SPFLAME 1.1 documentation

Carsten Eden, IFM–GEOMAR Kiel, Germany

Contents

1 Introduction

This is a short documentation of the SPFLAME code. SPFLAME is a successor of the FLAME code (FLAME Group, 1998) which is an extended version of the GFDL MOM-2.1 code (Pacanowski, 1995). FLAME is an acronym for Family of Linked Atlantic Model Experiments, more details concerning FLAME (configuration, experiments, papers) can be found at *http://www.ifm.uni-kiel.de/fb/fb1/tm/research/FLAME/index.html*. The original MOM-2.1 code, including a very detailed documentation, can be found at *http://www.gfdl.gov/~kd/MOMwebpages/MOMver2.html*. Several authors have worked on revising the original MOM2-1 code, which then became the FLAME code (mostly issues of parallelization and implementation of open boundaries), among them are R. Johannis, R. Redler, J. Dengg and K. Ketelsen. A complete recoding was done for SPFLAME[1], introducing several new features and modules for physical parameterizations. Most of the new features and modules were taken from GFDL MOM-3 (Pacanowski and Griffies, 1999). The main purpose of this text is to provide informations about various namelist variables and switches and a brief overview of the model structure.

However, to start we want to described the major changes with respect to MOM-2. The code utilizes many new Fortran90 features (where useful), all variables are explicitly declared (instead of implicit declarations, which are possible in Fortran, but sometime resemble a fizzy source for errors). The code can be run now in parallel mode, i. e. a horizontal decomposition of the model domain, in zonal as well as in meridional direction, can be integrated in parallel. Communication between different processors (PE's) is realized with the MPI library. Almost all model parameter, e. g. zonal, meridional and vertical extent of the basin, are not hardwired; instead, they are set by namelist input and all arrays are dynamically allocated during runtime. Another major change is the elimination of the "memory window", which was implemented in MOM-2 to reduce the memory size of the code. The elimination of the memory window results in a much easier code structure compared to MOM-2, but, however, in a larger memory requirement. All major arrays reside now in their full dimension in the memory. They are dimensioned for the horizontal and vertical extent for each PE domain, such that changing the decomposition of the domain should not effect the total memory consumption of the model (except for an increase in memory by overlapping parts of the domains and some small (1D and 2D) arrays which are dimensioned

[1] SP was standing originally for Small and Portable with respect to the old code, which was considered to be an advantage. It is however not very small anymore but still portable to some extent.

over the full domain). Some new features of the code are listed below. Note that some of them are taken from MOM-3.1.

- Nested sub domains, integrated in parallel.
- Prognostic, semi–diagnostic or corrected–prognostic model versions (Sheng et al., 2001; Eden and Greatbatch, 2002).
- Explicit free surface (Griffies et al., 2001) or traditional rigid lid formulation.
- Optional partial bottom cells (Pacanowski and Gnanadesikan, 1998).
- Several alternative advection schemes, i. e. Quicker, FCT, 4.th order, upstream or 2.nd order centered differences.
- Flux limiter for passive tracers (Lafore et al., 1998).
- Old Redi (1982) or new Griffies (1998) isopycnal mixing scheme.
- TKE vertical mixing scheme after Gaspar et al. (1990).
- Several alternative approximation to the equation of state.
- NetCDF based preparation routines for generating forcing and setup files.
- Convenient NetCDF or fast binary output.

The text is structured as follows. In section 2, the process of compiling is demonstrated. A "Makefile" is available, allowing an almost automatic compilation. However, some settings have to be made before compilation, accounting for platform dependencies and some preprocessor directives. The nested model concept is briefly introduced as well, since it effects the launching of the model. In section 3 all namelist variables are listed and explained which have to be adjusted for the integration. In section 4 the directory structure for and the source code itself is briefly discussed. Section 5 gives a short outline of the process for preparation of forcing and setup data files, which is also part of this code.

2 Compiling the code

2.1 Makefile

There is a top level Makefile in the main directory (usually "$HOME/spflame") which invokes all preprocessing/ compiling/ linking actions by calling (local) Makefile's in the sub directories in which the actual source code is located. The top level Makefile will include a template ("Makefile_host") with platform dependent configurations, including search paths, compiler options, etc. Currently, tested examples are given in the subdirectory ./configure to run the code on the following systems:

- **Linux** for the Intel f90 (ia32/ia64) compiler on SGI Altix systems, PC–clusters or stand alone Linux PC's (file "configure/Makefile_linux").
- **NEC** for NEC SX5/SX6 cross compiling systems (file "configure/Makefile_sx5")
- **AIX** for IBM AIX systems (file "configure/Makefile_aix").
- **IRIX** for SGI IRIX systems (file "configure/Makefile_irix")
- **CRAY/SGI UNICOS/mk** for CRAY T3E systems (file "configure/Makefile_t3e")
- **SUN** for SunOS systems (file "configure/Makefile_sun")
- **OSF** for DIGITAL UNIX (OSF1) systems (file "configure/Makefile_osf")

Before compiling it is necessary to copy one of these templates to a file named "Makefile_host" in the main directory, which is then included in the Makefiles. It is also necessary to change the value of the variable "SPFLAME" in the top level Makefile, which contains the full path of the main directory in which the code is located (set by default to "$HOME/spflame"). Note that this variable is defined only once in the first few lines of the top level Makefile and is passed calling the local Makefile's in the sub directories. Note that to execute the Makefile's in the sub directories without invoking the main Makefile, it is convenient to create an environment (shell) variable named "SPFLAME" which contains the path of the main directory. After compiling, the executables can be found in the sub directory "./bin".

There are several targets in the top level Makefile. Here are the most important targets:

- **help**: Default. Lists all possible targets.
- **spflame**: Builds the model executable ("spflame.x").
- **prep_routines**: Builds executables for the preparation of forcing files.
- **converter**: Builds executables used for creating NetCDF files from binary output from SPFLAME.
- **tools**: . Builds executables useful in handling output NetCDF files, etc.
- **clean**: Cleaning up all directories after compiling.

2.2 Preprocessor directives

Most of the setup of the model can be controlled by variables and switches set in the namelist file provided during runtime. They are listed in section 3. However, there are also some global preprocessor directives affecting the model setup, which enable or disable different code fragments during compile time at the preprocessing stage. The global directives are listed in the

file "include/options.inc" and can be set therein (not activated directives are, Fortran style–like, commented out in the file) with one important exception: There are directives determining the architecture of the machine for which the code is compiled for. One (and only one) of these directives have to be set. Presently they are set within the platform specific part of the Makefile (see section 2.1) and can be:

- *SX5_host* : NEC SX4, SX5 or SX6.
- *C90_host* : CRAY/SGI T90,C90,SV1
- *T3E_host* : CRAY/SGI T3E.
- *IRIX_host* : SGI IRIX systems.
- *ALPHA_host* : DIGITAL UNIX (OSF1) systems
- *LINUX_host* : Linux with Intel f90 compiler.
- *LINUX_real8_host* : Linux with Intel f90 compiler in double precision.
- *AIX_host* : IBM AIX system
- *SUN_host* : SunOS system
- *SR8000_host* : a Hitachi SR8000 system

All other global directives have to be set in the file "include/options.inc" which is included at the beginning of all other source code files (which can be affected by the directives). Note that by changing the file "include/options.inc" all source code files have to be recompiled (a dependency which is set in the Makefile's). Here is a short list of the directives:

- *no_mpp*
  Disables all calls of the MPI library. In effect, the MPI library is not needed in that case.
- *netcdf_diagnostics*
  Enables NetCDF output. The NetCDF library[2] (version 3.0 or higher) has to be linked in that case.
- *netcdf_real4*
  Using this option, it is enforced to write four bytes long (kind=4) real variables in the NetCDF file. Since the model variables may be at double precision (kind=8), this can interfere with the (single precision) NetCDF library.
- *partial_cell*
  Enables partially filled bottom cells (Pacanowski and Gnanadesikan, 1998).

[2] The NetCDF library can be found at *http://www.unidata.ucar.edu/packages/netcdf*.

- *linear_density*
  Enables a linear equation of state with no pressure dependence.

- *MWJ_density*
  Equation of state given by McDougall, Wright and Jackett (2002, submitted manuscript).

- *timing* and *detailed_timing*
  Gives (detailed) information about the time spent in the major subroutines at the end of the integration.

