Notes on drag and mixing in ROMS/CROCO

November 1, 2022

1 Bottom stress and KPP mixing in ROMS and CROCO versions

We compare here implementations of bottom stress and KPP vertical mixing in different versions of the code: ROMS 2008, ROMS 2017 ("latest Sasha's version"), ROMS 2022 (from bitbucket: 2022/10) and CROCO (master from gitlab: 2022/10).

ROMS 2017 and ROMS 2022 are almost identical in these aspects. So we will focus mostly on ROMS 2008, 2017 and CROCO in the comparisons below.

1.1 Test cases

1.1.1 Shelf test case

We first use a simple test case corresponding to high wind event on a shallow shelf. The domain is a west-east reentrant channel, with closed north and south boundaries. The grid is 100 x 100 x 200 points, with a horizontal domain 100 km x 100 km. The topography corresponds to a 5 meters deep shelf in the southern half of the domain, which deepens linearly in the northern part down to 100 m at the northern boundary. We use 200 vertical levels with surface and bottom stretching ($\theta_s = 5$, $\theta_b = 2$) . We parameterize bottom friction using a logarithmic law of the wall with a roughness length Zob = 0.01 m. The initial flow is at rest, with no stratification. We apply a constant zonal wind of $\tau_w = 1$ N/m^2.

After initialisation, the flow is supposed to accelerate until it reaches equilibrium, when surface and bottom stress balance each other (Fig. 1). The final state is however very sensitive to the way bottom stress and vertical mixing

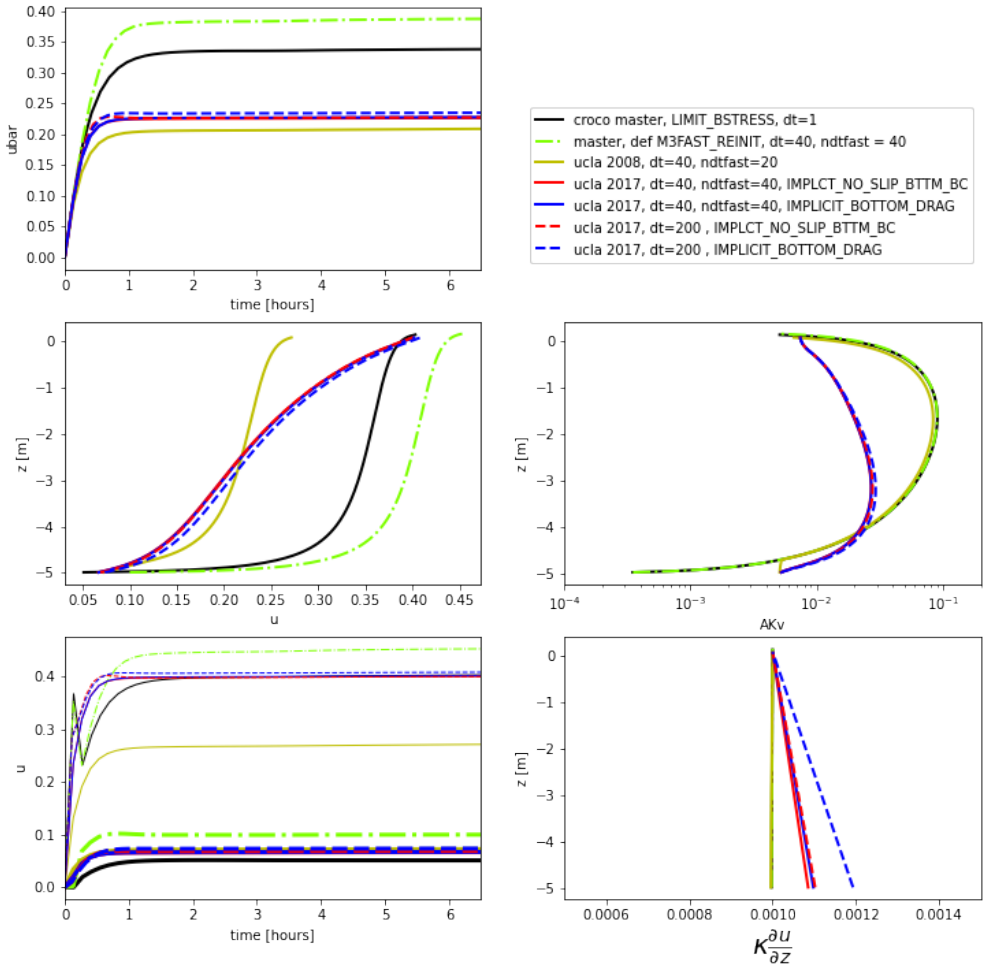

Figure 1: (a) Time evolution of barotropic velocity $\overline{u}$, (c) vertical profiles of velocity and (d) vertical mixing coefficient K_v (right) at the end of the simulations. (f) Vertical flux of momentum $\kappa\frac{\partial u}{\partial z}$

are parameterized in the model.

