HYbrid Coordinate Ocean Model (HYCOM)

User's Manual

Details of the numerical code

R. Bleck, Los Alamos National Laboratory, USA
G. Halliwell, University of Miami, USA
A. Wallcraft, Naval Research Laboratory, USA
S. Carroll, K. Kelly, K. Rushing, Planning Systems Inc., USA

Code version 2.0.01, Manual version 2.0.01, March 4, 2002
This manual is available at:
http://panoramix.rsmas.miami.edu/hycom/documentation.html
Report corrections to: ????

Contents

List of Figures

List of Tables

Preface

0.1 Background

This manual was written to describe HYCOM, *code* version 2.0.01. This version of the manual was patterned after the MICOM *manual* version 2.6A written by G. Langlois, *Agence de Développements en Hydrodynamique et Océanographie Côtière, France.* The original MICOM User's Manual was written in French, and revised and translated to the English version 2.9 by D. Brydon, Los Alamos National Laboratory, USA, R. Bleck, University of Miami, USA, and S. Dean, Los Alamos National Laboratory, USA.

0.2 Revisions to the manual

MICOM, Version 2.6A, February 7, 1997.

HYCOM, Version 2.0.01, March 4, 2002.

0.3 Acknowledgements

HYCOM version 2.0.01 was released July 2, 2001, with development the result of collaborative efforts between the University of Miami, the Los Alamos National Laboratory, and the Naval Research Laboratory.

0.4 Funding

Ongoing HYCOM research has been funded under the National Oceanographic Partnership Program (NOPP) and the Office of Naval Research (ONR) to develop it for use in a global ocean data assimilation system and as the ocean component of a coupled ocean-atmosphere model.

Introduction

The HYbrid Coordinate Ocean Model (HYCOM; (Halliwell *et al.*, 1998; 2000; Bleck, 2001) is a primitive equation ocean general circulation model that evolved from the Miami Isopycnic-Coordinate Ocean Model (MICOM) developed by Rainer Bleck and colleagues. MICOM has become one of the premier ocean circulation models, having been subjected to validation studies (e.g. Chassignet *et al.*, 1996; Roberts *et al.*, 1996; Marsh *et al.*, 1996) and used in numerous ocean climate studies (e.g., New and Bleck, 1995; New et al., 1995; Hu, 1996, 1997; Halliwell, 1997, 1998; Bleck 1998 and references therein). HYCOM was developed to address known shortcomings of the MICOM vertical coordinate scheme. MICOM vertical coordinates are isopycnic except for model layer 1, which is a non-isopycnic slab mixed layer. This leads to two significant problems: First, slab models must be used to govern mixed layer entrainment and detrainment. MICOM was equipped with a Kraus-Turner type model as described in Niiler and Kraus (1977), but using the modified turbulent kinetic energy balance parameterization of Gaspar (1988). Second, vertical coordinates are "wasted" because all model layers less dense than the mixed layer (layer 1) exist as zero-thickness layers at the mixed layer base. Although MICOM has produced good scientific results, improvements in the representation of vertical mixing, and the representation of oceanic flow in shallow-water and weakly stratified regions are constrained by these vertical coordinate limitations.

Vertical coordinates in HYCOM remain isopycnic in the open, stratified ocean. However, they smoothly transition to z coordinates in the weakly-stratified upper-ocean mixed layer, to terrain-following sigma coordinate in shallow water regions, and back to level coordinates in very shallow water. The latter transition prevents layers from becoming too thin where the water is very shallow. The vertical coordinates that were "wasted" in MICOM are used to provide vertical resolution within the surface mixed layer. This enables the use of more sophisticated non-slab closure schemes. One important goal for HYCOM is to provide the capability of selecting among several different vertical mixing schemes for both the surface mixed layer and the comparatively weak interior diapycnal mixing. No vertical mixing algorithm can provide a perfect representation of ocean mixing and its influence on ocean circulation and climate. When using an ocean model to study processes sensitive to vertical mixing, it is a good idea to run simulations with different mixing algorithms to quantify the sensitivity of scientific results to the mixing parameterizations. One specific example is the use of coupled ocean-atmosphere models to quantify global warming rates expected from greenhouse gas increases. This warming is likely to be sensitive to the parameterization of vertical mixing in the ocean model. HYCOM is designed to easily test these sensitivities whether used alone or in a coupled system. Sensitivity of ocean mixing to other factors such as the vertical structure and resolution of the vertical grid can also be readily tested.

The K-Profile Parameterization (KPP, Large et al., 1994; 1997) algorithm was included as the first non-slab mixed layer model for several reasons. It provides mixing throughout the water column with an abrupt but smooth transition between the vigorous mixing in the surface boundary layer and the relatively weak diapycnal mixing in the ocean interior. It works on a relatively coarse and unevenly spaced vertical grid. It parameterizes the influence of a larger suite of physical processes than other commonly used mixing schemes. In the ocean interior, the contribution of background internal wave breaking, shear instability mixing, and double diffusion (both salt fingering and diffusive instability) are parameterized. In the surface boundary layer, the influences of wind-driven mixing, surface buoyancy fluxes, and convective instability are parameterized. The KPP algorithm also parameterizes the influence of nonlocal mixing of T and S, which permits the development of countergradient fluxes. The Kraus-Turner slab model has also been included in HYCOM. With the release of HYCOM version 2.1, two additional mixed layer models have been included: the dynamical instability model of Price *et al.* (1986), and the Mellor-Yamada level 2.5 turbulence closure used in the Princeton Ocean Model (Mellor and Yamada, 1982; Mellor, 1998). Other mixed layer models will be included in the near future, such as the model developed recently by Canuto (2000).

1 FORTRAN 90 Version 2.0.01

In version 2.0.01 of the FORTRAN code of HYCOM, the *main* program is named `hycom.f`. The dimensions of various variables are introduced by means of symbolic constants. The corresponding `parameter` statements are grouped in the file **dimensions.h** (*cf.* § 1.1.1). The dimension declarations are grouped in the file **common_blocks.h** (*cf.* § 1.1). The program makes calls to different statement functions grouped in the file **stmt_fns.h** (*cf.* § 18).

Before iterative integration calls, it is necessary to define the different characteristics of a simulation. The setup of HYCOM 2.0.01 can be summarized in the following manner :

1. The first step consists of introducing the bathymetry (*cf.* § 1.4.3) and taking steps to distinguish the submerged zones from those on land. (*cf.* § 1.4.3).
2. The second phase has the goal of characterizing the projection used. Several options may be used, including mercator (dx.eq.dy), rotated mercator, uniform latitude (dx.ne.dy), and square uniform latitude (dx.eq.dy). Generally, in HYCOM the system of equations is solved on a Mercator grid with the x axis indicating East and the y axis pointing North. In HYCOM 2.0.01,the rotated mercator option is not functional. HYCOM uses MKS units throughout, with all input in MKS and with fluxes positive into the ocean.
3. Initialization of the variables is done in the third step (*cf.* § 1.2).
4. The iterative computations are based on ten main procedures depending on which model option is chosen. For the non-slab K-Profile Parameterization model setup (KPP), the subroutines are listed in the order called along with the function it performs below :

   `subroutine cnuity` : continuity equation ;
   `subroutine tsadvc` : advection equation ;
   `subroutine momtum` : momentum equations ;
   `subroutine barotp` : dynamic barotropic mode ;
   `subroutine thermf` : ocean-atmosphere exchanges ;
   `subroutine icloan` : ocean-ice exchanges ;
   `subroutine mxkpp` : k-profile vertical mixing ;
   `subroutine hybgen` : vertical coordinate remapping .

   Alternatively, HYCOM may be run with the Kraus-Turner (KT) model setup. The KT model governs mixing only within the mixed layer, while the KPP model provides mixing from surface to bottom. When the KT model is selected, interior diapycnal mixing can either be explicit (as in MICOM) or implicit (based on a subset of the KPP scheme). If the KT option is chosen, then the subroutine mxkpp is replaced by the following:

`subroutine mxkrta or mxkrtb` : bulk surface mixed layer ;
`subroutine convch` : vertical convection ;
`subroutine diapf1 or diapf2` : diapycnal mixing (explicit, implicit) .

If HYCOM is run in MICOM compatability mode, then mxkpp and hybgen are replaced by:

`subroutine mxkrtm` : bulk surface mixed layer ;
`subroutine convcm` : vertical convection ;
`subroutine diapf3` : diapycnal mixing (explicit MICOM mode) .

HYCOM's scheme needs results of the previous two computations. These results are saved via the last dimension, which has an extent of 2. The index m represents the solution at time step $n\Delta t$ and the index n represents alternately the results at $(n-1)\Delta t$ and $(n+1)\Delta t$.

5. After each iteration, the version 2.0.01 of HYCOM offers a certain number of outputs (writing files, graphical output, *etc.*) as well as one control test. (*cf.* § 1.3.2).

1.1 Declarations

1.1.1 Symbolic constants

The different parametric constants have been grouped in the following list :

`(itdm),idm,ii,ii1`	(total)grid dimension in i direction
`(jtdm),jdm,jj,jj1`	(total)grid dimension in j direction
`kdm,kk`	grid dimension in k direction
`iqr,jqr`	maximum number of tiles in i,j direction
`mxthrd`	maximum number of OpenMP threads
`nbdy`	halo size
`ms,msd`	maximum number of submerged segments in rows or columns
`ahalf,athird,afourth`	1/2, 1/3, 1/4

These values are fixed in **dimensions.h** and in **stmt_fns.h** :

1.1.2 Variables of state and auxiliary variables

The **common_blocks.h** file declares the variables found in the following table :

Table 1: Variables of file **common_blocks.h**

Variable	Type	Description
Common/hycom1r		
u, v	Real	Velocity components.
dp, dpu, dpv	Real	Layer thickness.
p, pu, pv	Real	Interface pressure.
dpold	Real	Layer thickness.
dpoldm	Real	Layer thickness.
corio	Real	Coriolis parameter.
psikk	Real	Montgomery potential in bottom layer.
thkk	Real	Virtual potential density in bottom layer.
potvor	Real	Potential vorticity.
temp	Real	Temperature.
saln	Real	Salinity.
th3d	Real	Potential density.
thstar	Real	Virtual potential density.
diaflx	Real	Time integral of diapyc.flux.
tracer	Real	Inert tracer (optional).
Common/hycom2r		
montg	Real	Montgomery potential.
uflx, vflx	Real	Mass fluxes.
uflxav, vflxav	Real	Average fluxes.
dpav	Real	Average fluxes.
ubavg, vbavg	Real	Barotropic velocity.
pbavg	Real	Barotropic pressure.
defor1, defor2	Real	Deformation components.
ubrhs, vbrhs	Real	Rhs of barotropic u,v equations.
utotm, vtotm, utotn, vtotn	Real	Total (barotropic + baroclinic) velocities at 2 time levels.
uflux, vflux, uflux1, vflux1, uflux2, vflux2, uflux3, vflux3	Real	Horizontal mass fluxes.

Variable	Type	Description
Common/hycom3r		
util1, util2, util3, util4	Real	Arrays for temporary storage.
scux, scuy	Real	Mesh size at u points in x, y direction.
scvx, scvy	Real	Mesh size at v points in x, y direction.
scu2, scv2	Real	Grid box size at u,v points.
scp2, scq2	Real	Grid box size at p, q points.
scuxi, scvyi	Real	Inverses of scux, scvy.
scp2i, scq2i	Real	Inverses of scpx, scqy.
pgfx, pgfy	Real	Horizontal pressure gradient.
gradx, grady	Real	Horizontal pressure gradient.
depthu, depthv	Real	Bottom pressure at u,v points.
pvtrop	Real	Potential vorticity of barotropic flow.
depths	Real	Water depth.
drag	Real	Bottom drag.
Common/hycom4r		
uja, ujb, via, vib	Real	Velocities at lateral neighbor points.
pbot	Real	Bottom pressure at t = 0.
sgain	Real	Salinity changes from diapyc.mix.
surflx	Real	Surface net thermal energy flux.
sswflx	Real	Surface swv thermal energy flux.
salflx	Real	Surface salinity flux.
buoyfl	Real	Net surface buoyancy flux.
buoysw	Real	Shortwave buoyancy flux.
ustar	Real	Friction velocity.
turgen	Real	Turbulent kinetic energy generation.
thkice	Real	Grid-cell average ice thickness (m).
covice	Real	Ice coverage (rel. units).
temice	Real	Ice surface temperature.
Common/hycom4i		
klist	Integer	K-index.
jerlov	Integer	Jerlov water type 1-5.

Variable	Type	Description
Common/hycom5r		
dpmixl	Real	Mixed layer depth.
t1sav	Real	Upper sublayer temperature.
s1sav	Real	Upper sublayer salinity.
tmlb	Real	Temperature in layer containing mlb.
smlb	Real	Salinity in layer containing mlb.
hekman	Real	Ekman layer thickness.
dpbl	Real	Turbulent boundary layer depth.
dpmold	Real	Mixed layer depth.
tmix	Real	Mixed layer temperature.
smix	Real	Mixed layer salinity.
thmix	Real	Mixed layer potential density.
umix, vmix	Real	Mixed layer velocity.
dp0sig	Real	Minimum sigma separation.
dp0k	Real	Minimum z-layer separation.
Common/hycom5i		
nmlb	Integer	Layer containing mlb.
Common/swtchs		
diagno	Logical	Output model fields and diagnostic messages.
thermo	Logical	Use thermodynamic forcing (flxflg > 0).
windf	Logical	Use wind stress forcing (wndflg > 0).
pcipf	Logical	Use evap-precip surface salinity flux.
relax	Logical	Activate lateral boundary nudging.
srelax	Logical	Activate surface salinity nudging.
trelax	Logical	Activate surface temperature nudging.
relaxf	Logical	Input relaxation fields (relax.or.srelax.or.trelax).
hybrid	Logical	Use hybrid vertical coordinates.
isopyc	Logical	Use isopycnic vertical coordinates (MICOM mode).
tbaric	Logical	Include thermobaricity (kappaf).
icegln	Logical	Use energy loan ice model (iceflg =1).
mxlkta	Logical	KT: activate original mixed layer model (mlflag = 2).
mxlktb	Logical	KT: activate alternative mixed layer model (mlflag = 3).
mxlkrt	Logical	KT: activate MICOM or HYCOM Kraus-Turner (mlflag = 2, 3).
pensol	Logical	KT: activate penetrating solar radiation.
mxlkpp	Logical	KPP: activate mixed layer model (mlflag = 1).
shinst	Logical	KPP: activate shear instability mixing.
dbdiff	Logical	KPP: activate double diffusion mixing.
nonloc	Logical	KPP: activate nonlocal boundary layer mixing.
difsmo	Logical	KPP: activate horizontal smooth diff coeffs.
trcrin	Logical	Initialize tracer from restart file.
trcout	Logical	Advect tracer and save results in history/restart file.

Variable	Type	Description
Common/hycom1c		
ctitle	Character	Four lines describing the simulation.
Common/frcing		
pwall	Real	Pressure boundary condition at sidewalls.
swall	Real	Salinity boundary condition at sidewalls.
twall	Real	Temperature boundary condition at sidewalls.
taux	Real	Wind stress in x direction.
tauy	Real	Wind stress in y direction.
wndspd	Real	Wind speed (tke source).
airtmp	Real	Air temperature.
vapmix	Real	Atmospheric vapor mixing ratio.
precip	Real	Precipitation.
radflx	Real	Net solar radiation.
swflx	Real	Net shortwave radiation.
rmu	Real	Weights for sidewall boundary conditions relax.
betard	Real	Red extinction coefficient.
betabl	Real	Blue extinction coefficient.
redfac	Red	Fraction of penetrating red light.
Common/kppr		
zgrid	Real	Grid levels in centimeters.
vcty	Real	Vertical viscosity coefficient.
difs	Real	Vertical scalar diffusivity.
dift	Real	Vertical temperature diffusivity.
ghats	Real	Nonlocal transport.
vonk	Real	Von karman constant
zmin, zmax	Real	Zehat limits for table.
umin, umax	Real	Ustar limits for table.
epsilon	Real	Vertical coordinate scale factor.
cmonob	Real	Constant for calculating monin-obukov length.
rinfty	Real	KPP: value for calculating rshear instability.
difm0	Real	KPP: maximum viscosity due to shear instability.
difs0	Real	KPP: maximum diffusivity due to shear instability.
difmiw	Real	KPP: background/internal wave viscosity (m^2/s).
difsiw	Real	KPP: background/internal wave diffusivity (m^2/s).
dsfmax	Real	KPP: salt fingering diffusivity factor (m^2/s).
rrho0	Real	KPP: salt fingering rp = (alpha*delT)/(beta*delS).
ricr	Real	KPP: critical bulk Richardson number.
cs	Real	KPP: value for nonlocal flux term.
cstar	Real	KPP: value for nonlocal flux term.

Variable	**Type**	**Description**
Common/kppr cont'd		
cv	Real	KPP: Value for turbulent shear contribution to bulk Richardson number.
c11	Real	KPP: Value for turbulent velocity scale.
deltaz	Real	Delta zehat in table.
deltau	Real	Delta ustar in table.
vtc	Real	Constant for estimating background shear in rib calculation.
cg	Real	Constant for estimating nonlocal flux term of diff. Equations.
dp0enh	Real	Distance for tapering diff. Enhancement at interface nbl-1.
Common/kppi		
niter	Integer	KPP: iterations for semi-implicit solution. (2 recommended).
Common/varblsr		
time	Real	Model time (days).
delt1	Real	Timestep (seconds).
dlt	Real	Delta T.
w0,w1,w2,w3	Real	Weights for atmospheric forcing time interpolation.
wr0,wr1,wr2,wr3	Real	Weights for relaxation climatology time interpolation.
Common/varblsd		
area	Real	Basing area (m^2).
avgbot	Real	Mean basin depth (m).
watcum	Real	Cumulative heat flux.
empcum	Real	Cumulative salt flux.
Common/varblsi		
nstep	Integer	Model time step number.
nstep1	Integer	First time step of this integration.
nstep2	Integer	Last time step of this integration.
lstep	Integer	Number of barotropic time steps per baroclinic time step.
l0,l1,l2	Integer	Atmospheric forcing sample index.
lr0, lr1, lr2	Integer	Relaxation climatology sample index.

Variable	Type	Description
Common/parms1r		
sigma (kdm)	Real	Layer target densities.
theta (kdm)	Real	Layer target densities (sigma) minus reference density (thbase).
thbase	Real	Reference density.
saln0	Real	Initial salinity value.
baclin	Real	Baroclinic time step.
batrop	Real	Barotropic time step.
veldff	Real	Diffusion velocity (m/s) for momentum dissipation.
temdff	Real	Diffusion velocity (m/s) for temperature/salinity mixing.
thkdff	Real	Diffusion velocity (m/s) for thickness diffusion.
viscos	Real	Nondimensional, used in deformation-dependent viscosity
biharm	Real	Fraction of diffusion that is biharmonic (0.0 to 1.0).
vertmx	Real	Diffusion velocity (m/s) for mom.mixing across mixed layer base.
diapyc	Real	KT: diapycnal diffusivity x buoyancy frequency.
dtrate	Real	KT: maximum permitted m.l. detrainment rate (m/day).
h1	Real	Depth interval used in lateral weighting of horizontal pressure gradient.
slip	Real	+ 1 for free-slip, - 1 for non- slip boundary conditions.
cb	Real	Coefficient of quadratic bottom friction.
cbar	Real	Rms flow speed (m/s) for linear bottom friction law.
dsurfq	Real	Number of days between model diagnostics at the surface.
diagfq	Real	Number of days between model diagnostics.
rstrfq	Real	Number of days between model restart output.
wuv1, wuv2	Real	Weights for time smoothing of u, v field.
wts1, wts2	Real	Weights for time smoothing of t, s field.
wbaro	Real	Weight for time smoothing of barotropic u, v, p field.
thkmin	Real	Minimum mixed-layer thickness.
thkbot	Real	Thickness of bottom boundary layer.
sigjmp	Real	Minimum density jump across interfaces (theta units).
tmljmp	Real	Equivalent temperature jump across the mixed layer (deg C).
salmin (kdm)	Real	Minimum salinity allowed in an isopycnic layer.
dp00	Real	Z-level spacing minimum thickness.
dp00f	Real	Z-level spacing stretching factor.
dp00x	Real	Z-level spacing maximum thickness.
dp00s	Real	Sigma spacing minimum thickness.

Variable	Type	Description
Common/parms1i		
mixfrq	Integer	KT: number of time steps between diapycnal mixing calculations.
nhybrd	Integer	Number of hybrid levels (0 = all isopycnal).
nsigma	Integer	Number of sigma levels (nhybrd - nsigma z-levels).
hybflg	Integer	Hybrid generator flag.
advflg	Integer	Scalar advection flag.
ntracr	Integer	Number of time steps between tracer transport.
clmflg	Integer	Climatology frequency flag.
dypflg	Integer	KT: diapycnal mixing flag.
iniflg	Integer	Initial state flag.
lbflag	Integer	Lateral barotropic boundary flag.
mapflg	Integer	Map flag.
yrflag	Integer	Days in year flag.
iversn	Integer	Hycom version number x10.
iexpt	Integer	Experiment number x10.
jerlv0	Integer	Initial jerlov water type (1 to 5).
iceflg	Integer	Ice model flag.
wndflg	Integer	Wind stress input flag.
flxflg	Integer	Thermal forcing flag.
Common/consts		
tenm, onem, tencm, onecm, onemm	Real	Pressure thickness values corresponding to 10m, 1m…
g	Real	Gravity acceleration.
thref	Real	Reference value of specific volume.
spcifh	Real	Specific heat of sea water.
epsil	Real	Small nonzero number used to prevent division by zero.
huge	Real	Large number used to indicate land points.
radian	Real	
pi	Real	
Common/testpt		
[ij]test	Integer	Local grid point where detailed diagnostics are desired.
[ij]ttest	Integer	Global grid point where detailed diagnostics are desired.
Common/pivot		
grido	Real	Mesh size of latitude/longitude grid in degrees.
ypivn	Real	The j-index of the equator.
gridn	Real	Mesh size of actual model grid in degrees longitude (xpivo, ypivo, xpivn not used).

Variable	Type	Description
Common/iovars		
flnmdep, flnmrsi, flnmrso, flnmflx, flnmarc, flnmovr, flnmfor, flnmforw	Character	Filenames.

1.1.3 Assignments

All the parameters are initialized by a *block data* sub-program whose statements are in **blkdat.f**.

Unlike MICOM, HYCOM 2.0.01 uses the MKS system of units.

1.2 Initializations

The field variables are initialized by two successive steps :

1. Calling the *subroutine* **inicon.f** whose main function is to set all initial values to zero. During this step, the Montgomery potential (*cf.* § 4.1) and the potential vorticity of the model ocean at rest are also calculated.

2. Reading of data from the preceding run.

1.3 Running HYCOM

1.3.1 Makefile

The make process is automated by the script **Make.com**, which should be used instead of directly invoking the make command. The makefile sources are found in the **/config** directory. The configuration files are $(ARCH)_$(TYPE), where ARCH defines the machine architecture to target and TYPE is the parallelization strategy and precision. The following configuration files are currently available for HYCOM 2.0.01:
The script **Make.com** should be edited by the user to define $(ARCH) appropriately for the machine. The following list of environment variables must be defined in each configuration file:

`FC`	Fortran 90 compiler.
`FCFFLAGS`	Fortran 90 compilation flags.
`CC`	C compiler.
`CCFLAGS`	C compilation flags.

E10K_one4	- Sun E10000, single processor {,real*4}
E10K_omp	- Sun E10000, OpenMP
alpha_one4	- Compaq Alpha, single processor {,real*4}
alpha_omp	- Compaq Alpha, OpenMP
o2k_one4	- SGI Origin 2000, single processor {,real*4}
o2k_omp	- SGI Origin 2000, OpenMP
sp3_one4	- IBM SMP Power3, single processor {,real*4}
sp3_omp	- IBM SMP Power3, OpenMP
sp3_q64omp	- IBM SMP Power3, OpenMP (64-bit memory model)
sp3_ompi	- IBM SMP Power3, OpenMP and MPI
sp3_mpi	- IBM SMP Power3, MPI

sunU2_one4	- Sun Ultra2, single processor {,real*4}
t3e_one	- Cray T3E, single processor
t3e_mpi	- Cray T3E, MPI
t3e_shmem	- Cray T3E, SHMEM

`CPP`	cpp preprocessor (may be implied by FC).
`CPPFLAGS`	cpp -D macro flags.
`LD`	Loader.
`LDFLAGS`	Loader flags.
`EXTRALIBS`	Extra local libraries (if any).

In addition, rules are required for .c.o, .f.o, and .F.o.

Once **Make.com** has been edited, the executable is created by the command:

./Make.com >& Make.log

1.3.2 Output for testing purposes

The coordinates of the point where detailed results are printed are specified in `common/testpt/` :

```
[***** ADD SOURCE CODE HERE *****]
```

Version 2.0.01 of HYCOM does one type of test :

1. Estimation of the meridional overturning rate. This is done in the subroutine **overtn.f**, where the zonally averaged meridional heat flux is calculated.

1.4 Configuring HYCOM

1.4.1 Implementation

The geometry of the application domain is stored in the file **dimensions.h**. The horizontal extent is defined by the two parameters `idm` and `jdm`. The specification of islands and continents is done with the help of the parameter `ms`. It sets the maximum number of interruptions of the oceanic domain in rows or in columns plus one. The equivalent in the diagonal direction is represented by the parameter `msd`. For each of the four points of the C grid $(u, v, \Delta p, Q)$, `common/gindex/` contains seven tables of entries :

1. One table of dimension `idm*jdm` : `iu(idm,jdm)`. At submerged points, `iu(i,j)=1`; else, `iu(i,j)=0`.
2. One table comprising the number of submerged segments in the row i : `isu(jdm)`;
3. A table giving the lower limits of each segment by column : `ifu(jdm,ms)` ;
4. A table giving the upper limits of each segment by column : `ilu(jdm,ms)` ;
5. A table comprising the number of segments of the column j : `jsu(idm)` ;
6. A table of lower limits of each segment by row : `jfu(idm,ms)` ;
7. A table of upper limits of each segment by row : `jlu(idm,ms)`.

A segment is defined as a number of contiguous submerged points. To represent the separation between land and sea along the diagonals, `common/diags/` contains 5 tables whose values come from numerical processing of the bathymetry file (*cf.* § 1.4.3):

1. A table of the number of segments in each diagonal : `nsec(idm+jdm)`;
2. A table of the abscissas of the lower limits of each segment : `ifd(idm+jdm,msd)` ;
3. A table of the abscissas of the upper limits of each segment : `ild(idm+jdm,msd)` ;
4. A table of the ordinates of the lower limits of each segment : `jfd(idm+jdm,msd)` ;
5. A table of the ordinates of the upper limits of each segment : `jld(idm+jdm,msd)`.

This information is only used in diagnosing the barotropic stream function.

1.4.2 Projections

The default (mapflg = 0) is a conventional Mercator projection with square grid cells. The array orientation is also conventional with the first array dimension, i, West to East and the second, j, South to North. The grid location is defined by the longitude (reflon) of one pressure grid point (pntlon), the latitudinal grid point (pntlat) on the equator, and the longitudinal grid size (grdlon) which is identical to the latitudinal grid size at the equator (grdlat).

1.4.3 Bathymetry

The distribution of variables in a C grid is such that only `idm-1*jdm-1` depths are required to represent the bathymetry of the 'rectangular' portion of the ocean considered. (*cf.* § 15). By convention, land zones are given a depth of zero. The distinction between exposed and submerged zones is carried out by the call :

```
[***** ADD SOURCE CODE HERE *****]
```

The boundaries of the segments along diagonals are needed by the optional Poisson solver (*cf.* § 1.3.2).) To integrate the system of equations over the whole domain, the boundaries of segments by rows and by columns have to be determined for the four points which bound the mesh (*cf.* § 15). To do this, in **bigrid.f**, we have the statements :

```
[***** ADD SOURCE CODE HERE *****]
```

2 Continuity equation : cnuity.f

2.1 Formalism and numerical techniques

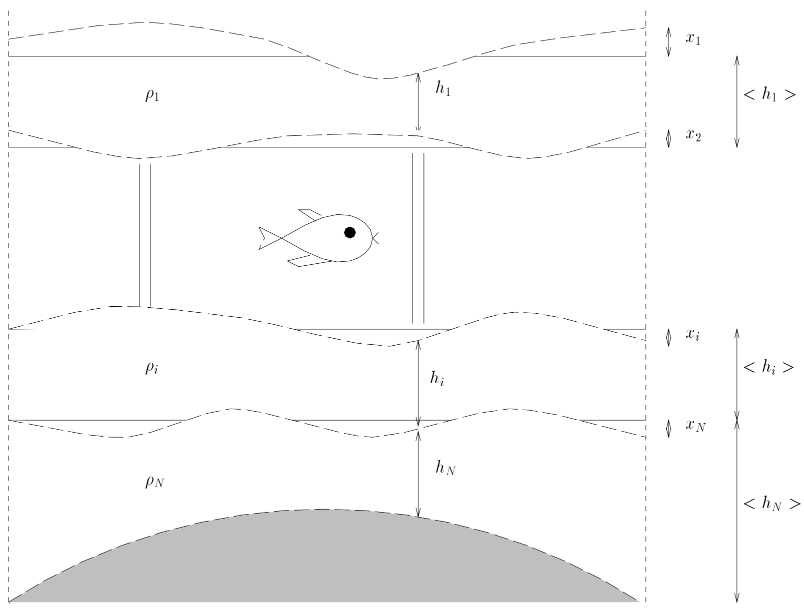


Figure 1: *Vertical discretization of the multilayer ocean*

Bleck & Smith (1990) assume that the difference in pressure between the two interfaces of the k^{th} layer has the form :

$$\Delta p_k = (1+\eta)\Delta p'_k \tag{1}$$

with $\Delta p_k = g\rho_k h_k$ and $\Delta p'_k = g\rho_k h'_k$. Elsewhere, the decomposition is introduced as :

$$\mathbf{u}_k = \overline{\mathbf{u}} + \mathbf{u}'_k \tag{2}$$

with :

$$\overline{\mathbf{u}} = \frac{\sum_{k=1}^{N} \rho_k h_k \mathbf{u}_k}{\sum_{k=1}^{N} \rho_k h_k} \tag{3}$$

and :

$$\overline{\mathbf{u}'_k} = 0 \tag{4}$$

The tendency equation for the component $\Delta p'$ entering in the total expression for the change of pressure in the layer k is then written (Bleck & Smith, 1990) :

$$\frac{\partial}{\partial t}\Delta p'_k + \nabla\cdot(\mathbf{u}\Delta p')_k = \frac{\Delta p'_k}{p'_b}\nabla\cdot(\overline{\mathbf{u}}p'_b) \tag{5}$$

Under the hydrostatic hypothesis, with a free surface, the equation is written :

$$p'_b = \sum_{k=1}^{N}\Delta p'_k = g\sum_{k=1}^{N}\rho_k h'_k = g\rho_r H \tag{6}$$

with H the water depth :

$$H(x,y) = \sum_{k=1}^{N}\langle h_k\rangle = \sum_{k=1}^{N} h_k - \xi_1 \tag{7}$$

$\langle h_k\rangle$ represents the initial thickness of the layer k and ξ_1 represents the change in the free surface (*c.f.* figure 1). ρ_r is the column mean density.

2.1.1 FCT (Flux-Corrected Transport) scheme

The base FCT scheme from the work of Zalesak (1986) is outlined here in 7 steps :

1. The first inference of the variable $\Delta p'_k$ is made by introducing a classic upstream scheme. The variable is denoted by $\Delta p'^{up}_{i,j,k}$ in the layer k. The diffusive fluxes of this first step are calculated from the total velocity field $\mathbf{u}$. Consider the problem (5) without its second term in a one-dimensional form and take $\mathbf{P'}_k = (\mathbf{u}\Delta p')_k$. Then write :

$$P'^{up}_{i+1/2,k} = (u\Delta p')^{up}_{i+1/2,k} = \begin{cases} u^{mid}_{i+1/2,k}\ \Delta p'^{old}_{i,k} & \text{if } u_{i+1/2,k} > 0 \\ u^{mid}_{i+1/2,k}\ \Delta p'^{old}_{i+1,k} & \text{if } u_{i+1/2,k} < 0 \end{cases} \tag{8}$$

   where *mid* and *old* refer to two successive instants in the leapfrog scheme. In a donor-cell scheme, the upstream fluxes as well as the velocity should be defined at the interfaces between the cells.

2. Always in using the total velocity field, proceed next to calculate the non-diffusive flux by a scheme second-order in space and centered in time :

$$P'^{*}_{i+1/2,k} = (u\Delta p')^{*}_{i+1/2,k} = u^{mid}_{i+1/2,k}\ \frac{\Delta p'^{mid}_{i,k} + \Delta p'^{mid}_{i+1,k}}{2}. \tag{9}$$

3. The flux of anti-diffusion **A** is introduced such that : $A_{i+1/2,k} = P'^{*}_{i+1/2,k} - P'^{up}_{i+1/2,k}$. To assure the stability of the scheme (*i.e.* to counter the appearance of negative values of $\Delta p'^{new}_{i,k}$), substitute the anti-diffusive flux **A** with the corrected flux $\mathbf{A}^c$ such that : $A^c_{i+1/2,k} = C_{i+1/2,k} A_{i+1/2,k}$ and with : $0 \leq C \leq 1$. For $C = 0$, we restore the flux of order 1 and $C = 1$ gives back the flux of order 2. The final solution is expressed by a combination of fluxes of orders one and two which moderates the perturbations of stability which appear. For each layer (Baraille & Filatoff, 1995),

$$C_{i+1/2,k} = \begin{cases} \min\left(R^+_{i+1,k}; R^-_{i,k}\right) & \text{if } A_{i+1/2,k} \geq 0 \\ \\ \min\left(R^+_{i,k}; R^-_{i+1,k}\right) & \text{if } A_{i+1/2,k} < 0 \end{cases} \tag{10}$$

where the following factors are successively introduced :

$$\mathbf{I} \begin{cases} P^+_{i,k} = \max\Big(0, A_{i-1/2,k}\Big) - \min\Big(0, A_{i+1/2,k}\Big) \\ \\ P^-_{i,k} = \max\Big(0, A_{i+1/2,k}\Big) - \min\Big(0, A_{i-1/2,k}\Big) \end{cases} \tag{11}$$

then :

$$\mathbf{II} \begin{cases} Q^+_{i,k} = \Delta p'^{max}_{i,k} - \Delta p'_{i,k} \\ \\ Q^-_{i,k} = \Delta p'_{i,k} - \Delta p'^{min}_{i,k} \end{cases} \tag{12}$$

and

$$\mathbf{III} \begin{cases} R^+_{i,k} = \begin{cases} \min\left(1, \frac{\Delta x}{\Delta t}\frac{Q^+_{i,k}}{P^+_{i,k}}\right) & \text{if } P^+_{i,k} > 0 \\ \\ 0 \text{ if } P^+_{i,k} = 0 \end{cases} \\ \\ R^-_{i,k} = \begin{cases} \min\left(1, \frac{\Delta x}{\Delta t}\frac{Q^-_{i,k}}{P^-_{i,k}}\right) & \text{if } P^-_{i,k} > 0 \\ \\ 0 \text{ if } P^-_{i,k} = 0 \end{cases} \end{cases} \tag{13}$$

$R^+_{i,k}$ and $R^-_{i,k}$ represent the biggest multiplicative factors of anti-diffusive flux which assure respectively : $\Delta p'^{n+1}_{i,k} \leq \Delta p'^{max}_{i,k}$ and $\Delta p'^{n+1}_{i,k} \geq \Delta p'^{min}_{i,k}$ with :

$$\begin{cases} \Delta p'^{max}_{i,k} = \max(\Delta p'^{n}_{i-1,k}, \Delta p'^{n}_{i,k}, \Delta p'^{n}_{i+1,k}) \\ \\ \Delta p'^{min}_{i,k} = \min(\Delta p'^{n}_{i-1,k}, \Delta p'^{n}_{i,k}, \Delta p'^{n}_{i+1,k}) \end{cases} \tag{14}$$

The formulas corresponding to the bidimensional case are given in Baraille & Filatoff (1995).

4. In summing the form (5) without the second term over the N layers of the vertical discretization, when $\partial p'_b/\partial t = 0$:

$$\mathbf{P}' = \sum_{k=1}^{N} \nabla \cdot \mathbf{P}'_k = 0 \tag{15}$$

It is clear that to satisfy this condition, a second order approximation of the flux $\mathbf{P}'$ is better than any lower order approximation. The following identity is then written :

$$\sum_{k=1}^{N} \left(P'^{*}_{i+1/2,k} - P'^{*}_{i-1/2,k} \right) = 0 \tag{16}$$

The sum of anti-diffusive flux corrections over the vertical introduces a bias in the conservation of p'_b which must be compensated. To do this, calculate the vertical sum of flux corrections (by means of a second order approximation) :

$$\mathcal{A}_{i+1/2} = \sum_{k=1}^{N} \left(1 - C_{i+1/2,k} \right) \mathbf{A}_{i+1/2,k} \tag{17}$$

5. To evaluate the effect of the anti-diffusive flux "integral", calculate by layer the new thickness :

$$\Delta p'^{\dagger}_{i,k} = \Delta p'^{up}_{i,k} - \Delta t \ \left(\nabla \cdot \mathbf{A}^c \right)_{i,k} \tag{18}$$

From the new thickness, the bottom pressure is then obtained :

$$p'^{\dagger}_{b_i} = \sum_{k=1}^{N} \Delta p'^{\dagger}_k \tag{19}$$

6. Unless $C_{i+1/2,k} = 1$ for each layer, $p'^{\dagger}_{b_i} \neq p'_{b_i}$. To remedy this problem, a second correction B is made to the upstream flux such that :

$$B_{i+1/2,k} = \frac{\Delta p'^{\dagger}_{i,k}}{p'^{\dagger}_{b_i}} \mathcal{A}_{i+1/2} \tag{20}$$

The final flux takes the form :

$$P'^{fin}_{i+1/2,k} = P'^{up}_{i+1/2,k} + A^c_{i+1/2,k} + B_{i+1/2,k} \tag{21}$$

and the preceding estimation $\Delta p'^{\dagger}_{i,k}$ is rectified by the relation :

$$\Delta p'^{\ddagger}_{i,k} = \Delta p'^{\dagger,k}_{i,k} - \Delta t \ \left(\nabla \cdot \mathbf{B} \right)_{i,k} \tag{22}$$

where the bottom pressure is set to :

$$p'^{\ddagger}_{b_i} = \sum_{k=1}^{N} \Delta p'^{\ddagger}_{i,k} \tag{23}$$

7. The last step is to account for the right hand side of the original equation (5). Once at this stage, the claim is made that the flux was corrected to best satisfy $\partial p'_b/\partial t = 0 \quad \forall i$. From this claim, the following statement can be written for each layer k :

$$\frac{\partial}{\partial t}\Delta p'^{fin}_k + \nabla \cdot \mathbf{P}'^{fin}_k = \frac{\Delta p'^{fin}_k}{p'_b}\nabla \cdot (\overline{\mathbf{u}} p'_b) \tag{24}$$

in order to eventually satisfy :

$$\sum_{k=1}^{N} \Delta p'^{fin}_k = p'_b. \tag{25}$$

For all points, the form (24) is summed over the vertical giving that :

$$\nabla \cdot \left(\sum_{k=1}^{N} \mathbf{P}'^{fin}_k \right) = \nabla \cdot (\overline{\mathbf{u}} p'_b). \tag{26}$$

In the divergence term, the vertical flux average and the flux sum are in equilibrium. On the other hand, the flux and therefore the layer thickness may need adjustment. Accounting for the second term of (5) in the calculation of the final thickness $\Delta p'^{fin}_k$ gives the simple equation :

$$\Delta p'^{fin}_k = \frac{\Delta p'^{\ddagger}_k}{p'^{\ddagger}_b} p'_b. \tag{27}$$

From this :

$$p'^{fin}_{k+1} = \sum_{l=1}^{k} \Delta p'^{fin}_l \quad \text{for} \quad k = 1, \ldots, N \tag{28}$$

with therefore : $p'^{fin}_{N+1} = p'^{fin}_b \equiv p'_b$. Then in the summations (6), (19), (23) and (28), the surface pressure is assumed to be z1ero ($p'_1 = 0$).

2.1.2 Interface diffusion

In the shallow-water multilayer model, the conservation of mass is cast in the form (Baraille & Filatoff, 1995) :

$$\frac{\partial}{\partial t}\Delta p_k + \nabla \cdot (\mathbf{u}\Delta p)_k = 0 \tag{29}$$

Once the decomposition (1) is introduced in the formula, one sees that formally equation (5) comes from the approximation: $(1 + \eta) \approx 1$. Notice that nevertheless the use of this approximation does not disturb in any case the property: $\partial p'_b/\partial t = 0$ (Baraille & Filatoff, 1995). Thus, the sum of the changes in $\Delta p'$ over the vertical in this approximation is such that at each point, p'_b stays constant. In practice, the accounting of η is such that :

$$1 + \eta = \frac{1}{\rho_r H} \sum_{k=1}^{N} \rho_k h_k \tag{30}$$

so as to restore the introduction of a diffusion term $\nabla \cdot (\nu \nabla \Delta p')$ in the conservation equation (5). As in finite differences, the product $\nu \partial(\Delta p)/\partial x$ is numerically equivalent to $u_d \delta p$ (where δp represents the growth of the thickness of a layer between two adjacent meshes), Bleck *et al.* (1992) introduce a "diffusion velocity" $u_d \equiv \nu/\Delta x$ (where Δx is the size of the mesh) to simulate the isopycnic diffusion. Typically $u_d = 0.5\ cm/s$ for the variable Δp.

