Abstract

“Tracers in Ocean Paradigm” (TOP) is the passive tracers engine of the NEMO ocean model (“Nucleus for European Modelling of the Ocean”). It is intended to be a flexible tool for studying the on/offline oceanic tracers transport and the biogeochemical processes (“green ocean”), as well as its interactions with the other components of the Earth climate system over a wide range of space and time scales. TOP is interfaced with the NEMO ocean engine, and, via the OASIS coupler, with several atmospheric general circulation models. This component provides the physical constraints and boundaries conditions for oceanic tracers transport and represents a generalized, hardwired interface toward biogeochemical models to enable a seamless coupling. In particular, transport dynamics are supplied by the ocean dynamical core thus enabling the use of all available advection and diffusion schemes in both on- and off-line modes. TOP is designed to handle multiple oceanic tracers through a modular approach and it includes different sub-modules: ocean water age, inorganic carbon (CFCs) & radiocarbon (C14b), built-in biogeochemical model (PISCES), and prototype for user-defined cases or coupling with alternative biogeochemical models (e.g., BFM).
Disclaimer

Like all components of the modelling framework, the TOP core engine is developed under the CECILL license, which is a French adaptation of the GNU GPL (General Public License). Anyone may use it freely for research purposes, and is encouraged to communicate back to the development team its own developments and improvements.

The model and the present document have been made available as a service to the community. We cannot certify that the code and its manual are free of errors. Bugs are inevitable and some have undoubtedly survived the testing phase. Users are encouraged to bring them to our attention.

The authors assume no responsibility for problems, errors, or incorrect usage of NEMO.

Other resources

Additional information can be found on:

- the website of the project detailing several associated applications and an exhaustive users bibliography
- the development platform of the model with the code repository for the shared reference and some main resources (wiki, ticket system, forums, . . . )
- the repository of the demonstration cases for research or training
- the online archive delivering the publications issued by the consortium (manuals, reports, datasets, . . . )
- two mailing lists: the newsletter for top-down communications from the project (announcements, calls, job opportunities, . . . ) and the forge updates (commits, tickets and forums)

Citation

Reference for papers and other publications is as follows:

# List of Figures

0.1. A schematic view of NEMO-TOP component .............................................. ii

1.1. Atmospheric CFC11, CFC12 and SF6 partial pressure evolution in both hemispheres. .......... 6
1.2. CFC11 solubility in mol m$^{-3}$ pptv$^{-1}$, calculated from the World Ocean Atlas 2013 temperature and salinity annual climatology. ........................................ 6
1.3. CFC11 vertical inventory in $\mu$mol m$^{-2}$, from one of the UK Earth System Model 1 model (UKESM1 - which uses NEMO as ocean component, with TOP for the passive tracers) historical run at year 2000. . 7
1.4. Time evolution of $^{14}$R inventory anomaly for equilibrium run with homogeneous ocean initial state. The anomaly (or drift) is given in % change in total ocean inventory per 50 years. Time on x-axis is in simulation year. ............................................................ 9
1.5. Atmospheric $\Delta^{14}$C (solid; left axis) and CO$_2$ (dashed; right axis) forcing for the $^{14}$C-bomb experiments. The $\Delta^{14}$C is illustrated for the three zonal bands (upper, middle, and lower curves correspond to latitudes $> 20$N, $\in [20$S, $20$N], and $< 20$S, respectively). ...................................................... 9
1.6. Atmospheric $\Delta^{14}$C (solid) and CO$_2$ (dashed) forcing for the Paleo experiments. The CO$_2$ scale is given on the right axis. ................................................................. 10
List of Namelists
1.1. Coefficients for fit of the CFCs solubility (Eq. 1.12). ........................................... 6
1.2. Coefficients for fit of the CFCs Schmidt number (Eq. 1.11). ................................. 6
1.3. Standard output fields for the C14 package. .............................................................. 11
TOP (Tracers in the Ocean Paradigm) handles oceanic passive tracers in NEMO. At present, this component provides the physical constraints and boundaries conditions for oceanic tracers transport and represents a generalized, hardwired interface toward biogeochemical models to enable a seamless coupling.

It includes three independent components:

- a transport code TRP sharing the same advection/diffusion routines with the dynamics, with specific treatment of some features like the surface boundary conditions, or the positivity of passive tracers concentrations
- sources and sinks - SMS - models that can be typically biogeochemical, biological or radioactive
- an offline option which is a simplified OPA 9 model using fields of physics variables that are previously stored to disk

There is two ways of coupling TOP to the dynamics:

- **online coupling**: the evolution of passive tracers is computed along with the dynamics
- **offline coupling**: the fields of physics variables are read from files and interpolated at each model time step, with no constraints on the time sampling in the input files

TOP is designed to handle multiple oceanic tracers through a modular approach and it includes different sub-modules:

- the ocean water age module (AGE) tracks down the time-dependent spread of surface waters into the ocean interior
- inorganic carbon (e.g. CFCs, SF6) and radiocarbon (C14) passive tracers can be modeled to assess ocean absorption timescales of anthropogenic emissions and further address water masses ventilation
- a built-in biogeochemical model (PISCES) to simulate lower trophic levels ecosystem dynamics in the global ocean
- a prototype tracer module (MY_TRC) to enable user-defined cases or the coupling with alternative biogeochemical models (e.g. BFM, MEDUSA, ERSEM, BFM, ECO3M)
TOP : Tracers in the Ocean Paradigm

Figure 0.1: A schematic view of NEMO-TOP component
# Contents

1. Model Description ....................................................... 1
   1.1. Basics ......................................................... 1
   1.2. The NEMO-TOP interface ........................................ 2
   1.3. The transport component: TRP .................................. 2
       1.3.1. Advection ............................................... 2
       1.3.2. Lateral diffusion ...................................... 2
       1.3.3. Tracer damping ........................................ 3
       1.3.4. Tracer positivity ....................................... 3
   1.4. The SMS modules .................................................. 3
       1.4.1. Ideal Age ............................................... 3
       1.4.2. Inert carbons tracer ..................................... 4
       1.4.3. Radiocarbon ............................................. 5
       1.4.4. PISCES biogeochemical model ............................. 11
       1.4.5. MY_TRC interface for coupling external BGC models .... 12
   1.5. The Offline Option ............................................... 12
2. Model Setup ..................................................................... 14
   2.1. Setting up a passive tracer configuration ....................... 14
   2.2. TOP Tracer Initialisation ......................................... 15
   2.3. TOP Boundaries Conditions ....................................... 15
3. Miscellaneous .................................................................... 16
   3.1. TOP synthetic Workflow .......................................... 16
       3.1.1. Model initialization ....................................... 16
       3.1.2. Time marching procedure .................................. 16
   3.2. Coupling an external BGC model using NEMO framework ........ 16
4. Coding Rules .................................................................... 17
   A.1. Introduction ......................................................... 18
   A.2. Overview and general conventions .................................. 18
   A.3. Architecture ........................................................ 19
   A.4. Style rules ............................................................ 19
       A.4.1. Argument list format ...................................... 19
       A.4.2. Array syntax ............................................... 19
       A.4.3. Case ............................................................ 19
       A.4.4. Comments .................................................... 20
       A.4.5. Continuation lines ......................................... 20
       A.4.6. Declaration of arguments and local variables ............. 20
       A.4.7. F90 Standard ............................................... 20
       A.4.8. Free-Form Source .......................................... 20
       A.4.9. Indentation .................................................... 21
       A.4.10. Loops .......................................................... 21
       A.4.11. Naming Conventions: files ................................. 21
       A.4.12. Naming Conventions: modules ............................. 21
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.4.13. Naming Conventions: variables</td>
<td>21</td>
</tr>
<tr>
<td>A.4.15. Pre processor</td>
<td>21</td>
</tr>
<tr>
<td>A.5. Content rules</td>
<td>22</td>
</tr>
<tr>
<td>A.5.1. Configurations</td>
<td>22</td>
</tr>
<tr>
<td>A.5.2. Constants</td>
<td>22</td>
</tr>
<tr>
<td>A.5.3. Declaration for variables and constants</td>
<td>22</td>
</tr>
<tr>
<td>A.5.4. Headers</td>
<td>23</td>
</tr>
<tr>
<td>A.5.5. Interface blocks</td>
<td>23</td>
</tr>
<tr>
<td>A.5.6. I/O Error Conditions</td>
<td>23</td>
</tr>
<tr>
<td>A.5.7. PRINT - ASCII output files</td>
<td>23</td>
</tr>
<tr>
<td>A.5.8. Precision</td>
<td>24</td>
</tr>
<tr>
<td>A.5.9. Structures</td>
<td>24</td>
</tr>
<tr>
<td>A.6. Packages coding rules</td>
<td>24</td>
</tr>
<tr>
<td>A.6.1. Bounds checking</td>
<td>24</td>
</tr>
<tr>
<td>A.6.2. Communication</td>
<td>24</td>
</tr>
<tr>
<td>A.6.3. Error conditions</td>
<td>24</td>
</tr>
<tr>
<td>A.6.4. Memory management</td>
<td>25</td>
</tr>
<tr>
<td>A.6.5. Optimisation</td>
<td>26</td>
</tr>
<tr>
<td>A.6.6. Package attribute: PRIVATE, PUBLIC, USE, ONLY</td>
<td>26</td>
</tr>
<tr>
<td>A.6.7. Parallelism using MPI</td>
<td>26</td>
</tr>
<tr>
<td>A.7. Features to be avoided</td>
<td>26</td>
</tr>
</tbody>
</table>

**Bibliography**

**Indexes**

- Namelist blocks .................................. 30
- Namelist parameters ................................ 31
1.1. Basics

The time evolution of any passive tracer $C$ follows the transport equation, which is similar to that of active tracer - temperature or salinity:

$$\frac{\partial C}{\partial t} = S(C) - \frac{1}{b_t} \left[ \frac{\partial e^{2u} e^{3u} u C}{\partial i} + \frac{\partial e^{1v} e^{3v} wv, C}{\partial i} \right] + \frac{1}{e_{3t}} \frac{\partial w C}{\partial k} + D^{hC} + D^{vC}$$

(1.1)

where expressions of $D^{hC}$ and $D^{vC}$ depend on the choice for the lateral and vertical subgrid scale parameterizations, see equations 5.10 and 5.11 in (Gurvan and Team, TBD)

$S(C)$, the first term on the right hand side of 1.1; is the SMS - Source Minus Sink - inherent to the tracer. In the case of biological tracer such as phytoplankton, $S(C)$ is the balance between phytoplankton growth and its decay through mortality and grazing. In the case of a tracer comprising carbon, $S(C)$ accounts for gas exchange, river discharge, flux to the sediments, gravitational sinking and other biological processes. In the case of a radioactive tracer, $S(C)$ is simply loss due to radioactive decay.

