

HYBRID **C**OORDINATE **O**CEAN **M**ODEL

HYCOM for Dummies

How to create a double gyre configuration with HYCOM from scratch?

Updated - December 2020

Alexandra Bozec

abozec@coaps.fsu.edu



Foreword	3
1. BB86_PACKAGE and what's in it	3
2. The Double Gyre configuration or BB86.....	4
3. HYCOM-BB86	5
a. What you need to get HYCOM running.....	5
i. Create input files.....	6
ii. Compiling the HYCOM source code	8
b. Let's make it run	11
i. blkdat.input	12
ii. script_exe.com	16
c. How to restart the simulation.....	17
4. Plotting the outputs	18
Appendix.....	20
References	22

Foreword

This document is destined to people who wants to use HYCOM but have no idea where to start, how to create HYCOM format input files, how to read HYCOM format outputs files or just want to play with a double gyre configuration and see what it does.

A small disclaimer, as much as one can check their codes, it is still possible for bugs to dig their way into our otherwise well-tested scripts. If you find one, first, I apologize in advance (Good Job for finding it!) and second I would appreciate if you can fix it and send me the fix (Even a Better Job!), so I can continue improving the package (Many Thank You!).

1. BB86_PACKAGE and what's in it

The BB86_package is available on the HYCOM website (<http://www.hycom.org>).

Once downloaded, untar it:

```
~/> tar xvf BB86_PACKAGE.tar.gz
BB86_PACKAGE/
BB86_PACKAGE/IDL/
BB86_PACKAGE/IDL/UTILITIES/
BB86_PACKAGE/IDL/UTILITIES/write_depth_hycom.pro
BB86_PACKAGE/IDL/UTILITIES/write_grid_hycom.pro
BB86_PACKAGE/IDL/UTILITIES/spherdist.pro
BB86_PACKAGE/IDL/UTILITIES/sigma2_hycom.pro
BB86_PACKAGE/IDL/UTILITIES/read_depth_hycom.pro
BB86_PACKAGE/IDL/UTILITIES/read_grid_hycom.pro
BB86_PACKAGE/IDL/UTILITIES/colorbar2.pro
BB86_PACKAGE/IDL/UTILITIES/sofsig.pro
BB86_PACKAGE/IDL/UTILITIES/tofsig.pro
BB86_PACKAGE/IDL/UTILITIES/write_relax_hycom.pro
.
.
.
BB86_PACKAGE/BB86/bb86.f
BB86_PACKAGE/BB86/input
BB86_PACKAGE/PS/
BB86_PACKAGE/PS/uv_dp_d100_bb86-hycom.ps
BB86_PACKAGE/PS/uv_dp_d200_bb86-hycom.ps
BB86_PACKAGE/PS/uv_dp_d360_bb86-hycom.ps
~/> ls
BB86_PACKAGE  BB86_PACKAGE.tar.gz
~/> cd BB86_PACKAGE
~/BB86_PACKAGE> ls
BB86  config  expt_01.0  force  IDL  MATLAB  PS  PYTHON  relax  src_2.3.01_one
topo
~/BB86_PACKAGE >
```

Your PACKAGE is now ready to be used. The layout is described Figure 2.

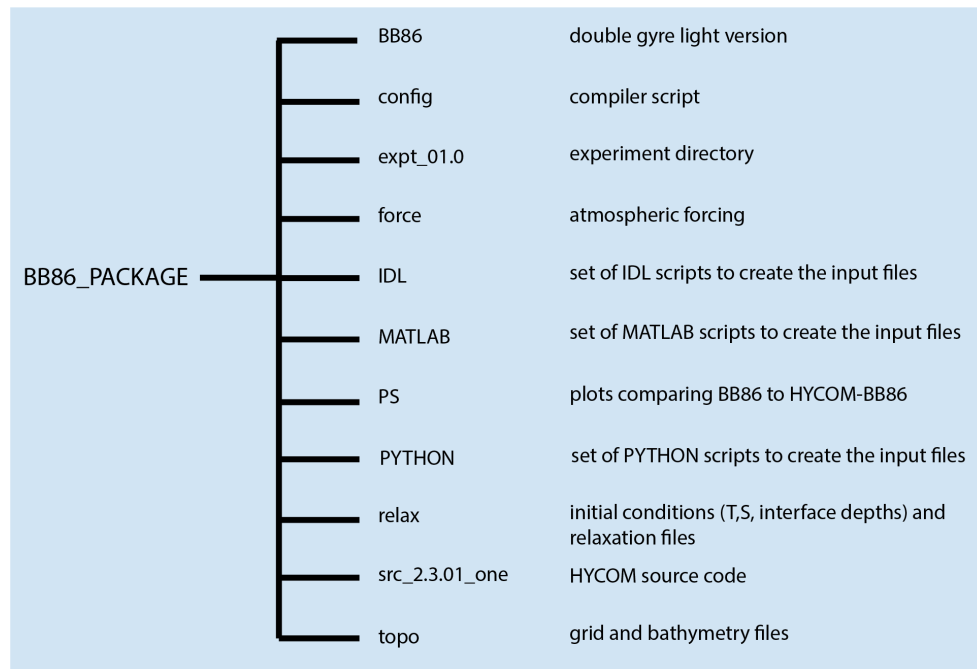


Fig 2: Double gyre configuration directories

2. The Double Gyre configuration or BB86

The double gyre configuration that we are going to run here is a relatively simple configuration based on the publication of Bleck and Boudra (1986). The model domain is 101x101 rectangular basin with a horizontal resolution of 20km and a flat bottom at 5000m. Coriolis varies with Beta and an eastward wind-stress is applied as described Figure 1.

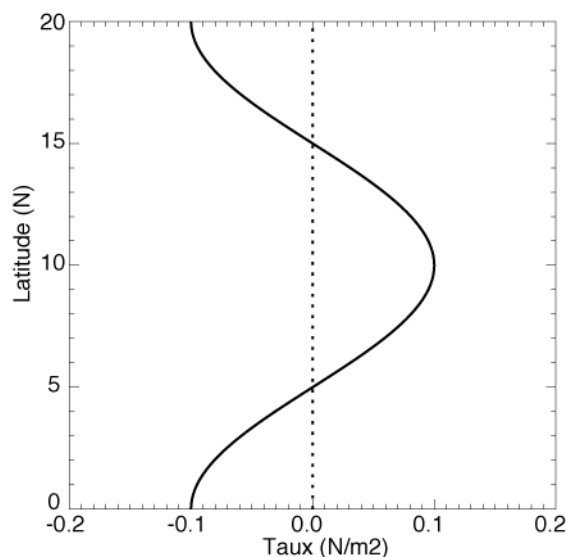


Fig 1: eastward wind-stress profile

We are now going to run a HYCOM version of this simple configuration, named hereafter HYCOM-BB86.