A single derived directive will be set within "include/options.inc". This is *vector_host* which specifies that certain vectorized code fragments should be used in favor to fragments better suited for scalar codes. Currently, this option will be set only for *SX5_host* and *C90_host*. Note, that also local (with respect to each source code file) directives are used in certain files, mainly for diagnostic purposes. See the corresponding source files for a description.

2.3 Nested model concept

It is possible to run several nested models with increased horizontal and vertical resolution (child domain) in parallel to the main model (mother domain). It is also possible to nest further child domains within the child domains of the mother model. The child domains are driven by open boundary conditions taken from the corresponding mother domains. Optional two–way nesting (feedback of the child domain to the mother domain) is realized by using the semi–prognostic approach, as described in Eden and Greatbatch (2002).

The configuration of each model is determined mainly by the namelist input (see section 3.11). The nested models (child domains) will receive their initial surface and open boundary conditions from the main model (mother domain), no forcing file has to be supplied for child domains. The namelists are read at the start of the integration from a file called "namelist.spflame_XXX" where XXX denotes the number of the domain starting with zero for the mother domain. It is therefore possible to supply different mother/child domains with different setups. The mother/child setup and the region of the nested child is set as well by namelist variables (see section 3).

Each domain is integrated on different processors (PE's), communication between domains is done via calls of the MPI library. In addition, each domain can be horizontally decomposed and the decomposed areas are then integrated by several PE's. The decomposition of each domain can be set via namelist variables "n_pes_i" and "n_pes_j". The actual distribution of all PE's (N) to the domains and the number of the domains (M) are given by command line parameters to the executable (here called *spflame.x*):

mpirun -np N *spflame.x* M "*PE's for first domain*" "*PE's for second domain*" ...

Note that all MPI–application have to be launched with the command "mpirun", which is part of the standard MPI distribution.

3 Namelist variables

A sample file with all namelists containing all variables with their default values can be found in the file "./doc/sample_namelist". All variables are listed in this section in groups with their default values and a short explanation; types of the variables are made obvious. Note that all namelists listed here are read at the start of the integration from a file called **"namelist.spflame_XXX"** where XXX denotes the number of the domain starting with zero for the mother domain. Note also that in many cases, the default values of the namelist variables have to be changed.

In general, the output (to standard out) of the model written as text during runtime is quite detailed and should be carefully checked for each new configuration to insure that the specific configuration is actually met by the namelist input. The text output might as well be useful to understand sources of errors, when the model stops (or crashes) at some point.

3.1 Namelist spflame_basic

This is a list of the most basic model parameter.

- imt = 0
  Zonal extent in grid points of the domain. For child domains, this variable will be calculated internally and overwritten.
- jmt = 0
  Same for the meridional extent.
- km = 0
  Same for the vertical.
- nt = 0
  Number of tracers. Must be greater or equal two.
- n_pes_i = 0
  Number of processors in zonal direction
- n_pes_j = 0
  Same for meridional direction
- time_step= 0.0
  The time step in seconds. For child domains this variable will be set internally and overwritten.
- runlen = 0.0
  The length of the integration in days. For child domains this variable will be set internally and overwritten.

3.2 Namelist spflame_general_setup

Here is a list of namelist switches (almost all Fortran logicals), determing the general setup of the model. These switches are contained in the namelist *spflame_general_setup* . Note that using the default values, many forcing and configuration files have to be provided for the integration, which may by unnecessary for simple (idealized) configurations. In this case it is more convenient to use (and change) the template routines for the model configuration in file "./model/setup_template.F".

- enable_freesurf =.false.
  If true, MOM-3's explicit free surface formulation (Griffies et al., 2001) will be used, otherwise the rigid lid formulation.

- enable_beta_plane =.false.
  If true, a β-plane approximation will be used. The reference value for the Coriolis parameter is given by beta_plane_j0 .

- beta_plane_j0 =1
  Reference grid point (in y-direction) for β or f-plane.

- enable_f_plane =.false.
  If true, a f-plane approximation will be used. The reference value for the Coriolis parameter is given by beta_plane_j0 .

- enable_rotated_grid =.false.
  If true, a spatially dependent Coriolis factor will be used, necessary for a rotated grid. There has to be a file called "glat.dta" containing the (real) latitudes of the unrotated grid on each (rotated) grid point. This file is generated using the preparation routines described in section 5.

- enable_simple_grid =.false.
  If true, a simple grid will be used. The definition of the grid can be set in file "./model/setup _template.F" in subroutine "grid_template". If false, there has to be a file called "grid.dta", prepared beforehand with the grid information. See section 5 for the generation of such a file.

- enable_simple_topo =.false.
  Same for topography and file "kmt.dta" (if option *partial_cell* is set, a file called "htp.dta" will be also referenced). See subroutine "kmt_template" (and "htp_template") in file "./ model/setup_template.F".

- enable_simple_initial_cond =.false.
  Same for initial conditions, file names are "temp.mom.ic","salt.mom.ic" and "tracer_ XXX. ic" for passive tracers. See section 5 for the generation of these files. See subroutine "init_cond_ template" in file "./model/setup_template.F" for simple initial conditions.

- enable_simple_sbc =.false.
  Same for surface boundary conditions. See subroutine "sbc_template" in file "./model/setup_template.F". The filenames are set in subroutine "sbc_initialize" in file "model/bc_io.F". See section 5 how to generate the forcing files.

- enable_simple_obc =.false.
  Same for open boundary conditions, filenames are "obc_north.mom", "obc_west.mom" defined in subroutine "obc_initialize" in file "model/bc_io.F". These files contain information of restoring values for tracers at the open boundary and prescribed streamfunction or surface height. Note that there are currently no templates for simple open boundary conditions, they are simply given by the standard radiation conditions for which no information has to be given. Related variables are enable_obc_north, enable_obc_south, ... restore_TS_obc_north , ... and prescribe_psi_obc_north, ...

- enable_sponge =.false.
  Same for arbitrary number of sponge zones, filenames are "sponge_XXX.mom", defined in subroutine "sponge_initialize" in file "model/bc_io.F" (XXX stands for the number of the sponge zone). See below section 5 how to generate files related to the sponge layers.

- enable_simple_spg =.false.
  In case that no files should be read but template routines should be modified, this variable can be set to true. Corresponding template routines can be found in file "./model/setup_template.F".

- cyclic =.false.
  If true, zonally cyclic boundary conditions are applied. Note that cyclic and eastern or western open boundary conditions are incompatible.

- read_restart =.false.
  If true, the model will read a restart file "restart_XXX.dta" in which XXX stand for the number of the sub domain. If false, initial conditions will be read or constructed.

3.3 Namelist spflame_hor_mixing_setup

This group of namelist switches (all Fortran logicals) controls the configuration of parameterizations for unresolved processes of horizontal nature (mostly of diffusive kind). No attempt is made to explain these parameterizations, but references are given if possible.

- enable_diffusion_harmonic = .false.
  If true, tracers will be mixed harmonically along geopotentials. Note, that the diffusivity is given by variable "ah".

- enable_diffusion_biharmonic = .false.
  Same for biharmonic diffusion, diffusivity is given by "ahbi".

- enable_diffusion_isoneutral = .false.
  If true, tracers will be mixed along isopycnals using the revised scheme by Griffies (1998). Related variables are "ahisop", "athkdf", "ahsteep", "slmx", "del_dm" and "s_dm".

- enable_diffusion_isopycnic = .false.
  Same but using the original Redi/Cox scheme (Redi, 1982) for isopycnal diffusion and bolus advection velocities. Related variables are the same as above, however "ahsteep" is not used.

- enable_friction_harmonic = .false.
  If true momentum will be mixed harmonically. Viscosity is given by "am".

- enable_friction_biharmonic = .false.
  Same but for biharmonic friction, related variable is "ambi".

- enable_friction_cosine_scal = .false.
  Scale "am" and "ambi" with cosine of latitude.

- enable_diffusion_cosine_scal= .false.
  Same for "ah" and "ahbi", but *not* for isoneutral/isopycnal diffusivities.

3.4 Namelist spflame_vert_mixing_setup

This group of namelist switches (all Fortran logicals) controls the configuration of parameterizations for unresolved processes of vertical nature (mostly of diffusive kind). No attempt is made to explain these parameterizations, but references are given if possible.

