1

A. Crise, E. Bournasky, Z. Bournazka

MOM1_NPD User Manual

OGS Technical Report I12/2000-DOGA1
31 August 2000

MOM-NPD Model for the Mediterranean Sea

MOM-NPD was written as a collaborative effort by Alessandro Crise, Guido Crispi using an ecological model originally designed by Renzo Mosetti. The hydrodynamical submodel was developed using GFDL MOM 1 scheme as implemented during MTP I and MTP II MERMAIDS projects in cooperation with Nadia Pinardi.

MOM 1 documentation

. Documentation for MOM 1 primarily consists of several README files . MOM 1 documentation is less than ideal. Some MOM users have referenced the information contained in the documentation files and within the MOM code itself as follows. The MOM 1code itself (written in Fortran-77), as well as supporting utilitiesand some documentation files, are available free of cost via anonymous ftp from a server located at GFDL in Princeton, NJ,. The NPD ecological submodel manual is described in this manual merged together with some relevant art of the MOM 1 manual in order to obtain a comprehensive user manual for the coupled model.

How to use MOM_NPD

Despiite its potential generality, the code should be installed, customoized., compliled and run according some rules,meant to help in particular the beginners.To run the MOM-NPD code, the keyword based customization of the modular code can be acheived in two ways as we will see later in further detail:

- type 1- C languiage preprocessor (cpp), or functional equivalent, and a Fortran-77 compiler
- type 2 - modern g77/f77 -like compilers

In both cases the source codes (identified with .F extenction)that contain C-like directives for the optional inclusion of part of the code, are precompiled with cpp in order to resolve but in type 1 procedure this must be done calling cpp or cc explicitely. With type2 procedure a simple calll to the compiler is sufficient to automatically resolve the cpp directives befor the actual compilation.Some non-essential, but very useful utilities provided with the model assume that the code will be managed on a UNIX(TM) system and that a C-language compiler is available.

The MOM-NPD code uses a small, straight-forward, subset of the directives available in cpp for two general purposes:

1. copying code from one file into other files.
2. identifying and testing blocks of code associated with various model options for conditional compilation.

Computers having C-language compilers and/or those running a UNIX-based operating system, can be expected to have cpp available as a stand alone program. (type 1 precedure).

Many Fortran compilers also have built into them the cpp functionality needed to run the MOM -NPD code, even if not explicitly stated.

Fortran 77 language and exceptions:

The vast majority of the MOM-NPD code conforms to the American National Standards Institute (ANSI) Fortran 77 standards (ANSI, 1978). Two extensions found in most Fortran 77 compilers also appear in the code.

1. Namelist input/output: Different compilers have different syntax specifications for namelist input/output statements. In extreme cases, namelist may not be available at all (since it is not part of the ANSI Fortran-77 standards). The user would then need to convert the code to use standard Fortran read and write statements instead of namelist.
2. Double quote string delimiters, which appear in some write statements in two MOM-NPD files, are not part of ANSI Fortran-77 standards. One can eliminate this by rewriting the statements to use numbered format statements

.Feedback from users indicates that very few if any Fortran code changes are needed to get MOM-NPD to compile on most machines. (We have learned that some Fortran compilers require that all data statements must appear in external block data routines. In such cases, users need to move data statements found in eight or so files into block data routines.)

Memory and Disk:

A single memory requirement for the MOM-NPD code can not be given because the model can be configured in a multitude of ways. The MOM1 code's memory requirements can be sized to match the machine it is to be run on. In fact, users often find their choice of model grid dimensions, input/output options, physics options, and other model characteristics are largely determined by the amount of memory available. Similarly, the amount of disk space required to run the MOM code depends upon the model configuration and code options selected.

It is recommended that the MOM-NPD code be managed on a UNIX-based system, although one can certainly manage the code successfully on other platforms. Here, examples are presented using UNIX commands. The MOM-NPD code has been designed in some ways to take advantage of certain UNIX utilities.