3. HYCOM-BB86

a. What you need to get HYCOM running

For those who do not know anything about HYCOM, I would recommend to and take a look at the HYCOM for Dummies (v2.2):

http://hycom.org/attachments/349_HYCOM_for_Dummies.pdf.

and/or browse the website documentation:

<http://www.hycom.org/hycom/documentation>

(N.B.: Before starting and as a reminder, one must know that all inputs and outputs are presented as a couple of .a and .b files. The .a file actually contains the binary data and the .b file describes what is in the .a file and how it is stored. In the following, the notation *.ab] will correspond to the couple *.a and *.b.)

As any ocean model, HYCOM requires a grid, a bathymetry, an initial state of temperature, salinity and layer interface depth, as well as an atmospheric forcing. For the double gyre experiment, here are the files that HYCOM is going to need:

- regional.grid.ab] : definition of the horizontal grid
- regional.depth.ab] : definition of the bathymetry of the domain
- relax.temp.ab] : temperature of the initial state
- relax.saln.ab] : salinity of the initial state
- relax.intf.ab] : pressure of the interface depths of each layer of the initial state
- forcing.tauewd.ab] : eastward wind stress
- forcing.taunwd.ab] : northward wind stress

Scripts to create those files in a HYCOM-friendly format are provided in IDL, MATLAB and PYTHON. As MATLAB and PYTHON users will notice, the scripts are “translated” from the IDL ones, that is why they are so similar and not using a more efficient (I am sure !) MATLAB or PYTHON syntax. Those scripts are of course in no way set in stone, so please feel free to change them for your needs. **(N.B.:** the IDL, PYTHON codes and MATLAB differs slightly first because IDL and PYTHON loops start at 0 and second because MATLAB arrays reverse the first two dimensions compared to the IDL (or FORTRAN).)

In all directories (IDL, MATLAB and PYTHON), you will find the scripts that have been used to create the input files of the example double gyre experiment (expt_01.0) and a directory named **UTILITIES**. The UTILITIES directory hosts functions able to easily write input and read output files.

Either in IDL, MATLAB or PYTHON, five files are provided and must be run in a certain order since some of them requires files created beforehand:

- 1- write_depth_bb86.pro/m/py : create the bathymetry
- 2- write_grid_bb86.pro/m/py : create the grid
- 3- write_relax_bb86.pro/m/py : create the T, S and layer interface pressure initial state
- 4- write_windstress_bb86.pro/m/py : create the wind-stress files

Although all the files are already there, a good exercise is to replicate an experiment to be sure the set-up is right and before playing with the bathymetry, the forcing or the parameters.... To do so, we are going to redo all the input files (with different names; we do not want to erase the original files), compile HYCOM, and run the simulation. If your output plots are identical or close (which is more likely to happened if different compiler are used...) to the ones provided with that package (~/.BB86_PACKAGE/PS/) then you will have succeeded in running HYCOM from scratch.

So let's start!

i. Create input files

To create new inputs files, first go into the directory IDL, MATLAB or PYTHON depending on your preferences. Since we are replicating a simulation, the only thing to be changed in the IDL/MATLAB/PYTHON script write_depth_bb86.pro/m/py is the name of the output file.

For example, you can change it from 'depth_BB86_01' to 'depth_DBG_01' as below:

```
~/BB86_PACKAGE> cd IDL
~/BB86_PACKAGE/IDL> vi write_depth_bb86.pro
PRO write_depth_bb86

;; include the Utilities functions
iodir = '/Net/yucatan/abozec/BB86_PACKAGE/IDL/' ;; where you are
!path=expand_path('+' +iodir+'UTILITIES')+' ':' +expand_path('+' +!dir)

close, /all
;; PATH
```

```

io = iodir+'../topo/'
file_bat_new = 'depth_DBG_01' ;; !! without .a or .b !!
pl = 0 ;; 1 or 0 for plot or not
.
.
.
write_depth_hycom, idm_hd, jdm_hd, io+file_bat_new, bathy_hd
print, 'Writing depth file done '

stop
END
~
~
"write_depth_bb86.pro" 74L, 2145C
~/BB86_PACKAGE/IDL>

```

Once changed, run the idl command and create the file as follow:

```

~/BB86_PACKAGE/IDL> idl
IDL> write_depth_bb86
% Compiled module: WRITE_DEPTH_BB86.
Depth Ok
% Compiled module: WRITE_DEPTH_HYCOM.
Writing depth file done
% Stop encountered: WRITE_DEPTH_BB86 73 ~/BB86_PACKAGE/IDL/write_depth_bb86.pro
IDL> exit
~/BB86_PACKAGE/IDL>

```

MATLAB users can follow the same steps in the MATLAB/PYTHON directory.

Congratulations! You now have a new bathymetry file in
~/BB86_PACKAGE/topo.

```

~/BB86_PACKAGE/topo> ls -l
total 1020
-rw-r--r-- abozec coaps 49152 May 13 10:46 depth_BB86_01.a
-rw-r--r-- abozec coaps 78 May 13 10:46 depth_BB86_01.b
-rw-r--r-- abozec coaps 933888 May 13 10:46 regional.grid.BB86.a
-rw-r--r-- abozec coaps 1054 May 13 10:46 regional.grid.BB86.b
-rw-r--r-- abozec coaps 49152 May 16 16:05 depth_DBG_01.a
-rw-r--r-- abozec coaps 70 May 16 16:05 depth_DBG_01.b
~/BB86_PACKAGE/topo>

```

Repeat the same steps for the grid, initial state (relax) and atmospheric files.

N.B.: one particularity to know about the grid file is that the HYCOM model does not even read the longitude/latitude of the grid. Only the dx, dy, angle of the grid, coriolis factor and the dx/dy aspect are actually used (respectively, [puvq]scx, [puvq]scy, pang, cori, pasp).