The second term (within brackets) represents the advection of the tracer in the three directions. It can be interpreted as the budget between the incoming and outgoing tracer fluxes in a volume $T$-cells $b_t = e_{1t} e_{2t} e_{3t}$

The third term represents the change due to lateral diffusion.

The fourth term is change due to vertical diffusion, parameterized as eddy diffusion to represent vertical turbulent fluxes:

$$D^{vC} = \frac{1}{e_{3t}} \frac{\partial}{\partial k} \left[ A^{vT} \frac{\partial C}{\partial k} \right]$$

(1.2)

where $A^{vT}$ is the vertical eddy diffusivity coefficient of active tracers
1.2. The NEMO-TOP interface

TOP is the NEMO hardwired interface toward biogeochemical models and provide the physical constraints/boundaries for oceanic tracers. It consists of a modular framework to handle multiple ocean tracers, including also a variety of built-in modules.

This component of the NEMO framework allows one to exploit available modules and further develop a range of applications, spanning from the implementation of a dye passive tracer to evaluate dispersion processes (by means of MY_TRC), track water masses age (AGE module), assess the ocean interior penetration of persistent chemical compounds (e.g., gases like CFC or even PCBs), up to the full set of equations involving marine biogeochemical cycles.

TOP interface has the following location in the code repository: `<repository>/src/TOP/` and the following modules are available:

- **TRP**: Interface to NEMO physical core for computing tracers transport
- **CFC**: Inert carbon tracers (CFC11, CFC12, SF6)
- **C14**: Radiocarbon passive tracer
- **AGE**: Water age tracking
- **MY_TRC**: Template for creation of new modules and external BGC models coupling
- **PISCES**: Built in BGC model. See (Aumont et al., 2015) for a thorough description.

1.3. The transport component : TRP

The passive tracer transport component shares the same advection/diffusion routines with the dynamics, with specific treatment of some features like the surface boundary conditions, or the positivity of passive tracers concentrations.

### 1.3.1. Advection

The advection schemes used for the passive tracers are the same than the ones for $T$ and $S$ and described in section 5.1 of (Gurvan and Team, TBD). The choice of an advection scheme can be selected independently and can differ from the ones used for active tracers. This choice is made in the nam_trc_adv namelist, by setting to true one and only one of the logicals ln_trcadv_xxx, the same way of what is done for dynamics. cen2, MUSCL2, and UBS are not positive schemes meaning that negative values can appear in an initially strictly positive tracer field which is advected, implying that false extrema are permitted. Their use is not recommended on passive tracers.

### 1.3.2. Lateral diffusion

The lateral diffusion schemes used for the passive tracers are the same than the ones for $T$ and $S$ and described in section 5.1 of (Gurvan and Team, TBD).
In NEMO v4.0, the passive tracer diffusion has necessarily the same form as the active tracer diffusion, meaning that the numerical scheme must be the same. However the passive tracer mixing coefficient can be chosen as a multiple of the active ones by changing the value of \( r_{n\_ldf\_multi} \) in namelist \( \text{namtrc\_ldf} \). The choice of numerical scheme is then set in the \&\text{namnamtra\_ldf} namelist for the dynamic described in section 5.2 of (Gurvan and Team, TBD).

### 1.3.3. Tracer damping

\[
\text{&namtrc\_dmp} \quad ! \text{passive tracer newtonian damping} \quad (\text{ln\_trcdmp}=\text{T})
\]

- \( nn\_zdmp\_tr = 1 \) \quad \text{vertical shape} = 0 \quad \text{damping throughout the water column} \quad \text{(z criteria)}
- \( = 1 \) no damping in the mixing layer \quad \text{(z criteria)}
- \( = 2 \) no damping in the mixed layer \quad \text{(rho criteria)}
- \( cn\_resto\_tr = \text{resto\_tr.nc} \quad ! \text{create a damping.coeff NetCDF file} \quad (=1) \quad \text{or not} \quad (=0) \)

The use of newtonian damping to climatological fields or observations is also coded, sharing the same routine dans active tracers. Boolean variables are defined in the namelist \text{top\_ref} to select the tracers on which restoring is applied. Options are defined through the \&\text{namnamtrc\_dmp} namelist variables. The restoring term is added when the namelist parameter \( \text{ln\_trcdmp} \) is set to true. The restoring coefficient is a three-dimensional array read in a file, which name is specified by the namelist variable \( cn\_resto\_tr \). This netcdf file can be generated using the DMP\_TOOLS tool.

### 1.3.4. Tracer positivity

\[
\text{&namtrc\_rad} \quad ! \text{treatment of negative concentrations}
\]

- \( \text{ln\_trcrad} = \text{.true.} \quad ! \text{artificially correct negative concentrations} \quad \text{(T)} \quad \text{or not} \quad \text{(F)} \)

Sometimes, numerical scheme can generates negative values of passive tracers concentration that must be positive. For example, isopycnal diffusion can created extrema. The trcrad routine artificially corrects negative concentrations with a very crude solution that either sets negative concentration to zero without adjusting the tracer budget, or by removing negative concentration and keeping mass conservation. The treatment of negative concentrations is an option and can be selected in the namelist \&\text{namnamtrc\_rad} by setting the parameter \( \text{ln\_trcrad} \) to true.

### 1.4. The SMS modules

#### 1.4.1. Ideal Age

\[
\text{&namage} \quad ! \text{AGE}
\]

- \( \text{rn\_age\_depth} = 10 \) \quad \text{depth over which age tracer reset to zero}
- \( \text{rn\_age\_kill\_rate} = -0.000138888 \) \quad \text{= -1/7200 recip of relaxation timescale} \quad \text{(s) for age tracer shallower than age\_depth}

An ‘ideal age’ tracer is integrated online in TOP when \( \text{ln\_age} = \text{.true.} \) in namelist \text{namtrc}. This tracer marks the length of time in units of years that fluid has spent in the interior of the ocean, insulated from exposure to the atmosphere. Thus, away from the surface for \( z < -H_{\text{Age}} \) where \( H_{\text{Age}} \) is specified by the \text{namage} namelist variable \( \text{rn\_age\_depth} \), whose default value is 10 m, there is a source \( \text{SMS}_{\text{Age}} \) of the age tracer \( A \):

\[
\text{SMS}_{\text{Age}} = 1\text{yr}^{-1} = 1/T_{\text{year}},
\]

where the length of the current year \( T_{\text{year}} = 86400 \times N_{\text{days in current year}} \) s, where \( N_{\text{days in current year}} \) may be 366 or 365 depending on whether the current year is a leap year or not. Near the surface, for \( z > -H_{\text{Age}} \), ideal age is relaxed back to zero:

\[
\text{SMS}_{\text{Age}} = -\lambda_{\text{Age}} A,
\]
where the relaxation rate $\lambda_{\text{Age}}$ (units s$^{-1}$) is specified by the namage namelist variable $rn_{\text{age\_kill\_rate}}$ and has a default value of 1/7200 s. Since this relaxation is applied explicitly, this relaxation rate in principle should not exceed $1/\Delta t$, where $\Delta t$ is the time step used to step forward passive tracers ($2 * \text{nn\_dttrc} * \text{rn\_rdt}$ when the default leapfrog time-stepping scheme is employed).

Currently the 1-dimensional reference depth of the grid boxes is used rather than the dynamically evolving depth to determine whether the age tracer is incremented or relaxed to zero. This means that the tracer only works correctly in $z$-coordinates. To ensure that the forcing is independent of the level thicknesses, where the tracer cell at level $k$ has its upper face $z = -\text{depw}(k)$ above the depth $-H_{\text{Age}}$, but its lower face $z = -\text{depw}(k + 1)$ below that depth, then the age source

$$\text{SMS}_{\text{Age}} = -f_{\text{kill}} \lambda_{\text{Age}} A + f_{\text{add}} / T_{\text{year}},$$

where

$$f_{\text{kill}} = 3 t_{k}^{-1} (H_{\text{Age}} - \text{depw}(k)),$$

$$f_{\text{add}} = 1 - f_{\text{kill}}.$$

This implementation was first used in the CORE-II intercomparison runs described e.g. in Danabasoglu et al. (2014).

1.4.2. Inert carbons tracer

Chlorofluorocarbons 11 and 12 (CFC-11 and CFC-12), and sulfur hexafluoride (SF6), are synthetic chemicals manufactured for industrial and domestic applications from the early 20th century onwards. CFC-11 (CCl$_3$F) is a volatile liquid at room temperature, and was widely used in refrigeration. CFC-12 (CCl$_2$F$_2$) is a gas at room temperature, and, like CFC-11, was widely used as a refrigerant, and additionally as an aerosol propellant. SF6 (SF$_6$) is also a gas at room temperature, with a range of applications based around its property as an excellent electrical insulator (often replacing more toxic alternatives). All three are relatively inert chemicals that are both non-toxic and non-flammable, and their wide use has led to their accumulation within the Earth’s atmosphere. Large-scale production of CFC-11 and CFC-12 began in the 1930s, while production of SF6 began in the 1950s, and their atmospheric concentration time-histories are shown in Figure 1.1. As can be seen in the figure, while the concentration of SF6 continues to rise to the present day, the concentrations of both CFC-11 and CFC-12 have levelled off and declined since around the 1990s. These declines have been driven by the Montreal Protocol (effective since August 1989), which has banned the production of CFC-11 and CFC-12 (as well as other CFCs) because of their role in the depletion of stratospheric ozone ($\text{O}_3$), critical in decreasing the flux of ultraviolet radiation to the Earth’s surface. Separate to this role in ozone-depletion, all three chemicals are significantly more potent greenhouse gases than CO$_2$ (especially SF6), although their relatively low atmospheric concentrations limit their role in climate change.