From the preceding FCT scheme, $p'^{fin}_{i,k}$ and $p'^{fin}_{i-1,k}$, the pressures at the k^{th} density interface in two adjacent points with coordinates x_i and x_{i-1}, are obtained. In these two points, the bottom pressures are respectively p'_{b_i} and $p'_{b_{i-1}}$. This statement is made concrete by the form :

$$D_{i-1/2,k} = \min \left\{ p'_{b_i} - p'_{i,k} \max \left[p'_{i-1,k} - p'_{b_{i-1}}, \frac{u_d \Delta t}{\Delta x} \left(p'_{i-1,k} - p'_{i,k} \right) \right] \right\}. \tag{31}$$

During a time interval Δt, the variation of pressure at the interface k comes from the expression :

$$\frac{\partial p'_{i,k}}{\partial t} + \frac{\Delta x}{\Delta t} \ (\nabla \cdot \mathbf{D})_{i,k} = 0 \tag{32}$$

This last equation is then written for the interfaces $k = 2, \ldots, N$:

$$p'^{n+1}_{i,k} = p'^{fin}_{i,k} - \left(D_{i+1/2,k} - D_{i-1/2,k} \right) \tag{33}$$

in which :

$$\Delta p'^{n+1}_{i,k} = p'^{n+1}_{i,k+1} - p'^{n+1}_{i,k} \tag{34}$$

To remain coherent with the previous step, the flux retained at the two interfaces of the layer are written :

$$\begin{cases} \mathbf{F}^{fin}_{i+1/2,k-1} & = \mathbf{P}'^{fin}_{i+1/2,k-1} + \frac{1}{\Delta t} \mathbf{D}_{i+1/2,k} \\ \\ \mathbf{F}^{fin}_{i+1/2,k} & = \mathbf{P}'^{fin}_{i+1/2,k} - \frac{1}{\Delta t} \mathbf{D}_{i+1/2,k} \end{cases}$$

2.2 Usage

In HYCOM 2.0.01, the numerical calculation of the baroclinic continuity equation is performed by the subprogram :

```
subroutine cnuity(m,n)
```

2.2.1 Order of operations

The algorithm is based on a consecutive treatment of isopycnic layers. First, the variables necessary to calculate the flux integrals over the vertical coordinate are initialized :

```
[***** ADD SOURCE CODE HERE *****]
```

The first calculational step consists of computing the two components of upstream flux in applying (8) :

```
[***** ADD SOURCE CODE HERE *****]
```

Then, the diffusive solution is determined :

```
[***** ADD SOURCE CODE HERE *****]
```

Following that, the second order flux is added and the anti-diffusive flux is calculated by simply taking the difference between the upstream flux and the second order flux. The second order fluxes are stored in the arrays `uflux(i,j)` and `vflux(i,j)`. Meanwhile, upstream fluxes are tracked in the arrays `uflx(i,j)` and `vflx(i,j)`. Note that the centering in space of the pressure term of equation (9) has been performed in the preceding calculation of the barotropic component of the current (subroutine `barotp`). The numerical values of the thicknesses at the calculational points of the two horizontal components of velocity are found in the global variables `dpu(i,j,k)` and `dpv(i,j,k)`.

```
[***** ADD SOURCE CODE HERE *****]
```

The third step decomposes into two parts :

1. Determination of the local extrema of the variable Δp. The results are saved in the arrays `util1(i,j)` and `util2(i,j)` ;
2. Computation of flux correctors following the method outlined above. The results are also stored in the arrays `util1(i,j)` and `util2(i,j)`

```
[***** ADD SOURCE CODE HERE *****]
```

In a fourth phase, the flux correction given by (17) is determined. The results are stored in variables `utotn(i,j)` and `vtotn(i,j)`.

```
[***** ADD SOURCE CODE HERE *****]
```

Then proceed to the inference of a new thickness given by (18).

```
[***** ADD SOURCE CODE HERE *****]
```

From there, it is then possible to determine the second correction to the flux expressed by (20), and to perform the estimation (22) of the thickness of the layer considered. The final values of the fluxes are placed in the arrays `uflx(i,j)` and `vflx(i,j)`.

```
[***** ADD SOURCE CODE HERE *****]
```

For the last step, the adjustment (27) is made.

```
[***** ADD SOURCE CODE HERE *****]
```

The treatment of the diffusion term consists of two principal steps :

1. To start, calculate the interfacial diffusive flux given by the form (31), and then correct the new flux with the aid of the expressions given in (35). The results are stored in the arrays `uflx(i,j)` and `vflx(i,j)`.

```
[***** ADD SOURCE CODE HERE *****]
```

2.2.2 Flowchart

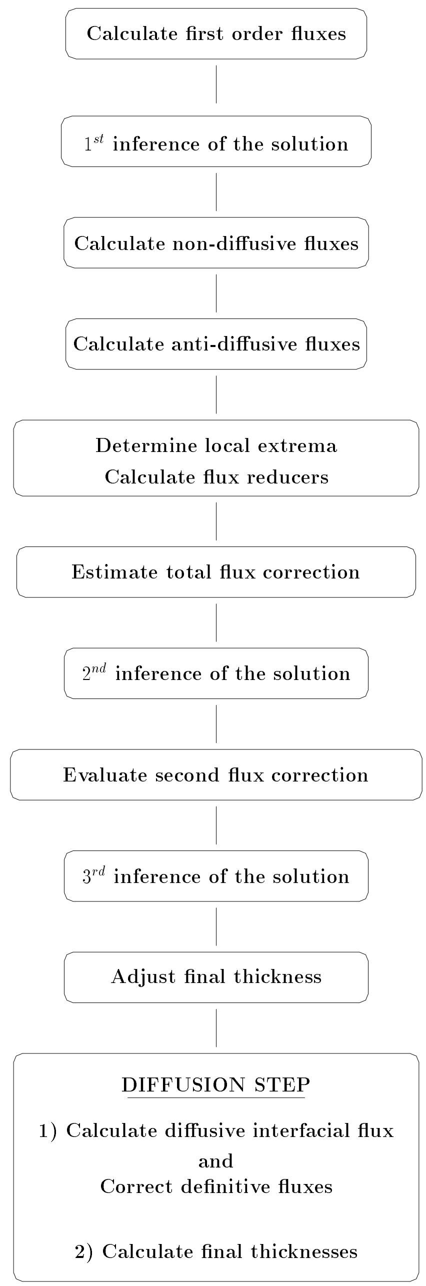


Figure 2: *Order of the treatment of the continuity equation for the baroclinic mode in HYCOM 2.0.01*

2.3 Variables

2.3.1 Identification

Notation in the theory	Notation in **cnuity.f**
$C_{i+1/2}$	`clip`
$p'_{b_i} - p'_{i,k}$	`flxhi`
$p'_{i-1,k} - p'_{b_{i-1}}$	`flxlo`
u, v	`utotm(i,j), vtotm(i,j)`
$\overline{u}, \overline{v}$	`ubavg(i,j,n), vbavg(i,j,n)`
u', v'	`u(i,j,n), v(i,j,n)`
$\left(\Delta p_i'^{mid} + \Delta p_{i+1}'^{mid}\right)/2$	`dpu(i,j,k), dpv(i,j,k)`

2.3.2 Local variables

Subroutine cnuity

`clip`	Correction factor for antidiffusive fluxes.
`dpdn`	Vertical excursion below the mixed layer base.
`dpkmin`	Minimum layer thicknesses.
`dpmin`	Variations in p'_b due to different corrections to the upstream flux.
`dpmn`	Minimum layer k thickness.
`dpup`	Vertical excursion above the mixed layer base.
`dtinv`	1/delt1
`flxhi, flxlo`	Differences in pressure between the interfaces of a layer and the bottom.
`i, ia, ib`	Array indices.
`iflip`	Time step selector.
`j, ja, jb`	Array indices.
`k`	Layer index.
`l`	Loop index.

`lpipe_cnuity`	Flag to compare two model runs.
`mask`	Comparison mask.
`mbdy`	Halo extent.
`pold`	Old pressure.
`q`	Intermediate variable used in the calculation of the fluxes.
`text`*12	Comparison title.

2.4 Procedures

Subroutines `cnuity`

3 Advection-diffusion : tsadvc.f

3.1 Formalism and numerical methods

HYCOM uses the same fundamental algorithms for horizontal advection and diffusion that were used by MICOM. When HYCOM is run with isopycnic vertical coordinates (MICOM mode), horizontal advection/diffusion is performed in the same manner as in MICOM. Temperature and salinity are advected and diffused in layer 1. Only salinity is advected and diffused in deeper layers, with temperature diagnosed from the equation of state to maintain constant density in these layers. When HYCOM is run with hybrid vertical coordinates, the user selects whether temperature and salinity, or just salinity, are advected and diffused within the upper n_{hyb} layers that the user declares to be hybrid layers. This option was included because the effects of cabbeling when both temperature and salinity are advected and diffused can lead to problems with the adjustment of vertical coordinates by the hybrid coordinate algorithm, particularly if the user selects to flux both temperature and salinity across the moving vertical coordinates (See Section 9). When salinity only is advected/diffused, these problems do not appear, but the tradeoff is that temperature is no longer conserved. In low-resolution simulations of Atlantic Ocean climate, the non-conservation of temperature did not have a large influence on simulated fields.

In isopycnic coordinates, the thermal evolution equation takes the form :

$$\frac{\partial}{\partial t}T\Delta p + \underbrace{\nabla\cdot(\mathbf{u}T\Delta p)}_{advec} + \underbrace{\left(\dot{s}\frac{\partial p}{\partial s}T\right)_{bot} - \left(\dot{s}\frac{\partial p}{\partial s}T\right)_{top}}_{dia-diff} = \underbrace{\nabla\cdot(\nu\Delta p\,\nabla T)}_{iso-diff} + \mathcal{H}_T. \qquad (35)$$

Δp is the thickness of layer k of temperature T. The radiative exchanges are represented by the term $\mathcal{H}_T$. The expression $(\dot{s}\partial p/\partial s)$ represents a vertical mass flux.

In HYCOM, the problem of advection of heat and salt is treated by MPDATA coming from the work of Smolarkiewicz & Clark (1986) and Smolarkiewicz & Grabowski (1990). The diapycnic diffusion is accounted for in the diapycnic mixing algorithm described in Section 14. Regarding the isopycnic diffusion, the numerical method used is based on the different procedures commented on in Section 3.1.3.

3.1.1 Maintaining the positivity of thickness

Consider the simple form of the problem of advection of an isopycnic layer of thickness Δp, of temperature T and driven by a horizontal velocity $\mathbf{u}$:

$$\frac{\partial}{\partial t}T\Delta p + \nabla\cdot(\mathbf{u}T\Delta p) = 0 \qquad (36)$$

From the solution of the continuity equation, for each layer, the mass flows $(u\Delta p)_{i,j}^{n+1}$ and $(v\Delta p)_{i,j}^{n+1}$ as well as a diagnostic value for $\Delta p_{i,j}^{n+1}$ are calculated. Let $\mathbf{P} = \mathbf{u}\Delta p$ and let h be the thickness of the layer of interest. Considering the conservation equation :

$$\frac{\partial h}{\partial t} + \nabla \cdot \mathbf{P} = 0 \tag{37}$$

and determining the variation δh follows from this form, we have :

$$\delta h(i,j) = -\frac{\Delta t}{\Delta x}\left[Px_{i+1,j}^{n+1} - Px_{i,j}^{n+1} + Py_{i,j+1}^{n+1} - Py_{i,j}^{n+1}\right] \tag{38}$$

Suppose the layer shrinks with time, *i.e.*, $\delta h < 0$. In this case, $|\delta h| > \Delta p^{n-1}$ brings about the unsatisfactory condition $\Delta p'^{n+1} < 0$. The fluxes resulting from the numerical treatment of the continuity equation are then inconsistent with the variations of thickness calculated in this same step. The mass flux as well as the layer thickness are necessary in the advection calculation. The mass fluxes are made to be consistent with the variation of layer thickness so that always in the case where $\delta h < 0$, then $|\delta h| \leq \Delta p^{n-1}$. Introduce the two utility thicknesses h_1 and h_2 :

$$h_1 = 1/2(\Delta p^{n+1} + \Delta p^{n-1} + \delta h) \;\; ; \;\; h_2 = 1/2(\Delta p^{n+1} + \Delta p^{n-1} - \delta h) \tag{39}$$

In the case where the fluxes are strictly consistent ($i.e.$: $\delta h = \Delta p^{n+1} - \Delta p^{n-1}$) :

$$h_1 = \Delta p^{n-1} \;\; \text{and} \;\; h_2 = \Delta p^{n+1}$$

When the flux is consistent, the utility thicknesses given by (39) are introduced. Otherwise, when the fluxes are inconsistent, the positivity of the layer thicknesses is maintained by setting :

$$h_1 = -\delta h \;\; \text{and} \;\; h_2 = 0.$$

For $\delta h > 0$, the respective values of h_1 and h_2 are obtained with identical reasoning.

3.1.2 Treatment of the tendency term

To assure a gradual transition of Δp towards a null value, the finite difference expression of the tendency term in equation (35) that is generally written in the form :

$$\frac{T^{new}\Delta p^{new} - T^{old}\Delta p^{old}}{\Delta t}$$

is replaced by :

$$\frac{T^{new}(\Delta p^{new} + \epsilon) - T^{old}(\Delta p^{old} + \epsilon)}{\Delta t}. \tag{40}$$

The small parameter ϵ will be non-zero when the loss of mass during a time step averages at least 90% of the previous value. More precisely :

$$\epsilon = A + (\epsilon_1^2 + A^2)^{1/2} \tag{41}$$

where $A = (0.1\Delta p^{old} - \Delta p^{new})/2$. The intermediate parameter ϵ_1 is therefore fixed to the numerical value of 10 cm, introduced to assure the validity of (40) when Δp^{old} and Δp^{new} are tending toward zero.

3.1.3 Treatment of the diffusion term

As in finite differences, the product $\nu \partial T/\partial x$ is numerically equivalent to $u_d \delta T$ (where δT represents the temperature difference between the two adjacent mesh points bracketing u_d). Bleck *et al.* (1992) introduce a diffusion velocity $u_d \equiv \nu/\Delta x$ (where Δx is the size of the mesh) to simulate the isopycnic mixing. Typically, $u_d = 1\ cm/s$ for the variables T and S. On the other hand, in the flux expression appearing in the diffusion term, the variation Δp is replaced by the harmonic average :

$$\widetilde{\Delta p} = \frac{2}{\Delta p_{i-1}^{-1} + \Delta p_i^{-1}} \tag{42}$$

The reasoning for this choice is not obvious. It can be found in Bleck *et al.* (1992). One can also understand it in the following way: Consider two neighboring mesh points to which are associated the two values Δp_{i-1} and Δp_i and the two temperatures T_{i-1} and T_i. In order for the introduction of the factor Δp in the flow expression not to have any effect, substitute a neutral value $\widetilde{\Delta p}$ that characterizes a neutral state. In this neutral context, two neighboring mesh points have the same amount of heat Q and an average temperature $\widetilde{T}$. If the turbulence and therefore the turbulent diffusion are interpreted as a mixing process, then : $\widetilde{T} = (T_{i-1} + T_i)/2$. Of course the neutral state does not have the same thermal content Q. Therefore :

$$\widetilde{T} = \frac{Q}{\widetilde{\Delta p}} = \frac{1}{2}\left(\frac{Q}{\Delta p_{i-1}} + \frac{Q}{\Delta p_i}\right) \tag{43}$$

an identity which led to the form (42). More over, as the inference of the new value $T^n_{i,j}$ requires a division by Δp, to treat the situations $\Delta p \to 0$, a residual thickness given the fixed numerical value of 1 mm is introduced.

3.1.4 Filtering

In the classical manner, to compensate for dispersion problems caused by the leapfrog scheme, Asselin filtering is used :

$$\widehat{T}^n \widehat{\Delta p}^n = \left[T^n(1-2\gamma)\Delta p^n + \gamma\left(\widehat{T}^{n-1}\widehat{\Delta p}^{n-1} + T^{n+1}\Delta p^{n+1}\right)\right] \tag{44}$$

So that this form remains valid when $\Delta p \to 0$, a residual thickness ϵ is introduced such that :

$$\widehat{T}^n = \left(\widehat{\Delta p}^n + \epsilon\right)^{-1}\left[T^n\left\{(1-2\gamma)\Delta p^n + \epsilon\right\} + \gamma\left(\widehat{T}^{n-1}\widehat{\Delta p}^{n-1} + T^{n+1}\Delta p^{n+1}\right)\right] \tag{45}$$

In application, ϵ takes a numerical value equivalent to 10^{-3} m. Concerning the thermodynamic variables, the value $\gamma = 0.015625$ is used. Moreover, in the code, the filtering is carried out in two steps. All things considered :

$$\widehat{[T\delta p]}^n = T^n \left\{(1-2\gamma)\Delta p^n + \epsilon\right\} + \gamma \widehat{T}^{n-1}\widehat{\Delta p}^{n-1} \tag{46}$$

Then, after having calculated T^{n+1} and filtering the layer thickness Δp by the formula :

$$\widehat{\Delta p}^n = (1-2\gamma)\Delta p^n + \gamma\left(\widehat{\Delta p}^{n-1} + \Delta p^{n+1}\right) \tag{47}$$

the second step is written :

$$\widehat{T}^n = \left(\widehat{\Delta p}^n + \epsilon\right)^{-1}\left\{\widehat{[T\delta p]}^n + \gamma T^{n+1}\Delta p^{n+1}\right\} \tag{48}$$

3.2 Usage

In the code of HYCOM 2.0.01, the numerical calculation of the horizontal equations of advection-diffusion of heat and salt is realized by the subroutine :

`subroutine tsadvc` (m,n)

Concerning the mixed layer, the variables treated are the specific volume indicated by the variable `thmix(i,j,n)` and the salinity `saln(i,j,1)`.

3.2.1 Order of operations

Concerning the mixed layer, the components Px and Py of mass flux are smoothed over three points. The results are put in the arrays `uflux(i,j)` and `vflux(i,j)`. Following this, for each computational point, the first phase of the filtering is performed following the formula (46) and the procedure verifying the coherence of the mass fluxes with the variation of thickness of the surface layer is started. The variables h_1 and h_2 correspond to the arrays `util1(i,j)` and `util2(i,j)`.

```
[***** ADD SOURCE CODE HERE *****]
```

Then, the advection of the mixed layer characteristics is realized by the instructions :

```
[***** ADD SOURCE CODE HERE *****]
```

After having filtered the layer thicknesses by the formula (47), the double step of filtering given by the relation (48) is executed. The negative characteristics are then ready to be diagnosed.

```
[***** ADD SOURCE CODE HERE *****]
```

It remains to calculate the neutral thickness for the mesh by accounting for the effect of diffusion over the mixed layer characteristics :

```
[***** ADD SOURCE CODE HERE *****]
```

Concerning the isopycnic layers, the order of operations is identical to that described for the mixed layer except for the fact that only the salinity is advected, then diffused. HYCOM 2.0.01 is designed to treat in parallel any tracer problem :

```
[***** ADD SOURCE CODE HERE *****]
```

3.2.2 Flowchart

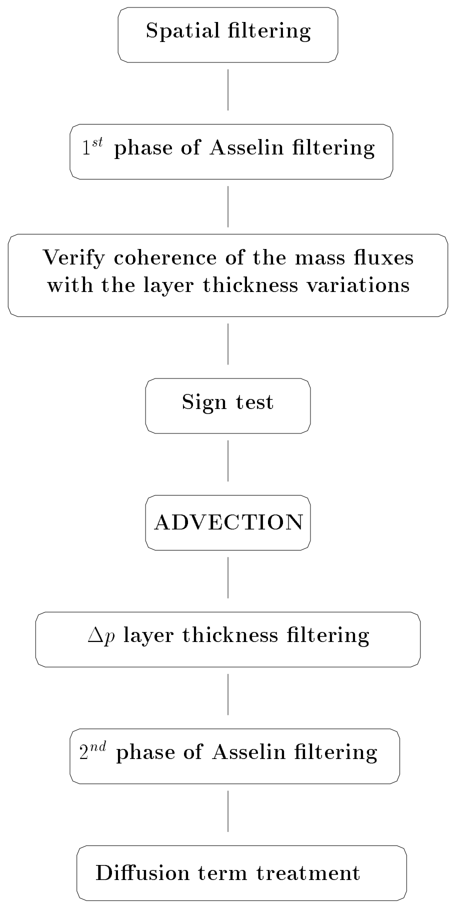


Figure 3: *Order of transport and horizontal diffusion calculations in HYCOM 2.0.01*

3.3 Variables

3.3.1 Identification

Notation in the theory	Notation in **tsadvc.f**
$\widetilde{\Delta p}$	`factor`
δh	`flxdiv`
h_1, h_2	`util1(i,j),util2(i,j)`

$\gamma, (1-2\gamma)$	`wts2,wts1`

3.3.2 Local variables

Subroutine tsadvc

`a, b`	First and argument (respectively) to *harmon*.
`baclin5`	5*baclin.
`factor`	Intermediate variable in the diffusion term calculation.
`flxdiv`	Variation of thickness δh due to the divergence of momentum flux.
`harmon`	Harmonic mean.
`i, j`	Array indices.
`ia, ib, ja, jb`	Loop counters.
`k`	Layer index.
`l`	Loop index.
`latemp`	Advect temperature flag.
`lath3d`	Advect density flag.
`lpipe_tsadvc`	Flag to compare two model runs.
`m, n`	Time step indices.
`mbdy`	Halo extent.
`offset`	Intermediate variable in the calculation of h_1 and h_2.
`pdtemp`	Temperature advection offset.
`pdth3d`	Density advection offset.
`pmid, pnew, pold`	Intermediate pressures.
`posdef`	Reference value of the specific volume in the mixed layer.
`smaxx, sminn`	Intermediate salinities.
`sold`	Old salinity.
`text`	Comparison title.

`tmaxx, tminn`	Intermediate temperatures.
`told`	Old temperature.
`wts2dp`	wts2/layer thickness.
`xmax(2*kdm)`	Maximums.
`xmin(2*kdm)`	Minimums.

3.4 Procedures

Functions	`harmon?????`
Subroutines	`advem`

3.5 The Smolarkiewicz MPDATA

3.5.1 Formalism

The formalism used by Bleck to treat the problem of advection of a non-negative quantity ψ in HYCOM is very similar to that used by Smolarkiewicz in MPDATA (Smolarkiewicz & Clark, 1986 and Smolarkiewicz & Grabowski, 1990). In a bidimensional form, the general equation in conservation form can be written :

$$\frac{\partial f\psi}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} = 0 \tag{49}$$

where $(F, G) = (u\psi, v\psi)$ represent the fluxes of the quantity ψ in the two directions x, y. f is an arbitrary positive function. To preserve the sign of ψ, we approximate the form (49) with a donor-cell scheme. In the first approximation, the upstream flux as well as the components of the velocity are defined at the interfaces between the cells. This flux can be generally expressed :

$$F^{up}_{i+1/2,j} = \frac{u_{i+1/2} + |u_{i+1/2}|}{2}\psi_{i,j} + \frac{u_{i+1/2} - |u_{i+1/2}|}{2}\psi_{i+1,j} \tag{50}$$

which, on introducing the two operators :

$$\overline{\psi}^x = (\psi_{i,j} + \psi_{i+1,j})/2 \quad \text{et} \quad \delta_x\psi = (\psi_{i+1,j} - \psi_{i,j})/\Delta x$$

then condenses to the form :

$$F^{up} = u\overline{\psi}^x - \frac{|u|\Delta x}{2}\delta_x\psi \tag{51}$$

Starting from the standard Taylor expansion :

$$\phi_{i\pm 1/2,j} = \phi_{ij} \pm \frac{\Delta x}{2}\frac{\partial \phi}{\partial x} + \frac{\Delta x^2}{8}\frac{\partial^2 \phi}{\partial x^2} + O(\Delta x^3)$$

where ϕ is a variable such that :

$$\overline{\phi}^x = \phi + O(\Delta x^2)$$

$$\delta_x \phi = \frac{\partial \phi}{\partial x} + O(\Delta x^2).$$

The introduction of the forms in (50) lead to :

$$\delta_x F^{up} = \frac{\partial}{\partial x}\left[u[\psi + O(\Delta x^2)] - \frac{|u|\Delta x}{2}[\frac{\partial \psi}{\partial x} + O(\Delta x^2)]\right] \tag{52}$$

Similarly, expanding in a Taylor series in the time step allows the identity :

$$\frac{(f\psi)^{n+1} - (f\psi)^n}{\Delta t} \equiv \delta_t f\psi = \frac{\partial f\psi}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 f\psi}{\partial t^2} + O(\Delta t^2)$$

Taking into account (49), the second derivative of the last term can be expressed as a function of spatial derivatives :

$$\begin{aligned}
\frac{\partial^2 f\psi}{\partial t^2} &= -\frac{\partial}{\partial t}\left[\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right] \\
&= -\frac{\partial}{\partial x}\left(\frac{u}{f}\frac{\partial f\psi}{\partial t}\right) - \frac{\partial}{\partial y}\left(\frac{v}{f}\frac{\partial f\psi}{\partial t}\right) \\
&= \frac{\partial}{\partial x}\left[\frac{u}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\frac{v}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right].
\end{aligned}$$

which gives :

$$\delta_t f\psi = \frac{\partial f\psi}{\partial t} + \frac{\Delta t}{2}\left\{\frac{\partial}{\partial x}\left[\frac{u}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\frac{v}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right]\right\} + O(\Delta t^2) \tag{53}$$

In combining (52) and (53), the following is obtained :

$$\begin{aligned}
\delta_t f\psi + \delta_x F^{up} + \delta_y G^{up} &= \frac{\partial f\psi}{\partial t} + \frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} \\
&\quad + \frac{\Delta t}{2}\left\{\frac{\partial}{\partial x}\left[\frac{u}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right] + \frac{\partial}{\partial y}\left[\frac{v}{f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)\right]\right\} \\
&\quad - \frac{\partial}{\partial x}\left[\frac{|u|\Delta x}{2}\frac{\partial \psi}{\partial x}\right] - \frac{\partial}{\partial y}\left[\frac{|v|\Delta y}{2}\frac{\partial \psi}{\partial y}\right] + O(\Delta t^2, \Delta x^2, \Delta y^2)
\end{aligned}$$

The essence of the MPDATA scheme is to correct truncation errors of the forward-upstream scheme by introducing the divergence of the anti-diffusive fluxes :

$$\begin{aligned}
F^{anti} &= \frac{|u|\Delta x}{2}\frac{\partial \psi}{\partial x} - \frac{u\Delta t}{2f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right) \\
G^{anti} &= \frac{|v|\Delta y}{2}\frac{\partial \psi}{\partial y} - \frac{v\Delta t}{2f}\left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y}\right)
\end{aligned} \tag{54}$$

and to ameliorate the precision of the initial scheme in resolving (49) with the form :

$$\delta_t f\psi = -(\delta_x F^{up} + \delta_y G^{up}) - (\delta_x F^{anti} + \delta_y G^{anti}). \tag{55}$$

In the numerical calculations, the final solution is evaluated in two steps :

1. Determination of the upstream fluxes F^{up} and G^{up} and approximation of the transported but diffused solution ψ^{up};

2. Establishment of the anti-diffusive fluxes. At this phase, the positivity is assured by submitting the fluxes F^{anti} and G^{anti} to a procedure of flux reduction identical to the one implemented in the algorithm of Flux Corrected Transport (*cf.* § 2.1.1). Since a C grid is used to spatially discretize the different variables, this step requires spatially-centered finite differences when approximating the anti-diffusive fluxes given by (54). So,

$$\begin{aligned} F_c^{up}(i,j) &= \frac{1}{2}u(i,j)\left[\psi^{up}(i-1,j) + \psi^{up}(i,j)\right] \\ G_c^{up}(i,j) &= \frac{1}{2}v(i,j)\left[\psi^{up}(i,j-1) + \psi^{up}(i,j)\right] \end{aligned} \tag{56}$$

then :

$$\begin{aligned} \frac{\partial F}{\partial x} &\equiv \frac{1}{\Delta x}\left[F_c^{up}(i+1,j) + F_c^{up}(i,j)\right] \\ \frac{\partial G}{\partial y} &\equiv \frac{1}{\Delta y}\left[G_c^{up}(i,j+1) + G_c^{up}(i,j)\right] \end{aligned} \tag{57}$$

Putting $Div\mathcal{F} = \partial F/\partial x + \partial G/\partial y$ and setting $f = 1$:

$$\begin{aligned} F_c^{anti} &= \frac{1}{2}|u(i,j)|\left[\psi^{up}(i,j) - \psi^{up}(i-1,j)\right] - \frac{\Delta t}{4}u(i,j)\left[Div\mathcal{F}(i-1,j) + Div\mathcal{F}(i,j)\right] \\ G_c^{anti} &= \frac{1}{2}|v(i,j)|\left[\psi^{up}(i,j) - \psi^{up}(i,j-1)\right] - \frac{\Delta t}{4}v(i,j)\left[Div\mathcal{F}(i,j-1) + Div\mathcal{F}(i,j)\right] \end{aligned} \tag{58}$$

3.5.2 Usage

The usage of the third order numerical scheme for advection of heat and salt, based on the work of Smolarkiewicz & Clark (1986) and Smolarkiewicz & Grabowski (1990), is implemented in the subroutine :

```
[***** ADD SOURCE CODE HERE *****]
```

The transported variable is `fld`. The arguments u and v represent the mass fluxes $u\Delta p$ and $v\Delta p$. Their numerical values come from solution of the continuity equation. The two arguments `fco` and `fc` correspond to the two utility thicknesses h_1 and h_2 of the layer considered, as understood in § 3.1.1. They are needed in the subroutine to be able to apply the form (40) and therefore to determine ψ^{up} from which the finite-difference expressions for the anti-diffusive fluxes are eventually calculated. For `iord = 1`, the numerical scheme returns a simple donor-cell. When the transport variable `fld` is worked out using the quantities $u\Delta p$ and $v\Delta p$ and not using the characteristics u and v of the velocity field, the calculation of a divergence of the anti-diffusive fluxes centered in time implies also storing the numerical values of the fluxes obtained by (58) with the quantities of average thickness :

$$\overline{\overline{\Delta p}} = \left[h_1(i,j,n-1) + h_2(i,j,n-1) + h_1(i,j,n) + h_2(i,j,n)\right]/4 \tag{59}$$

3.5.3 Order of operations

The algorithm includes a preliminary step which consists of determining one part, the local extrema, and another part, the fluxes F^{up} and G^{up} according to the upstream method.

```
[***** ADD SOURCE CODE HERE *****]
```

The following step returns an estimate of the divergence of the flux and enforces monotonicity. In this first phase of inferring a new value of the variable `fld` from the form (40), the parameter ϵ is introduced as defined in § 3.1.2.

```
[***** ADD SOURCE CODE HERE *****]
```

Next comes the calculation of the anti-diffusive fluxes with the evaluation of the fluxes F_c^{up} and G_c^{up} from the expressions given by (56). An estimation of the fluxes F_c^{anti} and G_c^{anti} is generated by (58). The results are stored in the arrays `flx(i,j)` and `fly(i,j)`.

```
[***** ADD SOURCE CODE HERE *****]
```

The numerical values obtained are then subjected to the flux reduction method.

```
[***** ADD SOURCE CODE HERE *****]
```

The last operation consists of calculating the divergence of these fluxes from which the final value of the transported variable is obtained taking into account, again, the expression (40).

[***** ADD SOURCE CODE HERE *****]

Flowchart

3.5.4 Flowchart

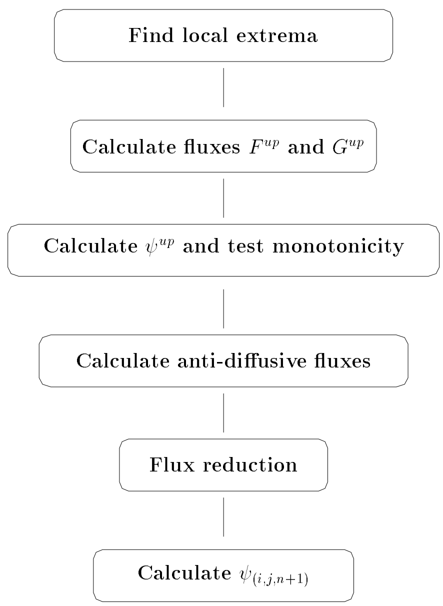


Figure 4: *Order of horizontal transport calculations in HYCOM 2.0.01*

3.5.5 Variables

Identification

Notation in the theory	Notation in **advem.f**
T, S	`fld(i,j)`
ϵ	`?????`
h_1, h_2	`fco(i,j),fc(i,j)`
F, G	`flx(i,j),fly(i,j)`
$Div\mathcal{F}$	`flxdiv(i,j)`
$u\Delta p, v\Delta p$	`u(i,j),v(i,j)`
$1/\Delta x$	`scali(i)`

The different arrays that store the intermediate results are introduced by the statement :

```
common/work/fmx(idm,jdm), fmn(idm,jdm), flp(idm,jdm), fln(idm,jdm),
flx(idm,jdm), fly(idm,jdm), flxdiv(idm,jdm)
```

Local variables

Subroutine advem

`amount`	Limited flux.
`clip`	Total clipped amount.
`dt`	Temporal increment.
`fco(i,j),fc(i,j)`	Depth of layer at previous and new time step.
`fld(i,j)`	Constituent.
`fln(i,j),flp(i,j)`	Flux reduction factors.
`flx(i,j),fly(i,j)`	Flux of constituent in each direction x and y.
`flxdiv`	Flux change.
`fmn(i,j),fmx(i,j)`	Extrema of `fld` in a nearby neighborhood.
`i, ia, ib`	Array indices.
`iord`	Order of scheme (1 or 2, 1 for simple donor cell scheme).
`j, ja, jb`	Array indices.

`l`	Loop index.
`lpipe_advem`	Flag to compare two model runs.
`mbdy_a`	Halo extent.
`onemu`	Small pressure.
`q`	Flux.
`scal(i)`	Spatial increments squared.
`scali(i)`	Inverse of scal.
`u1`	U-flux.
`u(i,j), v(i,j)`	Mass fluxes satisfying continuity equation.
`util1, util2`	Work array.
`v1`	V-flux.
`vlume`	Total volume.

4 Momentum Equation : momtum.f

In an isopycnic system, the momentum conservation equation is of the form :

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \frac{\mathbf{u}^2}{2} + (\zeta + f)\mathbf{k} \times \mathbf{u} = -\nabla M - g\frac{\partial \boldsymbol{\tau}}{\partial p} \tag{60}$$

with :

ζ : Relative vorticity
$\mathbf{k}$: Vertical unit vector
M : Montgomery potential
$\boldsymbol{\tau}$: Reynolds stress
f : Coriolis parameter

The subroutine `momtum` computes the following forcing terms of the momentum equation (60) :

1. the Montgomery potential;
2. the surface wind effects;
3. the bottom drag.

4.1 Forcing

4.1.1 Montgomery potential

To the initial distribution (figure 5), in HYCOM 2.0.01, the surface condition $p_1^{init} = 0$ is added. The portion of the ocean represented is at rest and the effects of the atmospheric pressure gradient are neglected. If, theoretically, the Montgomery potential is expressed in the general form $M = p + \rho g z$, it is not self-explanatory. However, a clear physical interpretation comes from letting Π_k be the potential that represents, along the vertical, the pressure deficit caused by the fact that the water column above the considered layer k has a density that is different from the one of this layer. The hydrostatic hypothesis is assumed. Over the initial profile, for the layer k, the pressure deficit is expressed by the recurrence relation :

$$\Pi_{k+1} = \Pi_k - g(\rho_{k+1} - \rho_k)\sum_{i=1}^{k}\langle h_k \rangle \tag{61}$$

As an initial condition, $\Pi_1 = 0$. In the bottom layer, where the density goes to ρ_N, Π_N is obtained by this relation.

When this stratified ocean is in motion, the height of the total column of water, not greater than H, is D such that :

$$D = \sum_{k=1}^{N} h_k \tag{62}$$

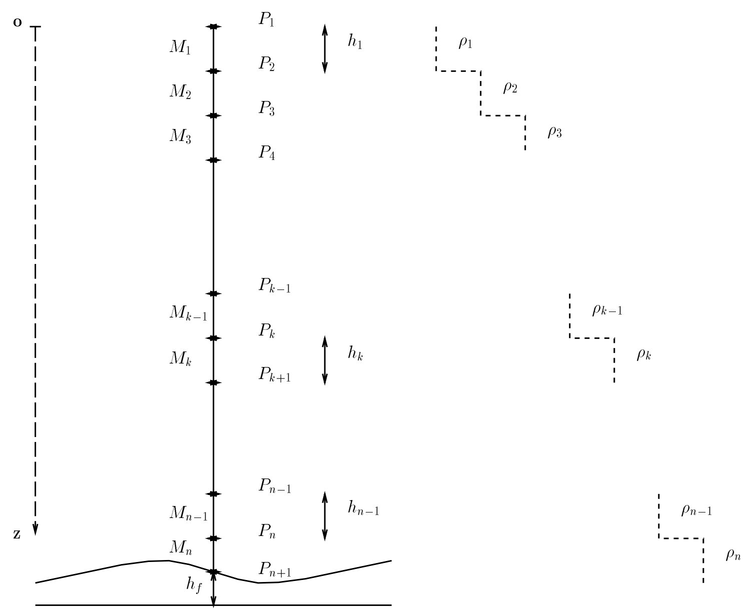


Figure 5: *Initial vertical distribution of pressure, density, and Montgomery potential*

In the moving ocean, the Montgomery potential M is equivalent to the potential Π established by the initial state. So the Montgomery potential in layer k is given by the recurrence relation :

$$M_k = M_{k+1} + g(\rho_{k+1} - \rho_k) \sum_{i=1}^{k} h_k \tag{63}$$

As η is independent of the layer K:

$$M_k = M_{k+1} + g(1+\eta)(\rho_{k+1} - \rho_k) \sum_{i=1}^{k} h'_k \tag{64}$$

To connect the potential Π_k to the Montgomery potential M_k present in the baroclinic momentum equations (*cf.* § 4.2), start at the bottom. Assume a homogeneous ocean of density ρ_N. At rest, the bottom pressure is $\pi_b = g\rho_N H$. When the ocean is moving this pressure becomes $\pi_b = g\rho_N D$. Because of this movement, the ocean bottom experiences a pressure deficit which is :

$$\delta\pi_b = -g\rho_N(D - H) \tag{65}$$

Since the decomposition (1) was used, the potential is therefore written :

$$\delta\pi_b = -g\eta\rho_N H \tag{66}$$

The potential Π_N reflects the pressure deficit at the bottom in an ocean where the density evolves from ρ_1 in the surface layer to ρ_N in the bottom layer. To obtain the baroclinic Montgomery potential, the pressure deficit at the bottom must be considered. It experiences a homogeneous ocean of instantaneous height $D = (1+\eta)H$ and of density ρ_1 :

$$\delta'\pi_b = -g\eta\rho_1 H \tag{67}$$

The general form giving the Montgomery potential at the bottom is finally obtained :

$$M_N = \Pi_N - g\eta H(\rho_N + \rho_1) \tag{68}$$

Once the potential M_N is calculated, the form (63) is applied to obtain the potential M_k of each layer up to the surface. In fact, in HYCOM 2.0.01, the surface layer is a mixed layer where the density ρ_s varies with time. Now, in the preceding calculations, a constant value ρ_1 serves to represent the surface density. To express the Montgomery potential for the mixed layer, this particularity needs to be considered. So, using the recurrence relation (63), write :

$$M_1 = M_2 + g(\rho_2 - \rho_s)h_1 \tag{69}$$

Then :

$$M_1^{fin} = M_1 + h_1\left[(\rho_s - \rho_1) - (\rho_2 - \rho_1)\right] \tag{70}$$

For the other part, if the sum over the N layers of the decomposition (1) is expressed using the form : $p_b = p'_b + p''_b$ where :

$$p''_b = \eta \sum_{k=1}^{N} \Delta p'_k = \eta p'_b, \tag{71}$$

the new report of the perturbation η to the reference ocean can be calculated in time by the relation :

$$\eta = \frac{p''_b}{p'_b} \tag{72}$$

4.1.2 Bottom drag

In HYCOM 2.0.01, to model dissipation by bottom drag, the following quadratic form is introduced :

$$\boldsymbol{\tau}_b = C_D\left|\overline{\mathbf{u}}_b\right|\overline{\mathbf{u}}_b \tag{73}$$

C_D is a drag coefficient. In version 2.0.01of HYCOM, $\overline{\mathbf{u}}_b$ represents the average velocity in a slice of water of thickness δz situated just above the bottom. Set $\delta z = 10\ m$. For the case when the thickness of layer N is less than the fixed value δz, it is necessary to use an average (the sum of the thicknesses of layers $N, N-1, \ldots, k$) :

$$\overline{\mathbf{u}'}_b = \frac{1}{\delta z}\left(\sum_{l=N}^{k} \mathbf{u}'_l h'_l + \mathbf{u}'_{k-1}\delta h'_{k-1}\right) \tag{74}$$

with :

$$\delta z = \sum_{l=N}^{k} h'_l + \delta h'_{k-1} \tag{75}$$

and put :

$$\overline{\mathbf{u}}_b = \overline{\mathbf{u}} + \overline{\mathbf{u}'}_b \tag{76}$$

So that the drag remains always significant, a residual velocity $\overline{c}$ is introduced and the following expression is calculated :

$$\mathcal{D}_{i,j} = C_D \left(\sqrt{\overline{u}_b^2 + \overline{v}_b^2} + \overline{c} \right) \tag{77}$$

In HYCOM 2.0.01, $\overline{c} = 10\ cm/s$ and $C_D = 3 \times 10^{-3}$. The vertical divergence of the bottom-induced stress is calculated as :

$$\frac{\partial \boldsymbol{\tau}_b}{\partial p} = \begin{cases} \frac{\partial \tau_{b_x}}{\partial p} \\ \\ \frac{\partial \tau_{b_y}}{\partial p} \end{cases} \tag{78}$$

Only the momentum in layers with indices $N, N-1, \ldots, k$ situated in the δz slice of water feels the dissipation term. Moreover, assume that the drag stress varies linearly inside each layer. So, if the bottom layer thickness is strictly equal to δz :

$$\left(\frac{\partial \tau_{b_x}}{\partial p} \right)_{i,j} = \frac{1}{2} u_{i,j,N} \left(\mathcal{D}_{i,j} + \mathcal{D}_{i-1,j} \right) / \delta z \tag{79}$$

On the other hand, when the bottom layer thickness is greater than δz, the fact that only a fraction of layer N feels the dissipation must be considered. So,

$$\left(\frac{\partial \tau_{b_x}}{\partial p} \right)'_{i,j} = \left(\frac{\partial \tau_{b_x}}{\partial p} \right)_{i,j} \frac{\delta z}{\Delta p'_{i,j,N}} \tag{80}$$

If the dissipation layer is made up of more layers, apply similar reasoning to distribute the dissipation over the N_f layers concerned.

4.1.3 Influence of the wind

In HYCOM 2.0.01, to model the surface mechanical effect of the wind, the monthly climatological files of wind stress $\boldsymbol{\tau}_s$, are used to interpolate the values during the simulation with the aid of the coefficients $w0, w1, w2, w3$.

4.2 Baroclinic System

To use the barotropic-baroclinic splitting of motion given by the form (2), once the system (94) describing the behavior of the barotropic mode is established, by simple subtraction from the general system (60), the baroclinic system is obtained (Baraille & Filatoff, 1995) :

$$\begin{aligned}
\frac{\partial u_k'}{\partial t} + \frac{1}{2}\frac{\partial(u_k^2+v_k^2)}{\partial x} - (\zeta_k + f)v_i' - \zeta_k\overline{v} + \frac{\partial}{\partial x}\left[M_k - \frac{1}{\rho_r}(\eta p_b')\right] &= -\frac{\partial \overline{u}^*}{\partial t} \\
\frac{\partial v_k'}{\partial t} + \frac{1}{2}\frac{\partial(u_k^2+v_k^2)}{\partial y} - (\zeta_k - f)u_i' - \zeta_k\overline{u} + \frac{\partial}{\partial y}\left[M_k - \frac{1}{\rho_r}(\eta p_b')\right] &= -\frac{\partial \overline{v}^*}{\partial t}
\end{aligned} \tag{81}$$

The index $_k$ represents the range of the isopycnic layer considered.

