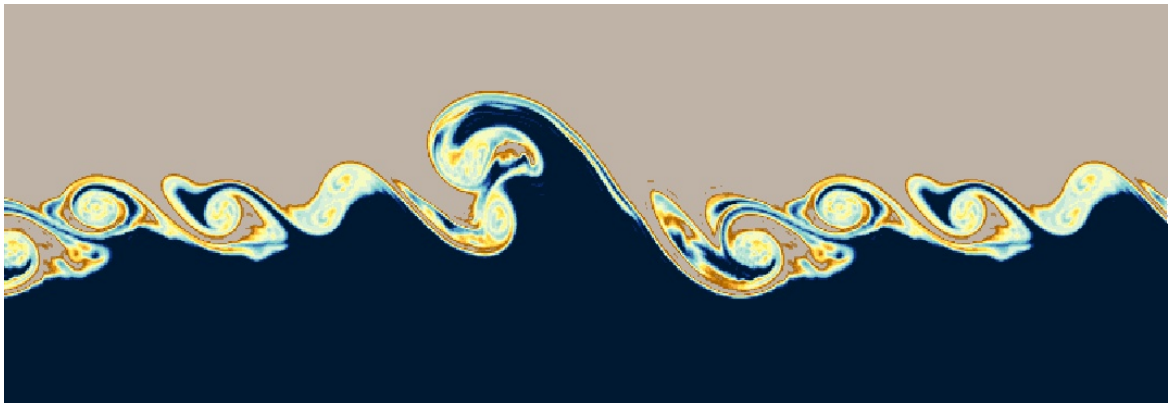
CPFLAME 0.98 documentation

Carsten Eden, KlimaCampus, University Hamburg, Germany

11. April 2011



Contents

1 Introduction

1.1 Summary of the model

CPFLAME is a numerical circulation model which was written for educational purpose. It is not meant to be a realistic ocean model, but a simple numerical tool to easily setup and run idealised numerical experiments. The main assumptions or restrictions are

- the neglection of any thermodynamic equation (i.e. the buoyancy budget is considered only, although passive tracers and also salinity can be included)
- the Boussinesq approximation (in its form applicable to the ocean, i.e. mass conservation is replaced by volume conservation and buoyancy perturbations act in the gravity term of the Navier-Stokes equation only)
- the rotating coordinate system is Cartesian together with a β-plane approximation.

On the other hand, fully non-hydrostatic situations as well as large-scale oceanic flows can be considered. Many idealised experiments and examples are preconfigured and can be easily chosen. Prerequisites for installation is the NetCDF-library only (since disk IO

is realized mainly using the NetCDF format) and a Fortran 90 compiler which is both provided as free software at the time of this writing. The numerical code can be vectorised and a simple parallelisation is realized based on the MPI(CH)-library (which is also free software) to enhance performance.

1.2 Navier-Stokes equations

The Navier-Stokes equations, the buoyancy conservation equation and the mass (or volume) conservation equation within the Boussinesq approximation in a rotating Cartesian coordinate system are solved numerically in CPFLAME. These equations are given here for reference

$$\begin{aligned} u_t &= F_u - p_x && (1)\\ v_t &= F_v - p_y && (2)\\ \epsilon w_t &= \epsilon F_w - p_z - b && (3)\\ b_t &= -\nabla\cdot(\boldsymbol{u}b) + K_h\nabla_h^2 b + K_v b_{zz} && (4)\\ 0 &= \nabla\cdot\boldsymbol{u} && (5) \end{aligned}$$

with the (scaled) pressure $p = p^*/\rho_0$, where ρ_0 is a constant reference density, the *negative* buoyancy $b = g\rho/\rho_0$ and

$$\begin{aligned} F_u &= fv - f_h w - \boldsymbol{u}\cdot\nabla u + A_h\nabla_h^2 u + A_v u_{zz} - A^h_{biha}\nabla_h^4 u - A^v_{biha}u_{zz} && (6)\\ F_v &= -fu - \boldsymbol{u}\cdot\nabla v + A_h\nabla_h^2 v + A_v v_{zz} - A^h_{biha}\nabla_h^4 v - A^v_{biha}v_{zz} && (7)\\ F_w &= f_h u - \boldsymbol{u}\cdot\nabla w + A_h\nabla_h^2 w + A_v w_{zz} - A^h_{biha}\nabla_h^4 w - A^v_{biha}w_{zz} && (8) \end{aligned}$$

The Boussinesq approximation was made, i.e. a constant reference density ρ_0 was used everywhere except for the gravity force in the vertical momentum equation. Note that f is related to the vertical component of the Coriolis force while and f_h to the horizontal component of the Coriolis force. Note also that the parameter ϵ is set to one, and that the hydrostatic approximation can be made by setting $\epsilon = f_h = 0$. Since a rotating Cartesian coordinate system is used, a β-plane approximation is employed as well. That means that the Coriolis parameters f and f_h are given by

$$f = 2\Omega\sin\phi_0 + \frac{2\Omega}{r}y\cos\phi_0 \quad , \quad f_h = 2\Omega\cos\phi_0 - \frac{2\Omega}{r}y\sin\phi_0 \qquad (9)$$

where r denotes Earth's radius, ϕ_0 a reference latitude and Ω the rotational frequency of the earth.