If everything goes according to plan, you should have now all the input files necessary to run HYCOM-BB86. We now need the *hycom* executable to actually run.

ii. Compiling the HYCOM source code

The HYCOM source code is located in the `src_2.3.01_one` directory. This set of subroutines is based on the latest available version of HYCOM (<https://github.com/HYCOM/HYCOM-src>). However, few minor changes have been performed to fit the BB86 configuration. These changes are mainly in the `mod_momtum.F90` routine and are flagged by “!!Alex” throughout the code. These changes also include the introduction of two new flags in the `blkdat.input` (*input parameter file of HYCOM; see `expt_01.0` directory*):

`cbar2` : linear bottom drag

`pstrsi` : depth over which the wind is applied (**N.B:** should be 0 if not in a BB86 configuration).

Therefore, the `blkdat.input` provided by the standard HYCOM package (<https://github.com/HYCOM/HYCOM-examples>) cannot be used with the BB86_package.

To compile the HYCOM source directory you first need to edit the `Make.csh` to define the right compiler. Compilation files for different types of machine are available in `~/BB86_PACKAGE/src_2.3.01_one/config`. The one used here is `intellFC_one` using the `ifort` compiler (**N.B.:** the **one** refers to `src_2.3.01_one` and the fact that we are going to run with only one processor; if you want to run with multiprocessor (**mpi**), please see choose one of the `mpi` scripts and copy `src_2.3.01_one` into `src_2.3.01_mpi` and copy `src_2.3.01_mpi/ALT_CODE/dimensions_mpi.h` into `src_2.3.01_mpi/dimensions.h`).

In this version of HYCOM, the equation of state is now also defined with Macros set at compilation. Here, we use `sigma-0` and a 7-term equation of state:

```
setenv OCN_SIG -DEOS_SIG0 ## Sigma-0
setenv OCN_EOS -DEOS_7T  ## EOS 7-term
```

```
~/BB86_PACKAGE/src_2.3.01_one/> cd config
~/BB86_PACKAGE/src_2.3.01_one/config> ls
amd64-relo_one          intelIFC-relo_one      xc40-gnu-relo_mpi
amd64-relo_one~         intel-relo_one         xc40-gnu-relo_omp
gfortran-relo_one       intelsse-impi-sm-relo_mpi xc40-gnu-relo_one
hpe-impi-relo_mpi       intelsse-impi-sm-relo_omp xc40-intel-relo_cesmb
hpe-impi-relo_omp       intelsse-impi-sm-relo_one xc40-intel-relo_cice
hpe-intel-relo_mpi      intelsse-pe-sm-relo_mpi xc40-intel-relo_mpi
hpe-intel-relo_omp      power9-xl-smpi-relo_mpi xc40-intel-relo_omp
hpe-intel-relo_one      README.src.config      xc40-intel-relo_one
```



```

intelGF-impi-sm-relo_mpi  README.src.config.macros  xc40-pgi-relo_mpialphaL_one
~/BB86_PACKAGE/src_2.3.01_one/config> cd ../
~/BB86_PACKAGE/src_2.3.01_one> vi Make.csh
#!/bin/csh
#
set echo
cd $cwd
#
# --- Usage:  ./Make.com >& Make.log
#
# --- make hycom with TYPE from this directory's name (src_*_$TYPE).
# --- assumes dimensions.h is correct for $TYPE.
#
# --- set ARCH to the correct value for this machine.
# --- ARCH that start with A are for ARCTIC patch regions
#
# unset echo
#module swap compiler compiler/intel/12.1.3
#module swap mpi      mpi/intel/ibmpe
#module list
#set echo
#setenv ARCH Aintelsse-pe-sm-relo
#
setenv ARCH intelIFC-relo
#
setenv TYPE `echo $cwd | awk -F"_" '{print $NF}`
echo "ARCH = " $ARCH "  TYPE = " $TYPE
#
if (! -e ./config/${ARCH}_${TYPE}) then
    echo "ARCH = " $ARCH "  TYPE = " $TYPE "  is not supported"
    exit 1
endif

# CPP flags for compilations
# Equation Of State
#setenv OCN_SIG -DEOS_SIG2 ## Sigma-2
setenv OCN_SIG -DEOS_SIG0 ## Sigma-0

setenv OCN_EOS -DEOS_7T  ## EOS  7-term
~
"Make.csh" 72L
~/BB86_PACKAGE/src_2.3.01_one>

```

Then run the Make.csh to compile the model:

```

~/BB86_PACKAGE/src_2.3.01_one> csh Make.csh
cd /Net/yucatan/abozec/GitHub/BB86_PACKAGE/src_2.3.01_one
echo /Net/yucatan/abozec/GitHub/BB86_PACKAGE/src_2.3.01_one
/Net/yucatan/abozec/GitHub/BB86_PACKAGE/src_2.3.01_one
setenv ARCH intelIFC-relo
setenv TYPE `echo $cwd | awk -F"_" '{print $NF}`
awk -F_ '{print $NF}'
echo /Net/yucatan/abozec/GitHub/BB86_PACKAGE/src_2.3.01_one
echo ARCH = intelIFC-relo  TYPE = one
ARCH = intelIFC-relo  TYPE = one
if ( ! -e ./config/intelIFC-relo_one ) then
setenv OCN_SIG -DEOS_SIG0
setenv OCN_EOS -DEOS_7T
setenv OCN_GLB

```

[illegible]

```

ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c overtn.F90
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c poflat.F90
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c prtmsk.F90
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c psmoo.F90
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c thermf.F90
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c trcupd.F90
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c machine.F90
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c wtime.F90
gcc -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -O -c machi_c.c
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c isnan.F90
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c s8gefs.F90
ifort -DIA32 -DREAL8 -DENDIAN_IO -DNAN2003 -DTIMER -DRELO -DEOS_SIG0 -DEOS_7T -g -assume byterecl -
cm -vec_report0 -w -O3 -tpp7 -xW -r8 -c hycom.F90
ifort -g -assume byterecl -cm -vec_report0 -w -O3 -tpp7 -xW -r8 -Bstatic -o hycom hycom.o
mod_dimensions.o mod_xc.o mod_zc.o mod_cb_arrays.o mod_pipe.o mod_incupd.o mod_floats.o mod_stokes.o
mod_tides.o mod_mean.o mod_archiv.o mod_tsadv.o mod_momtun.o mod_barotp.o mod_asselin.o mod_restart.o
mod_hycom.o bigrid.o blkdat.o cnuity.o convec.o diapfl.o dpthuv.o dpudpv.o forfun.o geopar.o
hybgen.o icloan.o inicon.o inigiss.o inikpp.o inimy.o latbdy.o matinv.o mxkprf.o mxkrt.o mxkrtn.o
mxpwp.o overtn.o poflat.o prtmsk.o psmoo.o thermf.o trcupd.o machine.o wtime.o machi_c.o isnan.o
s8gefs.o
ipo: warning #11010: file format not recognized for mod_cb_arrays.o
mod_cb_arrays.o: file not recognized: File truncated
make: *** [hycom] Error 1
if ( intelIFC-relo == Asp5 || intelIFC-relo == sp5 ) then
if ( intelIFC-relo == Asp6 || intelIFC-relo == sp6 ) then
if ( intelIFC-relo == Asp6-nofl || intelIFC-relo == sp6-nofl ) then
~/BB86_PACKAGE/src_2.3.01_one>

```

If nothing wrong happens, congrats! You have compiled successfully HYCOM. You should have an executable named **hycom** in your directory.