The ocean is a notable sink for all three gases, and their relatively recent occurrence in the atmosphere, coupled to the ease of making high precision measurements of their dissolved concentrations, has made them valuable in oceanography. Because they only enter the ocean via surface air-sea exchange, and are almost completely chemically and biologically inert, their distribution within the ocean interior reveals its ventilation via transport and mixing. Measuring the dissolved concentrations of the gases – as well as the mixing ratios between them – shows circulation pathways within the ocean as well as water mass ages (i.e. the time since last contact with the atmosphere). This feature of the gases has made them valuable across a wide range of oceanographic problems. One use lies in ocean modelling, where they can be used to evaluate the realism of the circulation and ventilation of models, key for understanding the behaviour of wider modelled marine biogeochemistry (e.g. (Dutay et al., 2002; Palmiéri et al., 2015)).

Modelling these gases (henceforth CFCs) in NEMO is done within the passive tracer transport module, TOP, using the conservation state equation 1.1.

Advection and diffusion of the CFCs in NEMO are calculated by the physical module, OPA, whereas sources and sinks are done by the CFC module within TOP. The only source for CFCs in the ocean is via air-sea gas exchange at its surface, and since CFCs are generally stable within the ocean, we assume that there are no sinks (i.e. no loss processes) within the ocean interior. Consequently, the sinks-minus-sources term for CFCs consists only of their air-sea fluxes, $F_{cfc}$, as described in the Ocean Model Inter-comparison Project (OMIP) protocol (Orr et al., 2017):
The standard outputs of the CFC module are seawater CFC concentrations (in mol m\(^{-3}\)) and the cumulative net air-sea flux (in mol m\(^{-2}\) d\(^{-1}\)) and the cumulative net air-sea flux (in mol m\(^{-2}\)). Using XIOS, it is possible to obtain outputs such as the vertical integral of CFC concentrations (in mol m\(^{-2}\); see Figure 1.3). This property, when divided by the surface CFC concentration, estimates the local penetration depth (in m) of the CFC.

**Notes**

In comparison to the OMIP protocol, the CFC module in NEMO has several differences:

- For instance, \(C_{sat}\) is calculated for a fixed surface pressure of 1 atm, what could be corrected in a further version of the module.

### 1.4.3. Radiocarbon

```fortran
! namcl4_fcg   ! files & dates
!
! namcl4_fcg   ! For Paleo-historical: specify tyrcl4_beg in yr BP
! namcl4_fcg   ! For Bomb: tyrcl4_beg=0
! namcl4_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
! namc14_fcg   !
!
Table 1.1: Coefficients for fit of the CFCs solubility (Eq. 1.12).

<table>
<thead>
<tr>
<th>Gas</th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
<th>b1</th>
<th>b2</th>
<th>b3</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFC-11</td>
<td>-218.0971</td>
<td>298.9702</td>
<td>113.8049</td>
<td>-1.39165</td>
<td>-0.143566</td>
<td>0.091015</td>
<td>-0.0153924</td>
</tr>
<tr>
<td>CFC-12</td>
<td>-229.9261</td>
<td>319.6552</td>
<td>119.4471</td>
<td>-1.39165</td>
<td>-0.142382</td>
<td>0.091459</td>
<td>-0.0157274</td>
</tr>
<tr>
<td>SF6</td>
<td>-80.0343</td>
<td>117.232</td>
<td>29.5817</td>
<td>0.0</td>
<td>0.0335183</td>
<td>-0.0373942</td>
<td>0.00774862</td>
</tr>
</tbody>
</table>

Table 1.2: Coefficients for fit of the CFCs Schmidt number (Eq. 1.11).

<table>
<thead>
<tr>
<th>Gas</th>
<th>a0</th>
<th>a1</th>
<th>a2</th>
<th>a3</th>
<th>a4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CFC-11</td>
<td>3579.2</td>
<td>-222.63</td>
<td>7.5749</td>
<td>-0.14595</td>
<td>0.0011874</td>
</tr>
<tr>
<td>CFC-12</td>
<td>3828.1</td>
<td>-249.86</td>
<td>8.7603</td>
<td>-0.1716</td>
<td>0.001408</td>
</tr>
<tr>
<td>SF6</td>
<td>3177.5</td>
<td>-200.57</td>
<td>6.8865</td>
<td>-0.13335</td>
<td>0.0010877</td>
</tr>
</tbody>
</table>

Figure 1.1: Atmospheric CFC11, CFC12 and SF6 partial pressure evolution in both hemispheres.

Figure 1.2: CFC11 solubility in mol m$^{-3}$ pptv$^{-1}$, calculated from the World Ocean Atlas 2013 temperature and salinity annual climatology.

```
cfilec14 = 'atmc14.dat' ! atmospheric c14 - Bomb
tyc14_beg = 0.00 ! starting year of experiment - Bomb
! cfileco2 = 'ByrdEdcCO2.txt' ! atmospheric co2 - Paleo
```
Chap. 1 Model Description

Figure 1.3: CFC11 vertical inventory in $\mu$mol m$^{-2}$, from one of the UK Earth System Model 1 model (UKESM1 - which uses NEMO as ocean component, with TOP for the passive tracers) historical run at year 2000.

```plaintext
! cfilec14 = 'intcal13.14c' ! atmospheric c14 - Paleo
! tyrc14_beg = 35000.00 ! starting year of experiment - Paleo (yr BP)
/

!-----------------------------------------------------------------------
&
! namc14_typ ! C14 - type of C14 tracer, default values of C14/C and pco2
!-----------------------------------------------------------------------
kc14typ = 0 ! Type of C14 tracer (0=equilibrium; 1=bomb transient; 2=past transient)
rc14at = 1.0 ! Default value for atmospheric C14/C (used for equil run)
pco2at = 280.0 ! Default value for atmospheric pCO2 [atm] (used for equil run)
rc14init = 0.85 ! Default value for initialization of ocean C14/C (when no restart)
/

!-----------------------------------------------------------------------
&
! namc14_sbc ! C14 - surface BC
!-----------------------------------------------------------------------
ln_chemh = .true. ! Chemical enhancement in piston vel.: yes/no
xdicsur = 2.0 ! Reference DIC surface concentration (mol/m3)
/

The C14 package implemented in NEMO by Anne Mouchet models ocean $\Delta^{14}$C. It offers several possibilities: $\Delta^{14}$C as a physical tracer of the ocean ventilation (natural $^{14}$C), assessment of bomb radiocarbon uptake, as well as transient studies of paleo-historical ocean radiocarbon distributions.

Method

Let $^{14}$R represent the ratio of $^{14}$C atoms to the total number of carbon atoms in the sample, i.e. $^{14}$C/$^{12}$C. Then, radiocarbon anomalies are reported as

$$\Delta^{14}C = \left( \frac{^{14}R}{^{14}R_{ref}} - 1 \right) \times 10^{3},$$

where $^{14}R_{ref}$ is a reference ratio. For the purpose of ocean ventilation studies $^{14}R_{ref}$ is set to one.

Here we adopt the approach of Fiadeiro (1982) and Toggweiler et al. (1989a,b) in which the ratio $^{14}$R is transported rather than the individual concentrations C and $^{14}$C. This approach calls for a strong assumption, i.e., that of a homogeneous and constant dissolved inorganic carbon (DIC) field (Toggweiler et al., 1989a; Mouchet, 2013). While in terms of oceanic $\Delta^{14}$C, it yields similar results to approaches involving carbonate chemistry, it underestimates the bomb radiocarbon inventory because it assumes a constant air-sea CO$_2$ disequilibrium (Mouchet, 2013). Yet, field reconstructions of the ocean bomb $^{14}$C inventory are also biased low (Naegler, 2009) since they assume that the anthropogenic perturbation
did not affect ocean DIC since the pre-bomb epoch. For these reasons, bomb $^{14}\text{C}$ inventories obtained with the present method are directly comparable to reconstructions based on field measurements.

This simplified approach also neglects the effects of fractionation (e.g., air-sea exchange) and of biological processes. Previous studies by Bacastow and Maier-Reimer (1990) and Joos et al. (1997) resulted in nearly identical $\Delta^{14}\text{C}$ distributions among experiments considering biology or not. Since observed $^{14}\text{R}$ ratios are corrected for the isotopic fractionation when converted to the standard $\Delta^{14}\text{C}$ notation (Stuiver and Polach, 1977) the model results are directly comparable to observations.

Therefore the simplified approach is justified for the purpose of assessing the circulation and ventilation of OGCMs. The equation governing the transport of $^{14}\text{R}$ in the ocean is

$$\frac{\partial}{\partial t}^{14}\text{R} = - \nabla \cdot (u^{14}\text{R} - K \cdot \nabla^{14}\text{R}) - \lambda^{14}\text{R},$$  

(1.14)

where $\lambda$ is the radiocarbon decay rate, $u$ the 3-D velocity field, and $K$ the diffusivity tensor.

At the air-sea interface a Robin boundary condition (Haine, 2006) is applied to (1.14), i.e., the flux through the interface is proportional to the difference in the ratios between the ocean and the atmosphere

$$F = \kappa_R (^{14}\text{R} - ^{14}\text{R}_a),$$  

(1.15)

where $F$ is the flux out of the ocean, and $^{14}\text{R}_a$ is the atmospheric $^{14}\text{C}/\text{C}$ ratio. The transfer velocity $\kappa_R$ for the radiocarbon ratio in (1.15) is computed as

$$\kappa_R = \frac{\kappa_{\text{CO}_2} K_0}{C_T} p^a_{\text{CO}_2},$$  

(1.16)

with $\kappa_{\text{CO}_2}$ the carbon dioxide transfer or piston velocity, $K_0$ the CO$_2$ solubility in seawater, $p^a_{\text{CO}_2}$ the atmospheric CO$_2$ pressure at sea level, and $C_T$ the average sea-surface dissolved inorganic carbon concentration.

