sigma_{\theta 1}}{\sigma_{\theta 1}} (1-\Gamma) + \Gamma g'_I \qquad (2.2.2.2)$$

$$B = \frac{g}{c_p \rho_1^2} (\alpha Q + \beta R) \tag{2.2.2.3}$$

$$R = \frac{c_p \rho_1}{S_1} (P - E) \tag{2.2.2.4}$$

$$B_s = \frac{\alpha g}{c_p \rho_1^2} Q_s \tag{2.2.2.5}$$

g' is the reduced gravity between the mixed layer and the next layer below, Ri_{crit} is the critical Richardson number, B is the total buoyancy flux through the surface, Q the total heat flux, R the corresponding equivalent heat flux due to the fresh water flux, $P - E$ denotes precipitation minus evaporation, B_s the buoyancy flux due to the solar radiative heat flux Q_s, α, β are the analytically determined expansion coefficients with respect to temperature and salinity, τ_x the zonal wind stress and Ω_y the y-component the earth's angular velocity. The variables m_0, m_1, m_2, m_3, m_4, m_5, and m_6 are tuning coefficients. The (favorite) parameter choice of JMO is $m_3 = m_4 = m_6 = 0$ and $m_5 = 1$. The parameter g'_0 limits the entrainment rate to a finite value when $g' = 0$.

The entrainment equation (2.2.2.1) contains many parameters. These allow the user to set up 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: $CMIX0 = CMIX2 = CMIX3 = CMIX5 = CMIX7 = 0$, $CMIX9 = 1$, $CMIX6 = 1.E6$, $TURBEN = 0$ and $TURBREL = 0$. This parameter set reduces the entrainment equation to $wg'h = 2m_0 u_*^3 + hB$.

The entrainment/detrainment rate w is related to the transfer rates w_k^l in the equations (2.1.2.1) to (2.1.2.5) by

$$w = \begin{cases} w_1^{+1} & \text{if} \quad w > 0 \\ w_{+1}^1 & \text{if} \quad w < 0 \end{cases} \tag{2.2.2.6}$$

The sea ice model described subsequently has some influence on the relation for entrainment. In the presence of sea ice $\Gamma = 1$, otherwise $\Gamma = 0$. 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 by

$$B_I = (qQ_1 + (1-q)\,Q) \frac{g\,(S_1 - S_I)}{c_{pm} \rho_1 \sigma_{\theta 1}} \frac{\partial}{\partial S} \sigma_\theta \Big|_{\theta,\,\rho} \tag{2.2.2.7}$$

where S_I is the ice salinity and $\frac{\partial \sigma_\theta}{\partial S}$ is the coefficient of expansion in 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 the reduced gravity in the entrainment equation (2.2.2.1) is modified by:

$$g'_I = g\left(\frac{\sigma_{\theta 1+}-\sigma_{\theta 1}}{\sigma_{\theta 1}} + \left(\frac{\partial}{\partial S}\sigma_\theta\Big|_{\theta,\rho} + \frac{c_p(S_1 - S_I)}{c_{pm}\rho_1\sigma_{\theta 1}}\frac{\partial}{\partial S}\sigma_\theta\Big|_{\theta,\rho}\right)(\theta_{1+} - \theta_1)\right) \qquad (2.2.2.8)$$

Compared with the entrainment equation without sea ice, we consider additional mechanisms. Entrainment provides some heat flux into the mixed layer. This flux is exactly balanced by a heat flux due to melting or freezing ice which keeps the mixed layer temperature at the freezing point. Finally, this fresh water flux induces a buoyancy flux.

The first term on the left hand side of equation (2.2.2.1) 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 TKE production due to wind stirring (u_* denotes the friction velocity), and the following terms describe the influence of the surface buoyancy fluxes for the ice covered and ice free conditions, respectively. The last term, which is only nonzero for ice free cases, represents the influence of penetrating solar radiation on the total buoyancy flux (Denman and Miyake; 1973). The free parameter m_0 represents the efficiency of how turbulence available for mixing is produced by the mean wind stress. Following Paulson and Simpson (1977), a fraction of solar radiation penetrates through the ocean surface. h_B is the depth at which the penetrated radiation has decayed to e^{-1}. In the relation for the entrainment rate the weighting coefficients m_1, m_2 are defined as exponential decay functions:

$$m_1 = exp\left(-\frac{h_1 f}{\kappa u_*}\right) \qquad (2.2.2.9)$$

$$m_2 = \begin{cases} exp\left(-\dfrac{h_1 f}{\kappa u_*}\right) & \text{if} \quad B < 0 \\ exp\left(-\dfrac{h_1 f}{\mu u_*}\right) & \text{if} \quad B > 0 \end{cases} \qquad (2.2.2.10)$$

where κ is the von Karman constant and μ a further tuning coeffiicient.

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 where 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:

$$m_1 = exp\left(-\frac{h_1}{h_{TKE}}\right) \tag{2.2.2.11}$$

$$m_2 = \begin{cases} exp\left(-\frac{h_1}{h_{TKE}}\right) & \text{if} \quad B<0 \\ exp\left(-\frac{h_1}{h_{BUO}}\right) & \text{if} \quad B>0 \end{cases} \tag{2.2.2.12}$$

where h_{TKE} and h_{BUO} are the dissipation length scales for negative and positive buoyancy forcing:

In the retreat phase of the mixed layer the depth is determined by setting $w = 0$ in equation (2.2.2.1) and solving for h. The resulting equation for the Monin-Obukhov length h_M in the case of no sea ice is:

$$2m_0 u_*^3 + h_M (B - \gamma B_s) + \gamma\left(B_s h_M\left(1 + exp\left(-\frac{h_M}{h_B}\right)\right)\right) - 2h_B\left(1 - exp\left(-\frac{h_M}{h_B}\right)\right) = 0 \tag{2.2.2.13}$$

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:

$$h_{Ri} = \frac{Ri_{crit}(\Delta u^2 + \Delta v^2)}{g'} \tag{2.2.2.14}$$

As a result there are two constraints that limit the MLD. First the MLD cannot be deeper than the Monin-Obukhov-length and secondly it cannot be shallower than h_{Ri}. If the two constraints contradict each other, h_{Ri} is taken as criterion. This means that as long as the MLD is smaller than h_M and larger than h_{Ri} the MLD is not altered by these diagnostic calculations, but as soon as the MLD becomes smaller than h_{Ri} or larger 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 by:

$$w = \frac{h_M^{n+1} - h_1^n}{\Delta t} \tag{2.2.2.15}$$

and is used to force the continuity equation, while in the first corrector step it is ensured that $h_1^{n+1} = h_M^{n+1}$.

2.2.3 Coupling of the Mixed Layer to the Interior Ocean

Entrainment is a process that does not act in 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 mass may be pumped. The strategy is described in JMO and can be found in subroutines <*MIXEXP*>, <*MIXIMP*> and <*CONVECT*>.

2.3 PARAMETERIZATION OF 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 and surface salinity. Except for the annual mean surface salinity the data are monthly mean climatological values. The flux of momentum is taken from Hellerman 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 on a $2^o \times 2^o$ grid, which is sufficiently fine for forcing an OGCM. These fields represent a 30-year climatology from 1950 to 1979 and have been extended to 1986. Alternatively, wind, wind stress and air temperature data are available on a $2.5^o \times 2.5^o$ grid from Trenberth et al. (1989) and also on a 1.1^o grid (directly derived from the ECMWF-analyses on the T106-grid).

2.3.1 Parameterization of the Surface Heat Fluxes

The total heat flux into the mixed layer is given by

$$Q_1 = Q_H + Q_L + Q_l + Q_s\left(1 - \gamma exp\left(-\frac{h_1}{h_B}\right)\right) \qquad (2.3.1.1)$$

and consists of the sensible heat flux Q_H, the latent heat flux Q_L, the net heat flux by long wave radiation Q_l and the heat flux Q_s 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:

$$Q_k = Q_s\left(exp\left(-\sum_{l=1}^{k-1}\frac{h_1}{h_B}\right) - exp\left(-\sum_{l=1}^{k}\frac{h_1}{h_B}\right)\right)\gamma \qquad (2.3.1.2)$$

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

$$Q_H = \rho_\alpha c_{p,\,air} c_H V (T_a - T_s) \tag{2.3.1.3}$$

$$Q_L = \rho_\alpha L_w c_L V (q_a - q_s) \tag{2.3.1.4}$$

where T_a is the air temperature, T_s the sea surface temperature, q_a the air specific humidity and q_s the specific humidity close to the surface, which is assumed to be the saturated value. $c_{p,\,air}$ is the specific heat of air and L_W the latent heat of evaporation. The specific humidity q is given by the water vapor pressure e and the atmospheric surface pressure p . In the following relations, (2.3.1.5) - (2.3.1.11), e and p are given in Pascal and T in degrees Kelvin.

$$e = 611 \times 10^{7.5\,((T - 273.16)/(T - 35.86))} \tag{2.3.1.5}$$

$$q = \frac{0.622 e}{p - 0.378 e} \tag{2.3.1.6}$$

From these relations $q_s = q(p, e(T_s))$ and $q_a = q(p, r \times e(T_a))$ are obtained, where r is the relative humidity. The bulk coefficients c_H and c_L are calculated as proposed by Large and Pond (1982). The respective formulae are given in the next section.

The net long wave radiation at the surface is taken from Berliand and Berliand (1952)

$$Q_l = 4\varepsilon\sigma T_a^3 (T_a - T_s) - \varepsilon\sigma T_a^4 (0.39 - 0.05\sqrt{\frac{e}{100}})\,(1 - \chi n^2) \tag{2.3.1.7}$$

where ε is the emissivity of water, σ the Stefan-Boltzmann constant and n the relative cloud cover. In order to account for differing properties of different cloud types, χ varies linearly with latitude, as proposed by Budyko (1974).

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. Following Reed (1977), the insolation is reduced by a cloudiness factor. The resulting relations to compute the daily mean downward short wave radiative flux Q_s are:

$$\cos\eta = \sin\delta\sin\varphi + \cos\delta\cos\varphi\cos t \tag{2.3.1.8}$$

$$\sin\eta_{noon} = \sin\delta\sin\varphi + \cos\delta\cos\varphi \tag{2.3.1.9}$$

$$\kappa = 1 - 0.62\eta + 0.0019\eta_{noon} \tag{2.3.1.10}$$

$$Q_s = \frac{\kappa(1 - \omega)}{2\pi} \left(\int_{t_1}^{t_2} \frac{S_o (\cos\eta)^2}{(\cos\eta + 2.7)\,(\frac{re(T_a)}{p}) + 1.085\cos\eta + 0.1} \right) (\frac{\bar{d}}{d})^2 dt \tag{2.3.1.11}$$

Here S_o is the solar constant, η the solar elevation 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 $(\frac{\bar{d}}{d})^2$ is estimated in terms of the Julian Day β for the present day orbit:

$$(\frac{\bar{d}}{d})^2 = 1.00011 + 0.00128\sin\beta + 0.034221\cos\beta + 0.000077\sin 2\beta + 0.000719\cos 2\beta \qquad (2.3.1.12)$$

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

$$\delta = 0.006918 + 0.070257\sin\beta - 0.399912\cos\beta + 0.000907\sin 2\beta - 0.006758\cos 2\beta + 0.00148\sin 3\beta - 0.002697\cos 3\beta \qquad (2.3.1.13)$$

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

2.3.2 Transfer Coefficients

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

$$c_H = \frac{c_{HN}\sqrt{\frac{c_M}{c_{MN}}}}{1 - \frac{c_{HN}}{\sqrt{c_{MN}}}\frac{\Psi_H}{\kappa}\frac{Z}{L}} \qquad (2.3.2.1)$$

$$c_L = \frac{c_{LN}\sqrt{\frac{c_M}{c_{MN}}}}{1 - \frac{c_{LN}}{\sqrt{c_{MN}}}\frac{\Psi_L}{\kappa}\frac{Z}{L}} \qquad (2.3.2.2)$$

$$\sqrt{\frac{c_M}{c_{MN}}} = \frac{1}{1 - \sqrt{c_{MN}}\frac{\Psi_M}{\kappa}\frac{Z}{L}} \qquad (2.3.2.3)$$

$$c_{MN} = \frac{\kappa^2}{\ln^2(\frac{Z}{Z_0})} \qquad (2.3.2.4)$$

$$c_{HN} = 0.0327\frac{\kappa}{\ln(\frac{Z}{Z_0})} \qquad (2.3.2.5)$$

$$c_{LN} = 0.0346 \frac{\kappa}{\ln\left(\frac{Z}{Z_0}\right)} \tag{2.3.2.6}$$

$$Z_0 = c_{char} \frac{u_*^2}{g} \tag{2.3.2.7}$$

$$u_*^2 = c_M u^2 \tag{2.3.2.8}$$

$$T_0 = T(1 + 1.7 \times 10^{-6} Tq) \tag{2.3.2.9}$$

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 we use:

$$\psi_M = \psi_H = \psi_L = -7\frac{Z}{L} \tag{2.3.2.10}$$

$$\frac{Z}{L} = -\frac{70Z}{u^2 T_0}(\Delta\theta + 2.5 \times 10^{-6} T_0^2 \Delta q) \tag{2.3.2.11}$$

while for unstable conditions we use:

$$\psi_M = 2\ln\left(\frac{1+X}{2}\right) + \ln\left(\frac{1+X^2}{2}\right) - \left|2\operatorname{atan} X + \frac{\pi}{2}\right| \tag{2.3.2.12}$$

$$\psi_L = \psi_H = 2\ln\left(\frac{1+X^2}{2}\right) \tag{2.3.2.13}$$

$$X = \left(1 - 16\frac{Z}{L}\right)^{1/4} \tag{2.3.2.14}$$

$$\frac{Z}{L} = -\frac{100Z}{u^2 T_0}(\Delta\theta + 1.7 \times 10^{-6} T_0^2 \Delta q) \tag{2.3.2.15}$$

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 (2.3.2.4), (2.3.2.7) and (2.3.2.8) describe the dependence of the neutral drag coefficient c_{MN} on the friction velocity u_*, the von 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 ms^{-1}, 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.

2.3.3 Evaluation of the Net Fresh Water Flux

In OPYC there are two methods available to compute the fresh water fluxes. Depending on switch *ISW*18 either the Newtonian forcing (see JMO) or a parameterized fresh water flux can be chosen:

1. The Newtonian forcing itself can be used in two different modes. These are:

- **Instantaneous Newton Relaxation:** Since the observations of precipitation, evaporation and river runoff $(P-E+R)$ are not very reliable, it is common (see also JMO) to compute $(P-E+R)$ by a Newtonian formulation. The model surface salinity is relaxed towards the observed sea surface salinity. The relation used is:

$$R_1^h = \rho_1 \delta \frac{S_{obs} - S_1}{S_1} \tag{2.3.3.1}$$

where δ is the relaxation time scale of the model salinity towards the observed salinity.

- **Annual Mean Newton Relaxation:** Since monthly surface salinities are on the global scale not reliable, only annual mean values are used. However, the Newtonian formulation will damp the seasonal cycle of salinity computed by OPYC, for instance as result of melting or freezing of sea ice. The artificially damped seasonal cycle of the salinity in the upper ocean will affect the fluxes into the ML due to entrainment. Alternatively the ocean can be forced with only the annual mean $(P-E+R)$.

$$R_1^h = \rho_1 \delta \int_{t_1}^{t_2} \frac{S_{obs} - S_1}{S_1} \delta t \tag{2.3.3.2}$$

where $t_2 - t_1$ is a multiple of a year. This formula is implemented in the code, so that OPYC is able to find the annual mean $(P-E+R)$ during the run (see switch *ISW*48).

2. **Parameterized Fresh Water Flux:** The other possibility is to compute $(P-E+R)$ from data for precipitation as prepared by Legates and Willmott (1990), from the computed evaporation being derived from the latent heat release and from an annual mean correction of $(P-E+R)$, in the best case reflecting only river runoff. Since no river runoff data are available yet operationally, the river runoff is computed in this mode similar to equation (2.3.3.1).

2.3.4 Estimate of Turbulent Kinetic Energy Input

The monthly mean absolute wind $\bar{V}$ and its standard deviation $\sigma(V)$ are available from COADS by Wright (1988) or from ECMWF analyses. $\sigma(V)$ is required to evaluate accurately the time-averaged third power of the friction velocity u_*, which occurs in the relation for the entrainment rate. Since $\bar{u}_*^3$,

determined from the monthly mean absolute wind $\bar{u}$ only, is much smaller than the required $\bar{u}_*^3$, an effective u_*^3 must be determined by the additional use of the monthly mean standard deviation of the absolute wind. The effective u_*^3 can be approximated by assuming that the amplitude of the fluctuations is not too large compared with the mean wind and a sinusoidal fluctuation around its mean value (see JMO). A similar relation is required to compute u_*^2 for the neutral drag coefficient:

$$u_*^3 = \sqrt{\frac{c_d \rho_a}{\rho_1}}^3 \bar{V} (\bar{V}^2 + 3\sigma^2 (V)) \tag{2.3.4.1}$$

$$u_*^2 = \sqrt{\frac{c_d \rho_a}{\rho_1}}^2 \bar{V} (\bar{V} + 2\sigma (V)) \tag{2.3.4.2}$$

2.4 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. For further discussion of how the vertical mixing processes are connected with the problem of maintaining the coordinates in an isopycnal OGCM see JMO.

2.4.1 Vertical Mixing / Coordinate Maintenance

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

$$w_k^{k-*} = \frac{2 m_0 u_*^3}{g'_{k-} h_k - Ri_{crit} ((u_k - u_{k-})^2 + (v_k - v_{k-})^2)} (1 + W) \tag{2.4.1.1}$$

$$w_k^{k+*} = \frac{2 m_0 u_*^3}{g'_{k+} h_k - Ri_{crit} ((u_k - u_{k+})^2 + (v_k - v_{k+})^2)} (1 + W) \tag{2.4.1.2}$$

$$W = \frac{g'_{k-} (h_k + h_{k-}) + g'_{k+} (h_k + h_{k+})}{h_w (g'_{k-} + g'_{k+})} \tag{2.4.1.3}$$

Thereby, g'_{k-} and g'_{k+} are the stabilities across the upper and the 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 effectively enhances vertical mixing in the deep ocean, while mixing is unchanged in the upper ocean. The problem of vertical mixing is underdetermined, since one is free to choose the amount 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

$$w_k = (1-\alpha)\, w_k^{k-*} + \alpha w_k^{k+*} \tag{2.4.1.4}$$

with individual entrainment rates

$$w_k^{k-} = (1-\alpha)\, w_k^{k-*} \tag{2.4.1.5}$$

$$w_k^{k+} = \alpha w_k^{k+*} \tag{2.4.1.6}$$

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 of the layer at the prescribed value σ_θ^*, at the same time compensating any 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:

$$(h_k + \Delta t\,(w_k^{k-} + w_k^{k+}))\,\sigma_{\theta k}^* = h_k \sigma_{\theta k} + \Delta t w_k^{k-} \sigma_{\theta k-} + \Delta t w_k^{k+} \sigma_{\theta k+} \tag{2.4.1.7}$$

where the potential density at the new time level should 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. Together with the equations (2.4.1.5) and (2.4.1.6) this yields the final equation for α:

$$\alpha = \frac{h_k(\sigma_{\theta k}^* - \sigma_{\theta k}) + \Delta t w_k^{k-*}(\sigma_{\theta k}^* - \sigma_{\theta k-})}{\Delta t w_k^{k-*}(\sigma_{\theta k}^* - \sigma_{\theta k-}) - \Delta t w_k^{k+*}(\sigma_{\theta k}^* - \sigma_{\theta k+})} \tag{2.4.1.8}$$

2.4.2 Convection

If the stratification becomes unstable it is removed by vertical mixing. All quantities are set to their vertical average over the mixed layer ML and the underlying layer UL. The procedure describing how the coordinate system is reorganized if an isopycnal becomes to heavy is described in JMO. A well-known problem is the determination of stability, when potential densities are defined with respect to some reference level. Currently the surface is used. This leads to the effect, that for the same water masses the model still analyses a weakly stable stratification, while a potential density defined with respect to the considered depth yields an unstable stratification. For consistency, only potential densities with respect to the surface are used, e.g. for horizontal pressure gradients, as well as for convection and vertical mixing. A version of the code that is able to use an arbitrary reference level is in preparation.

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 the nonlinearity of the equation of state their

mean values are distributed over the two adjacent unstable stratified layers. Layer thicknesses are not altered.

2.5 THE SNOW - SEA ICE MODEL

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. 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.

2.5.1 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 $(\vec{v}h)_0$, of the ice thickness h_0, of the ice concentration q_0 and of the snow depth s_0 are (with the latter taken as equivalent ice thickness),

$$\frac{\partial}{\partial t}(\vec{v}h)_0 = \vec{\nabla}\cdot A_0\vec{\nabla}(\vec{v}h)_0 - \vec{f}\times(\vec{v}h)_0 + gh_0\vec{\nabla}\left(\sum_{k=1}^{N} h_k + D\right) + \frac{\vec{\tau}_a}{\rho_i} + \frac{\vec{\tau}_o}{\rho_i} + \vec{F}_v \qquad (2.5.1.1)$$

$$\frac{\partial h_0}{\partial t} = \vec{\nabla}\cdot A_0\vec{\nabla}h_0 - \vec{\nabla}\cdot(\vec{v}h)_0 + F_h + F_a \qquad (2.5.1.2)$$

$$\frac{\partial q_0}{\partial t} = \vec{\nabla}\cdot A_0\vec{\nabla}q_0 - \vec{\nabla}\cdot(\vec{v}q)_0 + F_q \qquad (2.5.1.3)$$

$$\frac{\partial s_0}{\partial t} = \vec{\nabla}\cdot A_0\vec{\nabla}s_0 - \vec{\nabla}\cdot(\vec{v}s)_0 + F_s - F_a \qquad (2.5.1.4)$$

where τ_a and τ_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_0 is a constant diffusion coefficient. 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, $\vec{\vec{F}}_v$ 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. Following Hibler's notation, the forcing functions are defined by:

$$F_{vx} = \frac{\partial}{\partial x}\left((\eta + \zeta)\frac{\partial u}{\partial x} + (\zeta - \eta)\frac{\partial v}{\partial y} - \frac{P_0}{2}\right) + \frac{\partial}{\partial y}\left(\eta\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right) \tag{2.5.1.5}$$

$$F_{vy} = \frac{\partial}{\partial y}\left((\eta + \zeta)\frac{\partial v}{\partial y} + (\zeta - \eta)\frac{\partial u}{\partial x} - \frac{P_0}{2}\right) + \frac{\partial}{\partial x}\left(\eta\left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)\right) \tag{2.5.1.6}$$

$$\Delta = \left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right)\frac{1+e^2}{e^2} + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}\right)^2\frac{1}{e^2} + 2\frac{\partial u}{\partial x}\frac{\partial v}{\partial y}\frac{e^2-1}{e^2} \tag{2.5.1.7}$$

$$P_0 = \frac{P_i h_0}{\rho_i} exp\left(-\varepsilon_1 (1 - q_0)\right) \tag{2.5.1.8}$$

$$\zeta = \frac{P_0}{max\left(2\sqrt{\Delta}, \varepsilon_0\right)} \tag{2.5.1.9}$$

$$\eta = \frac{\zeta}{e^2} \tag{2.5.1.10}$$

P_i is a propotionally coefficient, e is the eccentricity 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 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 which cross-couple the velocity components.

2.5.2 The Thermodynamic Equations

The model is forced by atmospheric heat fluxes as well as fresh water fluxes given by precipitation minus 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 layer 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 resulting system

of equations is as follows:

$$\frac{k_i q_0}{h_0}(T_h - T_1) + \frac{k_s q_0 \rho_s}{s_0 \rho_i}(T_h - T_s) + c_{pi}\left(\frac{\rho_i h_0}{2}\frac{\partial T_h}{\partial t} + \frac{\rho_i s_0}{2}\frac{\partial}{\partial t}(T_h + T_s)\right) = 0 \qquad (2.5.2.1)$$

$$\frac{k_s q_0 \rho_s}{s_0 \rho_i}(T_h - T_s) + Q(T_s) = 0 \qquad (2.5.2.2)$$

Here, T_1 is the mixed layer temperature, k_s the conductivity of snow, k_i the conductivity of ice, $Q(T_s)$ the heat flux through the snow or at the ice surface, c_{pi} the specific heat capacity of ice and ρ_s is the density of snow. Thereby, h_0/q_0 is the effective thickness of an ice floe and $(\rho_i s_0)/(\rho_s q_0)$ is the effective snow depth. Note here that s_0 is a cell-averaged snow depth at the ice density. The equation for the heat accumulation (2.5.2.1) contains two time derivative terms. 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 (2.5.2.2) is used to modify (2.5.2.1) and the resulting equations are multiplied by the ice or snow mass. This finally yields:

$$k_i q_0 (T_h - T_1) + h_0 Q(T_s) + h_0 c_{pi}\left(\frac{\rho_i h_0}{2}\frac{\partial T_h}{\partial t} + \frac{\rho_i s_0}{2}\frac{\partial}{\partial t}(T_h + T_s)\right) = 0 \qquad (2.5.2.3)$$

$$\frac{k_s q_0 \rho_s}{\rho_i}(T_h - T_s) + s_0 Q(T_s) = 0 \qquad (2.5.2.4)$$

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 (2.5.2.1) to compute the ice surface temperature T_h. The cell-averaged skin temperature T_a is defined by:

$$T_a = T_s q_0 + T_1(1 - q_0) \qquad (2.5.2.5)$$

This yields the snow surface temperature T_s or the ice surface temperature T_h by simplifying the total heat flux Q from equation (2.3.1.1) through a Taylor expansion around T_a:

$$Q(T) = Q(T_a) + \frac{\partial Q}{\partial T}(T - T_a) \qquad (2.5.2.6)$$

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:

$$Q_s = Q(T_m) + \frac{k_s q_0 \rho_s}{s_0 \rho_i}(T_h - T_m) \tag{2.5.2.7}$$

$$Q_i = (1 - q_0) Q(T_1) + q_0 \frac{k_i q_0}{h_0}(T_h - T_1) \tag{2.5.2.8}$$

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 (2.5.2.2), however, this equation is not used when (2.5.2.3) and (2.5.2.4) yield $T_s > T_m$. In this case the error in (2.5.2.2) is interpreted as heat flux used for melting snow. The linearity of (2.5.2.1) and (2.5.2.2) ensures that this flux is always downward. The resulting change in the local ice thickness due to thermodynamics is then given by the heat flux Q_i and the heat flux induced by the entrainment rate *w*,

$$F_h = \frac{|w| + w}{2} \frac{c_p}{c_{pm}} (T_1 - T_{1+}) - \frac{Q_i}{c_{pm}} \tag{2.5.2.9}$$

where c_{pm} is the latent heat of fusion.