1.3 Discretisation

Variables are discretised on a Arakawa C-grid. Pressure p and buoyancy are centred in a tracer box. On the eastern, northern and upper sides of these boxes, the zonal, meridional

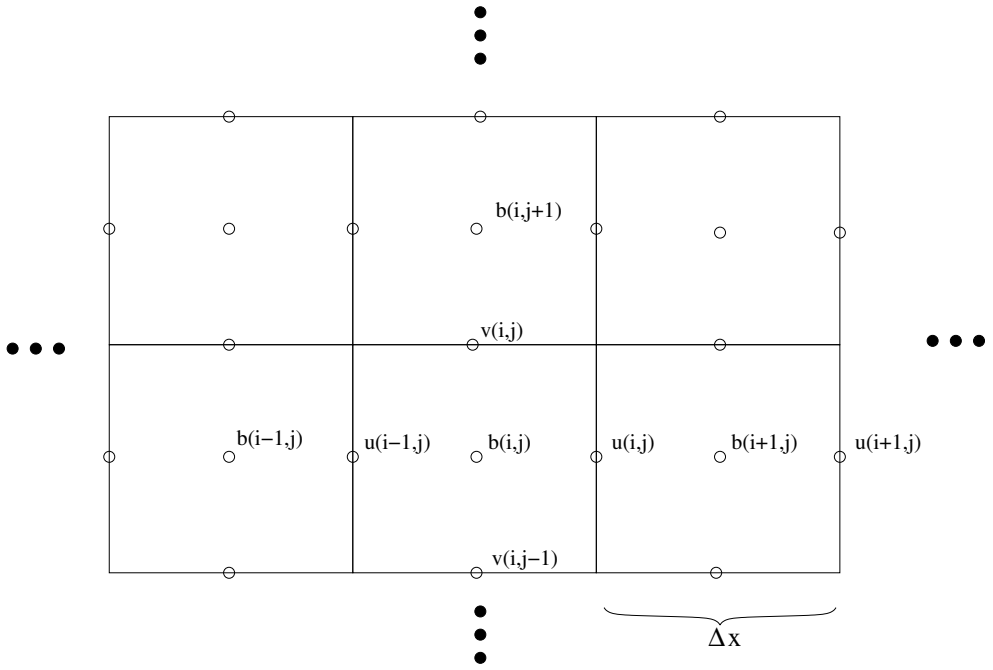


Figure 1: C-grid

and vertical velocities are placed. The horizontal grid arrangement is shown in Fig. 1. Advective fluxes of buoyancy across the eastern cell border, $F^E_{i,j,k}$ and the corresponding fluxes across the meridional and vertical boundary can be computed as

$$F^E_{i,j,k} = \Delta x \Delta z \, u_{i,j,k}(b_{i,j,k} + b_{i+1,j,k})/2 \tag{10}$$

$$F^N_{i,j,k} = \Delta x \Delta z \, v_{i,j,k}(b_{i,j,k} + b_{i,j+1,k})/2 \tag{11}$$

$$F^T_{i,j,k} = \Delta x^2 \, w_{i,j,k}(b_{i,j,k} + b_{i,j,k+1})/2 \tag{12}$$

Note that this discretisation represents the standard second order scheme, but that other higher order discretisation are possible. It is also obvious that diffusive fluxes can be discretised in a similar way. In any case, however, the convergence of these fluxes leads to a change in buoyancy content in the grid cell

$$(\Delta x^2 \Delta z)\frac{\partial}{\partial t} b_{i,j,k} = (F^E_{i,j,k} - F^E_{i-1,j,k}) + (F^N_{i,j,k} - F^N_{i,j-1,k}) + (F^T_{i,j,k} - F^T_{i,j,k-1}) \tag{13}$$

It is clear that fluxes of buoyancy across the boundaries must be set to zero. For momentum, free-slip or no-slip boundary conditions can be used.

The temporal discretisation is explicit (except for vertical diffusion, see below) using a so-called leapfrog time stepping for the advective terms as default (other advection schemes are possible) and a Robert's time filter to eliminate the numerical mode. The two-dimensional Poisson equation for the surface pressure (or implicit free surface, see below) and the three-dimensional equation for the non-hydrostatic pressure are solved using a simple preconditioned conjugate gradient solver.

2 Installation

2.1 Quick start

This short installation procedure assumes a Unix system and that all libraries and compilers are available. See below for necessary prerequisites in case of problems to compile.

- Make a new directory where the model code shall be placed. That directory is called `dir` hereafter.
- Extract the tar-archive in `dir`.
- Edit the file `Makefile`. The variable `CPFLAME` at the head of `Makefile` has to be set to `dir`.
- Copy one of the files in `dir/configure` to a (new) file `dir/Makefile_host` depending on the specific system and compiler.
- Type `make`. If something went wrong it is necessary to modify `dir/Makefile_host` as described below.
- Find the executable `model.x` in `dir/bin`.
- Run the executable, investigate the output and change model setup.

2.2 Detailed instructions

If the above procedure did not work in order to compile the model code, it is most likely that the file `dir/Makefile_host` has to be modified. This file is automatically included in the main Makefile `dir/Makefile` and in each of the Makefiles in the subdirectories. The file `Makefile` is read by the command `make` to obtain informations how and in which order to compile the code. The file `dir/Makefile_host` contains system and compiler specific details, which are supplied to `Makefile` (and `make`) and are explained in this section.

In order to compile the code a working Fortran90 compiler is needed. Examples are the freely available GNU `gfortran` or Intel's `ifort`. The compiler name (i.e. the way to invoke it) is specified in `dir/Makefile_host` by the variable `F90`. In addition to the Fortran90 compiler, a C compiler and a CPP compiler are also needed. Their names have to be specified by the variables `CC` and `CPP` in `dir/Makefile_host` respectively. Usually `CC = cc` and `CPP = ${F90} -E` should work. For linking of all compiled files a linker has to be specified by the variable `LINKER` which is usually identical to the Fortran90 compiler. Specific options can be passed to the Fortran90 and C compiler by the variables `F90_OPTS` and `CC_OPTS`. Note that it is assumed that the model in run in double precision, i.e. that real variables in Fortran90 code are represented by 8 bytes. Usually, this has to be specified to the Fortran90 compiler by a option in `F90_OPTS`.