- enable_implicit_vert_diff = .false.
  If true vertical diffusion will be treated semi–implicitly. No related variables. This scheme will be needed in case of large vertical diffusivities.

- enable_implicit_vert_fric = .false.
  Same for vertical friction.

- enable_const_vert_mixing = .false.
  If true constant vertical diffusivities and viscosities will be used. Related variables are "kappa_h" and "kappa_m". Note that at open boundaries constant values will always be used.

- enable_cgh_vert_mixing = .false.
  If true vertical diffusivity will be calculated following Cummins et al. (1990). Related variables are "cgh_vdcfac" (first entry), "diff_cbt_cut" and "diff_cbt_back".

- enable_cgh_vert_momentum_mixing = .false.
  If true same scheme will be used for momentum. Related variables are "cgh_vdcfac" (second entry), "visc_cbu_cut","visc_cbu_back" and "wndmix".

- enable_cgh_impl_convection = .false.
  If true Cummins et al. (1990) vertical mixing scheme will treat unstable stratification by a very large diffusivity. Related variable is "diff_cbt_cut".

- enable_kpp_vert_mixing = .false.
  Not yet implemented. Not a good idea anyway.

- enable_expl_convection = .false.
  If true, unstable stratifications will be mixed with an explicit scheme by Rahmstorf (1993). Note that this scheme is quite expansive on vector machines.

- enable_impl_convection = .false.
  If true, unstable stratifications will be mixed by using a very high vertical diffusivity. Use this option with constant vertical mixing only. Related variable is "diff_cbt_cut".

- enable_ktmix = .false.
  If true, a Kraus Turner type slab mixed layer model (Sterl and Kattenberg, 1994) will be used for wind stirring. It depends on the use of the explicit convection scheme. Related variables are "effwind" and "z_scale".

- enable_tkemix = .false.
  If true, a TKE closure by Gaspar et al. (1990) will be used to calculate vertical diffusivity and friction. This scheme handles also unstable stratifications. Related variables are not less than "C_eps", "C_kappa", "C_tke", "tke_min", "tke_min_surf", "tke_surf_fac", "nmxl" , "diff_cbt_cut", "diff_cbt_back", "visc_cbu_cut" and "visc_cbu_back".

3.5 Namelist spflame_advection_scheme

This group of namelist switches (all Fortran logicals) specifies the type of advection schemes used for the advection of tracers. The default setting leads to the use of the standard 2nd order centered difference scheme.

- enable_quicker_advection =.false.
  If true, the Quicker advection scheme (Farrow and Stevens, 1995) is used for all tracers instead of centered differences.

- enable_4th_advection =.false.
  If true, a 4th order advection scheme is used for all tracers instead of centered differences.

- enable_fct_advection =.false.
  If true, the FCT advection scheme (Gerdes et al., 1991) is used for all tracers instead of centered differences. Note, that this one is the most expansive (in terms of CPU time) but the only one which is positive definite.

- enable_upstream_advection =.false.
  If true, the simple but highly diffusive upstream advection scheme (Molenkamp, 1968) is used for all tracers instead of 2nd order centered differences. Note that this scheme is highly diffusive.

- enable_flux_delimiter =.false.
  If true, advective and biharmonic diffusive fluxes will be limited for passive tracers (Lafore et al., 1998), to obtain a positive definite advection scheme. This option can be used in combination with the other schemes. Note that passive denotes a tracer which do not affect density.

3.6 Namelist spflame_sbc_setup

See also section 3.2 for switches concerning surface boundary configurations.

- enable_icemask=.false.
  If true, a zero order ice-model will be used. Cooling is permitted only if sea surface temperature is above the freezing point. The other surface tracer fluxes are masked in the same way as the heat flux, while the momentum fluxes are not altered.

- enable_salt_flux_sbc =.false.
  If true the model will use a fixed (virtual) salt flux instead of the usual restoring condition for sea surface salinity. Prepare the file "sss_flux.mom" in that case with the preparation routines.

- enable_shortwave_sbc =.false.
  If true, the model will read a file "qsol.mom" in "model/bc_io.F" which have to generated before with the prep-routines. For that case, shortwave solar radiation is penetrating below the first model level. Related parameters are "efold1_shortwave" "efold2_shortwave" and "rpart_shortwave" accounting for the double–exponential penetration profile of the radiation.

- rpart_shortwave = 0.58

- efold1_shortwave = 35.0e0

- efold2_shortwave = 23.0e2
  A fraction of "rpart_shortwave" of the shortwave radiation penetrates exponentially with a length scale of "efold1_shortwave" the rest with "efold2_shortwave".

3.7 Namelist spflame_obc_setup

At the lateral boundaries of the model domain, open boundary formulations following Stevens (1990) can be used, instead of a solid wall. This group of namelist switches and variables controls the specific configuration.

- enable_obc_north =.false.
  If true, an open boundary will be used at the northernmost latitude. See also "enable_simple_ obc".
- enable_obc_south =.false.
  Same for the south.
- enable_obc_east =.false.
  Same for the east
- enable_obc_west =.false.
  Same for the west
- restore_TS_obc_north =.false.
  If true, temperature and salinity will be restored to data at northern open boundary. This is both for inflow and outflow conditions.
- restore_TS_obc_south =.false.
  Same for the south.
- restore_TS_obc_east =.false.
  Same for the east
- restore_TS_obc_west =.false.
  Same for the west
- prescribe_psi_obc_north =.false.
  If true a barotropic streamfunction profile will be prescribed at the northern open boundary in case of a rigid lid, or a profile of sea surface height for the explicit free surface formulation. Otherwise a radiation condition will be applied. Note that prescribing a (time varying) value for the streamfunction at the open boundary is not strictly correct; the same holds for the radiation condition. This error is, however, ignored.
- prescribe_psi_obc_south =.false.
  Same for the south.
- prescribe_psi_obc_east =.false.
  Same for the east
- prescribe_psi_obc_west =.false.
  Same for the west

- enable_obc_south_sponge =.false.
  If true a biharmonic sponge layer will by implied at the southern open boundary in which the biharmonic friction is scaled according to variables "obc_south_sponge_width", "obc_south_sponge_scale" and "obc_south_sponge_fac". Note that the sponge layer can only work if biharmonic friction is enabled.

- enable_obc_south_sponge_harm =.false.
  Same but here the harmonic (instead of biharmonic) friction is scaled using the same variables.

- enable_obc_south_sponge_diff =.false.
  Together with biharmonic or harmonic viscosity, scale also (z–level) diffusivity in the same manner.

- enable_obc_north_sponge =.false.
  Same as "enable_obc_south_sponge" but for the northern open boundary. Scaled according to variables "obc_north_sponge_ width", "obc_north_sponge_scale" and "obc_north_sponge_fac".

- enable_obc_north_sponge_harm =.false.
  Same but here the harmonic (instead of biharmonic) friction is scaled using the same variables.

- enable_obc_north_sponge_diff =.false.
  Together with biharmonic or harmonic viscosity, scale also (z–level) diffusivity in the same manner.

- obc_south_sponge_width = 45
  Biharmonic sponge at southern open boundary, see code (in file model/gridtopo.F) for a definition.

- obc_south_sponge_scale = 6.0
  Same

- obc_south_sponge_fac = 10.0
  Same

- obc_north_sponge_width = 45
  Biharmonic sponge at northern open boundary, see code (in file model/gridtopo.F) for a definition.

- obc_north_sponge_scale = 6.0
  Same

- obc_north_sponge_fac = 10.0
  Same

3.8 Namelist spflame_blue_setup

This group of namelist switches and variables controls the setup for the semi–prognostic method as recently proposed by Sheng et al. (2001) and Eden and Greatbatch (2002) (refer to both papers for a discussion of the methods).

- enable_blue = .false.
  If true the original semi–prognostic method be will be used, as described by Sheng et al. (2001).

- enable_blue_smooth = .false.
  If true (and either "enable_blue" or "enable_blue_mean" is also true) the smoothed semi–prognostic method (Eden and Greatbatch, 2002) will be used. Related variable is "ismooth_blue".

- enable_blue_tapered = .false.
  If true (and either "enable_blue" or "enable_blue_mean" is also true), the tapered semi–prognostic method of Eden and Greatbatch (2002) will be used. Can be used in combination with "enable_blue_smooth". Related variable is "ilook_blue".