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

$$F_q = \begin{cases} \dfrac{F_h (1 - q_0)}{h_i} & \text{if} \quad F_h > 0 \\ \dfrac{F_h q_0}{2 h_0} & \text{if} \quad F_h > 0 \end{cases} \tag{2.5.2.10}$$

where h_i 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 crystals to ice by assuming a simple snow depth and time scale γ dependence. The second term adjusts the snow depth when snow suppresses the snow depth below the sea level. Thus, this mechanism represents an upper threshold for the snow depth at a given ice thickness.

$$F_a = \gamma s_0 + \frac{1}{\Delta t} \frac{\rho_s}{\rho_i} \max\left(0, \frac{h_s \rho_s - h_i (\rho_1 - \rho_i)}{\rho_1}\right) \tag{2.5.2.11}$$

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

$$F_s = q_0 R_1^{P-E} + \frac{q_0 Q_s}{c_{pm}} + \frac{s_0}{q_0} min\,(F_q, 0) \qquad (2.5.2.12)$$

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

$$T_f = T_m - 0.0575S + 0.001710523S^{3/2} - 0.0002154996S^2 \qquad (2.5.2.13)$$

2.5.3 Sea Ice - Ocean Salt Coupling

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 more deeply 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

$$R_k^S = \frac{|F_h| + F_h}{2} \rho_1 (S_1 - S_I) \left(exp\left(-\frac{\sigma_{\theta k} - \sigma_{\theta 1}}{h_p} \sum_{l=1}^{k-1} h_l \right) - exp\left(-\frac{\sigma_{\theta k+} - \sigma_{\theta 1}}{h_p} \sum_{l=1}^{k} h_l \right) \right) \qquad (2.5.3.1)$$

where h_p 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

$$R_k^S = \rho_1 (S_1 - S_I) \left(F_h - \frac{|F_h| + F_h}{2} exp\left(-\frac{\sigma_{\theta 1+} - \sigma_{\theta 1}}{h_p} h_1 \right) \right) \qquad (2.5.3.2)$$

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 vertical diffusion yields an equilibrium state for the salinity stratification in areas covered by sea ice.

3. MODEL NUMERICS

3.1 Computer Performance

This part of the manual describes a code that is optimized for a vector computers, but it can also be run on a small number of processors computing in parallel with shared memory, e.g. a CRAY-type computer. The present code is not written for a massive-parallel type of machine with distributed memory. The strategy to achieve a high Mflop-rate (currently: 165 Mflops on a CRAY-YMP) is the following:

- 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, this operation always runs 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.
- Parallelization works best if the long vectors are split up into parts which then are distributed over the available processors. Currently, a few of the central routines have been optimized by CRAY-Research. The code achieves a speedup of 3.2 on 4 processors or 5.4 on 8 processors with autotasking except <*SOLVER1*>, <*TRIBLCK*>, <*MODIMP*>, <*TRACER*>, and <*FLUXES*>, that are macrotasked.
- Equations that must be solved either iteratively or directly, are treated so, that direct solutions are computed for slices on the zx-plane and iterated in y-direction. This allows one to solve many linear equations in parallel. This feature is used for vectorization and parallelization.

3.2 Discretization in Space on the Arakawa B-Grid

Our notation is that *K* is the 1st index, *I* the 2nd and *J* the 3rd. A grid point with the indizes (1,1,1) is located in the top layer, the mixed layer, and in the south-west corner of the model grid. The Arakawa B-grid distinguishes between scalar and vector points, as shown in Figure 1.

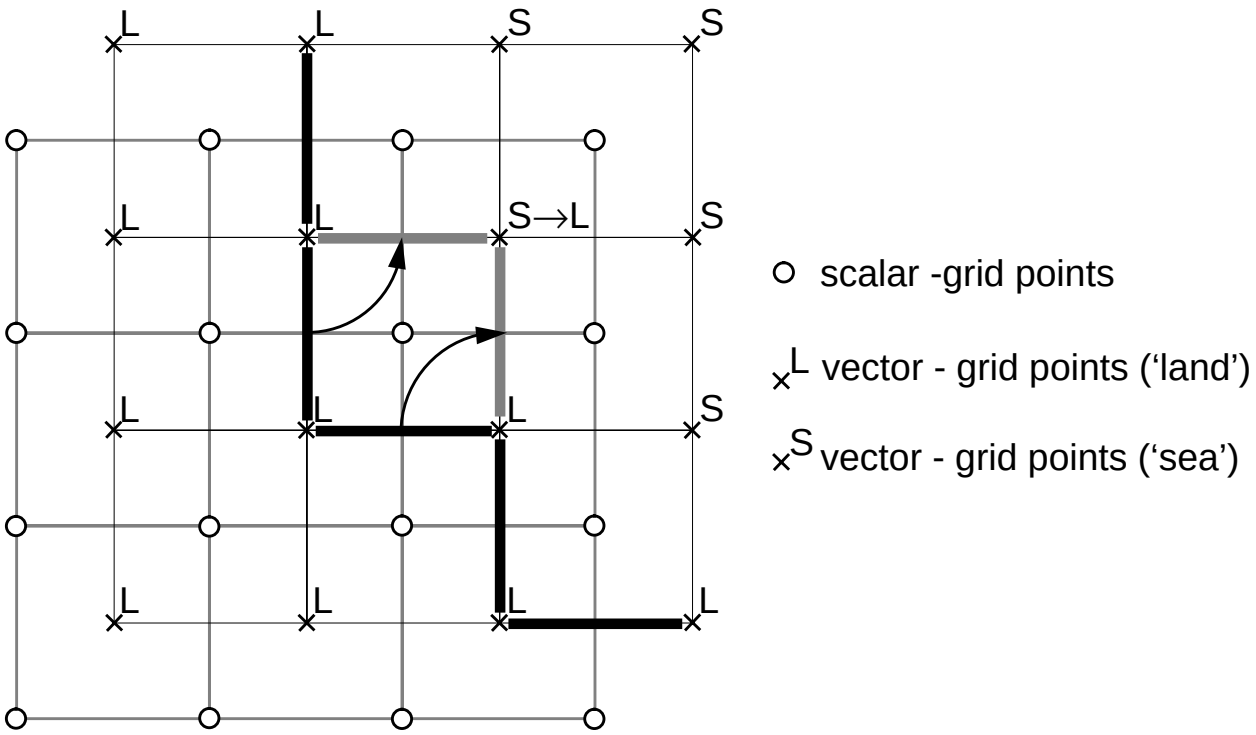

Figure 1 The Arakawa B-Grid.

Schematic layout of the Arakawa 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 a to land point at the edge of a zero-layer.

In the subsequent formulae scalar quantities have a subscript $S(K, I, J)$, while a vector point is 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 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:

$$P_{V(I, J)} = \frac{P_{S(I, J)} + P_{S(I+1, J)} + P_{S(I, J+1)} + P_{S(I+1, J+1)}}{4} \tag{3.2.1}$$

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

$$P_{S(I, J)} = \frac{P_{V(I, J)} + P_{V(I-1, J)} + P_{V(I, J-1)} + P_{V(I-1, J-1)}}{4} \tag{3.2.2}$$

The model coordinates ($X(NX)$ for the x-direction and $Y(NY)$ for the y-direction) are defined on vector points. The array X contains the x-values on the equator. The local grid distance on the sphere is obtained by multiplying with $\cos(\varphi)$ i.e. $SCALEX(NY)$ 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:

$$\Delta x_{S(I,J)} = \cos(\varphi)_{S(J)} \left(X_{V(I)} - X_{V(I-1)}\right) \tag{3.2.3}$$

while Δx on a vector point is:

$$\Delta x_{V(I,J)} = \cos(\varphi)_{V(J)} \frac{X_{V(I+1)} - X_{V(I-1)}}{2} \tag{3.2.4}$$

Δy is obtained on scalar points by:

$$\Delta y_{S(I,J)} = Y_{V(J)} - Y_{V(J-1)} \tag{3.2.5}$$

and for Δy on vector points

$$\Delta y_{V(I,J)} = \frac{Y_{V(J+1)} - Y_{V(J-1)}}{2} \tag{3.2.6}$$

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.

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 section 3.2.5, the sea level pressure gradient according to section 3.2.3, the ice mass and compactness diffusion according to section 3.2.7 and the mass and compactness convergence according to section 3.2.4. 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 section 3.2.4 and the result is differentiated on vector points according to section 3.2.3. Note that the 9-point formulae do not diffuse B-grid computational waves which appear as a checkerboard patterrn. However, these spurious waves do not appear in the solutions anyway, because the formulae are consistent to the divergence terms and thus permit an explicit forward treatment of the mass and compactness convergence.

3.2.1 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 plane. The divergence operator $\vec{\nabla}\cdot$ is given by:

$$\vec{\nabla}\cdot \quad = \left(\frac{1}{r\cos\varphi}\frac{\partial}{\partial\lambda}, \frac{1}{r\cos\varphi}\frac{\partial}{\partial\varphi}\cos\varphi\right) \tag{3.2.1.1}$$

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

$$\vec{\nabla} = \left(\frac{1}{r\cos\varphi}\frac{\partial}{\partial\lambda}, \frac{1}{r}\frac{\partial}{\partial\varphi}\right) \tag{3.2.1.2}$$

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

$$\vec{\nabla}\cdot\vec{\nabla} = \frac{1}{r^2\cos^2(\varphi)}\frac{\partial}{\partial\lambda^2}^2 + \frac{1}{r^2\cos(\varphi)}\frac{\partial}{\partial\varphi}\left(\cos(\varphi)\frac{\partial}{\partial\varphi}\right) \tag{3.2.1.3}$$

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

$$\vec{\nabla}\cdot\vec{\nabla}u = \frac{1}{r^2\cos^2(\varphi)}\frac{\partial}{\partial\lambda^2}^2 u + \frac{1}{r^2\cos(\varphi)}\frac{\partial}{\partial\varphi}\left(\cos(\varphi)\frac{\partial u}{\partial\varphi}\right) - \frac{u}{r^2\cos^2(\varphi)} - \frac{2\sin(\varphi)}{r^2\cos^2(\varphi)}\frac{\partial v}{\partial\lambda} \tag{3.2.1.4}$$

$$\vec{\nabla}\cdot\vec{\nabla}v = \frac{1}{r^2\cos^2(\varphi)}\frac{\partial}{\partial\lambda^2}^2 v + \frac{1}{r^2\cos(\varphi)}\frac{\partial}{\partial\varphi}\left(\cos(\varphi)\frac{\partial v}{\partial\varphi}\right) - \frac{v}{r^2\cos^2(\varphi)} + \frac{2\sin(\varphi)}{r^2\cos^2(\varphi)}\frac{\partial u}{\partial\lambda} \tag{3.2.1.5}$$

3.2.2 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 nonzero 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, for all deeper layers it is computed through the algorithm that sets the correct boundary conditions near vanishing or reappearing layers. *IFLG* contains 0 for land and 1 for the 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 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 has to decouple a zero layer from the ocean. Furthermore, physical processes that change the location where an isopycnal disappears / reappears or that create water masses with a not yet existing potential density must have their counterpart in a technique which allows the shifting of boundaries or the flooding of zero layers. This is explained in JMO.

Simplified, 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.

3.2.3 The Pressure Gradient

$$\frac{\partial}{\partial x} h_{V(I,J)} = \frac{(h_{S(I+1,J)} + h_{S(I+1,J+1)} - h_{S(I,J)} - h_{S(I,J+1)})}{2\Delta x_{V(I,J)}} \tag{3.2.3.1}$$

$$\frac{\partial}{\partial y} h_{V(I,J)} = \frac{h_{S(I,J+1)} + h_{S(I+1,J+1)} - h_{S(I,J)} - h_{S(I+1,J)}}{2\Delta y_{V(I,J)}} \tag{3.2.3.2}$$

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.

3.2.4 The Flux Divergence

$$\frac{\partial}{\partial x} (\rho u h)_{S(I,J)} = \frac{(\rho u h)_{V(I-1,J)} + (\rho u h)_{V(I,J)} - (\rho u h)_{V(I-1,J-1)} - (\rho u h)_{V(I-1,J)}}{\Delta x_{S(I,J)}} \tag{3.2.4.1}$$

$$\begin{aligned} \frac{\partial}{\partial y} (\rho v h)_{S(I,J)} = [& \cos(\varphi)_{V(J)} ((\rho v h)_{V(I-1,J)} + (\rho v h)_{V(I,J)}) \\ & - \cos(\varphi)_{V(J-1)} ((\rho v h)_{V(I-1,J-1)} + (\rho v h)_{V(I,J-1)})] \\ & / ((\cos(\varphi)_{V(J-1)} + \cos(\varphi)_{V(J)}) \Delta y_{S(I,J)}) \end{aligned} \tag{3.2.4.2}$$

3.2.5 Horizontal Diffusion of the Momentum

$$\frac{\partial}{\partial x}\left(A\frac{\partial}{\partial x}(\rho uh)_{V(I,J)}\right) = \frac{(A_{S(I,J+1)}+A_{S(I+1,J+1)})}{2}\frac{(\rho uh)_{V(I+1,J)}-(\rho uh)_{V(I,J)}}{\Delta x_{S(I+1)}\Delta x_{V(I)}} - \frac{(A_{S(I,J)}+A_{S(I+1,J)})}{2}\frac{(\rho uh)_{V(I,J)}-(\rho uh)_{V(I-1,J)}}{\Delta x_{S(I)}\Delta x_{V(I)}} \tag{3.2.5.1}$$

$$\frac{\partial}{\partial y}\left(A\frac{\partial}{\partial y}(\rho uh)_{V(I,J)}\right) = \frac{(A_{S(I,J+1)}+A_{S(I+1,J+1)})\cos(\varphi)_{S(J+1)}\left((\rho uh)_{V(I,J+1)}-(\rho uh)_{V(I,J)}\right)}{\Delta y_{S(J+1)}\Delta y_{V(I)}\left(\cos(\varphi)_{S(J)}+\cos(\varphi)_{S(J+1)}\right)} - \frac{\left((A_{S(I,J)}+A_{S(I+1,J)})\cos(\varphi)_{S(J)}\right)\left((\rho uh)_{V(I,J)}-(\rho uh)_{V(I,J-1)}\right)}{\Delta y_{S(J)}\Delta y_{V(J)}\left(\cos(\varphi)_{S(J)}+\cos(\varphi)_{S(J+1)}\right)} \tag{3.2.5.2}$$

The same discretization is used for (ρvh).

3.2.6 Horizontal Advection of the Momentum

Depending on the switch *ISW*32 in /*JMOFLAG*/ one of the following advection schemes is chosen:

3.2.6.1 The Upstream Scheme

$$\frac{\partial}{\partial x}\left(u(\rho uh)_{V(I,J)}\right) = \min\left(0, u_{V(I,J)}+u_{V(I+1,J)}\right)\frac{(\rho uh)_{V(I+1,J)}-(\rho uh)_{V(I,J)}}{2\Delta x_{S(I+1)}} + \max\left(0, u_{V(I-1,J)}+u_{V(I,J)}\right)\frac{(\rho uh)_{V(I,J)}-(\rho uh)_{V(I-1,J)}}{2\Delta x_{S(I)}} \tag{3.2.6.1.1}$$

$$\begin{aligned}\frac{\partial}{\partial x}\left(v(\rho uh)_{V(I,J)}\right) = {} & \min\left(0, u_{V(I,J)}+u_{V(I+1,J)}\right)\frac{(\rho uh)_{V(I+1,J)}-(\rho uh)_{V(I,J)}}{2\Delta x_{S(I+1)}} \\ & + \max\left(0, u_{V(I-1,J)}+u_{V(I,J)}\right)\frac{(\rho uh)_{V(I,J)}-(\rho uh)_{V(I-1,J)}}{2\Delta x_{S(I)}} \\ & + \min\left(0, (\rho uh)_{V(I,J)}+(\rho uh)_{V(I+1,J)}\right)\frac{u_{V(I+1,J)}-u_{V(I,J)}}{2\Delta x_{S(I+1)}} \\ & + \max\left(0, (\rho uh)_{V(I-1,J)}+(\rho uh)_{V(I,J)}\right)\frac{u_{V(I,J)}-u_{V(I-1,J)}}{2\Delta x_{S(I)}}\end{aligned} \tag{3.2.6.1.2}$$

3.2.6.2 The Crowley Scheme

Alternately, a scheme by Crowley (1968) can be used. However, this is rather diffusive at high CFL-numbers as is the upstream scheme in general:

$$\begin{aligned}\frac{\partial}{\partial x}(u\,(\rho uh)_{V(I,J)}) &= (u_{V(I,J)}+u_{V(I+1,J)})\,(1-(u_{V(I,J)}+u_{V(I+1,J)})\,\frac{\Delta t}{2\Delta x_{S(I+1)}})\\ &\quad *\frac{(\rho uh)_{V(I+1,J)}-(\rho uh)_{V(I,J)}}{2\Delta x_{S(I+1)}}\\ &\quad +(u_{V(I-1,J)}+u_{V(I,J)})\,(1-(u_{V(I-1,J)}+u_{V(I,J)})\,\frac{\Delta t}{2\Delta x_{S(I)}})\\ &\quad *\frac{(\rho uh)_{V(I,J)}-(\rho uh)_{V(I-1,J)}}{2\Delta x_{S(I)}}\end{aligned} \tag{3.2.6.2.1}$$

$$\begin{aligned}\frac{\partial}{\partial x}(v\,(\rho uh)_{V(I,J)}) &= (v_{V(I,J)}+v_{V(I,J+1)})\,(1-(v_{V(I,J)}+v_{V(I,J+1)})\,\frac{\Delta t}{2\Delta y_{S(J+1)}})\\ &\quad *\frac{\cos(\varphi)_{V(J+1)}\,(\rho uh)_{V(I,J+1)}-\cos(\varphi)_{V(J)}\,(\rho uh)_{V(I,J)}}{2\Delta y_{S(J+1)}\,(\cos(\varphi)_{V(J+1)}+\cos(\varphi)_{V(J)})}\\ &\quad +(v_{V(I,J-1)}+v_{V(I,J)})\,(1-(v_{V(I,J-1)}+v_{V(I,J)})\,\frac{\Delta t}{2\Delta y_{S(J)}})\\ &\quad *\frac{\cos(\varphi)_{V(J)}\,(\rho uh)_{V(I,J)}-\cos(\varphi)_{V(J-1)}\,(\rho uh)_{V(I,J-1)}}{2\Delta y_{S(J)}\,(\cos(\varphi)_{V(J+1)}+\cos(\varphi)_{V(J)})}\end{aligned} \tag{3.2.6.2.2}$$

3.2.6.3 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 these terms on the right side of the momentum equation (2.1.2.1) can be rewritten as, for the x-component:

$$\begin{aligned}&-\frac{\partial}{\partial x}(uu\rho h)-\frac{\partial}{\partial y}(vu\rho h)+fv\rho h+v\rho h\frac{u\tan\varphi}{r}=\\ &-\frac{\rho h}{2}\left(\frac{\partial u^2}{\partial x}+\frac{\partial v^2}{\partial x}\right)-u\left(\frac{\partial}{\partial x}u\rho h+\frac{\partial}{\partial y}v\rho h\right)+v\rho h\frac{u\tan\varphi}{r}+(f+\zeta)\,(v\rho h)\end{aligned} \tag{3.2.6.3.1}$$

and for the y-component:

$$-\frac{\partial}{\partial x}(uv\rho h) - \frac{\partial}{\partial y}(vv\rho h) - fu\rho h - u\rho h\frac{u\tan\varphi}{r} =$$
$$-\frac{\rho h}{2}\left(\frac{\partial u^2}{\partial y} + \frac{\partial v^2}{\partial y}\right) + \left(-v\left(\frac{\partial}{\partial x}u\rho h + \frac{\partial}{\partial y}v\rho h\right)\right) - u\rho h\frac{u\tan\varphi}{r} - (f+\zeta)\,u\rho h \tag{3.2.6.3.2}$$

where ζ is the relative vorticity. The right hand sides consist firstly of terms representing the energy gradient, momentum convergence and curvature and secondly 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 equation (2.5.2.13) which is the predictor step. The residual terms are treated implicitly in the first corrector step which also contains the momentum diffusion terms. Since the momentum components are cross-coupled in the equations, a linear equation in x is formulated simultaneously for both velocity components. The discretizations for the flux divergence are those in section 3.2.4, the energy gradients for *u* are approximated by (the same formulae are valid for *v*) :

$$\frac{\rho h}{2}\frac{\partial u^2}{\partial x} = \frac{(\rho uh)_{V(I,J)} + (\rho uh)_{V(I+1,J)}}{2}\frac{u_{V(I+1,J)} - u_{V(I,J)}}{\Delta x_{S(I+1)}}$$
$$+ \frac{(\rho uh)_{V(I-1,J)} + (\rho uh)_{V(I,J)}}{2}\frac{u_{V(I,J)} - u_{V(I-1,J)}}{\Delta x_{S(I)}} \tag{3.2.6.3.3}$$

$$\frac{\rho h}{2}\frac{\partial u^2}{\partial y} = \left((\rho uh)_{V(I,J+1)} + (\rho uh)_{V(I,J)}\right)\frac{\cos(\varphi)_{V(J+1)}u_{V(I,J+1)} - \cos(\varphi)_{V(J)}u_{V(I,J)}}{\left(\cos(\varphi)_{V(J+1)} + \cos(\varphi)_{V(J)}\right)\Delta y_{S(J+1)}}$$
$$+ \left((\rho uh)_{V(I,J)} + (\rho uh)_{V(I,J-1)}\right)\frac{\cos(\varphi)_{V(J)}u_{V(I,J)} - \cos(\varphi)_{V(J-1)}u_{V(I,J-1)}}{\left(\cos(\varphi)_{V(J-1)} + \cos(\varphi)_{V(J)}\right)\Delta y_{S(J)}} \tag{3.2.6.3.4}$$

The local vorticity ζ is discretized by:

$$\zeta_{V(I,J)} = \frac{v_{S(I,J-1)} + v_{S(I,J)} - v_{S(I-1,J-1)} - v_{S(I-1,J)}}{2\Delta x_{V(I)}}$$
$$\frac{u_{S(I-1,J)} + u_{S(I,J)} - u_{S(I-1,J-1)} - u_{S(I,J-1)}}{2\Delta y_{V(J)}} \tag{3.2.6.3.5}$$

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

3.2.7 Horizontal Diffusion of Scalar Variables

$$\frac{\partial}{\partial x}\left(A\frac{\partial}{\partial x}\theta_{S(I,J)}\right) = \left(A_{S(I,J)} + A_{S(I+1,J)}\right)\frac{\theta_{S(I+1,J)} - \theta_{S(I,J)}}{2\Delta x_{V(I)}\Delta x_{S(I)}} + \left(A_{S(I-1,J)} + A_{S(I,J)}\right)\frac{\theta_{S(I,J)} - \theta_{S(I-1,J)}}{2\Delta x_{V(I-1)}\Delta x_{S(I)}} \tag{3.2.7.1}$$

$$\frac{\partial}{\partial y}\left(A\frac{\partial}{\partial y}\theta_{S(I,J)}\right) = \frac{\left(A_{S(I,J+1)} + A_{S(I,J)}\right)\left(\theta_{S(I,J+1)} - \theta_{S(I,J)}\right)\cos(\varphi)_{V(J)}}{\Delta y_{S(J)}\Delta y_{V(J)}\left(\cos(\varphi)_{V(J-1)} + \cos(\varphi)_{V(J)}\right)} + \frac{\left(A_{S(I,J)} + A_{S(I,J-1)}\right)\left(\theta_{S(I,J)} - \theta_{S(I,J-1)}\right)\cos(\varphi)_{V(J-1)}}{\Delta y_{S(J)}\Delta y_{V(J-1)}\left(\cos(\varphi)_{V(J-1)} + \cos(\varphi)_{V(J)}\right)} \tag{3.2.7.2}$$

The same discretization is used for *S* and *C*.

3.2.8 Horizontal Advection of Scalar Variables

The switch *ISW*32 in /*JMOFLAG*/ is used to choose one of the following advection schemes:

3.2.8.1 The Upstream Scheme

$$\frac{\partial}{\partial x}\left(\theta\rho uh\right)_{S(I,J)} = \min\left(0, \left(\rho uh\right)_{V(I,J-1)} + \left(\rho uh\right)_{V(I,J)}\right)\frac{\theta_{S(I+1,J)} - \theta_{S(I,J)}}{2\Delta x_{V(J)}} + \max\left(0, \left(\rho uh\right)_{V(I-1,J-1)} + \left(\rho uh\right)_{V(I-1,J)}\right)\frac{\theta_{S(I,J)} - \theta_{S(I-1,J)}}{2\Delta x_{V(J-1)}} \tag{3.2.8.1.1}$$

3.2.8.2 The Crowley Scheme

Alternately, an advection scheme similar to that employed by Smolarkiewicz (1982) is used, namely an implicit version of the scheme presented by Crowley (1968). The scheme is less diffusive than the upstream scheme and avoids 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 artifical 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.

$$\frac{\partial}{\partial x}(\theta\rho uh)_{S(I,J)} = ((\rho uh)_{V(I,J)} + (\rho uh)_{V(I,J-1)})\frac{\theta_{S(I+1,J)} - \theta_{S(I,J)}}{2\Delta x_{S(J)}}$$
$$*\min(0, 1 - (u_{V(I,J-1)} + u_{V(I,J)})\frac{\Delta t}{2(\Delta x_{V(J)})})$$
$$+ ((\rho uh)_{V(I,J-1)} + (\rho uh)_{V(I,J)})\frac{\theta_{S(I,J)} - \theta_{S(I-1,J)}}{2\Delta x_{S(I-1)}} \quad (3.2.8.2.1)$$
$$*\max(0, 1 - (u_{V(I,J-1)} + u_{V(I,J)}))\frac{\Delta t}{2\Delta x_{V(J-1)}}$$

$$\frac{\partial}{\partial y}(\theta\rho vh)_{S(I,J)} = ((\rho uh)_{V(I-1,J)} + (\rho uh)_{V(I,J)})$$
$$*\min(0, 1 - (u_{V(I-1)} + u_{V(I,J)})\frac{\Delta t}{2\Delta y_{V(J)}})\frac{(\theta_{S(I,J+1)} - \theta_{S(I,J)})\cos(\varphi)_{V(J)}}{2\Delta y_{V(J)}(\cos(\varphi)_{V(J-1)} + \cos(\varphi)_{V(J)})} \quad (3.2.8.2.2)$$
$$+ ((\rho uh)_{V(I-1,J-1)} + (\rho uh)_{V(I,J-1)})$$
$$*\max(0, 1 - (u_{V(I-1,J-1)} + u_{V(I,J-1)})\frac{\Delta t}{\Delta y_{V(J-1)}})\frac{(\theta_{S(I,J)} - \theta_{S(I,J-1)})\cos(\varphi)_{V(J-1)}}{\Delta y_{V(J-1)}(\cos(\varphi)_{V(J-1)} + \cos(\varphi)_{V(J)})}$$

3.3 Discretization in Time

3.3.1 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 may be written symbolically:

$$\vec{\Psi}^{*n+1} - \frac{\Delta t}{2}\vec{G}_{\Psi}^{*n+1} = \vec{\Psi}^{n} + \frac{\Delta t}{2}\vec{G}_{\Psi}^{*n} + \Delta t\vec{F}_{\Psi}^{*n} \quad (3.3.1.1)$$

$$\Phi^{*n+1} - \frac{\Delta t}{2}G_{\Phi}^{*n+1} = \Phi^{n} + \frac{\Delta t}{2}G_{\Phi}^{*n} + \Delta t F_{\Phi}^{*n} \quad (3.3.1.2)$$

$$\Theta^{*n+1} - \frac{\Delta t}{2}G_{\Theta}^{*n+1} = \Theta^{n} + \frac{\Delta t}{2}G_{\Theta}^{*n} + \Delta t F_{\Theta}^{*n} \quad (3.3.1.3)$$

$$\Pi^{*n+1} - \frac{\Delta t}{2}G_{\Pi}^{*n+1} = \Pi^{n} + \frac{\Delta t}{2}G_{\Pi}^{*n} + \Delta t F_{\Pi}^{*n} \quad (3.3.1.4)$$

Here Ψ represents the mass flux, Φ the mass content, Θ the heat content and Π 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. $\vec{\Psi}^{*n+1}$, Φ^{*n+1}, Θ^{*n+1} and Π^{*n+1} are first guesses for the new time level n+1.