N.B.: If you change something in one of the routines and re-compile, it is usually better to remove all *.o and *.mod as well as the **hycom** executable beforehand.

b. Let's make it run

Now all the input files including the hycom executable are ready to be used. To run the model, go to the expt_01.0 directory. This directory includes one directory data, in which you are actually going to run, and two files:

- blkdat.input : the parameter file
- script_exe.com : the “submission” script.

```

~/BB86_PACKAGE/src_2.3.01_one> cd ../expt_01.0
~/BB86_PACKAGE/expt_01.0> ls -l
total 16
-rw-r--r-- 1 abozec coaps 11000 May 16 14:46 blkdat.input

```

```
drwxr-xr-x 2 abozec coaps      6 May 17 15:07 data
-rw-r--r-- 1 abozec coaps 2630 May 16 15:57 script_exe.com
~/BB86_PACKAGE/expt_01.0>
```

i. blkdat.input

Let's start with what is in the blkdat.input:

```
~/BB86_PACKAGE/expt_01.0> vi blkdat.input
BB86 config ; analytic wind-stress
3 layers (first one very thinto mimick a 2 layers config);
src_2.3.01;
1234567890123456789012345678901234567890123456789012345678901234567890
 23      'iversn' = hycom version number x10
010      'iexpt'  = experiment number x10
101      'idm'    = longitudinal array size
101      'jdm'    = latitudinal array size
  6      'itest'  = grid point where detailed diagnostics are desired
  6      'jtest'  = grid point where detailed diagnostics are desired
  3      'kdm'    = number of layers
  3      'nhybrd' = number of hybrid levels (0=all isopycnal)
  0      'nsigma' = number of sigma levels (nhybrd-nsigma z-levels)
 1.0     'dp00'   = deep z-level spacing minimum thickness (m)
 1.0     'dp00x'  = deep z-level spacing maximum thickness (m)
 1.0     'dp00f'  = deep z-level spacing stretching factor (1.0=const.space)
 1.0     'ds00'   = shallow z-level spacing minimum thickness (m)
 1.0     'ds00x'  = shallow z-level spacing maximum thickness (m)
 1.0     'ds00f'  = shallow z-level spacing stretching factor (1.0=const.space)
 1.0     'dp00i'  = deep iso-pycnal spacing minimum thickness (m)
 1.0     'isotop' = shallowest depth for isopycnal layers (m), <0 from file
0.1      'oneta0' = minimum 1+eta, must be > 0.0
37.0     'saln0'  = initial salinity value (psu), only used for iniflg<2
  1      'locsig' = locally-referenced pot. density for stability (0=F,1=T)
  0      'kapref' = thermobaric ref. state (-1=input,0=none,1,2,3=constant)
  0      'thflag' = reference pressure flag (0=Sigma-0, 2=Sigma-2)
27.01037 'thbase' = reference density (sigma units)
  0      'vsigma' = spacially varying isopycnal target densities (0=F,1=T)
27.01000 'sigma'  = layer 1 isopycnal target density (sigma units)
27.01037 'sigma'  = layer 2 isopycnal target density (sigma units)
27.22136 'sigma'  = layer 3 isopycnal target density (sigma units)
  2      'iniflg' = initial state flag (0=levl, 1=zonl, 2=clim)
  0      'jerlv0' = initial jerlov water type (1 to 5; 0 to use KPAR)
  0      'yrflag' = days in year flag (0=360, 1=366, 2=366J1, 3=actual)
  0      'sshflg' = diagnostic SSH flag (0=SSH,1=SSH&stericSSH)
 1.0     'dsurfq' = number of days between model diagnostics at the surface
 1.0     'diagfq' = number of days between model diagnostics
  0.0     'proffq' = number of days between model diagnostics at selected locs
  0.0     'tilefq' = number of days between model diagnostics on selected tiles
  0.0     'meanfq' = number of days between model diagnostics (time averaged)
10.0     'rstrfq' = number of days between model restart output
  0.0     'bnstfq' = number of days between baro nesting archive input
  0.0     'nestfq' = number of days between 3-d nesting archive input
```

In **yellow**, the domain size is defined with *idm*, *jdm* and *kdm*. The target density chosen in your relax files are defined for each layer with *sigma*. Here, the configuration uses the surface as a reference density so *thflag* = 0.

In **green**, the vertical structure of the water column is defined: *nhybrd* defines the number of layer to be hybrid (so able to go from z to isopycnal to terrain following or any combination); *nsigma* defines the number of **terrain-following** layers (!!! WARNING don't get it confused with the number of isopycnal layer which is *nhybrd* – *nsigma* !!!). **N.B.:** If using hybrid layers, the first layer is **always** a z-coordinate level; that is why to trick the model into a 2-layer ocean (as the BB86 configuration), we use 3 layers but make the first one thin with a 1m thickness. Then, we want to keep the two remaining layers moving freely as an isopycnal; the deep and shallow z-level spacing thicknesses are thus put to 1m (dp00, dp00x, ds00, ds00x).