Recommended Directory Structure for MOM-NPD Files

We suggest that users create a specific directory structure on their local machines to hold the MOM-NPD files. The following UNIX commands generate the desired directory structure. (user input in bold type).

cd [*HOME_MOM directory*]

```
mkdir    DATAic        ! directory with input data
mkdir   MOMor       ! directory with original source
mkdir   MOMout     !directory with output list
mkdir   MOMrun   ! directory with run script and symbolic links
mkdir    PERPEbc ! directory with perpetual year forcing functions
```

MOM-NPD Source Code Files

Before beginning to work with the MOM-NPD source code files, we strongly recommend that a copy of the original files be saved as a backup and protected against being overwritten.

* *.F and *.f files: .these are the Fortran source files (subroutines, main programs, and functions). Files ending with .F contain cpp directives, while those ending with .f do not.
* *.h files: these are the so-called include files (groups of parameter statements, common blocks, statement functions, etc., that are inserted into *.F files #include cpp directives).

Creating a MOM-NPD Executable

T Compiling the sorce code: Pre-processing

Computers having C-language compilers and/or those running a UNIX-based operating system, can be expected to have cpp available as a stand alone program. Many Fortran compilers also have built into them the cpp functionality needed to run the MOM-NPD code, even if not explicitly stated. Lines in MOM-NPD code files having a hash symbol (#, sometimes called a pound sign or number sign) in column one are directives recognized by cpp. The MOM 1 code uses a small, straight-forward, subset of the directives available in cpp for two general purposes:

1. copying code from one file into other files
2. identifying and testing blocks of code associated with various model

Options for conditional compilation.

The first type of cpp directive has the form #include "filename.h", where filename.h represents one of the approximately 40 MOM-NPD files having a .h suffix (hereafter referred to as .h files). The contents of .h files are copied to the location of #include directives by cpp. The most frequent use of .h files in the MOM-NPD code is to hold common blocks. Using #include directives ensures that the contents and arrangement of common blocks are identical in all routines in which they are included.

Various options available in MOM-NPD are designed to be turned on or off by

the user before compilation. The placement of a second kind of cpp directive inside MOM code files allows blocks of code to be identified and tested for conditional compilation. This second set of cpp directives will be referred to collectively as test directives. Blocks of code affected by cpp test directives are identified as beginning with lines having the form #ifdef option, #ifndef option or #if expression, where option represents the name of a MOM-NPD option and expression represents potentially more complicated tests involving code options. In the MOM-NPD code, test expressions are constructed using C-language operators (&&=and, ||=or, !=not). Each test directive has a matching #endif directive marking the end of the block of code. Test directives can be nested. If-then-else constructscan be made through the use of #else or #elif expression (else if) lines.

When executing the cpp command, users can specify which MOM code options are to be defined (turned on). For example, the command:

```
cpp -P -Dnamea -Dnameb mom.F mom.f
```

would cause cpp to take the file mom.F as input, process it, and create an output file named mom.f. Two code options (namea and nameb) have been turned on (defined), in this example. We have adopted the convention that Fortran routines containing cpp directives are given the .F suffix. The .f suffix is given to files ready for the Fortran compiler (those that had no cpp directives to begin with and those which already have been passed through cpp).

One can envision cpp as doing its work in the example case in two stages. During the first stage, #include "filename.h" directives in mom.F are replaced by the contents of the .h files themselves. In the second stage, cpp evaluates test directives based upon which code options have been defined by the -D option to cpp.

f77 execution

After the compile is complete, the .f files produced by cpp typically are no longer needed. Modifications should be made only to copies of the original .F, .f and .h files. Users should not attempt to make code modifications to the .f files produced by cpp.

We are aware of two types of situations in which one may wish to retain the cpp output files for a short period of time. Some users find portions of MOM 1 code .F files that contain a high density of cpp directives somewhat difficult to read. In such cases one may find it helpful to view the cpp output files, since only the code relevant to the options selected will

appear. Also, some debugging tools may require the presence of the .f output files.