In order to demonstrate how the technique works we explain here in detail how to derive the wave equation. In the first step, the momentum equation is discretized in time by using a centered Euler scheme. The x- and y-component of the momentum equation (2.1.2.1) then read:

$$(\rho uh)^{n+1} = (\rho uh)^{n} - \frac{\Delta t}{2} h^{n+1} \frac{\partial p^{n+1}}{\partial x} + \frac{\Delta t}{2} (f+\zeta) (\rho vh)^{n+1} + F^{n}_{\rho uh} \tag{3.3.1.5}$$

$$(\rho vh)^{n+1} = (\rho vh)^{n} - \frac{\Delta t}{2} h^{n+1} \frac{\partial p^{n+1}}{\partial y} - \frac{\Delta t}{2} (f+\zeta) (\rho uh)^{n+1} + F^{n}_{\rho vh} \tag{3.3.1.6}$$

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 $(\rho vh)^{n+1}$. This yields:

$$(\rho uh)^{n+1} = F^{*n}_{\rho uh} - \frac{\frac{\Delta t}{2} h^{n+1} \frac{\partial p^{n+1}}{\partial x} + \frac{\Delta t^2}{4} (f+\zeta) h^{n+1} \frac{\partial p^{n+1}}{\partial y}}{1 + \frac{\Delta t^2}{4} (f+\zeta)^2} \tag{3.3.1.7}$$

$$(\rho vh)^{n+1} = F^{*n}_{\rho vh} - \frac{\frac{\Delta t}{2} h^{n+1} \frac{\partial p^{n+1}}{\partial y} - \frac{\Delta t^2}{4} (f+\zeta) h^{n+1} \frac{\partial p^{n+1}}{\partial x}}{1 + \frac{\Delta t^2}{4} (f+\zeta)^2} \tag{3.3.1.8}$$

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

$$F^{*n}_{\rho uh} = \frac{(\rho uh)^{n} + F^{n}_{\rho uh} + \frac{\Delta t}{2} (f+\zeta) \left((\rho vh)^{n} + F^{n}_{\rho vh} \right)}{1 + \frac{\Delta t^2}{4} (f+\zeta)^2} \tag{3.3.1.9}$$

$$F^{*n}_{\rho vh} = \frac{(\rho vh)^{n} + F^{n}_{\rho vh} - \frac{\Delta t}{2} (f+\zeta) \left((\rho uh)^{n} + F^{n}_{\rho uh} \right)}{1 + \frac{\Delta t^2}{4} (f+\zeta)^2} \tag{3.3.1.10}$$

If the continuity equation is discretized in the same manner as the momentum equation we derive:

$$(\rho h)^{n+1} = (\rho h)^{n} - \frac{\Delta t}{2}\left(\frac{\partial}{\partial x}(\rho u h)^{n+1} + \frac{\partial}{\partial y}(\rho v h)^{n+1}\right) + F_{\rho h}^{n} \tag{3.3.1.11}$$

where $F_{\rho h}^{n}$ represents all the remaining terms taken at time level *n*. If the flux divergence now is eliminated by using equations (3.3.1.7) and (3.3.1.8) we obtain an equation for the layer thickness *h* only:

$$\begin{aligned}(h_k^{n+1} - h_k^n)\frac{\rho_k^n - \rho_k^{n+1}}{2} &= F_{\rho h}^{*n} - (\rho_k^{n+1} - \rho_k^n)\frac{h_k^n + h_k^{n+1}}{2} \\ &+ \frac{\Delta t^2}{4}\frac{\partial}{\partial x}\left(\frac{h_k^{n+1}}{1 + \frac{\Delta t^2}{4}(f+\zeta)^2}\frac{\partial}{\partial x}(p_k^{n+1})\right) + \frac{\Delta t^2}{4}\frac{\partial}{\partial y}\left(\frac{h_k^{n+1}}{1 + \frac{\Delta t^2}{4}(f+\zeta)^2}\frac{\partial}{\partial y}(p_k^{n+1})\right) \\ &+ \frac{\Delta t^2}{4}\frac{\partial}{\partial x}\left(\frac{h_k^{n+1}\frac{\Delta t}{2}(f+\zeta)}{1 + \frac{\Delta t^2}{4}(f+\zeta)^2}\frac{\partial}{\partial y}(p_k^{n+1})\right) - \frac{\Delta t^2}{4}\frac{\partial}{\partial y}\left(\frac{h_k^{n+1}\frac{\Delta t}{2}(f+\zeta)}{1 + \frac{\Delta t^2}{4}(f+\zeta)^2}\frac{\partial}{\partial x}(p_k^{n+1})\right)\end{aligned} \tag{3.3.1.12}$$

The following identity is used to seperate ρ and *h* from the product ρh.

$$(\rho h)^{n+1} - (\rho h)^{n} = (\rho^{n} + \rho^{n+1})\frac{h^{n+1} - h^{n}}{2} + (\rho^{n+1} - \rho^{n})\frac{h^{n+1} + h^{n}}{2} \tag{3.3.1.13}$$

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

$$F_{\rho h}^{*n} = F_{\rho h}^{n} - \frac{\Delta t}{2}\left(\frac{\partial F_{\rho u h}^{*n}}{\partial x} + \frac{\partial F_{\rho v h}^{*n}}{\partial y}\right) \tag{3.3.1.14}$$

Together with the equation of state (2.1.2.8) and the relation for the in situ pressure (2.1.2.9) and (2.1.2.11), equation (3.3.1.12) determines h_k^{n+1}. 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 equations (3.3.1.5) through (3.3.1.6).

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 directions and iterated in y, which is an application of the *line-relaxation* method. 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.

The above method is used if *ISW*36=0. However, a further method to solve the wave problem can be chosen if *ISW*36=1. In the latter the absolute vorticity terms are treated iteratively as part of the right side instead of eliminating them from the wave equation (see the derivation of the wave equation above). This method allows larger time steps and thus saves CPU time. However, a major disadvantage of this method is that Rossby waves converge more slowly than in the above described method. This becomes important for time steps much larger than the inertial period. On the other hand, if the time step is small due to fine grid resolution, the latter method has no disadvantage and therefore should be used for time steps smaller than half a day.

3.3.2 First Corrector Step

The entrainment and detrainment rate and the resulting changes in the mass fluxes are treated implicitly in the first corrector step. The solution for the mixed layer part 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:

$$\vec{\Psi}^{**n+1} - \frac{\Delta t}{2}\vec{G}_{\Psi}^{**n+1} = \vec{\Psi}^{*n+1} - \frac{\Delta t}{2}\vec{G}_{\Psi}^{**n} \qquad (3.3.2.1)$$

$$\vec{\Phi}^{**n+1} - \frac{\Delta t}{2}\vec{G}_{\Phi}^{**n+1} = \vec{\Phi}^{*n+1} - \frac{\Delta t}{2}\vec{G}_{\Phi}^{**n} \qquad (3.3.2.2)$$

$$\vec{\Theta}^{**n+1} - \frac{\Delta t}{2}\vec{G}_{\Theta}^{**n+1} = \vec{\Theta}^{*n+1} - \frac{\Delta t}{2}\vec{G}_{\Theta}^{**n} \qquad (3.3.2.3)$$

$$\Pi^{**n+1} - \frac{\Delta t}{2}\vec{G}_{\Pi}^{**n+1} = \Pi^{*n+1} - \frac{\Delta t}{2}\vec{G}_{\Pi}^{**n} \qquad (3.3.2.4)$$

$\vec{\Psi}^{**n+1}$, $\vec{\Phi}^{**n+1}$, $\vec{\Theta}^{**n+1}$ and $\vec{\Pi}^{**n+1}$ are the corrected guesses for the new time level $n+1$.

3.3.3 Second Corrector Step

Finally, the implicit part of the advection and diffusion of momentum is formulated by solving the resulting equation directly in x and iterating in y. The equation is:

$$\vec{\Psi}^{***n+1} - \frac{\Delta t}{2}\vec{G}_{\Psi}^{***n+1} = \vec{\Psi}^{**n+1} - \frac{\Delta t}{2}\vec{G}_{\Psi}^{***n} \qquad (3.3.3.1)$$

$\vec{\Psi}^{***n+1}$, $\vec{\Phi}^{**n+1}$, $\vec{\Theta}^{**n+1}$ and $\vec{\Pi}^{**n+1}$ are taken as final values for the new time level $n+1$. From these quantities the physical quantities u, v, h, ρ, θ, S, T and σ_θ are determined.

3.3.4 Discretization in Time of the Snow and Sea Ice Model

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 final step, the stress, the Coriolis terms and the sea ice rheology in the flux equation are treated implicitly. The coupling of the predictor and two corrector steps has already been outlined in the content of the ocean models. 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 Hibler's 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 (2.5.1.2) to (2.5.1.6) is sufficient to guarantee computational stability.

4. FLOW DIAGRAMS

4.1 GENERAL OVERVIEW OF FLOW

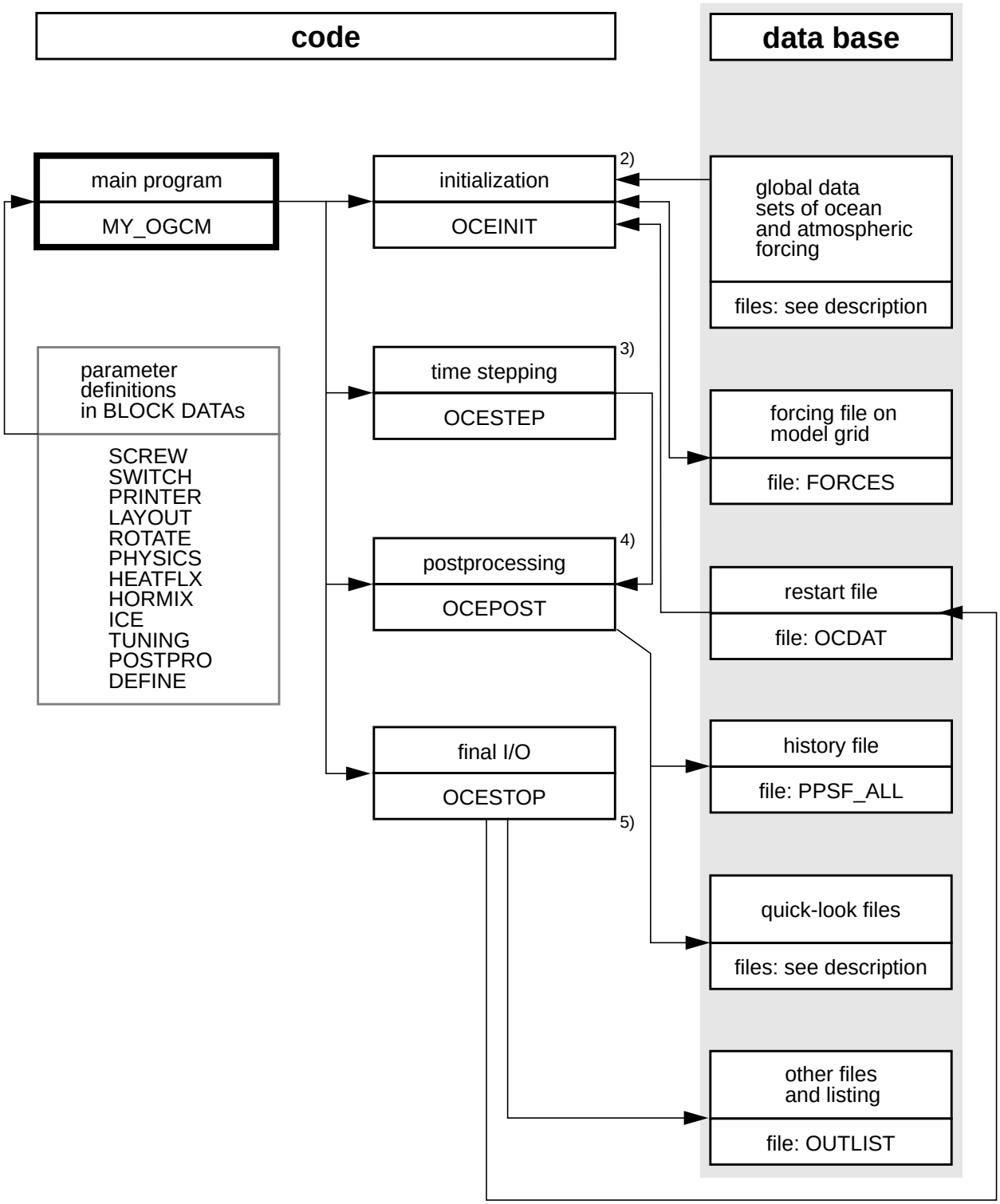

Figure 2 Main structure of the interaction between the code and the data base.

4.2 FLOW OF ROUTINE <OCEINIT> (2)

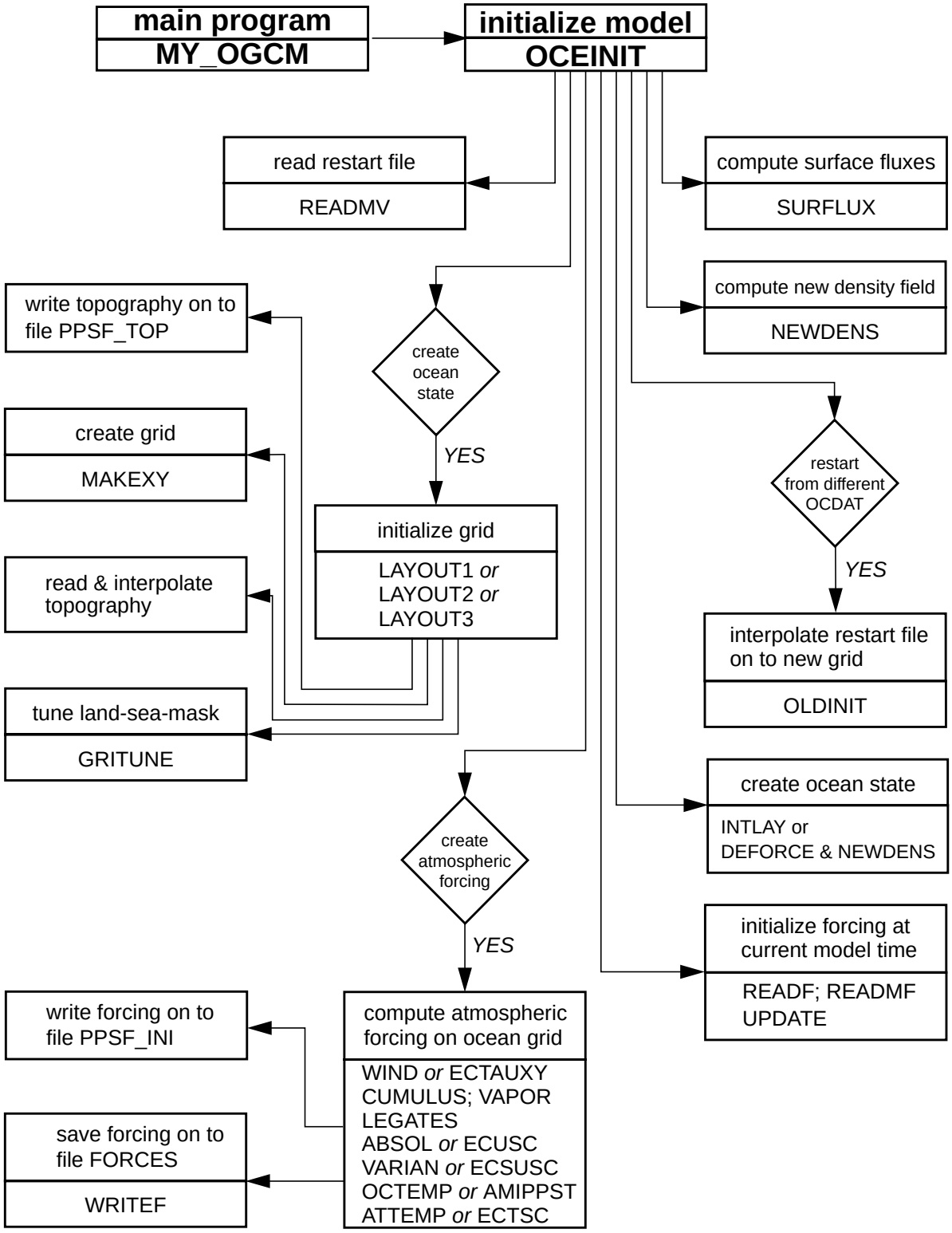

Figure 3 Main structure of <*OCEINIT*>, the driving routine for initialization.

4.3 FLOW OF ROUTINE <OCESTEP> (3)

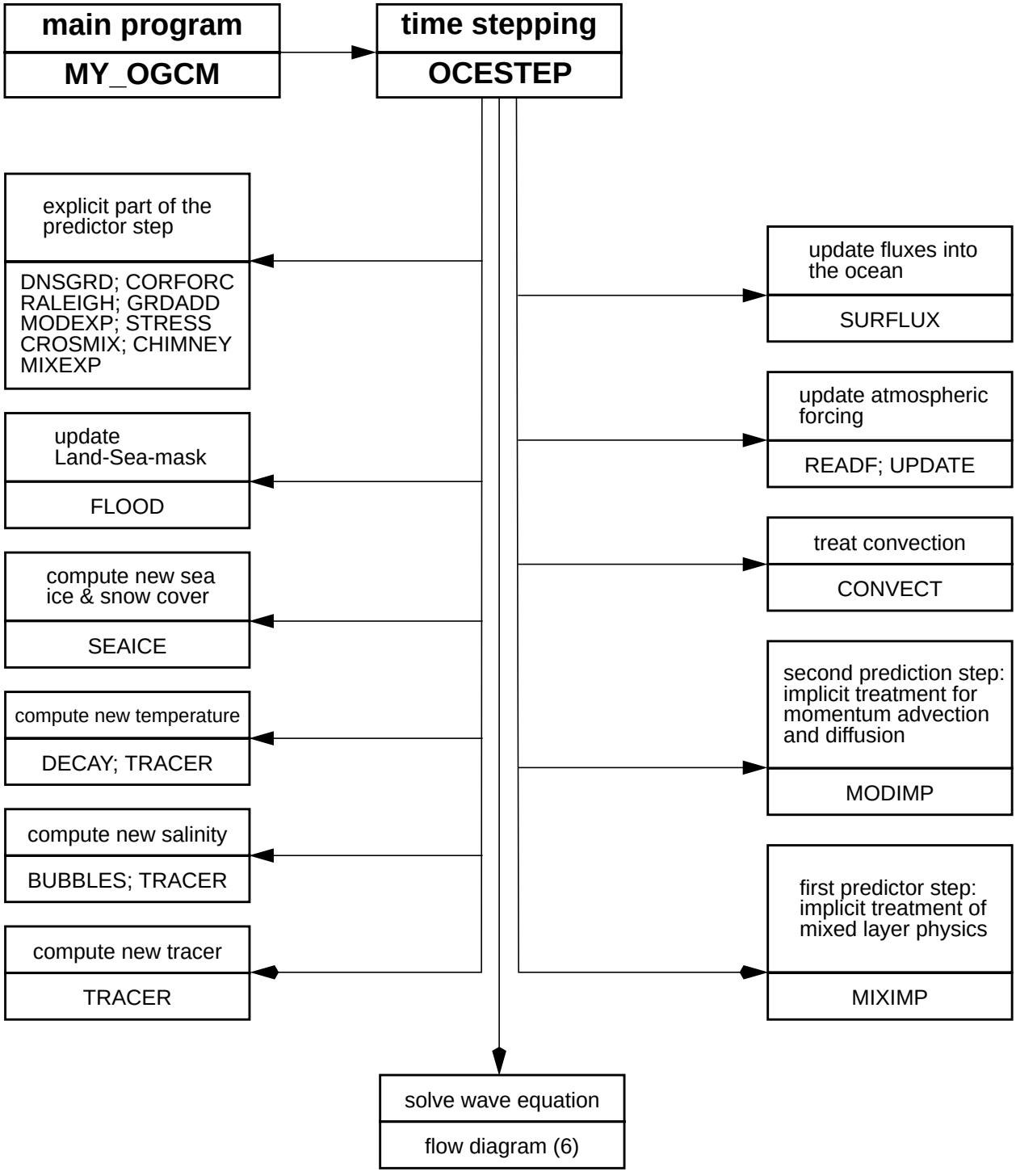

Figure 4 Main structure of <*OCESTEP*>, the driving routine to carry out time steps.

4.4 FLOW OF ROUTINE <OCEPOST> (4)

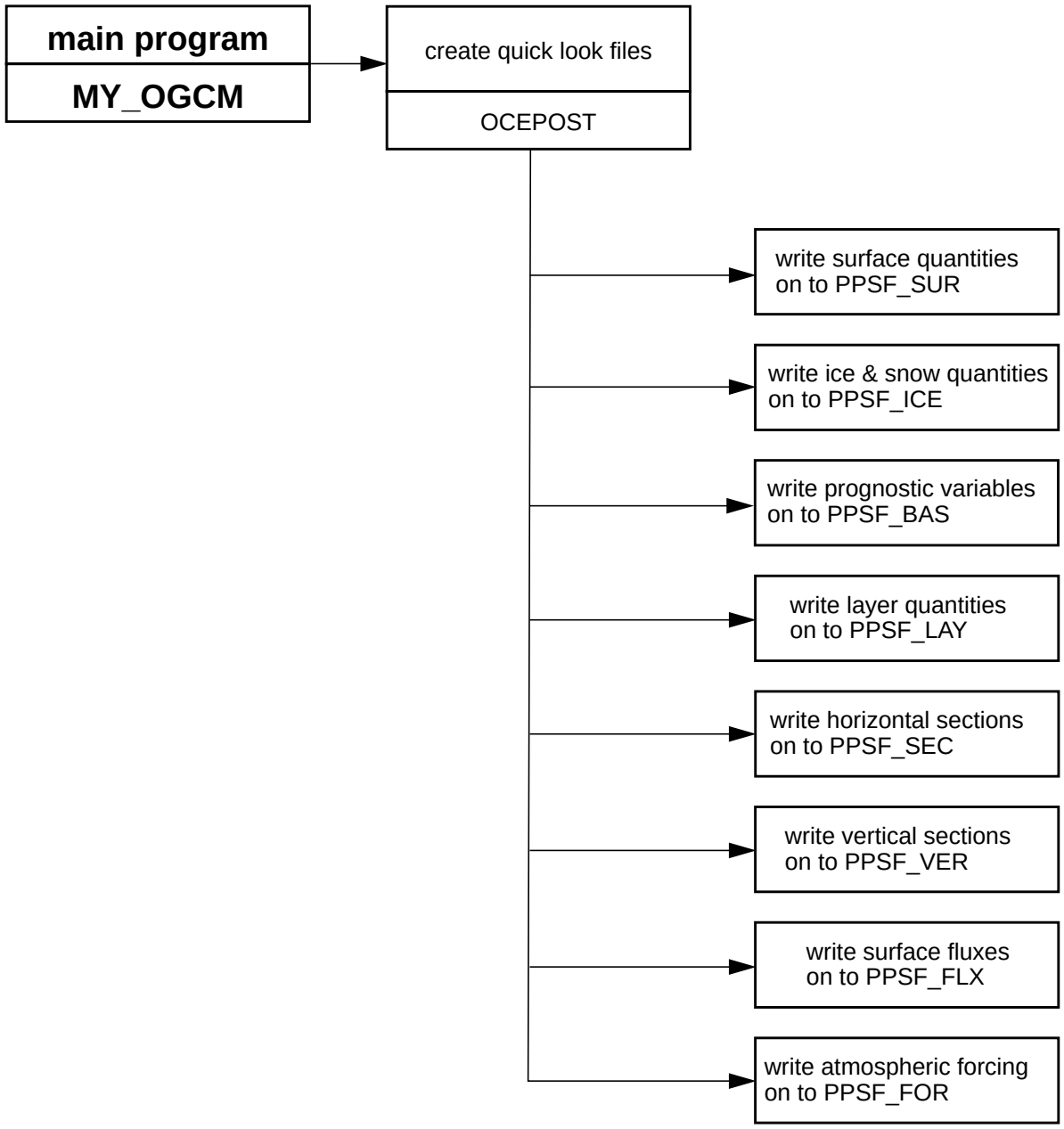

Figure 5 Main structure of <*OCEPOST*>, the driving routine for the data postprocessing.

4.5 FLOW OF ROUTINE <OCESTOP> (5)

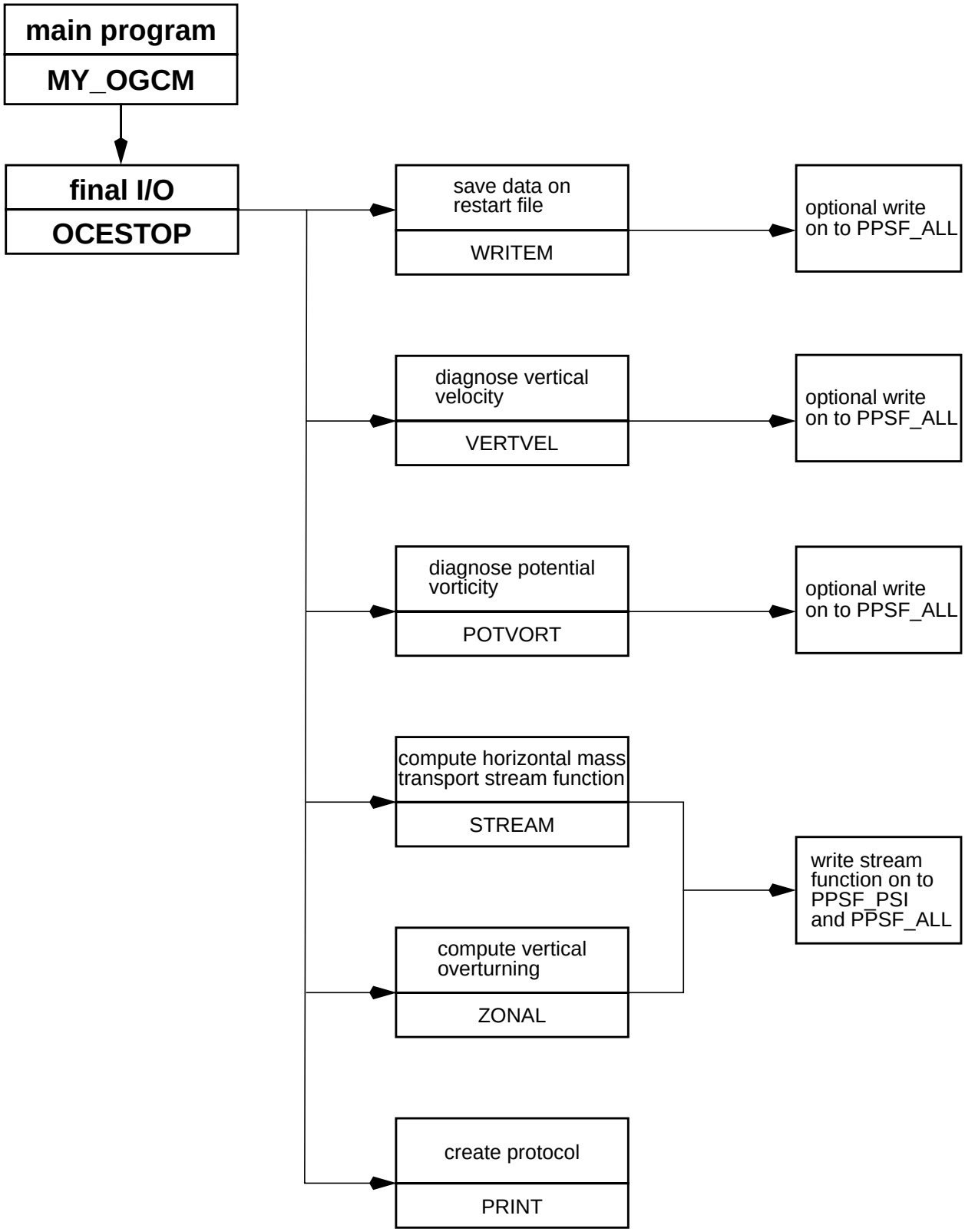

Figure 6 Main structure of <*OCESTOP*>, the driving routine to do final calculations.