In **blue**, output frequencies are defined in days: *diagfq* will give you an instantaneous output (i.e. a snapshot at midnight). For time-averaged output, use *meanfq*. However, users should be aware that the 3D variables are also averaged in time, which is fine when vertical fixed-coordinates are used, but can be tricky when working with a **time-varying vertical** grid as HYCOM. The way it is calculated in HYCOM is as follow:

$$T_m = \sum_{t=1}^n (T * thk * oneta) / \sum_{t=1}^n thk * oneta$$

```

0.125 'cplifq' = number of days (or time steps) between sea ice coupling
900.0 'baclin' = baroclinic time step (seconds), int. divisor of 86400
45.   'batrop' = barotropic time step (seconds), int. div. of baclin/2
0     'incflg' = incremental update flag (0=no, 1=yes, 2=full-velocity)
12    'incstp' = no. timesteps for full update (1=direct insertion)
1     'incupf' = number of days of incremental updating input
0     'stfflg' = stochastic anomaly forcing flag (0=no, 1=TS, 2=TSV)
50.0  'stfrdt' = stochastic T anomaly forcing e-folding depth (m)
50.0  'stfrds' = stochastic S anomaly forcing e-folding depth (m)
50.0  'stfrdv' = stochastic V anomaly forcing e-folding depth (m)
0.125 'ra2fac' = weight for Robert-Asselin time filter
0.125 'wbaro'  = barotropic time smoothing weight
1     'btrlfr' = leapfrog barotropic time step (0=F,1=T)
0     'btrmas' = barotropic is mass conserving (0=F,1=T)
0     'hybraf' = HYBGEN: Robert-Asselin flag (0=F,1=T)
8.0   'hybrlx' = HYBGEN: inverse relaxation coefficient (time steps)
0.01  'hybiso' = HYBGEN: Use PCM if layer is within hybiso of target density
3     'hybmap' = hybrid remapper flag (0=PCM, 1=PLM, 2=PPM, 3=WENO-like)
0     'hybflg' = hybrid generator flag (0=T&S, 1=th&S, 2=th&T)
0     'advflg' = thermal advection flag (0=T&S, 1=th&S, 2=th&T)
2     'advtyp' = scalar advection type (0=PCM,1=MPDATA,2=FCT2,4=FCT4)
2     'momtyp' = momentum advection type (2=2nd order, 4=4th order)
-1.0  'slip'   = +1 for free-slip, -1 for non-slip boundary conditions
0.005 'visco2' = deformation-dependent Laplacian viscosity factor

```

```

0.0      'visco4' = deformation-dependent biharmonic viscosity factor
0.0      'facdf4' = speed-dependent biharmonic viscosity factor
0.005    'velddf2' = diffusion velocity (m/s) for Laplacian momentum dissip.
0.00    'velddf4' = diffusion velocity (m/s) for biharmonic momentum dissip.
0.0      'thkdf2' = diffusion velocity (m/s) for Laplacian thickness diffus.
0.01     'thkdf4' = diffusion velocity (m/s) for biharmonic thickness diffus.
0.015    'temddf2' = diffusion velocity (m/s) for Laplacian temp/saln diffus.
1.0      'temdfc' = temp diffusion conservation (0.0,1.0 all dens,temp resp.)
0.e-5    'vertmx' = diffusion velocity (m/s) for momentum at MICOM M.L.base
0.02     'cbar'   = rms flow speed (m/s) for bottom friction
0.e-3     'cb'     = coefficient of quadratic bottom friction
1.e-7     'cbar2'  = linear bottom drag (bb86)
0.0      'drglim' = limiter for explicit friction (1.0 none, 0.0 implicit)
10.0     'thkbot' = thickness of bottom boundary layer (m)
0.0      'drgscl' = scale factor for tidal drag (0.0 for no tidal drag)
500.0    'thkdrgr' = thickness of bottom boundary layer for tidal drag (m)
10.0     'thkbot' = thickness of bottom boundary layer (m)
0.02     'sigjmpl' = minimum density jump across interfaces (kg/m**3)
0.3      'tmljmpl' = equivalent temperature jump across mixed-layer (degC)
.
.
.
0        'mlflag' = mixed layer flag (0=none,1=KPP,2-3=KT,4=PWP,5=MY,6=GISS)
0        'pensol' = KT: activate penetrating solar rad. (0=F,1=T)
999.0    'dtrate' = KT: maximum permitted m.l. detrainment rate (m/day)
19.2     'thkmin' = KT/PWP: minimum mixed-layer thickness (m)
0        'dypflg' = KT/PWP: diapycnal mixing flag (0=none, 1=KPP, 2=explicit)
6.e20    'mixfrq' = KT/PWP: number of time steps between diapycnal mix calcs
1.e-7     'diapyc' = KT/PWP: diapycnal diffusivity x buoyancy freq. (m**2/s**2)
0.25     'rigr'   = PWP: critical gradient richardson number
.
.
.

```

In yellow are defined the barotropic and baroclinic time steps in seconds (*batrop* and *baclin*). In green, the advection viscosity, diffusion as well as the advection type are selected. Note that *visco2* and *velddf2* are in **m/s**. To get the actual viscosity and diffusion factor, one need to multiply by the horizontal scale Δx (or Δx^3 for biharmonic). In blue, all coefficient related to bottom friction are defined. In purple, you will find the mixed layer parameterization flag. User can choose between KPP, Krauss-Turner, PWP, Mellor-Yamada or GISS. The one usually used is KPP (*mlflag* = 1). For this configuration, we do not use any vertical mixed-layer parameterization.

```

.
.
.
1        'ismpfl' = FLOATS: sample water properties at float (0=no; 1=yes)
4.63e-6  'tbvar'  = FLOATS: horizontal turb. vel. variance scale (m**2/s**2)
0.4      'tdecrl' = FLOATS: inverse decorrelation time scale (1/day)
0        'lbflag' = lateral barotropic bndy flag (0=none, 1=port, 2=input)