4.2.1 Numerical scheme

A leapfrog numerical scheme is used.

4.2.2 Turbulent viscosity

Though the nonlinear terms present in the momentum conservation equations of system (81) will not be called to play a significant role in ocean-scale applications, the equations are expressed in their complete form. The horizontal turbulent viscosity is defined by the relation :

$$\nu_u = \max\left\{u_d\Delta x, \lambda\left[\left(\frac{\partial u}{\partial x} - \frac{\partial v}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right)^2\right]^{1/2}\Delta x^2\right\} \tag{82}$$

u_d is a bottom diffusion velocity and λ a constant. In HYCOM 2.0.01, $u_d = 2\ cm/s$ and $\lambda = 1$.

4.2.3 Turbulent momentum flux

To the general equations given by the system (81), a diffusion term of the general form $(\Delta p)^{-1}\nabla\cdot(\nu_u\Delta p\nabla\mathbf{u})$ is added. Similarly, to the flux term $\nu_T\Delta p\nabla T$ appearing in the heat conservation equation (*cf.* § 3), the pressure increment Δp present in the momentum flux expression is replaced by the harmonic average $\widetilde{\Delta p} = 2/(\Delta p_{i-1}^{-1} + \Delta p_i^{-1})$ (*cf.* § 3.1.3). Moreover, writing this term in finite differences necessitates taking into account the possible presence of elevated bottom in neighboring cells. So, near partial boundaries, $(0 < w_{N,S,E,W} < 1)$, (*cf.* § 4.2.5), the turbulent flux results from the sum :

1. of flux pertaining to the portion of the cell boundary facing bathymetry; this flux is subject to the boundary condition $(s_L \pm 1)$;
2. of flux relating to the open part of the cell.

4.2.4 Intersection with the bathymetry

When an isocline intersects the bathymetry, a computational point of the Montgomery potential is then situated outside of the "wet" volume. In this case, in HYCOM 2.0.01, the gradient of the Montgomery potential is calculated by the weighted average values of four points of the neighboring grid $(i',j') = (i \pm 1, j), (i, j \pm 1)$ (Bleck & Smith, 1990) :

$$\left(\widetilde{\frac{\partial M}{\partial x}}\right)_{i,j,k} = \left[\sum_{i',j',k} \tilde{w}_{i',j',k} \left(\frac{\partial M}{\partial x}\right)_{i',j',k}\right] / \sum_{i',j'} \tilde{w}_{i',j',k} \tag{83}$$

The coefficients $\tilde{w}_{i',j',k}$ are defined by the formulae :

$$\begin{aligned} \tilde{w}_{i\pm1,j,k} &= \min\left(H_1, \Delta p'_{b_{i\pm1,j,k}}, \Delta p'_{s_{i\pm1,j,k}}\right) \\ \tilde{w}_{i,j\pm1,k} &= \min\left(H_1, \Delta p'_{b_{i,j\pm1,k}}, \Delta p'_{s_{i,j\pm1,k}}\right) \end{aligned} \tag{84}$$

with :

$$\Delta p'_{b_{i,j,k}} = p'_{b_{i,j}} - p'_{i,j,k} \quad \text{et} \quad \Delta p'_{s_{i,j,k}} = p'_{i,j,k+1} - p'_{s_{i,j}} \tag{85}$$

H_1 is the thickness given by weighting (83) of the Montgomery potential. For ocean applications, typically $H_1 = 10\ m$. Writing the coefficients $\tilde{w}_{i',j',k}$ allows simultaneous treatment of the problems of intersection of interfaces with the bathymetry and that of the outcropping of interfaces with the surface. In fact, since HYCOM 2.0.01 uses a surface mixed layer, it requires the mechanism of outcropping of isopycnic layers into the mixed layer to be modelled. Therefore, set $p'_{s_{i,j}} = p'_{i,j,2}$. To avoid discontinuities, the following quantity is introduced :

$$\left(\frac{\partial M}{\partial x}\right)^{fin}_{i,j,k} = \left[\left(\frac{\partial M}{\partial x}\right)_{i,j,k} + \left(H_1 - \tilde{w}_{i,j,k}\right)\left(\widetilde{\frac{\partial M}{\partial x}}\right)_{i,j,k}\right] / H_1 \tag{86}$$

4.2.5 Boundary conditions

In the presence of a solid boundary, two types of physical boundary conditions are introduced :

1. slip conditon : $\partial u_s / \partial \pi = 0$;
2. no-slip conditon : $u_s = 0$.

u_s is the velocity component tangential to the boundary and π is the normal direction. Suppose a domain is bounded by land to the East. In a point of (i,j,k) coordinates, to obtain an indication of the presence of this solid boundary, calculate the quantity :

$$w_{E_{i,j.k}} = \max\left\{0, \min\left[1, \frac{p'_{i,j,k+1} - p'_{b_{i,j+1}}}{\max\left(p'_{i,j,k+1} - p'_{i,j,k}, \epsilon\right)}\right]\right\} \tag{87}$$

ϵ is used in the denominator of this expression to avoid division by zero. Three cases arise (*cf.* figure 8) :

1. $w_E = 1$ if the layer k intersects the bathymetry ;
2. $w_E = 0$ if it does not intersect the bathymetry ;
3. $0 < w_E < 1$ if only one part of the layer intersects the jump in the bathymetry existing between the grid points (i, j) and $(i, j + 1)$.

Introducing the variable s_L to represent the boundary condition chosen :

a) $s_L = 1$: slip ;
b) $s_L = -1$: no-slip

If a solid boundary is detected, it is then possible to express the particular behavior of the fluid in the immediate surroundings of the boundary by establishing the auxiliary velocity :

$$u_{E_{i,j,k}} = (1 - w_{E_{i,j,k}})u_{i,j,k} + w_{E_{i,j,k}} u_{i,j,k} s_L \tag{88}$$

So if an Eastern boundary is detected (w_E=1),

a) if $s_L = 1$: $u_{E_{i,j,k}} = u_{i,j,k}$;
b) if $s_L = -1$: $u_{E_{i,j,k}} = -u_{i,j,k}$

For each layer k and at each point (i, j), the four coefficients w_N, w_S, w_E, w_W that represent the presence of boundaries respectively to the North, South, East and West of the domain are calculated. These variables establish the auxiliary velocity field u_N, u_S, u_E, u_W. This procedure remains valid as well for the case of a partial boundary ($0 < w_E < 1$).

4.2.6 Vorticity

In the formalism of HYCOM, the nonlinear terms and the Coriolis term are grouped to make the vorticity appear as ($\zeta = \partial v/\partial x - \partial u/\partial y$) :

$$\mathbf{u} \cdot \nabla \mathbf{u} + f\mathbf{k} \times \mathbf{u} = \nabla \frac{\mathbf{u}^2}{2} + (\zeta + f)\mathbf{k} \times \mathbf{u} \tag{89}$$

To conserve entropy, the components of the vorticity term $(\zeta + f)\mathbf{k} \times \mathbf{u}$ are written in the difference forms :

$$\begin{aligned} -v(\zeta + f) &= -\overline{V}^{x,y}\overline{Q}^{y} \\ u(\zeta + f) &= -\overline{U}^{x,y}\overline{Q}^{x} \end{aligned} \tag{90}$$

where the mass flux $(U, V) = u\Delta\overline{p}^{x}, v\Delta\overline{p}^{y}$ and the potential vorticity $Q = (\zeta + f)/\Delta\overline{p}^{x,y}$ are introduced. The operator $\overline{(\ )}$ permits the recentering of differences of pressure at computational points of the velocity components on a C grid. In an isopycnic shallow-water model, there is no guarantee that the instantaneous thickness of the considered layer remains non-zero. To address this problem, Bleck & Smith (1990) developed the "one-eighths rule" method. It remedies abnormal situations which arise when certain layer thickness values vanish. Before estimating the potential vorticity field $Q_{i,j,k}$, the sum of two larger values of the variable Δp among the 4 points surrounding the point considered is calculated for each layer at each point. This sum is denoted : $\Pi_{i,j}$. The denominator which is used in the vorticity expression then takes the value :

$$\Delta p_{max} = \max\left\{\Delta\overline{p}^{x,y}_{i,j}, \frac{1}{8}\max\left(\Pi_{i,j}, \Pi_{i-1,j}, \Pi_{i+1,j}, \Pi_{i,j-1}, \Pi_{i,j+1}\right)\right\} \tag{91}$$

In the case where the two arguments of this collation are zero, a residual numerical value δp is introduced. In HYCOM 2.0.01, $\delta p = 10^{-2}\ m$ is used.

4.3 Usage

In HYCOM 2.0.01, the numerical calculation of forcing terms in the momentum equations and the evolution of the baroclinic velocity components are realized by the subprogram :

```
subroutine momtum(m,n)
```

4.3.1 Order of operations

Calculation of Forcing Terms

The first step consists of calculating the Montgomery potential at the bottom. For this, the relation (68) is utilized. The potential Π_{N+1} is calculated from the field of initial conditions (subroutine **inicon**). Once the coefficient $(1+\eta)$ has been evaluated with the help of expression (72), it is then possible to determine the Montgomery potential at the N interfaces above by the relation (63).

```
[***** ADD SOURCE CODE HERE *****]
```

Next, the invariant part (along the vertical) of the bottom drag is estimated by applying the formula (77) introduced in Section 4.1.2.

```
[***** ADD SOURCE CODE HERE *****]
```

For each of the two horizontal components, the next step does the following operations in order :

1. Calculation of the spatial average of the barotropic vorticity originating from the barotropic-baroclinic splitting (*cf.* § 4.2, system 81).
2. Time interpolation of the wind stress.
3. Determination of the wind forcing of the surface layer term, described in Section 4.1.3 to which the calculation of the Montgomery potential gradient for the mixed layer is adjoined.

```
[***** ADD SOURCE CODE HERE *****]
```

Calculation of Velocity Components

The algorithm follows a treatment by layer.

The first step consists of calculating the total currents at times $(n-1)\Delta t$ and $n\Delta t$ and the associated fluxes. Then, to determine the optimum value of the difference in pressure given by the formula (91).

```
[***** ADD SOURCE CODE HERE *****]
```

The next operation consists of calculating indicators of the presence of bottom topography in neighboring cells by the method described in Section 4.2.5.

```
[***** ADD SOURCE CODE HERE *****]
```

In the next phase, the relation (82) is applied and the turbulent momentum fluxes are calculated by the procedure mentioned in Section 4.2.3. Then comes the inference of the weighted gradient of the Montgomery potential following the theory outlined in Section 4.2.4. If the layer concerned includes drag dissipation, it is integrated following the steps described in Section 4.1.2. The new values of the velocity components are now ready to be estimated.

```
[***** ADD SOURCE CODE HERE *****]
```

4.3.2 Flowcharts

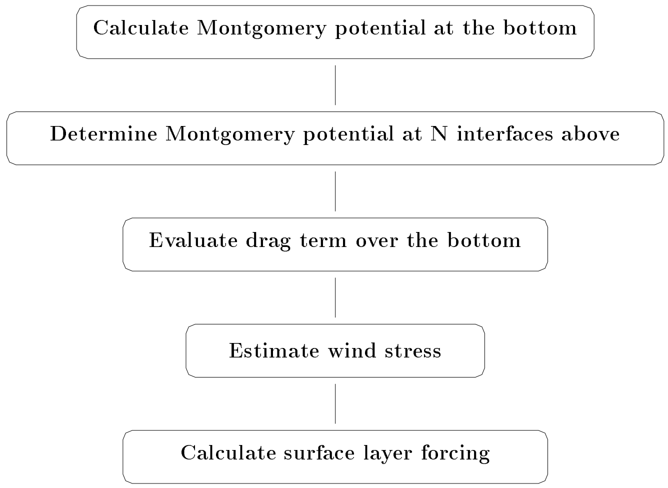


Figure 6: *Order of the treatment of the forcing terms in the momentum equations*

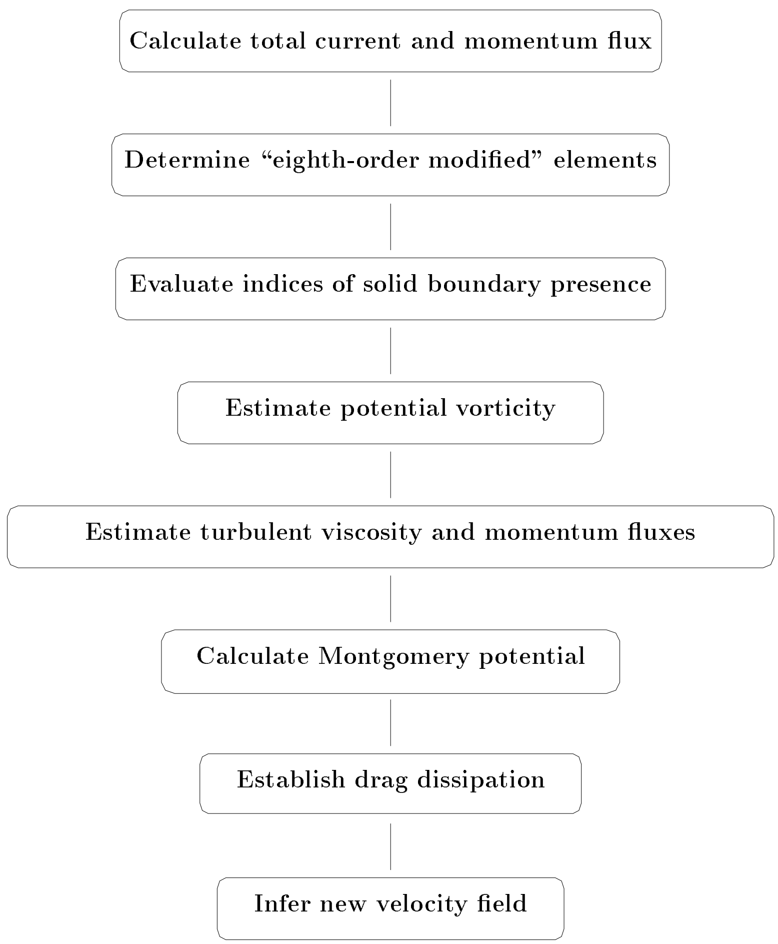


Figure 7: *Order of the treatment of the momentum equation*

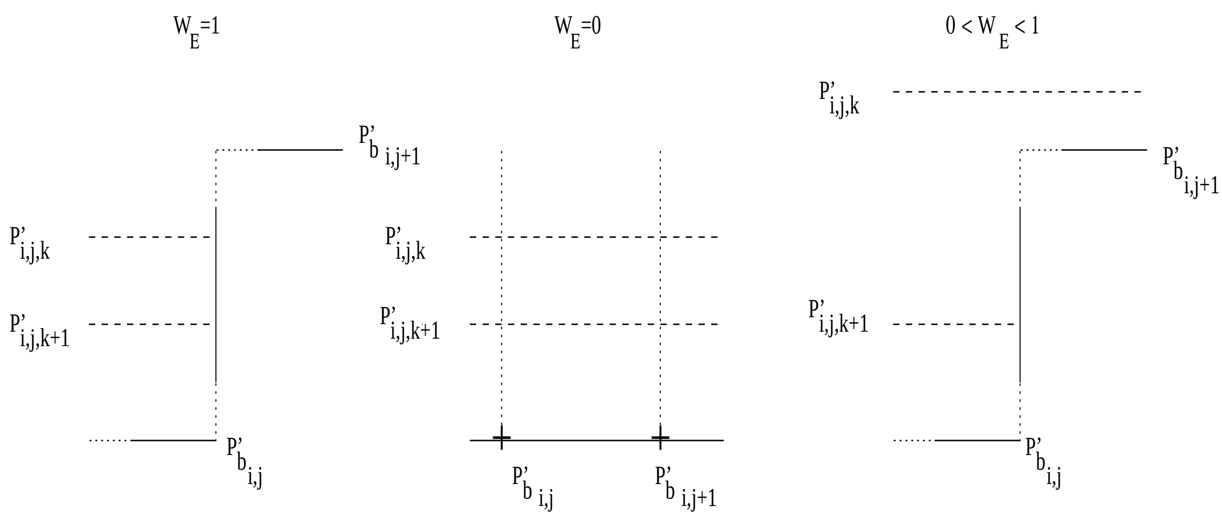


Figure 8: *Schematic of the intersection of layers with solid boundaries*

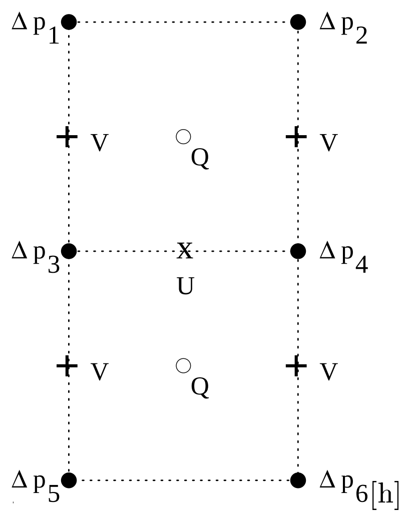


Figure 9: *Distribution of variables used in the evaluation of the Coriolis term at the central point* $u(i,j)$

4.4 Variables

4.4.1 Identification

Notation in the theory	Notation in **momtum.f**
$\mathcal{D}_{i,j}$	`drag(i,j)`
$M_{i,j,k}$	`montg(i,j,k)`
$p''_b = \eta p'_b$	`pbavg(i,j)`
$\overline{u}_b, \overline{v}_b$	`ubot(i,j),vbot(i,j)`
$\Pi_{i,j,N+1}$	`psikk(i,j)`
$\overline{u'}_b, \overline{v'}_b$	`util1(i,j),util2(i,j)`
δz	`thkbot`
$\zeta\overline{u}, \zeta\overline{v}$	`ubrhs(i,j),vbrhs(i,j)`
ρ_s, ρ_k	`thmix(i,j,n), theta(k)`
τ_{s_x}, τ_{s_y}	`stresx,stresy`
$f(x,y)$	`corio(i,j)`

H_1	`h1`
$(\partial M/\partial x)_{i',j'}, (\partial M/\partial y)_{i',j'}$	`pgfx(i,j),pgfy(i,j)`
$(\partial M/\partial x)^{fin}_{i,j}, (\partial M/\partial y)^{fin}_{i,j}$	`gradx(i,j),grady(i,j)`
$\Pi_{i,j}$	`dpmx(i,j)`
$(u_x - v_y)^2$	`defor1(i,j)`
$(v_x + u_y)^2$	`defor2(i,j)`
u_d	`veldff`
u_O, u_E	`uja(i,j),ujb(i,j)`
v_N, v_S	`via(i,j),vib(i,j)`
w_O, w_E	`wgtja(i,j),wgtjb(i,j)`
w_N, w_S	`wgtia(i,j),wgtib(i,j)`
$\tilde{w}_{i',j'}$	`util1(i,j)`
$\zeta(x,y)$	`vort(i,j)`
λ	`viscos`
ν	`visc(i,j)`
ϵ	`epsil`

4.4.2 Local variables

Subroutine momtum

`a, b`	First and second argument (respectively) to *hfharm*.
`botstr(i,j)`	Drag dissipation.
`cutoff`	Cutoff thickness (0.5 meter).
`dl2u`	Del-squared u.
`dl2uja`	Del-squared uja.
`dl2ujb`	Del-squared ujb.
`dl2v`	Del-squared v.
`dl2via`	Del-squared via.

`dl2vib`	Del-squared vib.
`dpdn`	Vertical excursion above the mixed layer base.
`dpia, dpib`	
`dpja, dpjb`	Intermediate variables in the calculation of turbulent flux in the presence of a boundary.
`dpmx`	Maximum (or minimum) layer thickness.
`dpup`	Vertical excursion above the mixed layer base.
`dpxy`	Layer thickness (at least one millimeter).
`dt1inv`	1/delt1
`hfharm`	Harmonic mean divided by two.
`i, ia, ib`	Array indices.
`ifirst`	First call flag.
`j, ja, jb`	Array indices.
`k, ka`	Layer indices.
`l`	Loop index.
`lpipe_momtum`	Flag to compare two model runs.
`m, n`	Time step index.
`mask`	Comparison mask.
`mbdy`	Halo extent.
`oneta`	1+eta = p_total/p_prime
`phi, plo`	Intermediate variables in the calculation of the average velocity near the bottom.
`pstres`	Layer thickness for wind stress.
`ptopl, pbotl`	Intermediate variables in the drag calculation.
`q`	Flux.
`scvxa, scvxb`	Grid spacing.
`scuya, scuyb`	Grid spacing.

`stress`	Top and bottom boundary layer stress.
`stresx, stresy`	Components of the wind stress.
`text`*12	Comparison title.
`thkbop`	Layer thickness in meters.
`ubot, vbot`	Components of velocity near the bottom.
`visca, viscb`	Viscosity.
`visu, visv`	U and V viscosity.
`vort`	Vorticity.
`wgtia, wgtib`	Sidewall extent.
`wgtja, wgtjb`	Sidewall extent.

4.5 Procedures

Functions	`harmon????`
Subroutines	`momtum`

5 Barotropic mode : barotp.f

5.1 Formalism and numerical techniques

In introducing the decomposition (1) and summing over all layers of the general continuity equation, the following form is obtained :

$$\frac{\partial \eta p_b'}{\partial t} + \nabla \cdot [(1+\eta)\overline{\mathbf{u}} p_b'] = 0 \tag{92}$$

To establish the corresponding equations of motion, consider the average of the general form (60) in the sense of (3). Next, the decomposition (2) is introduced, then the relations (4) and (1). Finally, the system is obtained (Baraille & Filatoff, 1995) :

$$\begin{aligned} &\frac{\partial}{\partial t}\left[\overline{u}(1+\eta)\right] - (1+\eta) f \overline{v} + \frac{1}{\rho_r}\frac{\partial}{\partial x}\left[\eta p_b'(1+\eta)\right] + \frac{R + Q(\eta)}{p_b'} = 0 \\ &\frac{\partial}{\partial t}\left[\overline{v}(1+\eta)\right] + (1+\eta) f \overline{u} + \frac{1}{\rho_r}\frac{\partial}{\partial y}\left[\eta p_b'(1+\eta)\right] + \frac{R' + Q'(\eta)}{p_b'} = 0 \end{aligned} \tag{93}$$

which is written more simply :

$$\begin{aligned} &\frac{\partial \overline{u}}{\partial t} - f\overline{v} + \frac{1}{\rho_r}\frac{\partial}{\partial x}(\eta p_b') = \frac{\partial \overline{u}^*}{\partial t} \\ &\frac{\partial \overline{v}}{\partial t} + f\overline{u} + \frac{1}{\rho_r}\frac{\partial}{\partial y}(\eta p_b') = \frac{\partial \overline{v}^*}{\partial t} \end{aligned} \tag{94}$$

$Q(\eta)$ and $Q'(\eta)$ are expressions grouping the nonlinear terms. R and R' represent the components of pressure gradient induced by the stratification. The pseudo-velocity $\overline{\mathbf{u}}^*(\overline{u}^*, \overline{v}^*)$ assures the property (4).

The time step of the external mode, Δt_B (barotropic), and the time step of the internal mode, Δt_b (baroclinic), are denoted such that :

$$\Delta t_b = \mathcal{N} \Delta t_B \tag{95}$$

The barotropic equations are therefore solved $\mathcal{N}$ times between two solutions of the baroclinic equations.

5.1.1 Rescaling of variables

In the last step of the continuity equation, a tendency calculation that uses layer thicknesses obtained at height points by the application of formula (33) necessitates a rescaling of the variables since they are distributed on a C grid. In fact, during initialization (subroutine `inicon`), the depths at velocity points have been introduced by the relations :

$$_u p'_{b_{i,j}} = \min\left(p'_{b_{i-1,j}}, p'_{b_{i,j}}\right) \text{ and } {}_v p'_{b_{i,j}} = \min\left(p'_{b_{i,j-1}}, p'_{b_{i,j}}\right) \tag{96}$$

To retain consistency with the initial definitions, at the point of the u component, the average thickness is introduced :

$$\begin{aligned} {}_u\Delta p'_{i,j,k} &= \max\{0., \\ & \min\left[{}_u p'_{b_{i,j}}, 1/2\left({}_u p'_{i,j,k+1} + {}_u p'_{i-1,j,k+1}\right)\right] \\ &- \min\left[{}_u p'_{b_{i,j}}, 1/2\left({}_u p'_{i,j,k} + {}_u p'_{i-1,j,k}\right)\right]\} \end{aligned} \tag{97}$$

The equivalent formula is used at the point of the v component.

5.1.2 Rearrangement of the velocity profile

When the thickness of a layer is so small as to be considered numerically zero, it may still contain momentum. When layer k "disappears", in terms of momentum it is still considered to exist and acquires the momentum of the layer above. To translate this mechanism, the variable q is introduced :

$$q = \delta - \min\left({}_u\Delta p'_k, \delta\right) = \begin{cases} 0 & \text{if } {}_u\Delta p'_k \geq \delta \\ \delta - {}_u\Delta p'_k & \text{if } {}_u\Delta p'_k < \delta, \end{cases} \tag{98}$$

which permits the weighted value to be defined (the same principle applies to the v component) :

$$\underline{u}_k = \frac{1}{\delta}\left[u'_k\left({}_u\Delta p'_k\right) + u'_{k-1}(\delta - {}_u\Delta p'_k)\right] \tag{99}$$

In HYCOM 2.0.01, $\delta = 10^{-1}$ m.

5.1.3 Filtering

In the same manner as in the advection step (*cf.* § 3), Asselin filtering for the velocity is introduced to fix the problems of dispersion caused by the leapfrog scheme. Take the component u' of the baroclinic velocity for the layer k and the node (i, j). Then write :

$$\widehat{u'}^n \widehat{\Delta p'}^n = \left[u'^n(1-2\gamma)(\Delta p'^n + \epsilon) + \gamma\left(\widehat{u'}^{n-1}\widehat{\Delta p'}^{n-1} + u'^{n+1}\Delta p'^{n+1}\right)\right] \tag{100}$$

A residual thickness ϵ is introduced for which this form remains valid when ${}_u\Delta p'_{i,j,k} \to 0$. In HYCOM, ϵ is set to a numerical value of 10^{-3} m and $\gamma = 0.25$. The thickness of the layer ${}_u\Delta p'_{i,j,k}$ is also filtered by the formula :

$$\widehat{\Delta p'}^n = (1-2\gamma)(\Delta p'^n + \epsilon) + \gamma\left(\widehat{\Delta p'}^{n-1} + \Delta p^{n+1}\right) \tag{101}$$

5.1.4 Continuity equation

The continuity equation (92) is treated with the simplification $(1+\eta) \approx 1$. As indicated in Section 2, using this approximation does not perturb the property : $\partial p'_b/\partial t = 0$ (Baraille & Filatoff, 1995). The treated variable is therefore $p''_b = \eta p'_b$. Combining forward time-stepping :

$$P''^{m+1}_b = P''^{m}_b - \Delta t_B \ \nabla \cdot (\overline{\mathbf{u}} p'_b)^m \tag{102}$$

with Asselin time filtering gives :

$$P''^{m+1}_b = (1-\text{w})P''^{m}_b + \text{w}P''^{m-1}_b - \Delta t_B (1+\text{w}) \nabla \cdot (\overline{\mathbf{u}} p'_b)^m \tag{103}$$

Set $\text{w} = 0.125$.

5.1.5 Equations of motion

The equations of motion given by the system (94) call the reference density ρ_r introduced to represent the ocean of reference depth H. In HYCOM 2.0.01, the identification $\rho_r \equiv \rho_0$ is made (*cf.* § 17).

The vector $\partial \overline{\mathbf{u}}^*/\partial t$ appearing in the right hand side of the equations of system (94) can then be seen as a forcing term in the generation of the linear barotropic mode. The solution of this system necessitates the extraction of the component $\overline{\mathbf{u}}^*$. In the preceding step (subroutine `momtum`), the baroclinic velocity profile expressed by the variable $\mathbf{u}''_k = \mathbf{u}'_k + \overline{\mathbf{u}}^*$ is calculated. In carrying out the sums :

$$S_u = \sum_{k=1}^{N} u''_k \Delta p'_k \quad \text{and} \quad S_v = \sum_{k=1}^{N} v''_k \Delta p'_k \tag{104}$$

and in accounting for the property (4), :

$$\overline{u}^* = \frac{S_u}{p'_b} \quad \text{and} \quad \overline{v}^* = \frac{S_v}{p'_b} \tag{105}$$

Note that the pseudo-vector $\overline{\mathbf{u}}^*$ is not a variable of state in the system whose evolution is sought. This is to say that in the preceding step (subroutine `momtum`), the transition is effectively inferred by : $\mathbf{u}'^{n}_{i,j,k} \rightarrow (\mathbf{u}'_k + \overline{\mathbf{u}}^*)^{n+1}_{i,j}$. Moreover, the weighting w is introduced such that :

$$\overline{u}^{m+1}_{i,j} = (1-\text{w})\overline{u}^{m}_{i,j} + \text{w}\overline{u}^{m-1}_{i,j} - \Delta t_B (1+\text{w}) \left[-\alpha_0 \left(\frac{\partial p''_b}{\partial x} \right)^{m+1}_{i,j} + \overline{f\overline{v}}^{m}_{i,j} + \overline{u}^{*,n+1}_{i,j}/\Delta t_b \right] \tag{106}$$

The Coriolis term is expressed by the centered form :

$$\overline{f\overline{v}}_{i,j} = 1/8(f'_{i,j} + f'_{i,j+1}) \left[(\overline{v}_v p'_b)_{i,j} + (\overline{v}_v p'_b)_{i-1,j} + (\overline{v}_v p'_b)_{i,j+1} + (\overline{v}_v p'_b)_{i-1,j+1} \right] \tag{107}$$

with the barotropic potential vorticity f' defined as

$$f'_{i,j} = \frac{f}{p'_{b_{i,j}}}. \tag{108}$$

The continuity equation is solved first. The pressure gradient of the equation of motion (106) uses the value of the state of perturbation η coming from this calculation. The combination of the forward time-stepping in the continuity equation and backward time-stepping in the momentum equations is called the forward-backward scheme.

5.2 Usage

In HYCOM 2.0.01, the numerical calculation of the evolution of the barotropic mode is performed in the subprogram :

```
[***** ADD SOURCE CODE HERE *****]
```

5.2.1 Order of operations

The new values of the layer thicknesses coming from the continuity equation are introduced by calculating the pressures at interfaces :

```
[***** ADD SOURCE CODE HERE *****]
```

Then, for each velocity component, the next step is to rescale at each point the thicknesses of N layers by applying the procedure formulated in (97) and to effect the rearrangement of the vertical velocity profile :

```
[***** ADD SOURCE CODE HERE *****]
```

From these profiles the corresponding components of the pseudo-velocity $\overline{\mathbf{u}}^*$ are extracted by the calculation of sums given by (104) :

```
[***** ADD SOURCE CODE HERE *****]
```

The final operation consists of using the Asselin filtering on the baroclinic velocity components $\mathbf{u}'$:

```
[***** ADD SOURCE CODE HERE *****]
```

In the step which follows, the treatment of the equations of continuity and motion are performed $\mathcal{N}$ times, as described previously (*cf.* § 5.1.4 and § 5.1.5). The new values of the barotropic component $\overline{\mathbf{u}}$ and of the state of perturbation η are calculated using the Asselin weighting factor W :

```
[***** ADD SOURCE CODE HERE *****]
```

5.2.2 Flowchart

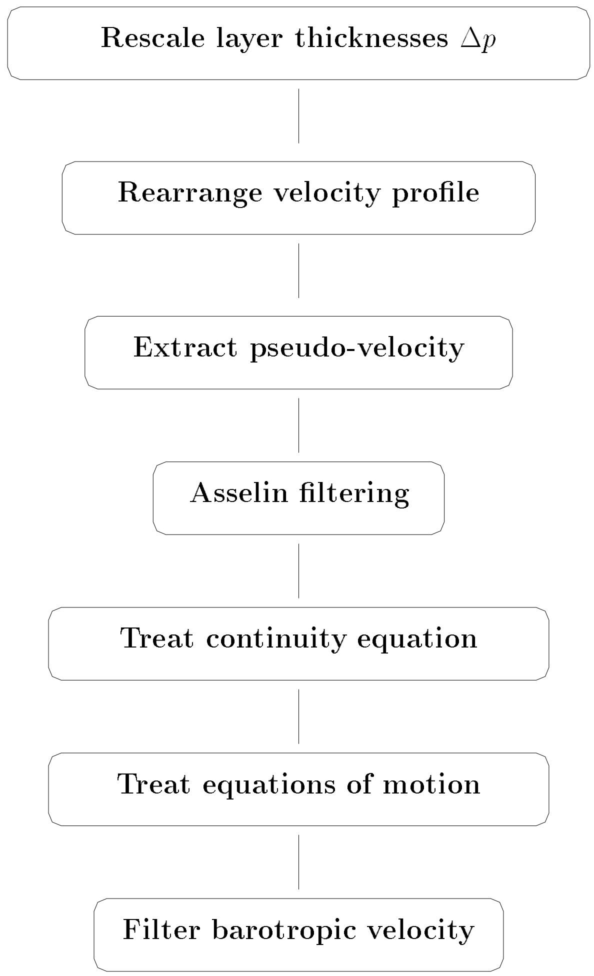


Figure 10: *Order of the barotropic mode calculation*

5.3 Variables

5.3.1 Identification

Notation in the theory	Notation in **barotp.f**
$\Delta t_B, \Delta t_b$	`dlt,?????`
$f'_{i,j}$	`pvtrop(i,j)`
$\gamma, (1-2\gamma)$	`????,????`
$\mathcal{N}$	`lstep`
$p''_{b_{i,j}}$	`pbavg(i,j)`
${}_u p'_{b_{i,j}}, {}_u p'_{b_{i,j}}$	`depthu(i,j),depthv(i,j)`
${}_u \Delta p'_{i,j,k}, {}_u p'_{i,j,k}$	`???(i,j,k),???(i,j,k)`
S_u, S_v	`utotn(i,j),vtotn(i,j)`
$\overline{u}^n_{i,j}, \overline{v}^n_{i,j}$	`ubavg(i,j,n),vbavg(i,j,n)`
W	`wbaro`

5.3.2 Local variables

Subroutine barotp

`i, j`	Array indices.
`l`	Loop index.
`ll`	Time step index.
`lll`	Time step loop index.
`lpipe_barotp`	Flag to compare two model runs.
`m, n, ml, nl, mn`	Time step indices.
`mbdy`	Halo extent.
`q`	Mean height correction.
`sump`	Total height.
`utndcy, vtndcy`	Tendency of the barotropic velocity components.
`vthenu`	V then U flag.

5.4 Procedures

Subroutines `barotp`

6 Ocean-atmosphere exchanges : thermf.f

In HYCOM 2.0.01, the ocean-atmosphere exchanges considered are of three types :

1. The radiative exchanges R : balance of incident solar radiation and radiation emitted by the sea surface.
2. Turbulent heat transfers
   - A latent heat transfer $\mathcal{E}$ due to evaporation of seawater ;
   - A sensible heat transfer $\mathcal{H}$ which is established by convection when a significant difference exists between the temperature of the sea surface and that of the air.
3. Mechanical energy transfers : essentially the effect of wind

6.1 Formalism and numerical techniques

The atmospheric thermal forcing is specified via the following surface fields:

. net radiation : `radflx(i,j,l)` ;
. shortwave radiation : `swflx(i,j,l)` ;
. wind speed at the sea surface (10m) : `wndspd(i,j,l)` ;
. air temperature : `airtmp(i,j,l)` ;
. water vapor content : `vapmix(i,j,l)` ;
. precipitation : `precip(i,j,l)`.

The fields are furnished at times $l\Delta\mathcal{T}$ where $\Delta\mathcal{T}$ is the sampling period and are interpolated to the simulation's timestep, $n\Delta t$, using the weighted average of four samples for monthly data (Hermite interpolation in time with weights w_0, w_1, w_2, w_3) or of two samples for higher frequency data (linear interpolation in time with weights w_0, w_1).

6.1.1 Heat balance

In HYCOM 2.0.01, the effect of ocean-atmosphere exchanges (except the mechanical effect of the wind) on the mixed layer is summed in the calculation of thermal balance :

$$B = R + \mathcal{H} + \mathcal{E} \tag{109}$$

A positive (*i.e.*, upward) flux of sensible heat corresponds to a loss, a contribution of energy by the sea following the sign of the difference between the sea and the atmospheric boundary layer :

$$\mathcal{H} = C_{p_{air}} E_x \; (T_s - T_a) \tag{110}$$

E_x is an exchange coefficient such that : $E_x = \rho_a C_T W$

ρ_a :	Mass/volume of the air
C_T :	Heat transfer coefficient
$C_{p_{air}}$:	Specific heat of the air
T_s :	Sea surface temperature
T_a :	Temperature in the atmospheric boundary layer
W :	Wind velocity

Responsible for important quantities of heat exchanged between the sea and atmosphere by evaporation, the latent heat flux always induces a heat loss by the ocean. To formulate this term in the balance estimation, in HYCOM 2.0.01, the following expression is used :

$$\mathcal{E} = E_x \mathcal{L}(H_u - E_v) \tag{111}$$

With :

$\mathcal{L}$:	latent heat of vaporization
H_u :	specific humidity
E_v :	evaporation

Moreover, the precipitation-evaporation balance in the evolution of the salinity (therefore the density) of the mixed layer is accounted for by prescribing the precipitation.

6.1.2 Mechanical energy transfers

A wind representable by the vector $\mathbf{W}(t)$ induces on the surface a wind stress $\boldsymbol{\tau}_s(t)$ which can be well-expressed by a quadratic expression of type :

$$\boldsymbol{\tau}_s = \rho_a \; C_D \; |\mathbf{W}|\mathbf{W} \tag{112}$$

C_D is a coefficient realizing sea surface drag. It is set at $C_D \simeq 10^{-3}$. The expression of the drag velocity at the surface u_* is obtained by the relation :

$$u_*^2 \, \mathbf{x} = \boldsymbol{\tau}_s / \rho_0 \tag{113}$$

At the point z, the classic parametrization of the concept of turbulent diffusion is used :

$$\boldsymbol{\tau}(z) = \rho \; K_M \; \frac{\partial \mathbf{u}}{\partial z} \tag{114}$$

with :
$\mathbf{x}$: horizontal unit vector
K_M : coefficient of turbulent momentum diffusion

6.1.3 Thermal forcing at sidewalls

∗ INSERT TEXT HERE ∗

6.1.4 Relaxation to SST and/or SSS

∗ INSERT TEXT HERE ∗

6.1.5 Alternative bulk parameterizations of air-sea fluxes

∗ INSERT TEXT HERE ∗

6.2 Usage

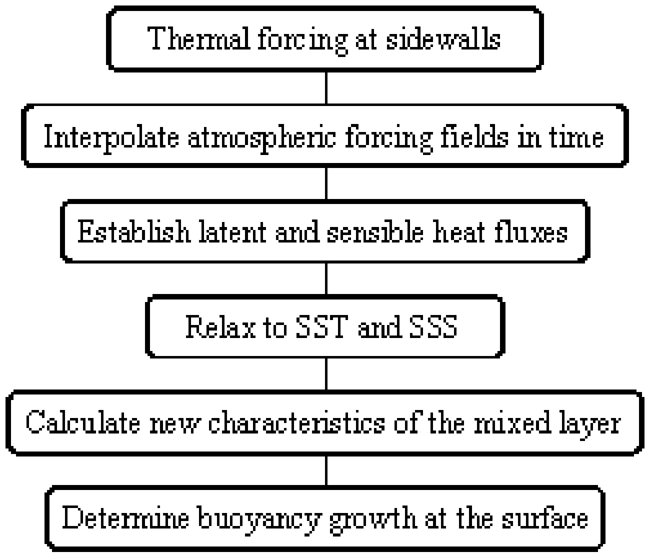
The numerical calculation of thermal balance is carried out by the subroutine :

```
subroutine thermf(m,n)
```

6.2.1 Order of operations

First, the values of thermodynamic parameters (radiation, wind, *etc*) are interpolated at an instant in the simulation. From there, it is possible to determine the flux of latent and sensible heat as well as the velocity drag at the surface. The following step accounts for the thermal balance by proceeding to the inference of new values of temperature, salinity, and density of the mixed layer. During the last phase, the buoyancy growth necessary to establish the Monin-Obukov length is calculated. This length is the reference in the step of modeling the evolution of the surface layer (*cf.* § 8).

6.2.2 Flowchart



ss

Figure 11: *Order of the thermal balance calculation in the mixed layer in HYCOM 2.0.01*

6.3 Variables

6.3.1 Identification

Notation in the theory	Notation in **thermf.f**
B	`surflx(i,j)`
$C_{p_{air}}$	`csubp`
$C_{p_{water}}$	`spcifh`
C_T	`ct`
E_x	`exchng`
E_v	`vpmx`
$\mathcal{E}$	`evap`
H_u	`qsatur`

$\mathcal{L}$	`??????`
R	`radflx`
u_*	`ustar`
W	`wind`
ρ_a	`airdns`

6.3.2 Local variables

Subroutine thermf

`d1, d2, d3, d4`	Field totals.
`i, j`	Array indices.
`k`	Layer index.
`l`	Loop index.
`m, n`	Time step indices.
`nm`	N or m.
`pmean`	Mean layer thickness.
`pwl`	Wall pressure.
`rareac`	1/sum extent.
`rmean`	Mean density.
`rmean0`	Initial mean density.
`runsec`	Model seconds in run.
`s1mean`	Mean surface salinity.
`secpyr`	Seconds per year.
`smean`	Mean salinity.
`smean0`	Initial mean salinity.
`t1mean`	Mean surface temperature.
`tmean`	Mean temperature.
`tmean0`	Initial mean temperature.

Subroutine thermfj

`airdns`	Air density at sea level. (kg/m**3)
`airt`	Air temperature (C).
`cd`	Drag coefficient.
`cd0`	Wind exchange coefficient.
`cekman`	Constant for calculating thickness of ekman layer.
`clh`	Latent heat exchange coefficient.
`clmax`	Maximum allowed cl.
`clmin`	Minimum allowed cl.
`cl0`	Latent heat polynomial coefficient.
`cl1`	Latent heat polynomial coefficient.
`csh`	Sensible heat exchange coefficient.
`csice`	Ice-air sensible exchange coefficient.
`csubp`	Specific heat of air at constant pressure. (j/kg/deg)
`ctl`	Thermal transfer coefficient (latent).
`cts1`	Thermal transfer coefficient (sensible, stable).
`cts2`	Thermal transfer coefficient (sensible, unstable).
`dsgdt`	Dsigdt at the surface.
`emnp`	Evaporation-precipitation balance.
`evap`	Evaporation balance.
`evaplh`	Latent heat of evaporation. (j/kg)
`i, j`	Array indices.
`l`	Loop index.
`m, n`	Time step indices.
`pairc`	Air pressure. (mb)*100

`prcp`	Precipitation (m/sec; > 0 since into ocean).
`qsatur`	Saturation specific humidity.
`radfl`	Net radiative thermal flux (w/m**2) into ocean.
`rair`	Density of air.
`rgas`	Gas constant. (j/kg/k)
`rmus`	Salinity relaxation coefficient.
`rmut`	Temperature relaxation coefficient.
`slat`	Latent heat coefficient.
`smn`	Time level averaged surface salinity.
`snsibl`	Sensible heat flux into atmosphere.
`ssen`	Sensible heat coefficient.
`swfl`	Shortwave radiative thermal flux (w/m**2) into ocean.
`tdif`	SST-airt.
`tmn`	Time level averaged surface temperature.
`tzero`	Celsius to kelvin temperature offset.
`ustrmn`	Minimum ustar.
`vpmx`	Evaporation.
`wind`	Wind speed. (m/s)
`wsmax`	Maximum allowed wind speed for *cl* and *cd*.
`wsmin`	Minimum allowed wind speed for *cl* and *cd*.
`wsph`	Wind speed between *wsmin* and *wsmax*.