The other important prerequisite for the model is the NetCDF library, which is freely available at `http://www.unidata.ucar.edu/software/netcdf`. However, on most system it should be already installed. The location of the library has to be specified in `dir/Makefile_host` by the variable `NETCDF_LIB_DIR` (location of the file `libnetcdf.a`) and `NETCDF_INCL_DIR` (location of the file `netcdf.inc`) which can be usually set to `/usr/lib` and `/usr/include` respectively. In case of parallel execution of the model, an implementation of the Message Parsing Interface (MPI) has to be provided. The location of the MPI library has to be set by the variables `MPI_LIB_DIR` and `MPI_INCL_DIR` in the same manner as for the NetCDF Library.

3 Directory structure and model configuration

3.1 Directory structure

The numerical code is distributed over several directories. Here is a list of these directories and a short characterisation of its content. All references to files or directories are relative to the directory in which CPFLAME is located.

- `./model`: the specific model code essential to any model run
- `./extra_modules`: specific add-ons which are not really needed for model runs
- `./extra_setup`: template subroutines for specific configurations for add-ons
- `./setup`: template subroutines containing specific configuration
- `./include`: contains a single top-level include file with CPP directives
- `./mpp`: (most of the) code specific to parallelisation
- `./misc_modules`: more general and multi purpose code
- `./configure`: contains alternatives for compiler specifications
- `./doc`: contains documentation
- `./bin`: contains executable after successful compilation

The top-level program of the model is `./model/driver.F`. Here, the work flow of the main routines can be seen. The main module is `./model/cpflame_module.F` containing all important parameters and model variables. A short description of each model variable can be found in this file.

3.2 Model configuration

The model configuration is realized by template subroutines for which specific routines are supplied for each specific experiment. The idea is to keep the whole model configuration in a single file containing all relevant routines. Specific examples can be found in the directory `./setup`. The configuration files are alternatively included at the head of the file `./model/setup.F` by using a CPP include directive. In other words, the specific configuration is chosen by including (uncommenting) the relevant model specifications in `./model/setup.F`. It is clear that only one specific model specification at a time can be chosen.

In the template routines most of the specific configuration can be chosen, including the initial and surface boundary conditions, relaxation zones, sub-grid-scale parameterisations, etc. The other important mean to influence the models behaviour is given by the (single) include file `./include/options.inc`. Here, a few top-level CPP directives can be chosen. By uncommenting the directives, certain code blocks are switched on or off by the cpp compiler. For instance by uncommenting the switch `no_mpp`, it is specified that no parallel routines are needed. The other switches are mainly for diagnostic purpose, simulation of passive tracers or Lagrangian particles, specialised sub-grid-scale parameterisations or by defining a background state, as explained above.

There are additional template routines needed for some of the add-ons in the direxstory `./extra_modules` which are contained in `./extra_setup`. Currently, there are only configurations for passive tracer simulations needed.

3.3 Running and restarting the model

After successful compilation, the executable `model.x` can be found in the directory `./bin`. The executable can be run directly in `./bin` or in any other directory. No further files are needed for a model integration.

The model integration will start from the initial conditions specified in the template configuration subroutines and will integrate over a time period specified by the variable `runlen` given in days. After that period the integration stops and a restart file `restart.dta` will be written by the model, containing the last two time steps of each model variable. This restart file will be read when the model is restarted. Note that the model will always try to read a restart file, which then overides the initial conditions specified in the template configuration routines. If there is no restart file present, the model will use the initial conditions. Note also that at the end of each integration, any existing restart file will be overwritten.

3.4 Sample configurations

This is an incomplete list of sample configurations which can be found in the directory `./setup`.

3.4.1 Non-hydrostatic configurations

- `kelv_helm1.F`

  This is a simple two-layer system with a large shear between the layers to demonstrate Kelvin-Helmholtz instabilities as in Eden et al. (2009). The domain is periodic in x and y, 14 m deep, 16 m long and there are no surface fluxes at the top or bottom. The initial conditions are two layers of equal thickness with a buoyancy difference corresponding to about 10 K moving with 1 m/s to the east and west. At the layer interface a small disturbance in buoyancy is introduced such that in the simulation Kelvin-Helmholtz instability will show up. The CPP directive `two_dim` at the head of the file `kelv_helm1.F` specifies that the domain is two-dimensional, i.e. in the x-z plane. The variable `fac` can be increased in order to increase the spatial (and temporal) resolution. For `fac=1.0` the spatial resolution is 25 cm in each direction. For sub-grid-scale parameterisation the TKE closure in file `./extra_modules/tke_closure.F` should be used (see section 4.11).

- `convection1.F`

  This is a setup with an initial stratification (with $N = 0.025s^{-1}$) at rest and prescribed surface buoyancy loss corresponding to a heat loss of 10 W/m^2 to simulate free convection. The CPP directive `two_dim` at the head of the file `convection1.F` specifies that the domain is two-dimensional, i.e. in the x-z plane. The variable `fac` can be increased in order to increase the spatial (and temporal) resolution. For `fac=1.0` the spatial resolution is 1 m in each direction. For sub-grid-scale parameterisation the TKE closure in file `./extra_modules/tke_closure.F` should be used (see section 4.11).