```

```

0      'lbmont' = baro nesting archives have sshflg=2
0      'tidflg' = TIDES: tidal forcing flag      (0=none,1=open-bdy,2=bdy&body)
0      'tidein' = TIDES: tide field input flag (0=no;1=yes;2=sal)
00000001 'tidcon' = TIDES: 1 digit per (Q1K2P1N2O1K1S2M2), 0=off,1=on
0.06    'tidsal' = TIDES: scalar self attraction and loading factor
0      'tidrg' = TIDES: tidal drag flag (0=no;1=scalar;2=tensor)
500.0   'thkdr' = TIDES: thickness of tidal drag bottom boundary layer (m)
0.0     'drsgcl' = TIDES: scale factor for tidal drag (0.0 when tidrg=0)
1       'tidgen' = TIDES: generic time (0=F,1=T)
1.0     'tidrmp' = TIDES: ramp time (days)
0.0     'tidt0' = TIDES: origin for ramp time (model day)
12      'clmflg' = climatology frequency flag (6=bimonthly, 12=monthly)
2       'wndflg' = wind stress input flag (0=none,1=u/v-grid,2,3=p-grid)
50.     'pstrsi' = depth over which the wind is apply (m) (> 0 for bb86 only )
4       'ustflg' = ustar forcing flag (3=input,1,2=wndspd,4=stress)
0       'flxflg' = thermal forcing flag (0=none,3=net-flux,1,2,4=sst-based)
0       'empflg' = E-P forcing flag (0=none,3=net_E-P, 1,2,4=sst-bas_E)
0.0     'emptgt' = E-P balance target (mm/week, into ocean)
0       'empbal' = E-P balance flag (0=none,1=offset,2=scale)
0       'dswflg' = diurnal shortwave flag (0=none,1=daily to diurnal corr.)
0       'albflg' = ocean albedo flag (0=none,1=const,2=L&Y)
0       'sssflg' = SSS relaxation flag (0=none,1=clim)
0       'sssbal' = SSS rlx balance flag (0=none,1=offset,2=scale)
0       'lwflag' = longwave (SST) flag (0=none,1=clim,2=atmos)
0       'sstflg' = SST relaxation flag (0=none,1=clim,2=atmos,3=observed)
0       'icmflg' = ice mask flag (0=none,1=clim,2=atmos,3=obs/coupled)
1.e5    'prsbas' = msl pressure is input field + prsbas (Pa)
0       'mslprf' = msl pressure forcing flag (0=F,1=T)
0       'stroff' = net strs offset flag (0=F,1=T)
0       'flxoff' = net flux offset flag (0=F,1=T)
0       'flxsmo' = smooth surface fluxes (0=F,1=T)
0       'relax' = activate lateral boundary nudging (0=F,1=T)
0       'trcrlx' = activate lat. bound. tracer nudging (0=F,1=T)
0       'priver' = rivers as a precipitation bogas (0=F,1=T)
0       'epmass' = treat evap-precip as a mass exchange (0=F,1=T)
"blkdat.input" 159L
~/BB86_PACKAGE/expt_01.0>

```

Finally, by the end of the blkdat.input, you will find in yellow the parameter *lbflag* for the treatment of the boundaries. Here we are in a “closed basin” (bathymetry at each boundary), so we do not need any boundary condition (*lbflag* = 0). **N.B.:** For E-W **cyclic boundary condition**, only open your eastern boundary (no land on the eastern boundary) and HYCOM will run automatically in “cyclic basin” (*lbflag* stays 0.).

In green, everything relevant to the atmospheric forcing is defined. For the HYCOM-BB86 configuration, only the wind-stress is used. Since we usually uses wind-stress on the *p-grid* (see below for more explanation on that), *wndflg* = 2. Parameter *pstrsi* defines the depth over which the wind-stress is applied. This parameter is specific to the double gyre configuration and do not appears in the standard version. Parameter *ustflg* defines the way to calculate u^* : here we calculate it from the wind-stress that we are providing, so *ustflg* = 4.

Small Interlude for those who have no idea what is the p-grid: HYCOM uses an Arakawa C-grid (you can look that up on the web to see the difference between A-, B-, C- and so on grid; Arakawa and Lamb, 1977):

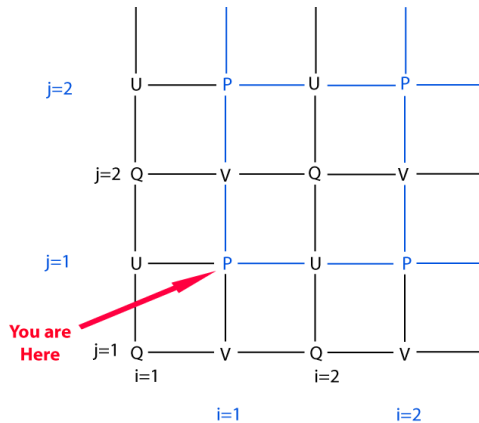


Fig. 4: Horizontal HYCOM C-grid, p-grid in blue.

ii. script_exe.csh

The script_exe.csh is a C-shell script that basically is going to get all the input files in the same location (~BB86_PACKAGE/expt_01.0/data), run the model and put the output in a safe directory (~BB86_PACKAGE/expt_01.0/data/output/). See below, the highlighted part of the code to change before running (if necessary):

```
~/BB86_PACKAGE/expt_01.0> vi script_exe.csh
#!/bin/csh

set echo
## CONFIGURATION BOX 20kmx20km
setenv R BB86    ## config name    <= TO BE CHANGED TO DBG in our case
setenv T 01      ## topography version (see name of your file..)
setenv E 010     ## experiment number (ex: 010)
setenv E1 `echo ${E} | awk '{printf("%04.1f", $1*0.1)}'` ## (ex: 01.0)

## time to run
setenv day1 0
setenv day2 11    <= Number of days you want to run
## restart run?
setenv restart -  ## - means no restart (+ means restart) <= Start from restart or not

## get all the files in the same directory
# P : path where you are going to launch the run
# D : path where you are actually running
setenv P /Net/yucatan/abozec/BB86_PACKAGE/expt_${E1}/    <= Your path
setenv D ${P}/data
```



```
mkdir -p ${D}

## get the grid and depth files
touch ${D}/regional.grid.[ab] ${D}/regional.depth.[ab]
/bin/rm ${D}/regional.grid.[ab] ${D}/regional.depth.[ab]

.
.
.
echo 'You are Done !'
"script_exe.com"      92L
~/BB86_PACKAGE/expt_01.0>
```

Then the BIG ONE: Let's actually run!!

```
~/BB86_PACKAGE/expt_01.0> csh script_exe.csh > test_DBG.log
.
.
.
echo You are Done !
~/BB86_PACKAGE/expt_01.0>
```

When the simulation is completed without hiccups, a message "You are Done !" should appear. **Congratulations!** You have successfully run HYCOM!

c. How to restart the simulation

The restart frequency is determined by *rstfrq* in the blkdat.input. The value for this simulation is 10 days. A restart is usually also created at the end of the simulation. The names of the restart output alternates between restart_out.[ab] and restart_out1.[ab]. To check which one is the latest one:

```
~/BB86_PACKAGE/expt_01.0/data> more restart_out.b
RESTART2: iexpt,iversn,yrflag,sigver =      10      22      0      1
RESTART2:  nstep,dttime,thbase   =                  10656      111.000000000000
27.01037000000000
u      : layer,tlevel,range =    1    1    -6.9688775E-02    3.6270935E-02
u      : layer,tlevel,range =    2    1    -6.0354847E-02    3.6645178E-02
u      : layer,tlevel,range =    3    1    -3.4554186E-03    6.1953445E-03
u      : layer,tlevel,range =    1    2    -6.7957908E-02    3.5531744E-02
u      : layer,tlevel,range =    2    2    -6.0342751E-02    3.6634427E-02
u      : layer,tlevel,range =    3    2    -3.4555271E-03    6.1951848E-03
v      : layer,tlevel,range =    1    1    -5.8089945E-02    1.5086172E-02
v      : layer,tlevel,range =    2    1    -6.1239716E-02    1.5086172E-02
v      : layer,tlevel,range =    3    1    -1.7712242E-03    7.7249869E-03
--More--
~/BB86_PACKAGE/expt_01.0/data>
```