The CO$_2$ transfer velocity is based on the empirical formulation of Wanninkhof (1992) with chemical enhancement (Wanninkhof and Knox, 1996; Wanninkhof, 2014). The original formulation is modified to account for the reduction of the air-sea exchange rate in the presence of sea ice. Hence

$$\kappa_{\text{CO}_2} = \left( K_W w^2 + b \right) \left( 1 - f_{\text{ice}} \right) \sqrt{660/S_c},$$  

(1.17)

with $w$ the wind magnitude, $f_{\text{ice}}$ the fractional ice cover, and $S_c$ the Schmidt number. $K_W$ in (1.17) is an empirical coefficient with dimension of an inverse velocity. The chemical enhancement term $b$ is represented as a function of temperature $T$ (Wanninkhof, 1992)

$$b = 2.5(0.5246 + 0.016256T + 0.00049946 \times T^2).$$  

(1.18)

Model setup

To activate the C14 package, set the parameter $\text{ln_c14} = .\text{true.}$ in namelist namtrc.

Parameters and formulations

The radiocarbon decay rate ($\kappa_{\text{LM14}}$; in trcnam_c14 module) is set to $\lambda = (1/8267) \text{ yr}^{-1}$ (Stuiver and Polach, 1977), which corresponds to a half-life of 5730 yr.

The Schmidt number $S_c$, Eq. (1.17), is calculated with the help of the formulation of Wanninkhof (2014). The CO$_2$ solubility $K_0$ in (1.16) is taken from Weiss (1974). $K_0$ and $S_c$ are computed with the OGCM temperature and salinity fields (trcsms_c14 module).

The following parameters intervening in the air-sea exchange rate are set in namelist_c14:

- The reference DIC concentration $C_T$ (xdicsur) intervening in (1.16) is classically set to 2 mol m$^{-3}$ (Toggweiler et al., 1989a; Orr et al., 2001; BUTZIN et al., 2005).
- The value of the empirical coefficient $K_W$ (xkwwind) in (1.17) depends on the wind field and on the model upper ocean mixing rate (Toggweiler et al., 1989a; Wanninkhof, 1992; Naegler, 2009; Wanninkhof, 2014). It should be adjusted so that the globally averaged CO$_2$ piston velocity is $\kappa_{\text{CO}_2} = 16.5 \pm 3.2 \text{ cm/h}$ (Naegler, 2009).
- Chemical enhancement (term $b$ in Eq. 1.18) may be set on/off by means of the logical variable $\text{ln_chemh}$.

Experiment type

The type of experiment is determined by the value given to $\kappa_{14\text{TYPO}}$ in namelist_c14. There are three possibilities:

1. natural $\Delta^{14}\text{C}$: $\kappa_{14\text{TYPO}}=0$
2. bomb $\Delta^{14}\text{C}$: $\kappa_{14\text{TYPO}}=1$
3. transient paleo-historical \( \Delta^{14}C \): \texttt{kc14typ}=2

**Natural or Equilibrium radiocarbon** \texttt{kc14typ}=0

Unless otherwise specified in \texttt{namelist_c14}, the atmospheric \( ^{14}R_{at} \) (\texttt{rc14at}) is set to one, the atmospheric \( CO_2 \) (\texttt{pc02at}) to 280 ppm, and the ocean \( ^{14}R \) is initialized with \texttt{rc14init}=0.85, i.e., \( \Delta^{14}C = -150\% \) (typical for deep-ocean, Fig 6 in Key et al., 2004).

Equilibrium experiment should last until 98\% of the ocean volume exhibit a drift of less than 0.001\%/year (Orr et al., 2000); this is usually achieved after few kyr (Fig. 1.4).

![Figure 1.4](image)

**Transient: Bomb** \texttt{kc14typ}=1

Performing this type of experiment requires that a pre-industrial equilibrium run be performed beforehand (\texttt{ln_rsttr} should be set to \texttt{.TRUE.}).

An exception to this rule is when wishing to perform a perturbation bomb experiment as was possible with the package \texttt{C14b}. It is still possible to easily set-up that type of transient experiment for which no previous run is needed. In addition...
to the instructions as given in this section it is however necessary to adapt the `atmc14.dat` file so that it does no longer contain any negative $\Delta^{14}C$ values (Suess effect in the pre-bomb period).

The model is integrated from a given initial date following the observed records provided from 1765 AD on (Fig. 1.5). The file `atmc14.dat` (G Enting et al., 1994, & I. Levin, personal comm.) provides atmospheric $\Delta^{14}C$ for three latitudinal bands: 90S-20S, 20S-20N & 20N-90N. Atmospheric CO$_2$ in the file `splco2.dat` is obtained from a spline fit through ice core data and direct atmospheric measurements (Orr et al., 2000, & J. Orr, personal comm.). Dates in these forcing files are expressed as yr AD.

To ensure that the atmospheric forcing is applied properly as well as that output files contain consistent dates and inventories the experiment should be set up carefully:

- Specify the starting date of the experiment: `nn_date0` in `namelist`. `nn_date0` is written as Year0101 where Year may take any positive value (AD).
- Then the parameters `nn_rstctl` in `namelist` (on-line) and `nn_rsttr` in `namelist_top` (off-line) must be set to 0 at the start of the experiment (force the date to `nn_date0` for the first experiment year).
- These two parameters (`nn_rstctl` and `nn_rsttr`) have then to be set to 2 for the following years (the date must be read in the restart file).

If the experiment date is outside the data time span then the first or last atmospheric concentrations are prescribed depending on whether the date is earlier or later. Note that `tyrc14_beg` (`namelist_c14`) is not used in this context.

**Transient: Past** $kC14$TYPE=2

![Figure 1.6: Atmospheric $\Delta^{14}C$ (solid) and CO$_2$ (dashed) forcing for the Paleo experiments. The CO$_2$ scale is given on the right axis.](image)

This experiment type does not need a previous equilibrium run. It should start 5–6 kyr earlier than the period to be analyzed. Atmospheric $^{14}R_a$ and CO$_2$ are prescribed from forcing files. The ocean $^{14}R$ is initialized with the value attributed to `rc14init` in `namelist_c14`.

The file `intcal13.14c` (Reimer et al., 2013) contains atmospheric $\Delta^{14}C$ from 0 to 50 kyr cal BP*. The CO$_2$ forcing is provided in file `ByrdEdcCO2.txt`. The content of this file is based on the high resolution record from EPICA Dome C (Monnin et al., 2004) for the Holocene and the Transition, and on Byrd Ice Core CO2 Data for 20–90 kyr BP (Ahn and Brook, 2008). These atmospheric values are reproduced in Fig. 1.6. Dates in these files are expressed as yr BP.

To ensure that the atmospheric forcing is applied properly as well as that output files contain consistent dates and inventories the experiment should be set up carefully. The true experiment starting date is given by `tyrc14_beg` (in yr BP) in `namelist_c14`. In consequence, `nn_date0` in `namelist` MUST be set to 00010101.

Then the parameters `nn_rstctl` in `namelist` (on-line) and `nn_rsttr` in `namelist_top` (off-line) must be set to 0 at the start of the experiment (force the date to `nn_date0` for the first experiment year). These two parameters have then to be set to 2 for the following years (read the date in the restart file).

If the experiment date is outside the data time span then the first or last atmospheric concentrations are prescribed depending on whether the date is earlier or later.

*cal BP: number of years before 1950 AD
Chap. 1 Model Description

Model output All output fields in Table 1.3 are routinely computed. It depends on the actual settings in iodef.xml whether they are stored or not.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Dim</th>
<th>Units</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC14</td>
<td>ptrc</td>
<td>3-D</td>
<td>-</td>
<td>Radiocarbon ratio</td>
</tr>
<tr>
<td>DeltaC14</td>
<td>diad</td>
<td>3-D</td>
<td>‰</td>
<td>$\Delta^{14}C$</td>
</tr>
<tr>
<td>C14Age</td>
<td>diad</td>
<td>3-D</td>
<td>yr</td>
<td>Radiocarbon age</td>
</tr>
<tr>
<td>RAge</td>
<td>diad</td>
<td>2-D</td>
<td>yr</td>
<td>Reservoir age</td>
</tr>
<tr>
<td>qtr_c14</td>
<td>diad</td>
<td>2-D</td>
<td>m$^{-2}$yr$^{-1}$</td>
<td>Air-to-sea net $^{14}$R flux</td>
</tr>
<tr>
<td>qint_c14</td>
<td>diad</td>
<td>2-D</td>
<td>m$^{-2}$</td>
<td>Cumulative air-to-sea $^{14}$R flux</td>
</tr>
<tr>
<td>AtmCO2</td>
<td>scalar</td>
<td>0-D</td>
<td>ppm</td>
<td>Global atmospheric $CO_2$</td>
</tr>
<tr>
<td>AtmC14</td>
<td>scalar</td>
<td>0-D</td>
<td>‰</td>
<td>Global atmospheric $\Delta^{14}C$</td>
</tr>
<tr>
<td>K_CO2</td>
<td>scalar</td>
<td>0-D</td>
<td>cm h$^{-1}$</td>
<td>Global CO$<em>2$ piston velocity ($\overline{\kappa}</em>{CO_2}$)</td>
</tr>
<tr>
<td>K_C14</td>
<td>scalar</td>
<td>0-D</td>
<td>m yr$^{-1}$</td>
<td>Global $^{14}$R transfer velocity ($\overline{\kappa}_{R}$)</td>
</tr>
<tr>
<td>C14Inv</td>
<td>scalar</td>
<td>0-D</td>
<td>$10^{26}$ atoms</td>
<td>Ocean radiocarbon inventory</td>
</tr>
</tbody>
</table>

The radiocarbon age is computed as $(-1/\lambda) \ln \left( ^{14}R \right)$, with zero age corresponding to $^{14}R = 1$.

The reservoir age is the age difference between the ocean uppermost layer and the atmosphere. It is usually reported as conventional radiocarbon age; i.e., computed by means of the Libby radiocarbon mean life (8033 yr; Stuiver and Polach, 1977)

$$^{14}\tau_c = -8033 \ln \left( 1 + \frac{\Delta^{14}C}{10^\lambda} \right),$$

where $^{14}\tau_c$, is expressed in years B.P. Here we do not use that convention and compute reservoir ages with the mean lifetime $1/\lambda$. Conversion from one scale to the other is readily performed. The conventional radiocarbon age is lower than the radiocarbon age by $\simeq 3\%$.