- `rayleigh1.F`

  This is a setup to demonstrate Rayleigh-Bernard convection. A $10m \times 30$ m basin is heated from below and cooled from above with a heat flux of 175 W/m^2. The CPP directive `two_dim` at the head of the file `rayleigh1.F` specifies that the domain is two-dimensional, i.e. in the x-z plane. The variable `fac` can be increased in order to increase the spatial (and temporal) resolution. For `fac=1.0` the spatial resolution is 0.5 m in each direction. For sub-grid-scale parameterisation the TKE closure in file `./extra_modules/tke_closure.F` should be used (see section 4.11).

- `mixedlayer1.F`

  Simulation of deep convection in the ocean. Similar to the experiment in Jones and Marshall (1996). Vertical resolution is 100 m while horizontal resolution is 250 m. The TKE closure is not used here instead biharmonic friction and diffusion.

3.4.2 Hydrostatic configurations

- `barbi_exp1.F`

Experiment DISTURB in Olbers and Eden (2003), i.e. a large scale buoyancy anomaly propagating westward in an ocean basin.

- `barbi_exp2.F`

  Experiment RIDGE in Olbers and Eden (2003), i.e. adjustment of a wind-driven gyre over topography resembling the North Atlantic Ridge to the flat-bottom solution.

- `wardlemarshall1.F`

  First experiment in Wardle and Marshall (2000), i.e. a reentrant channel driven by eastward wind stress profile.

- `wardlemarshall2.F`

  Second experiment in Wardle and Marshall (2000), i.e. spin-down of a baroclinically unstable front in a reentrant channel.

- `edenetal1.F`

  Experiment CHANNEL-3 in Eden et al. (2007), i.e. a reentrant channel with restoring zones at side walls.

- `theiss1.F`

  A wide channel model with relaxation at the side walls and interior damping as in Eden (2010), simulating strong eddy-driven zonal jets.

- `eady1.F`

  The classical Eady problem (Eady, 1949). A narrow channel on an f-plane with prescribed stratification and vertically sheared background zonal flow. The background state is prescribed using the CPP switch `enable_back_stratification`.

- `ACC1.F`

  A model with a channel attached to a closed basin, similar to the Southern and Atlantic Ocean as in Viebahn and Eden (2010). There is wind forcing over the channel part and buoyancy relaxation driving a large-scale meridional overturning circulation.

4 Parameterisations and boundary conditions

In this section, the most important sub-grid-scale parameterisations and boundary conditions which are used in the model and also other important aspect of the numerical implementation are listed and briefly discussed. Note that all references to files containing code are relative to the directory in which CPFLAME is located. It is also assumed that all parameters are given in SI units.

4.1 Rigid lid and surface pressure

To obtain the pressure in the Boussinesq system of equations, it is necessary to solve a diagnostic relation for the pressure. In case of the hydrostatic assumption, this relation simplified to one of the surface pressure only. The default treatment of the pressure is to assume a rigid lid, i.e. $w = 0$ at $z = 0$, which is briefly explained here.

It is useful to split the pressure in a hydrostatically balanced part and deviation from that. Within the hydrostatic approximation, the pressure p is given by

$$\int_z^0 \frac{\partial}{\partial z} p \, dz' = p(0) - p(z) = -\int_z^0 b \, dz' \quad , \quad p = p(0) + \int_z^0 b \, dz' \equiv p_s(x,y) + p_h(x,y,z) \tag{14}$$

where $p_s(x,y)$ denotes the surface pressure and $p_h = \int_z^0 b \, dz'$ the hydrostatic pressure. The surface pressure can be found from an elliptic equation given below. In general, however, the non-hydrostatic pressure p' will also add to give the full pressure p

$$p(x,y,z) = p_s(x,y,) + p_h(x,y,z) + \epsilon p'(x,y,z) \tag{15}$$

The non-hydrostatic pressure p' can be found by taking divergence of momentum equation

$$\frac{\partial}{\partial x} F_u + \frac{\partial}{\partial y} F_v + \frac{\partial}{\partial z} F_w - \nabla_h^2 (p_h + p_s) \quad = \quad \nabla^2 p' \tag{16}$$

A diagnostic equation for the surface pressure p_s can be found in case of a rigid lid from vertical integration of the momentum equation and the continuity equation

$$\nabla \cdot h \nabla p_s = \frac{\partial}{\partial x} \int_h^0 (F_u - (p_h + \epsilon p')_x) dz + \frac{\partial}{\partial y} \int_h^0 (F_v - (p_h + \epsilon p')_y) dz \tag{17}$$

Note that it is also possible to relax the rigid-lid assumption and to use a linearised free surface formulation.

4.2 Implicit free surface

An alternative surface boundary formulation to the rigid lid is given by the implicit free surface, which is activated by the switch `enable_free_surface` in the template configuration subroutine `set_parameter`, for which specific examples can be found in the directory `./setup`. Using the implicit free surface option, the surface pressure is replaced by the free surface η. The free surface equation is given by

$$\frac{\partial}{\partial t} \eta + \nabla_h \cdot \int_{-h}^{\eta} \boldsymbol{u}_h dz \approx \frac{\partial}{\partial t} \eta + \nabla_h \cdot \int_{-h}^{0} \boldsymbol{u}_h dz = 0 \tag{18}$$

where the free surface η is related to the surface pressure by $p_s = g\eta$. Discretisation of the time dependency yields

$$\eta^{n+1} + 2\Delta t \nabla_h \cdot \int_{-h}^{0} \boldsymbol{u}_h^{n+1} dz = \eta^{n-1} \tag{19}$$

and with $\boldsymbol{u}_h^{n+1} = \boldsymbol{u}_h^{n-1} + 2\Delta t(\boldsymbol{F}_h - \nabla_h p)$ this gives

$$\nabla_h \cdot gh\nabla_h \eta^{n+1} - \epsilon\eta^{n+1} = \nabla_h \cdot \int_{-h}^{0} (\boldsymbol{F}_h - \nabla_h p_h - \nabla_h p')dz - \epsilon\eta^{n-1} \tag{20}$$

with $\epsilon = (2\Delta t)^{-2}$.