In **yellow** is highlighted the day at which the restart is written. To restart from that day:

- Move the restart_out.[ab] in restart_in.[ab]
- Edit script_exe.csh by changing day1 from 0 to 111 and day2 from 111 to the new date you wish to finish your second simulation.
- Edit script_exe.csh by changing restart from - to +

```
~/BB86_PACKAGE/expt_01.0/data> mv restart_out.a restart_in.a
~/BB86_PACKAGE/expt_01.0/data> mv restart_out.b restart_in.b
~/BB86_PACKAGE/expt_01.0/data> cd ../
~/BB86_PACKAGE/expt_01.0> vi script_exe.csh
#!/bin/csh

set echo
## CONFIGURATION BOX 20kmx20km
setenv R BB86    ## config name
setenv T 01      ## topography version (see name of your file..)
setenv E 010     ## experiment number (ex: 010)
setenv E1 `echo ${E} | awk '{printf("%04.1f", $1*0.1)}'` ## (ex: 01.0)

## time to run
setenv day1 111      <= 1
setenv day2 360      <= 2
## restart run?
setenv restart +     ## - means no restart (+ means restart) <= 3

## get all the files in the same directory
# P : path where you are going to launch the run
# D : path where you are actually running
setenv P /Net/yucatan/abozec/BB86_PACKAGE/expt_${E1}/
setenv D ${P}/data
"script_exe.com" 92L
~/BB86_PACKAGE/expt_01.0>
```

Then, you can launch the second simulation:

```
~/BB86_PACKAGE/expt_01.0> csh script_exe.csh > test_2_bb86.log
```

4. Plotting the outputs

After the simulation is completed, the outputs from the HYCOM simulation are moved to ~/BB886_PACKAGE/expt_01.0/data/output (directory name that can be changed by editing the script_exe.csh). Example scripts to plot those results are available in ~/BB886_PACKAGE/IDL/plot_res_bb86.pro, ~/BB886_PACKAGE/MATLAB/plot_res_bb86.m or ~/BB886_PACKAGE/PYTHON/plot_res_bb86.py. Postscript figures from these example can be found in ~/BB886_PACKAGE/PS/.

Another small interlude to talk about the archive files (*.ab) of HYCOM:

If you are using instantaneous archive file (named **archv.***), then the **total** velocity is split into a **barotropic** component (u_btrop and v_btrop in the .b file) and a **baroclinic** component (u-vel and v-vel). If you are using mean archive files (named **archm.***) then u-vel and v-vel are the **total** zonal and meridional velocity. You can get back the **baroclinic** velocity by subtracting the **barotropic** components u_btrop and v_btrop. (**N.B.:** NetCDF velocity outputs provided on the hycom website are **total** velocities.)

Also, please check the IDL/MATLAB/PYTHON scripts for the information about the units of the different variables: archive files use pressure (or pressure like units) for most of the thing related to depth (mongt1, srfhgt, bl_dpht, thknss, etc...), so a conversion is necessary. If you are using the archv2ncdf scripts provided in the HYCOM-tools package (available on GitHub: <https://github.com/HYCOM/HYCOM-tools>), conversions are done when creating the NetCDF.

Below is a figure from the IDL script plot_res_bb86.pro showing the velocity fields and layer thickness anomaly from the HYCOM-BB86 at day 1800. The corresponding Matlab code, plot_res_bb86.m, will give the same representation with some differences in the colorbar and contours.

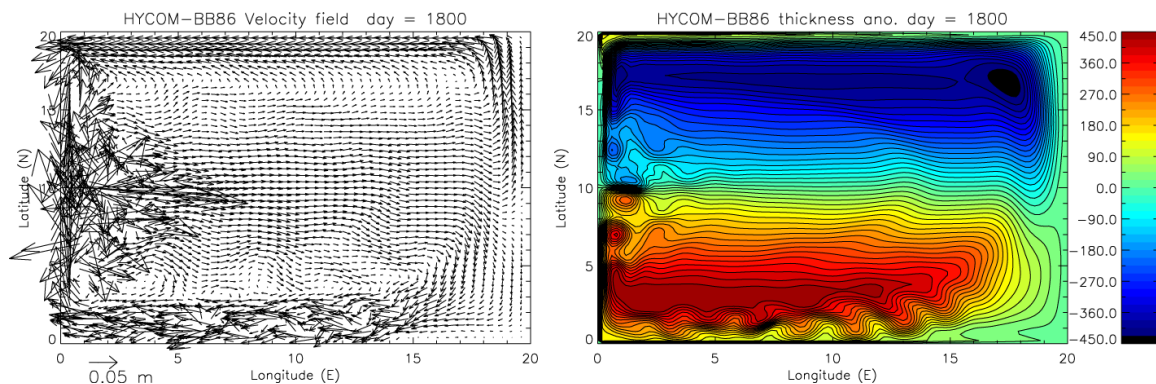


Fig. 5: HYCOM-BB86 results at day 1800 (i.e. after 5 years).

If your results look like that: Life is good! You can play with this configuration by changing whatever comes to mind: bathymetry, wind pattern, vertical distribution of layers ... Have Fun!