- enable_blue_const = .false.
  If true (and "enable_blue" is also true), constant forcing will be added to momentum instead of flow–interactive forcing of the original semi–prognostic method. This is the corrected–prognostic method of Eden and Greatbatch (2002) and incompatible with "enable_blue_smooth", "enable_blue_tapered" and "enable_blue_mean".

- enable_blue_mean = .false.
  Enables the mean semi–prognostic method, see Eden and Greatbatch (2002). Can be used in combination with enable_blue_smooth and "enable_blue_tapered", but is incompatible with "enable_blue".

- enable_blue_simple = .false.
  Enables simple configuration for a semi–prognostic model, instead of reading files.

- ilook_blue =1
  Length scale in grid points for the horizontal smoothing (moving average) of the difference of climatological density and model density for the semi–prognostic method (Eden and Greatbatch, 2002).

- ismooth_blue=2
  Length scale in grid points for boundary tapering of the tapered semi–prognostic method (Eden and Greatbatch, 2002).

3.9 Namelist spflame_bbl_setup

This namelist input sets up the Bottom Boundary Parameterization (BBL) by Beckmann and Döscher (1997).

- enable_bbl = .false.
  If true, the Bottom Boundary Parameterization (BBL) by Beckmann and Döscher (1997) will be used. Related variables are "ah_sigma" and "ah_sigma_min".
- enable_bbl_advection = .false.
  If true, BBL will be used for advective fluxes.
- enable_bbl_diffusion = .false.
  If true, BBL will be used for diffusive fluxes. Related variables are "ah_sigma" and "ah_sigma _min".
- ah_sigma = 1.e8
  Diffusivity inside the BBL in cm^2/s.
- ah_sigma_min = 1.e1
  Minimal diffusivity inside the BBL in cm^2/s.

3.10 Namelist spflame_solver

These varaiables are needed for the poisson solver for the rigid lid version of the code.

- eps_solver=1.e8
  Epsilon criterion for the Poisson solver. Basin-wide integral of error in $Sv\ m^2t^{-1}$. Convergence of iteration is reached, when the "error" of the solution is less than eps_solver.
- max_itts_solver=500
  Maximum number of iterations for the solver. If convergence is not reached according to the above given epsilon criterion, integration will stop.

3.11 Namelist spflame_nesting_setup

The following namelist variables control the specific configuration for nested models. If there is only one domain (no nested models) the default values of the namelist variables are sufficient for the integration. The concept of nested models is outlined in section 2.3. To control the message passing between the sub domains, each domain becomes an index, starting with 1 for the main domain, which is always present, and increasing subsequently for the child domains.

- mother = -1
  Index for the mother domain of the present domain. A value of -1 means that the present domain have no mother domain. A value greater than zero is the index of the corresponding mother domain.

- nr_childs = 0
  Number of child domains of the present domain. If zero there are no children.

- childs = (0,0,0,0,...)
  List of indices of the child domains of the present domain. The list should be as long as "nr_childs".

- zoom_fac = 0
  Horizontal zoom factor of the present domain, with respect to the horizontal resolution of the mother domain. The factor (an integer) must be greater than 1 and odd.

- is_zoom = 0
  Horizontal T-grid point in mother domain of westernmost point of the present domain.

- ie_zoom = 0
  Same for easternmost point

- js_zoom = 0
  Same for southernmost point

- je_zoom = 0
  Same for northernmost point

- zoom_fac_k = 1
  Vertical zoom factor with respect to the vertical resolution of the mother domain. Can be one, i. e. no enhanced vertical resolution in child domain, but must be odd.

- topo_smooth_iterations = 1
  Number the iterations for smoothing the topography passed from the mother domain to the present domain. 1 is a good value. If mother = -1, no effect.

3.12 Namelist spflame_mixing_parameters

This is a list of model parameters, which have to be supplied for the use of the parameterizations of unresolved processes, as configured in section 3.3 and section 3.4.

- ambi = 2.e19
  Biharmonic viscosity in cm^4/s.

- ahbi = 2.e19
  Biharmonic diffusivity in cm^4/s.

- ah = 2.e7
  Harmonic diffusivity in cm^2/s.

- am = 1.e8
  Harmonic viscosity in cm^2/s.

- ahisop = 1.e7
  Isopycnal diffusivity in cm^2/s.

- athkdf = 1.e7
  Thickness diffusivity in cm^2/s, used for the parameterization of the bolus velocity (Gent et al., 1995).

- ahsteep= 1.e7
  Harmonic diffusivity in cm^2/s used in case of too steep isopycnal slopes. Only used by isoneutral scheme of Griffies (1998).

- slmx = 1.e-2
  Condition for too steep slopes of isopycnals (no units). It gives the maximal allowed slope of isopycnals. In case of too steep slopes, "ahisop" and "athkdf" are tapered to zero according to a tanh(slope)–rule.

- del_dm = 4.e-3
  Transition for the tanh(slope)–rule. No unit but slope is meant. For slopes greater than del_dm, tapering will be effective.

- s_dm = 1.e-3
  Half width scaling for this tanh(slope)–rule. No unit but slope is meant.

- kappa_m = 10.0
  Constant vertical viscosity in cm^2/s. This value is also used at the open boundaries.

- kappa_h = 0.3
  Constant vertical diffusivity in cm^2/s. This value is also used at the open boundaries.

- cgh_vdcfac = (1.e-3,1.e-2)
  First entry gives the proportionality factor to the inverse of the square root of the Brunt-Vaisaelae frequency for the vertical diffusivity used in the vertical mixing scheme by Cummins et al. (1990). Second entry for viscosity.

- diff_cbt_cut = 4.0
  Maximal value for vertical diffusivity in cm^2/s. Values greater than diff_cbt_cut are set to diff_cbt_cut. This variable is used both by the scheme by Cummins et al. (1990) and the TKE closure scheme by Gaspar et al. (1990).

- visc_cbu_cut = 10.0
  Same for vertical viscosity.

- diff_cbt_back = 0.1
  Minimal value for vertical diffusivity in cm^2/s. Values less than diff_cbt_back are set to diff_cbt_back.

- visc_cbu_back = 1.0
  Same for vertical viscosity.

- wndmix = 50.0
  Minimal value of vertical viscosity for the first model level in cm^2/s to simulate high frequency wind stirring in the Ekman layer, used by the vertical mixing scheme by Cummins et al. (1990).

- cdbot =0.0
  Coefficient for a quadratic bottom drag rule.

- tidaloff=0.0
  A constant offset to this bottom drag due to tidal friction effects.

- effwind = 0.8
  Coefficient of effectiveness of wind forcing in the Kraus Turner type scheme.

- z_scale = 5000.0
  Vertical scale length of TKE dissipation in cm in the Kraus Turner type scheme.

- C_eps =0.7
  Coefficient for the Kolmogoroff dissipation in the TKE closure scheme by Gaspar et al. (1990).

- C_kappa =0.1
  Coefficient for vertical eddy viscosity in the TKE closure scheme.

- C_tke =30.0
  Coefficient for vertical TKE viscosity in the TKE closure scheme.

- tke_min =4.e-2
  Background value of TKE in cm^2/s in the TKE closure scheme.

- tke_min_surf =1.0
  Background value of TKE at the surface in cm^2/s in the TKE closure scheme.

- tke_surf_fac =3.0
  Coefficient for input of TKE at surface in the TKE closure scheme.

- nmxl =2
  Flag for different bounds of mixing length (l) in the TKE closure scheme. 0 means l is bounded by distance to surface/bottom, 1 means l is bounded by vertical scale factor and 2 means $|l_z|$ is bounded by level thickness.

3.13 Namelist spflame_diagnostic_setup

Finally, the remaining namelist switches and variables are controlling what is written out to data files during the model integration. Note that for some of the diagnostics, the amount of data written out is also controled by local cpp options in the respective source code (which can by found in subdirectory "./diagnostics". Note also that the format of output data (binary or NetCDF) depends on cpp options set in the file "./include/options.inc".

- enable_snapshots = .false.
  If true, snapshots will be written with frequency "snap_int" to NetCDF file "snap_file". Works only if compiler option "netcdf_diagnostics" is enabled otherwise no snapshots can be written. Use "enable_daily_averages" instead. To specify the data to be written out, local preprocessor directives in file "diagnostics/diag_snap.F" can be set.