In a type 2 procedure such compiling phase can be riduced as follows (referring to the base installation included in the test example):

```
f77 -O2    -o ../MOMrun/pe4l31.npdo3  \
              -Dskipland    -Drigidlid  -Dkeepterms -Dislands\
              -Doldrelax -Dconstvmix -Drestorst \
              -Dbiharmonic  -Ddiskless -Dimga   \
              -Ddprec   -Dbio  -Dqb4 -Dceh3 -Dnewlight *.F
```

The level of optimization can be raised to O3 but nome care must be taken because of the possible modifications of the results. Code crashes have been experienced on old IBM AiX 3.2.5 workstations and IBM *xlf* compiler with O3 optimization.
The compiling procedure must be run un MOMor directory and generats an executable code in MOMrun directory. This can be used succesfully also with g77 GNU public domain compiler.

Compiling directives and Options (excerpted from MOM1 user guide)

A very powerful feature of MOM is the ability to define ifdef options and configure the model in a manner suitable for the problem being investgated. The options are quite extensive and range from various subgrid mixing parameterizations to computing performance enhancements. The current implementation is such that if there are more than two options within a catagory there is no default value, so care should be taken to define the desired option. A subroutine "checks" ,wihin MOM, searches the selcted options to see if any of them conflict. For example, if a user in selecting an I/O scheme turned on both "diskless" (fully contained within memory) and "fio" (Fortran dirct access), an obvious inconsistency exists and a message to this effect will be printed and MOM will halt.

Options are enabled by using the "-D option" form on the compiling statement (as shown above in (vi)). This form may also be used on a "cpp" statement (see "cray.run") to preprocess the code before sending separate it to the compiler. Here is a terse summary of what's currently available:

External mode:

Basically, there are four Poisson solvers. All of them require the "rigidlid" option to also be enabled:

"oldrelax" is the one from Mike Cox's ocean model. If you use it, you should twiddle with "sor" (the overrelaxation constant) to minimize the number of scans for your geometry. "crit" controls the accuracy. See "relax.F" for more details.

"congrad5pt" is a conjugate gradient technique. "sor" is not needed here but "crit" still controls the accuracy. This scheme is faster than "oldrelax" but may be a bit less stable in the presents of steep topographic gradients. The surface pressure is calculated only as a basis for comparison with "congrad9pt". See "congr5.F" for more details.

"congrad9pt" is another conjugate gradient solver which uses a 9 point laplacian instead of the 5 point laplacians used by the other schemes. It's advantage is that it is the most accurate & allows the surface pressure to be calculated directly. Reducing "crit" will give better results (as shown by the closed line surface pressure integrals in the test case) than "congrad5pt" because, as in all other schemes, there is significant truncation due to the 5 point numerics. In fact, the accuracy of the streram funtion solution is limited only by "crit" and computer precision.
Its down side is that it may have stability problems with for topographies with sharp gradients and will take more time. How much you ask? It depends on your configuration. Try it and see. See "congr9.F" for more details.

"hypergrid" is a version of "oldrelax" solved using a checkerboard technique: first on the black squares, then on the red ones: Numerically it's very much like "oldrelax" ("sor" & "crit" comments apply)

Lateral mixing schemes:

Lateral mixing applies to both momentum and tracers. One (and only one) scheme must be enabled.

"consthmix" is a linear second order mixing with constant eddy coefficients: "am" for momentum & "ah" for heat. The idea is to choose them just large enough to suppress small scale numerical noise. Typical vaules would be "am" = 1.e7 cm**2/sec for a one degree resolution. Usually "ah" is chosen smaller so as not to diffuse away frontal features.

"biharmonic" is a linear fourth order mixing with constant

coefficients as in "consthmix": "am" for momentum & "ah" for heat. This is more scale selective than "consthmix" (ie: more severly damps the small scale features). "am" should be roughly 10**19 to 10**20 cm**4/sec (the minus sign is in the eqns) for a 1/3 degree resolution.

"nlhmix" is the non-linear horizontal subgrid-scale mixing after Smagorinsky. In this formulation, the horizontal eddy diffusion coefficient is proportional to the horizontal grid length and to the local deformation field. The coefficients are sensitive to the spatial scales of motion and therefore are relatively small in the open ocean, where the scales of motion are comparatively large, and are relatively large in regions where the scales are comparatively small.