4.6 FLOW OF SOLUTION OF WAVE EQUATION (6)

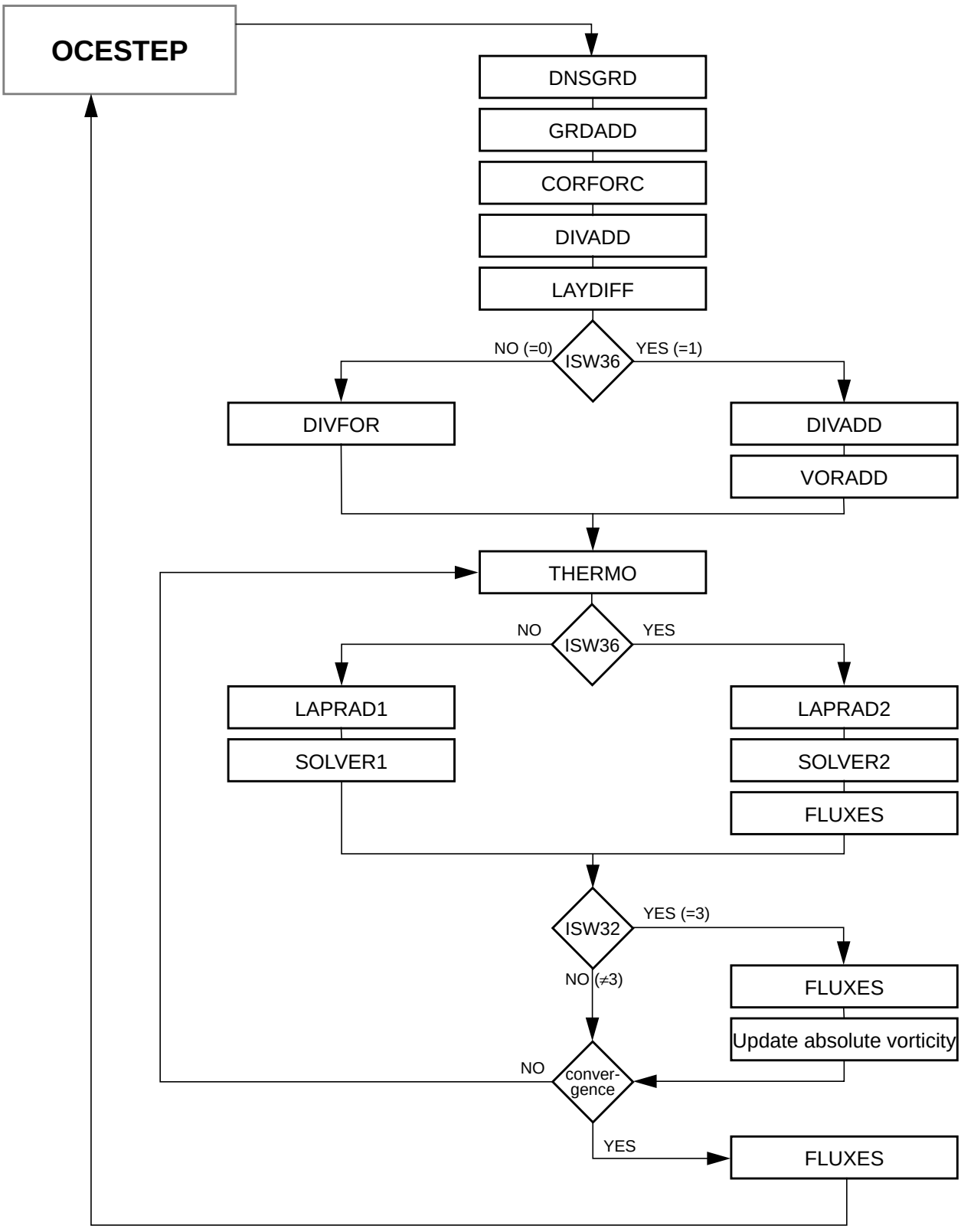

Figure 7 Main structure of solution of the wave equation within <*OCESTEP*>.

5. MODEL CODE DESCRIPTION

5.1 GENERAL REMARKS

The ocean code is devided up into 6 files which are [*ocemain.f*], [*ocestep.f*], [*oceinit.f*], [*ocepost.f*], [*ocestop.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, [*oceinit.f*] those routines required to initialize the model, [*ocepost.f*] contains all routines for the post processing and [*ocestop.f*] does final operations to stop a model run. [*ocemods.f*] optionally contains routines of [*ocestep.f*], [*oceinit.f*], [*ocepost.f*] or [*ocestop.f*] that required some specific changes and overwrite the original version of those routines while the code is loaded. In general, [*ocestep.f*], [*oceinit.f*], [*ocepost.f*] and [*ocestop.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 (see [*make.job*] and [*model.job*]).

5.2 CONTENTS OF [*ocemain.f*]

5.2.1 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 contains also calls to the atmosphere and to routines that carry out the communication between both systems. Thereby, the ocean is considered as black box and any data exchange with it is completed by the coupling interface routines.

5.2.2 Definition of Control Parameters

The meaning of the control parameters is listed in each of the BLOCK DATAs. Henceforth, we distinguish between default values and constants. Default values are marked with a '~' and can be understood as recommended and well-tuned 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 '~' and should be never changed. Values such as '0, 1, 2' are 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 INTERGER variables, while all other variables are REAL except for a few cases of CHARACTER definitions.

5.2.2.1 Block Data |*SCREW*|

|*SCREW*| defines the time step and its limits. These are the maximal CPU time variable of the run, *CALCMAX*, and the maximal number of time steps carried out during one call of <*OCESTEP*>, *MAXREP*. Furthermore, a time counter defines the number of time steps when the model stops, unless *MAXREP* or *CALCMAX* are set to expire before this time (see also Table 1 in which CPUs denotes seconds CPU-time and MODs denotes seconds of model-time). Switches control the coupling interface. *NHEAT*, *NFRESH*, *NSTRESS*, *NMIX*, *NICE* and *NSOL* are active only if *NINIT*=1. Depending of how the switches *NHEAT* through *NSOL* are set, the fluxes are either interpreted as total fields or as anomalies (see Table 2).

Table 1 Time Step Control Parameters

Variable	Value	Unit	Meaning	Reference
MAXREP	any > 0		maximal number of time steps	<*OCESTEP*>
CALCMAX	any > 0	CPUs	CPU-time after 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 ≥ 0	MODh	completed hours for model stop	<*OCESTEP*>
NDAY	any ≥ 0	MODd	completed days for model stop	<*OCESTEP*>
NMONTH	any ≥ 0	MODm	completed months for model stop	<*OCESTEP*>
NYEAR	any ≥ 0	MODy	completed years for model stop	<*OCESTEP*>

Table 2 Coupling Interface Parameters

Variable	Value	Default	Meaning	Reference
NINIT	0, 1	0	switch for coupling interface	*/JMOGLOB/*
NDRUCK	0, 1	0	switch for print	*/JMOGLOB/*
NHEAT	0, 1, 2	0	switch for heat coupling	<*SURFLUX*>
NFRESH	0, 1, 2	0	switch for salt coupling	<*SURFLUX*>
NSTRESS	0, 1, 2	0	switch for stress coupling	<*OCESTEP*>
NMIX	0, 1, 2	0	switch for mixing energy coupling	<*SURFLUX*>
NICE	0, 1, 2	0	switch for snow & sea-ice coupling	<*SURFLUX*>
NSOL	0, 1, 2	0	switch for solar radiation coupling	<*SURFLUX*>

5.2.2.2 Block Data |*HEATFLX*|

|*HEATFLX*| defines all variables required to compute heat and fresh water fluxes (see Table 3).

Table 3 Heat Flux Parameters

Variable	Value	Unit	Meaning	Reference
HEATLAT	2500800	$\mathrm{W s\,kg^{-1}}$	heat of evaporation/condensation	<*SURFLUX*>
SIGMA	5.67E-8	$\mathrm{W s\,K^{-1}}$	Stefan Boltzmann constant	<*SURFLUX*>
SOLAR	1367	$\mathrm{W m^{-2}}$	solar constant	<*SURFLUX*>
GAS	287	$\mathrm{m^2 K^{-1} s^{-2}}$	gas constant of dry air	<*SURFLUX*>
CPAIR	1005	$\mathrm{Ws\,(kgK)^{-1}}$	specific heat capacity of air	<*SURFLUX*>
CONSTS	0.0327		constant for sensible transfer coefficient	<*VARDRAG*>
CONSTF	0.0346		constant for latent tranfer coefficient	<*VARDRAG*>
CHARNCK	0.032		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 of water for long wave 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 snow and ice	<*SURFLUX*>

5.2.2.3 Block Data |*HORIMIX*|

|*HORIMIX*| defines parameters required for computing the horizontal diffusion coefficients for scalar and vector quantities (see Table 4). These parameters MUST be adjusted to the requirements of numerical stability, which depends on the grid resolution. Use equations (2.2.2.3) and (2.2.2.4) to determine which diffusion coefficients the model computes internally. See also <*ISODIFF*>, <*MODEXP*>, <*MODIP*> and <*TRACER*>. Note that too large values for *DIADIF*0 and *DIADIF*1 will result in numerical instability ($A^h < 500\mathrm{m^2 s^{-1}}$).

Table 4 Parameters for Horizontal Diffusion

Variable	Value	Unit	Meaning	Reference
PROPORT	~1.E-9	m	proportionality coefficient	<*ISODIFF*>
DIFTIMV	~1.E6	s	time scale for momentum	<*MODEXP*>
DIFTIMS	~1.E7	s	time scale for scalars	<*TRACER*>
DIFMIN	~2000	$m^2 s^{-1}$	smallest scalar diffusion	<*ISODIFF*>
DIADIF0	~200	$m^2 s^{-1}$	layer diffusion coefficient	<*DIADIFF*>
DIADIF1	~1.E-10	m	proportionality coefficient	<*DIADIFF*>

5.2.2.4 Block Data |*ICE*|

Table 5 Snow and Sea Ice Parameters

Variable	Value	Unit	Meaning	Reference
CPMELT	334000	$W s kg^{-1}$	heat of freezing/melting	<*SEAICE*>
CPICE	2090	$Ws\ (kgK)^{-1}$	specific heat capacity of ice	<*SEAICE*>
SALTICE	0.5	$g kg^{-1}$	salinity of sea ice	<*SEAICE*>
TMELT	273.15	K	freezing point of fresh water	<*SURFLUX*>
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	~10.0	$W s kg^{-1}$	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-8	$m s^{-1}$	coefficient that limits ice pressure	<*SEAICE*>
AGING	~1.E-7	s^{-1}	snow to ice conversion rate	<*SEAICE*>
CICE	~2.0	$W (K m)^{-1}$	heat conductivity for ice	<*SURFLUX*>
CSNOW	~0.3	$W (K m)^{-1}$	heat conductivity for snow	<*SURFLUX*>
EDDYS	~2000	$m^2 s^{-1}$	scalar diffusion coefficient	<*SEAICE*>
EDDYUV	~2000	$m^2 s^{-1}$	vector diffusion coefficient	<*SEAICE*>
RHOICE	~900	$kg m^{-3}$	ice density	<*SURFLUX*>
RHOSNOW	~300	$kg m^{-3}$	snow density	<*SURFLUX*>

|*ICE*| defines parameters for the snow and sea ice model (see Table 5). Note that *CMELT* and *CFREEZ* are tuning coefficients that allow changes at the rate at which heating causes the opening of leads or heat loss causes ice to form within leads.

5.2.2.5 Block Data |*LAYOUT*|

|*LAYOUT*| defines the model boundaries and the parameters used to create a focus, i.e., an area with

Table 6 Parameters for Grid Definition

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*>
ISWITCH	0, 1	0		switch for Eulerian angles	<*GEOTOMO*>
ALPHA	any	0	deg	first angle in longitude	<*GEOTOMO*>
BETA	any	0	deg	second angle in latitude	<*GEOTOMO*>
GAMMA	any	0	deg	third angle in new longitude	<*GEOTOMO*>

enhanced resolution. Furthermore, the Eulerian angles allow one to rotate the model longitudes and latitudes on the sphere. This is useful for rotating a coordinate pole out of the model area to avoid the coordinate convergence and the resulting small time steps near the pole (see Table 6). Furthermore, eliminating points allow one to remove lakes from the model area. They are disabled if set to zero (see Table 7). 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 8).

Table 7 Grid Tuning: Eating Points

Variable	Value	Default	Meaning	Reference
*KILL*1*I*	any index	0	x-index for eating point #1	<*ELIMIN*>
*KILL*1*J*	any index	0	y-index for eating point #1	<*ELIMIN*>
*KILL*2*I*	any index	0	x-index for eating point #2	<*ELIMIN*>
*KILL*2*J*	any index	0	y-index for eating point #2	<*ELIMIN*>
*KILL*3*I*	any index	0	x-index for eating point #3	<*ELIMIN*>
*KILL*3*J*	any index	0	y-index for eating point #3	<*ELIMIN*>
*KILL*4*I*	any index	0	x-index for eating point #4	<*ELIMIN*>
*KILL*4*J*	any index	0	y-index for eating point #4	<*ELIMIN*>
*KILL*5*I*	any index	0	x-index for eating point #5	<*ELIMIN*>
*KILL*5*J*	any index	0	y-index for eating point #5	<*ELIMIN*>
*KILL*6*I*	any index	0	x-index for eating point #6	<*ELIMIN*>
*KILL*6*J*	any index	0	y-index for eating point #6	<*ELIMIN*>

Table 8 Grid Tuning: Bridges and Channels

Variable	Value	Meaning	Reference
*JSW*1	0, 1	switch for Panama bridge	<*GRITUNE*>
*JSW*2	0, 1	switch for Florida Peninsula	<*GRITUNE*>
*JSW*3	0, 1, 2	switch for Thule land bridge or channels	<*GRITUNE*>
*JSW*4	0, 1, 2	switch for Hudson Bay closed or channel	<*GRITUNE*>
*JSW*5	0, 1	switch for cut off of Ochotskic Sea	<*GRITUNE*>
*JSW*6	0, 1	switch for bridge Spitzbergen-Norway	<*GRITUNE*>
*JSW*7	0, 1, 2	switch for bridge Alaska/Russia or channel	<*GRITUNE*>
*JSW*8	0, 1, 2	switch for bridge Gibraltar or channel	<*GRITUNE*>
*JSW*9	0, 1	switch for Weddell Sea Peninsula	<*GRITUNE*>
*JSW*10	0, 1, 2	switch for Cuba+Thaiti or wipeout	<*GRITUNE*>
*JSW*11	0, 1	switch for bridge Malaysia and Indonesia	<*GRITUNE*>
*JSW*12	0, 1	switch for channel Met. Sea and Black Sea	<*GRITUNE*>

Table 8 Grid Tuning: Bridges and Channels

Variable	Value	Meaning	Reference
*JSW*13	0, 1	switch for bridge: Australia-Indonesia	<*GRITUNE*>
*JSW*14	0, 1	switch for bridge: Greenland-Canada	<*GRITUNE*>
*JSW*15	0, 1	switch for island: Novaja Semlja	<*GRITUNE*>
*JSW*16	0, 1	switch for island: Severnaja Semlja	<*GRITUNE*>
*JSW*17	0, 1	switch for island: Japan	<*GRITUNE*>
*JSW*18	0, 1	switch for island: New Zealand	<*GRITUNE*>
*JSW*19	0, 1, 2	switch for bridge: Baltic-North Sea	<*GRITUNE*>
*JSW*20	0, 1, 2	switch for bridge: Red Sea and Indic ocean	<*GRITUNE*>
*JSW*21	0, 1, 2	switch for Mediterranean islands and bridges	<*GRITUNE*>

5.2.2.6 Block Data |*PHYSICS*|

KTOTAL defines the model time when initialized. *TEMMEAN* and *SALMEAN* determine the prescribed

Table 9 Parameters for Coordinate Generation

Variable	Value	Unit	Meaning	Reference
KTOTAL	any	s	initial model time	<*OCEINIT*>
KSHIFT	1296000	s	length of half a month	<*OCEINIT*>
SALMEAN	~35.0	$g\,kg^{-1}$	salinity used for coordinates	<*DEFORCE*>
TEMMEAN	~298.15	K	temperature used for coordinates	<*DEFORCE*>
HTOTAL	~6000	m	reference depth of ocean	<*OCEINIT*>
UMFANG	40030174	m	earth's circumference	<*MAKEXY*>

potential density that is used as potential density coordinate. The isopycnal coordinates 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. This results in better vertical resolution for higher potential density. It is a compromise between constant vertical resolution and resolution of potential densities (see Table 9). 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 10). The definition of *TURBLEV* allows one to tune the upper ocean thermocline, while *HMIX* is more responsible for the deep ocean (see Table 11). The mixed layer parameters are well-tuned (see Table 12).

Table 10 Some Physical Parameters

Variable	Value	Unit	Meaning	Reference
CP	4180	$\mathrm{Ws\,(kgK)^{-1}}$	specific heat capacity of water	*/JMOPAR/*
GRAV	9.81	$\mathrm{m\,s^{-2}}$	earth acceleration	*/JMOPAR/*
DRAGT	~5.E-6	$\mathrm{s^{-1}}$	coefficient for salinity forcing	<*SURFLUX*>
DRAGS	~1.5E-6		drag coefficient at surface	<*STRESS*>
DRAGV	~1.E-5		drag coefficient between layers	<*STRESS*>
DRAGB	~1.E-4		drag coefficient at bottom	<*STRESS*>

Table 11 Parameters for Diapycnal Diffusion

Variable	Value	Unit	Meaning	Reference
EPSILON	0.10	$\mathrm{m^2s^{-2}}$	vertical mixing damping coef.	<*CROSMIX*>
RICRIT	0.25		critical Richardson Number	<*CROSMIX*>
HMIX	~500	m	tuning coefficient	<*CROSMIX*>
TURBLEV	~4.E-7	$\mathrm{m^3s^{-3}}$	tuning coefficient	<*CROSMIX*>

Table 12 Parameters for Mixed Layer

Variable	Value	Unit	Meaning	Reference
*CMIX*0	~0	$\mathrm{m\,s^{-3}}$	linear damping term of ML	<*MIXEXP*>
*CMIX*1	~1.E6	m	constant TKE decay length scale	<*MIXEXP*>
*CMIX*2	0.25		critical Richardson number	<*MIXEXP*>
*CMIX*3	~2.5		Ekman dissipation for TKE	<*MIXEXP*>
*CMIX*4	~1.25		wind to wave energy conversion factor	<*MIXEXP*>
*CMIX*5	~0		Garwood-term tuning coefficient	<*MIXEXP*>
*CMIX*6	~1.E6	m	constant buoyancy decay length scale	<*MIXEXP*>
*CMIX*7	~0.5		Ekman dissipation for buoyancy flux	<*MIXEXP*>
*CMIX*8	~0.05	$\mathrm{m\,s^{-2}}$	threshold for g’	<*MIXEXP*>
*CMIX*9	~1.0		tuning coefficient for buoyancy flux	<*MIXEXP*>

Table 12 Parameters for Mixed Layer

Variable	Value	Unit	Meaning	Reference
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*>

5.2.2.7 Block Data |*POSTPRO*|

|*POSTPRO*| defines parameters that allow one to control which vertical or horizontal sections appear on the quick-look files. Up to 10 different horizontal and/or vertical sections can be selected (see Table 13).

Table 13 Parameters for Quick-Look System

Variable	Value	Default	Meaning	Reference
ISEC(10)	$1 \le ISEC \le NX$	0	x-indices for cross sections	<*OCEPOST*>
JSEC(10)	$1 \le JSEC \le NY$	0	y-indices for cross sections	<*OCEPOST*>
VSEC(10)	any in m	0	depth of horizontal sections	<*OCEPOST*>
KSEC	$1 \le KSEC \le NZ$	0	z-index for layer sections	<*OCEPOST*>

5.2.2.8 Block Data |*PRINTER*|

|*PRINTER*| defines switches that control which quantities are written to [*OUTLIST*]. A focus allows one to print every grid point within some area (see Table 14).

Table 14 Switches for Listing

Variable	Value	Default	Meaning	Reference
KSW	0, 1	0	switch for focus used to generate listing	<*DRUCK..*>
MAXIM	$1 \le MAXIM \le NZ$	1	number of printed layers	<*LISTING*>
ILEFT	any but < *IRIGHT*	1	left index used for focus	<*LISTING*>
IRIGHT	any but > *ILEFT*	1	right index used for focus	<*LISTING*>
JLOW	any but < *JUPP*	1	lower index used for focus	<*LISTING*>
JUPP	any but > *JLOW*	1	upper index used for focus	<*LISTING*>
*JSW*1	0, 1	0	switch for horizontal fluxes	<*LISTING*>

Table 14 Switches for Listing

Variable	Value	Default	Meaning	Reference
*JSW*2	0, 1	0	switch for level of surface and interfaces	<*LISTING*>
*JSW*3	0, 1	0	switch for layer thickness	<*LISTING*>
*JSW*4	0, 1	1	switch for temperature	<*LISTING*>
*JSW*5	0, 1	1	switch for salinity	<*LISTING*>
*JSW*6	0, 1	0	switch for tracer concentrations	<*LISTING*>
*JSW*7	0, 1	0	switch for barotropic stream functions	<*LISTING*>
*JSW*8	0, 1	0	switch for entrainment rate	<*LISTING*>
*JSW*9	0, 1	0	switch for diffusion coefficient	<*LISTING*>
*JSW*10	0, 1	0	switch for vertical velocity	<*LISTING*>
*JSW*11	0, 1	0	switch for potential vorticity	<*LISTING*>
*JSW*12	0, 1	1	switch for velocity	<*LISTING*>
*JSW*13	0, 1	0	switch for potential density	<*LISTING*>
*JSW*14	0, 1	1	switch for SST error	<*LISTING*>
*JSW*15	0, 1	1	switch for sea ice and snow	<*LISTING*>
*JSW*16	0, 1	0	switch for topography	<*LISTING*>
*JSW*17	0, 1	1	switch for index field	<*LISTING*>

5.2.2.9 Block Data |*DEFINE*|

|*DEFINE*| defines all coefficients required for the equation of state (UNESCO) and for the equation of Bryden that converts temperature into potential temperature.

5.2.2.10 Block Data |*SWITCH*|

|*SWITCH*| defines switches that control how the forcing is computed and whether observed or artificial data should be used (see Table 15). To access these different available modes use *ISW*18=0, in order to select the Newton relaxation for the surface salinity. The switch *ISW*48 is noteworthy. With *ISW*48=1 the model computes the running mean fresh water flux. If the length of the run with *ISW*48=1 is a multiple of a year the array *PMEQ* will contain the mean fresh water flux. If the user switches to *ISW*48=2 the model will continue to compute the running mean, however, it will also use this mean value for forcing the surface salinity. If the run with *ISW*48=2 is long enough and again a multiple of a year, one can start to use *ISW*48=3. In this mode the *PMEQ* is used as ocean forcing, but however, the annual mean is not updated any more. This procedure allows 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.

Table 15 Switches for Type of Forcing

Variable	Value	Default	Meaning	Reference
*ISW*1	0, 1	0	switch from artificial to observed forcing	<*OCEINIT*>
*ISW*5	0, 1	0	switch from artificial to observed initial state	<*OCEINIT*>
*ISW*8	0, 1	0	switch from prescribed to observed topography	<*OCEINIT*>
*ISW*13	0, 1	0	switch from linear to parameterized heat fluxes	<*SURFLUX*>
*ISW*15	0, 1	0	switch from closed to open cyclic boundaries	<*OCEINIT*>
*ISW*18	0, 1	0	switch from linear to parameterized salt fluxes	<*SURFLUX*>
*ISW*41	0, 1	0	switch from prescribed to observed atmosphere	<*OCEINIT*>
*ISW*48	0, 1, 2, 3	0	switch for mode of fresh water forcing	<*OCESTEP*>

Table 16 Switches for Model Physics

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 horizontal advection/diffusion	<*OCESTEP*>
ISW7	0, 1	1	switch for Coriolis force	<*OCEINIT*>
*ISW*9	0, 1	1	switch for coordinate curvature	<*OCEINIT*>
*ISW*14	0, 1	1	switch for convection	<*OCESTEP*>
*ISW*25	0, 1	1	switch for temperature calculation	<*OCESTEP*>
*ISW*26	0, 1	1	switch for salt calculation	<*OCESTEP*>
*ISW*28	0, 1, 2	0	switch for closed / open North Pole	<*BND....*>
*ISW*33	0, 1	0	switch for chimney parameterization	<*OCESTEP*>
*ISW*34	0, 1	1	switch for vertical diffusion	<*OCESTEP*>
*ISW*39	0, 1, 2	1	switch for mass budget -> sea level	<*OCESTEP*>
*ISW*44	0, 1	1	switch for sea ice calculation	<*OCESTEP*>

Table 17 Switches for Numerical Schemes

Variable	Value	Default	Meaning	Reference
*ISW*27	0, 1	1	switch for diapycnal pressure gradient	<*OCESTEP*>
*ISW*32	0, 1, 2, 3	3	switch for choice of transport scheme	<*MODEXP*>
*ISW*36	0, 1	0	switch from direct to iterative Coriolis force	<*OCESTEP*>
*ISW*37	0, 1	0	switch for acceleration of tracer equation	<*OCESTEP*>

Other switches control the physical content (Table 16), select numerical schemes (Table 17), control the output (Table 18) and select the data source used to create [*FORCES*] (Table 19). The usual choice is *ISW*32=3 which means that the potential vorticity conserving scheme is taken instead of the Crowley scheme for momentum (*ISW*32=2), or the upstream scheme (*ISW*32=1) for momentum and tracers. Note that it is possible to choose an open pole for the sea ice model only with *ISW*28=1 while *ISW*28=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.