- enable_timeseries = .false.
  If true, time series of certain variables will be written with frequency "snap_int" to either a NetCDF file called "anna.cdf" or a binary files called "anna.dta", dependend on the compiler option "netcdf_diagnostics". To specify the data to be written out, local preprocessor directives in file "diagnostics/diag_timeseries.F" can be set.

- enable_ts_monitor = .true.
  If true, the model time, the number of time step, basin-wide averaged kinetic energy and tracer contents will be written to standard out at each time step.

- enable_overturning = .false.
  If true, the meridional overturning streamfunction will be written with frequency "snap_int" to file "over_file", either in binary or in NetCDF format. Binary files can be converted to NetCDF with the program "cv_vsf.F" (make converter).

- enable_heat_tr = .false.
  Same for meridional tracer transports. Written to file "heat_tr_file" with frequency "snap_int" Binary files can be converted to NetCDF with the program "cv_heat_tr.F" (make converter).

- enable_diag_press = .false.
  Same for surface and bottom pressure. Written to file "diag_press_file" in frequency "snap_int". Binary files can be converted to NetCDF with the program "cv_press.F" (make converter).

- enable_diag_float = .false.
  Float diagnostics, written to file "diag_float_file" Binary files can be converted to NetCDF with the program "cv_float.F" (make converter). The deployment region and the number of floats, however, is set directly in the code (see file diagnostics/diag_floats.F).

- enable_annual_averages = .false.
  If true, annual averages will be calculated by summation of model variables (which are currently baroclinic velocity, tracers, surface tracer and momentum fluxes and barotropic mode) at each time step during the integration and written at the end of each year to a file named "averages_XXX_yYYYYmMMdDD.[dta/cdf]" for which XXX denotes the number of the domain, YYYY the year (starting with year 1900), MM the month and DD the day. Note that the file can be either in NetCDF format or binary format, depending on the compiler directive "netcdf_diagnostics". Note also that a normal calendar with equal years will be used for the integration. Binary files can be converted to NetCDF with the program "cv_ave.F" (make converter).

- enable_seasonal_averages = .false.
  Same for seasonal averages. A season is a 3 month interval, the first season is starting at Jan. 1. and is ending at Mar, 31.

- enable_monthly_averages = .false.
  Same for monthly averages. Note that the length of the months differ.

- enable_daily_averages = .false.
  Same for daily averages. Note that daily and annual averages are the only averages with equal periods.

- enable_seasonal_variances = .false.
  Same for seasonal variances. Currently, only velocity variances are calculated in order to obtain estimates of eddy and mean kinetic energy.

- enable_monthly_variances = .false.
  Same for monthly variances.

- enable_show_island_map = .false.
  If true, a map of the islands and island coast lines, used by the Poisson solver, will be shown at standard out.

- enable_diag_blue = .false.
  Diagnostics to diagnose correction term of the semi–prognostic methods. A file "blue_aver ages_XXX_yYYYYmMMdDD.dta" will be written out at the end of each month. Only binary format is used here. The file can be converted with "cv_blue.F" afterwards.

- enable_stability_tests = .false.
  If true, CFL criteria and Peclet numbers will be checked during the integration for each time step. Results are written to standard out. Note that this is very expansive and should be only used for testing purposes.

- enable_diag_zonal = .false.
  Same for zonally averaging related diagnostics. Written to file "zonal_000_yXXXXmXXdXX.cdf" with daily frequency. This is experimental.

- snap_int = 0.0
  Inverse frequency in days for writing various diagnostics.
- snap_file = 'snap.cdf'
  Filename for NetCDF snapshots.
- over_file = 'vsf.cdf'
  Filename for meridional streamfunction time series.
- heat_tr_file = 'heat_tr.cdf'
  Same for meridional tracer transports.
- diag_press_file = 'diag_press.cdf'
  Same for surface and bottom pressure.
- diag_float_file = 'float.cdf'
  Same for floats.
- eps_surf_press = 1.e-4
  Epsilon criterion for Poisson solver when solving for the diagnostic surface pressure.

4 Sub directories and files

All executables are copied to the sub directory "**./bin**", after compiling. In sub directory "**./configure**", architecture depend Makefiles can be found. One of them have to be included in the other Makefiles during compilation, as described above. In "**./doc**" you may find this documentation and a sample namelist file containing all namelist parameter with their default values. The source code files are organized in several sub directories which are "**./model**", "**./diagnostics**", "**./misc_modules**", "**./misc_tools**", "**./mpp**", "**./cv_cdf**" and "**./prep**", listed and described in detail in the following sections.

4.1 ./model

The main model routines are contained in this sub directory. Here is a list of the source code files with a brief description of their content.

- spflame_module.F
  Contains the main module for the model integration, including the namelist input (see below) and all major working arrays. Since in this module almost all important global variables and arrays are defined, an attempt was made to document the variables declaration more extensively than usual. However, note that *all* variables are explicitly declared throughout the code, i. e. no implicit declaration, as possible in Fortran was made in any source code file. Note also that this module, including the declared arrays, is initialized with the initialization routine in this file.

- driver.F
  Contains the main program and the main loop. Note that this is a good starting point to examine and understand the subsequent processing of the main subroutines.

- setup.F
  Contains routines dealing with all things which have to be done before the actual model integration. These routines call all initialization routines of the various modules.

- tracer.F
  Contains the subroutine "tracer" which performs one time step for temperature, salinity or passive tracers.

- clinic.F
  Contains the subroutine "clinic" which performs one time step for velocities (excluding the barotropic mode).

- adv_vel.F
  Contains routines dealing with advection velocities and advective fluxes. Several different advection schemes are implemented.

- rigid_lid.F
  Contains routines necessary to solve for the barotropic streamfunction (in case of a rigid lid). The actual Poisson solver is contained in congrad.F.

- freesurf.F
  If there is a free surface instead of the rigid lid, this file contains routines to explicitly time step the barotropic velocities (Griffies et al., 2001). In that case, no Poisson equation is solved. Note that there are, however, yet unsolved problem with regard to open boundaries.

- vert_mixing.F
  Routines dealing with vertical mixing and the vertical boundary conditions. There are several parameterization for vertical mixing and friction which are enabled or disabled via namelist input.

- dens.F
  Contains a module exporting several routines dealing with the equation of state. Note that these routines are written for best performance on vector architectures. However, they can also be used for other architectures without large loss of performance. Note also that the actual approximation to the equation of state used during the integration depends on the preprocessor options in the file "include/options.inc".

- passive_tracer.F
  All routines which deal with an additional passive tracer should be located here. Implemented are so far CFC11 and CFC12, a pelagic ecosystem model (NPZD) plus oxygen and dissolved inorganic carbon.

- npzd_model.F
  Routines implementing the pelagic ecosystem model (NPZD) plus (optional) oxygen and dissolved inorganic carbon.

- cfc_module.F
  Routines dealing with CFC as passive tracers.

- co2_module.F
  Routines dealing with dissolved inorganic carbon as passive tracers, mainly the calculation of sea surface pCO2.

- bc_io.F
  Contains three modules dealing with IO operations for and temporal interpolation of external data needed for surface boundary condition, open boundary conditions and restoring zones (sponge layers), respectively. For the temporal interpolation and data handling procedures to minimize memory allocation, routines from time_manager.F are used.

- isopycnic.F
  Module implementing isopycnal diffusion of tracer and the bolus advection velocity (Gent et al., 1995). The scheme is using essentially the original code of Redi (1982). Note that the vertical dependence of the isopycnal diffusivity and thickness diffusivity is hard wired in the code at the moment and have to be changed therein.

- isoneutral.F
  Module implementing the revised scheme for isopycnal diffusion and bolus velocity by (Griffies, 1998).

- bbl_module.F
  Module implementing the Bottom Boundary Layer (BBL) parameterization following Beckmann and Döscher (1997).

- setup_template.F
  Contains several template routines, which can be used for simple and/or idealized setups to avoid complicated configuration procedures before the model run. See also namelist file.

- blue.F
  Two modules which deal with the semi–prognostic method as described by Sheng et al. (2001) and Eden and Greatbatch (2002).