Vertical mixing schemes:

Vertical mixing applies to both momentum and tracers. One (and only one) scheme must be enabled. The default is for explicit solution. Option "implicitvmix" will solve them implicitly.

"constvmix" uses constant eddy coeficients "fkpm" & "fkph" for linear second order mixing of momentum & heat in the vertical. In the explicit case (no "implicitvmix"), convective adjustment handles regions of gravitational instability. If "implicitvmix" is selected, then the convective section is bypassed and the large limits "vdclim" & "vvclim" are used to set the mixing coeffiecints which handles the instability.

"ppvmix" is a vertical mixing scheme based on the Pacanowski & Philander richardson mixing scheme (jpo vol 11, #11, 1981).
This parameterization was designed primarily for equatorial models with vertical resolution of about 10 meters between levels in the upper ocean (we were most interested in the structure of the undercurrent). Previous versions of this code assumed the "t" grid was coincident with the "u" grid and gave good results. In the present case, we relaxed this assumption. We tried this a few ways and got lots of numerical noise, (particularly on the equator off Brazil).
The present configuration minimizes noise. If noise develops off steep shelf breaks it can sometimes be suppressed by turning on a bottom drag ("cdbot"). The background mixing "bvdc" should be kept << 1.0 to avoid diffusing the thermocline away on long time scales.
In regions of gravitational instability, vertical mixing limits "vdclim" & "vvclim" are used. In the explicit case (no "implicitvmix") they must satisfy the "cfl" criterion and may be too small to remove

the instability. However, the convective adjustment is operative and tries to remove the instability. Whether it does or not depends on "ncon" which controlls the number of passes through the convective section. If "implicitvmix" is used, "vdclim" & "vvclim" are set large and the convective adjustment section is bypassed.
This scheme also assumes the use of equal time steps on the density and momentum equations. If a longer time step is desired, try using the "implicitvmix" option.

"tcvmix" invokes the Mellor-Yamada level 2.5 turbulence closure scheme. Here the mixing coefficients are a function of the turbulence length scale (l), the turbulent kinetic energy (q**2), the analytically derived stability factors (Sm & Sh), and the boundary conditions. There are two options to compute the length scale:

"leq" solves an additional equation for q**2l from which the length scale may be derived at each level.

"lalg" obtains the length scale from an algebraic relationship. This option may be sufficient for the boundary layer but not in cases where there would be multiple turbulent regimes.

The horizontal diffusion coefficient of turbulent kinetic energy is set using the variable "aq" ("ctcmix.h", namelist, & "blkdta.f"). For this scheme to be effective "implicitvmix" should be enabled and also the vertical resolution should be sufficient.

"implicitvmix" solves the vertical diffusion term implicitly for all vertical mixing schemes. This allows for large mixing coefficients without the need to reduce the timestep. With this option enabled, convective adjustment is not done and the unstable density profile is mixed using large mixing coefficient limits (vvclim & vdclim). For "constvmix" and "ppvmix" these limits are set in "blkdta.F" and for "tcvmix" the coefficients are computed.
"implicitvmix" also requires that "aidif" - the implicit factor - be set (see "cvmix.h", namelist, and "blkdta.F").

0 < aidif < .5 , stable if kappa*dt/dz**2 < 1/(2-4*aidif) ;
.5 < aidif < 1 , always stable
where kappa is the max vertical mixing coefficient

Hybrid mixing schemes:

These are schemes which apply to either momentum or tracers but not both.

For all cases in which "implicitvmix" is not enabled, convective adjustment is the default. This mixes tracers vertically in regions of gravitational instability but NOT momentum. The effectiveness of convective adjustment is controlled by "ncon" which is the number of passes through the convective section (use "grep -i -n ncon *.[Ffh]" to see its usage). If "impllicitvmix" is enabled, the convective adjustment section is bypassed. The assumption is that the vertical mixing schemes will handle it.