6.4 Procedures

Subroutines	`thermf`, `thermfj`
Functions	`dsigds??`, `dsigdt??`, `qsatur??`

7 Energy Loan Sea Ice Model : icloan.f

7.1 Formalism and numerical techniques

In HYCOM, an energy loan sea ice model was developed to manage the energetics of water phase changes in a consistent yet simple manner. The model, which has much in common with the one developed by Semtner (1976) focusses on two aspects of the influence of sea ice:

1. the stabilization of ocean temperature near the freezing point through ice formation and melting;

2. the impact of the ice surface on ocean-atmosphere energy fluxes.

Concerning the stabilization of ocean temperature, the energy loan concept of the ice model ensures that the oceanic mixed layer temperature does not drop below the freezing point (-1.8°C) when the surface heat flux removes heat from the ocean. At each model grid point, the ocean borrows energy from an "energy bank" to stabilize temperature at the freezing point. The energy required to maintain this temperature comes from freezing an appropriate amount of seawater. Conversely, if the surface heat flux adds heat to the ocean, the energy loan must be repaid before the ocean temperature in a grid box is permitted to rise above freezing.

The influence of ice on surface fluxes is large, both by virtue of its high albedo compared to water and because an ice surface can be much colder than open water. In the present ice model, surface temperature is calculated based on the assumption that the system is energetically in a steady state; i.e., the heat flux through the ice matches the atmospheric heat flux.

7.1.1 Surface temperature flux

To illustrate this approach, the atmospheric heat flux is written as $F_{air} = a(T_i - T_a)$, and the heat flux through the ice as $F_{ice} = a(T_w - T_i)$, where T_i, T_a, and T_w represents ice, air, and water temperature while a and b are proportionality factors. Given T_a, T_w, and a first guess of T_i (the unknown in this problem), T_i is modified by an amount ΔT_i to minimize the difference between F_{air} and F_{ice}:

$$a(T_i + \Delta T_i - T_a) = b(T_w - T_i - \Delta T_i),$$

which yields

$$\Delta T_i = \frac{aT_a + bT_w}{a + b} - T_i. \tag{115}$$

To make this formula applicable in situations where $F_{air}$is a mixture of sensible, latent, and radiative heat fluxes, the expressions $aT_i - aT_i$ and $bT_i - bT_i$ are added to the numerator of (115), then the original definitions of F_{air} and F_{ice} are substituted:

$$\Delta T_i = \frac{F_{ice} - F_{air}}{a + b}. \tag{116}$$

The new temperature obviously must not be allowed to exceed the freezing point until the ice has melted completely.

Practical application of (116) requires knowledge of the coefficients a, b which represent the derivatives dF_{air}/dT_i and dF_{ice}/dT_i, respectively. Guidance on the magnitude of a can be obtained from the conventional heat flux "bulk" formula. It suggests that $a = c_t \rho c_p U$ where c_t is a nondimensional transfer coefficient (similar to the drag coefficient), ρ is the air density, c_p is the specific heat of air at constant pressure, and U is the wind speed. The formula for radiative energy loss, σT^4, suggests that the previous estimate for a should be increased by an amount $4\sigma T^3$. A reasonable choice for b is the ratio of ice thermal conductivity to ice thickness, k_{ice}/H_{ice}.

In the interest of computational efficiency in coupled climate models, information exchange with the atmosphere should be minimized. The coefficient a is therefore assumed to be independent of atmospheric state variables. To avoid oscillatory behavior in (116), a should be chosen somewhat larger than "typical" values of $c_t \rho c_p U + 4\sigma T^3$; in other words, a strategy of prudent under-relaxation of T_i is adopted.

Finally, a statement is needed to relate the rate of energy borrowing or repaying to the "composite" atmospheric heat flux:

$$F = cF_{ice} + (1 - c)F_{opw}, \tag{117}$$

where F_{opw} is the thermal energy flux over open water and c is the fractional ice coverage. As defined in (117), F is the energy flux felt by the ocean irrespective of the presence of ice. In other words, we assume that the energy flux between the atmosphere and ice, F_{ice}, equals the energy flux between ice and ocean. This assumption is compatible with the steady state (zero heat flux divergence) made to derive (116).

Concerning implementation of the ice model in HYCOM 2.0.01, the surface temperature is represented by the temperature of layer 1 regardless of whether the model is run with hybrid vertical coordinates or in MICOM mode. When the model is run with hybrid vertical coordinates and a non-slab mixed layer model, no attempt is made to reduce the thickness of layer 1 at a given grid point when ice forms.

7.2 Usage

In HYCOM 2.0.01, the numerical calculation of the influence of ice on surface temperature is carried out by the subroutine :

```
subroutine icloan(m,n)
```

7.2.1 Order of operations

* INSERT TEXT HERE *

7.2.2 Flowchart

∗ INSERT FLOWCHART HERE ∗

7.3 Variables

7.3.1 Identification

Notation in the theory	Notation in **icloan.f**
∗ INSERT TEXT HERE ∗	∗ INSERT TEXT HERE ∗

7.3.2 Local variables

Subroutine icloan

`covice`	Ice coverage (rel. units, 0.0 to 1.0).
`fusion`	Latent heat of fusion. (J/kg)
`hice`	Actual ice thickness. (m)
`i, j`	Array indices.
`icex`	Ice exceeding thkmax.
`l`	Loop index.
`m, n`	Time step indices.
`paybak`	Returned energy.
`rate`	Maximum ice melting rate. (m/sec)
`rhoice`	Density of ice. (kg/m^3)
`saldif`	Salinity difference water minus ice.
`salflx`	Salt flux (implied by fresh water flux).
`surflx`	Net total heatflux between atm and ocean or ice. (W/m^2)
`temice`	Ice surface temperature.
`tgrad`	Vertical temperature gradient inside ice. (deg/m)
`thicmx`	Maximum ice thickness. (m)
`thin`	Minimum ice thickness. (m)
`thkice`	Average ice thickness (=hice x covice(i,j)).

`thkmax`	Maximum ice thickness. (m)
`tice`	Melting point. (deg)
`tmxl`	Mixed-layer temp due to diab. forcing.

7.4 Procedures

Subroutines `icloan`

8 KPP Vertical Mixing : mxkpp.f

8.1 Formalism and numerical techniques

The K-Profile Parameterization (KPP; Large *et al.*, 1994; 1997) is the first non-slab mixed layer model included in HYCOM. The KPP model provides mixing from surface to bottom, smoothly matching the large surface boundary layer diffusivity/viscosity profiles to the weak diapycnal diffusivity/viscosity profiles of the interior ocean. There are numerous advantages to this model. It works on a relatively coarse and unevenly spaced vertical grid. It parameterizes the influence of a larger suite of physical processes than other commonly used mixing schemes. In the ocean interior, the contribution of background internal wave breaking, shear instability mixing, and double diffusion (both salt fingering and diffusive instability) are parameterized. In the surface boundary layer, the influences of wind-driven mixing, surface buoyancy fluxes, and convective instability are parameterized. The KPP algorithm also parameterizes the influence of nonlocal mixing of T and S, which permits the development of countergradient fluxes. Further details on physical assumptions and parameterizations used to construct the model are provided in Large *et al.*, 1994.

The KPP model is semi-implicit, requiring multiple iterations. For the first iteration, vertical profiles of diffusivity/viscosity coefficients are calculated at model interfaces from the initial profiles of model variables. The model variables are then mixed by solving the one-dimensional vertical diffusion equation at each grid point. For the second iteration, the vertically mixed profiles of model variables are used to estimate new diffusivity/viscosity profiles, which are then used to mix the original profiles of model variables. This procedure should be repeated until the mixed profiles of model variables differ insignificantly from the mixed profiles obtained from the previous iteration. HYCOM tests revealed that two iterations are reasonably adequate, given the computational overhead required for each one.

The full KPP procedure is first applied at pressure grid points, where thermodynamical variables and tracers are mixed. For this purpose, momentum components are horizontally interpolated to the pressure grid points. After completing the iterative procedure at pressure points, mixing is performed at momentum (u and v) points by interpolating the final viscosity profiles at the pressure points to the momentum points, then solving the vertical diffusion equation. The full iterative procedure is therefore not required at the momentum points.

The KPP algorithm does not require a convection algorithm that mixes adjacent layers when the upper layer is denser than the lower layer. HYCOM does perform convection when Kraus-Turner mixing is used, however.

8.1.1 Surface fluxes

Prior to executing the KPP algorithm, surface fluxes of thermodynamical variables and momentum are distributed entirely over the uppermost model layer, with the exception of penetrating shortwave radiation. Shortwave radiation can penetrate to deeper layers, with the penetration depth depending on water clarity. The two-component (red and blue light) exponential decay model of Jerlov (1976) is used to calculate penetrating shortwave radiation in HYCOM. If HYCOM is run in MICOM mode, or if the simple Kraus-Turner mixed layer model 2 is used, all shortwave radiation is absorbed in the mixed layer. If the full Kraus-Turner mixed layer model 1 is used, penetrating shortwave radiation can be invoked as an option. Otherwise, all shortwave radiation is absorbed in the mixed layer. Penetrating shortwave radiation is always used for KPP mixing and all non-slab mixed layer models that will be included in the future.

The depth of penetration is a function of water clarity, represented by the Jerlov water type. The water type is assigned integer values from 1 through 5, with 1 representing the clearest water. Given the incoming shortwave radiation flux S_0 at the surface, the flux passing through model interface k located at pressure p_k is

$$S_k = S_0 \left[r \exp\left(\frac{-p_{k+1}}{\beta_R}\right) + (1-r)\exp\left(\frac{-p_{k+1}}{\beta_B}\right)\right], \tag{118}$$

where r is the fraction of light that is red, β_R is the penetration depth scale of red light, and β_B is the penetration depth scale of blue light. The parameters for all five Jerlov water types are summarized in Table 2.

Table 2: Parameters for calculation of radiation flux based on Jerlov's water types

Jerlov Water Type	r	β_R	β_B
1	0.58	0.35	23.0
2	0.62	0.60	20.0
3	0.67	1.00	17.0
4	0.77	1.50	14.0
5	0.78	1.40	7.9

Presently, Jerlov water type is specified by the user for each grid point at the beginning of the model run. In the future, water type could be determined by biological or suspended sediment models.

There are two choices for bulk parameterization of surface fluxes in HYCOM. The first is the standard constant bulk coefficients. The second is the sophisticated parameterization scheme of Kara *et al.* (2000) that has been extensively tested by researchers at the Naval Research Laboratory, and that was embedded in HYCOM by Alan Wallcraft of NRL. The user also has the option of relaxing nearsurface temperature or salinity to climatology.

8.1.2 Diapycnal diffusivity in the ocean interior

Model variables are decomposed into mean (denoted by an overbar) and turbulent (denoted by a prime) components. Diapycnal diffusivities and viscosity parameterized in the ocean interior as follows:

$$\overline{w'\theta'} = -\nu_\theta \frac{\partial\overline{\theta}}{\partial z}, \quad \overline{w'S'} = -\nu_S \frac{\partial\overline{S}}{\partial z}, \quad \overline{w'\mathbf{v}'} = -\nu_m \frac{\partial\overline{\mathbf{v}}}{\partial z}, \tag{119}$$

where $(\nu_\theta, \nu_S, \nu_m)$ are the interior diffusivities of potential temperature, salinity (which includes other scalars), and momentum (viscosity), respectively. Interior diffusivity/viscosity is assumed to consist of three components, which is illustrated here for potential temperature:

$$\nu_\theta = \nu_\theta^s + \nu_\theta^w + \nu_\theta^d, \tag{120}$$

where ν_θ^s is the contribution of resolved shear instability, ν_θ^w is the contribution of unresolved shear instability due to the background internal wave field, and ν_θ^d is the contribution of double diffusion. Only the first two processes contribute to viscosity.

The contribution of shear instability is parameterized in terms of the gradient Richardson number calculated at model interfaces:

$$Ri_g = \frac{N^2}{\left(\frac{\partial\overline{u}}{\partial z}\right)^2 + \left(\frac{\partial\overline{v}}{\partial z}\right)^2}, \tag{121}$$

where mixing is triggered when $Ri_g = Ri_0 < 0.7$. Vertical derivatives are estimated at model interfaces as follows: Given model layer k bounded by interfaces k and $k+1$, the vertical derivative of $\overline{u}$ at interface $k$is estimated as

$$\frac{\partial\overline{u}}{\partial z} = \frac{\overline{u}^{k-1} - \overline{u}^k}{0.5 * (\delta h^k + \delta h^{k-1})} \tag{122}$$

where the denominator contains the thickness of layers k and $k-1$. The contribution of shear instability is the same for θ diffusivity, S diffusivity, and viscosity ($\nu^s = \nu_\theta^s = \nu_S^s = \nu_m^s$), and is given by

$$\begin{array}{ll} \frac{\nu^s}{\nu^0} = 1 & Ri_g < 0 \\ \frac{\nu^s}{\nu^0} = \left[1 - \left(\frac{Ri_g}{Ri_0}\right)^2\right]^P & 0 < Ri_g < Ri_0 \ , \\ \frac{\nu^s}{\nu^0} = 0 & Ri_g > Ri_0 \end{array} \tag{123}$$

where $\nu^0 = 50 \times 10^{-4}\mathrm{m}^2\mathrm{s}^{-1}$, $Ri_0 = 0.7$, and $P = 3$.

The diffusivity that results from unresolved background internal wave shear is given by

$$\nu_\theta^w = \nu_S^w = 0.1 \times 10^{-4}\mathrm{m}^2\mathrm{s}^{-1}. \tag{124}$$

Based on the analysis of Peters *et al.* (1988), Large *et al.* (1994) determined that viscosity should be an order of magnitude larger:

$$\nu_m^w = 1.0 \times 10^{-4} \mathrm{m}^2 \mathrm{s}^{-1}. \tag{125}$$

Regions where double diffusive processes are important are identified using the double diffusion density ratio calculated at model interfaces:

$$R_\rho = \frac{\alpha \frac{\partial \overline{\theta}}{\partial z}}{\beta \frac{\partial \overline{S}}{\partial z}}, \tag{126}$$

where α and β are the thermodynamic expansion coefficients for temperature and salinity, respectively. For salt fingering (warm, salty water overlying cold, fresh water), salinity/scalar diffusivity is given by:

$$\begin{array}{ll} \frac{\nu_S^d}{\nu_f} = \left[1 - \left(\frac{R_\rho - 1}{R_\rho^0 - 1}\right)^2\right]^P & 1.0 < R_\rho < R_\rho^0 \\ \frac{\nu_S^d}{\nu_f} = 0 & R_\rho \geq R_\rho^0 \end{array} \tag{127}$$

and temperature diffusivity is given by:

$$\nu_\theta^d = 0.7 \nu_S^d, \tag{128}$$

where $\nu_f = 10 \times 10^{-4} \mathrm{m}^2 \mathrm{s}^{-1}$, $R_\rho^0 = 1.9$, and $P = 3$. For diffusive convection, temperature diffusivity is given by:

$$\frac{\nu_\theta^d}{\nu} = 0.909 \exp\left\{4.6 \exp\left[-0.54\left(R_\rho^{-1} - 1\right)\right]\right\}, \tag{129}$$

where ν is the molecular viscosity for temperature, while salinity/scalar diffusivity is given by:

$$\begin{array}{ll} \nu_S^d = \nu_\theta^d \left(1.85 - 0.85 R_\rho^{-1}\right) R_\rho & 0.5 \leq R_\rho \leq 1 \\ \nu_S^d = \nu_\theta^d \left(0.15 R_\rho\right) & R_\rho < 0.5 \end{array} \tag{130}$$

8.1.3 Surface boundary layer thickness

Diagnosis of boundary layer thickness h_b is based on the bulk Richardson number

$$Ri_b = \frac{\left(B_r - B\right) d}{\left(\overline{\mathbf{v}}_r - \overline{\mathbf{v}}\right)^2 + V_t^2} \tag{131}$$

where B is buoyancy, d is depth, the subscript r denotes reference values, and where the two terms in the denominator represent the influence of resolved vertical shear and unresolved turbulent velocity shear, respectively. Reference values are averaged over the depth range εd, where $\varepsilon = 0.1$. At depth $d = h_b$, the reference depth εh_b represents the thickness of the surface layer where Monin-Obukhov similarity theory applies. In practice, if model layer 1 is more than 7.5m thick, reference values in (131) are set to layer 1 values. Otherwise, averaging is performed over depth range εd.

The surface boundary layer thickness (which is distinct from mixed layer thickness) is the depth range over which turbulent boundary layer eddies can penetrate before becoming stable relative to the local buoyancy and velocity. It is estimated as the minimum depth at which Ri_b exceeds the critical Richardson number $Ri_c = 0.3$. The Richardson number Ri_b is estimated in (131) as a layer variable, and thus assumed to represent the Richardson number at the middle depth of each layer. Moving downward from the surface, Ri_b is calculated for each layer. When the first layer is reached where $Ri_b > 0.3$, h_b is estimated by linear interpolation between the central depths of that layer and the layer above.

The unresolved turbulent velocity shear in the denominator of (131) is estimated from

$$V_t^2 = \frac{C_s \left(-\beta_T\right)^{1/2}}{Ri_c \kappa^2} \left(c_s \varepsilon\right)^{-1/2} dNw_s, \tag{132}$$

where C_s is a constant between 1 and 2, β_T is the ratio of entrainment buoyancy flux to surface buoyancy flux, $\kappa = 0.4$ is von Karman's constant, and w_s is the salinity/scalar turbulent velocity scale. The latter scale is estimated using

$$\begin{aligned} w_S &= \kappa \left(a_S u*^3 + c_S \kappa \sigma w*^3\right)^{1/3} \rightarrow \kappa \left(c_S \kappa \sigma\right)^{1/3} w* \quad \sigma < \varepsilon \\ w_S &= \kappa \left(a_S u*^3 + c_S \kappa \varepsilon w*^3\right)^{1/3} \rightarrow \kappa \left(c_S \kappa \varepsilon\right)^{1/3} w* \quad \varepsilon \le \sigma < 1 \end{aligned} \tag{133}$$

where a_x and c_x are constants, $w* = \left(-B_f/h\right)^{1/3}$ is the convective velocity scale with B_f being the surface buoyancy flux, and $\sigma = d/h_b$. Expressions to the right of the arrows represent the convective limit. In HYCOM, w_s values are stored in a two-dimensional lookup table as functions of $u*^3$ and $\sigma w*^3$ to reduce calculations.

If the surface forcing is stabilizing, the diagnosed value of h_b is required to be smaller than both the Ekman length scale $h_E = 0.7u*/f$ and the Monin-Obukhov length L.

8.1.4 Surface boundary layer diffusivity

After calculating interior diapycnal diffusivity at model interfaces and estimating h_b, surface boundary layer diffusivity/viscosity profiles are calculated at model interfaces and smoothly matched to the interior diffusivities and viscosity. Boundary layer diffusivities and viscosity are parameterized as follows:

$$\overline{w'\theta'} = -K_\theta \left(\frac{\partial\overline{\theta}}{\partial z} + \gamma_\theta\right), \quad \overline{w'S'} = -K_S \left(\frac{\partial\overline{S}}{\partial z} + \gamma_S\right), \quad \overline{w'\mathbf{v}'} = -K_m \left(\frac{\partial\overline{\mathbf{v}}}{\partial z}\right), \tag{134}$$

where γ_θ, γ_S are nonlocal transport terms. The diffusivity/viscosity profiles are parameterized as

$$K_\theta(\sigma) = h_b w_\theta(\sigma) G_\theta(\sigma), \quad K_S(\sigma) = h_b w_S(\sigma) G_S(\sigma), \quad K_m(\sigma) = h_b w_m(\sigma) G_m(\sigma) \ , \tag{135}$$

where G is a smooth shape function represented by a third-order polynomial function

$$G(\sigma) = a_0 + a_1\sigma + a_2\sigma^2 + a_3\sigma^3 \tag{136}$$

that is determined separately for each model variable. The salinity/scalar velocity scale w_S is estimated using (133). The potential temperature and momentum velocity scales w_θ, w_m are also estimated from equations analogous to (133), but with the two constants replaced by a_θ, c_θ and a_m, c_m, respectively. Since turbulent eddies do not cross the ocean surface, all K coefficients are zero there, which requires that $a_0 = 0$. The remaining coefficients of the shape function for a given variable are chosen to satisfy requirements of Monin-Obukhov similarity theory, and also to insure that the resulting value and first vertical derivative of the boundary layer K-profile match the value and first derivative of the interior ν profile for the same variable calculated using (120) through (125) and (126) through (130).

Application of this procedure is illustrated here for potential temperature only. The matching yields

$$\begin{array}{l} G_\theta(1) = \frac{\nu_\theta(h_b)}{h_b w_\theta(119)} \\ \frac{\partial}{\partial\sigma} G_\theta(1) = -\frac{\frac{\partial}{\partial z}\nu_\theta(h_b)}{w_\theta(1)} - \frac{\nu_\theta(h_b)\frac{\partial}{\partial\sigma}w_\theta(1)}{h w_\theta^2(1)} \end{array} \tag{137}$$

After determining the coefficients in (136), the K profile is calculated using

$$K_\theta = h_b w_\theta \sigma \left[1 + \sigma G_\theta(\sigma)\right], \tag{138}$$

where

$$G_\theta(\sigma) = (\sigma - 2) + (3 - 2\sigma) G_\theta(1) + (\sigma - 1) \frac{\partial}{\partial\sigma} G_\theta(1) \, . \tag{139}$$

At model interfaces within the surface boundary layer, the K profile for potential temperature is provided by (138). At model interfaces below the boundary layer, the K profile equals the interior diffusivity ($K_\theta = \nu_\theta$).

The nonlocal flux terms in (134) kick in when the surface forcing is destabilizing. The KPP model parameterizes nonlocal flux only for scalar variables. Although nonlocal fluxes may also be significant for momentum, the form that these fluxes take is presently not known. (Large *et* al., 1994). The nonlocal fluxes for scalar variables are parameterized as

$$\begin{array}{l} \gamma_\theta = 0 \quad \gamma_s = 0 \quad \zeta \geq 0 \\ \gamma_\theta = C_s \frac{\overline{w'\theta'_0} + \overline{w'\theta'_R}}{w_\theta(\sigma) h_b} \quad \gamma_S = \frac{\overline{w'S'_0}}{w_s(\sigma) h} \quad \zeta < 0 \end{array} \tag{140}$$

where ζ is a stability parameter equal to d/L and L is the Monin-Obukhov length. The terms $\overline{w'\theta'_0}$ and $\overline{w'S'_0}$ are surface fluxes while the term$\overline{w'\theta'_R}$ is the contribution of penetrating shortwave radiation.

8.1.5 Vertical mixing

The vertical diffusion equation is solved at each model grid point after model variables have been updated by the continuity equation, horizontal advection and diffusion, momentum equation, and surface fluxes, and is thus treated as a purely one-dimensional problem with zero-flux boundary conditions at the surface and bottom. In HYCOM 2.0.01, this equation is solved for the full KPP mixing algorithm, and also for the KPP-like interior diapycnal-mixing algorithm when chosen for use with the Kraus-Turner mixed layer model.

Decomposing model variables into mean (denoted by an overbar) and turbulent (denoted by a prime) components, the vertical diffusion equations to be solved for potential temperature, salinity, and vector momentum are

$$\frac{\partial\overline{\theta}}{\partial t} = -\frac{\partial}{\partial z}\overline{w'\theta'} \quad \frac{\partial\overline{S}}{\partial t} = -\frac{\partial}{\partial z}\overline{w'S'} \quad \frac{\partial\overline{\mathbf{v}}}{\partial t} = -\frac{\partial}{\partial z}\overline{w'\mathbf{v}'} \ . \tag{141}$$

Boundary layer diffusivities and viscosity are parameterized as follows:

$$\overline{w'\theta'} = -K_\theta\left(\frac{\partial\overline{\theta}}{\partial z} + \gamma_\theta\right), \quad \overline{w'S'} = -K_S\left(\frac{\partial\overline{S}}{\partial z} + \gamma_S\right), \quad \overline{w'\mathbf{v}'} = -K_m\left(\frac{\partial\overline{\mathbf{v}}}{\partial z} + \gamma_m\right), \tag{142}$$

where the γ terms represent nonlocal fluxes. For example, the KPP model includes nonlocal terms for θ and S, but not for momentum. The following solution procedure is valid for any mixing model in HYCOM that calculates the diffusivity/viscosity profiles at model interfaces, whether or not nonlocal terms are parameterized.

The following matrix problems are formulated and solved:

$$\mathbf{A_T}\Theta^{t+1} = \Theta^t + \mathbf{H}_\Theta \quad \mathbf{A_S}\mathbf{S}^{t+1} = \mathbf{S}^t + \mathbf{H_S} \quad \mathbf{A_M}\mathbf{M}^{t+1} = \mathbf{M}^t + \mathbf{H_M}, \tag{143}$$

where superscripts $t, t+1$ denote model times, and $\mathbf{M}$ is the vector of a momentum component, either u or v. The matrices $\mathbf{A}$ are tri-diagonal coefficient matrices, while the vectors $\mathbf{H_T}$ and $\mathbf{H_S}$ represent the nonlocal flux terms. Given K model layers with nonzero thickness, where an individual layer $k$of thickness δp_k is bounded above and below by interfaces located at pressures p_k and p_{k+1}, the matrix $\mathbf{A_S}$ is determined as follows:

$$\begin{array}{ll} \mathbf{A_S}^{1,1} = \left(1 + \Omega^+_{S1}\right) & \\ \mathbf{A_S}^{k,k-1} = -\Omega^-_{Sk} & 2 \le k \le \mathrm{K} \\ \mathbf{A_S}^{k,k} = \left(1 + \Omega^-_{Sk} + \Omega^+_{Sk}\right) & 2 \le k \le \mathrm{K} \\ \mathbf{A_s}^{k,k+1} = -\Omega^+_{Sk} & 1 \le k \le \mathrm{K}-1 \end{array}, \tag{144}$$

with

$$\begin{array}{l} \Omega_{Sk}^{-} = \frac{\Delta t}{\delta p_k} \frac{K_S(p_k)}{(p_{k+0.5}-p_{k-0.5})} \\ \Omega_{Sk}^{+} = \frac{\Delta t}{\delta p_k} \frac{K_S(p_{k+1})}{(p_{k+1.5}-p_{k+0.5})} \end{array} , \tag{145}$$

where $p_{k+0.5}$ represents the central pressure depth of model layer k. The nonlocal flux arrays are calculated using

$$\begin{array}{l} H_{S1} = \frac{\Delta t}{\delta p_1} K_S \left(p_{k+1}\right) \gamma_S \left(p_{k+1}\right) \\ H_{Sk} = \frac{\Delta t}{\delta p_k} \left[K_S \left(p_{k+1}\right) \gamma_S \left(p_{k+1}\right) - K_S \left(p_k\right) \gamma_S \left(p_k\right)\right] \quad 2 \leq k \leq \mathrm{K} \ . \end{array} \tag{146}$$

The solution is then found by inverting the tri-diagonal matrix **A**. After solving the equation for all variables at the pressure grid points (including velocity components interpolated from the momentum grid points), the KPP procedure is repeated beginning with equation (119) using the new profiles of all variables. The user can choose how many of these iterations are performed. In practice, two iterations are generally found to be adequate. Mixed layer thickness is diagnosed at the pressure grid points based through vertical interpolation to the depth where density exceeds the surface layer density by a prescribed amount.

After completing the mixing at pressure grid points, mixing is performed at the momentum grid points. Instead of repeating the entire KPP procedure, the K_m profiles estimated at the pressure grid points during the final iteration is horizontally interpolated to the u and v grid points, then the vertical diffusion equation is solved.

8.2 Usage

∗ INSERT TEXT HERE ∗

8.2.1 Order of Operations

∗ INSERT TEXT HERE ∗

8.2.2 Flowchart

∗ INSERT FLOWCHART HERE ∗

8.3 Variables

8.3.1 Identification

Notation in the theory	Notation in **mxkpp.f**
∗ INSERT TEXT HERE ∗	∗ INSERT TEXT HERE ∗

8.3.2 Local variables

Subroutine mxkpp

`delp`	Cushion function argument.
`i, j`	Array indices.
`k`	Layer index.
`l`	Loop index.
`m, n`	Time step indices.
`sigmlj`	

Subroutine mxkppaj, mxkppbj, and mxkppcj

`i, j`	Array indices.
`l`	Loop index.
`m, n`	Time step indices.

Subroutines mxkppaij, mxkppbij, mxkppcij, and mxkppcijv

`aa1, aa2, aa3`	
`alfadt(kdm+1)`	T contribution to density jump.
`betads(kdm+1)`	S contribution to density jump.
`bfsfc`	Surface buoyancy forcing.
`blmc(kdm+1,3)`	Boundary layer mixing coefficients.
`bref`	Nearsurface reference buoyancy.
`bvsq`	
`case`	= 1 in case A; = 0 in case B
`dat1(3)`	Derivative of shape functions at dnorm = 1.
`dbloc(kdm+1)`	Buoyancy jump across interface.
`del`	

`delta`	Fraction hbl lies beteen zgrid neighbors.
`diffdd`	Double diffusion diffusivity scale.
`diffm(kdm+1)`	
`diffs(kdm+1)`	
`difft(kdm+1)`	
`difsh`	
`difsp`	
`difth`	
`diftp`	
`dkm1(3)`	Boundary layer diffusions at nbl-1 level.
`dkmp2`	
`dnorm`	Normalized depth.
`dsaln`	Variation ΔS of salinity in the mixed layer.
`dstar`	
`dtemp`	Variation ΔT of Temperature in the mixed layer.
`dvdzdn`	
`dvdzup`	
`dvsq(kdm)`	Squared current shear for bulk Richardson number.
`dzb(kdm)`	
`f1`	Function of Rig.
`fri`	Function of Rig.
`gat1(3)`	Shape functions at dnorm = 1.
`ghat(kdm+1)`	
`ghatflux`	
`gm`	
`gs`	

`gt`	
`hbl`	Boundary layer depth.
`hblmax`	
`hblmin`	
`hm(kdm)`	
`hmonob`	
`hwide(kdm)`	Layer thicknesses in m.
`i, j`	Array indices.
`iter`	Iteration loop index.
`jrlv`	
`k, ka, kb`	Layer index.
`k1, k2`	Bulk Richardson number indices.
`kmask`	
`ksave`	
`m, n`	Time step indices.
`nbl`	Layer containing boundary layer base.
`nlayer`	
`prandtl`	Prandtl number.
`presu, presv`	
`q`	
`ratio`	
`rhs(kdm)`	Right-hand-side terms.
`rib(2)`	Bulk Richardson number.
`rigr`	Local Richardson number.
`ritop(kdm)`	Numerator of bulk Richardson number.
`rrho`	Double diffusion parameter.

`s1dn(kdm+1)`	
`s1do(kdm+1)`	
`sflux1`	
`shsq(kdm+1)`	Velocity shear squared.
`sigg`	
`sold(kdm+1)`	Old salinity.
`stable`	= 1 in stable forcing; = 0 in unstable.
`swfrac(kdm+1)`	Fractional surface shortwave radiation flux.
`swfrm1`	Fractional surface sw rad flux at ml base.
`t1dn(kdm+1)`	
`t1do(kdm+1)`	
`tcc(kdm)`	Central coefficient for (k-1) on k line of tridiagonal matrix.
`tcl(kdm)`	Lower coefficient for (k-1) on k line of tridiagonal matrix.
`tcu(kdm)`	Upper coefficient for (k-1) on k line of tridiagonal matrix.
`thold(kdm+1)`	
`told(kdm+1)`	Old temperature.
`tr1dn(kdm+1)`	
`tr1do(kdm+1)`	
`tri(kdm,0:1)`	Dt/dz/dz factors in a tridiagonal matrix.
`u1dn(kdm+1)`	
`u1do(kdm+1)`	
`uold(kdm+1)`	
`uref, vref, zref`	Nearsurface reference u, v, z.
`v1dn(kdm+1)`	
`v1do(kdm+1)`	
`vctyh`	

`visep`	
`vold(kdm+1)`	
`vtsq`	
`wm`	Momentum velocity scale.
`ws`	Scalar velocity scale.
`wz`	
`zm(kdm+1)`	

Subroutine wscale

`bfsfc`	Surface buoyancy forcing.
`dnorm`	Normalized depth.
`i, j`	Array indices.
`iz`	
`izp1`	
`ju`	
`jup1`	
`nustar`	
`nzehat`	
`ucube`	
`udiff`	
`ufrac`	
`wam`	
`was`	
`wbm`	
`wbs`	
`wm`	Momentum velocity scale.

`wmt`	Momentum velocity scale table.
`ws`	Scalar velocity scale.
`wst`	Scalar velocity scale table.
`zdiff`	
`zehat`	
`zfrac`	
`zlevel`	

8.4 Procedures

Subroutines `mxkpp, mxkppaj, mxkppbj, mxkppcj, mxkppaij, mxkppbij, mxkppcij, mxkppcijv, wscale`

9 Generalized Vertical Coordinates : hybgen.f

9.1 Formalism and numerical techniques

The generalized vertical coordinate adjustment algorithm implemented in HYCOM 2.0.01 is designed so that the isopycnic vertical coordinates present in the ocean interior transition smoothly to z coordinates in nearsurface, well-mixed regions, to sigma (terrain-following) coordinates in shallow water regions, and back to level coordinates in very shallow water to prevent layers from becoming too thin.

9.1.1 Vertical coordinate remapping

The HYCOM vertical coordinate adjustment in the open ocean works as follows: Let $\alpha_k > \alpha_{k+1}$ where α is specific volume. Interfaces are labeled such that interface k is the upper interface of layer k. Consider three consecutive isopycnic layers labeled 0, 1, and 2 in a stratified water column. Suppose that α_1 differs from its isopycnic reference value $\widehat{\alpha}_1$. To restore isopycnic conditions, it is necessary to re-discretize the water column in a manner that preserves the overall height of the column, represented by the integral $\int \alpha dp$, while changing α_1 to $\widehat{\alpha}_1$. Conservation of the integral requires that one or more layer interfaces must be relocated to different pressure levels. To adjust the density in layer 1, one interface will be moved in the spirit of the "donor cell" transport scheme. If layer 1 is too dense ($\alpha_1 < \widehat{\alpha}_1$), the upper interface is moved upward to transfer less-dense water from layer 0 to layer 1. If layer 1 is too light ($\alpha_1 > \widehat{\alpha}_1$), the lower interface is moved downward to transfer denser water from layer 2 to layer 1. This method does not work for the model layer in contact with the bottom if it is too light. A special algorithm was included to handle this case by extruding water into the layer above.

In HYCOM, the user specifies n_{hyb}, the number of hybrid layers. Interfaces bounding the upper n_{hyb} layers are adjusted according to the algorithm described below. Deeper layers are allowed to remain isopycnic.

Case 1: $\alpha_1 > \widehat{\alpha}_1$

This is the case where the upper interface is moved and mass is exchanged between layers 0 and 1. Conservation of $\int \alpha dp$ requires

$$\alpha_0(p_1 - p_0) + \alpha_1(p_2 - p_1) = \alpha_0(\widehat{p}_1 - p_0) + \widehat{\alpha}_1(p_2 - \widehat{p}_1), \tag{147}$$

where $\widehat{p_1}$ is the pressure of the upper interface after re-discretization. Solving (147) for $\widehat{p_1}$ yields the expression

$$\widehat{p}_1 = \frac{p_1(\alpha_0 - \alpha_1) + p_2(\alpha_1 - \widehat{\alpha}_1)}{\alpha_0 - \widehat{\alpha}_1}. \tag{148}$$

Since the weight assigned to p_2 is negative, (148) will not necessarily yield a solution $\widehat{p}_1 > p_0$ for large differences between α_1 and $\widehat{\alpha}_1$. To maintain the minimum thickness of layer 0, $\widehat{p}_1$ is replaced by

$$\widetilde{p}_1 = \max(\widehat{p}_1, p_0 + \Delta_0), \tag{149}$$

where Δ_0 is a specified minimum layer thickness. Of course, moving the interface to $\widetilde{p}$ instead of $\widehat{p}$ no longer permits isopycnic conditions to be restored.

Following Bleck and Benjamin (1993), the user chooses the absolute minimum thickness δ_0. The actual minimum thickness Δ_0 is then calculated in a manner that insures a smooth transition between the isopycnic and non-isopycnic domains. The function Δ_0 is determined by a continuously differentiable "cushion" function, which for large positive arguments $\Delta p \equiv \widehat{p_1} - p_0$ returns the argument Δp(meaning that $\widetilde{p}_1 = \widehat{p}_1$) and for large negative arguments returns δ_0:

$$\Delta_0 = \begin{cases} \delta_0 \left[1 + \left(\frac{\Delta p}{3\delta_0}\right) + \left(\frac{\Delta p}{3\delta_0}\right)^2\right] & if \quad \Delta p \leq 3\delta_0 \\ \Delta p & if \quad \Delta p > 3\delta_o \end{cases} \tag{150}$$

In practice, if temperature and salinity are fluxed across the relocated interfaces, then perfect restoration of isopycnic conditions is not possible ($\widetilde{\alpha_1} \neq \widehat{\alpha_1}$). In rare instances, the change in α_1 will not even have the expected sign. For example, in model layers just beneath the Mediterranean salt tongue, cases were observed where raising interface 1 unexpectedly decreased α_1. Repeated application of the vertical coordinate adjustment algorithm then acts to drive α_1 farther from its reference value and produces unacceptably large vertical coordinate migration. This problem can be exacerbated when temperature and salinity are advected. Code was included to suppress the vertical coordinate adjustment when the change in α_1 does not have the expected sign. Although this prevents α_1 from diverging rapidly from its reference value, restoration to its reference value is not possible. The only way to completely avoid this problem is to adjust temperature and density, or salinity and density.

Even without this problem, substantial deviations from layer reference density can still occur when temperature and salinity are adjusted. An iterative procedure was therefore included to insure in most cases that $|\widetilde{\alpha}_1 - \widehat{\alpha}_1|$ is smaller than a prescribed tolerance after applying the vertical coordinate adjustment algorithm. This iterative procedure works as follows: After $\widehat{p}_1$ is estimated from (148), changes in temperature and salinity resulting from fluxes across the relocated vertical coordinate are estimated, then $\widetilde{\alpha}_1$ is estimated from the model equation of state. If it is not within the required tolerance, then $\widehat{p}_1$ and $\widetilde{\alpha}_1$ are again updated and the tolerance test applied. A maximum of five iterations is performed. Specific details are provided later.

Case 1 has two sub-cases that must be treated separately:

Case 1a: $\widetilde{p}_1 < p_1$

In this case, the upper interface is raised, allowing lighter water to increase α_1 to a value $\widetilde{\alpha}_1$ that is closer, but not necessarily equal to, the reference value $\widehat{\alpha}_1$. The estimate of $\widetilde{\alpha}_1$ is obtained by substituting tildes for carats in (147):

$$\widetilde{\alpha}_1 = \frac{\alpha_1(p_2 - p_1) + \alpha_0(p_1 - \widetilde{p}_1)}{p_2 - \widetilde{p}_1}. \tag{151}$$

If the user has chosen to flux α and salinity or α and temperature, further modification of α is unnecessary. If the user has chosen to mix temperature and salinity, however, the next step is to determine if the actual change in α will have the expected sign, and if $|\widetilde{\alpha}_1 - \widehat{\alpha}_1|$ is within the desired tolerance. The new temperature value that will result from raising the interface is

$$\widetilde{T}_1 = \frac{p_1 - \widehat{p}_1}{p_2 - \widehat{p}_1}(T_0 - T_1). \tag{152}$$

The new salinity $\widetilde{S}_1$ is estimated in the same manner, and then a new estimate of $\widetilde{\alpha}_1$ is made using the model equation of state. If the change in α_1 has the wrong sign, then p_1 is not adjusted. Otherwise, If $|\widetilde{\alpha}_1 - \widehat{\alpha}_1|$ exceeds the required tolerance, then $\widehat{p}_1$ is recalculated using

$$\widehat{p}_1 = \widehat{p}_1 + \frac{\widetilde{\alpha}_1 - \widehat{\alpha}_1}{\alpha_0 - \widehat{\alpha}_1}(p_2 - \widehat{p}_1) \tag{153}$$

In this step, $\widehat{p}_1$ is not permitted to exceed p_1. The resulting specific volume $\widetilde{\alpha}_1$ is then re-calculated, and if the prescribed tolerance is not achieved, then $\widehat{p}_1$ is re-calculated by the same procedure. The procedure is repeated up to five times if necessary. If the upward relocation of interface 1 is sufficiently limited by an interior blocking layer above, then the special algorithm to break this logjam is invoked.

Case 1b: $\widetilde{p}_1 > p_1$

In this case, the upper interface must be lowered to preserve the minimum thickness of layer 0. Conservation of $\int \alpha dp$ then leads to

$$\widetilde{\alpha}_0 = \frac{\alpha_1(\widetilde{p}_1 - p_0) + \alpha_0(p_1 - p_0)}{\widetilde{p}_1 - p_0}. \tag{154}$$

The preservation of minimum layer thickness by increasing the thickness of layer 0 overrides all attempts to restore isopycnic conditions. Consequently, none of the special algorithms described for case 1a are invoked.

Case 2: $\alpha_1 > \widehat{\alpha}_1$

This is the case where the lower interface is moved downward and mass from layer 2 is entrained into layer 1. Conservation of $\int \alpha dp$ requires

$$\alpha_1(p_2 - p_1) + \alpha_2(p_3 - p_2) = \widehat{\alpha}_1(\widehat{p}_2 - p_1) + \alpha_2(p_3 - \widehat{p}_2). \tag{155}$$

The lower interface must be relocated to

$$\widehat{p}_2 = \frac{p_1(\widehat{\alpha}_1 - \alpha_1) + p_2(\alpha_1 - \alpha_2)}{\widehat{\alpha}_1 - \alpha_2}. \tag{156}$$

To maintain the minimum thickness of layer 2 $\widehat{p}_2$ is replaced by

$$\widetilde{p}_2 = \min(\widehat{p}_2, p_3 - \Delta_2), \tag{157}$$

where Δ_2 is determined from the cushion function

$$\Delta_2 = \begin{cases} \delta_2 \left[1 + \left(\frac{\Delta p}{3\delta_2}\right) + \left(\frac{\Delta p}{3\delta_2}\right)^2\right] & if \quad \Delta p \leq 3\delta_2 \\ \Delta p & if \quad \Delta p > 3\delta_2 \end{cases} \tag{158}$$

with $\Delta p \equiv p_3 - \widehat{p}_2$. The iterative algorithm to improve restoration of isopycnic conditions is also executed for this case when temperature and salinity are adjusted. This algorithm is executed throughout the water column with the exception of the deepest layer with nonzero thickness, which intersects the bottom. The following special case is executed for this layer.