4.3 Harmonic friction and diffusivity

Harmonic friction acting on the horizontal velocity (u, v) and harmonic diffusivity acting on the buoyancy (b) is always present and controlled by the horizontal viscosity `A_h`, the vertical viscosity `A_v`, the horizontal diffusivity `K_h`, and the vertical diffusivity `K_v`. Default values are zero. Note that in case of a non-hydrostatic simulation, the harmonic vertical and horizontal viscosity is also acting on the vertical velocity (w). Note also that all viscosities and diffusivities should be specified in the template configuration subroutine `set_parameter` for which specific examples can be found in the directory `./setup`.

4.4 Biharmonic friction and diffusivity

Biharmonic horizontal friction acting on the horizontal velocity (given e.g. by $A_{biha}\nabla^2 u$ in Eq. (6)) and biharmonic horizontal diffusion acting on buoyancy can be implemented by setting the logical variable `enable_biharmonic_friction` and `enable_biharmonic_diffusion` respectively to a true value in the configuration subroutine `set_parameter`. The horizontal biharmonic viscosity and diffusivity are given by `Ahbi` and `Khbi` respectively. Default values are zero.

Vertical biharmonic friction and diffusion can be implemented by setting the logical variable `enable_vert_biha_friction` and `enable_vert_biha_diffusion` respectively to a true value in the configuration subroutine `set_parameter`. The vertical biharmonic viscosity and diffusivity are given by `Avbi` and `Kvbi` respectively. Note that in case of a non-hydrostatic simulation, the biharmonic vertical and horizontal viscosity is also acting on the vertical velocity (w).

4.5 No-slip or free-slip boundary condition

Since the spatial discretisation is realized using a C-grid (see also section 1.3) it is natural to implement zero diffusive fluxes of momentum across boundaries of the model. This choice is equivalent a free-slip boundary condition and the default choice. No-slip lateral boundary conditions, assuming zero tangential velocities at the lateral boundaries implying a diffusive momentum flux across the lateral boundaries, are implemented setting the logical variable `enable_noslip` to a true value in the configuration subroutine `set_parameter` for which specific examples can be found in the directory `./setup`. Note that for the upper and lower boundaries free-slip conditions are always used. Note also that the no/free-slip conditions are implemented for harmonic and biharmonic friction.

4.6 Other boundary and initial conditions

The rigid-lid boundary condition is used as a default but a linearised implicit free surface formulation can be switched on by setting the the logical variable `enable_free_surface` to a true value in the configuration subroutine `set_parameter`. Lateral boundary conditions are either no-flux boundary condition for buoyancy and passive tracers (and no/free-slip for momentum as discussed above) or periodic boundary conditions. The latter can be chosen by setting the logical variables `enable_cyclic_x` and `enable_cyclic_y` to a true value in the configuration subroutine `set_parameter` to apply periodic boundary conditions in zonal and meridional direction respectively. Surface and eventually bottom fluxes of buoyancy and momentum can be specified in the configuration subroutine `boundary_conditions`. The same holds for the initial conditions and the configuration subroutine `initial_conditions`.

4.7 Bottom and interior friction

Bottom friction can be implemented by setting the logical variable `enable_bottom_stress` to a true value in the configuration subroutine `set_parameter`. The inverse time scale for bottom friction is given by `cdbot`. Default value is zero. Adding interior Rayleigh damping is implemented by the switch `enable_interior_stress` and the parameter `cdint`. For the bottom, a no-slip boundary condition is implemented by the switch `enable_bottom_noslip`.

4.8 Advection schemes

Although the choice of the advection scheme is not a sub-grid-scale parameterisation, it strongly affects the dissipation in the model and is therefore discussed here as well. It is possible to choose between the classical second-order central differences scheme as the default, the first order upwind schemen, a forth-order accurate central difference scheme and the Quicker advection scheme. They are implemented by setting the logical variables `enable_upwind_advection`, `enable_4th_advection` or `enable_quicker_advection` to a true value, respectively, in the configuration subroutine `set_parameter`. Note that this choice refers to the advection scheme of the tracers (buoyancy, passive tracer, TKE, etc) but not to the advection of momentum. The advection scheme for momentum is controlled by the variables `enable_4th_mom_advection` and `enable_quicker_mom_advection`, default is also the classical second-order central differences scheme.

Note that Quicker introduces a certain amount of numerical diffusion on buoyancy, such that sometimes no additional diffusion is needed in the model simulation. The standard and the forth-order accurate scheme do not contain any numerical diffusion, but are dispersive. The upwind scheme is very diffusive.

4.9 Hydrostatic approximation and convective adjustment

The hydrostatic approximation is switched on by setting the logical variable `enable_hydrostatic` to a true value in the configuration subroutine `set_parameter`. Note that when using the hydrostatic approximation the three-dimensional Poisson equation for the full pressure does not have to be solved, such that the model integration can be much faster.