This test case was originally used to test stability of the CROCO code. The Atlantic configuration was strongly limited by blow-ups in shallow water due

to the formulation of the explicit bottom stress. This was partially solved by introducing a sub-time-stepping for bottom stress (option M3FAST or BSTRESS_FAST).

This test case is quite specific, in particular because is it fully mixed, and surface and bottom mixed-layer overlap. Another idealized test case with a deeper ocean and a stronger stratification will be used next, and finally some of the comparisons will be performed on a fully realistic nest.

1.2 Bottom drag coefficient

A first difference between versions is the computation of the bottom drag coefficient (r_d). A small modification is done in ROMS 2017 version to ensure that no singularity occurs as Hz approaches the roughness Zob, which is the case in the test case described in this document. This seems to be the best choice to ensure a regular variation of r_d in such configurations (to be included in CROCO as well).

There are also differences between ROMS and CROCO in the way u and v bottom velocities are discretized, but they seem to have pretty minor consequences.

ROMS 2008	ROMS 2017	CROCO
$r_d = \left[\frac{\kappa}{\log(\frac{Hz}{Zob})}\right]^2 \|u_b\|$	$r_d = \left[\frac{\kappa}{\log(1+\frac{0.5Hz}{Zob})}\right]^2 \|u_b\|$	$r_d = \left[\frac{\kappa}{\log(\frac{0.5Hz}{Zob})}\right]^2 \|u_b\|$

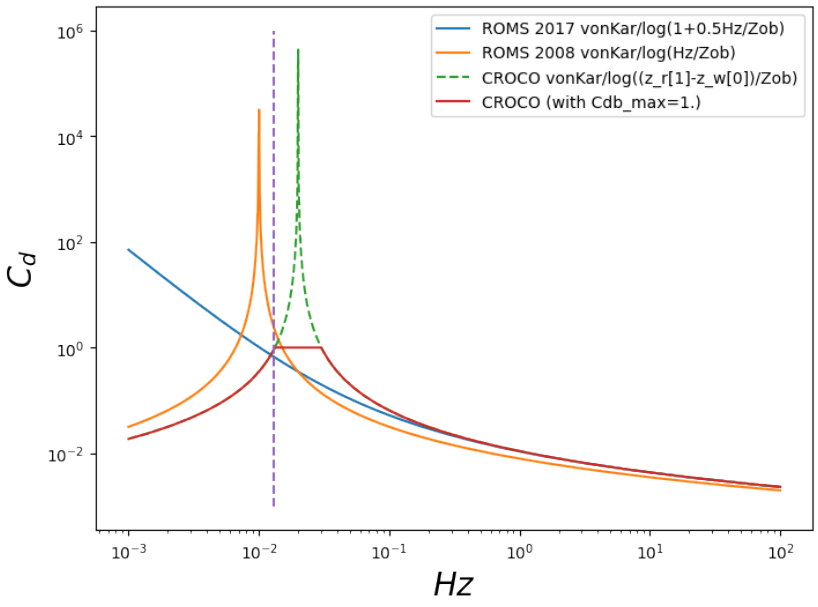

Figure 2: Computation of bottom drag coefficient r_d in the different codes.

1.3 Implicit versus explicit bottom drag

The main differences between versions come from the way bottom stress in handled.

By default, **CROCO uses an explicit, CFL limited bottom drag** (define LIMIT_BSTRESS). The stress is computed at the predictor stage (based on u^n), and the drag coefficient has to be limited to $r_D < 0.75Hz/dt$ to avoid numerical instabilities. The problem is that when Hz is very small as is the case here, the drag coefficient is limited to very small values and the currents reach unrealistically high values (and ultimately the model blows up). Furthermore the result is very sensitive to the time-step (dt) and one needs to get to very small values ($dt \leq 1$s) to reach convergence (Fig. 3).

An alternative solution designed by P. Marchesiello is to solve the bottom drag explicitly, but with a smaller time-step than the baroclinic time-step. This is what is achieved by the **BSTRESS_FAST** option, which reuses part of the NBQ structure of the code. In this case the bottom stress is computed inside the step3d_fast.F part of the code (which replaces the step2d.F). This allows the code to run with a slightly higher time-step ($dt = 40$ s) than the default version (Fig. 1).