Case 2a: $\alpha_1 > \widehat{\alpha}_1$, Where Layer 1 is the Bottom Layer

Since interface 2 cannot move downward in this case, it is necessary to use constraint (147) and move interface 1 downward to restore isopycnic conditions in layer 1. To achieve this goal, the water in layer 1 must be restratified into two sublayers such that the density of the upper sublayer exactly equals the density of layer 0, the density of the lower sublayer is close to the desired reference density, and the density averaged over the two sublayers equals the original layer density. Given

$$q = \frac{\alpha_1 - \widehat{\alpha}_1}{\alpha_0 - \widehat{\alpha}_1},$$

interface 1 is relocated using

$$\widehat{p}_1 = p_1 + q(p_2 - p_1). \tag{159}$$

Thermodynamical variables in the lower sublayer are then calculated using

$$\widehat{T}_1 = T_1 - \left(\frac{q}{1-q}\right)(T_1 - T_0) \tag{160}$$

for temperature, and using the analogous equation for salinity and density. Again, two of the thermodynamical variables are diagnosed using (160) with the third estimated from the equation of state. The closeness of lower sublayer density to the reference density is sacrificed if necessary to achieve two goals: 1) to prevent the thickness of layer 1 from decreasing by more than 50% using

$$\widetilde{p}_1 = \min(\widehat{p}_1, p_2 - \Delta_2), \tag{161}$$

where $\Delta_2 = (p_2 - p_1)/2$, and 2) to prevent runaway changes in temperature and salinity using

$$|T_1 - T_0| \leq |T_0 - T_{-1}| \tag{162}$$

for temperature and the analogous equation for salinity.

9.1.2 Adjustment of vertical coordinates in shallow water regions

A very simple scheme is used to insure the transition from the open ocean, where a non-isopycnic coordinate domain exists above an isopycnic coordinate ocean interior, to a sigma coordinate domain in shallow water regions and a level coordinate domain in very shallow regions. When the user specifies the minimum thickness δ_n for each model layer n at grid points where the ocean is sufficiently deep, the coordinate adjustment algorithm produces the non-isopycnic z coordinate domain overlying an isopycnic domain as described previously. Without changes in this algorithm, first the isopycnic coordinates, then the non-isopycnic coordinates, will intersect the bottom towards shallower water an not provide optimum vertical resolution in shallow water regions.

To insure the proper coordinate transition toward shallower water, the user specifies the number of model layers N_s that are to become sigma coordinates along with the absolute minimum thickness $\delta_{\min}$ that is permitted for the sigma coordinates. The minimum thickness of sigma coordinates is given by

$$\delta_s = \frac{D}{N_s} \tag{163}$$

where D is the total water depth. A new minimum thickness for each model layer is then calculated using

$$\widetilde{\delta}_n = \max\left[\delta_{\min}, \min\left(\delta_n, \delta_s\right)\right]. \tag{164}$$

In a given model layer, the transition to sigma coordinates occurs where the water depth becomes sufficiently shallow to make $\delta_s < \delta_n$. The transition back to level coordinates occurs where the water depth becomes sufficiently shallow to make $\delta_s < \delta_{\min}$. Thus, the proper coordinate transformation is assured if δ_n is replaced by $\widetilde{\delta_n}$ before executing the vertical coordinate adjustment.

9.1.3 Adjustment of temperature, salinity, density, and momentum

The adjustment of thermodynamical variables and momentum ideally must conserve their vertically averaged values and, for the thermodynamical variables, restore density as close to the isopycnic reference value as possible. The algorithm included in HYCOM remaps each variable from the old to the new vertical grid. It satisfies these two conditions, and furthermore does not depend on the direction (top to bottom or bottom to top) in which it is executed, in contrast to a pure donor cell scheme.

The user has the option of adjusting any two of the thermodynamical variables and diagnosing the third using the equation of state. When the hybrid coordinate generator is called, it is executed separately at each grid point. Thermodynamical variables are adjusted first at the pressure grid points. Before adjusting the vertical coordinates, the initial one-dimensional profiles of temperature, salinity, and density, plus the one-dimensional array of interface pressures, are saved. If the full Kraus-Turner mixed layer model is selected, the model layer containing the mixed layer base is divided into two sublayers, and the mixed layer base is temporally considered to be an additional vertical coordinate. The sublayers above and below the mixed layer base are thus temporarily considered to be two model layers. Thermodynamical variables in the two sublayers are estimated using the "unmixing" algorithm described in the Kraus Turner Model Section.

Once the profiles are saved, the vertical adjustment of vertical coordinates is performed at the pressure grid points as outlined in the previous sections. Each variable is then remapped from the old to the new vertical coordinates as illustrated here for temperature. The interface pressures on the old grid are $p_m, m = 1, M$ while the pressures on the new adjusted grid are $\widetilde{p}_n, n = 1, N$, where N is the number of model layers. Note that $M = N$ unless the full Kraus-Turner mixed layer model is used, in which case $M = N + 1$ to account for the two sublayers within the layer containing the mixed layer base. The old temperature profile is mapped onto the new adjusted vertical coordinates using

$$\widetilde{T}_n = \frac{1}{\widetilde{p}_{n+1} - \widetilde{p}_n} \sum_{m=1}^{M} T_m \left\{ \max\left[\widetilde{p}_n, \min\left(p_{m+1}, \widetilde{p}_{n+1}\right)\right] - \min\left[\widetilde{p}_{n+1}, \max\left(p_m, \widetilde{p}_n\right)\right] \right\}. \quad (165)$$

In practice, the summation is performed between $m = m_1$ to $m = m_2$ in order to eliminate layers on the old vertical grid that do not overlap layer n on the new grid.

After adjusting the thermodynamical variables, the momentum components are adjusted to assure that vertically-averaged momentum is conserved at each grid point. The old and new vertical coordinates obtained at pressure grid points are first interpolated to the u grid points. The adjustment of u is performed using an equation of the form (165). The same procedure is used to update v at the v grid points.

9.1.4 Running HYCOM with isopycnic vertical coordinates (MICOM Mode)

To facilitate the comparison between HYCOM and MICOM, the capability of running the HYCOM code to mimic MICOM was built into the code. If MICOM mode is selected, the hybrid vertical coordinate adjustment is not performed at all. Vertical mixing is then performed using the same Kraus-Turner mixed layer model and the same interior diapycnal mixing algorithm that were included in MICOM version 2.8. Advection and diffusion operate on density and salinity in layer 1 (the slab mixed layer), and only on salinity (with temperature diagnosed from the equation of state) in other layers. The user can therefore avoid having to perform the conversions (mesh and units) that would be required to compare HYCOM to MICOM version 2.8.

9.2 Usage

∗ INSERT TEXT HERE ∗

9.2.1 Order of Operations

∗ INSERT TEXT HERE ∗

9.2.2 Flowchart

∗ INSERT FLOWCHART HERE ∗

9.3 Variables

9.3.1 Identification

Notation in the theory	Notation in **hybgen.f**
∗ INSERT TEXT HERE ∗	∗ INSERT TEXT HERE ∗

9.3.2 Local variables

Subroutine hybgen

`i, j`	Array indices.
`k`	Layer index.
`l`	Loop index.
`lpipe_hybgen`	Flag to compare two model runs.
`m, n`	Time step indices.
`text`*12	Comparison title.

Subroutine hybgenaj

`cusha, cushb`	Cushion function constant.
`cushn`	Cushion function.

`delp`	Cushion function argument.
`dels, delsm`	Change in salinity.
`delt, deltm`	Change in temperature.
`dp0`	Cushion function argument.
`dp0cum(kdm+1)`	
`i, j`	Array indices.
`iter`	Iteration loop index.
`k, k1, kp`	Layer indices.
`ksubl`	Layer containing the mixed layer base.
`l`	Loop index.
`m, n`	Time step indices.
`p_hat, p_hat0,`	
`p_hat2, p_hat 3`	P hat.
`pres(kdm+2)`	Pressure profile.
`psum, pwidth`	Total layer thickness.
`q`	Scale factor.
`qq`	Internal function for cushn.
`qqmn, qqmx`	qq function constant.
`qts`	Split layer scale factor.
`rpsum`	1/psum
`ssal(kdm+1)`	Old salinity profile.
`ssum`	Salinity profile sum.
`thnew`	New density.
`thsum`	Density profile sum.
`trsum`	Tracer profile sum.
`tsum`	Temperature profile sum.

`ttem(kdm+1)`	Old temperature profile.
`tthe(kdm+1)`	Old density profile.
`ttrc(kdm+1)`	Old tracer profile.

Subroutine hybgenbj

`i, j`	Array indices.
`k, k1, kp`	Layer indices.
`ksubl`	Layer containing the mixed layer base.
`l`	Loop index.
`m, n`	Time step indices.
`pres(kdm+2)`	Pressure profile.
`psum, pwidth`	Total layer thickness.
`q`	Scale factor.
`thknss`	Layer thickness.
`usum`	U-velocity profile sum.
`uu(kdm+1)`	U-velocity profile.
`vsum`	V-velocity profile sum.
`vv(kdm+1)`	V-velocity profile.

9.4 Procedures

Subroutines `hybgen`

10 Kraus-Turner Mixed Layer Model : mxkrta.f or mxkrtb.f

10.1 Formalism and numerical techniques

The Kraus-Turner mixed layer is a vertically homogenized slab of water whose depth is diagnosed from the turbulence kinetic energy (TKE) equation converted into a diagnostic equation by setting the time-dependent term to zero; *i.e.*, by assuming a balance between sources and sinks of TKE in the water column. This model has been incorporated in MICOM for many years by designating the uppermost model layer to be a non-isopycnic slab layer. The K-T model was incorporated into HYCOM to facilitate comparisons with MICOM, and to serve as a benchmark for evaluating new mixed layer models, as they are included.

Since the K-T model only governs mixing within the surface mixed layer, the user must also use one of the interior diapycnal mixing algorithms provided in HYCOM (Section 11).

10.1.1 Full K-T model (hybrid coordinates with unmixing)

The greatest difficulty in incorporating a Kraus-Turner mixed layer model within a hybrid coordinate ocean model is to properly handle the mixed layer base (MLB), the depth of which is a model prognostic variable. Since the MICOM slab mixed layer is identically layer 1, the MLB always coincides with a model vertical coordinate. This is not true in a hybrid coordinate model, so special bookkeeping is required to keep track of MLB depth along with discontinuities in thermodynamical and dynamical variables that occur at the MLB. The buoyancy change across the MLB must be known to estimate the contribution of entrainment to the TKE balance, and the magnitude of the discontinuity in a given property must be known to calculate changes in the value of that property within the mixed layer caused by entrainment.

In hybrid coordinates, the model layer containing the MLB, here assumed to be layer k, can be divided into one sublayer above the MLB that is part of the mixed layer and one sublayer below that is part of the stratified ocean interior. Water properties must be known in both sublayers to quantify discontinuities across the MLB. Since the basic model code tracks only vertically averaged properties within model layers, an algorithm has been added to artificially create estimates of water properties within the two sublayers by "unmixing" the water column. During the development and testing of the unmixing scheme, it became clear that it had to be designed to reduce as much as possible numerically induced property exchanges between the mixed layer and the deeper ocean. This point is illustrated using mixed layer temperature. This temperature equals the average temperature between the surface and the MLB, and is thus the average over model layers 1 through $k-1$ plus the upper sublayer of layer k. If the estimate of upper sublayer temperature generated by the unmixing algorithm is not accurate, neither the mean mixed layer temperature nor the lower sublayer temperature beneath the MLB will be accurate. An inaccurate unmixing algorithm will thus lead to an erroneous flux of temperature across the MLB. In test simulations of the Atlantic Ocean, it was found that model performance was very sensitive to unmixing errors, particularly at high latitudes. Substantial effort was invested in making the unmixing algorithm as accurate as possible, which eventually improved the realism of model simulations in the high latitude North Atlantic.

Between individual implementations of K-T mixing, water properties in layer k change due to processes such as horizontal advection, diffusion, and penetrating shortwave radiation. With previous sublayer information lost, it is impossible to generate perfect estimates of upper and lower sublayer variables that do not lead to erroneous property fluxes across the MLB. To reduce this problem as much as possible, the layer number k containing the mixed layer base, upper sublayer variables, and the averaged variables within model layer k are saved at the end of the Kraus-Turner mixed layer algorithm. The next time the K-T algorithm is called, the previously saved values of these variables are used to make a best-guess of upper sublayer variables during the unmixing process as illustrated below.

This Kraus-Turner model is implemented as follows: Thermodynamical variables are mixed first at the pressure grid points. A search is conducted to determine the model layer k that contains the mixed layer base. Temperature and salinity are then averaged over layers 1 through $k-1$. Before proceeding further, convection is performed if necessary. The density associated with the averaged values of temperature and salinity is calculated, and if it is greater than the density of layer k, the MLB is moved to the base of layer k (interface k+1). Temperature and salinity are then averaged from the surface through layer k, and if the resulting density is greater than the density of layer $k+1$, the MLB is moved down to interface $k+2$. This process is repeated until a layer with greater density than the mixed layer is encountered. If convection occurs, the MLB will reside on a model interface, so no unmixing is required. In practice, whenever the MLB is located within one centimeter of a model interface, it is moved there and no unmixing is required.

If the MLB is still located within a model layer, the unmixing algorithm is performed. If $k \neq \widehat{k}$, where $\widehat{k}$ is the model layer that contained the MLB during the previous time step, then the following first guesses are made for upper sublayer temperature and salinity:

$$T_{up} = T_{k-1}, \quad S_{up} = S_{k-1}. \tag{166}$$

If $k = \widehat{k}$, the following first guesses are made:

$$T_{up} = \widehat{T}_{up} + T_k - \widehat{T}_k, \quad S_{up} = \widehat{S}_{up} + S_k - \widehat{S}_k \ . \tag{167}$$

The upper sublayer variables are therefore assumed to have changed by an amount equal to the change that occurred in the full layer k variables. Lower sublayer variables are then estimated using

$$T_{dn} = \frac{p_m - p_k}{p_{k+1} - p_m} \left(T_k - T_{up}\right) \tag{168}$$

and

$$S_{dn} = \frac{p_m - p_k}{p_{k+1} - p_m} \left(S_k - S_{up} \right), \tag{169}$$

where p_m is the pressure level of the MLB. Spurious extrema of T and S are prevented by requiring that T_{dn} be within the envelope of values defined by (T_{up}, T_k, T_{k+1}) and S be within the envelope of values defined by (S_{up}, S_k, S_{k+1}). If T_{dn} has to be adjusted, then T_{up} is recomputed using

$$T_{up} - T_{up} + \frac{p_{k+1} - p_m}{p_m - p_k} \left(T_{dn} - T_k \right). \tag{170}$$

If S_{dn} has to be adjusted, then S_{up} is recomputed using

$$S_{up} - S_{up} + \frac{p_{k+1} - p_m}{p_m - p_k} \left(S_{dn} - S_k \right). \tag{171}$$

After the unmixing is completed, the density profile is provided to the K-T TKE algorithm to calculate the new mixed layer depth. There are two possibilities here: First, density could be averaged over the mixed layer to provide a homogeneous slab mixed layer profile with a discontinuity at the mixed layer base. Second, the unmixed density profile above the mixed layer base (including the upper sublayer) could be provided to the K-T TKE algorithm. This turns out to be a significant consideration. Since the previous time step, differential advection and diffusion within the mixed layer results in an inhomogeneous profile in the mixed layer. Homogenizing the mixed layer prior to calling the K-T TKE algorithm then alters the energetics of the mixed layer, and generally leads to a different MLB depth being calculated by the K-T algorithm. Tests conducted in the Atlantic Ocean revealed that providing the inhomogeneous profile to the K-T algorithm improved the realism of the simulated fields, in particular at high latitudes.

Temperature and salinity are homogenized over the maximum of the old and new mixed layer depths, and the surface fluxes are distributed over the new depth. The final step is to store the layer number k containing the mixed layer base, the upper sublayer temperature and salinity, and the layer k average temperature and salinity to be used as the old values the next time the K-T algorithm is executed.

Mixing is then performed for momentum components on the u and v grid points. Momentum is mixed from the surface to the maximum of the old and new mixed layer thickness, denoted by $\widetilde{p}_m$, that is interpolated from pressure grid points. If $\widetilde{p}_m$ is within one centimeter of a model interface $k+1$ at a u grid point, then u is homogenized from the surface through layer k. If the MLB is located in layer k, unmixing is performed. It was found that model simulations are not sensitive to the accuracy of the estimates of u and v in the two sublayers. For both u and v, the first guess for the upper sublayer value is the value in layer $k-1$.

The present model includes optional penetrating shortwave radiation as described in Section 8.1.1. If this option is not selected, all shortwave radiation is assumed to be absorbed in the mixed layer.

Another important consideration is that the mixed layer base is a material surface that can be advected by the flow field. This advection is partly accounted for by an algorithm added to the model continuity equation. If the MLB is contained within layer k, and model interfaces k and $k+1$ are adjusted vertically during solution of the continuity equation by δp_k and δp_{k+1}, respectively, the Kraus-Turner prognostic mixed layer base located at p_m is adjusted vertically by

$$\delta p_m = \frac{p_{k+1} - p_m}{p_{k+1} - p_k}\delta p_k + \frac{p_m - p_k}{p_{k+1} - p_k}\delta p_{k+1}. \tag{172}$$

The vertical motion at the MLB is assumed to be the linearly interpolated value between model interfaces k and $k+1$. It is also necessary to add the vertical motion resulting from time smoothing of pressure interfaces, so the interface adjustments δp_k and δp_{k+1} in (172) represent the sum of the continuity and time smoothing adjustments. As a result of this vertical adjustment, a mixed layer base located at a model interface would always remain at the depth of this interface if no vertical mixing or diabatic heating/cooling occurs.

Additional vertical motion of the mixed layer base is possible, however, because the MLB can slope relative to the model vertical coordinates. The flow component normal to the MLB can therefore produce additional vertical motion. This part of the advection is not included, however, due to significant ambiguities in determining the flow field at the MLB in hybrid coordinates.

10.1.2 Simplified K-T model (hybrid coordinates without unmixing)

Due to the complexities of devising an unmixing scheme that reduces numerically induced property exchange between the mixed layer and ocean interior, a simplified alternative K-T model was developed by relaxing the requirement that the MLB be a prognostic variable. The tradeoff is between improved computational efficiency and an increase in numerical errors. Since the full K-T model 1 is far from perfect, this simplified model may be adequate for many purposes. In tests conducted in the Atlantic basin, use of this simplified model caused some problems at high latitudes, where unrealistic patterns of deep water formation occurred and unrealistic pathways were observed for the Gulf Stream and North Atlantic Current. In tropical and subtropical latitudes with relatively strong stratification, this simplified model clearly performed "as well" as the full model 1.

The K-T algorithm computes the change of mixed layer thickness over a small time interval Δt, and thus requires the old mixed layer information as the initial condition. This depth, which can be viewed as the cumulative result of past applications of the K-T algorithm at a given point, can either be carried as a prognostic variable (as in K-T model 1 above) or diagnosed *ad-hoc*. The present scheme uses option 2 to avoid the necessity of unmixing, and to avoid the ambiguities of constructing a representative mixed layer velocity field to advect the mixed layer base (which as a material variable needs to follow the flow).

Beginning at the top of the water column (layer 1), the algorithm searches for the first layer k whose density exceeds the average density $\overline{\rho}$ of the overlying layers combined (layers 1 through $k-1$). This layer is assumed to contain the mixed layer base. Layer k is divided into sublayers with different densities. The density ρ_{up} of the upper sublayer is set equal to $\overline{\rho}$ while the density ρ_{lo} of the lower sublayer is assigned a value within the range $(\rho_k + \rho_{k+1})$, for example $\rho_{lo} = (\rho_k + \rho_{k+1})/2$. Density conservation during sublayer formation yields the depth of the interface separating the sublayers, which is taken to be the mixed layer depth. This pressure interface is given by

$$p_m = p_k(1-q) + p_{k+1}q \tag{173}$$

where p_k, p_{k+1} are the upper and lower interface pressures of layer k, and

$$q = \frac{\rho_{lo} - \rho_k}{\rho_{lo} - \overline{\rho}}. \tag{174}$$

Assignment of temperature/salinity values for the depth interval $(0, p_{k+1})$ follows the above philosophy of assuming homogeneity between the surface and p_m. With the mixed layer depth already determined by (173), lower sublayer values of temperature and salinity follow from the requirement that their column integrals be invariant during the redistribution process. Specifically, we set $T_{up} = \overline{T}$ and $S_{up} = \overline{S}$, where the overbar again indicates the average over layers k through $k-1$. The lower sublayer values become

$$T_{lo} = \frac{T_k - \overline{T}q}{1-q}, \quad S_{lo} = \frac{S_k - \overline{S}q}{1-q} \ , \tag{175}$$

with q defined by (174).

Given the nonlinear nature of the equation of state, the upper and lower sublayer densities must be recomputed after temperature and salinity have been homogenized over layers 1 through $k-1$ and unmixed in layer k. After this, the density profile, characterized by homogeneous conditions between the surface and pressure p_m and a density discontinuity at that pressure, is provided to the K-T TKE algorithm to calculate the new mixed layer depth. Temperature and salinity are homogenized over the new mixed layer depth, and the surface fluxes are distributed over the same depth range. The resulting temperature and salinity profiles are then projected back onto the original hybrid coordinate profile. The sublayer information is not saved.

It is important to note that the sublayer formation and deletion cycle by itself does not cause a drift in the T/S profile provided that layers 1 through $k-1$ are fully homogenized to begin with. In other words, the original T/S profile is recovered if $\Delta t \to 0$. Initial experiments with this scheme indicate that best results are obtained by assigning ρ_{lo}a value closer to ρ_k than to ρ_{k+1}. To skip the necessity of forming sublayers, set $\rho_{lo} = \rho_k$ (i.e., $q = 0$). The T/S profile transmitted to the K-T TKE algorithm in this case is obtained from the original profile by homogenizing temperature and salinity down to depth p_k and setting $p_m = p_k$.

10.2 Usage

∗ INSERT TEXT HERE ∗

10.2.1 Order of Operations

∗ INSERT TEXT HERE ∗

10.2.2 Flowchart

∗ INSERT FLOWCHART HERE ∗

10.3 Variables

10.3.1 Identification

Notation in the theory	Notation in **mxkrta.f or mxkrtb.f**
∗ INSERT TEXT HERE ∗	∗ INSERT TEXT HERE ∗

10.3.2 Local variables

Subroutines mxkrta and mxkrtb

`depnew`	
`i, j`	Array indices.
`k`	Layer index.
`l`	Loop index.
`m, n`	Time step indices.
`tndcys`	
`tndcyt`	
`tosal`	
`totem`	

Subroutines mxkrtaaj and mxkrtbaj

`alf1`	Stability parameter h/l.
`alf2`	Stability parameter in the absence of rotation h/l_p.

`ape`	Term in the expression of entrainment.
`bound1`	
`bound2`	
`cc4`	Intermediate coefficient parameterizing turbulent effects.
`cp1, cp3`	Coefficients in the expression of the entrainment.
`delp(kdm)`	Cushion function argument.
`dens(kdm)`	
`depnew`	
`dp1, dp2`	
`dpth`	Transformation of the pressure difference in the mixed layer in thickness units (cm).
`dsaln(idm)`	Variation ΔS of salinity in the mixed layer.
`dtemp(idm)`	Variation ΔT of temperature in the mixed layer.
`dtemp2`	
`dtrmax`	
`ea1, ea2`	
`ekminv`	Inverse of the Ekman length.
`em1,...,em5`	Coefficients of the turbulent effects parameterization.
`ex`	Exponential of the Monin-Obukov stability parameter h/L.
`i, j`	Array indices.
`k, k0, k1`	Layer indices.
`kmxbot`	
`l`	Loop index.
`m, n`	Time step indices.
`obuinv`	Inverse of the Monin-Obukov length.
`pres(kdm+1)`	Pressure profile.

`q`	Relative variation of momentum in the mixed layer due to its deepening.
`rho`	
`s1, s2`	
`sal`	
`sdp(idm)`	
`sflux1`	
`smax`	
`smin`	
`spe`	Term in the expression of the entrainment.
`ssal(kdm)`	Old salinity profile.
`sum1, sum2`	
`swfold`	
`swfrac`	
`t1, t2`	
`tdp(idm)`	
`tem`	
`thet`	
`thknew`	
`thknss`	Layer thickness.
`thkold`	
`tmax, tmin`	
`tndcys`	
`tndcyt`	
`tosal`	
`totem`	
`tr2`	

`trmax, trmin`	
`ttem(kdm)`	Old temperature profile.
`ustar3`	`ustar(i,j)**3`
`value`	

Subroutine mxkrtabj and mxkrtbbj

`depnew`	
`dp1, dp2`	
`i, j, ja`	Array indices.
`k, k1, km`	Layer indices.
`l`	Loop index.
`m, n`	Time step indices.
`q`	
`s1, s2, s3`	
`slo`	
`small`	
`smax`	
`smin`	
`sup`	
`uv1, uv2`	
`uvmax`	
`uvmin`	
`zlo`	
`zup`	

10.4 Procedures

Subroutines `mxkrta, mxkrtaaj, mxkrtabj, mxkrtb, mxkrtbaj, mxkrtbbj`

11 Kraus-Turner Model - Diapycnal Mixing : diapf1.f or diapf2.f

11.1 Formalism and numerical techniques

When HYCOM is run using the Kraus-Turner Model, diapycnal mixing can be calculated using the KPP-style implicit diapycnal mixing algorithm outlined in Section 8.1.2 (subroutine `diapf1`). The second option is to calculate diapycnal mixing using a MICOM-style explicit diapycnal mixing algorithm for hybrid coordinates (subroutine `diapf2`) described in the following section.

11.2 Hybrid coordinate explicit algorithm

The explicit diapycnal-mixing algorithm used in MICOM is based on the model of McDougall and Dewar (1998). The central problem in implementing diapycnal mixing in an isopycnic coordinate model is to exchange potential temperature (θ), salinity (S), and mass (layer thickness, expressed as $\partial p/\partial\rho$) between model layers while preserving the isopycnic reference density in each layer, at the same time satisfying the following conservation laws:

$$\left(\frac{\partial\theta}{\partial t}\right)_\rho + \left(\frac{\partial\rho}{\partial t}\frac{\partial p}{\partial\rho}\right)\frac{\partial\theta}{\partial p} = -\frac{\partial F_\theta}{\partial p} \tag{176}$$

$$\left(\frac{\partial S}{\partial t}\right)_\rho + \left(\frac{\partial\rho}{\partial t}\frac{\partial p}{\partial\rho}\right)\frac{\partial S}{\partial p} = -\frac{\partial F_S}{\partial p} \tag{177}$$

$$\frac{\partial}{\partial t}\left(\frac{\partial p}{\partial\rho}\right)_\rho + \frac{\partial}{\partial\rho}\left(\frac{\partial\rho}{\partial t}\frac{\partial p}{\partial\rho}\right) = 0. \tag{178}$$

The expression $(\partial\rho/\partial t)\,(\partial p/\partial\rho)$ is the generalized vertical velocity in isopycnic coordinates while F_θ, F_S are the diapycnal heat and salt fluxes. Although this model was derived for isopycnic layer models, its use is valid for hybrid layer models with the actual layer densities, not their isopycnic reference densities, preserved during the mixing process.

Mutually consistent forms of the turbulent heat, salt, and mass fluxes are derived by integrating (176) through (178) across individual layers and layer interfaces. In a discrete layer model, whether or not it is isopycnic, layer variables such as θ, S, and momentum are assumed to be constant within layers and to have discontinuities at interfaces. Model layers will be denoted by index k increasing downward. Interfaces will also be denoted by index k, with interfaces k and $k+1$ being located at the top and bottom of layer k, respectively.

Integration of (176) and (177) across the interior of layer k yields

$$\delta p^k \left(\frac{\partial\theta^k}{\partial t}\right) = F_\theta^{k,u} - F_\theta^{k,l} \tag{179}$$

$$\delta p^k \left(\frac{\partial S^k}{\partial t} \right) = F_S^{k,u} - F_S^{k,l}, \tag{180}$$

where $\delta p^k = p^{k+1} - p^k$ is the thickness of layer k and the fluxes represent values infinitely close to the interfaces. The fluxes are assumed to have discontinuities at interfaces, but in contrast to the layer variables, they vary linearly with p in each coordinate layer consistent with the fact that $\partial\theta/\partial t$ and $\partial S/\partial t$ are p-independent. Equations (179) and (180) must satisfy the constraint that ρ remains constant while θ and S change. The required condition is

$$\frac{1}{\rho}\frac{d\rho}{dt} = \beta\frac{\partial S}{\partial t} - \alpha\frac{\partial \theta}{\partial t} = 0, \tag{181}$$

where

$$\alpha = -\tfrac{1}{\rho}\left(\tfrac{\partial\rho}{\partial\theta}\right)_{S,p} \qquad \beta = \tfrac{1}{\rho}\left(\tfrac{\partial\rho}{\partial S}\right)_{\theta,p}$$

are the thermodynamic expansion coefficients for potential temperature and salinity, respectively. Combining (179) through (181), the constraint becomes

$$\beta^k \left(F_S^{k,l} - F_S^{k,u} \right) - \alpha^k \left(F_\theta^{k,l} - F_\theta^{u} \right) = 0. \tag{182}$$

Physical intuition suggests that the turbulent fluxes immediately above and below an interface, while usually different, should depend linearly on the magnitude of the discontinuity of the fluxed variable at the interface. It is also reasonable to postulate that the proportionality factor is independent of the fluxed variable. These assumptions are expressed analytically as

$$\tfrac{F_\theta^{k+1,u}}{\theta^{k+1}-\theta^k} = \tfrac{F_S^{k+1,u}}{S^{k+1}-S^k} = m^{k+1,u} \qquad \tfrac{F_\theta^{k,l}}{\theta^{k+1}-\theta^k} = \tfrac{F_S^{k,l}}{S^{k+1}-S^k} = m^{k,l}. \tag{183}$$

Combining (182) and (183) yields

$$m^{k,u}\left[\beta^k\left(S^k - S^{k-1}\right) - \alpha^k\left(\theta^k - \theta^{k-1}\right)\right] = m^{k,l}\left[\beta^k\left(S^{k+1} - S^k\right) - \alpha^k\left(\theta^{k+1} - \theta^k\right)\right].$$

To satisfy this relation in general, $m^{k,u}$ and $m^{k,l}$ must be of the form

$$m^{k,l} = \tfrac{c^k}{\beta^k\left(S^k-S^{k-1}\right)-\alpha^k\left(\theta^k-\theta^{k-1}\right)} \qquad m^{k,u} = \tfrac{c^k}{\beta^k\left(S^{k+1}-S^k\right)-\alpha^k\left(\theta^{k+1}-\theta^k\right)} .$$

Substitution of these expressions into (183) yields

$$F_\theta^{k,u} = c^k \tfrac{\theta^k-\theta^{k-1}}{\beta^k\left(S^k-S^{k-1}\right)-\alpha^k\left(\theta^k-\theta^{k-1}\right)} \qquad F_\theta^{k,l} = c^k \tfrac{\theta^{k+1}-\theta^k}{\beta^k\left(S^{k+1}-S^k\right)-\alpha^k\left(\theta^{k+1}-\theta^k\right)} \tag{184}$$

plus analogous expressions for $F_S^{k,u}$ and $F_S^{k,l}$ involving the same constant c^k.

To determine the proportionality factor c^k in (184), first note that the denominators in (184) represent a relative jump $\partial\rho/\rho$, the fluxes in (184) are finite-difference analogs of expressions of the general form $F_Q = c\rho\partial Q/\partial\rho$ for variable Q. If the turbulent fluxes of Q are also represented in the context of K theory, $F_Q = -K\partial Q/\partial p$. Equating these two expressions for F_Q results in

$$c^k = -\frac{K}{\rho}\frac{\partial\rho}{\partial p} = -K\frac{N^2}{g}.$$

The fluxes then become

$$\begin{aligned} F_\theta^{k,u} &= -G^{k,u}\left(\theta^k - \theta^{k-1}\right) & F_\theta^{k,l} &= -G^{k,l}\left(\theta^{k+1} - \theta^k\right) \\ F_S^{k,u} &= -G^{k,u}\left(S^k - S^{k-1}\right) & F_S^{k,l} &= -G^{k,l}\left(S^{k+1} - S^k\right) \end{aligned} \tag{185}$$

where

$$G^{k,u} \equiv \frac{\left(K\frac{N^2}{g}\right)^k}{\beta^k\left(S^k - S^{k-1}\right) - \alpha^k\left(\theta^k - \theta^{k-1}\right)} \quad G^{k,l} \equiv \frac{\left(K\frac{N^2}{g}\right)^k}{\beta^k\left(S^{k+1} - S^k\right) - \alpha^k\left(\theta^{k+1} - \theta^k\right)} \ . \tag{186}$$

Due to the physical uncertainties surrounding the magnitude of the exchange coefficient K, considerable freedom exists in evaluating the term $\partial\rho/\partial z$ in N^2. One obvious choice was selected for use:

$$\frac{\Delta\rho}{\Delta p} = \frac{\rho^{k+1} - \rho^{k-1}}{2\left(p^{k+1} - p^k\right)}.$$

Having derived expressions for the vertical fluxes, the expressions used to advance layer thickness (mass flux), θ, and S in time will now be derived. Considering the mass flux first, the expression $(d\rho/dt)(\partial p/\partial\rho)$ is evaluated by converting (176) and (177) to flux form and integrating them across interfaces. This step is taken to remove the ambiguity that $\partial\theta/\partial t$ and $\partial S/\partial t$ are indeterminate at interfaces. The flux equations for heat and salt, obtained in the usual manner by combining the advective forms of the equations with (178), are

$$\frac{\partial}{\partial t}\left(\theta\frac{\partial p}{\partial\rho}\right) + \frac{\partial}{\partial\rho}\left(\frac{d\rho}{dt}\frac{\partial p}{\partial\rho}\theta\right) = -\frac{\partial F_\theta}{\partial\rho} \tag{187}$$

and

$$\frac{\partial}{\partial t}\left(S\frac{\partial p}{\partial\rho}\right) + \frac{\partial}{\partial\rho}\left(\frac{d\rho}{dt}\frac{\partial p}{\partial\rho}S\right) = -\frac{\partial F_S}{\partial\rho} \tag{188}$$

Integrating these equations over an infinitesimally thin slab straddling layer interface $k+1$ at the base of layer k (which will be denoted by $k+1/2$ to avoid confusion with the layer index), the time tendency terms drop out because of the smallness of $\partial p/\partial\rho$, producing

$$\left(\frac{d\rho}{dt}\frac{\partial p}{\partial \rho}\right)^{k+1/2} = -\frac{F_\theta^{k+1,u} - F_\theta^{k,l}}{\theta^{k+1} - \theta^k} \tag{189}$$

and

$$\left(\frac{d\rho}{dt}\frac{\partial p}{\partial \rho}\right)^{k+1/2} = -\frac{F_S^{k+1,u} - F_S^{k,l}}{S^{k+1} - S^k}. \tag{190}$$

By virtue of (185) and (186), the two previous expressions reduce to

$$\left(\frac{d\rho}{dt}\frac{\partial p}{\partial \rho}\right)^{k+1/2} = G^{k,l} - G^{k+1,u}. \tag{191}$$

This expression gives the mass flux at interface $k+1/2$ which, in conjunction with the heat and salt fluxes given by (185), form a consistent set that can be used to solve the conservation equations for layer thickness, potential temperature, and salinity.

Inserting (191) into (178) and integration over layer k yields

$$\frac{\partial}{\partial t}(\delta p)^k + \left(G^{k,u} + G^{k,l}\right) - G^{k-1,l} - G^{k+1,u} = 0. \tag{192}$$

The G terms have been grouped in a manner that mimics the second derivative of G with respect to k, which illustrates that diapycnal mixing tends to transfer mass from thick layers (N^2 small) to thin layers (N^2 large). The prognostic equations for heat and salt are obtained from the flux form of the conservation equations (187) and (188), which are integrated over the interior portion of layer k to produce

$$\frac{\partial}{\partial t}\left[\theta^k (\partial p)^k\right] + \left(G^{k+1,u} + G^{k-1,l}\right)\theta^k - G^{k,u}\theta^{k-1} - G^{k,l}\theta^{k+1} = 0 \tag{193}$$

and

$$\frac{\partial}{\partial t}\left[S^k (\partial p)^k\right] + \left(G^{k+1,u} + G^{k-1,l}\right)S^k - G^{k,u}S^{k-1} - G^{k,l}S^{k+1} = 0. \tag{194}$$

The G terms have again been arranged to highlight the diffusive nature of the equations. Equations (192) through (194) are integrated subject to the boundary conditions

$$\left(\frac{d\rho}{dt}\frac{\partial p}{\partial \rho}\right)^{ktop-1/2} = F_\theta^{ktop,u} = F_S^{ktop,u} = 0 \qquad \left(\frac{d\rho}{dt}\frac{\partial p}{\partial \rho}\right)^{kbot+1/2} = F_\theta^{kbot,l} = F_S^{kbot,l} = 0.$$

11.3 Usage

* INSERT TEXT HERE *

11.3.1 Order of Operations

* INSERT TEXT HERE *

11.3.2 Flowchart

∗ INSERT FLOWCHART HERE ∗

11.4 Variables

11.4.1 Identification

Notation in the theory	Notation in **diapf1.f or diapf2.f**
∗ INSERT TEXT HERE ∗	∗ INSERT TEXT HERE ∗

11.4.2 Local variables

Subroutines diapf1, diapf1aj, and diapf1bj

`i, j`	Array indices.
`l`	Loop index.
`m, n`	Time step indices.

Subroutine diapf1aij

`alfadt(kdm+1)`	T contribution to density jump.
`betads(kdm+1)`	S contribution to density jump.
`dbloc(kdm+1)`	Buoyancy jump across interface.
`diffdd`	Double diffusion diffusivity scale.
`diffs(kdm+1)`	
`difft(kdm+1)`	
`dzb(kdm)`	
`fri`	Function of rig.
`froglp`	
`hm(kdm)`	
`hwide(kdm)`	Layer thicknesses in (m).
`i, j`	Array indices.
`k, ka`	Layer indices.

`m, n`	Time step indices.
`kmask`	
`mixflo`	
`nlayer`	
`prandtl`	Prandtl number.
`ratio`	
`rhs(kdm)`	Right-hand-side terms.
`rigr`	Local Richardson number.
`rrho`	Double diffusion parameter.
`s1dn(kdm+1)`	
`s1do(kdm+1)`	
`shsq(kdm+1)`	Velocity shear squared.
`t1dn(kdm+1)`	
`t1do(kdm+1)`	
`tcc(kdm)`	Central coefficient for (k-1) on k line of tridiagonal matrix.
`tcl(kdm)`	Lower coefficient for (k-1) on k line of tridiagonal matrix.
`tcu(kdm)`	Upper coefficient for (k-1) on k line of tridiagonal matrix.
`tr1dn(kdm+1)`	
`tr1do(kdm+1)`	
`tri(kdm,0:1)`	Dt/dz/dz factors in tridiagonal matrix.
`zm(kdm+1)`	

Subroutine diapf1uij

`diffm(kdm+1)`	
`dzb(kdm)`	
`froglp`	

`hm(kdm)`	
`i, j`	Array indices.
`k, ka`	Layer indices.
`kmask(idm)`	
`m, n`	Time step indices.
`mixflg`	
`nlayer`	
`presu`	
`tcc(kdm)`	Central coefficient for (k-1) on k line of tridiagonal matrix.
`tcl(kdm)`	Lower coefficient for (k-1) on k line of tridiagonal matrix.
`tcu(kdm)`	Upper coefficient for (k-1) on k line of tridiagonal matrix.
`tri(kdm)`	Dt/dz/dz factors in tridiagonal matrix.
`u1dn(kdm+1)`	
`u1do(kdm+1)`	
`zm(kdm+1)`	

Subroutine diapf1vij

`diffm(kdm+1)`	
`dzb(kdm)`	
`froglp`	
`hm(kdm)`	
`i, j`	Array indices.
`k, ka`	Layer indices.
`kmask(idm)`	
`m, n`	Time step indices.
`mixflg`	

`nlayer`	
`presv`	
`rhs(kdm)`	Right-hand-side terms.
`tcc(kdm)`	Central coefficient for (k-1) on k line of tridiagonal matrix.
`tcl(kdm)`	Lower coefficient for (k-1) on k line of tridiagonal matrix.
`tcu(kdm)`	Upper coefficient for (k-1) on k line of tridiagonal matrix.
`tri(kdm,0:1)`	Dt/dz/dz factors in tridiagonal matrix.
`v1dn(kdm+1)`	
`v1do(kdm+1)`	
`zm(kdm+1)`	

Subroutine diapf2

`j`	Array index.
`m, n`	Time step indices.

Subroutine diapf2j

`alfa`	
`amount`	Limited flux.
`beta`	
`clips(idm)`	
`clipt(idm)`	
`delp`	Pressure variation at interfaces.
`ennsq`	
`flngth(idm,kdm)`	
`flxl(idm,kdm)`	
`flxu(idm,kdm)`	

`froglp`	
`i, j`	Array indices.
`k, ka`	Layer indices.
`m, n`	Time step indices.
`kmax(idm)`	
`kmin(idm)`	
`pdot(idm,kdm)`	
`q`	
`qmax`	
`qmin`	
`sflxl(idm,0:kdm+1)`	
`sflxu(idm,0:kdm+1)`	
`small`	
`sold(idm,2)`	Old salinity.
`tflxl(idm,0:kdm+1)`	
`tflxu(idm,0:kdm+1)`	
`tndcys`	
`tndcyt`	
`tosal(idm)`	
`totem(idm)`	
`trflxl(idm,0:kdm+1)`	
`trflxu(idm,0:kdm+1)`	
`trold(idm,2)`	

11.5 Procedures

Subroutines	`diapf1, diapf1aj, diapf1bj, diapf1aij, diapf1uij, diapf1vij, diapf2, diapf2j`

12 MICOM Mode - K-T Model 3 : mxkrtm.f

When HYCOM is run with isopycnic vertical coordinates (MICOM mode), the model automatically uses K-T model 3, which is essentially the mixed layer model embedded in MICOM version 2.8.