Using the hydrostatic approximation, static instabilities are removed by setting the vertical diffusivity to a large value. This procedure is sometimes called convective adjustment. Vertical diffusion is calculated with a fully implicit time stepping scheme in that case. Find the code in the file `./model/convection.F`. The model variable containing the large vertical diffusivity is called `K_b` and written in the snapshot NetCDF file `cpflame.cdf`.

4.10 Meso-scale eddy parameterisation

4.10.1 Standard implementation

A meso-scale eddy parameterisation is implemented in the form of an eddy-driven velocity, which is added to the Eulerian mean velocity of the model to advect the mean buoyancy. The eddy-driven velocity is parameterized as proposed by Gent and McWilliams (1990), specifying the *thickness diffusivity*. Tracers with gradients on mean isopycnals are also subject to isopycnal diffusion for which an isopycnal diffusivity has to be specified in addition. The standard implementation of the Gent and McWilliams (1990) parameterisation can be found in the file `./extra_modules/bolus_velocity` and is activated by the preprocessor switch `enable_bolus_velocity` in the file `./include/options.inc`. In the file `./extra_modules/bolus_velocity`, the thickness diffusivity `K_const` has to be specified in m^2/s. The same value will be used for the isopycnal diffusivity. Default value is $1000\,m^2/s$. Further, two parameters controlling the maximal allowed isopycnal slopes are specified, `slopec` and `dslope`, for which reasonable default values are prescribed (there is not much reason to modify these parameters).

Diagnostic output in form of the vector streamfunction for the eddy-driven velocity and the thickness diffusivity written to the NetCDF file `bolus.cdf` can be activated by the preprocessor switch `enable_bolus_diag` in the file `./extra_modules/bolus_velocity.F`. It is also possible to suppress the application of the eddy-driven velocity, but still using isopycnal diffusion of passive tracer using the switch `enable_bolus_isopycnal_diffusion_only` in the file `./extra_modules/bolus_velocity.F`. This might become necessary when using the Residual Mean formulation (see below) instead of the eddy-driven velocity of Gent and McWilliams (1990). For the case that tracers like concentrations become negative by spurious undershooting caused by the isopycnal mixing scheme, it is possible to prevent this by the switch `delimit_tracer_fluxes` in the file `./extra_modules/bolus_velocity.F`.

Note that the implementation of the Gent and McWilliams (1990) parameterisation in CPFLAME follows the standard one as in Pacanowski (1995), which was shown by Griffies (1998) to be slightly unstable. Griffies (1998) also suggested a stable scheme for the standard implementation, which is, however, rather complicated and is therefore not implemented.

4.10.2 Residual Mean formulation

A simpler alternative than the standard one to implement the eddy-driven velocity is given by the Residual Mean formulation by Andrews et al. (1987). In this formulation, the momentum budget is considered as the one for the Residual velocity, which is given by the sum of the eddy-driven velocity of the Gent and McWilliams (1990) parameterisation and the normal Eulerian Mean velocity. The necessary modification of the momentum budget is given by adding a vertical friction term with viscosity Kf^2/N^2, where K denotes the thickness diffusivity of the Gent and McWilliams (1990) parameterisation. Since Kf^2/N^2 can become large, an implicit formulation for the vertical friction is necessary, analogous to the parameterisation of convection.

The formulation is implemented in the file `./extra_modules/vert_friction.F` and activated by the switch `enable_vert_friction` in the file `./include/options.inc`. The thickness diffusivity `K_const` has to be specified in m^2/s in `./extra_modules/vert_friction.F`. Default value is $1000\,\mathrm{m}^2/\mathrm{s}$. Further, a minimal threshold for the stability frequency N has to be specified in $1/\mathrm{s}$ and a maximal threshold for f^2/N^2. Diagnostic output in form of the thickness diffusivity and f^2/N^2 written to the NetCDF file `vert_friction_trm.cdf` can be activated by the preprocessor switch `enable_vert_friction_diagnostics` in the file `./extra_modules/vert_friction.F`.

Note that while the buoyancy equation does not need a further parameterisation for meso-scale eddies using the Residual Mean formulation, in the tracer equations isopycnal diffusion should be added. This can be done using the standard formulation using the switch `enable_bolus_isopycnal_diffusion_only` in the file `./extra_modules/bolus_velocity.F`.

4.11 Large eddy simulations (LES)

For LES simulations the closure by Raasch and Etling (1991) can be used. This closure is for non-hydrostatic simulations only and implemented in `./extra_modules/tke_closure.F`. and can be enabled by the respective switch in the include file `./include/options.inc`. Note that the code has to be recompiled for the switch to be effective. Note also that harmonic friction and diffusion should be set to zero using this closure and that biharmonic diffusion or friction should not be used in addition.

In the TKE closure, the model variables are considered as the mean variables while fluctuations remain unresolved and have to be parameterised. The Reynolds fluxes in the mean equations are parameterised as

$$\overline{u_i' b'} = -c_b \sqrt{\bar{e}} L \frac{\partial}{\partial x_i} \bar{b} \qquad \text{and} \qquad \overline{u_i' u_j'} = -c_m \sqrt{\bar{e}} L \frac{\partial}{\partial x_i} \bar{u}_j \tag{21}$$

where $\bar{e} = \frac{u'^2+v'^2+w'^2}{2}$ denotes turbulent kinetic energy (TKE) and L an eddy length scale. A prognostic budget for TKE is carried in the closure given by

$$\bar{e}_t + \bar{\boldsymbol{u}} \cdot \nabla \bar{e} \;=\; -\overline{\boldsymbol{u}' \cdot \nabla e} - \nabla \cdot \overline{\boldsymbol{u}' p'} - \overline{b' w'} - \overline{u' \boldsymbol{u}'} \cdot \nabla \bar{u} - \overline{v' \boldsymbol{u}'} \cdot \nabla \bar{v} - \overline{w' \boldsymbol{u}'} \cdot \nabla \bar{w} - \epsilon \tag{22}$$