- implvmix.F
  This file contains two subroutines implementing implicit vertical mixing and friction, splitted in vector and scalar code.

- congrad.F
  Contains routines for a Poisson solver (conjugate gradient method), needed for the calculation of the time tendency of the barotropic streamfunction and the diagnostic surface pressure.

- stevens_obc.F
  Two routines dealing with the open boundary formulation for tracers.

- gridtopo.F
  Routines to generate grid and topography related variables.

- checks.F
  This file contains one routine which deals with various (but far from complete) checks of setup and parameters.

4.2 ./diagnostics

All the model code which deals with diagnostic operations is located here. Here is a list of the source code files with a brief description of their content. Note that for some of the diagnostics, the amount of data written out is controled by namelist input but also by local cpp options in the respective source code (which can by found in subdirectory "./diagnostics".

- diag.F
  Contains the main diagnostic routine which initialize and calls other routines which implement the various diagnostics. Provides also a routine implementing the time step monitor.

- diag_snap.F
  Contains initialization and actual routine to write snapshots of various model variables to a NetCDF file. Binary snapshot output is not implemented. Note that the amount of data written out, which can be quite extensive, is controlled by local preprocessor directives in the file.

- diag_timeserie.F
  Module implementing a mean to write out time series of model variables, similar to snapshots but more restricted in size and both in NetCDF and binary format (which can be converted to NetCDF afterwards with cv_anna.F). Note that the amount of data written out is controlled again by local preprocessor directives in the file.

- diag_averages.F
  Module for averaging and writing out model variables in either NetCDF or binary format (which can be converted to NetCDF by cv_ave.F afterwards). At the moment only velocity, tracers, surface fluxes and external mode are averaged. Averaging periods can be days, months, seasons or years depending on corresponding namelist variables.

- diag_variances.F
  Module for averaging some second order moments in either NetCDF or binary format (which can be converted to NetCDF by cv_var.F afterwards). At the moment only velocity is averaged (for estimates of mean and eddy kinetic energy). Averaging period is either one month or one season (3 months).

- diag_heat_tr.F
  Routines to diagnose (zonally and vertically integrated) northward tracer transports. Unlike all other routines in this sub directory (which are called in diag.F) this one is called inside tracer.F since the advective fluxes of tracer are needed. Format is either NetCDF or binary (which can be converted to NetCDF by cv_gyre.F afterwards). Note that the binary format corresponds to the original format of MOM-2.

- diag_over.F
  Routines to calculate and write the meridional (overturning) streamfunction. Format is either NetCDF or binary (which can be converted to NetCDF by cv_vsf.F afterwards). Note that the binary format corresponds to the original format of MOM-2.

- diag_press.F
  Routines to diagnose the surface pressure (in the rigid lid formulation) and the hydrostatic pressure at the bottom (vertical integral of density). Format is either NetCDF or binary (which can be converted to NetCDF by cv_press.F afterwards).

- diag_stab.F
  Routine calculating some numerical stability conditions, such as CFL and Peclet Number. The output is to standard out only at the moment.

- diag_float.F
  Module to calculate neutrally buoyant floats. Output format is either NetCDF or binary (which can be converted to NetCDF by cv_float.F afterwards). The deployment region for the floats have to be set in the source code file. Note that it is also possible to calculate the float in an offline mode.

- diag_netcdf.F
  Some routines useful to deal with the NetCDF library.

- diag_blue.F
  Diagnostic routines concerning the semi–prognostic method.

- diag_zonalave.F
  experimental

- diag_diapycnic.F
  experimental

- diag_barbi.F
  experimental

4.3 ./mpp

All code dealing with parallel processing using the MPI library.

- sub_mpif.h
  Simple include file, used by all other routines in this sub directory. It is needed to wrap float and integer definitions of the MPI standard to proper formats on the various platforms. It will also include the include file "mpif.h" of the MPI standard distribution.

- mpp_basic.F
  Basic routines, necessary for message passing inside the decomposed region of the domain.

- mpp.F
  Some more elaborated routines concerning message passing between regions inside one domain.

- domain_exchg.F
  Routines used for message passing between the domains.

4.4 ./misc_modules

This sub directory contains all modules and utility files which are of a more general purpose and which do not depend on other modules.

- c_helpers.c
  Contains a single C routine which re–directs the standard output to a file. This can be necessary to prevent confusion, since all processors of one MPI application are sharing the same standard output. Here, the re–direction is currently used only for the nested sub domains, since for each domain, only the first processor (rank zero) is usually writing to standard output. Note that the interface for calling this routine from Fortran subroutines depends on the specific compiler.

- island_modules.F
  This module contains island and island perimeter mapping routines, which generate informations concerning the island location and path integrals which are needed for the Poisson solver solving for the barotropic streamfunction. The original code was taken from GFDL MOM2.1. The module exports only two subroutines, named isleperim and showmap. The first essentially takes the topography information performs some checks on the landmask and generates some arrays containing the island path integrals and a map of the landmask. The latter prints the map of the landmask.

- time_manager.F
  This modules supplies routines for the time management. It is meant to handle simple time stepping schemes with an Euler forward or backward time step between several leapfrog time steps. Additionally, it supplies routines for interpolation between time averages of forcing functions driving the model, as e.g. surface boundary conditions. It depends on the module time_type.F. Note that this module has to be initialized using the initialization subroutine of this module.

- time_type.F
  This module supplies a type definition for time and is taken from MOM3.0. It is possible to use different types of calendars. The default calendar is a year with 365 days (no leap years) and months of different length. This module is used by time_manager.F.

- timing.F
  This module supplies simple subroutines for measuring the elapsed user cpu time between different calls of these routines. Note that the appropriate routine which gives the actual elapsed cpu time depends on the platform.

- util.F
  Here are all routines collected which do not fit into other categories but are of general purpose. Among them are routines for reading command line parameters, manipulating character variables, finding free IO units and performing special open statements. There are also routines for interpolation, box averaging and extrapolation of data, used during the preparation of forcing files as described in section 5.

4.5 ./misc_tools

Some useful stand alone programs of general purpose. Can be compiled with "make tools" in main directory and found in subdirectory "./bin" after compilation.

- nc_ave.F
  To average a variable over several netcdf files, e.g. averaging 12 monthly means to an annual mean. Syntax can be infered by typing "nc_ave.x" only.

- nc_cat.F
  To concatenate a netcdf variable in several files along the last dimension of the variable (usually the time coordinate). Syntax can be infered by typing "nc_cat.x" only.

- nc_merge.F
  To merge netcdf variables in several files in one file. Syntax can be infered by typing "nc_merge.x" only.

- sum_var.F
  Simple program which sums files written (in binary format) by subroutine diag_variances. Useful to sum monthly averages to seasonal or annual averages. Syntax can be infered by typing "sum_var.x" only.

- nc_util.F
  Collection of useful subroutines, used by nc_ave.F, etc.

4.6 ./cv_cdf

Some useful stand alone programs converting binary output of a model integration to NetCDF files. Can be compiled with "make converter" in main directory and found in subdirectory "./bin" after compilation.

- cv_anna.F
  To convert the output of diag_timeserie.F.
- cv_ave.F
  To convert output of diag_averages.F.
- cv_var.F
  To convert output of diag_variances.F.
- cv_gyre.F
  To convert output of diag_heat_tr.F.
- cv_press.F
  To convert output of diag_press.F.
- cv_vsf.F
  To convert output of diag_over.F.
- cv_float.F
  To convert output of diag_float.F.
- cv_blue.F
  To convert output of diag_blue.F.
- cv_blue_mean.F
  To convert some diagnostic output of written in file blue.F, which is, as an exception, not done in code contained in sub directory ./diagnostics.

4.7 ./prep

Code which generates forcing files for model integrations. See also section 5.

- prep_main.F
  Main program for preparation of forcing files.
- prep_grid.F
  Contains routines to generate a (B) grid. Called by prep_main.F.
- prep_topo.F
  Contains routines to generate the model topography grid. Called by prep_main.F.

- prep_out.F
  Contains routines to write interpolated forcing data and configuration information to a NetCDF file. Called by prep_main.F.

- prep_template.F
  Contains template routines (by including other source code files, see below) to read from binary or netcdf files containing necessary datasets, such as topography, hydrographic climatologies, etc. The template routines are called in prep_main.F.