The ocean radiocarbon inventory is computed as

$$N_A^{^{14}}R_{oxa} \overline{C_T} \left( \int \Omega ^{^{14}}R d\Omega \right) /10^{26},$$

where $N_A$ is the Avogadro’s number ($N_A = 6.022 \times 10^{23}$ at/mol), $^{14}R_{oxa}$ is the oxalic acid radiocarbon standard ($^{14}R_{oxa} = 1.176 \times 10^{-12}$; Stuiver and Polach, 1977), and $\Omega$ is the ocean volume. Bomb $^{14}$C inventories are traditionally reported in units of $10^{26}$ atoms, hence the denominator in (1.20).

All transformations from second to year, and inversely, are performed with the help of the physical constant $rs_{S义EA}$ the sideral year length expressed in seconds$^1$.

The global transfer velocities represent time-averaged$^2$ global integrals of the transfer rates:

$$\overline{\kappa}_{CO_2} = \int_S r_{CO_2} dS$$

and

$$\overline{\kappa}_{R} = \int_S \kappa_R dS$$

1.4.4. PISCES biogeochemical model

PISCES is a biogeochemical model which simulates the lower trophic levels of marine ecosystem (phytoplankton, microzooplankton and mesozooplankton) and the biogeochemical cycles of carbon and of the main nutrients (P, N, Fe, and Si). The model is intended to be used for both regional and global configurations at high or low spatial resolutions as well as for short-term (seasonal, interannual) and long-term (climate change, paleoceanography) analyses. Two versions of PISCES are available in NEMO v4.0:

PISCES-v2, by setting in namelist_pisces_ref ln_p4z o true, can be seen as one of the many Monod models (Monod, 1958). It assumes a constant Redfield ratio and phytoplankton growth depends on the external concentration in nutrients. There are twenty-four prognostic variables (tracers) including two phytoplankton compartments (diatoms and nanophytoplankton), two zooplankton size-classes (microzooplankton and mesozooplankton) and a description of the carbonate chemistry. Formulations in PISCES-v2 are based on a mixed Monod/Quota formalism: On one hand,

$^1$The variable (nyear_len) which reports the length in days of the previous/current/future year (see oce_trc.F90) is not a constant.

$^2$the actual duration is set in iodef.xml
stoichiometry of C/N/P is fixed and growth rate of phytoplankton is limited by the external availability in N, P and Si. On the other hand, the iron and silicium quotas are variable and growth rate of phytoplankton is limited by the internal availability in Fe. Various parameterizations can be activated in PISCES-v2, setting for instance the complexity of iron chemistry or the description of particulate organic materials.

PISCES-QUOTA has been built on the PISCES-v2 model described in Aumont et al. (2015). PISCES-QUOTA has thirty-nine prognostic compartments. Phytoplankton growth can be controlled by five modeled limiting nutrients: Nitrate and Ammonium, Phosphate, Silicate and Iron. Five living compartments are represented: Three phytoplankton size classes/groups corresponding to picophytoplankton, nanophytoplankton and diatoms, and two zooplankton size classes which are microzooplankton and mesozooplankton. For phytoplankton, the prognostic variables are the carbon, nitrogen, phosphorus, iron, chlorophyll and silicon biomasses (the latter only for diatoms). This means that the N/C, P/C, Fe/C and Chl/C ratios of both phytoplankton groups as well as the Si/C ratio of diatoms are prognostically predicted by the model. Zooplankton are assumed to be strictly homeostatic (e.g., Sterner and Elser, 2003; Woods and Wilson, 2013; Meunier et al., 2014). As a consequence, the C/N/P/Fe ratios of these groups are maintained constant and are not allowed to vary. In PISCES, the Redfield ratios C/N/P are set to 122/16/1 (Takahashi et al., 1985) and the -O/C ratio is set to 1.34 (Körtzinger et al., 2001). No silicified zooplankton is assumed. The bacterial pool is not yet explicitly modeled.

There are three non-living compartments: Semi-labile dissolved organic matter, small sinking particles, and large sinking particles. As a consequence of the variable stoichiometric ratios of phytoplankton and of the stoichiometric regulation of zooplankton, elemental ratios in organic matter cannot be supposed constant anymore as that was the case in PISCES-v2. Indeed, the nitrogen, phosphorus, iron, silicon and calcite pools of the particles are now all explicitly modeled. The sinking speed of the particles is not altered by their content in calcite and biogenic silicate ("The ballast effect", (Honjo, 1996; Armstrong et al., 2001)). The latter particles are assumed to sink at the same speed as the large organic matter particles. All the non-living compartments experience aggregation due to turbulence and differential settling as well as Brownian coagulation for DOM.

1.4.5. MY_TRC interface for coupling external BGC models

The NEMO-TOP has only one built-in biogeochemical model - PISCES - but there are several BGC models - MEDUSA, ERSEM, BFM or ECO3M - which are meant to be coupled with the NEMO dynamics. Therefore it was necessary to provide to the users a framework for easily add their own BGCM model, that can be a single passive tracer. The generalized interface is pivoted on MY_TRC module that contains template files to build the coupling between NEMO and any external BGC model. The call to MY_TRC is activated by setting ln_my_trc = .true. in namelist namtrc

The following 6 fortran files are available in MY_TRC with the specific purposes here described.

• par_my_trc.F90 : This module allows to define additional arrays and public variables to be used within the MY_TRC interface

• trcini_my_trc.F90 : Here are initialized user defined namelists and the call to the external BGC model initialization procedures to populate general tracer array (trn and trb). Here are also likely to be defined suport arrays related to system metrics that could be needed by the BGC model.

• trcnam_my_trc.F90 : This routine is called at the beginning of trcini_my_trc and should contain the initialization of additional namelists for the BGC model or user-defined code.

• trcsms_my_trc.F90 : The routine performs the call to Boundary Conditions and its main purpose is to contain the Source-Minus-Sinks terms due to the biogeochemical processes of the external model. Be aware that lateral boundary conditions are applied in trcnxt routine. IMPORTANT: the routines to compute the light penetration along the water column and the tracer vertical sinking should be defined/called in here, as generalized modules are still missing in the code.

• trcice_my_trc.F90 : Here it is possible to prescribe the tracers concentrations in the seaice that will be used as boundary conditions when ice melting occurs (nn_ice_tr =1 in namtrc_ice). See e.g. the correspondent PISCES subroutine.

• trcwri_my_trc.F90 : This routine performs the output of the model tracers using IOM module (see Manual Chapter Output and Diagnostics). It is possible to place here the output of additional variables produced by the model, if not done elsewhere in the code, using the call to iom_put.

1.5. The Offline Option
Chap. 1 Model Description

Coupling passive tracers offline with NEMO requires precomputed physical fields from OGCM. Those fields are read from files and interpolated on-the-fly at each model time step. At least the following dynamical parameters should be absolutely passed to the transport: ocean velocities, temperature, salinity, mixed layer depth and for ecosystem models like PISCES, sea ice concentration, short wave radiation at the ocean surface, wind speed (or at least, wind stress). The so-called offline mode is useful since it has lower computational costs for example to perform very longer simulations - about 3000 years - to reach equilibrium of CO2 sinks for climate-carbon studies.

The offline interface is located in the code repository: `<repository>/src/OFF/`. It is activated by adding the CPP key `key_offline` to the CPP keys list. There are two specifics routines for the Offline code:

- `dtadyn.F90`: this module allows to read and compute the dynamical fields at each model time-step
- `nemogcm.F90`: a degraded version of the main nemogcm.F90 code of NEMO to manage the time-stepping
2.1. Setting up a passive tracer configuration

The usage of TOP is activated

- by including in the configuration definition the component TOP_SRC
- by adding the macro key_top in the configuration cpp file

As an example, the user can refer to already available configurations in the code, GYRE_PISCES being the NEMO biogeochemical demonstrator and GYRE_BFM to see the required configuration elements to couple with an external biogeochemical model (see also section §1.4).

Note that, since version 4.0, TOP interface core functionalities are activated by means of logical keys and all submodules preprocessing macros from previous versions were removed.

There are only three specific keys remaining in TOP

- key_top : to enables passive tracer module
- key_trdtrc and key_trdmaxl_trc : trend computation for tracers

For a remind, the revisited structure of TOP interface now counts for five different modules handled in namelist_top :

- PISCES, default BGC model
- MY_TRC, template for creation of new modules couplings (maybe run a single passive tracer)
- CFC, inert carbon tracers dynamics (CFC11,CFC12,SF6) Updated with OMIP-BGC guidelines (Orr et al, 2016)
- C14, radiocarbon passive tracer
- AGE, water age tracking revised implementation

The modular approach was implemented also in the definition of the total number of passive tracers (jptra). This results from to user setting from the namelist namtrc
2.2. TOP Tracer Initialisation

2.3. TOP Boundaries Conditions
3.1. TOP synthetic Workflow

3.1.1. Model initialization

3.1.2. Time marching procedure

3.2. Coupling an external BGC model using NEMO framework

The coupling with an external BGC model through the NEMO compilation framework can be achieved in different ways according to the degree of coding complexity of the Biogeochemical model, like e.g., the whole code is made only by one file or it has multiple modules and interfaces spread across several subfolders.

Beside the 6 core files of MY_TRC module, see, let’s assume an external BGC model named “MYBGC” and constituted by a rather essential coding structure, likely few Fortran files. The new coupled configuration name is NEMO_MYBGC.