"isopycmix" is a scheme in which a mixing tensor is computed from the local isopycnal slope and the diffusion of tracers is then conducted along that direction. It therefore influences both the lateral and vertical mixing of tracers in the model. Its use is intended to partially mitigate a perceived shortcoming of z-coordinate primitive equation ocean models in parameterizing oceanic mixing due to mesoscale motions. Since such mixing is believed to take place along isopycnal surfaces, the "isopycmix" option seeks to orient the mixing of tracers along isopycnal surfaces rather than purely horizontal surfaces. As described by Cox (Ocean Modelling, issue 74, pg 1-5, June 1987), in addition to specifying the mixing coefficient for along isopycnal diffusion "ahisop", a non-zero background horizontal mixing coefficient "ah" is often needed to supress gridpoint noise. Additionally, a constraint is placed on the steepest isopycnal slope "slmxr" that the scheme will consider when computing the components of the mixing tensor. A typical value of "slmxr" is 100.0, which translates into a slope of 1:100. Steeper slopes would then be considered to be 1:100 for the purpose of computing the mixing tensor. A vertical mixing scheme must be specified for use with "isopycmix". The zz component of the isopycnal mixing is added to the vertical diffusion coefficient produced by the vertical mixing scheme. Since "isopycmix" only affects tracers, one must also specify a lateral mixing scheme from the above list to be used for momentum. This additional lateral mixing scheme has no effect on tracer diffusion.

Grid options:

"cyclic" is for setting cyclic boundary conditons in a zonal direction. If enabled, anything exiting the eastern side of the grid comes back in through the western side. If not enabled, then solid walls are assumed on the eastern and western boundaries of the model.

"symmetry" is for setting a symmetric condition across the northern boundary of the model. This assumes the line of symmetry is at the equator which should be row jmt-1 on the velocity grid.
The condition is :
Tracers at row jmt on the "t" grid = tracers at row jmt-1 on the "t" grid.
meridional velocity at row jmt-1 on the "u" grid is zero.
meridional velocity at row jmt on the "u" grid is the negative of the meridional velocity at row jmt-2 on the "u" grid.
stream funcion at row jmt = negative of the stream function at row jmt-1 on the "t" grid.
If not enabled, solid walls exist at the northern and southern boundaries.

Optimizations:

"skipland" allows calculations to be done over blocks of ocean while skipping land points. The trade off here is the time saved by not doing the caluclation over land versus the time used in starting & stoping a lot of vectors. It's really a function of land mass geometry. A global ocean would be a good candidate for this option, but a idealized basin woulod not.

"multitasking" allows the slabs to be divided into tasks. Each task contains approximately the same number of slabs and there should ideally be one task per processor. This is all handled by simply spedcifying "ntasks" to be the number of processors (we've defaulted it to 8 in "blkdta.F"). Since all tasks are independent, all tasks can be worked on simultaneously. This reduces the total wall clock time for the job but NOT the total cpu time.
Note: "ntasks" may be set > 1 even when unitasking (one processor) but this is not recommended since there is extra work being done at the interfaces between tasks. Also, on diagnostic, tracer averaging, and data saving time steps, MOM reverts to one task of jmt-2 slabs so effectively multitasking is shut down for these time steps.
"multitasking" currently doesn't work with the "diskless" (see below) option because only two ("ntlev"=2) time levels are used for slabs on simulated disk to save memory and this is incompatible with "ntasks" > 1. This condition may, in principle, be removed by setting "ntlev" = 3 when enabeling the "diskless" option. We haven't explored it yet.
So far we have gotten to the point where all combinations of unitasked, multitasked, & autotasked runs give the same answers down

to the last hex digit.
To really utilize this feature on the CRAY YMP, the SSD must be used for the slabs (unless large memory & "diskless" is used). We have not yet added the best i/o scheme for the SSD, so the wall clock times are higher than they should be.

"timing" gives cp & wall clock times for running on a CRAY YMP. Additionally, the time per grid point per time step is calculated.

Filtering:

Filtering is used to combat the effect of the convergence of meridians (shrinking longitudinal grid spacing) near the poles. The problem is that the grid spacing severly limits the time step (especially when using large density time steps) due to the "CFL" criterian. Filtering relaxes this constraint by truncating high wave numbers.