Appendix

A light version of this configuration (BB86) is available within the BB86_PACKAGE (~/.BB86_PACKAGE/BB86/). To run this configuration, simply go to the directory and compile (**N.B.:** only an *ifort* compiler is available; but feel free to use your own):

```
~/BB86_PACKAGE> cd BB86
~/BB86_PACKAGE/BB86> ls
-rwxr-xr-x 1 abozec coaps 619233 May 17 15:14 bb86.exe
-rw-r--r-- 1 abozec coaps 45516 May 17 15:14 fft.f
-rw-r--r-- 1 abozec coaps 125 May 17 15:15 compil.csh
-rw-r--r-- 1 abozec coaps 38456 May 17 15:17 bb86.f
-rw-r--r-- 1 abozec coaps 33 May 31 14:36 input
~/BB86_PACKAGE/BB86> csh compil.csh
bb86.f(197): remark #8291: Recommended relationship between field width 'W' and
the number of fractional digits 'D' in this edit descriptor is 'W>=D+7'.
554      FORMAT(2X'Viscosity : 'e10.4)
-----^
bb86.f(193): remark #8291: Recommended relationship between field width 'W' and
the number of fractional digits 'D' in this edit descriptor is 'W>=D+7'.
552      FORMAT(2X'GRID SIZE : 'E6.1)
-----^
Compilation done !
~/BB86_PACKAGE/BB86>
```

If you get the message “Compilation done !” and a new executable *bb86* appears in your list of files we are in good shape !

```
~/BB86_PACKAGE/BB86> ls
-rwxr-xr-x 1 abozec coaps 619233 May 17 15:14 bb86.exe
-rw-r--r-- 1 abozec coaps 45516 May 17 15:14 fft.f
-rw-r--r-- 1 abozec coaps 125 May 17 15:15 compil.csh
-rw-r--r-- 1 abozec coaps 38456 May 17 15:17 bb86.f
-rwxr-xr-x 1 abozec coaps 960574 May 17 15:17 bb86
-rw-r--r-- 1 abozec coaps 33 May 31 14:36 input
~/BB86_PACKAGE/BB86>
```

While the most part of the parameters is hard-coded in *bb86.f* (Coriolis, wind-stress...), some of them are in the *input* file.

```
~/BB86_PACKAGE/BB86> more input
0,110
1200
500.e5,4500.e5
-1.0,2
~/BB86_PACKAGE/BB86>
```

In **yellow**, the starting and ending day of the run are defined; in **green**, we define the time step; in **blue**, the thickness of the first and second layer are

defined (*in dyn/cm²*); finally in **purple**, we define the amplitude of the wind-stress (-1.0) **and** free-slip/no-slip condition (1: free-slip/2: no-slip). Please check the bb86.f routine to see how it works.

Then to run the model:

```
~/BB86_PACKAGE/BB86> ./bb86 < input
nday1=          0 nday2 =          360
dp0(1) =  5.0000000E+07 dp0(2) =  4.5000000E+08
stressa =  -1.000000
Model run with no-slip boundary cond.
  NUMBER OF GRID ELEMENTS IN EACH DIRECTION : 101
  GRID SIZE : .2E+07
  TIME STEP : 1200.0
  Viscosity : 0.1000E+07

MODEL INTEGRATION STARTS FROM TIME STEP          0

-1.249E-04-1.404E-03-2.146E-03-2.903E-03-3.646E-03-4.360E-03-5.038E-03-5.697E-03-6.308E-03-6.905E-03-
7.440E-03-7.976E-03
-1.249E-04-1.404E-03-2.146E-03-2.903E-03-3.646E-03-4.360E-03-5.038E-03-5.697E-03-6.308E-03-6.906E-03-
7.440E-03-7.977E-03
T I M E   S T E P           72                D A Y           1.0
THKN  4.94E+07  EPOT  3.40E+01  EKin  1.70E+03  EKINT  6.02E-02  XCONT -6.15E-02  STRESS  1.22E-01
DISSIP -5.45E-04  1
THKN  4.49E+08  EPOT  3.49E+02  EKin  1.90E+03  EKINT  5.38E-02  XCONT  5.42E-02  STRESS -3.74E-04
DISSIP -8.84E-05  2
THKN  4.99E+08  EPOT  3.83E+02  EKin  3.60E+03  EKINT  1.14E-01  XCONT -7.30E-03  STRESS  1.22E-01
DISSIP -6.33E-04
  3.083E-04-3.533E-03-5.352E-03-7.085E-03-8.763E-03-1.035E-02-1.180E-02-1.321E-02-1.447E-02-1.569E-02-
1.673E-02-1.780E-02
  3.082E-04-3.533E-03-5.352E-03-7.085E-03-8.764E-03-1.035E-02-1.180E-02-1.321E-02-1.447E-02-1.569E-02-
1.673E-02-1.781E-02
T I M E   S T E P          144                D A Y           2.0
THKN  4.87E+07  EPOT  1.33E+02  EKin  3.28E+03  EKINT  4.04E-02  XCONT -1.06E-01  STRESS  1.48E-01
DISSIP -1.92E-03  1
THKN  4.49E+08  EPOT  1.41E+03  EKin  8.44E+03  EKINT  9.53E-02  XCONT  9.74E-02  STRESS -1.68E-03
DISSIP -3.55E-04  2
THKN  4.98E+08  EPOT  1.54E+03  EKin  1.17E+04  EKINT  1.36E-01  XCONT -8.27E-03  STRESS  1.46E-01
DISSIP -2.28E-03
  1.372E-03-6.465E-03-9.798E-03-1.275E-02-1.555E-02-1.814E-02-2.045E-02-2.267E-02-2.455E-02-2.641E-02-
2.790E-02-2.947E-02
  1.372E-03-6.465E-03-9.798E-03-1.275E-02-1.555E-02-1.814E-02-2.045E-02-2.268E-02-2.455E-02-2.641E-02-
2.790E-02-2.947E-02
T I M E   S T E P          216                D A Y           3.0
.
.
.
T I M E   S T E P        25920                D A Y          360.0
THKN  3.23E+07  EPOT  2.34E+03  EKin  1.46E+05  EKINT  1.52E-03  XCONT -6.93E-01  STRESS  7.51E-01
DISSIP -5.73E-02  1
THKN  4.39E+08  EPOT  5.32E+06  EKin  4.47E+05  EKINT -4.21E-03  XCONT  2.76E-01  STRESS -8.94E-02
DISSIP -1.90E-01  2
THKN  4.71E+08  EPOT  5.33E+06  EKin  5.93E+05  EKINT -2.69E-03  XCONT -4.17E-01  STRESS  6.62E-01
DISSIP -2.48E-01
(NORMAL)
~/BB86_PACKAGE/BB86>
```

Outputs of the BB86 are stored in Fortran binary files named fort.3[012]; restart files are stored in fort.2[012]. File fort.30 is the total zonal velocity *U* (*cm/s*), fort.31 is the total meridional velocity *V* (*cm/s*) and fort.32 is the layer thickness (*pressure units: dyn/cm²*).

References

- Bleck, R. and Boudra, D., 1986: Wind-driven spin-up in eddy-resolving ocean models formulated in isopycnic and isobaric coordinates. *Journal of Geophysical Research*, 91 (C6), p7611-7621.
- Arakawa, A. and V.R. Lamb, 1977: Computational design and the basic dynamical processes of the UCLA general circulation Model. *Methods in Computational Physics*, 17. p173.