The best solution is to have all files (the modified MY_TRC routines and the BGC model ones) placed in a unique folder with root <MYBGCPATH> and to use the `makenemo` external readdressing of MY_SRC folder. The coupled configuration listed in `cfg.txt` will look like

```
NEMO_MYBGC OPA_SRC TOP_SRC
```

and the related `cpp_MYBGC.fcm` content will be

```
bld::tool::fppkeys  key_iomp5  key_mpp_mpi  key_top
```

the compilation with `makenemo` will be executed through the following syntax

```
makenemo -n NEMO_MYBGC -m <arch_my_machine> -j 8 -e <MYBGCPATH>
```

```
bld::tool::fppkeys  key_iomp5  key_mpp_mpi  key_top
src::MYBGC::initialization  <MYBGCPATH>/initialization
src::MYBGC::pelagic  <MYBGCPATH>/pelagic
src::MYBGC::benthic  <MYBGCPATH>/benthic
bld::pp::MYBGC  1
bld::tool::fppflags::MYBGC  %FPPFLAGS
bld::tool::fppkeys  %bld::tool::fppkeys MYBGC_MACROS
```
Table of contents

A.1. Introduction ................................................. 18
A.2. Overview and general conventions ............................. 18
A.3. Architecture .................................................. 19
A.4. Style rules .................................................... 19
  A.4.1. Argument list format ................................... 19
  A.4.2. Array syntax ............................................. 19
  A.4.3. Case ....................................................... 19
  A.4.4. Comments ................................................ 20
  A.4.5. Continuation lines ...................................... 20
  A.4.6. Declaration of arguments and local variables .......... 20
  A.4.7. F90 Standard ............................................ 20
  A.4.8. Free-Form Source ....................................... 20
  A.4.9. Indentation .............................................. 21
  A.4.10. Loops .................................................... 21
  A.4.11. Naming Conventions: files ............................ 21
  A.4.13. Naming Conventions: variables ......................... 21
  A.4.15. Pre processor .......................................... 21
A.5. Content rules ................................................ 22
  A.5.1. Configurations .......................................... 22
  A.5.2. Constants .............................................. 22
  A.5.3. Declaration for variables and constants ............... 22
  A.5.4. Headers ................................................ 23
  A.5.5. Interface blocks ........................................ 23
  A.5.6. I/O Error Conditions ................................... 23
  A.5.7. PRINT - ASCII output files ............................ 23
  A.5.8. Precision ............................................... 24
  A.5.9. Structures .............................................. 24
A.6. Packages coding rules ........................................ 24
  A.6.1. Bounds checking ........................................ 24
  A.6.2. Communication ......................................... 24
  A.6.3. Error conditions ........................................ 24
  A.6.4. Memory management ..................................... 25
  A.6.5. Optimisation ............................................ 26
  A.6.6. Package attribute: PRIVATE, PUBLIC, USE, ONLY .... 26
  A.6.7. Parallelism using MPI .................................. 26
A.7. Features to be avoided ....................................... 26
Sect. A.3 Introduction

A "model life" is more than ten years. Its software, composed of a few hundred modules, is used by many people who are scientists or students and do not necessarily know every aspect of computing very well. Moreover, a well thought-out program is easier to read and understand, less difficult to modify, produces fewer bugs and is easier to maintain. Therefore, it is essential that the model development follows some rules:

- well planned and designed
- well written
- well documented (both on- and off-line)
- maintainable
- easily portable
- flexible.

To satisfy part of these aims, NEMO is written with a coding standard which is close to the ECMWF rules, named DOCTOR (?). These rules present some advantages like:

- to provide a well presented program
- to use rules for variable names which allow recognition of their type (integer, real, parameter, local or shared variables, etc.).

This facilitates both the understanding and the debugging of an algorithm.

A.1. Introduction

This document describes conventions used in NEMO coding and suggested for its development. The objectives are to offer a guide to all readers of the NEMO code, and to facilitate the work of all the developers, including the validation of their developments, and eventually the implementation of these developments within the NEMO platform.

A first approach of these rules can be found in the code in ./src/OCE/module_example where all the basics coding conventions are illustrated. More details can be found below.

This work is based on the coding conventions in use for the Community Climate System Model *, the previous version of this document ("FORTRAN coding standard in the OPA System") and the expertise of the NEMO System Team. After a general overview below, this document will describe:

- The style rules, i.e. the syntax, appearance and naming conventions chosen to improve readability of the code;
- The content rules, i.e. the conventions to improve the reliability of the different parts of the code;
- The package rules to go a step further by improving the reliability of the whole and interfaces between routines and modules.

A.2. Overview and general conventions

NEMO has 3 major components: ocean dynamics (.src/OCE), sea-ice (.src/ICE) and marine biogeochemistry (.src/MBG). In each directory, one will find some FORTRAN files and/or subdirectories, one per functionality of the code: ./src/OCE/BDY (boundaries), ./src/OCE/DIA (diagnostics), ./src/OCE/DOM (domain), ./src/OCE/DYN (dynamics), ./src/OCE/LDF (lateral diffusion), etc...

All name are chosen to be as self-explanatory as possible, in English, all prefixes are 3 digits.

English is used for all variables names, comments, and documentation.

Physical units are MKS. The only exception to this is the temperature, which is expressed in degrees Celsius, except in bulk formulae and part of SI³ sea-ice model where it is in Kelvin. See .src/OCE/DOM/phycst.F90 files for conversions.

*UCAR conventions
A.3. Architecture

Within each directory, organisation of files is driven by orthogonality, \textit{i.e.} one functionality of the code is intended to be in one and only one directory, and one module and all its related routines are in one file. The functional modules are:

- SBC surface module
- IOM management of the I/O
- NST interface to AGRIF (nesting model) for dynamics and biogeochemistry
- OBC, BDY management of structured and unstructured open boundaries
- C1D 1D (vertical) configuration for dynamics, sea-ice and biogeochemistry
- OFF off-line module: passive tracer or biogeochemistry alone
- ...

For example, the file \textit{domain.F90} contains the module \texttt{domain} and all the subroutines related to this module (\texttt{dom_init, dom_nam, dom_ctl}).

A.4. Style rules

A.4.1. Argument list format

Routine argument lists will contain a maximum 5 variables per line, whilst continuation lines can be used. This applies both to the calling routine and the dummy argument list in the routine being called. The purpose is to simplify matching up the arguments between caller and callee.

```fortran
SUBROUTINE tra_adv_eiv( kt, pun, pvn, pwn )
   CALL tra_adv_eiv( kt, zun, zvn, zwn )
END SUBROUTINE
```

A.4.2. Array syntax

Except for long loops (see below), array notation should be used if possible. To improve readability the array shape must be shown in brackets, \textit{e.g.}:

```fortran
onedarraya(:) = onedarrayb(:) + onedarrayc(:)
twodarray(:,1) = scalar * anothertwodarray(:,1)
```

When accessing sections of arrays, for example in finite difference equations, do so by using the triplet notation on the full array, \textit{e.g.}:

```fortran
twodarray(:,2:len2) = scalar * ( twodarray2(:,1:len2-1) - twodarray2(:,2:len2) )
```

For long, complicated loops, explicitly indexed loops should be preferred. In general when using this syntax, the order of the loops indices should reflect the following scheme (for best usage of data locality):

```fortran
DO  jk = 1, jpk
   DO  jj = 1, jpj
      DO  ji = 1, jpi
         threedarray(ji,jj,jk) = ...
      END DO
   END DO
END DO
```

A.4.3. Case

All FORTRAN keywords are in capital: \texttt{DIMENSION, WRITE, DO, END DO, NAMELIST, ...} All other parts of the \texttt{NEMO} code will be written in lower case.
A.4.4. Comments

Comments in the code are useful when reading the code and changing or developing it.
The full documentation and detailed explanations are to be added in the reference manual (TeX files, aside from the code itself).
In the code, the comments should explain variable content and describe each computational step.
Comments in the header start with “!!”. For more details on the content of the headers, see Content rules/Headers in this document.
Comments in the code start with “!”.
All comments are indented (3, 6, or 9 blank spaces).
Short comments may be included on the same line as executable code, and an additional line can be used with proper alignment. For example:

```
zx = zx + zy  ! Describe what is going on and if it is
! too long use another ! for proper
! alignment with automatic indentation
```

More in-depth comments should be written in the form:

```
! Check of some namelist values
```

or

```
!
! !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
! ! Bottom boundary condition on the
! !<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
!```

Key features of this style are

1. it starts with a "!" in the column required for proper indentation,
2. the text is offset above and below by a blank line or a content line built for underlying.

A.4.5. Continuation lines

Continuation lines can be used with precise alignment for readability. For example:

```
avmu(ji,jj,jk) = avmu(ji,jj,jk) + ( un(ji,jj,jk-1) - un(ji,jj,jk) ) / 6
  + ( ub(ji,jj,jk-1) - ub(ji,jj,jk) ) / 6
  / { fse3uw_n(ji,jj,jk) }
  & fse3uw_b(ji,jj,jk) )
```

Code lines, which are continuation lines of assignment statements, must begin to the right of the column of the assignment operator. Due to the possibility of automatic indentation in some editor (emacs for example), use a “&” as first character of the continuing lines to maintain the alignment.

A.4.6. Declaration of arguments and local variables

In a routine, input arguments and local variables are declared 1 per line, with a comment field on the same line as the declaration. Multiple comment lines describing a single variable are acceptable if needed. For example:

```
INTEGER :: kstp  ! ocean time-step index
```

A.4.7. F90 Standard

*NEMO* software adheres to the *Fortran 90* language standard and does not rely on any specific language or vendor extensions.

A.4.8. Free-Form Source

Free-form source will be used. The F90/95 standard allows lines of up to 132 characters, but a self-imposed limit of 80 should enhance readability, or print source files with two columns per page. Multi-line comments that extend to column 100 are unacceptable.
A.4.9. Indentation

Code as well as comment lines within loops, if-blocks, continuation lines, **MODULE** or **SUBROUTINE** statements will be indented 3 characters for readability (except for **CONTAINS** that remains at first column).