"fourfil" is a fourier filter which acts on longitudinal strips of ocean points. The number of wave numbers to be allowed at each latitude is decided upon & wavenumbers above are truncated (without using any reasonable window). The best that can be said for it is that it is time consuming, messes the solution up & should only be done as a last resort. It is provided as a backward compatability feature for the COX code. Note: Both in MOM and in the COX code, the amount of filtering depends on the number of ocean points within the longitudinal stip. This actually leads to possibly different truncations at the same latitude!

"firfil" is a simple finite impluse response filter based on the familiar 1/4, 1/2, 1/4 weights (the response function is a cosine). It comes in two flavors: One to with symmetric boundary conditions (ie: for tracers) & one for asymmetric boundary conditons (ie: momentum). It also works on longitudinal strips of ocean points & may be applied an arbitrary number of times for an arbitrary number of latitudes. The best that can be said for it is that the reason for using it is the same as for "fourfil", it is quite fast, & we really haven't done any long running comparisons to comapre it to "fourfil".

Miscellaneous:

"keepterms" allows the use of arrays instead of statement functions for component terms in the equations. This may be desirable when using component terms over & over for some purpose. The trade off is the

extra memory needed for the arrays versus the extra time needed to recalculate the statement functions. See "size.F" to look at resource requirements.

"nohilats" stands for no high latitudes. If the latitudinal domain of the model is limited to equatorial redions, for instance, then the metric nonlinear terms in the momentum equations may be dropped. It will save some time.

"restorst" stands for restore surface tracers to some prescribed values by a newtonian damping term. In the test case, the restoration is globally back to the annual mean conditions on a time scale of 50 days. Without much work, this option could easily be extended to damp certain regions while leaving others alone by simple making the damping time scale a function of latitude (for instance).

"testcfl" tests whether any velocity exceeds the local "cfl" criterion by a factor of "cflcrt" (see "blkdta.F"). It will print out the places where the "cfl" criterion is most nearly met on diagnostic time steps. If the "cflcrt" factor is exceeded on any time step, MOM will stop & print out the surronding variables. It's not meant to be left on all the time. If MOM blows up, it can quickly point to where it's happening.

I/O options:

"diskless" simulates disk usage using an array in memory. Since there is no "wall clock" time lost during disk transfers (because they are memory to memory transfers), this method gives the best efficiency (cp time / total time). The down side is that the model may not be able to fit into available memory. As a memory saving feature, "diskless" keeps only two time levels of prognostic variables instead of the usual three. See "size.F" to look at resource requirements. Also see "multitasking"

"crayio" for now uses the "getwa/putwa" routines. If the disk units are set up on "non SSD" type disk, then the efficiency (cp time / total time) will be bad. This shows the mismatch in speed between computation & disk use. SSD, however, is really the way to go. It can be looked at as an extended memory & the efficiency is much better.

"fio" is for fortran direct access i/o. The comments from "crayio" apply here also.

Biological options

" bio" introduces the code part responsible for the evolution of biological state variables; this option has to be used in conjunction with biological input files for what regards the initial condions in the basin plus the restoring data in the Gibraltar buffer zone.

"bioinit" but be called at the same time with "bio" option in order to initialize biological variables. It has besn meant tomallow a re initialization of the biological code after a spinup of the dynamical part. (5 years of spinup for example are standard for such model with the paramenters used in the sample run and only then the biological part can be reinitialized to avoid possible problems at the beginning whre the baroiclinic adjustment introduces hish spurous advective fluxes, pretty dangerous for the correct num,erical approximation of the biogeochemical tracer advection).

"bioinit" must be called only once and then the program must be recomplied with simply the "bio" option to hold the restart information unaltered.