12.1 Formalism and numerical techniques

Modelling the seasonal evolution of the thermal regime of the surface ocean implies including the effects of forcing by the atmosphere, whose fundamental properties are totally different from those of the ocean. The solar influence translates in part to a gain in radiative heat. Moreover, it has been known for a long time that the effect of the prevailing winds in the atmospheric boundary layer manifests itself through turbulent transport of momentum in the surface layers. To account explicitly for the processes connected to ocean-atmosphere exchanges in the context of an isopycnic theory, an integral-type model whose development was initiated by Kraus & Turner (1967) is used by HYCOM in MICOM mode . In such a model of the surface layer behavior, the problem of closing the turbulence equations is greatly simplified since the turbulent fluxes at the boundaries of the mixed layer simply need to be determined. The progressive shrinking of the thickness h of the surface layer by diapycnic mixing is counterbalanced by a mechanism increasing the thickness. For this, an entrainment velocity w_e is introduced at the base of the layer such that :

$$\begin{aligned} w_e &= \frac{dh}{dt} \quad \text{if} \quad \frac{dh}{dt} > 0 \\ w_e &= 0 \quad \text{if} \quad \frac{dh}{dt} \leq 0 \end{aligned} \tag{195}$$

The originality of this type of model is to express the vertical turbulent fluxes at the base of the mixed layer by an equation of the general form :

$$-\left(\overline{a'w'}\right)_{-h} = w_e \Delta a \tag{196}$$

where Δa depicts the discontinuity of the variable at the base of the layer. In this model, the effect of the wind is represented by the surface stress $\boldsymbol{\tau_s}(\tau_{s_x}, \tau_{s_y})$.

12.1.1 Internal energy and turbulent kinetic energy

The evolution equation of the internal energy of a homogeneous layer of temperature θ_s is then given by :

$$h\frac{\partial \theta_s}{\partial t} = \left(\overline{\theta' w'}\right)_{-h} - \left(\overline{\theta' w'}\right)_0 + \frac{1}{\rho C_p}(R_0 - R_h) - \left[K_H \left(\frac{\partial \theta}{\partial z}\right)\right]_{-h} \tag{197}$$

with : $-\left(\overline{\theta' w'}\right)_0 = \frac{P}{\rho C_p}$ and $-\left(\overline{\theta' w'}\right)_{-h} = w_e \Delta\theta$. R_z is the solar radiation at the depth z and P the surface heat losses (infrared radiation, latent and sensible heat flux). $\Delta\theta$ represents the temperature discontinuity at the base of the mixed layer. The flux $\left[K_H\left(\frac{\partial\theta}{\partial z}\right)\right]_0$ is accounted for at the surface. If the latter is null, it is situated in the upper limit of the homogeneous layer.

The knowledge of the entrainment velocity necessitates a supplementary equation. In a homogeneous layer model, this calculation is carried out using the integrated equation of turbulent kinetic energy $E/2$:

$$\underbrace{\frac{1}{2}\int_{-h}^{0}\frac{\partial E}{\partial t}\,dz}_{1} = \underbrace{\left[\overline{\left(\frac{E}{2}+\frac{\pi'}{\rho_0}w'\right)}\right]_{-h}}_{2} - \underbrace{\left[\overline{\left(\frac{E}{2}+\frac{\pi'}{\rho_0}w'\right)}\right]_{0}}_{3} - \underbrace{\int_{-h}^{0}\overline{\mathbf{u}'w'}.\left(\partial\mathbf{u}/\partial z\right)\,dz}_{4} + \underbrace{\int_{-h}^{0}\overline{b'w'}\,dz}_{5} - \underbrace{h\epsilon_m}_{6} \quad (198)$$

with :

$\mathbf{u}(u,v)$: horizontal velocity ;
$b = g(\rho_0-\rho)/\rho_0$: buoyancy ;
π : pressure ;
ϵ_m : TKE average heat dissipated in the mixed layer.

In the complete form, this equation expresses therefore the equilibrium between :

1. the TKE tendency contained in the mixed layer ;
2. the TKE flux at the base of the mixed layer ;
3. the TKE surface flux ;
4. a production term from cutting the current ;
5. a consumption term corresponding to the buoyancy of the homogeneous layer ;
6. a heat dissipation term.

Niller & Kraus (1977) have shown that the TKE flux at the base of the mixed layer is generally negligible. On the other hand, the TKE content of this layer is more often considered constant. Further, following Kraus & Turner (1967), the surface TKE flux originating principally from the turbulent agitation at the surface (waves, swells, ...) is parametrized as :

$$\left[\overline{\left(\frac{E}{2}+\frac{\pi'}{\rho_0}w'\right)}\right]_0 = m_2 u_*^3 \quad (199)$$

where u_* is the surface drag velocity such that :

$$u_*^2\mathbf{x} = \boldsymbol{\tau}_s/\rho \quad (200)$$

As for the production term, simply write :

$$\int_{-h}^{0} \overline{\mathbf{u}'w'}.\left(\partial \mathbf{u}/\partial z\right)\ dz = m_3 u_*^3 \tag{201}$$

In referring to a reference state with temperature T_0, salinity S_0, and in hydrostatic equilibrium, let :

$$\rho = \rho_0[1 - \alpha_T(T - T_0) + \beta_S(S - S_0)] \tag{202}$$

with the expansion coefficients α_T and β_S defined such that :

$$\alpha_T = -\frac{1}{\rho}\left(\frac{\partial \rho}{\partial T}\right)_S \quad \text{et} \quad \beta_S = \frac{1}{\rho}\left(\frac{\partial \rho}{\partial S}\right)_T \tag{203}$$

$$\int_{-h}^{0} \overline{b'w'}\ dz = -\frac{1}{2}h\left(w_e \Delta b + B(h)\right) \tag{204}$$

$B(h)$ then represents the sum of surface flux $-\left(\overline{b'w'}\right)_0$ and of the increase of buoyancy due to the absorption of solar radiation :

$$B(h) = -\left(\overline{b'w'}\right)_0 + \frac{\alpha_T g}{\rho C_p}\left(R_0 + R_h - \frac{2}{h}\int_{-h}^{0} R(z)\ dz\right) \tag{205}$$

The TKE conservation equation in the mixed layer can then finally be written as :

$$(m_2 + m_3)u_*^3 - \frac{h}{2}\left(B(h) - w_e \Delta b\right) - h\epsilon_m = 0 \tag{206}$$

12.1.2 Parametrization of turbulent dissipation

To treat this 'handicap' and to address a generalization of the parametrization of the turbulent dissipation ϵ, Gaspar (1988) reintroduced the vertical dissipation scale of Kolmogorov l_ϵ such that :

$$\epsilon = E^{3/2}/l_\epsilon \tag{207}$$

Let, then, over the mixed layer :

$$\epsilon_m = u_e^3/l \tag{208}$$

u_e is a characteristic velocity scale of the turbulence in the mixed layer and l a dissipation length which can be expressed formally as a function of diverse parameters :

$$l = F(h, L, \lambda, L_\Delta, L_N) \tag{209}$$

with :

$L = u_*^3/B(h)$: Monin-Obukov length
$\lambda = u_*/f$: Ekman length
$L_\Delta = (\Delta u^2 + \Delta v^2)/\Delta b$: relative scale at the entrainment zone
$L_N = \left(E^{1/2}/N\right)_{-h}$: scale of stratification at the base of the mixed layer
N : Brunt-Vaïsälä frequency

12.1.3 A recent prediction model of the mixed layer

Since the dynamic instability present at the base of the mixed layer is typically of a time scale on the order of the inertial period, Gaspar (1988) notes that this phenomenon can be neglected in the seasonal studies. In a parallel way, the influence of the TKE dissipation produced by this term will therefore be omitted : L_Δ does not enter in the determination (209) of l. On the other hand, to introduce L_N will not have an important effect if ΔT is very weak. This is rarely the case in the ocean. Finally, (208) can be expressed as :

$$h\epsilon_m = u_e^3 G(h/L, h/\lambda) \tag{210}$$

G is a function of the stability parameter of Monin-Obukov h/L and of the rotation parameter h/λ. Realizing that setting $u_e = u_*$ causes underestimation of the turbulent velocity scale which causes convective deepening, Gaspar (1988) writes first of all :

$$u_e^2 = E_m = \frac{1}{h}\int_0^h E\ dz \tag{211}$$

From the earlier studies of the subject, it appears that the stability parameter can be expressed by the relation :

$$G(h/L, h/\lambda) = \frac{h}{l} = a_1 + a_2\ \max\left[1, h/(0.4\lambda)\right]\exp\left(h/L\right) \tag{212}$$

Finally, the prediction of h is carried out using the expression (206), in which the turbulent dissipation comes from the forms (210), (211) and (212) :

$$(m_2 + m_3)u_*^3 - \frac{h}{2}\left(B(h) - w_e\Delta b\right) - (h/l)E_m^{3/2} = 0 \tag{213}$$

Moreover, the turbulent vertical characteristic velocity scale is introduced such that :

$$u_w^2 = \frac{1}{h}\int_0^h \overline{w'^2}\ dz = W_m \tag{214}$$

Then, the TKE equation at the base of the mixed layer is written :

$$-\left(\overline{b'w'}\right)_{-h} = -\frac{\partial}{\partial z}\left[\overline{\left(\frac{E}{2} + \frac{p'}{\rho_0}w'\right)}\right]_{-h} - \left[\overline{\mathbf{u}'w'}.\left(\partial\mathbf{u}/\partial z\right)\right]_{-h} - \epsilon_{-h} \tag{215}$$

In consideration of the relative values of each term of this relation, Gaspar puts it in the general entrainment form :

$$h\Delta b w_e = m_1 u_e^2 u_w \tag{216}$$

Finally, let :

$$h\Delta b w_e = m_1 E_m W_m^{1/2} \tag{217}$$

The equation giving W_m is obtained by integrating the tendency equation of $\overline{w'^2}$ over the homogeneous layer. Gaspar obtains the form :

$$\left(\frac{1}{2}-\frac{m_5}{3}\right)[h\Delta bw_e + hB(h)] = \frac{h}{3}\left(\frac{m_4}{l_p}-\frac{1}{l}\right)E_m^{3/2} + \frac{m_3 m_5}{3}u_*^3 - \frac{m_4 h}{l_p}E_m^{1/2}W_m \tag{218}$$

l_p is a characteristic length in the absence of rotation :

$$\frac{h}{l_p} = a_1 + a_2 \exp{(h/L)} \tag{219}$$

The equations (213), (217) and (218) constitute the total system of the Oceanic Mixed Layer model described in the article of Gaspar (1988). The entrainment velocity is calculated numerically from the formula :

$$h\Delta bw_e = \frac{\left[(0.5A_p - C_{p1}S_p)^2 + 2C_4(h/l)^2 A_p S_p\right]^{1/2} - (0.5A_p + C_{p1}S_p)}{C_4(h/l)^2 - C_{p1}} \tag{220}$$

With :

$$A_p = C_{p3}u_*^3 - C_{p1}hB(h) \tag{221}$$

$$S_p = (m_2+m_3)u_*^3 - \frac{1}{2}hB(h) \tag{222}$$

$$C_{p1} = [2(1-m_5)(l_p/l) + m_4]/6 \tag{223}$$

$$C_{p3} = [m_4(m_2+m_3) - (l_p/l)(m_2+m_3-m_5m_3)]/3 \tag{224}$$

$$C_4 = 2m_4 m_1^{-2} \tag{225}$$

12.1.4 Entrainment condition

From equation (213), it is clear that entrainment does not manifest itself if the condition $S_p > 0$ is true. In the oceanic mixed layer, the TKE resulting from the production mechanisms less the energy consumed to homogenize the heat gain due to thermal input (all solar radiation + surface losses) should be positive.
On the other hand, the relation (217) implies :

$$W_m > 0 \tag{226}$$

In eliminating $h\Delta bw_e$ of (213) and (218), the following equation is obtained :

$$W_m = \frac{2C_{p1}}{m_4}E_m - \frac{C_2 l_p}{m_4 h}u_*^3 E_m^{-1/2} \tag{227}$$

where W_m is expressed as a function of E_m and C_2 is a positive constant :

$$C_2 = [(3-2m_5)(m_2+m_3) - m_5m_3]/3 \tag{228}$$

W_m is cancelled by :

$$E_{m0} = u_*^2 \left(\frac{C_2 l_p}{2C_{p1} h} \right)^{2/3} \tag{229}$$

The entrainment condition (226) is therefore equivalent to :

$$E_m > E_{m0} \tag{230}$$

Entrainment does not appear if the TKE exceeds a minimum given by the form (229). After having eliminated $h\Delta b w_e$ of (213) and (217), E_m is given by the zero of the function :

$$F(E_m) = \frac{1}{2} m_1 E_m W_m^{1/2} + \frac{h}{l} E_m^{3/2} - S_p \tag{231}$$

in which W_m depends on E_m by the intermediary of the relation (227). For $E_m > E_{m0}$, F is a strictly increasing function of E_m. The entrainment can not appear if :

$$F(E_{m0}) < 0 \tag{232}$$

Entrainment can not exist therefore if the TKE balance S_p is greater than the minimal dissipation (*i.e.* the dissipation associated with E_{m0}, the minimum value of E_m).
Writing

$$A_p = S_p - \frac{h}{l} E_{m0}^{3/2}, \tag{233}$$

the necessary and sufficient condition for entrainment at the base of the mixed layer ($w_e > 0$) is :

$$A_p > 0. \tag{234}$$

In the case where this condition is not satisfied, the hypothesis is that h automatically adjusts itself to maintain $A_p = 0$. When the heat balance $B(h)$ is not zero, this is equivalent to :

$$h = \frac{C_{p3}}{C_{p1}} L \tag{235}$$

Following the theory of Niiler & Kraus (1977), the equilibrium of the mixed layer is attained by :

$$h = \frac{2mu_*^3}{(-B_0)} \tag{236}$$

B_0 is the surface buoyancy flux. The authors assume that it varies linearly inside the mixed layer and is zero at the base. The Monin-Obukov length is then expressed as $L = u_*^3/\kappa B_0$. κ is the Von Karman constant ($\kappa \simeq 0.4$). To remain in accord with this definition, set $m = 1.25$.

12.1.5 Constants and numerical parameters

Referring to diverse earlier studies, Gaspar retains the following values by the different parameters introduced above : $m_1 = 0.45$; $m_2 = 2.6$; $m_3 = 1.9$; $m_4 = 2.3$; $m_5 = 0.6$; $a_1 = 0.6$; $a_2 = 0.3$

12.2 Numerical techniques

12.2.1 Entrainment algorithm

During a time interval Δt, after having detected a deepening (entrainment) ($w_e > 0$), the growth of the thickness of the mixed layer can be written :

$$\Delta h = \frac{E\Delta t}{b_{mix} - b_{sub}} \tag{237}$$

With :

$E\Delta t$: TKE variation in the mixed layer
b_{mix} : buoyancy of the mixed layer of thickness h
b_{sub} : buoyancy of the adjacent isopycnic layer

The form (237) can not be incorporated directly in a numerical calculation (the denominator can be zero). Moreover, this expression assumes that under the mixed layer exists a layer of buoyancy b_{sub} of finite thickness. As there is no *a priori* reason for these two assumptions to be true, Bleck *et al.* (1989) have developed a particular method for solving (237).
In HYCOM, calculating $E\Delta t$ is optional. Option (1) uses the formulation of the TKE evolution equation following the method of Kraus & Turner (1967) ($m \neq 0$; $n = 0.15$; $s = 0$). Option (2) employs the relation (220).

Let b_1 and z_1 be the buoyancy and depth of the mixed layer at a point P of the domain. The variables z_k are introduced to represent the lower positions with respect to the sub-adjacent isopycnic layers of buoyancy b_k. The potential energy (PE) of the column of water is expressed as :

$$PE = \int_{z_0}^{z_N} bz \; dz = \frac{1}{2} \sum_{k=1}^{N} b_k (z_k^2 - z_{k-1}^2) \tag{238}$$

with $z_0 = 0$. The same column of water, but mixed has the PE :

$$PE_{mix} = b_{mix} \frac{z_N^2}{2} \tag{239}$$

with :

$$b_{mix} = \frac{1}{z_N} \sum_{k=1}^{N} b_k (z_k - z_{k-1}) \tag{240}$$

Also let :

$$PE_{mix} = \frac{z_N}{2} \sum_{k=1}^{N} b_k (z_k - z_{k-1}) \tag{241}$$

In remarking that the kinetic energy used to mix the heat acquired by radiation in the surface layer is the same as the difference (241)-(238) over the depth h :

$$h = \frac{2E\Delta t + G(l-1) - b_l z_{l-1}^2}{F(l-1) - b_l z_{l-1}} \tag{242}$$

with : $F(l) = \sum_{k=1}^{l} b_k(z_k - z_{k-1})$ and $G(l) = \sum_{k=1}^{l} b_k(z_k^2 - z_{k-1}^2)$. Note that the calculation of h by such a formulation implies knowledge of the number l of the concerned layers.

12.2.2 Detrainment algorithm

Consider the situation after a time interval Δt in the case where the term $E\Delta t$ becomes negative. The index $()_1$ always serves to identify the mixed layer. Suppose the buoyancy b_1 lies between the two discrete values b_k and b_{k-1} of the chosen initial distribution. In this case, the Monin-Obukov length satisfies the condition $L < z_1$.
In principle, the buoyancy transfer can be described by the conservation equation :

$$(b_1' - b_1)L = (b_1 - b_k)(z_1 - L). \tag{243}$$

But then, the following two conditions must be satisfied :

condition A : $b_1' < b_{max}$ where b_{max} is the buoyancy which a fictitious mixed layer of thickness L would acquire during a time interval Δt under the influence of B_0. It is a threshold procedure. Its usage permits avoiding surface reheating of the ocean caused by errors in the ocean-atmosphere flux which serve to force the model. b_{max} is given by the expression :

$$b_{max} = b_1 - B_0 \Delta t \left(\frac{1}{L} - \frac{1}{z_1} \right) \tag{244}$$

If the value of b_1' evaluated by (243) exceeds b_{max}, this threshold is translated into the new mixed layer thickness :

$$z_1' = z_1 \frac{b_1 - b_k}{b_{max} - b_k}. \tag{245}$$

condition B : $b_1' < b_{k-1}$. If this condition B is true, it will be necessary to partition the buoyancy transfer over the layers k and $k-1$. This provision handles well the non-isopycnic behavior of the mixed layer between discrete values b_k and b_{k-1}. For $b_1' > b_{k-1}$, it is possible to distribute a part of the buoyancy over a layer of density ρ_{k-1} and of thickness $z_1' - L$ such that :

$$z_{k-1}' = z_1 \frac{b_1 - b_k}{b_{k-1} - b_k} - L \frac{b_{max} - b_{k-1}}{b_{k-1} - b_k} \tag{246}$$

If condition B is not satisfied, the mixed-layer detrainment mechanism therefore leads to inflation of two sub-surface layers. The process is decomposed in three steps (figure 12) :

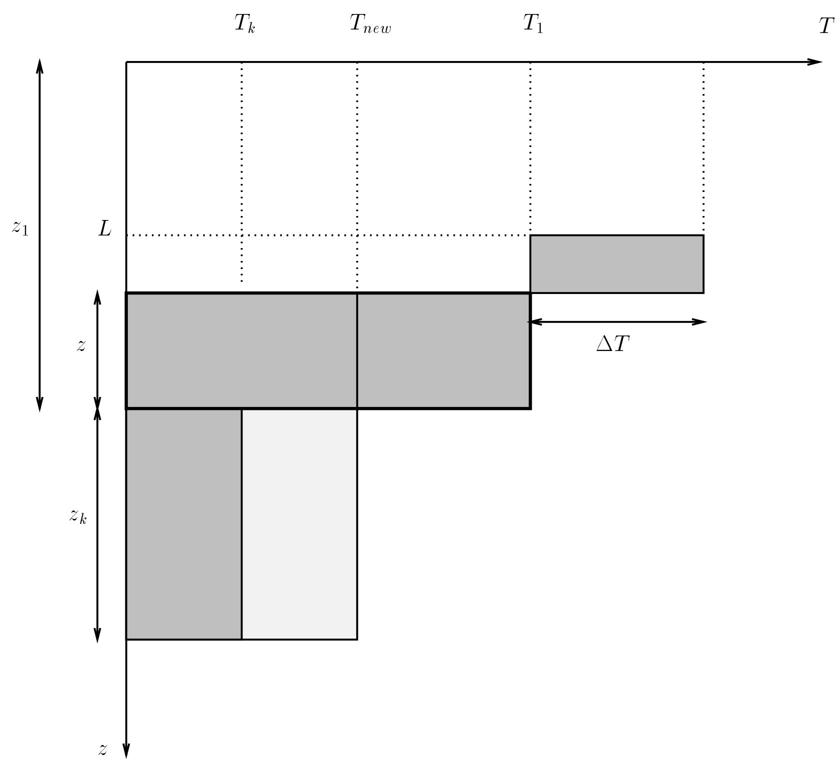


Figure 12: *Conservation of heat during detrainment of the mixed layer.*

1. Formation of the new mixed layer of thickness equal to the Monin-Obukov length and of buoyancy b_{max} ;
2. Inflation of an intermediate layer $(k-1)$;
3. Adjustment of the old sub-adjacent layer (k) to the mixed layer.

In fact, the application of relation (246) (*i.e.* creating an intermediate layer of density σ_{k-1}) is not possible if the buoyancy is not a function of only one variable. Taking into account the joint effects of salinity and temperature introduces a degree of supplementary freedom. In this case, it is necessary to extract from the layer k a quantity of heat equal to the sum of the heat required to match the buoyancy value of the part above the slice of water entrained and of the heat corresponding to the increment $(b'_1 - b_1)$ of the mixed layer. In this framework, Bleck *et al.* (1992) have developed a particular method to schematize these transfers. Assume a surface layer of temperature T_1, of salinity S_1 and of thickness z_1. Let T_k, S_k, z_k be the corresponding characteristics for the layer k. If $(z_1 - L) > 0$, the procedure transfers the excess of the heat contained in a layer of thickness $(z_1 - L)$ to the layer k. Since the model is using isopycnic coordinates, this transfer will be paired with a heat transfer of this ascending layer to the mixed layer. Let ΔT be the maximum allowed growth of the temperature of the mixed layer allowed during a time step. Note by z the thickness of the fraction of the intermediate layer of vertical extension $(z_1 - L)$ which allows an augmentation of temperature ΔT. Since the mixed layer has a buoyancy greater than that of layer k, the problem becomes one of determining z. The heat transfer to layer k leads to a temperature T_{new} such that (figure 12) :

$$T_1 z + T_k z_k - \Delta T(z_1 - z) + T_{new}(z + z_k) \tag{247}$$

Let :

$$T_{new} = \frac{(T_1 + \Delta T)\, z - \Delta T z_1 + T_k z_k}{z + z_k} \tag{248}$$

and the salinity :

$$S_{new} = \frac{S_1 z + S_k z_k}{z + z_k} \tag{249}$$

These two characteristics must satisfy $\sigma(T_{new}, S_{new}) = \sigma_k$. In assuming an equation of state of third degree :

$$\sigma(T,S) = c_1 + c_2 T + c_3 S + c_4 T^2 + c_5 TS + c_6 T^3 + c_7 T^2 S \tag{250}$$

and in expressing T_{new} and S_{new} through the forms $(az+b)/(cz+d)$ and $(ez+f)/(cz+d)$, the conservation of σ_k requires solving the polynomial equation :

$$a_3 z^3 + a_2 z^2 + a_1 z + a_0 = 0 \tag{251}$$

The expressions for coefficients a_0, a_1, a_2, a_3 are given in appendix E of Bleck *et al.* (1992). Moreover, the introduction of a supplementary layer σ_{k-1} implies also the restoration of motion parameters. Now, the MICOM theory assumes that the exchanges of momentum occur immediately after the mass transfers between the mixed layer and sub-adjacent layers (Bleck *et al.*, 1989). For a given layer, the momentum remains constant during the process of rearranging the sub-surface layers concerned in the evolution algorithm of the mixed layer.

12.3 Usage

In the MICOM mode, the numerical calculation of the evolution of the surface mixed layer is carried out by the subroutine :

```
[***** ADD SOURCE CODE HERE *****]
```

12.3.1 Order of operations

Recall that, while deepening the mixed layer is relatively easy to model, to reproduce its recession is more complex. Assume a surface layer of characteristics T_1, S_1, z_1. In the original detrainment algorithm developed by Bleck *et al.* (1989) for one variable of state, later adapted by Bleck *et al.* (1992) for the two variables T and S, the heat acquired by the ocean surface during a time step is not distributed over a slice of water of thickness equal to the Monin-Obukov length L. In fact, the thickness h given by the relation (235) proposed by Gaspar (1988) is introduced. So, the old mixed layer is divided into a new surface layer of characteristics T_1', S_1, h and a fossil layer of thickness $(z_1 - h)$. This last layer is also partitioned into two sub-layers :

1. a slice of water of density equal to that of the sub-adjacent layer σ_k and of salinity S_1 ;
2. an intermediate layer of temperature T_1' and of salinity S_1.

The option of parametrization of dissipation actually used in version 2.0 of MICOM is the formulation of Gaspar (1988). The lines of code of option 1 based on the method of Kraus & Turner (1967) are still available but commented out.

```
[***** ADD SOURCE CODE HERE *****]
```

Once A_p has been calculated with relation (221), the following test is carried out :

if $A_p > 0$, the TKE variation $(E\Delta t)$ of the mixed layer during a time step is positive. To maintain the equilibrium formulated by the equation (213), this growth represents the TKE that will be consumed by entrainment into the mixed layer. It is calculated from the form (**??**). A first inference of the new thickness z_1' of the mixed layer is given by z_1;

if $A_p < 0$, the TKE variation is negative. The TKE contained in the mixed layer will therefore diminish by a quantity $E\Delta t = A_p$. A first inference of the new thickness z_1' of the mixed layer is given by the relation (235) under the condition that it be less than z_1.

```
[***** ADD SOURCE CODE HERE *****]
```

Then, calculate the form (242) iterating over k as long as the value of h obtained remains greater than that of the upper interface of the k^{th} layer situated at the position z_{k-1}. When this condition is no longer true, the order l is obtained of the relation (242) and as a consequence, a second inference of z_1'.

```
[***** ADD SOURCE CODE HERE *****]
```

Before validating this first result, the numerical value which will be retained is required to be located between the bottom and a minimum thickness value of the mixed layer.

```
[***** ADD SOURCE CODE HERE *****]
```

After this, the sign of the variation $\Delta z = z_1' - z_1$ is used as a test :

if $\Delta z > 0$, the result $h_s = z_1'$ is confirmed and the new values of temperature, salinity, and density of the mixed layer are calculated ;

```
[***** ADD SOURCE CODE HERE *****]
```

if $\Delta z < 0$, proceed in two steps :

1. from the general equation of internal energy (197), determine the growth of temperature ΔT undergone by a fictitious layer of thickness z_1' given by the relation (235) from the theory of Gaspar (1988).
2. for the mixed layer to maintain this thickness z_1', the salt content (see figure 14) is conserved. Then, apply the relation of state giving the temperature T_{new} as a function of σ_k (which is conserved) and of salinity S_{new} which is wanted to calculate. The growth Δz of the thickness of the sub-adjacent layer involves a heat transfer in the fictitious surface layer which should be positive. (see figure 15).

```
[***** ADD SOURCE CODE HERE *****]
```

At this stage, it is therefore possible to compare this heat gain taken to z_1' (which translates to an increase of the surface temperature dT) with the temperature growth ΔT found before :

a) if $dT \leq \Delta T$, in this configuration, 100% detrainment is possible. The new values T_{new} and S_{new} are attributed to layer k then, concerning the mixed layer, put $T_1' = T_1 + dT$ and $z'1 = z1 + \Delta z$. In operating this way the non-representative surface heating is guarded against because, unless $dT = \Delta T$ is obtained exactly, the depth of the mixed layer which was fixed is greater than what it should be. In all cases, it should be greater than the threshold value which was fixed by the run. The surface density becomes $\sigma_1' = \sigma(T_1', S_1)$

```
[***** ADD SOURCE CODE HERE *****]
```

b) if $dT > \Delta T$, the heat is distributed over a surface layer of temperature $T_1'' = T_1 + dT$ and of thickness z_1'' such that $z_1'' > z_1'$ (see figure 16). The sub-adjacent layer k is also going to thicken and the new characteristics T_{new} and S_{new} of this layer should also satisfy :

$$\sigma(T_{new}, S_{new}) = \sigma_k \tag{252}$$

In keeping with the preceding case, partial detrainment is now possible. The density of the mixed layer now becomes $\sigma_1'' = \sigma(T_1'', S_1)$. The new characteristics of layer k have particular forms given by appendix E of Bleck *et al.* (1992), the conservation of σ_k comes from the solution of a polynomial of the form (251).

```
[***** ADD SOURCE CODE HERE *****]
```

The specific volume of the surface layer is given by the equation of state.

```
[***** ADD SOURCE CODE HERE *****]
```

The following operation consists of accounting for the thickness variation of the surface layer.

```
[***** ADD SOURCE CODE HERE *****]
```

The last step of the calculation consists of the redistribution of momentum over the water column, accounting for the possibility of momentum mixing by diffusion.

```
[***** ADD SOURCE CODE HERE *****]
```

12.3.2 Flowchart

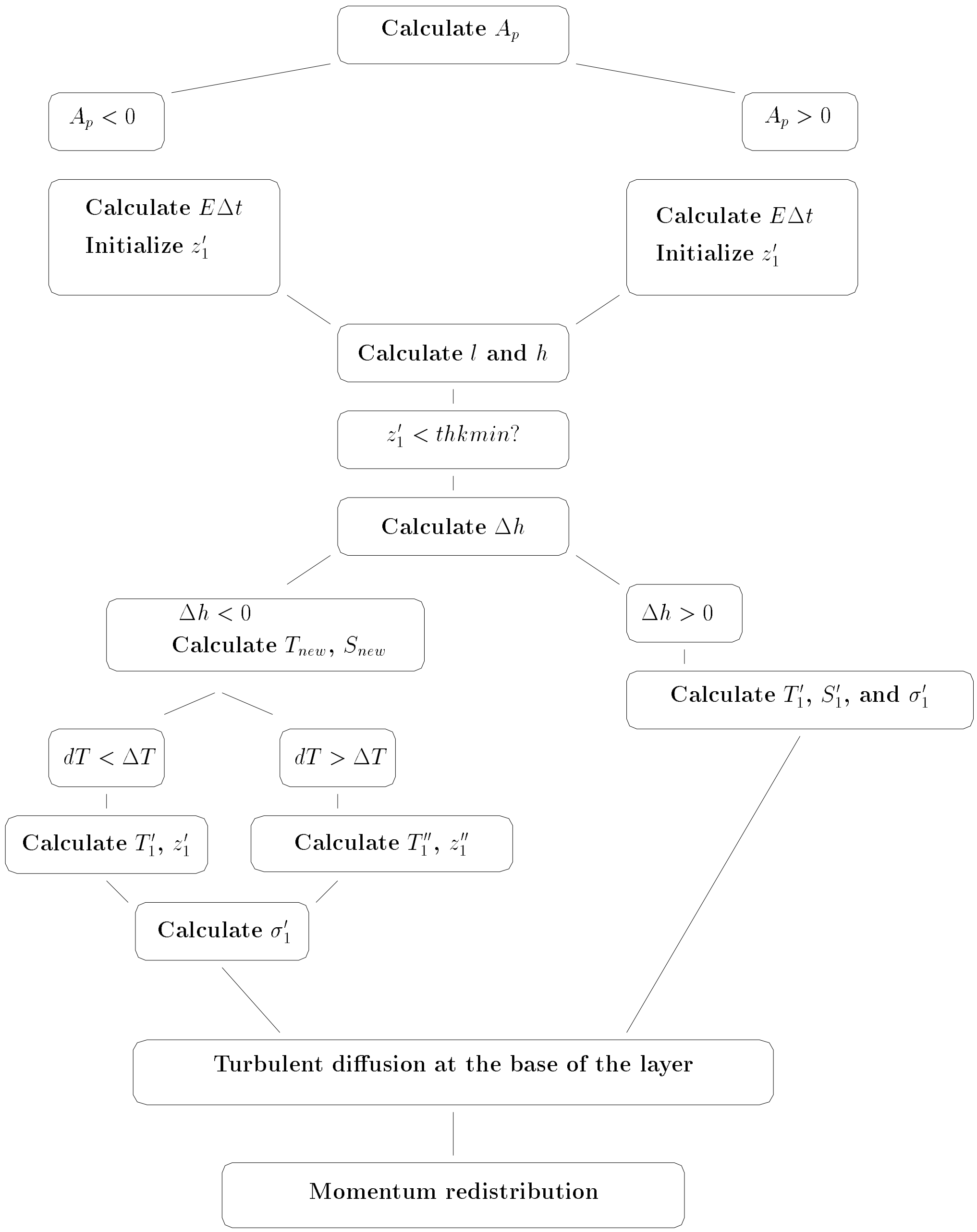


Figure 13: *Flowchart of the calculation of the mixed layer evolution in MICOM mode*

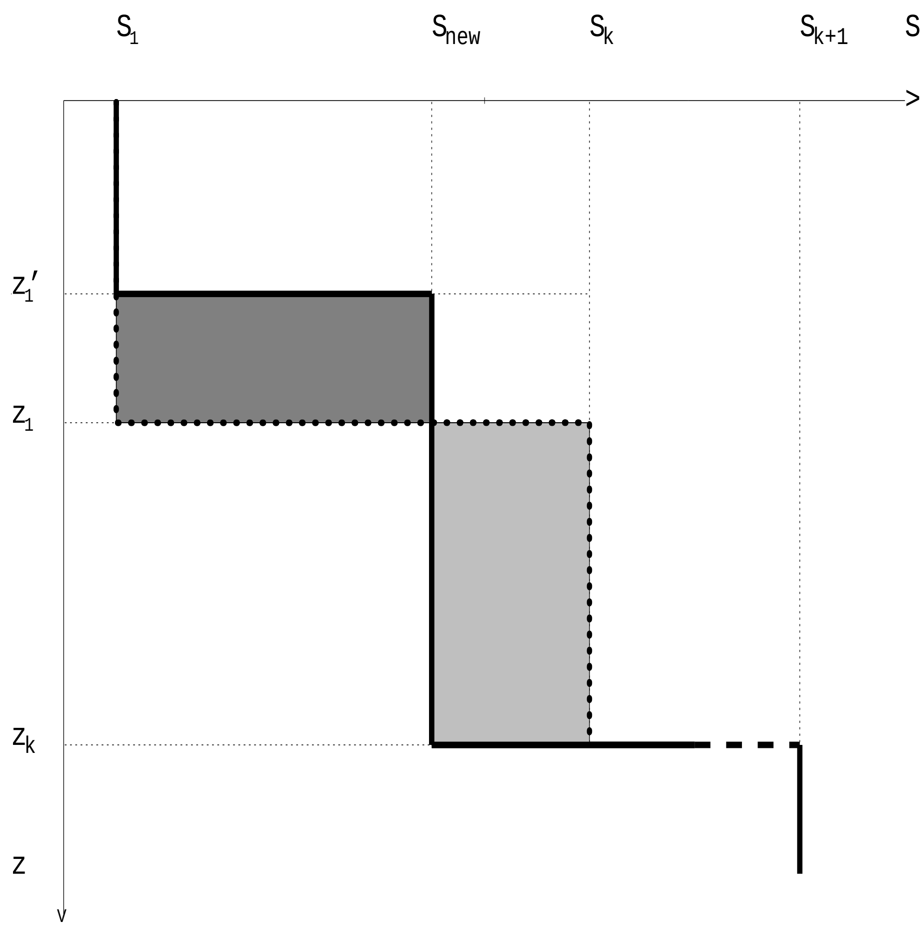


Figure 14: *Conservation of salt in updating the mixed layer.*

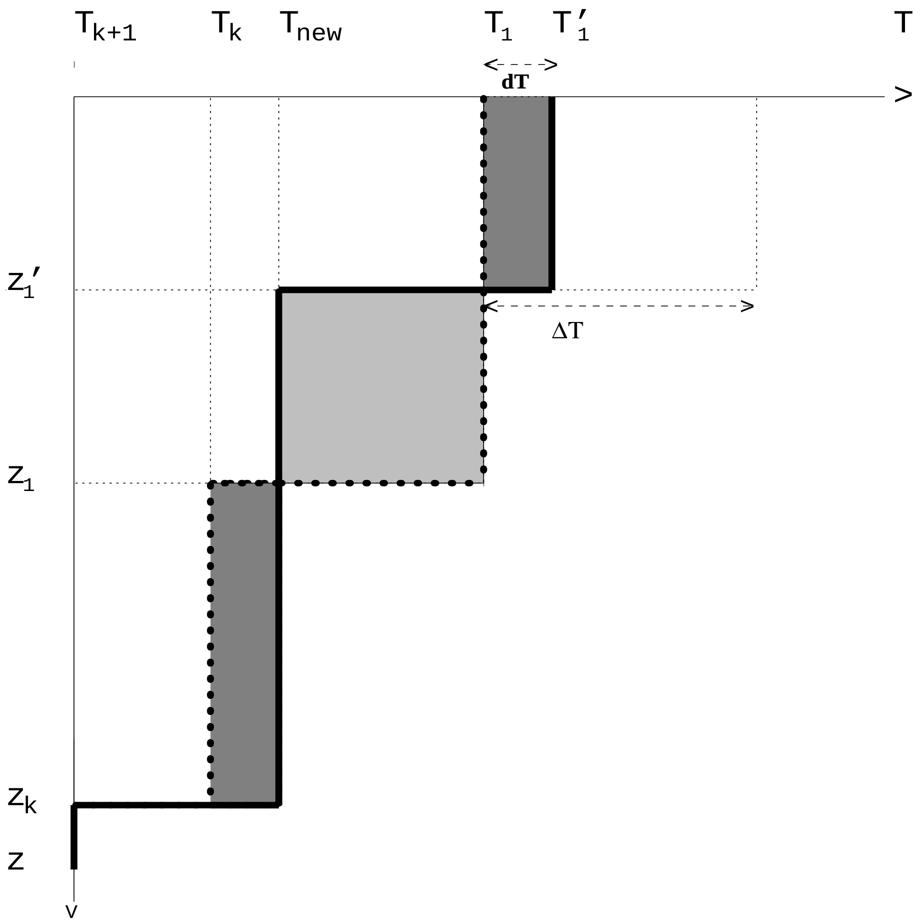


Figure 15: *Illustration of the mechanism of updating the mixed layer in the case when 100% detrainment is possible.*

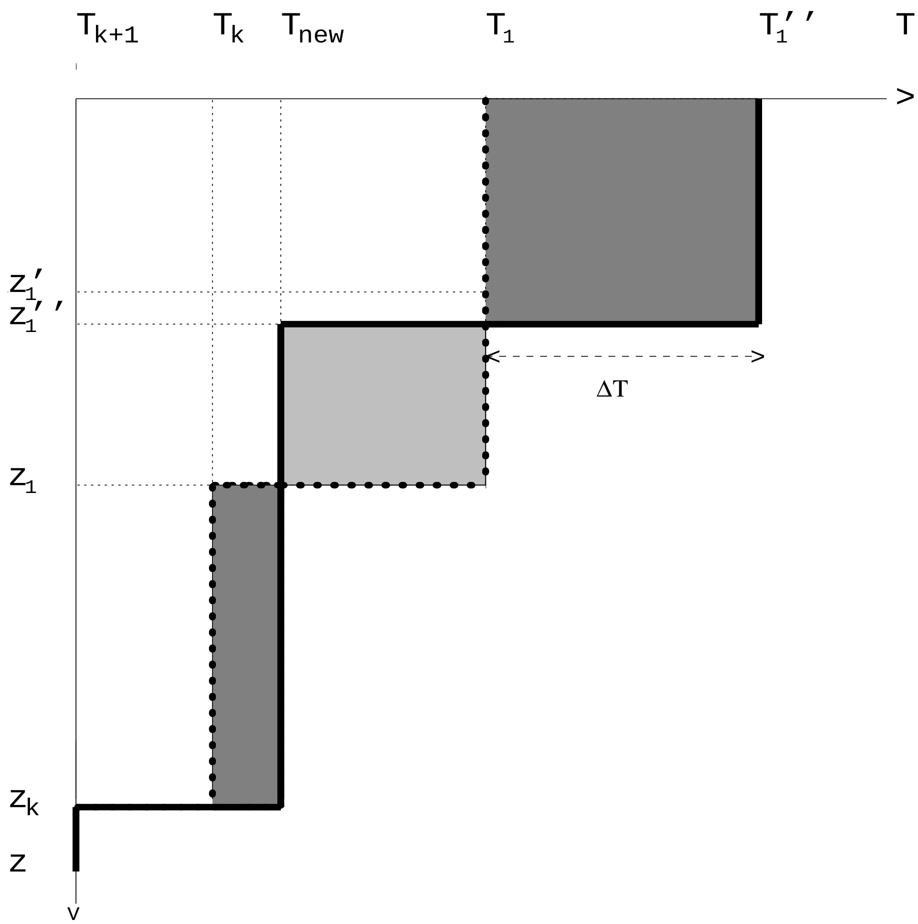


Figure 16: *Illustration of the mechanism of updating the mixed layer in the case when partial detrainment is possible.*

12.4 Variables

12.4.1 Identification

Notation in the theory of Gaspar (1988)	Notation in **mxkrtm.f**
h/l	`alf1`
h/l_p	`alf2`
A_p, S_p	`ape,???`
B	`buoyfl(i,j)`
C_4	`cc4`
C_{p1}, C_{p3}	`cp1,cp3`
h	`dpth`
a_1, a_2	`ea1,ea2`
$1/\lambda$	`ekminv`
$m_1, ..., m_5$	`em1,...,em5`
$\exp(h/L)$	`ex`
$1/L$	`obuinv`
$E\Delta t$	`turgen(i,j)`
u_*	`ustar(i,j)`

Notation in the theory of Bleck *et al.* (1992)	Notation in **mxkrtm.f**
a, b, c, d, e, f	`a,b,?,d,e,f`
a_0, a_1, a_2, a_3	`cc0,cc1,cc2,cc3`
$c_1, c_2, c_3, c_4, c_5, c_6, c_7$	`c1,c2,c3,c4,c5,c7`
$d(c_1 - \sigma_k)$	`c1msig`

12.4.2 Local variables

Subroutine mxkrtm

`delp`	Pressure variation at interfaces.
`i, j`	Array indices.
`k`	Layer index.
`kmax`	
`l`	Loop index.
`m, n`	Time step indices.
`q`	Relative variation of momentum in the mixed layer due to its deepening.
`sdot`	
`thk`	Augmentation of mixed layer thickness by turbulent diffusion.
`tndcys`	
`tndcyt`	
`tosal`	
`totem`	
`work(3)`	

Subroutine mxkrtmaj

`alf1`	Stability parameter h/l.
`alf2`	Stability parameter in the absence of rotation h/l_p.
`ape`	Term in the expression of entrainment.
`cc4`	Intermediate coefficient parameterizing turbulent effects.
`cp1, cp3`	Coefficients in the expression of the entrainment.
`c1msig`	Coefficient in the calculation of zeroes of the polynomial giving the new characteristics of the layer sub-adjacent to the mixed layer.