```
MODULE mod1
  REAL(wp) xx
CONTAINS
  SUBROUTINE sub76(pz, px, py, px, pa, &
  pb, pc, pd, pe)
    <instruction>
  END SUBROUTINE sub76
END MODULE mod1
```

A.4.10. Loops

Loops, if explicit, should be structured with the do-end do construct as opposed to numbered loops. Nevertheless non-numeric labels can be used for a big iterative loop of a recursive algorithm. In the case of a long loop, a self-descriptive label can be used (i.e. not just a number).

A.4.11. Naming Conventions: files

A file containing a module will have the same name as the module it contains (because dependency rules used by "make" programs are based on file names). ³

A.4.12. Naming Conventions: modules

Use a meaningful English name and the “3 letters” naming convention: first 3 letters for the code section, and last 3 to describe the module. For example, zdftke, where “zdf” stands for vertical diffusion, and “tke” for turbulent kinetic energy. Note that by implication multiple modules are not allowed in a single file. The use of common blocks is deprecated in Fortran 90 and their use in NEMO is strongly discouraged. Modules are a better way to declare static data. Among the advantages of modules is the ability to freely mix data of various types, and to limit access to contained variables through the use of the **ONLY** and **PRIVATE** attributes.

A.4.13. Naming Conventions: variables

All variable should be named as explicitly as possible in English. The naming convention concerns prefix letters of these name, in order to identify the variable type and status.

Never use a Fortran keyword as a routine or variable name.

The table below lists the starting letter(s) to be used for variable naming, depending on their type and status:


Use of the operators `<`, `>`, `<=`, `>=`, `==`, `/=` is strongly recommended instead of their deprecated counterparts (`.lt.`, `.gt.`, `.le.`, `.ge.`, `.eq.`, `.ne.`). The motivation is readability. In general use the notation: `< Blank > < Operator > < Blank >`

A.4.15. Pre processor

Where the use of a language pre-processor is required, it will be the C pre-processor (cpp).

The cpp key is the main feature used, allowing to ignore some useless parts of the code at compilation step.

The advantage is to reduce the memory use; the drawback is that compilation of this part of the code isn’t checked.

The cpp key feature should only be used for a few limited options, if it reduces the memory usage. In all cases, a logical variable and a FORTRAN IF should be preferred. When using a cpp key **key_optionname**, a corresponding logical variable **lk_optionname** should be declared to allow FORTRAN IF tests in the code and a FORTRAN module with the same name (i.e. **optionname.F90**) should be defined. This module is the only place where a "#if defined" command appears, selecting either the whole FORTRAN code or a dummy module. For example, the TKE vertical physics, the module name is **zdftke.F90**, the CPP key is **key_zdftke** and the associated logical is **lk_zdftke**.

The following syntax:

³For example, if routine A "USE"s module B, then "make" must be told of the dependency relation which requires B to be compiled before A. If one can assume that module B resides in file B.o, building a tool to generate this dependency rule (e.g. A.o: B.o) is quite simple. Put another way, it is difficult (to say nothing of CPU-intensive) to search an entire source tree to find the file in which module B resides for each routine or module which "USE"s B.
### Sect. A.5 Content rules

<table>
<thead>
<tr>
<th>Type / Status</th>
<th>integer</th>
<th>real</th>
<th>logical</th>
<th>character</th>
<th>double precision</th>
<th>complex</th>
</tr>
</thead>
<tbody>
<tr>
<td>public or module variable</td>
<td>m n</td>
<td>a b e f g h o</td>
<td>l but not</td>
<td>c but not</td>
<td>d but not</td>
<td>y but not</td>
</tr>
<tr>
<td>dummy argument</td>
<td>k</td>
<td>but not</td>
<td>p</td>
<td>ld</td>
<td>ed</td>
<td>dd</td>
</tr>
<tr>
<td>local variable</td>
<td>i</td>
<td>z</td>
<td>ll</td>
<td>cl</td>
<td>cd</td>
<td>yl</td>
</tr>
<tr>
<td>loop control</td>
<td>j</td>
<td>but not</td>
<td>pp</td>
<td>cp</td>
<td>dp</td>
<td>yp</td>
</tr>
<tr>
<td>parameter</td>
<td>jp</td>
<td>pp</td>
<td>lp</td>
<td>cp</td>
<td>dp</td>
<td>yp</td>
</tr>
<tr>
<td>namelist</td>
<td>nn_</td>
<td>rn_</td>
<td>ln_</td>
<td>cn_</td>
<td>dn_</td>
<td></td>
</tr>
<tr>
<td>CPP macro</td>
<td>kf</td>
<td>sf</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Is to be used rather than the #ifdef abbreviate form since it may have conflicts with some Unix scripts. Tests on cpp keys included in NEMO at compilation step:

- The CPP keys used are compared to the previous list of cpp keys (the compilation will stop if trying to specify a non-existing key)
- If a change occurs in the CPP keys used for a given experiment, the whole compilation phase is done again.

## A.5. Content rules

### A.5.1. Configurations

The configuration defines the domain and the grid on which NEMO is running. It may be useful to associate a CPP key and some variables to a given configuration, although the part of the code changed under each of those keys should be minimized. As an example, the "ORCA2" configuration (global ocean, 2 degrees grid size) is associated with the cpp key `key_orca2` for which

```c
cp_cfg = "orca"
jp_cfg = 2
```

### A.5.2. Constants

Physical constants (e.g., π, gas constants) must never be hard-wired into the executable portion of a code. Instead, a mnemonically named variable or parameter should be set to the appropriate value, in the setup routine for the package. We realize that many parameterizations rely on empirically derived constants or fudge factors, which are not easy to name. In these cases it is not forbidden to leave such factors coded as "magic numbers" buried in executable code, but comments should be included referring to the source of the empirical formula. Hard-coded numbers should never be passed through argument lists.

### A.5.3. Declaration for variables and constants

**Rules**

Variables used as constants should be declared with attribute `PARAMETER` and used always without copying to local variables, in order to prevent from using different values for the same constant or changing it accidentally.

- Usage of the `DIMENSION` statement or attribute is required in declaration statements.
• The “::” notation is quite useful to show that this program unit declaration part is written in standard FORTRAN syntax, even if there are no attributes to clarify the declaration section. Always use the notation `<blank> ::= <three blanks>` to improve readability.

• Declare the length of a character variable using the `CHARACTER (len=xxx)` syntax.

• For all global data (in contrast to module data, that is all data that can be accessed by other module) must be accompanied with a comment field on the same line. For example:

```fortran
REAL(wp), DIMENSION(jpi, jpj, jpk) :: ua  ! i-horizontal velocity (m/s)
```

**Implicit None**

All subroutines and functions will include an `IMPLICIT NONE` statement. Thus all variables must be explicitly typed. It also allows the compiler to detect typographical errors in variable names. For modules, one `IMPLICIT NONE` statement in the modules definition section is needed. This also removes the need to have `IMPLICIT NONE` statements in any routines that are `CONTAINS`ed in the module. Improper data initialisation is another common source of errors. To avoid problems, initialise variables as close as possible to where they are first used.

**Attributes**

`PRIVATE / PUBLIC`: All resources of a module are `PUBLIC` by default. A reason to store multiple routines and their data in a single module is that the scope of the data defined in the module can be limited to the routines which are in the same module. This is accomplished with the `PRIVATE` attribute.

`INTENT`: All dummy arguments of a routine must include the `INTENT` clause in their declaration in order to improve control of variables in routine calls.

**A.5.4. Headers**

Prologues are not used in NEMO for now, although it may become an interesting tool in combination with ProTeX auto documentation script in the future. Rules to code the headers and layout of a module or a routine are illustrated in the example module available with the code: `./src/OCE/module_example`

**A.5.5. Interface blocks**

Explicit interface blocks are required between routines if optional or keyword arguments are to be used. They also allow the compiler to check that the type, shape and number of arguments specified in the `CALL` are the same as those specified in the subprogram itself. FORTRAN 95 compilers can automatically provide explicit interface blocks for routines contained in a module.

**A.5.6. I/O Error Conditions**

I/O statements which need to check an error condition will use the `iostat=<integer variable>` construct instead of the outmoded `end=` and `err=`. Note that a 0 value means success, a positive value means an error has occurred, and a negative value means the end of record or end of file was encountered.

**A.5.7. PRINT - ASCII output files**

Output listing and errors are directed to `numout` logical unit =6 and produces a file called `ocean.output` (use `ln_prt` to have one output per process in MPP). Logical `lwp` variable allows for less verbose outputs. To output an error from a routine, one can use the following template:

```fortran
IF ( nstop /= 0 .AND. lwp ) THEN  ! error print
  WRITE(numout,cform_err)  
  WRITE(numout,*) nstop, ' error have been found'
ENDIF
```

---

‡The `len` specifier is important because it is possible to have several kinds for characters (e.g. Unicode using two bytes per character, or there might be a different kind for Japanese e.g. NEC).

§This allows an easy research of where and how a variable is declared using the unix command: “grep var *90 | grep !:.”

¶A variable could contain an initial value you did not expect. This can happen for several reasons, e.g. the variable has never been assigned a value, its value is outdated, memory has been allocated for a pointer but you have forgotten to initialise the variable pointed to.
### A.5.8. Precision

Parameterizations should not rely on vendor-supplied flags to supply a default floating point precision or integer size. The F95 `KIND` feature should be used instead. In order to improve portability between 32 and 64 bit platforms, it is necessary to make use of kinds by using a specific module `/src/OCE/par_kind.F90` declaring the "kind definitions" to obtain the required numerical precision and range as well as the size of `INTEGER`. It should be noted that numerical constants need to have a suffix of `_kindvalue` to have the according size.

Thus `wp` being the "working precision" as declared in `/src/OCE/par_kind.F90`, declaring real array `zpc` will take the form:

```fortran
REAL(wp), DIMENSION(jpi,jp,jpk) :: zpc ! power consumption
```

### A.5.9. Structures

The `TYPE` structure allowing to declare some variables is more often used in `NEMO`, especially in the modules dealing with reading fields, or interfaces. For example:

```fortran
! Definition of a tracer as a structure
TYPE PTRACER
  CHARACTER(len = 20) :: sname ! short name
  CHARACTER(len = 80) :: lname ! long name
  CHARACTER(len = 20) :: unit ! unit
  LOGICAL :: lini ! read in a file or not
  LOGICAL :: lsav ! output the tracer or not
END TYPE PTRACER

TYPE(PTRACER) , DIMENSION(jptr) :: tracer
```

Missing rule on structure name??

### A.6. Packages coding rules

#### A.6.1. Bounds checking

`NEMO` is able to run when an array bounds checking option is enabled (provided the cpp key `key_vectopt_loop` is not defined).

Thus, constructs of the following form are disallowed:

```fortran
REAL(wp) :: arr(1)
```

where "arr" is an input argument into which the user wishes to index beyond 1. Use of the (*) construct in array dimensioning is forbidden also because it effectively disables array bounds checking.

#### A.6.2. Communication

A package should refer only to its own modules and subprograms and to those intrinsic functions included in the Fortran standard.

All communication with the package will be through the argument list or namelist input. 

#### A.6.3. Error conditions

When an error condition occurs inside a package, a message describing what went wrong will be printed (see PRINT - ASCII output files). The name of the routine in which the error occurred must be included. It is acceptable to terminate execution within a package, but the developer may instead wish to return an error flag through the argument list, see `stpctl.F90`.

---

1The point behind this rule is that packages should not have to know details of the surrounding model data structures, or the names of variables outside of the package. A notable exception to this rule is model resolution parameters. The reason for the exception is to allow compile-time array sizing inside the package. This is often important for efficiency.
A.6.4. Memory management

The main action is to identify and declare which arrays are PUBLIC and which are PRIVATE. As of version 3.3.1 of NEMO, the use of static arrays (size fixed at compile time) has been deprecated. All module arrays are now declared ALLOCATABLE and allocated in either the <module_name>_alloc() or <module_name>_init() routines. The success or otherwise of each ALLOCATE must be checked using the stat=<integer variable> optional argument.

In addition to arrays contained within modules, many routines in NEMO require local, “workspace” arrays to hold the intermediate results of calculations. In previous versions of NEMO, these arrays were declared in such a way as to be automatically allocated on the stack when the routine was called. An example of an automatic array is:

```fortran
SUBROUTINE sub(x)
  REAL :: a(n)
END SUBROUTINE
```

The downside of this approach is that the program will crash if it runs out of stack space and the reason for the crash might not be obvious to the user.

Therefore, as of version 3.3.1, the use of automatic arrays is deprecated. Instead, a new module, wrk_nemo.F90, has been introduced which contains 1-, 2-, 3- and 4-dimensional workspace arrays for use in subroutines. These workspace arrays should be used in preference to declaring new, local (allocatable) arrays whenever possible. The only exceptions to this are when workspace arrays with lower bounds other than 1 and/or with extent(s) greater than those in the wrk_nemo.F90 module are required.

The 2D, 3D and 4D workspace arrays in wrk_nemo.F90 have extents jpi, jpj, jpk and jpts (x, y, z and tracers) in the first, second, third and fourth dimensions, respectively. The 1D arrays are allocated with extent MAX(jpi × jpj, jpk × jpts).

The REAL (KIND = wp) workspace arrays in wrk_nemo.F90 are named e.g. wrk_1d_1, wrk_4d_2 etc. and should be accessed by USE’ing the wrk_nemo.F90 module. Since these arrays are available to any routine, some care must be taken that a given workspace array is not already being used somewhere up the call stack. To help with this, wrk_nemo.F90 also contains some utility routines; wrk_in_use() and wrk_not_released(). The former first checks that the requested arrays are not already in use and then sets internal flags to show that they are now in use. The wrk_not_released() routine un-sets those internal flags. A subroutine using this functionality for two, 3D workspace arrays named zwrk1 and zwrk2 will look something like:

```fortran
SUBROUTINE sub()
  USE wrk_nemo, ONLY: wrk_in_use, wrk_not_released
  USE wrk_nemo, ONLY: zwrk1 => wrk_3d_5, zwrk2 => wrk_3d_6
  IF(wrk_in_use(3, 5, 6)) THEN
    CALL ctrl_stop('sub: requested workspace arrays unavailable.')
    RETURN
  END IF
  IF(wrk_not_released(3, 5, 6)) THEN
    CALL ctrl_stop('sub: failed to release workspace arrays.')
  END IF
END SUBROUTINE
```

The first argument to each of the utility routines is the dimensionality of the required workspace (1–4). Following this there must be one or more integers identifying which workspaces are to be used/released. Note that, in the interests of keeping the code as simple as possible, there is no use of POINTERs etc. in the wrk_nemo.F90 module. Therefore it is the responsibility of the developer to ensure that the arguments to wrk_in_use() and wrk_not_released() match the workspace arrays actually being used by the subroutine.

If a workspace array is required that has extent(s) less than those of the arrays in the wrk_nemo.F90 module then the advantages of implicit loops and bounds checking may be retained by defining a pointer to a sub-array as follows:

```fortran
SUBROUTINE sub()
  USE wrk_nemo, ONLY: wrk_in_use, wrk_not_released
  USE wrk_nemo, ONLY: wrk_3d_5
  REAL(wp), DIMENSION(:, :, :), POINTER :: zwrk1
  IF(wrk_in_use(3, 5)) THEN
    CALL ctrl_stop('sub: requested workspace arrays unavailable.')
  END IF
END SUBROUTINE
```
Sect. A.7 Features to be avoided

Here, instead of “use associating” the variable zwrk1 with the array wrk_3d_5 (as in the first example), it is explicitly declared as a pointer to a 3D array. It is then associated with a sub-array of wrk_3d_5 once the call to wrk_in_use() has completed successfully. Note that in F95 (to which NEMO conforms) it is not possible for either the upper or lower array bounds of the pointer object to differ from those of the target array.

In addition to the REAL (KIND = wp) workspace arrays, wrk_nemo.F90 also contains 2D integer arrays and 2D REAL arrays with extent (jpi, jpk), i.e. xz. The utility routines for the integer workspaces are iwrk_in_use() and iwrk_not_released() while those for the xz workspaces are wrk_in_use_xz() and wrk_not_released_xz().

Should a call to one of the wrk_in_use() family of utilities fail, an error message is printed along with a table showing which of the workspace arrays are currently in use. This should enable the developer to choose alternatives for use in the subroutine being worked on.

When compiling NEMO for production runs, the calls to wrk_in_use() / wrk_not_released() can be reduced to stubs that just return .false. by setting the cpp key key_no_workspace_check. These stubs may then be in-lined (and thus effectively removed altogether) by setting appropriate compiler flags (e.g. “-finline” for the Intel compiler or “-Q” for the IBM compiler).

A.6.5. Optimisation

Considering the new computer architecture, optimisation cannot be considered independently from the computer type. In NEMO, portability is a priority, before any too specific optimisation.

Some tools are available to help: for vector computers, key_vectopt_loop allows to unroll a loop

A.6.6. Package attribute: PRIVATE, PUBLIC, USE, ONLY

Module variables and routines should be encapsulated by using the PRIVATE attribute. What shall be used outside the module can be declared PUBLIC instead. Use USE with the ONLY attribute to specify which of the variables, type definitions etc... defined in a module are to be made available to the using routine.

A.6.7. Parallelism using MPI

NEMO is written in order to be able to run on one processor, or on one or more using MPI (i.e. activating the cpp key key_mpp_mpi). The domain decomposition divides the global domain in cubes (see NEMO reference manual). Whilst coding a new development, the MPI compatibility has to be taken in account (see .//src/LBC/lib_mpp.F90) and should be tested. By default, the x-z part of the decomposition is chosen to be as square as possible. However, this may be overridden by specifying the number of sub-domains in latitude and longitude in the nammpp section of the namelist file.

A.7. Features to be avoided

The code must follow the current standards of FORTRAN and ANSI C. In particular, the code should not produce any WARNING at compiling phase, so that users can be easily alerted of potential bugs when some appear in their new developments. Below is a list of features to avoid:

- COMMON block (use the declaration part of MODULE instead)
- EQUIVALENCE (use POINTER or derived data type instead to form data structure)
- Assigned and computed GOTO (use the CASE construct instead)
- Arithmetical IF statement (use the block IF, ELSE, ELSEIF, ENDIF or SELECT CASE construct instead)
- Labelled DO construct (use unlabelled END DO instead)
- FORMAT statement (use character parameters or explicit format- specifiers inside the READ or WRITE statement instead)
- GOTO and CONTINUE statements (use IF, CASE, DO WHILE, EXIT or CYCLE statements or a contained ?)
• **PAUSE**

• **ENTRY** statement: a sub-program must only have one entry point.

• **RETURN** is obsolete and so not necessary at the end of program units

• **FUNCTION** statement

• Avoid functions with side effects. **

• **DATA** and **BLOCK DATA** (use initialisers)

**First, the code is easier to understand, if you can rely on the rule that functions don't change their arguments. Second, some compilers generate more efficient code for PURE functions (in FORTRAN 95 there are the attributes PURE and ELEMENTAL), because they can store the arguments in different places. This is especially important on massive parallel and as well on vector machines.**
Ahn, J. and E. J. Brook, 2008: Atmospheric co2 and climate on millenial time scales during the last glacial period. Science, 322 (5898), 8385, URL.


Aumont, O., C. Ethé, A. Tagliabue, L. Bopp, and M. Gehlen, 2015: Pisces-v2: an ocean biogeochemical model for carbon and ecosystem studies. Geoscientific Model Development, 8 (8), 24652513, URL.


Bullister, J., 2017: Atmospheric histories (1765-2015) for cfc-11, cfc-12, cfc-113, cc14, sf6 and n2o (ncei accession 0164584), URL.


Danabasoglu, G., S. G. Yeager, D. Bailey, E. Behrens, M. Bentsen, D. Bi, A. Biastoch, C. Böning, A. Bozec, V. M. Canuto, and et al., 2014: North atlantic simulations in coordinated ocean-ice reference experiments phase ii (core-ii). part i: Mean states. Ocean Modelling, 73, 76107, URL.


Haine, T. W. N., 2006: On tracer boundary conditions for geophysical reservoirs: How to find the boundary concentration from a mixed condition. Journal of Geophysical Research, 111 (C5), URL.

Honjo, S., 1996: Fluxes of Particles to the Interior of the Open Oceans. John Wiley & Sons, URL.


Mouchet, A., 2013: The ocean bomb radiocarbon inventory revisited. Radiocarbon, 55 (34), URL.

Naegler, T., 2009: Reconciliation of excess14c-constrained global co2piston velocity estimates. Tellus B: Chemical and Physical Meteorology, 61 (2), 372384, URL.


Wanninkhof, R., 1992: Relationship between wind speed and gas exchange over the ocean. *Journal of Geophysical Research*, **97** (C5), 7373, URL.

———, 2014: Relationship between wind speed and gas exchange over the ocean revisited. *Limnology and Oceanography: Methods*, **12** (6), 351362, URL.


<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;namntraldf</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>&amp;namntrcdrad</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>&amp;namntrcdmp</td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

### Namelist blocks

- &namntraldf
- &namntrcdrad
- &namntrcdmp
<table>
<thead>
<tr>
<th>Symbols</th>
<th>I</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>.................</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>