Running the executable

To run the executable a number of step must be done in order to the script with your computational environment:

In the following a sample script is reported together with main comments (the most probable modifications are highlited in *italics)* . The large part of the run script is general for MOM-NPD formulation,although can be altered in case of digfferent forcing functions (sa in the case of high frequency forcings)

```
#=========================================================
# Script file to run the MOM model on the SGI Origin 2000
#                  architecture
#=========================================================
#
#-----run identification-----
echo '***NPD*Emil&Zvetanka**'
echo 'no clouds'
echo 'run identification', $1

#----- working directory and subdirectories -----
#
run=/oga3/echouser/NPD/MOMrun
src=/oga3/echouser/NPD/MOMor
indata=/oga3/echouser/NPD/DATAic
```

```
outdata=/oga3/echouser/NPD/MOMout
forcing=/oga3/echouser/NPD/PERPEbc
#
#----- symbolical links to initial and surface boundary conditions
ln -s -f   $indata/medtop431.dat        fort.33
ln -s -f   $indata/medts431.dat         fort.53
ln -s -f   $indata/levsal               fort.44
ln -s -f   $indata/initbio.data         initbio.data
ln -s -f   $indata/azoini.data          azoini.data
#
#------Assign output units-----
#
ln -s -f $outdata/rst.$1   fort.21
ln -s -f $outdata/hst.$1   fort.22
ln -s -f $outdata/tke.$1   tke
ln -s -f $outdata/bio.$1   bio
#ln -s -f  /dev/null    fort.70
#
#------link clouds files  -----
#
ln -s -f $forcing/clouds/cloud_apr80    cloud_apr80
ln -s -f $forcing/clouds/cloud_aug80    cloud_aug80
ln -s -f $forcing/clouds/cloud_dec80    cloud_dec80
ln -s -f $forcing/clouds/cloud_feb80    cloud_feb80
ln -s -f $forcing/clouds/cloud_jan80    cloud_jan80
ln -s -f $forcing/clouds/cloud_jul80    cloud_jul80
ln -s -f $forcing/clouds/cloud_jun80    cloud_jun80
ln -s -f $forcing/clouds/cloud_mar80    cloud_mar80
ln -s -f $forcing/clouds/cloud_may80    cloud_may80
ln -s -f $forcing/clouds/cloud_nov80    cloud_nov80
ln -s -f $forcing/clouds/cloud_oct80    cloud_oct80
ln -s -f $forcing/clouds/cloud_sep80    cloud_sep80
#
#------link NMC files  -----
#
ln -s -f $forcing/yr1980/apr80_ave      apr80_ave
ln -s -f $forcing/yr1980/aug80_ave    aug80_ave
ln -s -f $forcing/yr1980/dec80_ave    dec80_ave
ln -s -f $forcing/yr1980/feb80_ave    feb80_ave
ln -s -f $forcing/yr1980/jan80_ave    jan80_ave
ln -s -f $forcing/yr1980/jul80_ave    jul80_ave
ln -s -f $forcing/yr1980/jun80_ave    jun80_ave
ln -s -f $forcing/yr1980/mar80_ave    mar80_ave
ln -s -f $forcing/yr1980/may80_ave    may80_ave
ln -s -f $forcing/yr1980/nov80_ave    nov80_ave
ln -s -f $forcing/yr1980/oct80_ave    oct80_ave
ln -s -f $forcing/yr1980/sep80_ave    sep80_ave
```

```
#
# run the model
time pe4l31.$1 << EOF > $outdata/out.$1 2>&1
#time pe4l31.$1 << EOF
 &contrl  days=360.,dgnstc=90.,tsi=10,
          nmix=20, eb=T,restrt=T,init=T, ncon=20, travg=90.,
          snaps=30, taver=4000.,&end
 &eddy    am=1.6e19, ah=.8e19, fkpm=1.5, fkph=0.3, aidif=0.0,
          &end
 &tsteps  dtts=2400., dtuv=2400., dtsf=2400., &end
 &parms   acor=0., mxscan=400, sor=1.60, crit=.5e8,&end
 &iland   alonis(1) = 1.25, alonis(2)= 3.25,
          alonis(3)=9.5,    alonis(4)=25.25, alonis(5)=33.5,
          alatis(1)=39.0,   alatis(2)= 39.5, alatis(3)=41.,
          alatis(4)=35.25, alatis(5)= 35.0,&end
 &eco     kn=0.25, g=1.157e-5, kp=1.0,alpha=0.75,
          drate=5.55e-7, rprime=1.18e-6, rkphy=0.000, rkdet=0.000,
          conc=0., acl=0, growth_max=6.83e-6, texp=0.0633,&end
EOF
rm cloud_*
rm *ave
rm *.data
rm fort.*
echo MOMdone
```