`dpth`	Transformation of the pressure difference in the mixed layer in thickness units (cm).
`ea1, ea2`	
`ekminv`	Inverse of the Ekman length.
`em1,...,em5`	Coefficients of the turbulent effects parameterization.
`ex`	Exponential of the Monin-Obukov stability parameter h/L.
`i, j`	Array indices.
`k`	Layer index.
`l`	Loop index.
`m, n`	Time step indices.
`obuinv`	Inverse of the Monin-Obukov length.
`pnew`	Mixed layer thickness (cm).
`sdot`	
`spe`	Term in the expression of the entrainment.
`thknss`	Layer thickness.
`ustar3`	`ustar(i,j)**3`

Subroutine mxkrtmbj

`a, b, d, e, f`	Coefficients in the expressions of new characteristics T and S of layer k.
`c1msig`	Coefficient in the calculation of zeroes of the polynomial giving the new characteristics of the layer sub-adjacent to the mixed layer.
`cc0,...,cc4`	Intermediate coefficients parameterizing turbulent effects.
`ccubim`	
`ccubq`	
`ccubqr`	
`ccubr`	

`ccubr1`	
`ccubs1`	
`ccubs2`	
`dpn`	
`dsaln`	Variation ΔS of salinity in the mixed layer.
`dtemp`	Variation ΔT of temperature in the mixed layer.
`i, j`	Array indices.
`k`	Layer index.
`l`	Loop index.
`m, n`	Time step indices.
`num`	Number of real roots.
`pnew`	Mixed layer thickness (cm).
`root`	
`root1`	
`root2`	
`root3`	
`s1`	
`sdot`	
`sdp(idm)`	
`smx1`	
`sn`	
`snew`	Salinity of the mixed layer.
`s_up`	
`t1`	
`tdp(idm)`	
`thknss`	

`tmx1`	
`tn`	
`tnew`	Temperature of the mixed layer.
`z`	Real root of the polynomial.

12.5 Procedures

Functions	`ccubq`, `ccubr`, `ccubqr`, `ccubs1`, `ccubs2`, `ccubrl`, `ccubim`, `root`, `root1`, `root2`, `root3`
Subroutines	`mxkrtm, mxkrtmaj, mxkrtmbj`

13 Convection - Kraus-Turner or MICOM Mode : convch.f or convcm.f

When HYCOM is run using the Kraus Turner model or in MICOM-compatibility mode, convection is performed by the model. The variations of mixed layer characteristics result from two principal steps :

1. Explicitly taking into account ocean-atmosphere exchanges (*c.f.* § 12) ;
2. Modeling advection with the velocity field (*c.f.* § 12). Consequently, an inversion at the base of the mixed layer is not excluded. Phase 1 is optional. When radiative exchanges and heat turbulence are not considered, the mechanical effect due to the wind can always be treated through the surface Reynolds tension (*c.f.* § 4). The advection step is systematically used. It can, by itself, generate a surface inversion.

13.1 Usage

Within subroutine `convcm`, for each point, this possible inversion is diagnosed, its vertical extension is determined, and it is addressed by conserving the content of heat and salt in the layers concerned.

13.1.1 Order of operations

The first step consists of traversing the vertical and testing the value of the density deviation $\rho_k - \rho_1$. When it is negative, the situation is unstable and is addressed by mixing layer k with the layer above it, conserving heat and salt. Then infer the density ρ_1' which serves as a new reference to test for a possible inversion with layer $k+1$.

```
[***** ADD SOURCE CODE HERE *****]
```

The second step consists of conserving the momentum by integrating it over the new surface layer.

```
[***** ADD SOURCE CODE HERE *****]
```

The final operation consists of storing this new vertical distribution.

```
[***** ADD SOURCE CODE HERE *****]
```

13.1.2 Flowchart

∗ INSERT FLOWCHART HERE ∗

13.2 Variables

13.2.1 Identification

Notation in the theory	Notation in **convcm.f**
* INSERT TEXT HERE *	* INSERT TEXT HERE *

13.2.2 Local variables

Subroutine convch

`coluin(idm)`	
`i, j`	Array indices.
`iter`	Iteration loop index.
`itmax`	
`k, kp, ks`	Layer indices.
`l`	Loop indices.
`m, n`	Time step indices.
`q`	
`sal`	
`siglo`	
`sigup`	
`tem`	
`thet`	
`tre`	
`ulo`	
`uup`	
`vlo`	
`vup`	

Subroutine convcm

`delp`	Pressure variation at interfaces.
`dthet`	
`i, j`	Array indices.
`k`	Layer index.
`l`	Loop index.
`m, n`	Time step indices.

13.3 Procedures

Subroutines `convch, convcm`

14 MICOM Mode - Diapycnal mixing : diapf3.f

14.1 Formalism and numerical techniques

14.1.1 Turbulent diffusion

Using θ to represent the vertical distribution of density in the ocean, the standard concept of turbulent diffusion is used to express the turbulent fluxes associated with this variable of state :

$$\overline{w'\theta'} = K_H \partial\theta/\partial z \tag{253}$$

where K_H is the diffusivity. w' and θ' represent the turbulent fluctuations in the vertical component of velocity and in density in the Reynolds sense. Without a source term, diffusion, or horizontal advection, the evolution equation of the variable θ at depth z is then written as :

$$\left(\frac{\partial\theta}{\partial t}\right)_z = \frac{\partial}{\partial z}\left(K_H \frac{\partial\theta}{\partial z}\right) \tag{254}$$

The local tendency of the variable θ then results simply from the divergence of vertical turbulent flux associated with this parameter.
Set $F = K_H \partial\theta/\partial z$. Under the assumption that the turbulent flux is zero at the surface ($F_s = 0$) and at the bottom ($F_b = 0$), integrating equation (254) over the vertical conserves the quantity θ in a column of water.
In the hypothetical case where the flux at the boundary is zero, since a source term such as solar radiation is not taken into account, the vertical component of the turbulent flux F can result only from the initial profile of the parameter θ. Over the profile at time t, this flux will cause a vertical displacement of the isocline θ, situated at the level z which is described by the equation :

$$\left(\frac{\partial z}{\partial t}\right)_\theta = -\frac{\partial F}{\partial \theta} \tag{255}$$

The variable θ is assumed to be stepwise constant in the vertical :

$$\theta(z) = \theta_k \quad \text{for} \quad z_{k-1/2} \le z \le z_{k+1/2}$$

The indices $k - 1/2$ and $k + 1/2$ are the positions of the interfaces above and below the layer k.
In a layer configuration, the user wants to express the form (255) in centered finite differences considering that the vertical distribution of turbulent flux can also be represented by a succession of layers of flux F_k centered at z_k. The user then has :

$$\left(\frac{\partial F}{\partial \theta}\right)_{k+1/2} = -\frac{(F_{k+1} - F_k)}{\theta_{k+1} - \theta_k} \tag{256}$$

The layer scheme therefore imposes the knowledge of the divergence of flux at successive interfaces. To calculate a vertical distribution of fluxes centered at z_k an artificial gradient

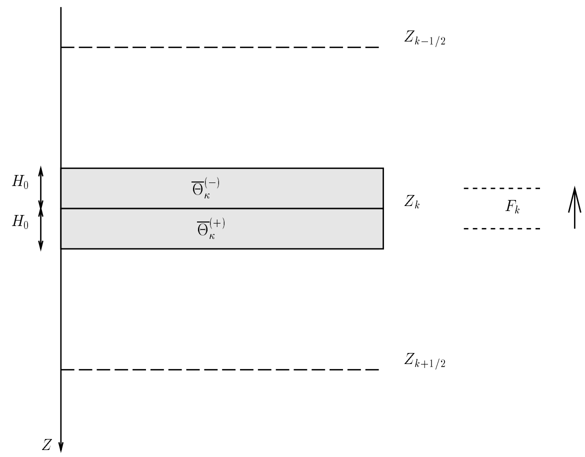


Figure 17: *Illustration of the calculation of turbulent flux* F_k.

on the interior of each layer k is created.
The diapycnic mixing algorithm implemented in MICOM mode is a simplified version of the algorithm developed by Hu (1991).

14.1.2 Turbulent heat flux

To illustrate the mechanism of diapycnic mixing, imagine that the turbulent instability due to the temperature discontinuity at the interface between two layers manifests itself in the appearance of a thin intermediate mixed layer. At the upper limit of the layer k, at the coordinate $z_{k-1/2}$, the temperature of this micro-layer is posited to be the same as θ'_{sup} such that :

$$\theta'_{sup} = \frac{1}{2}\left(\theta_k + \theta_{k-1}\right) \tag{257}$$

Then, the continuous function is introduced by breaking up :

$$\theta'(z) = \frac{1}{z_{k+1/2} - z_{k-1/2}}\left[\frac{\theta_k + \theta_{k-1}}{2}(z_{k+1/2} - z) + \frac{\theta_k + \theta_{k+1}}{2}(z - z_{k-1/2})\right] \tag{258}$$

for : $z_{k-1/2} \leq z \leq z_{k+1/2}$. The distribution $\theta'(z)$ linearly varies the influence of the boundary conditions of the diapycnic mixing on the interior of layer k in such a manner that the influence of the upper boundary condition is zero at the lower interface and *vice versa*. The surface mixed layer is represented at the surface by introducing the upper discontinuity : $\theta'(z) = \theta_1$ for $z \leq z_{3/2}$.

The new distribution $\theta'(z)$ is integrated over two intervals of finite thickness H_0, situated respectively above and below the middle point of layer k of depth $z_k = 1/2\left(z_{k-1/2} + z_{k+1/2}\right)$.

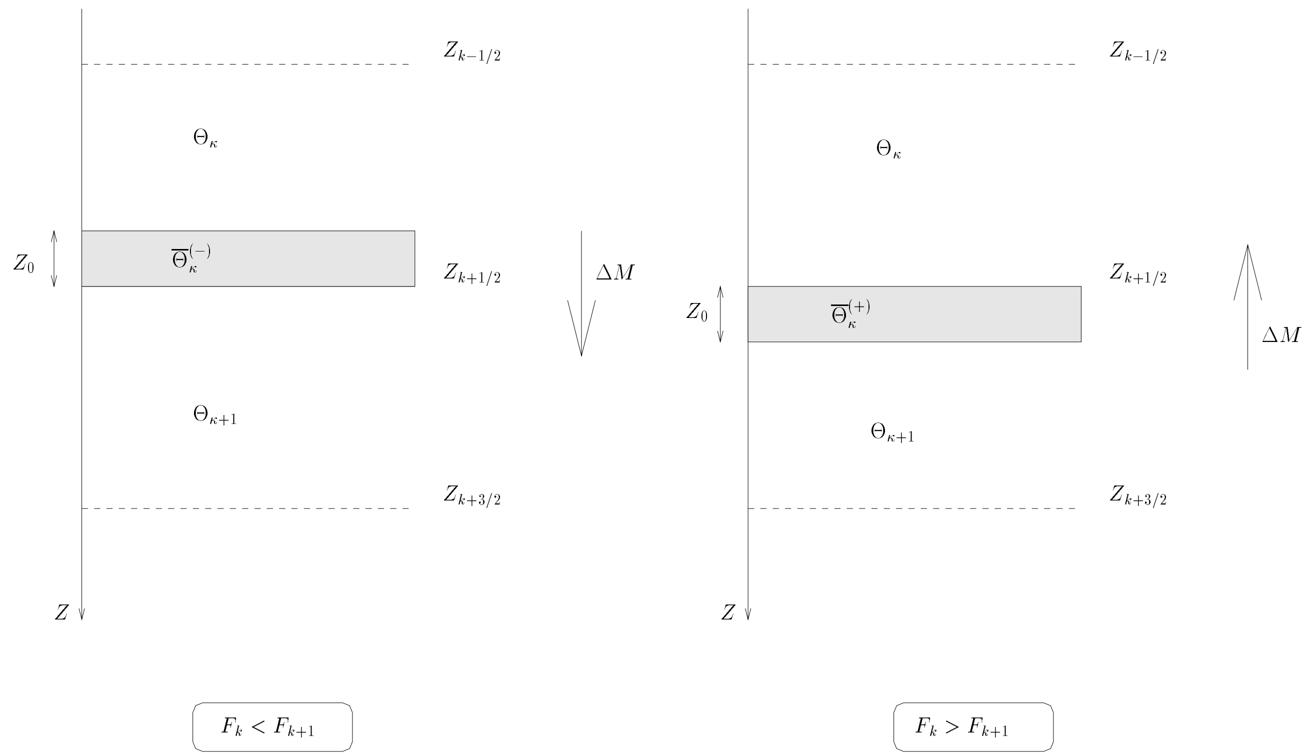


Figure 18: *Illustration of the calculation of the divergence of the turbulent flux F_k.*

The two new variables are obtained (figure 17):

$$\overline{\theta}_k^{(-)} = \frac{1}{H_0} \int_{z_k - H_0}^{z_k} \theta' \; dz \tag{259}$$

$$\overline{\theta}_k^{(+)} = \frac{1}{H_0} \int_{z_k}^{z_k + H_0} \theta' \; dz \tag{260}$$

The flux in k is then defined as

$$F_k = K_H \frac{\overline{\theta}_k^{(+)} - \overline{\theta}_k^{(-)}}{H_0} \tag{261}$$

Note that for layers of thickness greater than $2H_0$, the following relation is given :

$$F_k = K_H \frac{\theta_{k+1} - \theta_{k-1}}{2(z_{k+1/2} - z_{k-1/2})}$$

14.1.3 Numerical implementation

During a time interval Δt, after having been identified with the relation (255), the form (256) expresses that a column of water with a thickness of $\Delta z = (\partial z / \partial t) \, \Delta t$ is going to

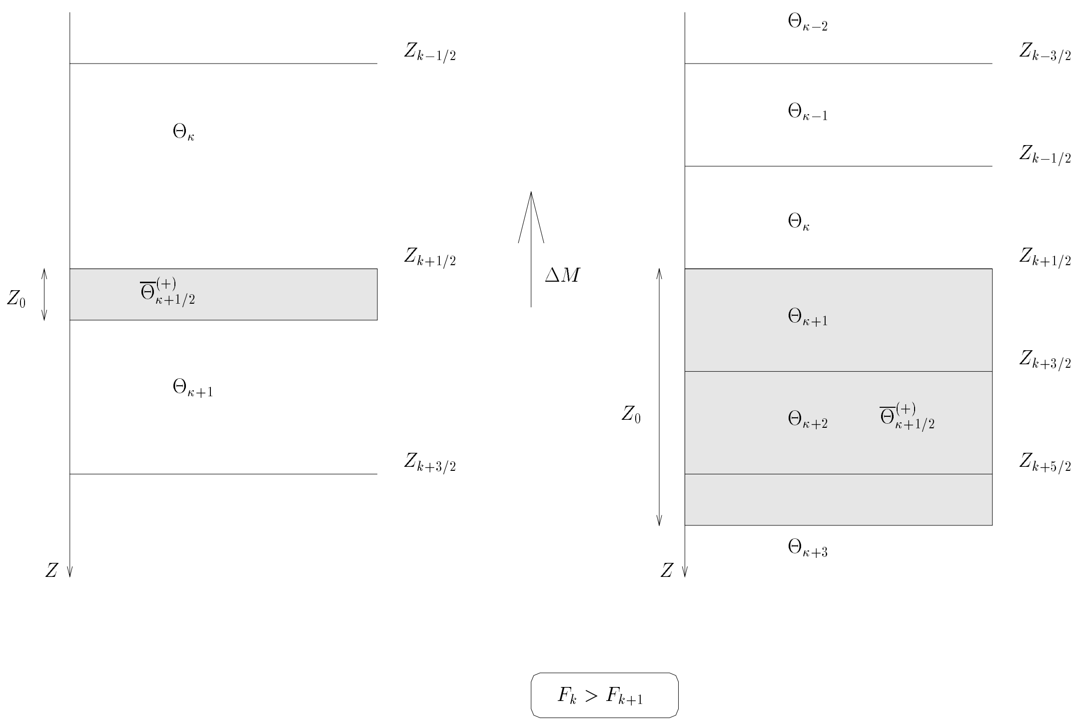


Figure 19: *Illustration of the generalization of Hu (1991) in the case of an ascending mass transfer.*

see its density change from θ_{k+1} to θ_k (or *vice-versa* depending on the sign of $\partial F/\partial\theta$). In other words, in the deep regime, during Δt, the diapycnic mixing causes a vertical transfer of the quantity Δz of the layer k to the layer $k+1$. Numerically, this transfer can pose problems if Δz exceeds the thickness of layer k. This fact most concerns layers of small thickness. It can also appear during a transfer of water from the mixed layer to the first layer below where the increment $(\theta_k - \theta_{k+1})$ is often small.

Consider the case $F_k > F_{k+1}$. The user will want to do a mass transfer from layer $k+1$ to layer k. To generalize the expression (256) Hu (1991) substitutes θ_{k+1} with the average expression (figure 18) :

$$\overline{\theta}^{(+)}_{k+1/2} = Z_0^{-1} \int_{z_{k+1/2}}^{z_{k+1/2}+Z_0} \theta \, dz \tag{262}$$

where the discrete distribution $\theta(z)$ reappears.

In this case, the general form (256) becomes :

$$\left(\frac{\partial F}{\partial \theta}\right)_{k+1/2} = -\frac{(F_{k+1} - F_k)}{\overline{\theta}^{(+)}_{k+1/2} - \theta_k} \tag{263}$$

In identifying this last expression with the relation (255), it is clear that the displacement Δz of the interface $k+1/2$ represents the thickness of a layer whose density evolves from $\overline{\theta}^{(+)}_{k+1/2}$ to θ_k. Only two alternatives are possible (figure 19) :

1. if layer k+1 is thicker than Z_0, then $\overline{\theta}^{(+)}_{k+1/2} = \theta_{k+1}$. Equation (263) takes the form of (256). More frequently, to carry out a transfer into the layer k, it is possible to extract the ascending mass only from layer $k+1$;
2. if layer $k+1$ is thinner than Z_0, the approach of Hu says that the layers $k+1$, $k+2$,... contribute to the total displacement Δz in the same proportions in which they contribute to the integral (262).

Concerning the case where $F_k < F_{k+1}$, the form (256) is then substituted with :

$$\left(\frac{\partial F}{\partial \theta}\right)_{k+1/2} = -\frac{(F_{k+1} - F_k)}{\theta_{k+1} - \overline{\theta}^{(-)}_{k+1/2}} \tag{264}$$

where,

$$\overline{\theta}^{(-)}_{k+1/2} = Z_0^{-1} \int_{z_{k+1/2}-Z_0}^{z_{k+1/2}} \theta \, dz. \tag{265}$$

The mass of the column of water of thickness Δz and of density $\overline{\theta}^{(-)}_{k+1/2}$ transferred from layer k into layer $k+1$ comes from layers k, k-1, k-2, *etc.* in the same proportions in which they contribute to the integral (265).

14.2 Usage

The numerical calculation of diapycnic mixing is carried out by the subprogram :

```
[***** ADD SOURCE CODE HERE *****]
```

14.2.1 Order of operations

For each point of the calculation, the index of the first isopycnic layer of which the specific volume is strictly larger than that of the mixed layer is determined. The result is stored in the array `klist(i,j)`. Then, the user must evaluate an adequate integration thickness H_0. The result is stored in the array `dpmx(i,j)`. In the first step, the operation consists of finding two limiting values : `h0lo` in the surface and `h0hi` from a fixed pressure `delp`. These values are bounded by 1/4 of the total water height. Between these two depths, the final value varies linearly.

```
[***** ADD SOURCE CODE HERE *****]
```

Then, explore the part above in the column of water to find layers which will comprise the interval Z_0 and calculate $\overline{\theta}_{k+1/2}^{(-)}$ over this slice of water. Next, explore the part below in the column of water to find the layers which will comprise the interval Z_0 and calculate $\overline{\theta}_{k+1/2}^{(+)}$ over this slice of water. The results are stored in the arrays `thup(i,k)` and `thdn(i,k)`. The thickness Z_0 is also put to the initial value H_0. Similarly if it is small (10 meters for ocean applications), it is possible that the instantaneous thickness of a layer be even smaller, or zero. The algorithm accounts for this.

```
[***** ADD SOURCE CODE HERE *****]
```

The next operation consists of determining the two temperatures $\overline{\theta}_k^{(-)}$ and $\overline{\theta}_k^{(+)}$ defined by the expressions (259) and (260). For this, the whole column of water is inspected to calculate a bulk characteristic temperature of each of two slices of water situated respectively above and below an interior point in layer k. The thickness `dpmx(i,j)` of each of the two intermediate layers is limited by half of the vertical extension of the layer studied. It is also necessary to account for the fact that a layer can have zero size.

```
[***** ADD SOURCE CODE HERE *****]
```

The corresponding flux is calculated by using the relation (261).

[***** ADD SOURCE CODE HERE *****]

In the last step, the generalization of Hu (1991) is used first :

a) if $F_{k-1} - F_k < 0$, then substitute θ_{k-1} by $\overline{\theta}^{(-)}_{k-1/2}$;

b) if $F_{k-1} - F_k \geq 0$, then substitute θ_k by $\overline{\theta}^{(+)}_{k-1/2}$

then, calculate the corresponding vertical increment Δz. The result is stored in the array `sdot(i,j)`. The maximal positive excursion of the lower interface is bounded by the ocean depth. A negative displacement of the upper interface is limited by 90% of the initial immersion of this interface.

[***** ADD SOURCE CODE HERE *****]

Finally, extract the mass of the layers above or below so as to determine the new thickness of each layer concerned.

[***** ADD SOURCE CODE HERE *****]

To end, the user must calculate the interface pressures.

[***** ADD SOURCE CODE HERE *****]

14.2.2 Flowchart

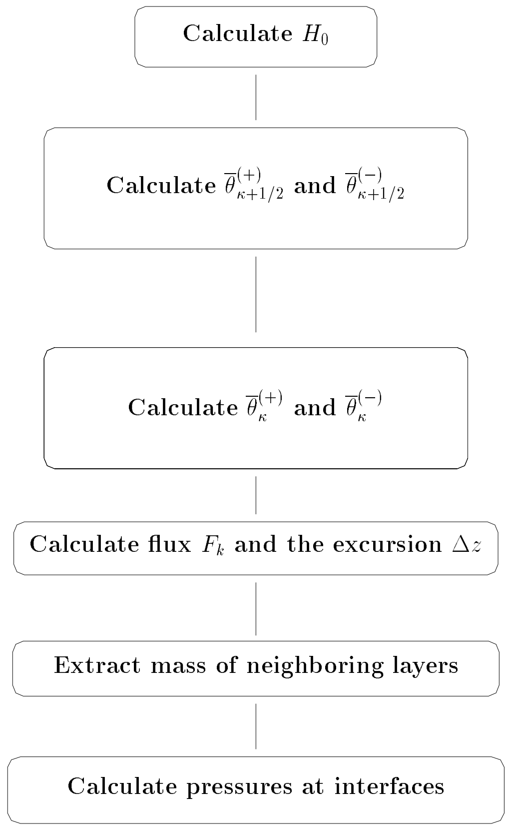


Figure 20: *Order of the mixed layer calculations in HYCOM 2.0.01*

14.3 Variables

14.3.1 Identification

Notation in the theory of Hu (1991)	Notation in **diapfl.f**
H_0	`????`
Z_0	`delp`
$\overline{\theta}_k^{(-)}$	`????(i,1)`
$\overline{\theta}_k^{(+)}$	`????(i,1)`
$\overline{\theta}_{k-1/2}^{(-)}$	`????(i,k)`
$\overline{\theta}_{k+1/2}^{(+)}$	`????(i,k)`

14.3.2 Local variables

Subroutine diapf3

`j`	Array index.
`m, n`	Time step indices.

Subroutine diapf3j

`alfa`	
`beta`	
`ennsq`	
`flngth(idm,kdm)`	
`flxl(idm,kdm)`	
`flxu(idm,kdm)`	
`froglp`	
`i, j`	Array indices.
`k`	Layer index.
`l`	Loop index.
`m, n`	Time step indices.

`kmax(idm)`	
`kmin(idm)`	
`pdot(idm,kdm)`	
`q`	
`salt`	
`small`	
`smax`	
`smin`	
`sold(idm,2)`	Old salinity.
`trflx(idm,kdm+1)`	
`trmax`	
`trmin`	
`trold(idm,2)`	

14.4 Procedures

Subroutines `diapf3, diapf3j`

15 Calculational grid

Both HYCOM and MICOM use the “C” grid, but use different horizontal meshes. In the MICOM mesh, the positive x direction was southward and the positive y direction was eastward. The HYCOM mesh was converted to standard Cartesian coordinates, with the x axis pointing eastward and the y axis pointing northward. The HYCOM mesh is illustrated below for pressure (P), velocity component (U and V), and vorticity (Q) grid points. This case is for 7 x 7 pressure grid points. The grid meshes for the other variables have 8 x 8 grid points. All fields in this case would be dimensioned 8 x 8, with the eighth row and column unused for variables on pressure grid points.

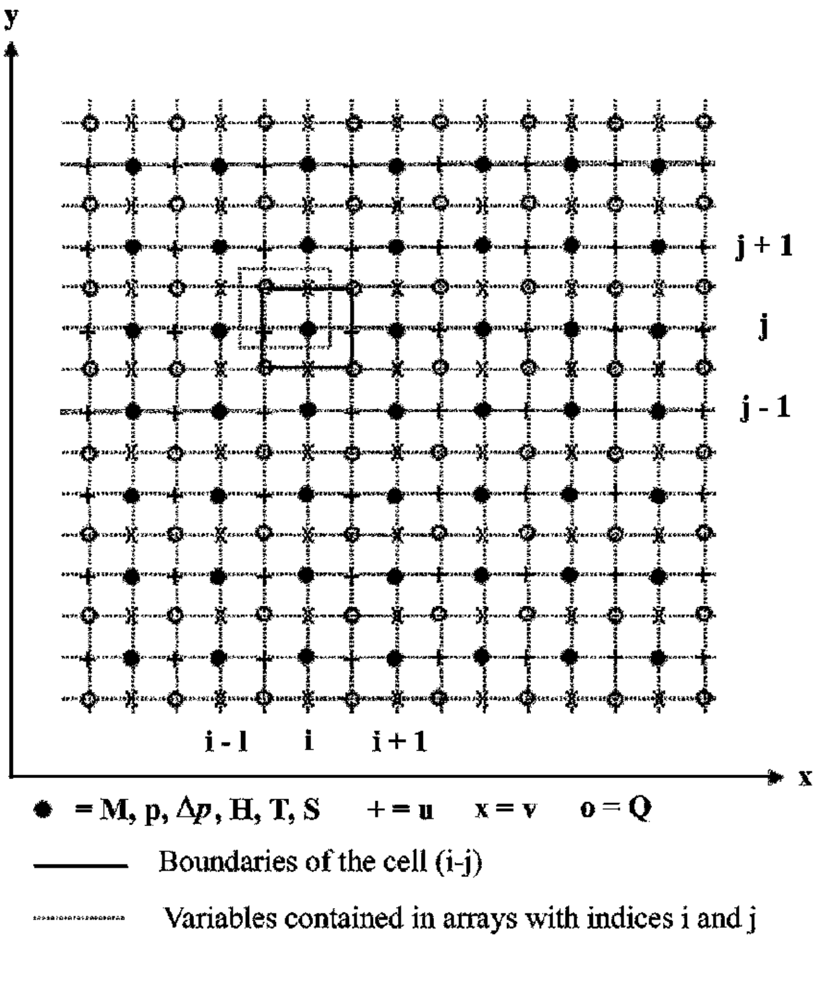


Figure 21: *Distribution of variables on an Arakawa C grid*

16 Boundary conditions in HYCOM

HYCOM 2.0.01 is equipped with two types of boundary conditions: Newtonian relaxation in sponge layers, and full open-ocean boundary conditions.

16.1 Relaxation Boundary Conditions

HYCOM contains a simple Newtonian relaxation scheme that can be used for sponge boundary zones, or more generally for relaxation to climatology within any model subdomain specified by the user. Within relaxation boundary zones, temperature, salinity, and vertical coordinate pressure levels are updated as follows for each time step:

$$\begin{aligned} T_{t+1}^k &= T_t^k + \Delta t \mu \left(\widehat{T}_t^k - T_t^k \right) \\ S_{t+1}^k &= S_t^k + \Delta t \mu \left(\widehat{S}_t^k - S_t^k \right) \\ p_{t+1}^k &= p_t^k + \Delta t \mu \left(\widehat{p}_t^k - p_t^k \right), \end{aligned} \tag{266}$$

where the hat denotes Levitus climatology, k is the layer or interface number, and μ^{-1} is the relaxation time scale. The user specifies the values of μ^{-1} at each grid point, setting it to nonzero values where relaxation is to be performed. This results in a two-dimensional mask that defines the relaxation zones. Software is provided with HYCOM that first horizontally interpolates Levitus climatology to model grid points at the original z levels, then transforms these vertical profiles to isopycnic coordinates at each model grid point. Thus, the profiles of $\widehat{T}$, $\widehat{S}$, and $\widehat{p}$ used in (266) are isopycnic beneath the surface mixed layer.

When HYCOM is run with isopycnic vertical coordinates (MICOM mode), temperature and salinity are both relaxed in the non-isopycnic mixed layer (layer 1) while salinity only is relaxed in deeper layers with temperature diagnosed from the equation of state to preserve the isopycnic reference density. When HYCOM is run with hybrid vertical coordinates, both temperature and salinity are relaxed in the upper n_{hyb} layers, where n_{hyb} is the user-specified number of hybrid layers, and salinity alone is relaxed in deeper layers with temperature diagnosed from the equation of state. When HYCOM is run with hybrid coordinates, all pressure interfaces are relaxed to climatology. When the model is run with isopycnic coordinates, all interfaces except 2 are relaxed to avoid adjusting the mixed layer base. Of course, all interfaces greater than 2 are prevented from becoming shallower than interface 2.

16.2 Open Boundary Conditions

Due to the fundamental ill posedness of the open boundary value problem in hydrostatic models (*e.g.* Oliger and Sundstrom, 1978), limited-area modeling with the primitive equations is more an art than a science. Open boundary conditions that have worked reasonably well in MICOM, and that have been adapted for HYCOM, are discussed here.

The main features of the open boundary scheme are as follows:

1. No distinction is made between inflow and outflow boundaries.
2. Boundary conditions for the barotropic and baroclinic mode are formulated separately.
3. The well-posed boundary conditions developed by Browning and Kreiss (1982, 1986), which work well in single-layer, shallow-water models, are applied to the barotropic mode, specifically the pressure field and normal velocity component.
4. Barotropic tangential velocity components are prescribed.
5. Baroclinic velocities normal to the boundary, as well as total (barotropic plus baroclinic) mass fluxes, are prescribed.
6. Baroclinic tangential velocity components are nudged toward prescribed values.
7. Other boundary conditions for the baroclinic mode are applied not only at points directly on the boundary, but in a finite-width "sponge" zone. They include interface pressure nudging, damping of the tendency term in the continuity equation, and enhanced viscosity in the momentum equations.

16.2.1 No distinction between inflow and outflow boundaries

The first approach listed above is taken in recognition of the fact that, regardless of the direction of the physical flow, information generally passes through the boundary in both directions. Making a distinction between inflow and outflow boundaries is therefore justified only with regard to advection of material properties, such as temperature, salinity, and potential vorticity.

16.2.2 Well-posed boundary conditions

Browning and Kreiss (1982, 1986) suggest that well-posed boundary conditions for modeling fluid flow in open domains can be derived from the theory of characteristics. In the case of two independent variables x, t, characteristics are curves $x(t)$, which, if used as coordinate axes, reduce a set of coupled p.d.e.'s to a set of uncoupled o.d.e.'s. They arise during attempts to construct, through Taylor series expansion, the solution of a system of p.d.e.'s in the vicinity of a boundary curve in x, t space along which the dependent variables and their normal derivatives are prescribed. Specifically, characteristics are curves that are unsuitable as boundary curves because the Taylor series coefficients cannot be uniquely determined from conditions prescribed along these curves.

Consider a simple hyperbolic system describing gravity wave propagation in a shallow fluid layer moving at speed U:

$$\begin{aligned} u_t + U_0 u_x + g h_x &= 0 \\ h_t + U_0 h_x + H u_x &= 0. \end{aligned} \tag{267}$$

There are two sets of characteristics in this problem; their respective slopes in the x, t plane are

$$\left(\frac{\partial x}{\partial t}\right)_{char} = U_0 \pm c, \tag{268}$$

where $c = \sqrt{gH}$ is the gravity wave phase speed. Following the characteristics in x, t space is equivalent to tracking gravity waves that propagate upstream and downstream through the moving fluid.

The o.d.e.'s obtained by transforming x, t derivatives in the set of p.d.e.'s in (267) into derivatives taken along characteristics are

$$\begin{aligned} u_s + c_1 h_s &= 0 \\ u_s - c_1 h_s &= 0, \end{aligned} \tag{269}$$

where $c_1 = \sqrt{g/H}$ and subscript s denotes differentiation along a characteristic. After integration over s, these equations state that $u + c_1 h$ and $u - c_1 h$, respectively, are constant along the two sets of characteristics. The solution at a given point x, t can therefore be constructed by superimposing u, h combinations carried along the two characteristics intersecting at x, t. This applies to interior as well as boundary points.

Let $c \gg U_0 > 0$. On the upstream or "western" boundary, two characteristics, $U_0 + c$ going from west to east and $U_0 - c$ going from east to west, bring in information from the exterior (provided by observations or a coarse-mesh model) and from the interior, respectively. The combination of these two characteristics yields the final boundary values of u and h. If superscript o denotes values obtained from the outer coarse-mesh model or data, i denotes values from the inner, fine-mesh model, and $*$ denotes the actual boundary values, then

$$\begin{aligned} u* + c_1 h* &= u^o + c_1 h^o \\ u* - c_1 h* &= u^i - c_1 h^i. \end{aligned} \tag{270}$$

This is a system of two equations for the two sought-after boundary values $u*, h*$. The solutions are

$$\begin{aligned} u* &= \tfrac{1}{2}\left[u^o + u^i + c_1\left(h^o - h^i\right)\right] \\ h* &= \tfrac{1}{2}\left[h^o + h^i + c_1^{-1}\left(u^o - u^i\right)\right]. \end{aligned} \tag{271}$$

Boundary conditions for the case $U_0 < 0$ and the eastern boundary are analogous. If the model contains thermodynamic variables satisfying conservation laws dominated by advection processes, the method of characteristics suggests that these variables should be updated by coarse mesh fields or data at inflow points, and from within the model at outflow points.

16.2.3 Barotropic and baroclinic velocities

Prescribing the mass flux across boundaries is the most direct way to stabilize the time-mean circulation in subbasins forced by strong inflow/outflow. Nudging is performed by replacing a grid point value ϕ by a linear combination of ϕ and a prescribed value ϕ_b:

$$\phi_{new} = (1 - w)\,\phi + w\phi_b. \tag{272}$$

If nudging is performed in a finite-width sponge zone, the weight w should gradually increase from zero in the interior of the domain to a finite value ≤ 1 at the boundary. The width of the sponge zone where $w > 0$ and the rate at which w increases toward the boundary is best determined by experimentation.

The same goes for the viscosity enhancement factor and the damping factor applied to the layer thickness tendency. The damping factor should increase from near-zero at the domain boundary to a value of 1 at the inner edge of the sponge zone. Viscosity may be increased stepwise to as much as 5 or even 10 times its value outside the sponge zone.

17 Equation of state and Related Issues

17.1 Equation of state

The equation of state embedded in HYCOM is the approximation to the UNESCO equation of state described by Brydon *et al.* (2001). At a given reference pressure level p, the density in sigma units is given by a seven term polynomial function cubic in potential temperature θ and linear in salinity S:

$$\sigma(\theta, S, p) = C_1(p) + C_2(p)\theta + C_3(p)S + C_4(p)\theta^2 + C_5(p)S\theta + C_6(p)\theta^3 + C_7(p)S\theta^2.$$

One advantage of this simple polynomial representation is that it can be inverted to calculate $\theta(\sigma, S, p)$ and $S(\sigma, \theta, p)$.

Three sets of coefficients are provided in HYCOM for reference pressures of 0, 20, and 40 Mpa. The sigma values calculated with these sets of coefficients are referred to as σ_0, σ_2, and σ_4, respectively. If the user selects σ_0 to represent model vertical coordinates, the density structure will be represented reasonably well in the upper ocean. In the deep ocean, however, regions will exist where σ_0 does not increase monotonically with depth, causing model vertical coordinate interfaces to fold over. The user must consider this problem in choosing the proper set of density coordinates for model simulations.

17.2 Cabbeling

Cabbeling is not an issue when HYCOM is run in MICOM mode since there is no penetrating shortwave radiation, and since salinity alone is advected and diffused within isopycnic layers. When HYCOM is run with hybrid vertical coordinates, however, cabbeling is an issue because temperature and salinity are always mixed in the vertical and shortwave radiation can penetrate into the isopycnic coordinate domain. Horizontal advection and diffusion of temperature and salinity, and fluxes of temperature and salinity across vertical coordinates relocated by the hybrid coordinate algorithm can also contribute to cabbeling. When HYCOM is run with hybrid vertical coordinates, reliance is placed on the hybrid vertical coordinate adjustment algorithm to restore and maintain isopycnic conditions.

There are two ways by which the user can reduce the influence of cabbeling. HYCOM contains the option of horizontally advecting and diffusing salinity and density, or temperature and density, instead of temperature and salinity. HYCOM also has the option to flux salinity and density, or temperature and density, instead of temperature and salinity, across vertical coordinates relocated by the hybrid coordinate algorithm. For further information on the problems that can be caused by cabbeling, refer to Sections 3 and 9.

17.3 Thermobaric compressibility

HYCOM 2.0.01 is equipped with the algorithm of Sun *et al.* (1999) to account for the dynamical influence of thermobaric compressibility, or thermobaricity.

17.4 Usage

∗ INSERT TEXT HERE ∗

17.4.1 Order of Operations

∗ INSERT TEXT HERE ∗

18 Sub-programs

18.1 Functions

Calculation of the harmonic average of two variables `a` and `b` :

```
[***** ADD SOURCE CODE HERE *****]
```

Calculation of the latitude `alat` as a function of the distance `dist1` to the equator and of the size of the mesh `grid` in degrees of latitude :

```
alat(dist1,grid)=(2.*atan(exp(dist1*grid/radian))-pi/2.)
```

Calculation of the distance `dist` to the equator as a function of the latitude `alat` and of the size of the mesh `grid` in degrees of latitude :

```
dist(alat1,grid)=log(tan((2.*alat1+pi)/4.))*radian/grid
```

Determination of the depth at calculational points u and v :

```
[***** ADD SOURCE CODE HERE *****]
```

18.2 Initialization Subroutines

Table 3: Initialization Subroutines

File	Subroutine	Description
blkdat.f	blkdat	Initializes common variables.
	blkinr	Reads in one named real value on unit 99.
	blkini	Reads in one named integer value on unit 99.
	blkinl	Reads in one named logical value on unit 99.
inicon.f	inicon	Sets all initial values to zero.
inikpp.f	inikpp	Initialize Large, Mc.Williams, Doney KPP vertical mixing scheme.

18.2.1 Subroutine BLKDAT

BLKDAT initializes common variables. It contains no arguments or common blocks.

18.2.2 Subroutine BLKINR

BLKINR reads in one named real value on unit 99.

Calling Sequence: Subroutine blkinr(rvar, cvar, cfmt)

Data Declaration: Character cvar, cfmt
Real rvar

Arguments: rvar
cvar
cfmt

18.2.3 Subroutine BLKINI

BLKINI reads in one named integer value on unit 99.

Calling Sequence: Subroutine blkini(ivar, cvar)

Data Declaration: Character cvar
Integer ivar

Arguments: ivar
cvar

18.2.4 Subroutine BLKINL

BLKINL reads in one named logical value on unit 99.

Calling Sequence: Subroutine blkinl(lvar, cvar)

Data Declaration: Character cvar
Logical lvar

Arguments: lvar
cvar

18.2.5 Subroutine INICON

INICON performs mass field initialization.

Calling Sequence: Subroutine inicon(mnth)

Data Declaration: Integer mnth

Arguments: mnth

18.2.6 Subroutine INIKPP

INIKPP initializes Large, MC Williams, Doney KPP vertical mixing scheme. It contains no arguments.

Common Blocks: kppltr

18.3 Bathymetry Subroutines

Table 4: Bathymetry Subroutines

File	Subroutine	Description
bigrid.f	bigrid	Determines boundaries for the four points which bound the mesh.
	indxi, indxj	Determines i/j loop indices corresponding to a land/sea mask.
geopar.f	geopar	Set up model parameters related to geography.

18.3.1 Subroutine BIGRID

BIGRID sets loop bounds for irregular basin in c-grid configuration.

Calling Sequence: Subroutine bigrid(depth, util1, util2, util3)

Data Declaration: Real depth, util1, util2, util3

Arguments:

depth	Basin depth array, zero values indicate land.
util1	
util2	
util3	

18.3.2 Subroutine INDXI

INDXI determines i loop index corresponding to a land/sea mask.

Calling Sequence: Subroutine indxi(ipt, if, il, is)

Data Declaration: Integer ipt, if, il, is

Arguments:

ipt	Input array. Contains 1 at grid point locations, 0 elsewhere.
if	Row index of first point in column j for k^{th} section.
il	Row index of last point in column j for k^{th} section.
is	Number of sections in column j (maximum: ms).

18.3.3 Subroutine INDXJ

INDXJ determines j loop index corresponding to a land/sea mask.

Calling Sequence: Subroutine indxj(jpt,jf,jl,js)

Data Declaration: Integer jpt, jf, jl, js

Arguments:

jpt	Input array. Contains 1 at grid point locations, 0 elsewhere.
jf	Column index of first point in row i for k-th section.
jl	Column index of last point in row i for k-th section.
js	Number of section in row i (maximum: ms).

18.3.4 Subroutine GEOPAR

GEOPAR sets up model parameters related to geography. It contains no arguments or common blocks.

18.4 Main HYCOM Subroutines

Table 5: Main HYCOM Subroutines

File	Subroutine	Description
barotp.f	barotp	Advance barotropic equations from baroclinic time level -m- to level -n-.
cnuity.f	cnuity	Continuity equation (flux-corrected transport version).
convec.f	convch	Convective adjustment.
	convcm	Entrain water lighter than mixed-layer water into mixed layer.
diapfl.f	diapf1	KPP-style implicit interior diapycnal mixing.
	diapf1aj	
	diapf1bj	
	diapf1aij	Diapycnal mixing, single i,j point (part A).
	diapf1uij	Diapycnal mixing, single i,j point, momentum at u grid points.
	diapf1vij	Diapycnal mixing, single i,j point, momentum at v grid points.
diapfl.f	diapf2	MICOM-style explicit interior diapycnal mixing for hybrid coordinates.
	diapf2j	Diapycnal mixing, single j-row.
	diapf3	MICOM-style explicit interior diapycnal mixing for isopycnal coordinates.
	diapf3j	Diapycnal mixing, single j-row.
hybgen.f	hybgen	Hybrid grid generator.
	hybgenaj	Hybrid grid generator, single j-row (part A).
	hybgenbj	Hybrid grid generator, single j-row (part B).
icloan.f	icloan	'Energy loan' ice model. No advection, no dynamics.
latbdp.f	latbdp	Apply lateral boundary conditions to barotropic flow field.
momtum.f	momtum	Momentum equation.
mxkpp.f	mxkpp	Large, Mc.Williams, Doney KPP vertical diffusion.
	mxkppaj	Calculate viscosity and diffusivity.
	mxkppbj	Final mixing at p points.
	mxkppcj	Final velocity mixing at u,v points.
	mxkppaij	KPP vertical diffusion, single j-row (part A).
	mxkppbij	KPP vertical diffusion, single j-row (part B). Performs the final vertical mixing at p points.