Table 18 Switches for Output

Variable	Value	Default	Meaning	Reference
*ISW*10	0, 1	0	switch for monthly output of basic fields	<*OCESTEP*>
*ISW*11	0, 1	0	switch for exit after topography generation	<*OCEINIT*>
*ISW*17	any ≤ 0	0	switch for generation of history file	<*OCESTEP*>
*ISW*21	0, 1	1	switch for generation of vertical velocity	<*OCESTOP*>
*ISW*22	0, 1	1	switch for generation of potential vorticity	<*OCESTOP*>
*ISW*23	0, 1	0	switch for generation of stream functions	<*OCESTOP*>
ISW24	0, 1	0	switch for monthly tracer forcing data	<*OCESTEP*>
ISW29	0, 1, 2	0	switch for massive error analysis	/*JMOFLAG*/
ISW31	0, 1	0	switch from binary to ASCII restart file [OCDAT]	<*WRITEM*>
ISW35	0, 1	1	switch for emergency exit	<*OCESTEP*>
*ISW*40	0, 1	0	switch for output for coupling runs	<*OCESTEP*>
*ISW*46	0, 1	0	switch for output of monthly data	<*OCESTEP*>

Table 19 Switches for Data Sources

Variable	Value	Default	Meaning	Reference
*ISW*12	0, 1	0	switch for use of other [OCDAT]	<*OCEINIT*>
*ISW*16	0, 1	1	switch from H&R to ECMWF stresses	<*OCEINIT*>
*ISW*30	0, 1	1	switch from COADS to AMIP SST	<*OCEINIT*>
*ISW*38	0, 1	1	switch from Scripps to NOAA topography	<*OCEINIT*>
*ISW*42	0, 1, 2	0	switch for initialization mode	<*OCEINIT*>
*ISW*43	0, 1	0, 1	switch from COADS to ECMWF air temperature	<*OCEINIT*>
*ISW*45	0, 1	0	switch for use of externally defined mask	<*OCEINIT*>
*ISW*47	0, 1	0, 1	switch from COADS to ECMWF winds	<*OCEINIT*>

*ISW*42 determines the initialization mode. If *ISW*42=0, then the model tries to initialize [*OCDAT*] and [*FORCES*] when an empty [*OCDAT*] is detected. If *ISW*42=1, then the model creates [*FORCES*] independent of whether an [*OCDAT*] exists. In case of *ISW*42=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.

5.2.2.11 Block Data |*TUNING*|

|*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 20). 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 *ISW*35, causes an error exit (see Table 21). Threshold parameters finally allow one to limit certain variables mainly for stability purposes (see Table 22).

Table 20 Time Weights for Implicit Scheme

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

Table 21 Numerical Tuning Variables

Variable	Value	Unit	Meaning	Reference
OVER	~1.25		overrelaxation coefficient	*/JMOTUNE/*
UNDER1	~1.05		underrelaxation except for ice	*/JMOTUNE/*
UNDER2	~1.05		underrelaxation for ice	*/JMOTUNE/*
NITMIN	~5		minimal repetition of any iteration	*/JMOITER/*
NITMAX	~20		maximal repetition of any iteration to exit	*/JMOITER/*
ACCURU	~0.1	$kg\,(sm)^{-1}$	required accuracy for momentum	<*MODIMP*>
ACCURH	~0.01	m	required accuracy for thickness	<*SOLVER*>
ACCURT	~0.00002	K	required accuracy for temperature	<*TRACER*>
ACCURS	~0.00001	$g\,(kg)^{-1}$	required accuracy for salinity	<*TRACER*>
ACCURF	~0.00001		required accuracy for tracer	<*TRACER*>

Table 22 Threshold Variables

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

5.3 THE OCEAN MODEL: [*ocestep.f*]

5.3.1 The Interior Ocean Model

5.3.1.1 Subroutine <*OCESTEP*>

This is the driving routine to carry out the time step. At each call *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.

5.3.1.2 Subroutine <*CONVECT*>

<*CONVECT*> computes the exchange of momentum, heat, salt and tracer concentration between two adjacent layers in each 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. Thereby, the coordinates are rearranged if the coordinate error exceeds a certain threshold.

5.3.1.3 Subroutine <*CORFORC*>

<*CORFORC*> computes the Coriolis force and adds the contribution to the momentum budget to the arrays *RESTU* and *RESTV.*

5.3.1.4 Subroutine <*CROSMIX*>

<*CROSMIX*> determines the upward and downward mass flux due to diapycnal mixing and writes the result on *VERTUP* and *VERTDN*, which are later used in <*TRACER*> to compute the related changes in temperature, salinity and tracer concentration. The contribution to the momentum budget is added to *RESTU* and *RESTV.*

5.3.1.5 Subroutine <*ISODIFF*>

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

5.3.1.6 Subroutine <*DIADIFF*>

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

5.3.1.7 Subroutine <*DIVADD*>

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

5.3.1.8 Subroutine <*DIVFOR*>

<*DIVFOR*> computes parts of the right side of the wave equation (3.3.1.13) and adds the contribution to *RESTH*. Compare with equations (3.3.1.9) and (3.3.1.10).

5.3.1.9 Subroutine <*DNSGRD*>

<*DNSGRD*> computes that part of the pressure gradient explicitly that is related to horizontal density gradients. Compare with equation (2.1.2.12). The result is added to *RESTU* and *RESTV.*

5.3.1.10 Subroutine <*FLUXES*>

<*FLUXES*> computes the new fluxes from the pressure gradient at the new time level and the already determined right side of the momentum equations that is not treated implicitly. Compare with equations (3.3.1.7) and (3.3.1.8).

5.3.1.11 Subroutine <*GRADY*>

<*GRADY*> computes explicitly that part of the pressure gradient that is related to horizontal layer thickness gradients. The result still lacks the horizontal grid spacing, time step etc. See also <*GRDADD*>.

5.3.1.12 Subroutine <*GRADMAT*>

<*GRADMAT*> 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 side of the linear equation system of the wave equation.

5.3.1.13 Subroutine <*GRDADD*>

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

5.3.1.14 Subroutine <*LAPRAD1*>

<*LAPRAD1*> computes that contribution to the right hand side of the wave equation that contains the gradients of density as a Laplacian operator. The routine updates this contribution during the iteration for the new layer thicknesses. It is used only when the Coriolis force is treated directly (*ISW*36=0).

5.3.1.15 Subroutine <*LAPRAD2*>

Similar to <*LAPRAD1*> but used when the Coriolis force is updated by an iterative process (*ISW*36=1).

5.3.1.16 Subroutine <*VORADD*>

<*VORADD*> computes the contribution of the rotation of the Coriolis force to the right side of the wave equation. It is required only if *ISW*36=1.

5.3.1.17 Subroutine <*LAYDIFF*>

<*LAYDIFF*> determines (explicitly) the layer thickness changes resulting from the layer thickness diffusion and adds the result to *RESTH*. *RESTH* is treated implicitly as part of a matrix within <*SOLVER1*> and <*SOLVER2*>.

5.3.1.18 Subroutine <*MODEXP*> & Entry <*MODIMP*>

<*MODEXP*> and <*MODIMP*> determine the momentum advection and diffusion. While <*MODEXP*> computes these terms explicitly and adds their contribution to *RESTU* and *RESTV,* <*MODIMP*> uses the line-relaxation method to treat these terms directly in x- and z-direction and iterates in y-direction.

5.3.1.19 Subroutine <*SETMAT*>

<*SETMAT*> determines the weighting coefficients that are required to compute the matrix for the wave equation.

5.3.1.20 Subroutine <*SOLVER1*>

<*SOLVER1*> is one of the major routines of the entire model. It performs two complete iterations for the wave equation and yields the layer thickness as result. It is used to solve the wave equation for the case that the Coriolis terms are treated directly as part of the linear equation system. It uses a line-relaxation method that solves a linear equation system directly on xz-slices and iterates in the y-direction.

5.3.1.21 Subroutine <*SOLVER2*>

<*SOLVER*2> is similar to <*SOLVER1*> but used when the Coriolis term is treated iteratively. The major difference from <*SOLVER1*> is that the matrix coefficients have to be computed slightly differently.

5.3.1.22 Subroutine <*STRESS*>

<*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.*

5.3.1.23 Subroutine <*TRACER*>

<*TRACER*> computes the new distribution of temperature or 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 <*OCESTEP*>.

5.3.2 The Mixed Layer Model

5.3.2.1 Subroutine <*EQUILIB*>

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

5.3.2.2 Subroutine <*MECHAN*>

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

5.3.2.3 Subroutine <*MIXEXP*> & Entry <*MIXIMP*>

<*MIXEXP*> and <*MIXIMP*> compute the mixed layer physics. While <*MIXEXP*> computes the entrainment and detrainment rates diagnostically and delivers changes of layer thickness and momentum to <*OCESTEP*>, <*MIXIMP*> treats the mixed layer implicitly by an iterative procedure to find the new layer thickness for the corrector step.

5.3.2.4 Subroutine <*SHEAR*>

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

5.3.3 The Sea Ice Model

5.3.3.1 Subroutine <*SEAICE*>

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

5.3.3.2 Subroutine <*BUBBLES*>

<*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*>.

5.3.3.3 Subroutine <*PFREEZ*>

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

5.3.4 The Equation of State

5.3.4.1 Subroutine <*EXPANDT*> & Entry <*EXPANDS*> / <*EXPANDP*> / <*EXPANDR*>

These routines compute the expansion coefficients in respect to temperature, salinity, in situ pressure and potential temperature from the underlying UNESCO formula. The differentiation is carried out analytically.

5.3.4.2 Subroutine <*NEWDENS*>

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

5.3.4.3 Subroutine <*REPRESS*>

<*REPRESS*> computes the in situ pressure.

5.3.4.4 Subroutine <*STATETR*> & Entry <*STATETP*>

<*STATETR*> and <*STATETP*> compute the in situ density or the potential density from temperature, salinity and in situ pressure for one layer.

5.3.4.5 Subroutine <*STATEVR*> & Entry <*STATEVP*>

<*STATEVR*> and <*STATEVP*> compute the in situ density or the potential density from temperature, salinity and in situ pressure for all layers.

5.3.4.6 Subroutine <*THERMO*>

<*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.

5.3.4.7 Subroutine <*TTOTET*>

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

5.3.4.8 Subroutine <*TTOTETA*> & Entry <*TETATOT*>

<*TTOTETA*> and <*TETATOT*> compute the potential temperature from temperature or temperature from potential temperature by inverting the formula of Bryden.

5.3.5 Model Forcing

5.3.5.1 Subroutine <*SURFLUX*>

<*SURFLUX*> computes the heat fluxes, fresh water fluxes and buoyancy fluxes from the atmospheric data, the model SST and 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 only for the dynamical part.

5.3.5.2 Subroutine <*VARDRAG*>

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

5.3.5.3 Subroutine <*DECAY*>

<*DECAY*> determines the heat flux into each layer resulting of the penetrating solar radiation. This is used when <*TRACER*> is called to determine the new temperature distribution.

5.4 PREPROCESSING: [*oceinit.f*]

5.4.1 Grid Initialization

5.4.1.1 Subroutine <*MAKEXY*>

<*MAKEXY*> initializes the x-and y-coordinates of the model using either the definitions for the model boundary (*XGL*, *XGR*, *YGU* and *YGO*) or the Gaussian grid for the T21, T42 or T106 resolutions. These grids are obtained when the preprocessor <*precomp.f*> activates the concerning parts in the code.

5.4.1.2 Subroutine <*ELIMIN*>

<*ELIMIN*> deletes sea points in the land-sea mask *IFLG* which are connected to the defined eating point indices (*KILLI.*, *KILLJ.*) by sea points. This routine is useful to delete undesired lakes or sea points that are not connected to the ocean region of interest.

5.4.1.3 Subroutine <*GRITUNE*>

<*GRITUNE*> calls <*ELIMIN*>, <*CHANNEL*> and <*BARRIER*> 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.

5.4.1.4 Subroutine <*FLAGMOD*>

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

5.4.1.5 Subroutine <*YCORT21*>

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

5.4.1.6 Subroutine <*YCORT42*>

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

5.4.1.7 Subroutine <*YCORT16*>

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

5.4.1.8 Subroutine <*BARRIER*>

<*BARRIER*> switches off sea points between two points in geographical coordinates and thus creates a land-bridge between two points.

5.4.1.9 Subroutine <*CHANNEL*>

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

5.4.1.10 Subroutine <*GEOTOMO*> & Entry <*MOTOGEO*>

These transform a pair of (x,y)-coordinates from geographical into model coordinates on the sphere or vice versa. 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.

5.4.2 Initialization of Atmospheric Forcing

Each atmospheric quantity used to compute fluxes into the ocean is initialized by its own routine. Thereby, global data sets are interpolated onto the model grid and stored by <*OCEINIT*> in [*FORCES*] via <*WRITEF*>. The file names used as data source from the following routines are the names of those of the permanent files, and not of the local files. See [*model.job*] for the local names.

5.4.2.1 Subroutine <*DEFORCE*>

This routine defines an artificial forcing when *ISW*1=0. In this case no atmospheric data are used.

5.4.2.2 Subroutine <*ABSOL*>

<*ABSOL*> reads one month of the COADS surface wind speed from [*UVABS*] and interpolates it onto the model grid.

5.4.2.3 Subroutine <*VARIAN*>

<*VARIAN*> reads one month of the COADS standard deviation of the absolute wind speed from [*UVDEV*] and interpolates it onto the model grid.

5.4.2.4 Subroutine <*OCTEMP*>

<*OCTEMP*> reads one month of the COADS sea surface temperature from [*SSTEMP*] and interpolates it onto the model grid.

5.4.2.5 Subroutine <*ATTEMP*>

<*ATTEMP*> reads one month of the COADS air temperature from [*AIRGLOB*] and interpolates it onto the model grid.

5.4.2.6 Subroutine <*CUMULUS*>

<*CUMULUS*> reads one month of the COADS cloud cover from [*COVER*] and interpolates it onto the model grid.

5.4.2.7 Subroutine <*ECSUSC*>

<*ECSUSC*> reads one month of the ECMWF standard deviation of the wind from [*STDV_USC*] and interpolates it onto the model grid.

5.4.2.8 Subroutine <*ECTAUXY*>

<*ECTAUXY*> reads one month of the ECMWF surface wind stress from [*TAUXY*] and interpolates it onto the model grid.

5.4.2.9 Subroutine <*ECTSC*>

<*ECTSC*> reads one month of the ECMWF surface temperature from [*TSC*] and interpolates it onto the model grid.

5.4.2.10 Subroutine <*ECUSC*>

<*ECUSC*> reads one month of the ECMWF absolute wind speed from [*USC*] and interpolates it onto the model grid.

5.4.2.11 Subroutine <*LEGATES*>

<*LEGATES*> reads one month of the Legates precipitation data set from [*RAIN1DEG*] and interpolates it onto the model grid.

5.4.2.12 Subroutine <*SEASALT*>

<*SEASALT*> reads the Levitus annual sea surface salinity from [*SSSALT*] and interpolates it onto the model grid.

5.4.2.13 Subroutine <*AMIPSST*>

<*AMIPSST*> reads one month of the AMIP sea surface temperature from [*AMIPSST*] and interpolates it onto the model grid.

5.4.2.14 Subroutine <*VAPOR*>

<*VAPOR*> reads one month of the COADS surface relative humidity from [*WETNESS*] and interpolates it onto the model grid.

5.4.2.15 Subroutine <*WIND*>

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

5.4.2.16 Subroutine <*UPDATE*>

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

5.4.2.17 Subroutine <*WRITEF*> & Entry <*READF*>

<*WRITEF*> and <*READF*> save the interpolated atmospheric forcing on [*FORCES*] or read the forcing from [*FORCES*].

5.4.3 Initialization of Ocean State

5.4.3.1 Subroutine <*LAYOUT1*>

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

5.4.3.2 Subroutine <*LAYOUT2*>

<*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.

5.4.3.3 Subroutine <*LAYOUT3*>

<*LAYOUT3*> is a special version of <*LAYOUT2*> but is used for grid rotation. Note that <*LAYOUT2*> and <*LAYOUT3*> are very expensive.

5.4.3.4 Subroutine <*DEFTOPO*>

<*DEFTOPO*> allows the user to define his own topography and land-sea distribution. It is called by choosing I*SW*8=0.

5.4.3.5 Subroutine <*INTLAY*>

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

5.4.3.6 Subroutine <*INTERPO*>

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

5.4.3.7 Subroutine <*LAYDEPT*>

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

5.4.3.8 Subroutine <*INTEGRA*>

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

5.4.3.9 Subroutine <*CREATIV*>

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

5.5 POSTPROCESSING: [*ocepost.f*]

The postprocessing driving routines are <*OCEPOST*> and its four entries <*OCEPST1*>, <*OCEPST2*>, <*OCEPST3*> and <*OCEPST4*>. The choice of which data are to be written out onto the quick-look files and/or the history files is made here and depends on the model's application.

Important: 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*]).

5.5.1 Subroutine <*OCEPOST*>

<*OCEPOST*> writes the final state of the model onto the quick-look files (see section 6.4.5.1).

5.5.2 Entry <*OCEPST1*> of Subroutine <*OCEPOST*>

<*OCEPST1*> writes data onto the history file at each time step.

5.5.3 Entry <*OCEPST2*> of Subroutine <*OCEPOST*>

<*OCEPST2*> writes data onto the history file at each 15th of a month.

5.5.4 Entry <*OCEPST3*> of Subroutine <*OCEPOST*>

<*OCEPST3*> writes data onto the history file at each nth time step, where n is defined by *ISW*17.

5.5.5 Entry <*OCEPST4*> of Subroutine <OCEPOST>

<*OCEPST4*> writes data onto the history file at the end of the job run.

5.5.6 General OUTPUT Routine

The output of data is not straightforward in a model that uses Lagrangian coordinates. Since the data post-processing 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.

5.5.6.1 Subroutine <*PPSFOUT*> & Entry <*PPSFAST*>

These routines write data in well-defined format (see section 6.4.2) onto one of the history or onto one of the quick-look files. The data format henceforth is called PPSF (Post Processing System Format).

5.5.7 Output of Averaged Quantities

For many purposes it is sufficient to provide data on 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.

5.5.7.1 Subroutine <*QENTXY*>

<*QENTXY*> writes out the time-averaged entrainment and detrainment.

5.5.7.2 Subroutine <*QFLUXY*>

<*QFLUXY*> writes out the time-averaged heat flux, fresh water flux and wind stress components.

5.5.7.3 Subroutine <*QHEATXY*>

<*QHEATXY*> writes out the time-averaged heat budget of the surface mixed layer for its separate terms, namely, the external heating, the horizontal advection, horizontal diffusion and the entrainment.

5.5.7.4 Subroutine <*QICEXY*>

<*QICEXY*> writes out time-averaged ice flow components, ice thickness and ice compactness.

5.5.7.5 Subroutine <*QMLDXY*>

<*QMLDXY*> writes out the time-averaged mixed layer depth onto the history file.

5.5.7.6 Subroutine <*QSSTXY*>

<*QSSTXY*> writes out the time-averaged sea surface temperature onto the history file.

5.5.7.7 Subroutine <*QSALTXY*>

<*QSALTXY*> writes out the time-averaged sea surface salinity onto the history file.

5.5.7.8 Subroutine <*QSLDXY*>

<*QSLDXY*> writes out the time-averaged sea level onto the history file.

5.5.7.9 Subroutine <*QUVXY*>

<*QUVXY*> writes out the time-averaged sea surface horizontal velocity components onto the history file.

5.5.7.10 Subroutine <*QUCOXZ*>

<*QUCOXZ*> writes out the time-averaged x-component of the flow on a xz-section onto the history file.

5.5.7.11 Entry <*QVCOXZ*> of Subroutine <*QUCOXZ*>

<*QVCOXZ*> writes out the time-averaged y-component of the flow on a xz-section onto the history file.

5.5.7.12 Entry <*QTETAXZ*> of Subroutine <*QUCOXZ*>

<*QTETAXZ*> writes out the time averaged temperature on a xz-section onto the history file.

5.5.7.13 Entry <*QSALTXZ*> of Subroutine <*QUCOXZ*>

<*QSALTXZ*> writes out the time averaged salinity on a xz-section onto the history file.

5.5.7.14 Entry <*QDENSXZ*> of Subroutine <*QUCOXZ*>

<*QDENSXZ*> writes out the time averaged potential density on a xz-section onto the history file.

5.5.7.15 Subroutine <*QUCOYZ*>

<*QUCOYZ*> writes out the time-averaged x-component of the flow on a yz-section onto the history file.

5.5.7.16 Entry <*QVCOYZ*> of Subroutine <*QUCOYZ*>

<*QVCOYZ*> writes out the time-averaged y-component of the flow on a yz-section onto the history file.

5.5.7.17 Entry <*QTETAYZ*> of Subroutine <*QUCOYZ*>

<*QTETAYZ*> writes out the time averaged temperature on a yz-section onto the history file.

5.5.7.18 Entry <*QSALTYZ*> of Subroutine <*QUCOYZ*>

<*QSALTYZ*> writes out the time averaged salinity on a yz-section onto the history file.

5.5.7.19 Entry <*QDENSYZ*> of Subroutine <*QUCOYZ*>

<*QDENSYZ*> writes out the time averaged potential density on a yz-section onto the history file.

5.5.8 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.

5.5.8.1 Subroutine <*VALL*> & Entry <*SALL*>

<*VALL*> and <*SALL*> write out all layers of a model array without any transformation. These routines should be used if one wants to reconstruct model quantities during the data post-processing.

5.5.8.2 Subroutine <*VINT*> & Entry <*SINT*>

<*VINT*> and <*SINT*> write out a precalculated single value e.g. as integrals over all dimensions.

5.5.8.3 Subroutine <*VYINTXZ*> & Entry <*SYINTXZ*>

<*VYINTXZ*> and <*SYINTXZ*> write out a precalculated one-dimension vector for a xz-section.

5.5.8.4 Subroutine <*VXINTXZ*> & Entry <*SXINTXZ*>

<*VXINTXZ*> and <*SXINTXZ*> write out a precalculated one-dimension vector for a yz-section.

5.5.8.5 Subroutine <*VXONE*> & Entry <*SXONE*>

<*VXONE*> and <*SXONE*> write out a precalculated one-dimensional vector in x-direction.

5.5.8.6 Subroutine <*VYONE*> & Entry <*SYONE*>

<*VYONE*> and <*SYONE*> write out a precalculated one-dimensional vector in y-direction.

5.5.8.7 Subroutine <*VXSEC*> & Entry <*SXSEC*>

<*VXSEC*> and <*SXSEC*> write out a one-dimensional vector in x-direction.

5.5.8.8 Subroutine <*VYSEC*> & Entry <*SYSEC*>

<*VYSEC*> and <*SYSEC*> write out a one-dimensional vector in y-direction.

5.5.8.9 Subroutine <*VZSEC*> & Entry <*SZSEC*>

< *VZSEC*> and <*SZSEC*> write out a one-dimensional vector in z-direction for certain levels.

5.5.8.10 Subroutine <*VXYONE*> & Entry <*SXYONE*>

<*VXYONE*> and <*SXYONE*> write out a precalculated xy-section at a given depth.

5.5.8.11 Subroutine <*VXYSEC*> & Entry <*SXYSEC*>

<*VXYSEC*> and <*SXYSEC*> write out an internally computed xy-section at a given depth.

5.5.8.12 Subroutine <*VXZSEC*> & Entry <*SXZSEC*>

<*VXZSEC*> and <*SXZSEC*> write out an internally computed xz-section at a given latitude.

5.5.8.13 Subroutine <*VYZSEC*> & Entry <*SYZSEC*>

<*VYZSEC*> and <*SYZSEC*> write out an internally computed yz-section at a given longitude.

5.6 OTHER ROUTINES: [*ocestop.f*]

5.6.1 Tridiagonal Solvers

5.6.1.1 Subroutine <*TRIDIAX*>

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

5.6.1.2 Subroutine <*TRIDIAY*>

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

5.6.1.3 Subroutine <*TRIDIA3*>

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

5.6.1.4 Subroutine <*TRIDIAM*>

<*TRIDIAM*> computes the solution of *NV* linear equations of tridiagonal type for *NX* number of equations. $NV = (NY-1)/2$ is the result of the line-relaxation method used. It is used by <*STREAM*>.

5.6.1.5 Subroutine <*TRIDIAV*>

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

5.6.1.6 Subroutine <*TRIONE*>

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

5.6.1.7 Subroutine <*TRITWO*>

<*TRITWO*> 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 <*MODIMP*> and is required to couple the x- and y-components of the advection and curvature terms, since flux components are cross-referenced in the momentum equations.

5.6.1.8 Subroutine <*TRIBLCK*>

<*TRIBLCK*> 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. It is called by <*SOLVER1*> and <*SOLVER*2> .

5.6.2 Output of Diagnostic Variables

5.6.2.1 Subroutine <*POTVORT*>

<*POTVORT*> computes the potential vorticity for each layer.

5.6.2.2 Subroutine <*VERTVEL*>

<*VERTVEL*> computes the vertical velocity for each layer.

5.6.2.3 Subroutine <*STREAM*>

<*STREAM*> computes the horizontal mass transport stream function by a line-relaxation method.

5.6.2.4 Subroutine <*ZONAL*>

<*ZONAL*> computes the vertical overturning stream function by a line-relaxation method.

5.6.3 Further Routines

5.6.3.1 Subroutine <*TOUCHLW*> & Entry <*TOUCHUP*>

<*TOUCHLW*> and <*TOUCHUP*> compute the indices of the next upper physical layer for each horizontal grid box within a given layer. They use a critical number *EPSHGT* to detect a physical layer.

5.6.3.2 Subroutine <*CONTLW*> & Entry <*CONTUP*>

<*CONTLW*> and <*CONTUP*> work similarly to <*TOUCHLW*> and <*TOUCHUP*>, but use the flag field *IFLG* to detect a physically existing layer.

5.6.3.3 Subroutine <*MASSCON*>

<*MASSCON*> computes the mass content of the ocean.

5.6.3.4 Subroutine <*HEATCON*>

<*HEATCON*> computes the heat content of the ocean.

5.6.3.5 Subroutine <*SALTCON*>

<*SALTCON*> computes the salt content of the ocean.

5.6.3.6 Subroutine <*HGTCOR*>

<*HGTCOR*> removes residual negative layer thicknesses.

5.6.3.7 Subroutine <*ADJUST*>

<*ADJUST*> optionally corrects for the mass, heat and salt content when one of the budgets is not closed.

5.6.3.8 Subroutine <*ZEROLAY*>

<*ZEROLAY*> fills values in cells that have physically vanished. This routine is only needed for cosmetics to provide some numbers for a zero-layer.

5.6.4 Boundary Conditions

The basic idea behind the model's structure is that boundary conditions do 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.

5.6.4.1 Subroutine <*BNDH1*>, Entry <*BNDU*1> & Entry <*BNDV1*>

There are three different versions of that routines that provide the boundary conditions along the model margin. <*BNDH1*> is used for scalar quantities on scalar grid points, <*BNDU1*> is used for scalar quantities on vector grid points and <*BNDV1*> is used for vector quantities on vector grid points. Note that the extension '1' is taken to mean that only arrays containing 1 layer are treated by these routines.

5.6.4.2 Subroutine <*BNDHNZ*>, Entry <*BNDUNZ*> & Entry <*BNDVNZ*>

These routines are similar to above, however, full model arrays are treated with these routines.

5.6.4.3 Subroutine <*UVPOLE1*> & Entry <*SPOLE1*>

These routines are called from <*BND...*> and compute the northern boundary values 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.

5.6.4.4 Subroutine <*UVPOLEN*> & Entry <*SPOLEN*>

Similar to above, but these are used for full model arrays.

5.6.5 Routines for Listing

5.6.5.1 Subroutine <*PRINT*>

<*PRINT*> 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.

5.6.5.2 Subroutine <*DRUCKV*> & Entry <*DRUCKH*>

These routines print vector or scalar fields onto 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.

5.6.5.3 Subroutine <*EXTRAV*> & Entry <*EXTRAH*>

Same as before, but able to print all grid points within a portion of the model domain.

5.6.6 Routines for Model I/O

5.6.6.1 Subroutine <*WRITEM*>

<*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 ASCII file.