where $\epsilon = \nu \overline{\frac{\partial u_i'}{\partial x_j}^2}$ denotes dissipation. All terms on the r.h.s have to be parameterised. In the one-equation model by Raasch and Etling (1991), the eddy length scale given by

$$L = \min(\Delta x, 0.76\frac{\sqrt{\bar{e}}}{N}) \tag{23}$$

where Δx denotes the grid scale. The dissipation is given by

$$\epsilon = c_\epsilon \frac{\bar{e}^{3/2}}{L} \tag{24}$$

The parameter c_ϵ is given by $c_\epsilon = 0.19 + 0.51\frac{L}{\Delta x}$. The pressure correlation and the triple correlation in the TKE budget are modelled as diffusion of TKE, i.e. as

$$-\overline{\boldsymbol{u}' \cdot \nabla e} - \nabla \cdot \overline{\boldsymbol{u}' p'} = \nabla \cdot c_{tke} \nabla \bar{e} \tag{25}$$

where the parameter $c_{tke} = 0.2$. Using the parameterisation for the fluxes with parameters $c_m = 0.1$ and $c_b = c_m(1 + \frac{L}{\Delta x})$ the TKE budget reads

$$\bar{e}_t + \bar{\boldsymbol{u}} \cdot \nabla \bar{e} \;\; = \;\; \nabla \cdot c_{tke} \nabla \bar{e} - c_b \sqrt{\bar{e}} L N^2 + c_m \sqrt{\bar{e}} L \frac{\partial \bar{u}_i}{\partial x_j}^2 - c_\epsilon \frac{\bar{e}^{3/2}}{L} \tag{26}$$

Another reference for the closure and the choice of the parameters is Denbo and Skyllingstad (1996)

In order to insure a stable integration, diffusivities given by $K_b = c_b\sqrt{\bar{e}}L$ and $K_m = c_m\sqrt{\bar{e}}L$ are set to minimal values corresponding to a minimal value (which can be chosen) of the local grid Peclet number given by $P_e = |\boldsymbol{u}| \min(\Delta x, \Delta z)/K_b$ and $P_e = |\boldsymbol{u}| \min(\Delta x, \Delta z)/K_m$. The minimal P_e should be of the order of 10. A maximal value for the diffusivities in explicit time discretisation is given by $K_{max} = \min(\Delta x, \Delta z)^2/(2\Delta t)$ and is also checked. Output is written to the NetCDF file `tke.cdf`.

4.12 Non-linear viscosity

An alternative LES closure for non-hydrostatic model simulations is given by using a non-linear viscosity according to (Smagorinsky, 1963) Using the switch `enable_smagorinsky_friction`, activates the schemes. Output is written to the NetCDF file `smagorinsky.cdf`.

4.13 Setting a background state

To add the effect of a prescribed background state use the switch `enable_back_stratfication` in the file `./include/options.inc`. This background state is given by a stationary field of buoyancy and zonal or meridional flow (but not meridional and zonal flow) and is defined by the configuration subroutine `set_back_stratification` for which specific examples can be found in the directory `./setup`. Buoyancy, pressure and velocity predicted by the model are perturbations on the specified background state for this formulation.

Note that it is important that the background state is balanced with respect to the underlying equations. Some examples are given in the directory `./setup` and in the beginning of the file `./extra_modules/back_stratification`. In this file, the local preprocessor switches `apply_back_stratification` and `apply_back_meridional_velocity` or `apply_back_zonal_velocity` control the application of the forcing terms due to the background state to the buoyancy and momentum equation, respectively. The local switch `apply_back_convection` determines if the convection parameterisation in case of the hydrostatic approximation, uses the full buoyancy, i.e. perturbation and background buoyancy, or only the perturbation buoyancy to detect static instabilities.

5 Diagnostic output

For each time step, the time, basin averaged kinetic energy and buoyancy content and the number of iterations of the two- and three-dimensional Poisson solver are written to standard output (i.e on the screen). More complex diagnostic output is written in NetCDF format following usual conventions. The output can be easily visualised by e.g. the free software `ferret` available at `http://ferret.wrc.noaa.gov/Ferret`.

5.1 Snapshots

Snapshots of the model variables are written subsequently to a NetCDF file `cpflame.cdf`. Any existing file of this name will be overwritten during model start. The frequency of the output is controlled by the variable `snap_int` in the configuration subroutine `set_parameter` for which specific examples can be found in the directory `./setup`. The value of `snap_int` gives the number of days between subsequent snapshots written to `cpflame.cdf`. It can be fractions of days.

5.2 Passive tracer

The contemporary simulation of one or several passive tracers is implemented in the file `./extra_modules/tracer.F` and can be enabled by the respective switch in the include file `./include/options.inc`. Note that the code has to be recompiled for the switch to be effective. Initial and surface boundary conditions and relaxation zones, sources and sinks, etc of the passive tracer are specified in template subroutines in a file contained in the directory `./extra_setup`. These files are alternatively included at the head of the file `./extra_modules/tracer.F` by using a CPP include directive. Output is written to the NetCDF file `tracer.cdf` with frequency given by `snap_int` in the configuration subroutine `set_parameter`.

Note that at the beginning of the simulation, a file `tracer_restart.dta` will be read if it exists, which contains the last two time steps for the passive tracers from a proceeding simulation with the same configuration. At the end of each simulation, the restart file `tracer_restart.dta` will be overwritten.