File	Subroutine	Description
	mxkppcij	KPP vertical diffusion, single j-row (part A), momentum at u grid points.
	mxkppcijv	KPP vertical diffusion, single j-row (part A), momentum at v grid points.
	wscale	Subroutine to compute turbulent velocity scales for KPP mixing scheme.
mxkrt.f	mxkrta	Kraus-Turner vertical diffusion for the bulk surface mixed layer.
	mxkrtaaj	
	mxkrtabj	
	mxkrtb	
	mxkrtbaj	
	mxkrtbbj	
mxkrtm.f	mxkrtm	
	mxkrtmaj	
	mxkrtmbj	
thermf.f	thermf	Calculates thermal balance.
	thermfj	
tsadvc.f	advem	Calculates third-order numerical scheme for advection of heat and salt.
	tsadvc	Calculates the horizontal equations of advection-diffusion of heat and salt.

18.5 Atmospheric Forcing Subroutines

Table 6: Atmospheric Forcing Subroutines

File	Subroutine	Description
dpthuv.f	dpthuv	Defines water depth (bottom pressure) at u,v points.
dpudpv.f	dpudpv	Defines layer depth at u,v points.
	dpudpvj	Defines layer depth at u,v points, single row.
	rdmonth	Reads a single array field from unit iunit.
	rdpall	Defines layer depth at u,v points.
	rdpall1	Copies field(:,:,2) into field(:,:,1), and reads a high frequency forcing field into field(:,:,2).
	rdforf	Read forcing functions for one month.
	rdrlax	Read relaxation fields for one month, monthly (clmflg = 12) or bi-monthly (clmflg = 6) data.
forfun.f	forday	Converts model day to "calendar" date (year,julian-day,hour).
	forfuna	Initializes input of atmospheric forcing fields.
	forfunh	High frequency atmospheric forcing field processing.
	forfunr	Initializes input of relaxation forcing fields.

18.5.1 Subroutine DPTHUV

DPTHUV defines water depth (bottom pressure) at u,v points. It contains no arguments or common blocks.

18.5.2 Subroutine DPUDPV

DPUDPV defines layer depth at u,v points.

Calling Sequence: Subroutine dpudpv(p, depthu, depthv, dpu, dpv)

Data Declaration: Real p, depthu, depthv, dpu, dpv

Arguments:
p
depthu
depthv
dpu
dpv

18.5.3 Subroutine DPUDPVJ

DPUDPVJ defines layer depth at u,v points, single row.

Calling Sequence: Subroutine dpudpvj(p, depthu, depthv, dpu, dpv, j)

Data Declaration:

Integer	j
Real	p, depthu, depthv, dpu, dpv

Arguments:

p
depthu
depthv
dpu
dpv
j

18.5.4 Subroutine RDMONTH

RDMONTH reads a single array field from unit iunit.

Calling Sequence: Subroutine rdmonth(field, iunit)

Data Declaration:

Integer	iunit
Real	field

Arguments:

iunit
field

18.5.5 Subroutine RDPALL

RDPALL defines layer depth at u,v points.

Calling Sequence: Subroutine rdpall(dtime0, dtime1)

Data Declaration:

Real	dtime0, dtime1

Arguments:

dtime0
dtime1

18.5.6 Subroutine RDPALL1

RDPALL1 copies field(:,:,2) into field(:,:,1), and reads a high frequency forcing field into field(:,:,2).

Calling Sequence: Subroutine rdpall1(field, dtime, iunit)

Data Declaration:

Real	field, dtime
Integer	iunit

Arguments: field
dtime
iunit

18.5.7 Subroutine RDFORF

RDFORF reads forcing functions for one month.

Calling Sequence: Subroutine rdforf(mnth, lslot)

Data Declaration: Integer mnth, lslot

Arguments: mnth
lslot

Common Blocks: rdforfi

18.5.8 Subroutine RDRLAX

RDRLAX defines layer depth at u,v points.

Calling Sequence: Subroutine rdrlax(month, lslot)

Data Declaration: Integer month, lslot

Arguments: month
lslot

Common Blocks: rdforfi

18.5.9 Subroutine FORDAY

FORDAY converts model day to "calendar" date (year, julian-day, hour).

Calling Sequence: Subroutine forday(dtime, yrflag, iyear, iday, ihour)

Data Declaration:

Integer	yrflag, iyear, iday, ihour
Real	dtime

Arguments:

dtime
yrflag
iyear
iday
ihour

18.5.10 Subroutine FORFUNA

FORFUNA initializes input of atmospheric forcing fields. It does not require any arguments.

Common Blocks: rdforfi

18.5.11 Subroutine FORFUNH

FORFUNH processes high frequency atmospheric forcing fields.

Calling Sequence: Subroutine forfunh(dtime)

Data Declaration:

Real	dtime

Arguments: dtime

18.5.12 Subroutine FORFUNR

FORFUNR initializes input of relaxation forcing fields. It does not contain any arguments.

Common Blocks: rdforfi

18.6 Matrix Inversion Subroutines

Table 7: Matrix Inversion Subroutines

File	Subroutine	Description
matinv.f	tridcof	Matrix inversion subroutine for implicit solution of vertical diffusion equation - tri-diagonal matrix.
	tridrhs	Matrix inversion subroutine for implicit solution of vertical diffusion equation - tri-diagonal matrix.
	tridmat	Matrix inversion subroutine for implicit solution of vertical diffusion equation - tri-diagonal matrix.

18.6.1 Subroutine TRIDCOF

TRIDCOF computes coefficients for tridiagonal matrix (dimension=kdm).

Calling Sequence: Subroutine tridcof(diff, tri, nlayer, tcu, tcc, tcl)

Data Declaration:

Integer	nlayer
Real	diff, tcu, tcc, tcl, tri

Arguments:

diff	Diffusivity profile on interfaces.
tri	Dt/dz/dz factors in a tridiagonal matrix.
nlayer	
tcu	Upper coefficient for (k-1) on k line of tridiagonal matrix.
tcc	Central coefficient for (k) on k line of tridiagonal matrix.
tcl	Lower coefficient for (k-1) on k line of tridiagonal matrix.

18.6.2 Subroutine TRIDRHS

TRIDRHS computes the right hand side of the tridiagonal matrix for scalar fields.

Calling Sequence: Subroutine tridrhs(h, yo, diff, ghat, ghatflux, tri, nlayer, rhs)

Data Declaration:

Integer	nlayer
Real	diff, h, yo, ghat, ghatflux, rhs, tri

Arguments:

diff	Diffusivity profile on interfaces.
tri	Dt/dz/dz factors in a tridiagonal matrix.

nlayer	
h	Layer thickness.
yo	Old profile.
ghat	Ghat turbulent flux.
ghatflux	Surface flux for ghat: includes solar flux.
rhs	Right hand side.

18.6.3 Subroutine TRIDMAT

TRIDMAT solves a tridiagonal matrix for new vector yn, given right hand side vector rhs.

Calling Sequence: Subroutine tridmat(tcu, tcc, tcl, nlayer, h, rhs, yo, yn, diff)

Data Declaration:

Integer	nlayer
Real	diff, h, yo, tcu, tcc, tcl, rhs, yn

Arguments:

diff	Diffusivity profile on interfaces.
nlayer	
h	Layer thickness.
yo	Old profile.
rhs	Right hand side.
tcu	Upper coefficient for (k-1) on k line of tridiagonal matrix.
tcc	Central coefficient for (k) on k line of tridiagonal matrix.
tcl	Lower coefficient for (k-1) on k line of tridiagonal matrix.
yn	New field.

18.7 Communication Subroutines

The following table contains descriptions of all HYCOM communication subroutines. With the exception of XCHALT, all subroutines are assumed to be called with identical argument lists by all processors when using SPMD message passing. Two versions of each subroutine are provided, mod_xc_mp.F for message passing, and mod_xc_sm.F for a single processor. The appropriate version is included in mod_xc.F under control of cpp macros. The routines are configured as a module, and all HYCOM routines should start with use mod_xc to allow HYCOM communication routines to be invoked when required.

Table 8: HYCOM Communication Subroutines

File	Subroutine	Description
mod_xc_mp.f	xcaget	Converts an entire 2-D array from tiled to non-tiled layout.
	xcaput	Converts an entire 2-D array from non-tiled to tiled layout.
	xceget	Finds the value of a(ia,ja) on the non-tiled 2-D grid.
	xceput	Fills a single element in the non-tiled 2-D grid.
	xchalt	Emergency stops all processes, called by one process.
	xclget	Extracts a line of elements from the non-tiled 2-D grid.
	xclput	Fills a line of elements in the non-tiled 2-D grid.
	xcmaxr_0	Replaces a scalar a with its element-wise maximum over all tiles.
	xcmaxr_1	Replaces an array a with its element-wise maximum over all tiles.
	xcminr_0	Replaces a scalar a with its element-wise minimum over all tiles.
	xcminr_1	Replaces an array a with its element-wise minimum over all tiles.
	xcspmd	Initializes data structures that identify the tiles.
	xcstop	Stops all processes, called by all processes.
	xcsum	Sums a 2-D array, where mask = 1.
	xcsumj	Row-sum of a 2-D array, where mask = 1, on first processor only.
	xcsync	Barrier, no processor exits until all arrive (and flush stdout).
	xctbar	Provides synchronization with processors ipe1 and ipe2.
	xctilr	Updates the tile overlap halo of a 3-D real array.
	xctmri	Initializes timers.
	xctmr0	Starts timer n.
	xctmr1	Adds time since call to xctim0 to timer n.
	xctmrn	Registers name of timer n.

	xctmrp	Prints all active timers.
mod_xc_sm.f		This file contains the shared memory version of all the subroutines in mod_xc_mp.

18.7.1 Subroutine XCAGET

XCAGET converts an entire 2-D array from tiled to non-tiled layout.

Calling Sequence: Subroutine xcaget(aa, a, mnflg)

Data Declaration: Integer mnflg
Real aa, a

Arguments:

aa	Non-tiled target array.
a	Tiled source array.
mnflg	Node-return flag. = 0, all nodes; = n, node number n (mnproc=n).

Common Blocks: xcmpii

18.7.2 Subroutine XCAPUT

XCAPUT converts an entire 2-D array from non-tiled to tiled layout.

Calling Sequence: Subroutine xcaput(aa, a, mnflg)

Data Declaration: Integer mnflg
Real aa, a

Arguments:

aa	Non-tiled source array.
a	Tiled target array.
mnflg	Node source flag. = 0, all nodes; = n, node number n (mnproc=n).

18.7.3 Subroutine XCEGET

XCEGET finds the value of a(ia,ja) on the non-tiled 2-D grid.

Calling Sequence: Subroutine xceget(aelem, a, ia, ja)

Data Declaration: Integer ia, ja
Real aelem, a

Arguments:

aelem	Required element.
a	Source array.
ia	1st index into *a*.
ja	2nd index into *a*.

Common Blocks: xcmpii
xcegetr

18.7.4 Subroutine XCEPUT

XCEPUT fills a single element in the non-tiled 2-D grid.

Calling Sequence: Subroutine xceput(aelem, a, ia, ja)

Data Declaration: Integer ia, ja
Real aelem, a

Arguments:

aelem	Element value.
a	Target array.
ia	1st index into *a*.
ja	2nd index into *a*.

Common Blocks: xcmpii

18.7.5 Subroutine XCHALT

XCHALT stops all processes. Only one process need call this routine, i.e. it is for emergency stops. Use ‘xcstop’ for ordinary stops called by all processes.

Calling Sequence:	Subroutine xchalt(cerror)	
Data Declaration:	Character cerror	
Arguments:	cerror	Error message.
Common Blocks:	xcmpii	

18.7.6 Subroutine XCLGET

XCLGET extracts a line of elements from the non-tiled 2-D grid.

Calling Sequence:	Subroutine xclget(aline, nl, a, i1, j1, iinc, jinc, mnflg)	
Data Declaration:	Integer n1, i1, j1, iinc, jinc, mnflg Real aline, a	
Arguments:	aline	Required line of elements.
	n1	Dimension of aline.
	a	Source array.
	i1	1st index into *a*.
	j1	2nd index into *a*.
	iinc	1st index increment.
	jinc	2nd index increment.
	mnflg	Node return flag. = 0, all nodes; = n, node number n (mnproc=n).
Common Blocks:	xcmpii xclgetr	

18.7.7 Subroutine XCLPUT

XCLPUT fills a line of elements in the non-tiled 2-D grid.

Calling Sequence:	Subroutine xclput(aline, nl, a, i1, j1, iinc, jinc)

Data Declaration:	Integer n1, i1, j1, iinc, jinc Real aline, a	
Arguments:	aline	Line of element values.
	n1	Dimension of aline.
	a	Target array.
	i1	1st index into *a*.
	j1	2nd index into *a*.
	iinc	1st index increment.
	jinc	2nd index increment.

18.7.8 Subroutine XCMAXR_0

XCMAXR_0 replaces scalar *a* with its element-wise maximum over all tiles.

Calling Sequence:	Subroutine xcmaxr_0(a)	
Data Declaration:	Real a	
Arguments:	a	Target variable.

18.7.9 Subroutine XCMAXR_1

XCMAXR_1 replaces array *a* with its element-wise maximum over all tiles.

Calling Sequence:	Subroutine xcmaxr_1(a)	
Data Declaration:	Real a	
Arguments:	a	Target array.

18.7.10 Subroutine XCMINR_0

XCMINR_0 replaces scalar *a* with its element-wise minimum over all tiles.

Calling Sequence: Subroutine xcminr_0(a)

Data Declaration: Real a

Arguments: a Target variable.

18.7.11 Subroutine XCMINR_1

XCMINR_1 replaces array *a* with its element-wise minimum over all tiles.

Calling Sequence: Subroutine xcminr_1(a)

Data Declaration: Real a

Arguments: a Target array.

Common Blocks: xcmpii
xcmaxr4

18.7.12 Subroutine XCSPMD

XCSPMD initializes data structures that identify the tiles. It contains no arguments.

Common Blocks: xcmpii

18.7.13 Subroutine XCSTOP

XCSTOP stops all processes. All processes must call this routine.

Calling Sequence: Subroutine xcstop(cerror)

Data Declaration: Character cerror

Arguments: cerror Error message.

Common Blocks: xcmpii

18.7.14 Subroutine XCSUM

XCSUM sums a 2-D array, where mask=1.

Calling Sequence:	Subroutine xcsum(sum, a, mask)	
Data Declaration:	Integer mask Real sum, a	
Arguments:	sum	Sum of a.
	a	Source array.
	mask	Mask array.
Common Blocks:	xcmpii xcsum8	

18.7.15 Subroutine XCSUMJ

XCSUMJ is a row-sum of a 2-D array, where mask==1, on first processor only.

Calling Sequence:	Subroutine xcsum(sumj, a, mask)	
Data Declaration:	Integer mask Real sum, a	
Arguments:	sumj	Row-sum of a.
	a	Source array.
	mask	Mask array.
Common Blocks:	xcmpii xcsum8	

18.7.16 Subroutine XCSYNC

XCSYNC is a barrier, no processor exits until all arrive (and flush stdout). Some MPI implementations only flush stdout as a collective operation, and hence the lflush=.true. option to flush stdout. Typically, this is just a wrapper to the “BARRIER” macro.

Calling Sequence: Subroutine xcsync(lflush)

Data Declaration: Logical lflush

Arguments:

lflush	
a	Source array.
mask	Mask array.

Common Blocks: xcmpii

18.7.17 Subroutine XCTBAR

XCTBAR is a global collective operation, and the calls on ipe1 and ipe2 must list this processor as one of the two targets. This is used in place of a global barrier in halo operations, but it only provides syncronization of one or two processors with the local processor. ipe1 and/or ipe2 can be null_tile, to indicate no processor.

Calling Sequence: Subroutine xctbar(ipe1, ipe2)

Data Declaration: Integer ipe1, ipe2

Arguments:

ipe1	Processor 1.
ipe2	Processor 2.

Common Blocks: halobp

18.7.18 Subroutine XCTILR

XCTILR updates the tile overlap halo of a real array.

Calling Sequence: Subroutine xctilr(a, l1, ld, mh, nh, itype)

Data Declaration: Integer l1, ld, mh, nh, itype
Real a

Arguments:	a	Target array.
	l1	Third dimension start index.
	ld	Third dimension of a.
	mh	First (EW) update halo size.
	nh	Second (NS) update halo size.
	itype	Grid and field type.
		= 1, p-grid, scalar field;
		= 2, q-grid, scalar field;
		= 3, u-grid, scalar field;
		= 4, v-grid, scalar field;
		= 11, p-grid, vector field;
		= 12, q-grid, vector field;
		= 13, u-grid, vector field;
		= 14, v-grid, vector field.

Common Blocks: xcmpii
xctilr4

18.7.19 Subroutine XCTMR*

XCTMR* are timer subroutines.

XCTMRI	Initializes timers. Timers 1 to 32 are for message passing routines, timers 33 to 80 are for general hycom routines, timers 81 to 96 are for user selected routines, and timer 97 is the total time.
XCTMR0	Starts timer n.
XCTMR1	Stops timer n and adds event to timer sum.
XCTMRN	Registers a name for timer n.
XCTMRP	Printout timer statistics (called by xcstop).

Calling Sequence: Subroutine xctmri()
Subroutine xctmr0(n)
Subroutine xctmr1(n)
Subroutine xctmrn(n, cname)
Subroutine xctmrp()

Data Declaration: Character cname
Integer n

Arguments:	n	Timer number.
	cname	Registered name of timer.

Common Blocks: xcmpii

18.8 Machine Dependent I/O Subroutines

Table 9: Machine Dependent I/O Subroutines

File	Subroutine	Description
machine.f	machine	Machine-specific initialization.
	flush	Wrapper for flush system call under AIX.
	ieee_retrospective	Dummy routine to turn off ieee warning messages on a Sun.
	getenv	This subroutine provides getenv functionality on the t3e, using pxfgetenv.
mod_za_mp1.f		Machine dependent I/O routines. Message passing version, with all I/O from first processor. Contained in module mod_za.
	zaiopn	Machine specific routine for opening a file for array I/O.
	zaiope	Machine specific routine for opening a file for array I/O.
	zaiopf	Machine specific routine for opening a file for array I/O.
	zaiopi	Is an array I/O unit open?
	zaiost	Machine specific routine for initializing array I/O.
	zaiocl	Machine specific routine for array i/o file closing.
	zaiofl	Machine specific routine for array I/O buffer flushing.
	zaiorw	Machine specific routine for array I/O file rewinding.
	zaiord3	Machine specific routine for 3-D array reading.
	zaiord	Machine specific routine for array reading.
	zaiordd	Direct access reads a single record.
	zaiosk	Machine specific routine for skipping an array read.
	zaiowr3	Machine specific routine for 3-D array writing.
	zaiowr	Machine specific routine for array writing.
	zaiowrd	Direct access writes a single record.
mod_za_sm.f		Machine dependent I/O routines. This file contains the single processor version of the subroutines contained in mod_za_mp1.

18.8.1 Subroutine MACHINE

MACHINE performs machine-specific initialization.

18.8.2 Subroutine FLUSH

FLUSH is a wrapper for flush system call under AIX.

Calling Sequence: Subroutine flush(iunit)

Data Declaration: Integer iunit

Arguments: iunit

18.8.3 Subroutine IEEE_RETROSPECTIVE

IEEE_RETROSPECTIVE is a dummy routine to turn off iee warning messages on a Sun.

18.8.4 Subroutine GETENV

GETENV provides getenv functionality on the t3e using PXFGETENV.

Calling Sequence: Subroutine getenv(cname, cvalue)

Data Declaration: Character cname, cvalue

Arguments: cname
cvalue

18.8.5 Subroutine ZAIOPN

ZAIOPN is a machine specific routine for opening a file for array i/o.

Calling Sequence: Subroutine zaiopn(cstat, iaunit)

Data Declaration: Character cstat
Integer iaunit

Arguments:

cstat	File type, it can be 'scratch', 'old', or 'new'.
iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatability, assume that iaunit+1000 refers to a fortran i/o unit anyway.

Common Blocks: czioxx
czioxw

18.8.6 Subroutine ZAIOPE

ZAIOPE is a machine specific routine for opening a file for array i/o.

Calling Sequence: Subroutine zaiope(cenv, cstat, iaunit)

Data Declaration: Character cenv, cstat
Integer iaunit

Arguments:

cenv	Environment variable from which the filename is taken.
cstat	File type, it can be 'scratch', 'old', or 'new'.
iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatability, assume that iaunit+1000 refers to a fortran i/o unit anyway.

Common Blocks: czioxx
czioxw

18.8.7 Subroutine ZAIOPF

ZAIOPF is a machine specific routine for opening a file for array i/o.

Calling Sequence: Subroutine zaiopf(cfile, cstat, iaunit)

Data Declaration: Character cfile, cstat
Integer iaunit

Arguments:

cfile	Variable from which the filename is taken.
cstat	File type, it can be 'scratch', 'old', or 'new'.
iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatibility, assume that iaunit+1000 refers to a fortran i/o unit anyway.

Common Blocks: czioxx
czioxw

18.8.8 Subroutine ZAIOPI

ZAIOPI determines if an i/o array is open.

Calling Sequence: Subroutine zaiopi(lopen, iaunit)

Data Declaration: Integer iaunit
Logical lopen

Arguments:

lopen	
iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatibility, assume that iaunit+1000 refers to a fortran i/o unit anyway.

Common Blocks: czioxx

18.8.9 Subroutine ZAIOST

ZAIOST is a machine specific routine for initializing array i/o. It contains no arguments.

Common Blocks: czioxx

18.8.10 Subroutine ZAIOCL

ZAIOCL is a machine specific routine for array i/o file closing.

Calling Sequence: Subroutine zaiocl(iaunit)

Data Declaration: Integer iaunit

Arguments:

iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatibility, assume that iaunit+1000 refers to a fortran i/o unit anyway.

Common Blocks: czioxx

18.8.11 Subroutine ZAIOFL

ZAIOFL is a machine specific routine for array i/o buffer flushing.

Calling Sequence: Subroutine zaiofl(iaunit)

Data Declaration: Integer iaunit

Arguments: iaunit iaunit+1000 is the i/o unit used for arrays.
Array i/o might not use fortran i/o units, but,
for compatibility, assume that iaunit+1000 refers to a fortran i/o unit anyway.

Common Blocks: czioxx

18.8.12 Subroutine ZAIORW

ZAIORW is a machine specific routine for array i/o file rewinding.

Calling Sequence: Subroutine zaiorw(iaunit)

Data Declaration: Integer iaunit

Arguments: iaunit iaunit+1000 is the i/o unit used for arrays.
Array i/o might not use fortran i/o units, but,
for compatibility, assume that iaunit+1000 refers to a fortran i/o unit anyway.

Common Blocks: czioxx

18.8.13 Subroutine ZAIORD3

ZAIORD3 is a machine specific routine for 3-D array reading.

Calling Sequence: Subroutine zaiord3(h, l, mask, lmask, hmin, hmax, iaunit)

Data Declaration:	Integer	l, mask, iaunit
	Logical	lmask
	Real	h, hmin, hmask

Arguments:	h	Array to be read.
	l	Number of times zaiord3 is called.
	mask	
	lmask	
	hmin	Minimum value in the array *h*, ignoring array elements set to 2.0**100.
	hmax	Maximum value in the array *h*, ignoring array elements set to 2.0**100.
	iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatibility, assume that iaunit+1000 refers to a fortran i/o unit anyway.

18.8.14 Subroutine ZAIORD

ZAIORD is a machine specific routine for array reading.

Calling Sequence: Subroutine zaiord(h, mask, lmask, hmin, hmax, iaunit)

Data Declaration:	Integer	mask, iaunit
	Logical	lmask
	Real	hmin, hmax

Arguments:	h	Array.
	mask	
	lmask	
	hmin	Minimum value in the array *h*, ignoring array elements set to 2.0**100.
	hmax	Maximum value in the array *h*, ignoring array elements set to 2.0**100.
	iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatibility, assume that iaunit+1000 refers to a fortran i/o unit anyway.

Common Blocks: czioxx
czioxw

czioxr

18.8.15 Subroutine ZAIORDD

ZAIORDD is direct access to read a single record. It is expressed as a subroutine because i/o with implied do loops can be slow on some machines.

Calling Sequence: Subroutine zaiordd(a, n, iunit, irec, ios)

Data Declaration:

Integer	n, iunit, irec, ios
Real	a

Arguments:

a
n
iunit
irec
ios

18.8.16 Subroutine ZAIOSK

ZAIOSK is a machine specific routine for skipping an array read.

Calling Sequence: Subroutine zaiosk(iaunit)

Data Declaration: Integer iaunit

Arguments:

iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatibility, assume that iaunit+1000 refers to a fortran i/o unit anyway.

Common Blocks: czioxx

18.8.17 Subroutine ZAIOWR3

ZAIOWR3 is a machine specific routine for 3-D array writing.

Calling Sequence:	Subroutine zaiowr3(h, l, mask, lmask, hmin, hmax, iaunit, lreal4)	

Data Declaration:	Integer	l, mask, iaunit
	Logical	lmask, lreal4
	Real	hmin, hmask

Arguments:	h	
	l	Number of times subroutine is called.
	mask	
	lmask	
	hmin	Minimum value in the array *h*, ignoring array elements set to 2.0**100.
	hmax	Maximum value in the array *h*, ignoring array elements set to 2.0**100.
	iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatibility, assume that iaunit+1000 refers to a fortran i/o unit anyway.
	lreal4	

18.8.18 Subroutine ZAIOWR

ZAIOWR is a machine specific routine for array writing.

Calling Sequence:	Subroutine zaiowr(h, mask, lmask, hmin, hmax, iaunit, lreal4)	

Data Declaration:	Integer	iaunit, mask
	Logical	lmask, lreal4
	Real	h, hmin, hmax

Arguments:	h	
	mask	
	lmask	
	hmin	Minimum value in the array *h*, ignoring array elements set to 2.0**100.
	hmax	Maximum value in the array *h*, ignoring array elements set to 2.0**100.
	iaunit	iaunit+1000 is the i/o unit used for arrays. Array i/o might not use fortran i/o units, but, for compatibility, assume that iaunit+1000 refers to a

fortran i/o unit anyway.

Common Blocks: czioxx
czioxw
czioxr

18.8.19 Subroutine ZAIOWRD

ZAIOWRD is direct access to write a single record. It is expressed as a subroutine because i/o with implied do loops can be slow on some machines.

Calling Sequence: Subroutine zaiowrd(a, n, iunit, irec, ios)

Data Declaration:

Integer	n, iunit, irec, ios
Real	a

Arguments: a
n
iunit
irec
ios

18.9 Pipe Comparison Subroutines

Table 10: Pipe Comparison Subroutines

File	Subroutine	Description
mod_pipe.f	pipe_init	Initializes the pipe comparison process.
	pipe_compare	Subroutine checks whether data stored in 'field' are identical.
	pipe_comparall	Writes out a standard menu of arrays for testing.

18.9.1 Subroutine PIPE_INIT

PIPE_INIT initializes the pipe comparison process.

18.9.2 Subroutine PIPE_COMPARE

PIPE_COMPARE checks whether or not data stored in 'field' are identical.

Calling Sequence: Subroutine pipe_compare(field, mask, what)

Data Declaration:

Real	field
Integer	mask
Character	what

Arguments:

field
mask
what

18.9.3 Subroutine PIPE_COMPARALL

PIPE_COMPARALL writes out a standard menu of arrays for testing.

Calling Sequence: Subroutine pipe_comparall(m, n, cinfo)

Data Declaration:

Integer	m, n
Character	cinfo

Arguments:

m,n
cinfo

18.10 Diagnostic Output Subroutines

Table 11: Diagnostic Output Subroutines

File	Subroutine	Description
overtn.f	overtn	Diagnose meridional heat flux in basin mode.
stencl.f	stencl	Writes 5 x 5 point cluster of grid point values centered on (itest,jtest). (Present, but not used in HYCOM)

18.10.1 Subroutine OVERTN

OVERTN is used to diagnose meridional heat flux in basin the model.

Calling Sequence: Subroutine overtn(dtime, dyear)

Data Declaration: Real dtime, dyear

Arguments: dtime
dyear

18.10.2 Subroutine STENCL

STENCL writes a 5 x 5 point cluster of grid point values centered on (itest, jtest).

Calling Sequence: Subroutine stencl(k1, n)

Data Declaration: Integer n, k1

Arguments: k1
n

18.11 Plotting Subroutines

18.11.1 Subroutine PRTMSK

PRTMSK deletes 'array' elements outside 'mask', then breaks 'array' into sections, each 'nchar' characters wide, for printing.

Calling Sequence: Subroutine prtmsk(mask, array, work, idm, ii, jj, offset, scale, title)

Table 12: Plotting Subroutines

File	**Subroutine**	**Description**
prtmsk.f	prtmsk	Delete 'array' elements outside 'mask'. Then break 'array' into sections, each 'nchar' characters wide, for printing.
psmoo.f	psmooth	Ragged boundary version of basic 9-point smoothing routine. This routine is set up to smooth data carried at -p-points.
	psmooth_max	Ragged boundary version of basic 9-point smoothing routine. This routine is set up to smooth data carried at -p- points and to return the maximum of the original and smoothed value.
zebra.f	zebra	Find nice contour interval resulting in 7 to 10 contour lines and draw contours on line printer through the following set of grid points: array(1,nj); array(ni,nj), array(1,1) and array(ni,1).
	zebram	Find nice contour interval resulting in 7 to 10 contour lines and draw contours on line printer through the following set of grid points: array(1,1); array(1,jj); array(ii,jj) and array(ii,jj).
	digplt	Simulates a contour line plot on the printer.

Data Declaration:

Real	array, work, offset, scale
Integer	idm, mask, ii, jj
Character	title

Arguments:

mask
array
work
idm
ii
jj
offset
scale
title

Common Blocks: linepr

18.11.2 Subroutine PSMOOTH

PSMOOTH is a ragged boundary version of basic 9-point smoothing routine. PSMOOTH is set up to smooth data carried at p points.

Calling Sequence: Subroutine psmooth(a, margin_smooth)

Data Declaration:

Real	a
Integer	margin_smooth

Arguments:

a
margin_smooth

18.11.3 Subroutine PSMOOTH_MAX

PSMOOTH_MAX is a ragged boundary version of basic 9-point smoothing routine. PSMOOTH_MAX is set up to smooth data carried at p points and to return the maximum of the original and smoothed value.

Calling Sequence: Subroutine psmooth_max(a, margin_smooth)

Data Declaration:

Real	a
Integer	margin_smooth

Arguments:

a
margin_smooth

18.11.4 Subroutine ZEBRA

Subroutine ZEBRA finds a nice contour interval resulting in 7 to 10 contour lines and draws contours on line printer through grid points: (1,nj); (ni,nj); (1,1); and (ni,1).

Calling Sequence: Subroutine zebra(array, idim, ni, nj)

Data Declaration:

Integer	idim, ni, nj
Real	array

Arguments: idim
array
ni,nj

18.11.5 Subroutine ZEBRAM

Subroutine ZEBRAM finds a nice contour interval resulting in 7 to 10 contour lines and draws contours on line printer through grid points: (1,1); (1,jj); (ii,jj); and (ii,jj). It is a MICOM version of subroutine ZEBRA.

Calling Sequence: Subroutine zebra(array, idim, ni, nj)

Data Declaration:

Integer	idim, ii, jj
Real	array

Arguments: idim
array
ii,jj

18.11.6 Subroutine DIGPLT

Subroutine DIGPLT simulates a contour line plot on the printer.

Calling Sequence: Subroutine digplt(array, idim, ii, jj, dec)

Data Declaration:

Integer	idim, ii, jj
Real	array, dec

Arguments: idim
array
ii,jj
dec

Acronyms

FCT	Flux-Corrected Transport scheme
HCYCOM	HYbrid Coordinate Ocean Model
KPP	K-Profile Parameterization model setup
KT	Kraus-Turner model setup
MICOM	Miami-Isopycnic-Coordinate Ocean Model
MKS	Meter, Kilogram and Second unit system
MLB	Mixed Layer Base
MPDATA	Multidimensional Positive Definite Advection Transport Algorithm
NOPP	National Oceanographic Partnership Program
NRL	Naval Research Laboratory
ONR	Office of Naval Research
PE	Potential Energy
POM	Princeton Ocean Model
TKE	Turbulent Kinetic Energy
UNESCO	United Nations Educational, Scientific and Cultural Organization

References

Baraille, R. and Filatoff, N., (1995). Modèle shallow-water muticouches isopcynal de Miami. *Rapport d'Etude*, CMO/RE No 003/95.

Bleck, R., (1978). Finite difference equations in general vertical coordinates. *Contrib. Atmos. Phys.*, **51**: 360-372.

Bleck, R. and Boudra, D., (1981). Initial testing of a numerical ocean circulation model using a hybrid (quasi-isopycnic) vertical coordinate. *J. Phys. Oceanogr.*, **11**: 755-770.

Bleck, R. and Boudra, D.B., (1986). Wind-driven spin-up in eddy-resolving ocean models formulated in isopycnic and isobaric coordinates. *J. Geophys. Res.*, **91**(C): 7611-7621.

Bleck, R., Hanson, H.H., Hu, D., and Kraus, E.B., (1989). Mixed layer-thermocline interaction in a 3D isopycnic coordinate model. *J. Phys. Oceanogr.*, **19**: 1417-1439.

Bleck, R. and Smith L.T., (1990). A wind-driven isopycnic coordinate model of the north and equatorial Atlantic ocean. I- model development and supporting experiments. *J. Geophys. Res.*, **95**(C): 3273-3285.

Bleck, R., Rooth, C., Hu, D., and Smith, L.T., (1992). Salinity-driven thermocline transients in a wind- and thermohaline-forced isopycnic coordinate model of the North Atlantic. *J. Phys. Oceanogr.*, **22**: 1486-1505.

Bleck, R., Rooth, C., Hu, D., and Smith, L.T., (1992). Ventilation patterns and mode water formation in a wind- and thermodynamically driven isopycnic coordinate model of the North Atlantic. *J. Phys. Oceanogr.*, **22**: 1486-1505.

Bleck, R. and Benjamin, S., (1993). Regional weather prediction with a model combining terrainfollowing and isentropic coordinates, Part I: Model Description. *Mon. Wea. Rev.*, **121**: 1770-1785.

Bleck, R., (1998). "Ocean Modeling in Isopycnic Coordinates" Ocean Modeling and Parameterization. Chassignet E.P. and Vernon J., eds., NATO Science Series C: Mathematical and Physical Sciences, Vol. 516. **Kluwer Academic Publishers**, p. 4223-448???.

Bleck, R., (2001). An oceanic general circulation model framed in hybrid isopycnic-Cartesian coordinates. *Submitted to???*.

Browning, G.L. and Kreiss, H.-O., (1982). Initialization of the shallow water equations with open boundaries by the bounded derivative method. *Tellus*, **34**: 334-351.

Browning, G.L. and Kreiss, H.-O., (1986). Scaling and computation of smooth atmospheric motions. *Tellus*, **38A**: 295-313.

Brydon, D., Sun, S. and Bleck, R., (2001). A new approximation of the equation of state for sea water, suitable for numerical models. *?????????*.

Canuto, V.M., (2000). Ocean Turbulence: A model with shear stratification and salinity. NASA Goddard Institute for Space Studies, Unpublished Manuscript.

Chassignet, E.P., Smith, L.T., Bleck, R., and Bryan, F.O., (1996). A model comparison: Numerical simulations of the north and equatorial Atlantic oceanic cirrculation in depth and isopycnic coordinates. *J. Phys. Oceanogr.*, **26**: 1849-1867.

Chassignet, E.P., Smith, L., Halliwell, G.R., and Bleck, R., (2001). North Atlantic simulations with the HYbrid Coordinate Ocean Model (HYCOM): Impact of the vertical coordinate choice and resolution, reference density and thermobaricity. To be submitted to *(currently undecided)*.

Cushman-Roisin, B., (1994). Introduction to Geophysical Fluid Dynamics. **Prentice Hall**, New-Jersey.

Friedrich, H. and Levitus, S., (1972). An approximation to the equation of state for sea water, suitable for numerical ocean models. *J. Phys. Oceanogr.*, **2**: 514-517.

Gaspar, P., (1988). Modelling the seasonal cycle of the upper ocean. *J. Phys. Oceanogr.*, **18**: 161-180.

Halliwell Jr., G.R., (1997). Simulation of decadal/interdecadal variability the North Atlantic driver by the anomalous wind field. *In the Proc. on Climate Variations*, Long Beach, CA, 97-102.

Halliwell Jr., G.R., (1998). Simulation of North Atlantic decadal/multi-decadal winter SST anomalies driven by basin-scale atmospheric circulation anomalies. *J. Phys. Oceanogr.*, **28**: 5-21.

Halliwell Jr., G.R., Bleck, R. and Chassignet, E., (1998). Atlantic ocean simulations performed using a new HYbrid Coordinate Ocean Model (HYCOM). *EOS, Fall AGU Meeting.*

Halliwell Jr., G.R., Bleck, R., Chassignet, E.P., and Smith, L.T., (2000). Mixed layer model validation in Atlantic ocean simulations using the HYbrid Coordinate Ocean Model (HYCOM). *EOS, 80, OS304*.

Halliwell Jr., G.R., (2001). Evaluation of vertical coordinate and vertical mixing algorithms in the HYbrid Coordinate Ocean Model (HYCOM). To be submitted to *Ocean Modeling.*

Hu, D., (1991). A joint mixed layer/isopycnic coordinate numerical model of wind- and thermohaline- driven ocean general circulation with model sensitive study. *Ph.D.*, University of Miami, Florida, p. 220.

Hu, D., (1996). On the sensitivity of thermocline depth and meridional heat transport to vertical diffusivity in OGCMs. *J. Phys. Oceanogr.*, **26**: 1480-1494.

Hu, D., (1997). Global-scale water masses, meridional circulation, and heat transport simulated with a global isopycnic ocean model. *J. Phys. Oceanogr.*, **27**: 96-120.

Jerlov, N.G., (1976). Marine Optics. **Elsevier Publishing**, New York.

Kara, A.B., Rochford, P.A., and Hurlburt, H.E., (2000). Efficient and accurate bulk parameterizations of air-sea fluxes for use in general circulation models. *J. Atmos. Ocean Tech.*, **17**: 1421-1438.

Kraus, E.B. and Turner, J.S, (1967). A one-dimensional model of the seasonal thermocline, Part II: The general theory and its consequences. *Tellus*, **19**: 98-105.

Large, W.G., McWilliams, J.C. and Doney, S.C., (1994). Oceanic vertical mixing: A review and a model with a nonlocal boundary layer parameterization. *Rev. Geophys.*, **32**: 363-403.

Large, W.G., Danabasoglu, G., Doney, S.C., and McWilliams, J.C., (1997). Sensitivity to surface forcing and boundary layer mixing in a global ocean model: Annual-mean climatology. *J. Phys. Oceanogr.*, **27**: 2418-2447.

Marsh, R., Roberts, M.J., Wood, R.A., and New, A.L., (1996). An intercomparison of a Bryan-Cox-type ocean model and an isopycnic ocean model, Part II: The subtropical gyre and meridional heat transport. *J. Phys. Oceanogr.*, **26**: 1528-1551.

McDougall, T.J. and Dewar, W.K., (1998). Vertical mixing and cabbeling in layered models. *J. Phys. Oceanogr.*, **28**: 1458-1480.

New, A., Bleck, R., Jia, Y., Marsh, R., Huddleston, M., and Barnard, S., (1995). An isopycnic model of the North Atlantic, Part I: Model Experiments. *J. Phys. Oceanogr.*, **25**: 2667-2699.

New, A. and Bleck, R., (1995). An isopycnic model of the North Atlantic, Part II: Interdecadal variability of the subtropical gyre. *J. Phys. Oceanogr.*, **25**: 2700-2714.

Niiler, P.P. and Kraus, E.B., (1977). A one-dimensional model of the upper ocean. In Modelling and Prediction of the Upper Layers of the Ocean. **Pergamon Press**, New York, p. 143-172.

Oliger, J. and Sundstrom, A., (1978). Theoretical and practical aspects of some initial boundary value problems in fluid mechanics. *SIAM Appl. Math.*, **35**: 419-446.

Peters, H., Gregg, M.C. and Toole, J.M., (1998). On the parameterization of equatorial turbulence. *J. Geophys. Res.*, **93**: 1199-1218.

Roberts, M.J., Marsh, P., New, A.L., and Wood, R.A., (1996). An intercomparison of a Bryan-Cox-type ocean model and an isopycnic ocean model, Part I: The subpolar gyre and high-latitude processes. *J. Phys. Oceanogr.*, **26**: 1495-1527.

Semtner Jr., A.J., (1976). A model for the thermodynamic growth of sea ice in numerical simulations of climate. *J. Phys. Oceanogr.*, **6**: 379-389.

Smith, L.T., Chassignet, E.P., Bleck, R., and Halliwell, G.R., (2000). A hybrid coordinate open-boundary-condition regional model for the inter-American seas. *EOS, 80, OS71.*

Smolarkiewicz, P.K., (1984). A fully multi-dimensional positive definite advection transport with small implicit diffusion. *J. Comput. Phys.*, **54**: 325-362.

Smolarkiewicz, P.K. and Clark, T.L., (1986). The multi-dimensional positive definite advection transport algorithm: further development and applications. *J. Comput. Phys.*, **67**: 396-438.

Smolarkiewicz, P.K. and Grabowski, W.W., (1990). The multi-dimensional positive definite advection transport algorithm: non-oscillatory Options. *J. Comput. Phys.*, **86**: 355-375.

Sun, S., Bleck, R. and Chassignet, E.P., (1993). Layer outcropping in numerical models of stratified flows. *J. Phys. Oceanogr.*, **23**: 1877-1884.

Sun, S., Bleck, R., Rooth, C., Dukowicz, J., Chassignet, E., and Kilworth, P., (1999). Inclusion of thermobaricity in isopycnic-coordinate ocean models. *J. Phys. Oceanogr.*, **29**: 2719-2729.

Zalesak, S., (1979). Fully multi-dimensional flux-corrected transport algorithms for fluids. *J. Comput. Phys.*, **31**: 335-362.