- prep_template_topo.F
  Contains the template routines to read topography dataset. This file is included by prep_template.F.

- prep_template_ic.F
  Contains the template routines to read datasets containing (monthly) climatologies for the tracers used for the intial conditions, restoring zones, salinity restoring and the the open boundaries. Included by prep_template.F.

- prep_template_tau.F
  Same for the wind stress.

- prep_template_sflx.F
  Same for the surface tracer flux formulations.

- prep_template_obcpsi.F
  Same for the values of the streamfunction applied at the open boundaries.

- prep_template_sponge.F
  Contains routines to specify restoring zones for tracer (sponge layers) in the model domain. Called by prep_main.F and included in prep_template.F.

- prep_fold.F
  A program to fold the model domain. See section 5 for details.

- prep_killfilt.F
  A program to "Killworth–filtering" (Killworth, 1996) some forcing data. See section 5 for details.

- prep_to_bin.F
  A program to read the information contained in the NetCDF file written by prep.F, etc and to convert to binary files used during the model integration. See also section 5 for details.

- map.m
  Matlab script to manipulate model topography.

- prep_blue_const.F
  A program to convert data necessary for a semi–prognostic odel version (corrected–prognostic).

5 Preparation of forcing and setup files

5.1 Overview

The simplest and easiest way to run the model is by using (and changing) the template setup routines, given in file "model/setup_template.F", together with the proper setting corresponding namelist variables of section 3.2. No special forcing and setup files have to be generated beforehand in that case. However, a more realistic setup demands the preparation of some setup and forcing files before the model run. This can be done using preparation routines, which are shortly discussed in this section.

The main program *prep.x* constructs a grid, a topography mask, surface and boundary forcing as well as initial conditions. Datasets of realistic topography, climatological hydrography and forcing functions have to be provided by the user, but can be supplied on request. The raw data will be interpolated on the model grid and written to a NetCDF file by the main preparation routine. Afterwards, the model grid can be folded[3] to reduce the extent of the model domain using the program *fold.x*. Additionally, some of the forcing file can be "Killworth–filtered" (Killworth, 1996) using the program *killfilt.x*. For the actual model run, the forcing and setup files have to be converted from NetCDF format to binary files with *to_bin.x*. The preparation routines can be build by typing "make prep_routines". Four executables are generated: *prep.x*, *fold.x*, *killfilt.x* and *to_bin.x*. They can be found in the sub directory "./bin".

The program "prep.x" is build in the sub directory "./prep" from several routines (see section 4). The program reads a namelist from its standard input, called "prep" (note that the command to execute the program looks therefore like "./prep.x < namelist_file"). The default values for this namelist are given in Table 1. All variables are subsequently discussed in the course of this section. However, *prep.x* is meant to be run in two steps: The first step is to generate the grid and the raw topography mask. In a second step the forcing functions are generated. The raw topography should be manipulated between both steps somehow. The prevent the program to proceed to the forcing function generation, the logical *stop_after_topo* can be set to true. The program will write grid and topography (together with the undiscretized topography) to a NetCDF file named "forcing.cdf" to be viewed and to binary files "new_grid.dta" and "new_kmt.dta" as well. The binary file for the topography is subject to manipulation between both steps. A Matlab script called "map.m" in sub directory "prep" is provided to manipulate the topography interactively. It works currently for Matlab version 5.3. If the topography appears ready, the binary files are copied to files called "old_grid.dta" and "old_kmt.dta". Setting the namelist switches *stop_after_topo* to false and *read_grid_topo* to true, these "old" files are read by the program and overriding the initial interpolation results. Afterwards the program is proceeding to the interpolation of the forcing functions.

The text output written to standard out by "prep.x" (and the other programs) during the execution is rather detailed and one should carefully read it, to check whether the program is

[3] "Folding" a model domain means, moving parts of the ocean zonally to land covered regions, where possible, and using cyclic zonally boundary conditions.

nxlon	no default
nylat	no default
nzdepth	no default
x_lon	no default
dx_lon	no default
y_lat	no default
dy_lat	no default
z_depth	no default
dz_depth	no default
isotropic	false
flame_vert_grid	false
cyclic	false
gltnp	no default
glnnp	no default
gltpp	no default
glnpp	no default
topo_smooth_ntimes	0
kmt_min	4
enable_obc_north	false
enable_obc_south	false
enable_obc_east	false
enable_obc_west	false
read_grid_topo	false
stop_after_topo	false
rotated_grid	true
enable_blue	false
enable_blue_mean	false
number_tr	2

Table 1: Table with namelist "prep"

actually setting up the desired configuration, reading the correct data files, etc. If something goes wrong, e. .g. during the interpolation of the raw data on the model grid, the text output of "prep.x" is as well useful to understand the reason for the failure.

5.2 Setting up the grid

The grid generation follows the method described in Pacanowski (1995) (Chapter 5 therein). Here, this procedure is briefly reviewed. The integers *nxlon*, *nylat* and *nzdepth* in the namelist "prep" specify the number of regions (plus one) with different zonal, meridional or vertical grid spacing. The real (vector) variables *dx_lon(nxlon)*, *dy_lon(nylat)* and *dz_depth(nzdepth)* specify the different grid spacings at the border of each region, in degrees longitude, latitude and centimeters, respectively. The positions of the borders of the regions are given by the real vectors *x_lon(nxlon)*, *y_lat(nylat)* and *z_depth(nzdepth)* in the same units as above.

A simple example: To construct an equidistant grid, one would set *nxlon*, *nylat* and *nzdepth* to 2, *dx_lon*, *dy_lon* and *dz_depth* to the desired zonal, meridional and vertical resolution and *x_lon*, *y_lat* and *z_depth* to the starting and ending positions of the grid. Clearly, it is possible to define regions with different resolution by setting the integers to values greater than two and the remaining variables to appropriate values. Note, that it is not always possible to contructed the desired grid. The program will report the reason and stop in that case.

Additionally, the horizontal grid can be rotated using the (real) variables *gltnp*, *glnnp*, *gltpp* and *glnpp*, where the first two variables denote the longitude and latitude of the rotated north pole in the normal geographical coordinates, and the latter two the coordinates of the prime point, determing the zeroth meridian. Using *gltnp=90.0*, *glnnp=90.0*, *gltpp=0.0* and *glnpp=0.0* means no rotation at all The file "glat.dta" which is needed for the model run, if using a rotated grid, will be generated at the end of the procedure, see below.

Setting the logical *isotropic* to true, leads to an "isotropic" grid, which means that the latitudinal grid spacing matches the longitudinal grid spacing (in e. g. centimeters) approximately. Note that this option makes less sense on a rotated grid. Setting the logical *cyclic* means applying cyclic zonal boundary conditions, useful for a global grid.

5.3 Setting up the topography

To interpolate a topography mask onto the generated grid it is necessary to have a dataset of the realistic depths of the oceans. In file "./prep/prep_template_topo.F" two routines deal with reading such a dataset: "read_rose_dim" and "read_rose". The first is called by the main program to determine the zonal and meridional extent of the dataset, the latter to read the actual data. Currently these routines read a NetCDF file containing the ETOPO5 (1988) dataset and it is assumed that the NetCDF file is named "./data/etopo5/etopo5.cdf" relative to the local directory in which "prep.x" is exectuted. The user may want to modify these routines for his needs.

The dataset is interpolated (or box averaged) horizontally and dicretized vertically onto the model grid. Some namelist parameters apply to this procedure: *kmt_min* is an integer and denotes the minimal value of vertical levels for shallow regions (4 is safe, 2 is adventurous, 1 is not possible). *topo_smooth_ntimes* denotes the number of smoothing operations applied to the topography mask. Note that a smooth topography is mandatory to obtain smooth model results. The smoothing operation uses a simple 2–dimensional symmetric 3–point filter based on the weighting 1/4, 1/2, 1/4. Specifying the logicals *cyclic*, *enable_obc_north*, *enable_obc_south*, *enable_obc_east* and *enable_obc_west* appropriately, the program is able to handle the open boundary or cyclic boundary conditions correctly for the topography. The generated topography mask must be most likely edited. Use the Matlab script "map.m".