The script can be activated typing

```
./run npdo
```

Namelists

The namelists included in this version of MOM-NPD in the version compiled with the type 2 procedure above reported are basically mentioned

CZCS Dataset ***(after GDAAC-MLD-01_013 CZCS Guide, modified)***

The color of the ocean is determined primarily by the abundance of phytoplankton and their associated photosynthetic pigments. As the concentration of phytoplankton pigments increases, ocean color shifts from blue to green. Taking advantage of this change, NASA developed the Coastal Zone Color Scanner (CZCS) which was launched on the Nimbus-7 satellite in October 1978. During its 7 1/2 year life-time

(October 1978 - June 1986), CZCS acquired nearly 68,000 im-ages, each covering up to 2 million square kilometers of oceansurface.

The Coastal Zone Color Scanner (CZCS) was a multi-spectral line scanner devoted principally to measurements of ocean color. Ithad six spectral bands (channels). There were four channels de-voted to ocean color, each of 20 nanometer band width and centered at 443, 520, 550, and 670 nanometers. These are referred to as channels 1 through 4, respectively. Channel 5 sensed reflectedsolar radiance and had a 100 nanometer bandwidth centered at 750 nanometers and a dynamic range which was more suited to land.

Channel 6 operated in the 10.5 to 12.5 micrometer region and sensed emitted thermal radiance for derivation of equivalent black body temperature. The CZCS level 1, 2 and 3 data productsare available from the Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC).For further details see

http://eosdata.gsfc.nasa.gov/DATASET_DOCS/czcs_dataset.html

The Coastal Zone Color Scanner Level 3 images are available for the Mediterranean in the GrADS machine-independent format and can be prcocessed according the Goddard Space Flight Center prcedure.

CZCS data procedure for the conversion of chlorophyll into Nitrogen equivalent biomass

Original data t be inserted with newtonian relaxation derive from 12 1/4 degreee matrices with Chl mnthly mean CZCS data averaged in the perid 1979 1986 expressed in mg Chl/m**3. These dataset is expressed in arbitrary units (counts) which are converted into pigments measurements through an empirical relationship. Further informations of data processing can be obtainied from the above www address.

The conversion of Chl in Nitrogen- equivalent Phyto (according to Cloern et al., 1995)needs to take into account Nitrogen concentration, water temperature T, and the angular integrated Photosynthetic Available Radiation I

$$Chl(T,I,N) = [0.003 + 0.0154 * \exp(0.050T) * \exp(-0.0059I)\ N/(N+Cn) * PhytoC]$$

where Chl and Carbon are expressed in mg, Phyto is expressed in mgAtN/m**3 and

$$PhytoC = Phyto * 14 * Redfield\_ratio$$

where

$Redfield\_ratio$ = C:N ratio in phytoplankton = 5.68 mgC/mgN

Inverting the above formula we have

$$\begin{aligned} Phyto &= PhytoC/(14*Redfield\_ratio) \\ &= [Chl - 0.003]/[0.0154*\exp(0.050T)*\exp(-0.0059I)\ N/(N+Cn)] \\ &\quad *1/(14*Redfield\_ratio) \end{aligned}$$

T,I and N can be obtained from the the prognostic fields of the models, *t(i,k,j,nm,1)* is temperature (°C),*t(i,k,j,nm,4) i*s dissolved inorganic nitrogen(matN/m**3) and irradiance is expressed in mol quanta m**-2 day**-1

Acknowledgments

For the dynamical part these notes are inspired by

Pacanowski, R., K. Dixon and A. Rosati (1991,1993), The G.F.D.L Modular Ocean Model Users Guide version 1, GFDL Ocean Group Technical Report No. 2, NOAA/Geophysical Fluid Dynamics Laboratory, Princeton, NJ.

The original coupled model was developed under EU MAS2-CT93-0055 and MAS3-CT96-0051 'MEDNET' contracts . The data assimilation scheme was implemented under NATO Linkage Grant 974997