5.6.6.2 ENTRY <*READM*> of Subroutine <*WRITEM*>

<*READM*> reads the first part of the data from [*OCDAT*] that have been created by <*WRITEM*>. If the file is an ASCII version of [*OCDAT*], <*READM*> also uses formatted I/O.

5.6.6.3 ENTRY <*READF*> of Subroutine <*WRITEM*>

<*READF*> reads the second part of the data from [*OCDAT*] that have been created by <*WRITEM*>. If the file is an ASCII version of [*OCDAT*], <*READF*> also uses formatted I/O.

5.7 COUPLING INTERFACE: [*ocestop.f*]

The following described routines allow an easy use of the model either in coupled mode or with simple forcing with atmospheric data that are not supported by the model organization. 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 analyze 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.

5.7.1 Data Transfer into the Ocean Model

I**mportant:** The data passed into the subsequent routines must already be defined on the ocean model grid and must have the same dimensions as the grid!

5.7.1.1 Subroutine <*AOPME*>

<*AOPME*> writes the salt flux into the common data area.

5.7.1.2 Subroutine <*AOQFLX*>

<*AOQFLX*> writes the heat flux and the solar radiation into the common data area.

5.7.1.3 Subroutine <*AOTAU*>

<*AOTAU*> writes the wind stress components into the common data area.

5.7.1.4 Subroutine <*AOUSTAR*>

<*AOUSTAR*> writes the turbulent kinetic energy u_*^3 into the common data area.

5.7.1.5 Subroutine <*AOICE*>

<*AOICE*> writes the snow surface and snow-ice interface temperature into the common data area.

5.7.1.6 Subroutine <*FOR2MOD*>

<*FOR2MOD*> interpolates data from an arbitary 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. <*FOR2MOD*> is also able to fill gaps in the data source using a Laplacian filter.

5.7.2 Data Transfer out of Ocean Model

The output data of the following routines are defined on the ocean model grid.

5.7.2.1 Subroutine <*OAGRID*>

<*OAGRID*> provides the land-sea mask on the x- and y- coordinates of the ocean model to be used, e.g. for interpolation between two different model or data grids.

5.7.2.2 Subroutine <*OASST*>

<*OASST*> reads the model sea surface temperature and the $\frac{\partial Q}{\partial t}$ out of the common data area.

5.7.2.3 Subroutine <*OAICE*>

<*OAICE*> reads the ice thickness and ice compactness out of the common data area, e.g. for later heat flux computations in another model.

5.7.2.4 Subroutine <*OAPMEF*>

<*OAPMEF*> reads the conversion factor between salt flux and fresh water flux out of the common data area. This conversion factor is variable as salinity and in situ density enter the corresponding formula.

5.7.3 Internal Data Transfer

The following routines supply the model with data from the common data area which is /*JMOCOUP*/, however, they should not be used to feed the model with data.

5.7.3.1 Subroutine <*HEATMOD*>

<*HEATMOD*> extracts the total surface heat flux from the common data area. See <*SURFLUX*>.

5.7.3.2 Subroutine <*SOLMOD*>

<*SOLMOD*> extracts the solar radiation from the common data area. See <*SURFLUX*>.

5.7.3.3 Subroutine <*PMEMOD*>

<*PMEMOD*> extracts the fresh water flux from the common data area. See <*SURFLUX*>.

5.7.3.4 Subroutine <*TAUMOD*>

<*TAUMOD*> extracts the wind stress components from the common data area. See <*OCESTEP*>.

5.7.3.5 Subroutine <*MIXMOD*>

<*MIXMOD*> extracts the turbulent kinetic energy for the mixed layer from the common data area. See <*MIXEXP*> and <*MIXIMP*>.

5.7.3.6 Subroutine <*ICEMOD*>

<*ICEMOD*> extracts the snow surface and snow-ice interface temperature from the common data area. See <*SURFLUX*>.

5.8 THE TRACER MODEL: TPYC

There exists a tracer model as separately running code. It basically consists of the routine <*TRACER*> and further software for I/O. Instead of being forced with flow data that are computed in the same code as in OPYC, this model reads the flow field from the file [*BASIC*] that can be created optionally with OPYC. [*BASIC*] must contain flow data from a run that covers at least one year of integration. In this case the file contains the model quantities for layer thickness, mass flux and mass exchange rates as monthly values. Based on these data TPYC can run for thousands of years to compute the distribution of passive tracers. A documentation of this model is in preparation.

6. HOW TO USE OPYC

6.1 SETTING UP OPYC

In this section we describe how to adjust the code for a certain purpose. OPYC is not developed for only 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 type of computers. The model was tested on work-stations like SUN, IBM-RISC or HP on mini-computers like Micro VAX, and on main-frame computers like IBM3090, Cyber205, NEC-, Fujitsu- or CRAY-type computers.

6.1.1 The Code Preprocessor

The code is written for scalar, vector and parallel computers with shared memory. The optimization was primarily done for CRAY-type computers. Since the same code cannot be used for all types of computers, control statements have been included that are used by [*precomp.f*] either to activate or to 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 work station. 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, Vector or Parallel:** Depending on the machine, DO-loops are written differently to help compilers to obtain the best performance. The keywords are '_SCALAR_', '_VECTOR_' and '_AUTASK_'.

- **CRAY or Not-CRAY Computer:** Depending on the type of computer it is sometimes necessary to specify statements. Therefore, OPYC distinguishes between a CRAY-type and other-type of computer. The keywords for vector statements are '_CRAVEC_' and '_NOTVEC_'. For parallel statements the keywords are '_CRATSK_' and '_NOTTSK_'. Since FORTRAN precompiler statements differ between computers, the precompiler statements mainly for tasks like DO-loop unrolling or parallelization are marked with '_CRADIR_' for the CRAY. For other precompilers than 'fpp' and 'fmp' new directives can be added without changing the philosophy of [*precomp.f*]. However, [*precomp.f*] must then be extended.

- **Choice of the Grid:** The model can either define the grid by itself, or to use predefined grids as for the T21, T42 and T106 Gaussian grid. 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, '_T42MOD_' for the T42-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*] can be executed interactively or used within the script [*setup*] or the batch job [*setup.job*]. 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.

6.1.2 Defining the Grid

If one of the T21, T42, or T106 grids is chosen, the work is already done by [*precomp.f*]. If the grid is self-defined, then the corresponding switch in [*precomp.f*] has to be set accordingly. The model margins are then given 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 $Y(NY-1) = YGO$.

6.1.3 Specification of the Dimensions

For the case of a constant grid spacing Δx and Δy, the following procedure determines *NX* and *NY*:

$$NX = \frac{XGR - XGL}{\Delta x} + 2 \qquad (6.1.3.1)$$

$$NY = \frac{YGO - YGU}{\Delta y} + 2 \qquad (6.1.3.2)$$

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, so it should not be chosen too big for Cache-type computers.

6.1.4 Defining a Focus

A focus here is an area of enhanced resolution. The code allows for any definition of the x- and y-coordinates as long as the values are monotonically increasing. 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 side and the right hand side 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 last section). 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:

$$NX = \frac{XGR - XGL}{\Delta x} + 2 + 2*IDENS \tag{6.1.4.1}$$

$$NY = \frac{YGR - YGL}{\Delta y} + 2 + 2*JDENS \tag{6.1.4.2}$$

where 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:

$$ICENTER = \frac{XGC - XGL}{\Delta x} + 1 + IDENS \tag{6.1.4.3}$$

$$JCENTER = \frac{YGC - YGU}{\Delta y} + 1 + JDENS \tag{6.1.4.4}$$

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

$$IBAND = \frac{XGR - XGL}{2\Delta X} \tag{6.1.4.5}$$

$$JBAND = \frac{YGO - YGU}{2\Delta Y} \qquad (6.1.4.6)$$

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 area and in the focus center.

The focus is switched off, if $IDENS = JDENS = 0$, $IBAND = JBAND = 0$, and $XCOM = YCOM = 1$.

6.1.5 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 *JSW*1-*JSW*20 one is able to treat the standard problems like closing the Panama channel, etc. These switches activate predefined 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 divisions by the number of sea points. Therefore, the first run to initialize the model should be performed with *JSW*1-*JSW*20 and *KILL*1*I*-*KILL*6*J* 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.

6.1.6 Selecting the Forcing Data

As explained later in detail, the model offers the choice of either the COADS data or ECMWF analysis to force the model. Two topography data sets are offered, one with 1 degree and the other with 5 minutes resolution. 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 meaningful data, a global model should use the ECMWF data. The choice of the topography data set depends on whether the roughest possible topography should be used in an experiment. The chosen data must be defined as switches in the code (see Table 19) as well as switches in the batch job [*model.job*] that runs the model. The data format in general is '20I4'. For details see references given in section 6.3.3.1.

6.1.7 Defining the Output

The model delivers the following files:

1. [***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 *ISW*31). During the start, the model is able to read the binary or the formatted version of [*OCDAT*]. The data format is recognized automatically.

2. [***FORCES***] contains all monthly mean atmospheric forcing data as well as the monthly mean SST on the model grid and the annual means surface salinity. 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.job*]). Depending on *ISW*42 [*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.

3. [***OUTLIST***] is the model's protocol (see later).

4. [***PPSF_ALL***] is the history file which is created depending on the switch *ISW*17. Other files with an initial [*PPSF_...*] are created if switch *ISW*20 is set (for details see Quick-Look System).

6.2 EXAMPLES FOR SPECIFIC MODEL LAYOUTS

In this section we demonstrate how simple it is to setup OPYC for any geometry and forcing. Starting with any model version of OPYC, the code preprocessor [*precomp.f*] has to be called by [*setup.job*] 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. All values listed with a '~' are valid for large scale purposes. High resolution grids require some adjustments.

6.2.1 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-, T42-, or T106-grids has to be enabled by [*setup.job*].

2. Ensure that the following dimensions have been chosen:

- **T21-Grid**: *NX*=66 and *NY*=33
- **T42-Grid**: *NX*=130 and *NY*=63.
- **T106-Grid**: *NX*=322 and *NY*=152.

3. Open cyclic boundary conditions must be enabled by *ISW*15=1.
4. It is recommended to set *ISW*1=1, *ISW*5=1, *ISW*8=1, *ISW*13=1, *ISW*18=0, *ISW*19=0, *ISW*41=0.
5. Switch full physics on, but use *ISW*33=0.
6. Ensure *ISW*48=0.
7. Ensure that Eulerian angles are switched off.
8. Choose optionally an open North Pole with *ISW*28=1 or *ISW*28=2.

Note that the variables for defining the model margins *XGL*, *XGR*, *YGU* and *YGO* are not used when one of the T21, T42 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*.

6.2.2 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 *ISW*15 for the cyclic boundary condition and *ISW*28 must be chosen according to the application.

6.2.3 Box-Model

A model that is driven by artificial data can be realized by setting *ISW*1=0, *ISW*5=0, *ISW*8=0 and *ISW*41=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*.

6.3 RUNNING OPYC

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

1. **[*setup.job*]** 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. A script [*setup*] is available that can be used interactively.

2. [***make.job***] generates the object files for [*ocestep.f*], [*oceinit.f*], [*ocepost.f*], [*ocestop.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.job***] 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. [***read.job*]** reads files from a tape system required to run [*model.job*] and after competition submits [*model.job*]. This script must be adjusted to any application.

5. [***tape.job*]** is optionally submitted by [*model.job*] and saves data on a tape system. This script must be adjusted to any application.

6. [***plot.job*]** is optionally submitted by [*model.job*] and plots the data written on the postprocessing files. Depending on the switches various files are plotted. [*plot.job*] uses [*ocepict.f*] and [*oceplot.f*] as objects.

6.3.1 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 first step, [*setup.job*] creates the FORTRAN code.The batch job [*model.job*] requires object files of [*ocestep.f*], [*oceinit.f*], [*ocepost.f*], [*ocestop.f*] and optionally [*ocemods.f*]. These files are created by [*make.job*]. After having adjusted all control parameters in [*ocemain.f*] the source code is ready for compilation. In case any modification of some routine is necessary, save it on [*ocemods.f*].

6.3.2 Start from Scratch

There are several steps to take 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 *ISW*11=1 in [*ocemain.f*] when starting [*model.job*] for the first time. Since the x- and y-coordinates as well as the topography are initialized first, *ISW*11 allows the model to stop after a short CPU-time. Before that, it saves [*PPSF_TOP*] if *SAVEPOST*=1 in [*model.job*]. Ensure that *TOPFORC* in [*model.job*] is consistently defined with *ISW*38. After having set *PLOTTOP*=1 in [*plot.job*] you are able to plot the topography and coastline geometry. 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. **Model Initialization:** After setting *ISW*11=0, *SAVFORC*=1, *UPDATED*=1 and having chosen

VELFORC, *TEMFORC* and *SSTFORC* consistent with *ISW*30, *ISW*43 and *ISW*47, run [*model.job*]. After completion it delivers [*FORCES*] and [*OCDAT*] as well as all quick-look files. Ensure a short run by setting *MAXREP*=1.

3. **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.

4. **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.

6.3.3 Data Preprocessing

In order to avoid extra work 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.

6.3.3.1 Data Bank

Next we briefly outline 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.

6.3.3.1.1 File [*TOP1DEG*]

[*TOP1DEG*] contains the so-called Scripps topography on a $1° \times 1°$ 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 1 J=1,180
      READ(*,'(1X,12(I5,1X))') (IDEPTH(I),I=1,360)
      DO 1 I=1,360
      DEPTH(I,J)=HOTAL-IDEPTH(I)
    1 CONTINUE
```

6.3.3.1.2 File [*TOP5MIN*]

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

6.3.3.1.3 File [*SALTEMP*]

[*SALTEMP*] contains the annual mean temperature and salinity of Levitus for the global ocean on a $1° \times 1°$ 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 1 I=1,360
      DO 1 J=1,180
      READ(*,'(32I4)') (ISALT(K),K=1,32)
      READ(*,'(32I4)') (ITEMP(K),K=1,32)
      DO 1 I=1,32
      TEMP(I,J)=ITEMP(K)/100.+273.16
      SALT(I,J)=ISALT(K)/100.
    1 CONTINUE
```

6.3.3.1.4 File [*SSSALT*]

[*SSSALT*] contains the annual mean sea surface salinity from Levitus on a $1° \times 1°$ grid. It is read by <*SEASALT*> with the following simplified statements:

```
      DIMENSION ISSS(360),SSS(360,180)
      DO 1 J=1,180
      READ(*,'(20I4)') (ISSS(I),I=1,360)
      DO 1 I=1,360
      SSS(I,J)=ISSS(I)/100.+35.
    1 CONTINUE
```

6.3.3.1.5 File [*SSTGLOB*]

[*SSTGLOB*] contains a merged data set from the COADS and the Reynolds data set on a $2° \times 2°$ 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 1 MONTH=1,12
      DO 1 J=1,90
      READ(*,'(20I4)') (ISST(I),I=1,180)
      DO 1 I=1,180
      SST(I,J,MONTH)=ISST(I)/100.+273.16
    1 CONTINUE
```

6.3.3.1.6 File [*AMICLIM*]

[*AMICLIM*] contains a 10 year average of the AMIP SST data set (1979-1988) on a $2° \times 2°$ grid. It is read by <*AMIPSST*> from the local file [*SSTEMP*] with the following simplified statements:

```
      DIMENSION ISST(180),SST(180,91,12)
      DO 1 MONTH=1,12
      DO 1 J=1,91
      READ(*,'(20I4)') (ISST(I),I=1,180)
      DO 1 I=1,180
      SST(I,J,MONTH)=ISST(I)/100.+273.16
    1 CONTINUE
```

6.3.3.1.7 File [*RAIN1DEG*]

[*RAIN1DEG*] contains the precipitation data set of Legates & Willmott (1989). However, it has been interpolated to a $1° \times 1°$ grid from the original $0.5° \times 0.5°$ grid. It is read by <*LEGATES*> from the local file [*MONRAIN*] with the following simplified statements:

```
      DIMENSION IRAIN(360),RAIN(360,180,12)
      DO 1 MONTH=1,12
      DO 1 J=1,180
      READ(*,'(20I4)') (IRAIN(I),I=1,360)
      DO 1 I=1,360
      RAIN(I,J,MONTH)=IRAIN(I)/(1000.*2592000.)
    1 CONTINUE
```

6.3.3.1.8 File [*COVER*]

[*COVER*] contains the monthly mean fractional cloud cover from COADS on a $2° \times 2°$ grid. It is read by <*CUMULUS*> with the following simplified statements:

```
      DIMENSION ICLOUDS(180),CLOUDS(180,90,12)
      DO 1 MONTH=1,12
      DO 1 J=1,90
      READ(*,'(20I4)') (ICLOUDS(I),I=1,180)
      DO 1 I=1,180
      CLOUDS(I,J,MONTH)=ICLOUDS(I)/80.
```

```
1 CONTINUE
```

6.3.3.1.9 File [*WETNESS*]

[*WETNESS*] contains the monthly mean relative humidity from COADS on a $2° \times 2°$ grid. It is read by <*VAPOR*> with the following simplified statements:

```
DIMENSION IVAPOR(180),VAPOR(180,90,12)
DO 1 MONTH=1,12
DO 1 J=1,90
READ(*,'(20I4)') (IVAPOR(I),I=1,180)
DO 1 I=1,180
VAPOR(I,J,MONTH)=IVAPOR(I)/1000.
1 CONTINUE
```

6.3.3.1.10 File [*PRESURF*]

[*PRESURF*] contains the monthly mean sea level pressure from COADS on a $2° \times 2°$ grid. It is in preparation to use this file.

6.3.3.1.11 File [*UVGLOB*]

[*UVGLOB*] contains the monthly mean surface wind stress and its annual mean from Hellerman and Rosenstein as pseudo wind stresses on a $2° \times 2°$ grid. The first block of data is the annual mean, the second one then 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 1 MONTH=1,13
DO 1 J=1,90
READ(*,'(20I4)') (ITAUX(I),I=1,180)
READ(*,'(20I4)') (ITAUY(I),I=1,180)
DO 1 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)
1 CONTINUE
```

6.3.3.1.12 File [*UVABS*]

[*UVABS*] contains the monthly mean wind speed from the COADS on a $2° \times 2°$ grid. It is read by <*ABSOL*> with the following simplified statements:

```
DIMENSION IABSOL(180),UVABSOL(180,90,12)
DO 1 MONTH=1,12
DO 1 J=1,90
```

```
      READ(*,'(20I4)') (IABSOL(I),I=1,180)
      DO 1 I=1,180
      UVABSOL(I,J,MONTH)=IABSOL(I)/10.
    1 CONTINUE
```

6.3.3.1.13 File [*UVDEV*]

[*UVDEV*] contains the monthly mean wind speed standard deviation from the COADS on a $2° \times 2°$ grid. It is read by <*VARIAN*> with the following simplified statements:

```
      DIMENSION IVARIAN(180),UVARIAN(180,90,12)
      DO 1 MONTH=1,12
      DO 1 J=1,90
      READ(*,'(20I4)') (IVARIAN(I),I=1,180)
      DO 1 I=1,180
      UVARIAN(I,J,MONTH)=IVARIAN(I)/100.
    1 CONTINUE
```

6.3.3.1.14 File [*TAUXY*]

[*TAUXY*] contains the surface wind stress from the ECMWF analysis on a $2.5° \times 2.5°$ grid. Since ECMWF data are useless for forcing an ocean model at low latitudes (too weak winds) this data set has been blended with [*UVGLOB*] within a $20°$ band north and south of the equator. Also, the wind speeds in mid-latitudes are stronger by 10% compared with the COADS, only 81% of the ECMWF stresses are taken. It is read by <*ECTAUXY*> from the local file [*UVGLOB*] with the following simplified statements:

```
      DIMENSION ITAUX(144),ITAUY(144),TAUX(144,72,12),TAUY(144,72,12)
      DO 1 MONTH=1,12
      DO 1 J=1,72
      READ(*,'(20I4)') (ITAUX(I),I=1,144)
      READ(*,'(20I4)') (ITAUY(I),I=1,144)
      DO 1 I=1,144
      TAUX(I,J,MONTH)=(ITAUX(I)-5000.)/5000.
      TAUY(I,J,MONTH)=(ITAUY(I)-5000.)/5000.
    1 CONTINUE
```

6.3.3.1.15 File [*USC*]

[*USC*] contains the monthly mean wind speed from the ECMWF analysis on a $2.5° \times 2.5°$ grid. The same comments concerning data errors also apply for this data set. It has been blended with [*UVABS*] within a $20°$ band north and south of the equator. Since the wind speeds in mid-latitudes are too strong, only 90% of it is taken in order to tune it towards the COADS. It is read by <*ECUSC*> from the local file [*UVABS*] with the following simplified statements:

```
      DIMENSION IUVABSOL(44),UVABSOL(144,72,12)
```

```
      DO 1 MONTH=1,12
      DO 1 J=1,72
      READ(*,'(20I4)') (IUVABSOL(I),I=1,144)
      DO 1 I=1,144
      UVABSOL(I,J,MONTH)=IUVABSOL(I)/500.
    1 CONTINUE
```

6.3.3.1.16 File [*STDV_USC*]

[*STDV_USC*] contains the monthly mean wind speed standard deviation from the ECMWF analysis on a $2.5° \times 2.5°$ grid. The same comments concerning data errors also apply for this data set. [*UVDEV*] is taken for the tropics. It is read by <*ECSUSC*> from the local file [*UVDEV*] with the following simplified statements:

```
      DIMENSION IVARIAN(144),UVARIAN(144,72,12)
      DO 1 MONTH=1,12
      DO 1 J=1,72
      READ(*,'(20I4)') (IVARIAN(I),I=1,144)
      DO 1 I=1,144
      UVARIAN(I,J,MONTH)=IVARIAN(I)/500.
    1 CONTINUE
```

6.3.3.1.17 File [*AIRGLOB*]

[*AIRGLOB*] contains the monthly mean surface air temperature on a $2° \times 2°$ 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 1 MONTH=1,12
      DO 1 J=1,90
      READ(*,'(20I4)') (IAIRT(I),I=1,180)
      DO 1 I=1,180
      AIRTEMP(I,J,MONTH)=IAIRT(I)/100.-60.+273.16
    1 CONTINUE
```

6.3.3.1.18 File [*TSC*]

[*TSC*] contains the surface air temperature from the ECMWF analysis on the T106-Gaussian grid from the period May 1985 to May 1990. It is read by <*ECTSC*> from the local file [*AIRTEMP*] with the following simplified statements:

```
      DIMENSION IAIRT(320),AIRT(320,160,12)
      DO 1 MONTH=1,12
      DO 1 J=1,160
      READ(*,'(20I4)') (IAIRT(I),I=1,320)
      DO 1 I=1,320
      AIRT(I,J,MONTH)=IAIRT(I)*120./10000.-75.+273.16
    1 CONTINUE
```

6.3.3.1.19 File [*T42GRID*]

[*T42GRID*] contains a predefined land-sea mask as e.g. from the ECHAM-T42. Such masks are used if *ISW*45=1 and overwrite the land-sea mask that is derived by OPYC during the topography initialization, however, [*T42GRID*] is used only if one of the non-self-defined grids is activated via [*precomp.f*]. It is read by <*GRITUNE*> from the local file [*LSMASK*] with the following simplified statements:

```
      DIMENSION IFLGT42(128,64)
      DO 1 J=1,64
      READ(*,'(128I1)') (IFLGT42(I,J),I=1,128)
    1 CONTINUE
```

6.3.3.2 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.

6.3.4 About Numerical Stability of OPYC

The CFL-number does not provide a necessary condition for numerical stability, as OPYC 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 behavior the following lines appear in the protocol [*OUTLIST*]:

```
+-------------> WAVE EQUATION                 >-+
| +-----------> MOMENTUM ADVECTION & DIFFUSION  |
| | +---------> TEMPERATURE EQUATION            |
| | | +-------> SALINITY EQUATION               +-> ITERATIONS
| | | | +-----> TRACER EQUATION                 |
| | | | | +---> ICE THICKNESS & CONCENTRATION   |
| | | | | | +-> ICE MOMENTUM                  >-+
| | | | | | |
| | | | | | |   +---> USED TIME STEP
| | | | | | |   |     +---> MEAN EXPLICIT ACCELERATION
| | | | | | |   |     |     +---> COMPUTED CHANGE OF FLUX
| | | | | | |   |     |     |     +---> COMPUTED CHANGE OF H
| | | | | | |   |     |     |     |      +---> TOTAL MOMENTUM
| | | | | | |   |     |     |     |      |
4 4 4 4 4 4 4 1.000 1716.  346.  411.   3082. 107-24.732 (11,003,061) 246 237
4 4 4 4 4 4 4 1.000 1720.  342.  404.   3090.  91 -8.900 (11,003,060) 241 261
4 4 4 4 4 4 4 1.000 1648.  371.  413.   3096. 120-45.817 (11,003,061) 268 255
4 4 4 4 4 4 4 1.000 1814.  304.  406.   3095. 103-44.600 (11,003,061) 239 292
4 4 4 4 4 4 4 1.000 1703.  386.  417.   3107. 119 -7.333 (09,129,031) 305 258
                                             |    |     |        |   |   |
                     FOUND NEGATIVE H POINTS <-+    |     +----+---+   |   |
                         MAXIMAL FOUND NEGATIVE H <-+          |       |   |
                                 (Z,X,Y) INDEX OF THAT POINT <-+       |   |
                                        NUMBER OF CELLS BEING OPENED <-+   |
                                            NUMBER OF CELLS BEING CLOSED <-+
```

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 1-7 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 *ISW*35, OPYC 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 8 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 convergent conditions are satisfied by the reduced time step.
Row 9 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 dependent, but must not exceed values of about 5000.
The following two rows 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.
Rows 14-16 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 last two rows list the number of cells that have been opened and closed per time step, which is also some measure for numerical noise.

6.3.5 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.job*] 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. 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.job*] and that *ISW*12=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 jump 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.

6.3.6 Diagnostic Output

During the model run, a number of formatted files appear that allow a quick check of the model physics. These are:

1. **[*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. **[*MONDATA*]** contains temperature and salinity as horizontal means for several depth levels. These data are printed out once a month. The file is created by <*DATMON*> which is called by <*OCESTEP*>.

3. **[*MEANS*]** contains layer averages for temperature, salinity, in situ density and potential density. These data are written out by <*STATIST*> at each time step. [*MEANS*] is called by <*OCESTEP*>.

4. **[*BASIC*]** contains data for separately running the tracer model TPYC that is basically built around the routine <*TRACER*>. TPYC obtains flow and isopycnal data from the file [*BASIC*] as monthly values. See switch *ISW*24.

5. **[*PERFORM*]** contains an overview of CPU-times from the most CPU-time consuming routines.

6. **[*BUDGET*]** contains the heat and fresh water budget as well as conservation errors for each time step. The data are written out from <*OCESTEP*>.

6.3.7 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 use a time step of a day. In this case set *MAXREP*=1 and *DTMAX*=86400., in order to isochrones 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*>.