5.3 Lagrangian particles

The contemporary simulation of a number of Lagrangian particles (following $d\boldsymbol{x}/dt = \boldsymbol{u}$) is implemented in `./extra_modules/particles.F` and can be enabled by the respective switch in the include file `./include/options.inc`. Note that the code has to be recompiled for the switch to be effective. Initial distribution and total number of the particles are specified locally in the file `./extra_modules/particles.F`. Output is written to the NetCDF file `float.cdf` with frequency given by `snap_int` in the configuration subroutine `set_parameter`.

Note that at the beginning of the simulation, a file `restart_float.dta` will be read if it exists, which contains the last two time steps for the passive tracers from a proceeding simulation with the same configuration. At the end of each simulation, the restart file `restart_float.dta` will be overwritten.

5.4 Zonal averages

Zonal average diagnostic is implemented in `./extra_modules/zonal_averages.F` and can be enabled by the switch `enable_diag_zonalave` in the include file `./include/options.inc`. Note that the code has to be recompiled for the switch to be effective. Several local preprocessor switches are also located in the file `./extra_modules/zonal_averages.F` defining the specific zonally averaged output. Output is written to the NetCDF file `zonal_ave.cdf` with frequency given by `snap_int` in the configuration subroutine `set_parameter`.

5.5 Time averages

Time average diagnostic is implemented in `./extra_modules/time_averages.F` and can be enabled by the respective switch `enable_diag_timeave` in the include file `./include/options.inc`. Note that the code has to be recompiled for the switch to be effective. Several local preprocessor switches are also located in the file `./extra_modules/time_averages.F` defining the specific output. Output is written to the NetCDF file `time_ave.cdf` with frequency given by `snap_int` in the configuration subroutine `set_parameter`.

Note that at the beginning of the simulation, a file `restart_ave.dta` will be read if it exists, which contains the unfinished time averages from a proceeding simulation with the same configuration. At the end of each simulation, the restart file `restart_ave.dta` will be overwritten.

5.6 Overturning streamfunctions

The switch `enable_diag_over` in `./include/options.inc` enables the diagnostic of isopycnal streamfunctions, which are written to the file `over.cdf` with frequency given by `snap_int` in the configuration subroutine `set_parameter`. Note that the code has to be recompiled for the switch to be effective. The code specific to this option can be found in the file `./extra_modules/overturning.F`.

5.7 Peclet, Reynolds, etc numbers

The switch `enable_diag_numbers` in `./include/options.inc` enables the diagnostic of several local and global important numbers, such as Peclet, Reynolds, Courant numbers, which are written in the file `numbers.cdf` with frequency given by `snap_int` in the configuration subroutine `set_parameter`. Note that the code has to be recompiled for the switch to be effective. The code can be found in `./extra_modules/numbers.F`.

Acknowledgments

The conjugate gradient solvers are taken from GFDL MOM-2 and MOM-3, although modified. The same holds for the time type routines in `misc_modules/time_type.F` and many auxilliary routines in `misc_modules/util.F`.

References

Andrews, D. G., J. R. Holton, and C. B. Leovy, 1987: *Middle Atmosphere Dynamics*. Academic Press.

Denbo, D. and E. Skyllingstad, 1996: An ocean large-eddy simulation model with application to deep convection in the Greenland Sea. *J. Geophys. Res.*, **101**, 1095–1110.

Eady, E., 1949: Long waves and cyclone waves. *Tellus*, **1**, 33–52.

Eden, C., 2010: Parameterising meso-scale eddy momentum fluxes based on potential vorticity mixing and a gauge term. *Ocean Modelling*, **32**(1-2), 58–71.

Eden, C., R. J. Greatbatch, and D. Olbers, 2007: Interpreting eddy fluxes. *J. Phys. Oceanogr.*, **37**, 1282–1296.

Eden, C., D. Olbers, and R. J. Greatbatch, 2009: A generalised Osborn-Cox relation. *J. Fluid Mech.*, **632**, 457–474.

Gent, P. R. and J. C. McWilliams, 1990: Isopycnal mixing in ocean circulation models. *J. Phys. Oceanogr.*, **20**, 150–155.

Griffies, S. M., 1998: The Gent-McWilliams skew flux. *J. Phys. Oceanogr.*, **28**, 831–841.

Jones, H. and J. Marshall, 1996: Convection with rotation in a neutral ocean: A study of open-ocean deep convection. *J. Phys. Oceanogr.*, **23**, 1009–1039.

Olbers, D. and C. Eden, 2003: A model with simplified circulation dynamics for a baroclinic ocean with topography. Part I: Waves and wind-driven circulations. *J. Phys. Oceanogr.*, **33**, 2719–2737.

Pacanowski, R. C., 1995: MOM 2 Documentation, User's Guide and Reference Manual. Technical report, GFDL Ocean Group, GFDL, Princeton, USA.

Raasch, S. and D. Etling, 1991: Modeling deep ocean convection: Large eddy simulation in comparison with laboratory experiments. *J. Phys. Oceanogr.*, **28**, 1786–1802.

Smagorinsky, J., 1963: General circulation experiments with the primitive equations. I. The basic experiment. *Mon. Wea. Rev.*, **91**, 164.

Viebahn, J. and C. Eden, 2010: Towards the impact of eddies on the response of the southern ocean to climate change. *Ocean Modelling*, **34**(3-4), 150–165.

Wardle, R. and J. Marshall, 2000: Representation of Eddies in Primitive Equation Models by a PV Flux. *J. Phys. Oceanogr.*, **30**, 2481–2503.