5.4 Setting up the forcing functions

Having successfully generated a grid and a topography, datasets for the forcing functions are needed. First a monthly climatology of temperature and salinity, covering the model domain, has to be provided. This climatology is needed for the initial conditions as well as for sponge layers and open boundary conditions (it may serve also for the (restoring) boundary condition for salinity). In the file "./prep/prep_template_ic.F" routines "read_ic" and "read_ic_dim" provide the informations to the main program as before for the topography. Several versions of both routines have been implemented in "./prep/prep_template_ic.F". Which one is actually selected depends on preprocessor settings in the file "./prep/prep_template.F" (first section). Currently, reading from three different datasets is implemented: A global (1x1deg) monthly mean climatology by Levitus and Boyer (1994), a high resolution (1/4x1/4 deg) monthly mean climatology given by a combination of the datasets of Levitus and Boyer (1994) and Boyer and Levitus (1997) spanning the Atlantic domain only, and a global, annual mean climatology as given by WOCE SAC. It is assumed that the file can be found directories named "./data/WOA94","./data/WOA9497" and "./data/SAC" respectively, relative to the local directory in which "prep.x" is exectuted. For more details can be refered in "./prep/prep_template_ic.F". These routines may have to be modified by the user. Note that a temporary binary file "ts.dta" is generated by subroutine "prep_ic" in "prep_main.F" during the interpolation of the initial conditions.

Then, the same procedure is needed for the wind stress datasets (see routines "read_dims_tau" and "read_tau" in file "prep_template_tau.F" and corresponding preprocessor options in "prep_template.F") for the surface tracer fluxes (see routines "read_tr" and "read_dims_tr" in file "prep_template_sflx.F" and corresponding preprocessor options in "prep_template.F") and the friction velocity[4] (see routines "read_ustar" and "read_dims_utar" in file "prep_template_tau.F" and corresponding preprocessor options in "prep_template.F"). Currently, reading from a variety of different monthly mean and daily or monthly mean time series of wind stress climatologies is implemented, the same holds for the surface fluxes.

[4]Friction velocity denotes the input of turbulent kinetic energy at the surface and is needed for the TKE closure scheme by Gaspar et al. (1990) and the Kraus Turner mixing scheme.

After the generation of initial conditions and surface boundary conditions the program tries to set up sponge layers and forcing function for open boundary conditions using the previously interpolated fields for monthly climatological temperature and salinity. Setting up sponge layers appears highly model dependent, thus everything is included in a separate file "./prep/prep_template_sponge.F". Several examples are given in that file how to set up a sponge layer, depending on the prepsocessor options in "prep_template.F". The open boundary conditions need one last input: the prescribed monthly mean values for streamfunction (or free surface elevation). Again template routines are given in "./prep/prep_template_obcpsi.F" for that purpose.

5.5 Folding, "Killworth–filtering" and conversion to binary files

Compiling and running the executable "prep.x" supplemented by the namelist input at the second step of the procedure, will produce a NetCDF file called "forcing.cdf". The model domain can now be folded with the program "fold.x" build from the source code file "prep/prep_fold.F". Note that is not possible to fold, if there are eastern or western open boundaries or if the model resides on a rotated grid. The program "fold.x" is fully automatic, but can take a single command line parameter, denoting the new zonal model extent in grid points. If this parameter is not given the program will determine and use the smallest possible value for the zonal extent. Any data and information of the unfolded grid contained in "forcing.cdf", is transfered be "fold.x" to the folded grid and written to a file called "forcing_folded.cdf".

Is is possible to "Killworth–filter" (Killworth, 1996) some of the data in "forcing_folded.cdf" with the program "killfilt.x" build from the source code file "prep/prep_killfilt.F'. This is mainly done for the wind stress forcing, but review the source code file.

Finally, at the end of the procedure, the program "to_bin.x" generated from the source code contained in "prep/prep_to_bin.F" will convert the data contained now in the file "forcing_folded.cdf" to several binary files, which are read by the model at the start and during each model integration. Note that the binary format of the resulting forcing files depends now on the specific system.

References

Beckmann, A. and R. Döscher, 1997: A method for improved representation of dense water spreading over topography in geopotential-coordinate models. *J. Phys. Oceanogr.*, **27**, 581–591.

Boyer, T. P. and S. Levitus, 1997: Objective analyses of temperature and salinity for the world ocean on a 1/4 degree grid. Technical report, NOAA Atlas NESDIS 11, U.S. Gov. Printing Office, Washington, D.C.

Cummins, P. F., G. Holloway, and A. E. Gargett, 1990: Sensitivity of the GFDL ocean general circulation model to a parameterization of vertical diffusion. *J. Phys. Oceanogr.*, **20**, 817–830.

Eden, C. and R. J. Greatbatch, 2002: Adiabatically correcting an eddy–permitting North Atlantic model using large–scale hydrographic data. *J. Climate*. Submitted.

ETOPO5, 1988: Digital relief of the surface of the earth. Worldwide bathymetry/topography data annoucenment 88-MGG-02. Technical report, National Geophysical Data Center, NOAA, Code E/GC3, Boulder, CO.80303-3328.

Farrow, D. E. and D. P. Stevens, 1995: A new tracer advection scheme for bryan and cox type ocean general circulation models. *J. Phys. Oceanogr.*, **25**, 1731–1741.

FLAME Group, 1998: FLAME- a Family of Linked Atlantic Model Experiments. Technical report, AWI Bremerhaven, Germany.

Gaspar, P., Y. Gregoris, and J.-M. Lefevre, 1990: A simple eddy kinetic energy model for simulations of the oceanic vertical mixing: tests at station PAPA and Long-Term Upper Ocean Study site. *J. Geophys. Res.*, **95**, 16179–16193.

Gent, P. R., J. Willebrand, T. J. McDougall, and J. C. McWilliams, 1995: Parameterizing eddy-induced tracer transports in ocean circulation models. *J. Phys. Oceanogr.*, **25**, 463–474.

Gerdes, R., C. Köberle, and J. Willebrand, 1991: The influence of numerical advection schemes on the results of ocean general circulation models. *Climate Dyn.*, **5**, 211–226.

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

Griffies, S. M., R. Pacanowski, M. Schmidt, and V. Balaji, 2001: Tracer conservation with an explicit free surface method for z-coordinate ocean models. *mwr*, **129**, 1081–1098.

Killworth, P. D., 1996: Time interpolation of forcing fields in ocean models. *J. Phys. Oceanogr.*, **26**, 136–143.

Lafore, J., J. Stein, N. Asencio, P. Bougeault, V. Ducrocq, J. Duron, C. Fischer, P. Hereil, P. Mascart, V. Masson, J. L. Pinty, J. P. Redelsberger, R. E., and V. G. de Arellano, 1998: The Meso–NH atmospheric simulation system. Part 1: adiabatic formulation and control simulations. *Ann. Geophys.*, **16**, 90–109.

Levitus, S. and T. P. Boyer, 1994: World Ocean Atlas 1994. Technical report, NOAA, U.S. Gov. Print. Off., Washington D.C,USA.

Molenkamp, C. R., 1968: Accuracy of finite–difference methods applied to the advection equation. *J. Appl. Meteor.*, **7**, 160–167.

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

Pacanowski, R. C. and A. Gnanadesikan, 1998: Transient response in a z-level ocean model that resolves topography with partial-cells. *Mon. Wea. Rev.*, **126**(12), 3248–3270.

Pacanowski, R. C. and S. M. Griffies, 1999: The MOM 3 manual. Technical report, Geophysical Fluid Dynamics Labaratory, Princeton, NJ, USA.

Rahmstorf, S., 1993: A fast and complete convection scheme for ocean models. Ocean Modelling, unpublished manuscript.

Redi, M. H., 1982: Oceanic isopycnal mixing by coordinate rotation. *J. Phys. Oceanogr.*, **12**, 1154–1158.

Sheng, J., R. J. Greatbatch, and D. Wright, 2001: Improving the utility of ocean circulation models through adjustment of the momentum balance. *J. Geophys. Res.*, **106**, 16711–16728.

Sterl, A. and A. Kattenberg, 1994: Embedding a mixed layer model into an ocean general circulation model of the Atlantic: The importance of surface mixing for heat flux and temperature. *J. Geophys. Res.*, **99**, 14,139–14,157.

Stevens, D. P., 1990: On open boundary conditions for three dimensional primitiv equation ocean circulation models. *Geophys. Astrophys. Fluid Dyn.*, **51**, 103–133.