```
      .
      PARAMETER (NX=130,NY=63)
      DIMENSION USTERN(NX,NY),TAUX(NX,NY)  ,TAUY(NX,NY)
      DIMENSION SOLAR(NX,NY) ,FACTOR(NX,NY),FRESH(NX,NY),HEAT(NX,NY)
      CALL OCEINIT(MSEC,MINUTE,MHOUR,MDAY,MONTH,MYEAR)
      OPEN(90,FILE='FORCING',FORM='UNFORMATTED')
      DO 999 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 OAPMEF (FACTOR)
      DO 1 J=1,NY
      DO 1 I=1,NX
      FRESH(I,J)=-FRESH(I,J)*FACTOR(I,J)
    1 CONTINUE
      CALL AOPME  (FRESH)
      CALL OCESTEP(MSEC,MINUTE,MHOUR,MDAY,MONTH,MYEAR)
  999 CONTINUE
      CLOSE(90)
      CALL OCEPOST
      CALL OCESTOP
      .
      .
```

6.4 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, an ASCII version can be created. In order to keep the use of tape media tolerably small, there is also a GRIB version of the PPSF. This allows the user to reduce the file size by a factor of about 4. 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 analyze the model fields.

6.4.1 Code Definitions

6.4.2 PPSF - Binary Format

The following write statements demostrate how a data block for one quantity is generated:

```
      .
      .
 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)
      .
      .
```

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 23 through Table 28.
2. ***KODES*** describes the type of section on which the quantity is written out. See also Table 29 and Table 30.
3. ***N*1** is the first dimension of the array.
4. ***N*2** is the second dimension of the array.
5. ***N*3** is the third dimension of the array. Note that this allows one 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.

Table 23 Definition of Quantity Code: Prognostic Variables

Code Number	Variable	Unit	Reference
1	layer velocity components	$kg(ms)^{-1}$	*UH, VH*
2	layer thickness	m	*HEIGTH*
3	layer potential temperature	K	*TEMP*
4	layer temperature	K	
5	layer salinity	gkg^{-1}	*SALT*
6	layer tracer concentration		*TRAC*
7	layer in situ density	kgm^{-3}	*DENSITY*
8	layer potential density	kgm^{-3}	*POTDENS*
9	layer in situ pressure	m^2s^{-2}	*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*

7. ***IFLG*** is the land-sea mask with dimension (*N*1,*N*2). 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.

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.

Table 24 Definition of Quantity Code: Level Quantities

Code Number	Variable	Unit	Reference
10	level potential density	$\mathrm{kg\,m^{-3}}$	
11	level velocity components	$\mathrm{ms^{-1}}$	*U, V*
12	level interface height	m	*HEIGTH*
13	level potential temperature	K	
14	level in situ temperature	K	
15	level salinity	$\mathrm{gkg^{-1}}$	
16	level tracer concentration		
65	surface temperature budget	$\mathrm{gs^{-1}}$	<*QHEATXY*>
66	sea surface temperature	K	*TEMP*(1,*I*,*J*)
67	mixed leyer depth	m	*HEIGTH*(1,*I*,*J*)
68	sea surface elevation	m	*HEIGTH*(1,*I*,*J*)
69	sea surface velocity components	$\mathrm{ms^{-1}}$	*U*(1,*I*,*J*), *V*(1,*I*,*J*)
70	sea surface salinity	$\mathrm{gkg^{-1}}$	*SALT*(1,*I*,*J*)

16. ***EXPERIM*** is a character string, declared by 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.

Table 25 Definition of Quantity Code: Diagnostic Variables

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*
32	chimney production rate	ms^{-1}	*CHIMVER*
33	cross isopycnal mixing	ms^{-1}	*VERTUP, VERTDN*
34	downward bubble salt flux	ms^{-1}	
38	SST error	K	
40	convective potential energy flux		
41	maximal depth of surface convection	m	*<QCONV>*
42	convective momentum flux		
43	convective heat flux	$W m^{-2}$	*<QCONV>*
44	convective salt flux	kgs^{-1}	*<QCONV>*
45	convective tracer flux	s^{-1}	*<QCONV>*
46	convective density flux	kgs^{-1}	*<QCONV>*

Table 26 Definition of Quantity Code: Forcing Variables

Code Number	Variable	Unit	Reference
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 27 Definition of Quantity Code: Flux Variables

Code Number	Variable	Unit	Reference
21	total surface heat flux	$\mathrm{W\,m^{-2}}$	*QFLUX*
2	solar radiation	$\mathrm{W\,m^{-2}}$	*QSOLAR*
23	long wave radiation	$\mathrm{W\,m^{-2}}$	*QLONG*
24	sensible heat flux	$\mathrm{W\,m^{-2}}$	*QSENSIB*
25	latent heat flux	$\mathrm{W\,m^{-2}}$	*QLATENT*
26	net fresh water -> salt flux	$\mathrm{W\,(Km)^{-2}}$	*SFLUX*
27	surface buoyancy flux	$\mathrm{m^2 s^{-3}}$	*BFLUX*
28	solar buoyancy flux	$\mathrm{m^2 s^{-3}}$	*BSOLAR*
29	friction velocity	$\mathrm{ms^{-1}}$	*USTERN*
61	drag coefficient		*DRAG*
62	transfer coefficient for sensible heat		*<VARDRAG>*
63	transfer coefficient for latent heat		*<VARDRAG>*
64	feedback coefficient	$\mathrm{W\,(Km)^{-2}}$	*DQDT*
75	vertical velocity	$\mathrm{ms^{-1}}$	*<VERTVEL>*
76	layer relative vorticity	$\mathrm{s^{-1}}$	
77	layer absolute vorticity	$\mathrm{s^{-1}}$	
78	layer potential vorticity	$\mathrm{s^{-1}}$	*CORIOL*

Table 28 Definition of Quantity Code: Transport Variables

Code Number	Variable	Unit	Reference
35	barotropic stream function	$\mathrm{kgs^{-1}}$	*PSIXY*
36	meridional overturning	$\mathrm{kgs^{-1}}$	*PSIYZ*
47	overturning in Atlantic Ocean	$\mathrm{kgs^{-1}}$	[*makepsi.f*]
48	overturning in Indic Ocean	$\mathrm{kgs^{-1}}$	[*makepsi.f*]
49	overturning in Pacific Ocean	$\mathrm{kgs^{-1}}$	[*makepsi.f*]
97	barotropic zonal heat transport	W	
98	baroclinic zonal heat transport	W	

Table 28 Definition of Quantity Code: Transport Variables

Code Number	Variable	Unit	Reference
99	barotropic zonal salt transport	kgs^{-1}	
100	baroclinic zonal salt transport	kgs^{-1}	
101	barotropic zonal tracer transport		
102	baroclinic zonal tracer transport		
103	barotropic meridional heat transport	W	
104	baroclinic meridional heat transport	W	
105	barotropic meridional salt transport	kgs^{-1}	
106	baroclinic meridional salt transport	kgs^{-1}	
107	barotropic meridional tracer transport		
108	baroclinic meridional tracer transport		

Table 29 Definition of Section Code: Part I

Code Number	Section Type	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*>

Table 29 Definition of Section Code: Part I

Code Number	Section Type	Reference
17	monthly means along line section in y-direction	<*PPSFOUT*>
18	monthly means along line section in z-direction	<*PPSFOUT*>

Table 30 Definition of Section Code: Part II

Code Number	Section Type	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*>

6.4.3 PPSF - ASCII Format

Except the use of formatted I/O, the layout of the data block in ASCII format is the same as for the binary version. See [*makefor.f*] and [*makebin.f*] for details.

6.4.4 PPSF-GRIB Format

The underlying routines are those used for the GRIB-code (Edition 0). However, the sequence of data is the same as in the binary format. For details see [*rwgrib.c*] and [*gribcode.f*].

6.4.5 List of PPSF-Files

6.4.5.1 The Quicklook 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.

6.4.5.1.1 File [*PPSF_SUR*]

[*PPSF_SUR*] contains surface quantities for velocity, sea level, mixed layer depth, temperature and salinity.

6.4.5.1.2 File [*PPSF_ICE*]

[*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.

6.4.5.1.3 File [*PPSF_BAS*]

[*PPSF_BAS*] contains the momentum fluxes, layer thicknesses temperature, salinity and tracer concentration in the original model coordinates. This file can be used by [*makepsi.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.

6.4.5.1.4 File [*PPSF_LAY*]

[*PPSF_LAY*] contains the flow, temperature, salinity and tracer concentration of the layer *KSEC* (see Table 13).

6.4.5.1.5 File [*PPSF_SEC*]

[*PPSF_SEC*] contains the flow, temperature, salinity, tracer concentration and vertical velocity on those depth levels that are defined by *VSEC* (see Table 13).

6.4.5.1.6 File [*PPSF_VER*]

[*PPSF_VER*] contains xz- and yz-sections of temperature, salinity, tracer concentration, 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 (see Table 13).

6.4.5.1.7 File [*PPSF_FLX*]

[*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. Depending on *ISW*48, it also can contain the long term mean heat and fresh water flux.

6.4.5.1.8 File [*PPSF_FOR*]

[*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.

6.4.5.1.9 File [*PPSF_PSI*]

[*PPSF_PSI*] contains the horizontal mass transport stream function and the vertical overturning stream function when *ISW*23 is set.

6.4.5.1.10 File [*PPSF_TOP*]

[*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.

6.4.5.1.11 File [*PPSF_INI*]

[*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.

6.4.5.2 The History File [*PPSF_ALL*]

The purpose of the history file is to accumulate model quantities that are written out with some frequency but in unsorted order. Which quantities are written out by <*OCEPST1*>, <*OCEPST*2>, <*OCEPST*3> and <*OCEPST4*> depends on the particular model experiment. With the help of the postprocessing output routines (see section 5.5) it can be decided which quantities are saved on the history file, and the frequency of storing them.

6.4.6 Utilities for Data Analysis

The following routines allow one to manipulate the contents of a PPSF-file and to perform mathematical operations. See also [*help.doc*] for further information.

6.4.6.1 Utilities for Reordering

6.4.6.1.1 Script [*append*]

[*append*] merges a sequence of input files and delivers one output file. See also [*append.f*].

6.4.6.1.2 Script [*extract*]

[*extract*] extracts blocks of data from an input file by defining the first block number, the last one and an increment. See also [*extract.f*].

6.4.6.1.3 Script [*extime*]

[*extime*] extracts a Hovmueller-diagram data out of an input file that contains a sequence of the quantity ordered in time. The location of the section is defined by one parameter. See also [*extime.f*].

6.4.6.1.4 Script [*filter*]

[*filter*] extracts all data blocks with a prescribed quantity code *KODEQ*, section code *KODES* and location code *KODEL* from an input file. See also [*process.f*].

6.4.6.1.5 Script [*point*]

[*point*] extracts a given grid point from a sequence of data blocks and creates an output file that contains one vector. The number of vector elements corresponds to the number of data blocks. See also [*point.f*].

6.4.6.1.6 Script [*print*]

[*print*] creates a listing of the arrays for each data block from an input file. The layout of the listing is similar to that which is used for [*OUTLIST*].

6.4.6.1.7 Script [*process*]

[*process*] extracts a maximum of ten quantities out of a sequence of input files and delivers each quantity to its own output file. See also [*process.f*].

6.4.6.1.8 Script [*read_tape*]

[*read_tape*] reads a number of subsequent files from a cartridge.

6.4.6.1.9 Script [*write_tape*]

[*write_tape*] writes a number of files sequentially to a cartridge.

6.4.6.2 Utilities for Manipulations

6.4.6.2.1 Script [*addc*]

[*addc*] adds a constant value to all quantities of an input file and delivers the result to an output file. See also [*add.f*].

6.4.6.2.2 Script [*differ*]

[*differ*] computes the difference for each pair of data blocks of two input files and delivers one output file. Note that the section code is set to its negative value to mark that a mathematical operation was carried out. This feature is used by the plot system. See also [*differ.f*].

6.4.6.2.3 Script [*intxy*]

[*intxy*] computes the integral over a xy-section and delivers one value for each data block to an output file. See also [*intxy.f*].

6.4.6.2.4 Script [*lev2mod*]

[*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. See also [*lev2mod.f*].

6.4.6.2.5 Script [*makepsi*]

[*makepsi*] 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. See also [*makepsi.f*].

6.4.6.2.6 Script [*mean*]

[*mean*] computes the mean from a sequence of data blocks from an input file and delivers a data block that contains the average onto an output file. See also [*mean.f*].

6.4.6.2.7 Script [*plot*]

[*plot*] creates plots of all quantities from an input file. This is an interactive version of [*plot.job*]. See also [*ocepict.f*] and [*oceplot.f*].

6.4.6.2.8 Script [*product*]

[*product*] computes the product of two corresponding data blocks from two input files and delivers the result to an output file. See also [*product.f*].

6.4.6.2.9 Script [*stdv*]

[*stdv*] computes the standard deviation of a quantity that is ordered as sequential data blocks from an input file and delivers the result to an output file. See also [*stdv.f*].

6.4.6.2.10 Script [*sumxy*]

[*sumxy*] computes the sum of a quantity that appears as sequential data blocks from an input file and delivers the result to an output file. See also [*sumxy.f*].

6.4.6.2.11 Script [*triade*]

[*triade*] computes the triade $d = a + b*c$ of corresponding data blocks of three input files and delivers the result to one output file. See also [*triade.f*].

6.4.6.3 Utilities for Format Conversion

6.4.6.3.1 Script [*ppsf2grib*]

[*ppsf2grib*] converts the binary PPSF-format to the GRIB-format. See also [*rwgrib.c*] and [*gribcode.f*].

6.4.6.3.2 Script [*grib2ppsf*]

[*grib2ppsf*] converts the GRIB-code into the binary PPSF-format. See also [*rwgrib.c*] and [*gribcode.f*].

6.4.6.3.3 Script [*makefor*]

[*makefor*] converts the binary PPSF-format into the formatted PPSF-format. Note that accuracy is lost due to this transformation. See also [*makefor.f*].

6.4.6.3.4 Script [*makebin*]

[*makebin*] converts the formatted PPSF-format into the binary PPSF-format. See also [*makebin.f*].

6.4.6.3.5 Script [*spr2dpr*]

[*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. See also [*spr2dpr.f*].

6.4.6.3.6 Script [*dpr2spr*]

[*dpr2spr*] converts a binary PPSF-format that has been created with double precision into a binary PPSF-format with single precision. See also [*dpr2spr.f*].

6.5 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 intension is for the plot system without additional changes to produce any chosen picture unless the subsequently listed control parameters have been specified by a new model layout.

6.5.1 The Plot Program

Similar to the ocean model, the plot program is split up into a simple main PROGRAM and several BLOCK DATA routines on [*ocepict.f*] and the executable program on [*oceplot.f*]. The object code of [*oceplot.f*] is created by [*make.job*]. Currently, the whole plot system requires the plot library [*jmolib.f*], the [*spps.f*] from the NCAR-Graphics and GKS software. [*plot.job*] loads all the object files and runs the plot program. The input files can be specified with switches. It can be one of the PPSF-files.

6.5.1.1 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.

6.5.1.2 Data Block |PLOTPAR|

The parameters of Table 31 control the overall behavior 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 32 define in more detail the size, content and labels used for plotting. Finally, Table 32 allows to define the Eulerian anglesessentially to add lines of geographical longitude and latitude on each frame.

Table 31 Definition of Main Switches

Variable	Value	Default	Meaning	Reference
NPIC	any ≥ 1	999	number of pictures to be plotted	*/JMOPLT1/*
NCOL	-1, 0, 1, 2	-1	switch for black &white or colour 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/*
NFORM	0, 1	0	switch GKS file format	*/JMOPLT1/*
NQUAL	-3, -2, -1, 1, 2, 3	-1	switch for quality of contour plots	*/JMOPLT1/*
MUE	any ≥ 1	1	x-refinement factor for contours	*/JMOPLT1/*
NUE	any ≥ 1	1	y-refinement factor for contours	*/JMOPLT1/*

Table 31 Definition of Main Switches

Variable	Value	Default	Meaning	Reference
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/*
KODEX	any of codes	0	choice which quantity code is plotted	*/JMOPLT1/*
KENNEX	any of codes	0	choice which section code is plotted	*/JMOPLT1/*
NLOCEX	any code	-9999	choice which location code is plotted	*/JMOPLT1/*
NFILM	0, 1	0	switch from sheet to movy mode	*/JMOPLT1/*
NOFFSET	any	0	time-offset to model time counter	*/JMOPLT1/*

Table 32 Definition of Margins

Variable	Value	Default	Unit	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 > Y*LOW*	deg	90	location of upper margin	*/JMOPLT2/*
NXS	any ≥ 2		-13	number of labels for x-direction	*/JMOPLT2/*
*LX*0	any		0	first label for x-direction	*/JMOPLT2/*
LX	any		360	label difference for x-direction	*/JMOPLT2/*
NYS	any ≥ 2		-7	number of labels for y-direction	*/JMOPLT2/*
*LY*0	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 33 Definition of Eulerian Angles

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 new longitude	*/JMOPLT6/*

6.5.1.3 Data Block |*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.

6.5.1.4 Data Block |*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.

6.5.1.5 Data Block |*UNITS*|

|*UNITS*| contains the unit definition 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.

6.5.1.6 Data Block |*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 Table 34).

Table 34 Definition of Layout Parameters

Variable	Value	Meaning	Reference
IROT	0, 1	switch for rotation of the colour table	*/JMOIND/*
KEYCOL	1, 2, 3, 4	switch to control the margins of colour table	*/JMOIND/*
IPOL	0, 1, 2, 3	switch to select a projection	*/JMOIND/*
ICON	any ≥ 0	switch for contours in colour plots	*/JMOIND/*

6.5.1.7 Data Block |*MINMAXS*|

|*MINMAXS*| contains the definitions for the lowest and highest level that is plotted by contours or colored areas (see also Table 35).

Table 35 Definition of Interval Parameters

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/*

6.5.1.8 Data Block |*CVARIAB*|

|*CVARIAB*| allows one to specify nonequidistant contour or color intervals.

6.5.1.9 Data Block |*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 36).

Table 36 Definition of Scaling Parameters

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/*

6.5.1.10 Executing Program OCEPLOT

OCEPLOT uses the plot library [*jmolib.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.

6.5.2 The Underlying Library JMOLIB

[*jmolib.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 on-line manual available that describes each routine. The library uses the CALCOMP-routines. A converter between these routines and the SPPS-routines of NCAR-graphics is added. The plot software finally requires only a GKS-system.

6.5.3 Example Plots

Figure 8 shows a typical example for a vector plot. In order to improve the appearance, each second arrow is taken out by default. Here the land-sea mask of the model is used to mark the continents. Figure 9 is a standard example for a contour plot on a xy-section. Figure 10 is an example for the layout of a vertical section. Vertical sections are plotted twice; once for the upper 500 m and another 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. Figure 11 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 Figure 9. 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 as shown in Figure 11 for the Arctic ice flow. Finally, Figure 11 gives an example showing the surface flow of a North Atlantic - Arctic model, which uses a 1^o resolution and a focus of 0.5^o in the north-east of the Atlantic.

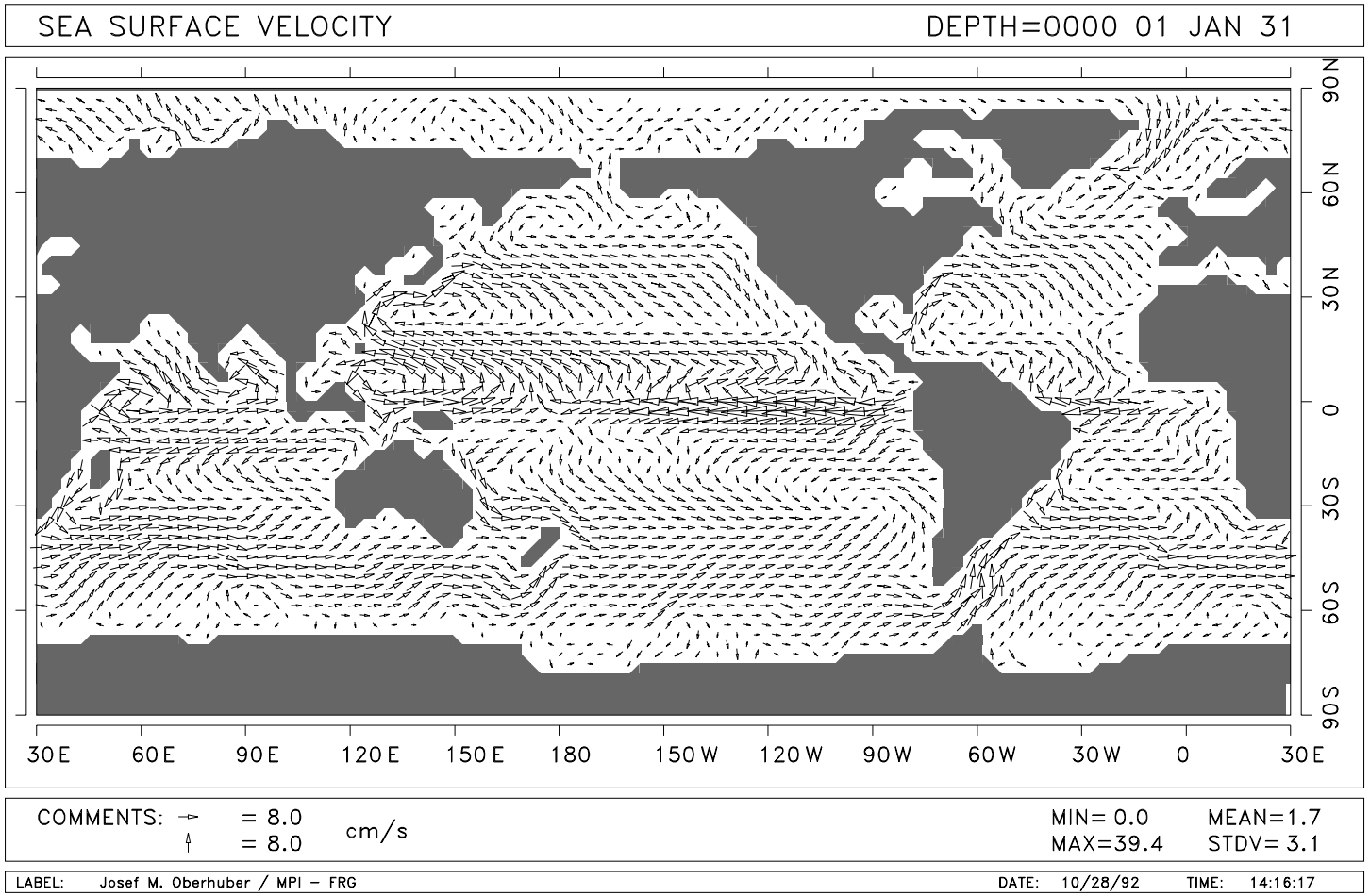

Figure 8 Surface flow of OPYC / T42.

Example showing a vector plot for the surface flow of OPYC/T42. The quantity code is *KODEQ*=69, the section code is *KODES*=1 for xy-section and the location code is *KODEL*=0 for zero depth.

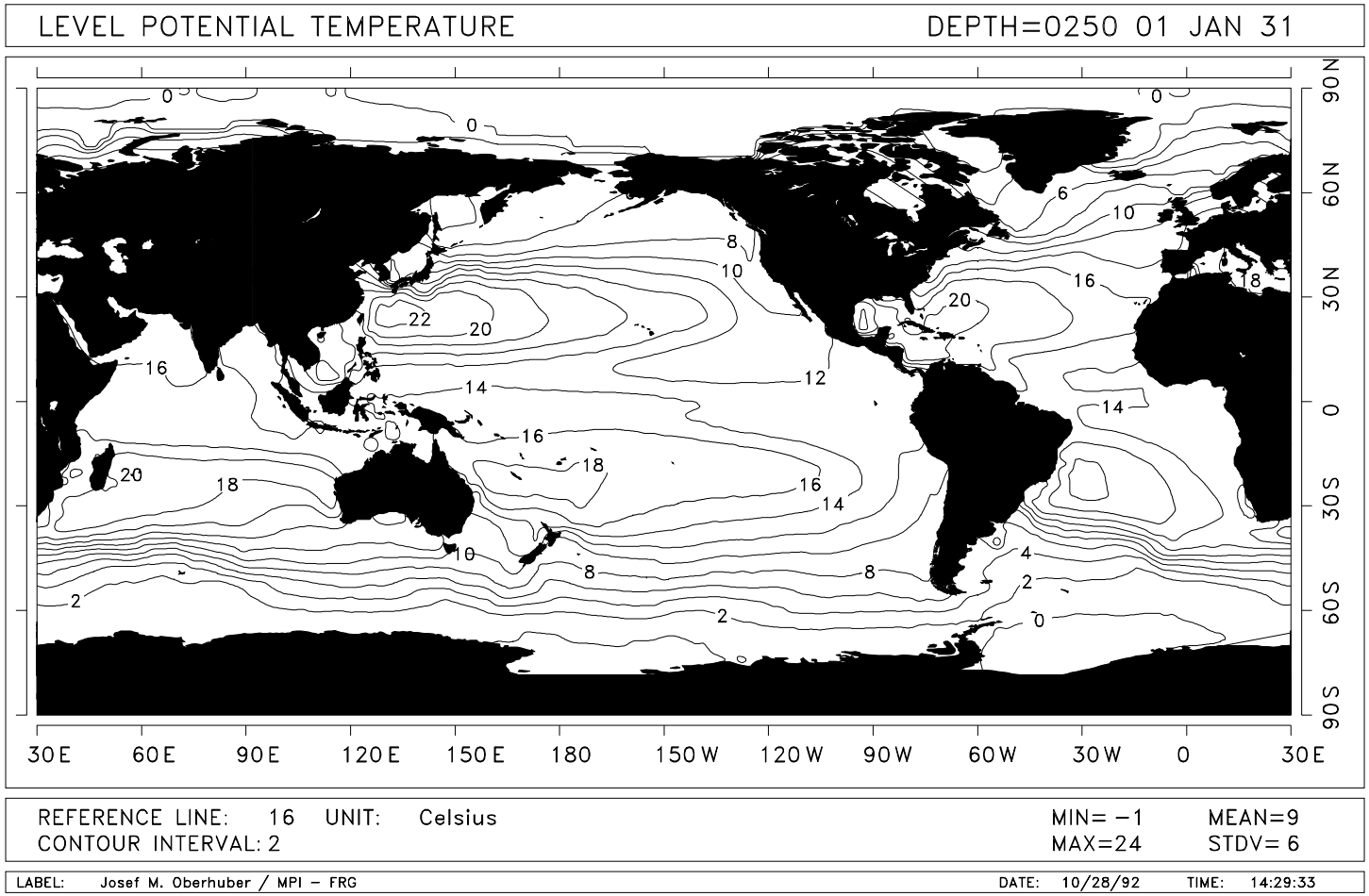

Figure 9 Temperature at 250m Depth in OPYC / T106.

Example showing a contour plot for the temperature of the OPYC/T106 at 250m depth. The quantity code is *KODEQ*=13, the section code is *KODES*=1 for xy-section and the location code is *KODEL*=250 for 250m depth.

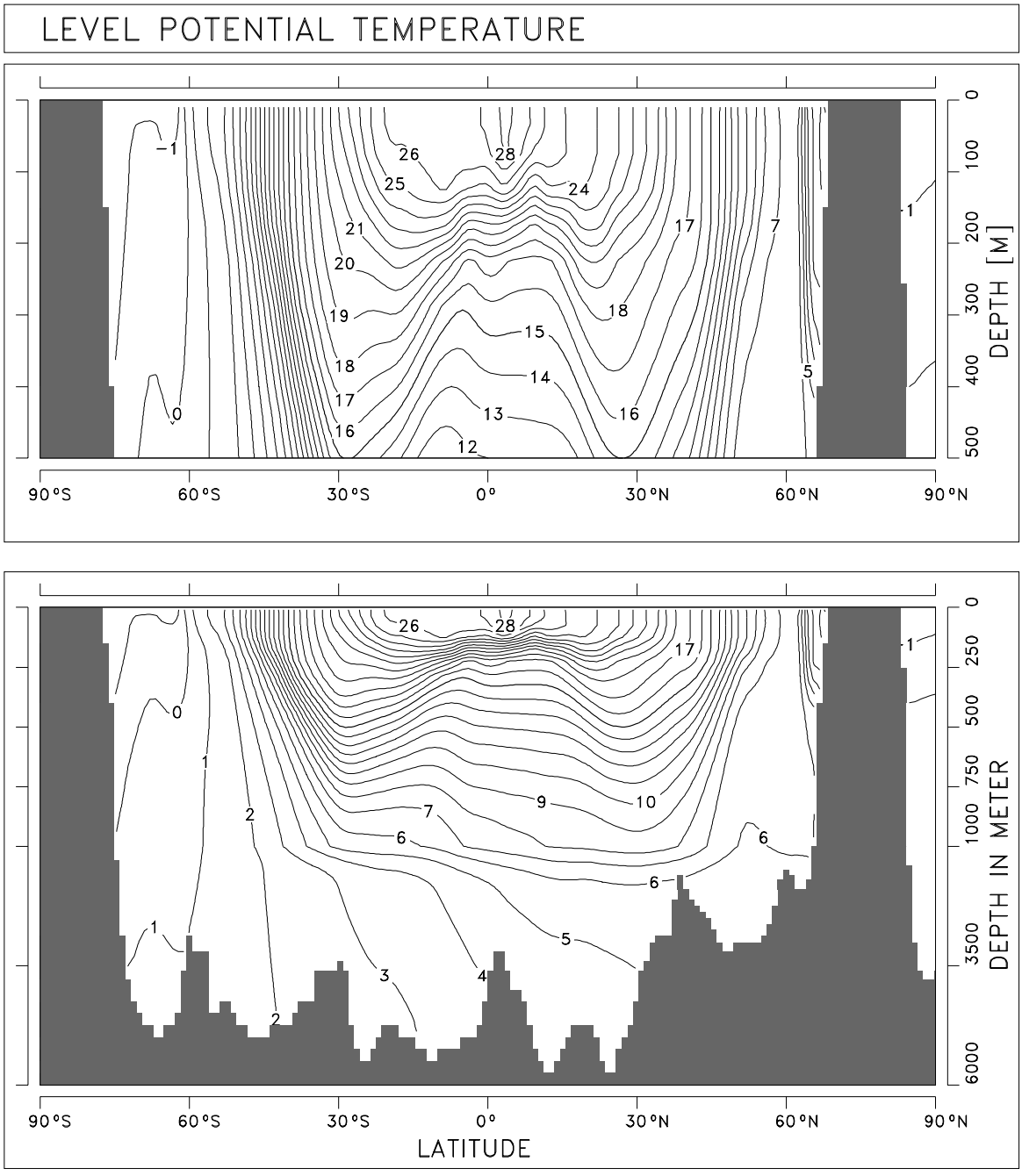

Figure 10 Temperature Section at 30^{o}W in OPYC / T106.

Example showing a yz-section for the potential temperature at 30^{o}W from the OPYC / T106. The quantity code is *KODEQ*=13, the section code is *KODES*=3. For yz-section and the location code is *KODEL*=330 for 30^{o}W.

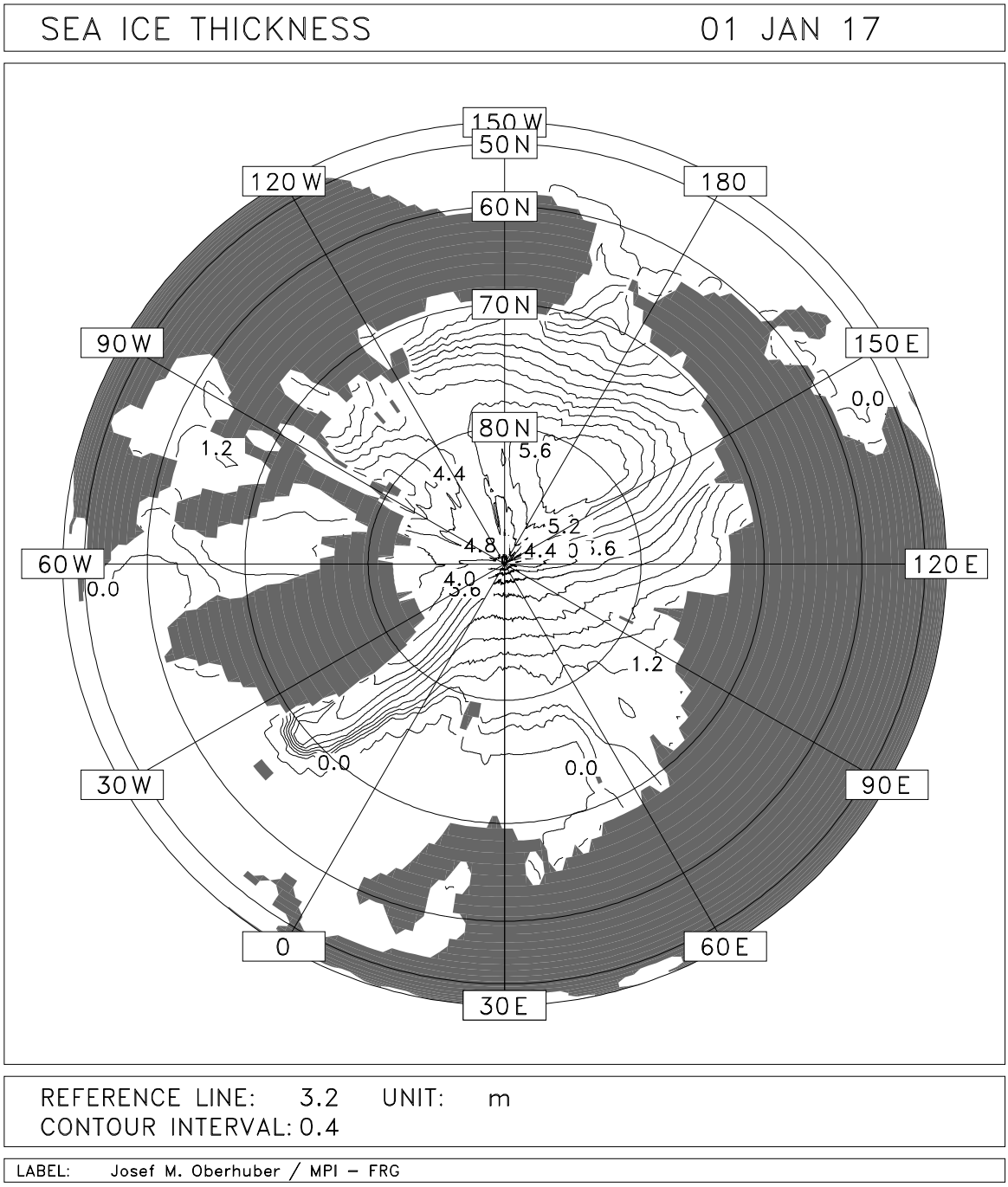

Figure 11 Sea Ice Thickness in the Arctic in OPYC / T106.

Example showing a north-polar projection for the sea ice thickness of OPYC/T106. The quantity code is *KODEQ*=18, the section code is *KODES*=1 for a xy-section and the location code is *KODEL*=0 for the surface.

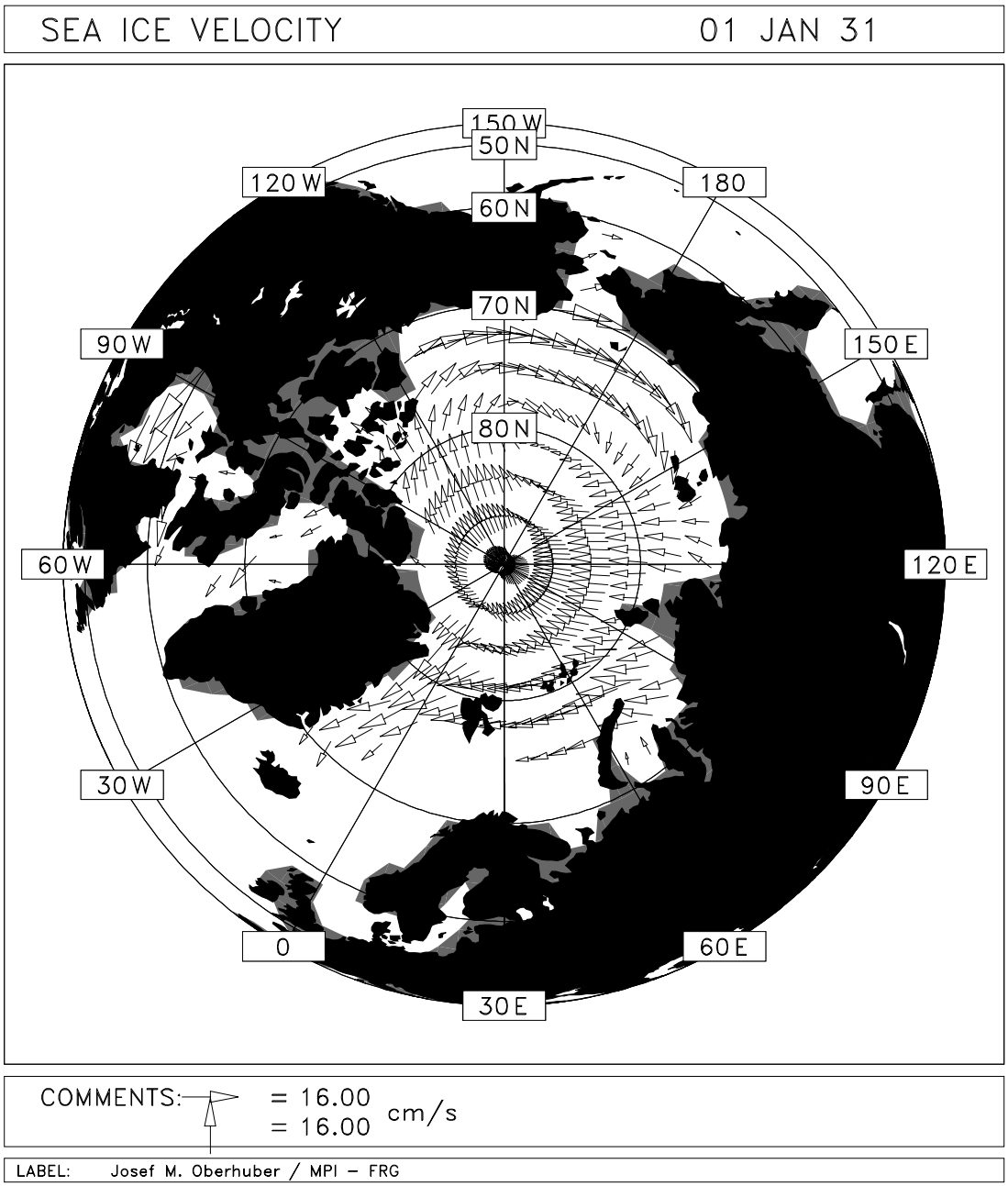

Figure 12 Sea Ice Velocity in the Arctic in OPYC / T42.

Example showing a north-polar projection for the sea ice velocity of OPYC/T42 using *NPRETTY*=1. The quantity code is *KODEQ*=17, the section code is *KODES*=1 and the location code is *KODEL*=0.

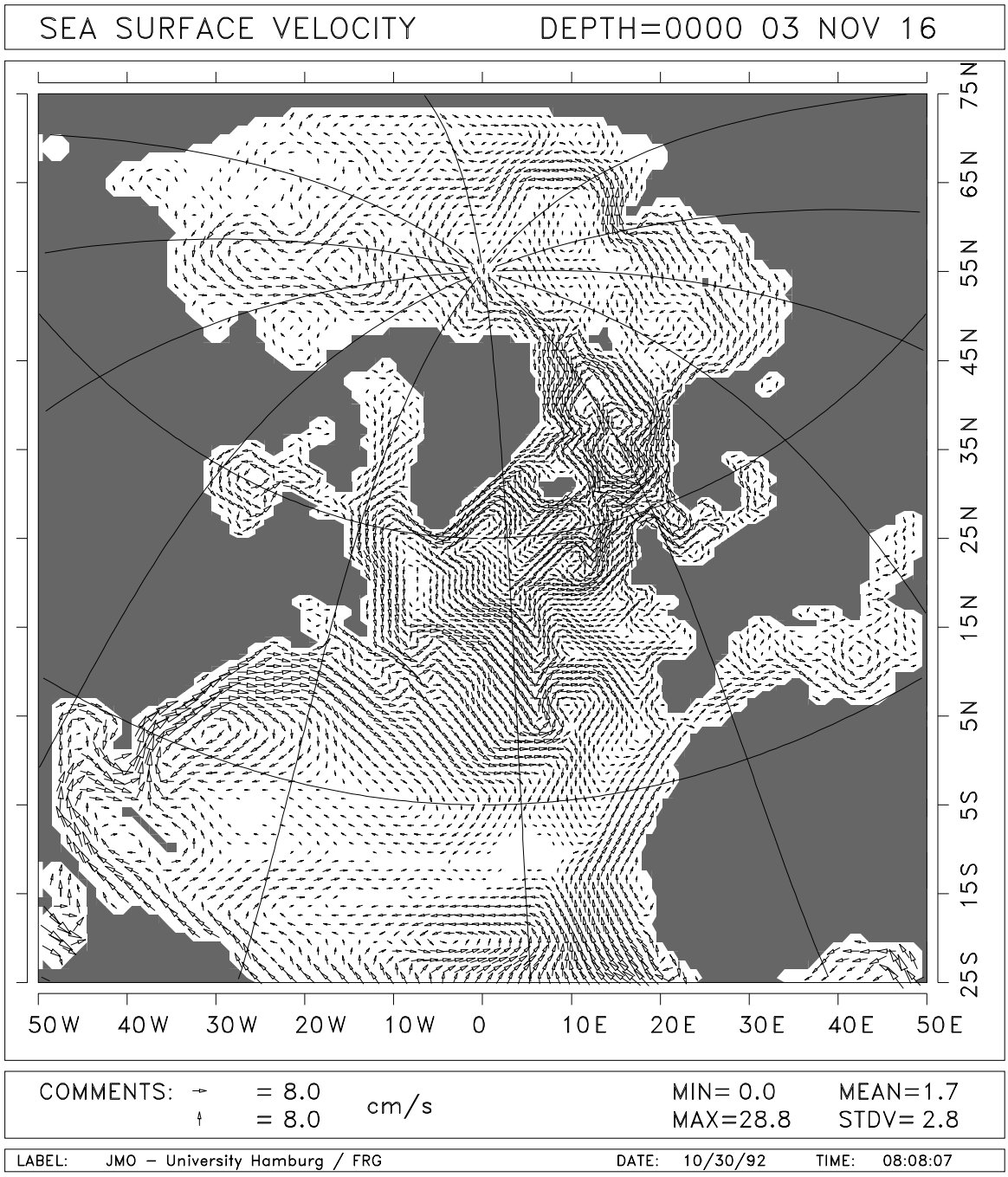

Figure 13 Surface Flow of a North Atlantic-Arctic model.

Example showing the surface flow of a North Atlantic-Arctic model using a rotated grid (*ALPHA*=-35., *BETA*=-35., *GAMMA*=0.). Other parameters are similar to those in Figure 8.

7. REFERENCES

7.1 CODE REFERENCES

Table 37 Names for Prognostic Ocean Quantities

Code Name	Formulae Name	Meaning	Unit	Equation
UH	ρhu	zonal mass flux	$\mathrm{kg(ms)}^{-1}$	(2.1.2.1)
VH	ρhv	meridional mass flux	$\mathrm{kg(ms)}^{-1}$	(2.1.2.1)
HEIGTH	h	layer thickness	m	(2.1.2.2)
TEMP	θ	potential temperature	K	(2.1.2.3)
SALT	S	salinity	gkg^{-1}	(2.1.2.4)
TRAC	C	tracer concentration		(2.1.2.5)

Table 38 Names for Diagnostic Ocean Quantities

Code Name	Formulae Name	Meaning	Unit	Equation
DENSITY	ρ	in situ density	kgm^{-3}	(2.1.2.8)
POTDENS	σ_θ	potential density	kgm^{-3}	(2.1.2.8)
PRESS	P	in situ pressure	$\mathrm{m}^2\mathrm{s}^{-2}$	(2.1.2.9)
DIFFUSE	A^s	scalar diffusion coeff.	$\mathrm{m}^2\mathrm{s}^{-1}$	(2.1.2.17)
VERTUP	w_k^{k+}	upward diffusion	ms^{-1}	(2.1.2.1)
VERTDN	w_k^{k-}	downward diffusion	ms^{-1}	(2.1.2.1)

Table 39 Names for Surface Flux Parameters

Code Name	Formulae Name	Meaning	Unit	Equation
QFLUX	Q	net surface heat flux	$\mathrm{W\,m}^{-2}$	(2.3.1.1)
QSOLAR	Q_s	downward solar heat flux	$\mathrm{W\,m}^{-2}$	(2.3.1.1)
QSENSIB	Q_H	surface sensible heat flux	$\mathrm{W\,m}^{-2}$	(2.3.1.3)
QLATENT	Q_L	surface latent heat flux	$\mathrm{W\,m}^{-2}$	(2.3.1.4)
QLONG	Q_l	longwave radiation budget	$\mathrm{W\,m}^{-2}$	(2.3.1.7)
SFLUX	R	equivalent salt flux	$\mathrm{W\,(Km)}^{-2}$	(2.2.2.4)

Table 39 Names for Surface Flux Parameters

Code Name	Formulae Name	Meaning	Unit	Equation
BFLUX	B	net surface buoyancy flux	$m^2 s^{-3}$	(2.2.2.3)
BSOLAR	B_s	solar buoyancy flux	$m^2 s^{-3}$	(2.2.2.5)
USTERN	u_*	friction velocity	ms^{-1}	(2.3.4.1)
HEATLAT	L_w	heat of fusion	$Ws\,(kg)^{-1}$	(2.3.1.4)
SIGMA	s	Stefan Boltzmann constant	$kg\,(ms)^{-1}$	(2.3.1.7)
SOLAR	S_0	solar constant	$W m^{-2}$	(2.3.1.11)
CPAIR	$c_{p,\,air}$	specific heat capacity	$Ws\,(kgK)^{-1}$	(2.3.1.3)
CHARNCK	c_{char}	Charnock constant		(2.3.2.7)
RKARMAN	κ	von Karman constant		(2.3.2.4)
ALBEDO	γ	albedo		(2.3.1.11)
ABSORB	ε	emmisivity of water		(2.3.1.7)
TURBID	h_B	penetration depth of Q_s	m	(2.3.1.1)
TURBREL	g	transmission coefficient	$W m^{-2}$	(2.3.1.1)

Table 40 Names for Prognostic Snow and Sea Ice Variables

Code Name	Formulae Name	Meaning	Unit	Equation
UICE	$(uh)_0$	zonal flux of ice	$m^2 s^{-1}$	(1.73)
VICE	$(vh)_0$	meridional flux of ice	$m^2 s^{-1}$	(1.73)
HICE	h_0	ice thickness	m	(1.74)
COMPACT	q_0	ice compactness		(1.75)
HSNOW	s_0	snow depth	m	(1.76)
TICE	T_h	ice temperature	K	(1.83)
TSNOW	T_s	snow temperature	K	(1.84)

Table 41 Names for Diagnostic Mixed Layer Variables

Code Name	Formulae Name	Meaning	Unit	Equation
VERTIC	w	entrainment rate	ms^{-1}	(2.2.2.2)
HEQU	h_M	Monin-Obukhov length	m	(2.2.2.13)
ENERGY	u_*^3	turbulent kinetic energy	m^3s^{-3}	(2.2.2.2)
TAUX	τ_x	Garwood-term	m^2s^{-2}	(2.2.2.2)

Table 42 Names for Horizontal and Vertical Diffusion Parameters

Code Name	Formulae Name	Meaning	Unit	Equation
DIFTIMV	T_v	damping time scale for monetum diffusion	s	(2.2.2.3)
DIFTIMS	T_s	damping time scale for scalar diffusion	s	(2.2.2.4)
PROPORT	$A^{s,1}$	proportionality coefficient	m^2s^{-2}	(2.1.2.17)
DIFLAY0	$A^{h,0}$	background layer diffusion	m^2s^{-1}	(2.2.2.1)
DIFLAY1	$A^{h,1}$	proportionality coefficient	m^2s^{-2}	(2.2.2.1)
TURBLEV	$2m_o u_*^3$	*TKE* for vertical mixing	m^3s^{-3}	(2.4.1.1)
HMIX	h_w	reference thickness	m	(2.4.1.1)
DRAGT	δ	time constant for salinity forcing	s^{-1}	(2.3.3.1)
DRAGS	c_d	drag coefficient at surface		(2.1.2.7)
DRAGV	c_d	drag coefficient between layers		(2.1.2.7)
DRAGB	c_d	drag coefficient at bottom		(2.1.2.7)

Table 43 Names for Mixed Layer Parameters

Code Name	Formulae Name	Meaning	Unit	Equation
*CMIX*0	m_4	linear damping term of *ML*	m^2s^{-3}	(2.2.2.5)
*CMIX*1	h_{TKE}	constant *TKE* decay length scale	m	(2.2.2.11)
*CMIX*2	Ri_{crit}	critical Richardson number		(2.2.2.5)
*CMIX*3	κ^{-1}	Ekman dissipation for *TKE*		(2.2.2.9)

Table 43 Names for Mixed Layer Parameters

Code Name	Formulae Name	Meaning	Unit	Equation
*CMIX*4	m_o	wave enery conversion factor		(2.2.2.5)
*CMIX*5	m_3	Garwood term tuning coefficient		(2.2.2.5)
*CMIX*6	h_{BUO}	constant buoyancy decay length scale	m	(2.2.2.12)
*CMIX*7	μ^{-1}	Ekman dissipation for buoyancy		(2.2.2.10)
*CMIX*8	g'_0	threshold for g'	$\mathrm{m\,s^{-2}}$	(2.2.2.5)
*CMIX*9	m_5	tuning coefficient for buoyancy flux		(2.2.2.5)
TURBEN	m_6	tuning parameter	$\mathrm{m^3 s^{-3}}$	(2.2.2.5)

Table 44 Names of Snow and Sea Ice Parameters

Code Name	Formulae Name	Meaning	Unit	Equation
EDDYUV	A_0^v	diffusion coefficient	$\mathrm{m^2 s^{-1}}$	(2.5.1.1)
EDDYS	A_0^s	diffusion coefficient	$\mathrm{m^2 s^{-1}}$	(2.5.1.2)
RHOICE	ρ_i	ice density	$\mathrm{kg\,m^{-3}}$	(2.5.2.1)
RHOSNOW	ρ_s	snow density	$\mathrm{kg\,m^{-3}}$	(2.5.2.2)
CICE	k_i	ice conductivity	$\mathrm{W\,(Km)^{-1}}$	(2.5.2.1)
CSNOW	k_s	snow conductivity	$\mathrm{W\,(Km)^{-1}}$	(2.5.2.2)
CPICE	c_{pi}	specific heat of ice	$\mathrm{Ws\,(kgK)^{-1}}$	(2.5.2.1)
CPMELT	c_{pm}	heat of fusion	$\mathrm{Ws\,kg^{-1}}$	(2.5.2.9)
EXCENT	e	eccentricity		(2.5.1.10)
PRESICE	P_i/ρ_i	proportionality	$\mathrm{m^2 s^{-2}}$	(2.5.1.8)
PDECAY	ε_1	proportionality	$\mathrm{m^2 s^{-2}}$	(2.5.1.8)
HNULL	h_i	tuning coefficient	m	(2.5.2.10)
EPSNULL	ε_0	tuning coefficient	$\mathrm{s^{-1}}$	(2.5.1.9)
AGING	γ	snow aging parameter	$\mathrm{s^{-1}}$	(2.5.2.11)
SALTICE	S_i	salinity of sea ice	$\mathrm{g\,kg^{-1}}$	(2.2.2.7)
PENET	h_p	parameter for salt ejection	$\mathrm{m^3 kg^{-1}}$	(2.5.3.1)

Table 45 Names of Observed Quantities

Code Name	Formulae Name	Meaning	Unit	Equation
ATU	u	zonal surface wind component	$\mathrm{m\,s^{-1}}$	(2.1.2.7)
ATV	v	meridional surface wind component	$\mathrm{m\,s^{-1}}$	(2.1.2.7)
UVABSOL	V	wind speed	$\mathrm{m\,s^{-1}}$	(2.3.4.1)
UVARIAN	$\sigma(V)$	standard deviation of wind speed	$\mathrm{m\,s^{-1}}$	(2.3.4.1)
AIRT	T_a	surface air temperature	K	(2.3.1.3)
SST		sea surface temperature	K	
SSS	S_{obs}	sea surface salinity	$\mathrm{g\,kg^{-1}}$	(2.3.3.1)
CLOUDS	n	cloudiness		(2.3.1.10)
HUMID	r	relative humidity		(2.3.1.4)
RAIN	P	precipitation	$\mathrm{m\,s^{-1}}$	
DEPTH	D	topography depth	m	(2.1.2.12)

7.2 LITERATURE REFERENCES

Berliand, M.E. and T.G. Berliand, 1952:
Determining the Net Long-Wave Radiation of the Earth with Consideration of the Effect of Cloudiness.
Isv. Akad. Nauk. SSSR Ser. Geofis, 1.

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., 11, 755-770.

Bryden, H., 1973:
New Polynomials for Thermal Expansion, Adiabatic Temperature Gradient and Potential Temperature of Seawater.
Deep Sea Res., 20, 401-408.

Budyko, M.I., 1974:
Climate and Life.
Academic Press, 508 pp.

Crowley, W.P., 1968:
Numerical Advection Experiments.
Mon. Wea. Rev., 1, 1-11.

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. Muller, 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. Muller, 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.

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.

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-Correcred Global Precipitation.
Internat. J. Climatol. Res., 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.

Millero, F.J., 1978:
Freezing Point of Seawater.
Eighth Report of the Joint Panel on Oceanographic Tables and Standards,
Unesco Techn. Pap. in Mar. Sci., 28, 1Annex 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, VI, Wiley Interscience, 97-115.

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 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 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 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., 23, 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., 23, 830-845

Paltridge, G.W and C.M.R. Platt, 1976:
Radiative processes in Meteorology and Climatology.
Developments in Atmospheric Science, Vol. 5, Elsevier Scientific Publ. Co., Amsterdam, 318 pp.

Paulson, C.A., J.J. Simpson, 1977:
Irradiance Measurements in the Upper Ocean.
J. Phys. Oceanogr., 7, 952-956.

Reed, R.K., 1977:
On Estimating Insolation over the Ocean.
J. Phys. Oceanogr., 7, 482-485.

Robert A., J. Henderson and C. Turbnell, 1972:
An Implicit Time Step Scheme for Baroclinic Models of the Atmosphere.
Mon. Wea. Rev., 100, 329-335.

Shea, D.J., 1986:
Climatological Atlas.
NCAR Technical Note, NCAR / TN-338-STR, 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.

Trenberth, K.E., J.G. Olson and W.G. Large, 1989:
A Global Ocean Wind Stress Climatology Based on ECMWF Analyses.
NCAR Technical Note, NCAR / TN-338-STR, Boulder, Colorado.

UNESCO, 1981:
The Practical Salinity Sscale 1978 and the International Equation of State of Seawater 1980.
Unesco Techn. Pap. in Mar. Sci., 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 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., Dep. of the Interior, Canberra, Australia.