In **ROMS 2008** the default is to use a **fully implicit approach**. r_d is computed at the corrector step (based on velocities at $u^{n+1/2}$) and the computation of the bottom stress is integrated in the implicit vertical viscosity solver (in rhs3d.F - step3d_uv1.F), if NO_SLIP_BOTTOM is defined (which is the default choice):

```
            DC(i,1)=(u(i,j,1,nnew) +DC(i,0)*ru(i,j,1)
     &                              +FC(i,1)*DC(i,2))
     &                    /( 0.5*(Hz(i,j,1)+Hz(i-1,j,1))
# ifdef NO_SLIP_BOTTOM
     &                     +dt * 0.5*(rd(i,j)+rd(i-1,j))
# endif
     &                          +FC(i,1)*(1.-CF(i,1)) )
```

The bottom stress thus computed is then used in the barotropic solver (step2d.F). Finally in step3d_uv2.F the only thing done is the coupling between the 2d and 3d solutions. See `http://people.atmos.ucla.edu/alex/ROMS/SaltLakeCity2012Talk.pdf` for a detailed explanation. The implicit code brings much more stability, as it is possible to reach the same final state with time-steps $dt = 1000$ s (Fig. 4).
However, the fully implicit solution used in ROMS 2008 is possible only with the corrector-coupled structure, while CROCO is only available in the predictor-coupled version. The implicit approach is incompatible with

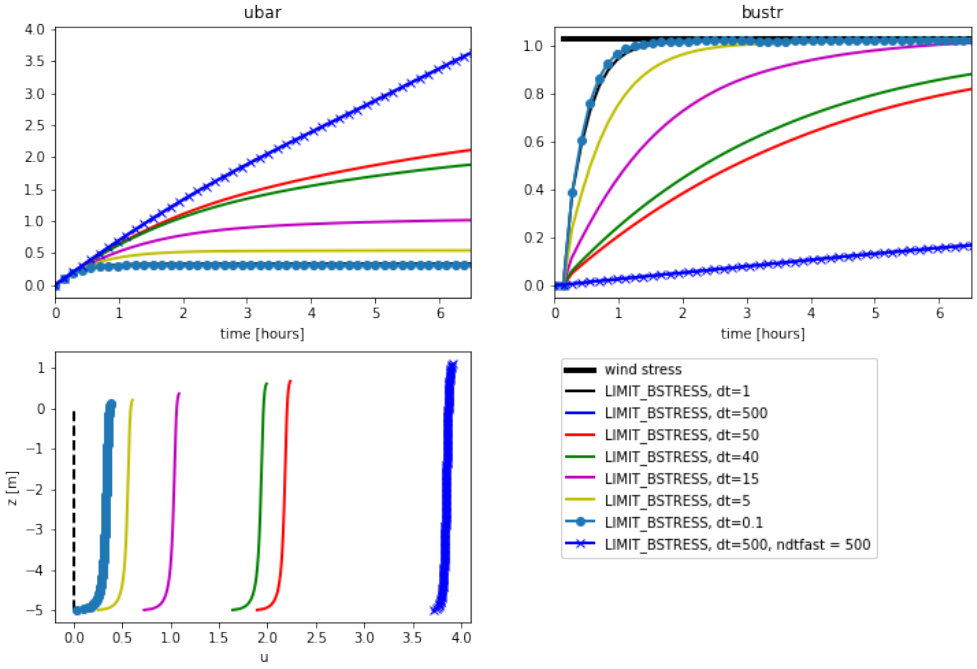

Figure 3: (a) Time evolution of barotropic velocity $\overline{u}$, (b) bottom stress *bustr*, and (c) vertical profiles of velocity at the end of the simulations for CROCO simulations with explicit bottom drag and various dt.

predictor-coupled structures because the drag needs to be computed for both r.h.s. 3D and for barotropic forcing to avoid splitting errors, but the implicit step for vertical viscosity is only done at the corrector stage (thus after the barotropic solver).

In **ROMS 2017** there are several possible options. First the code can be run as predictor-coupled or corrector-coupled. Second the bottom drag can be done in several ways with the use of the options:

- IMPLICIT_BOTTOM_DRAG
- IMPLCT_NO_SLIP_BTTM_BC

The default choice in ROMS 2017 (and only choice in ROMS 2022) is currently:

#undef PRED_COUPLED_MODE

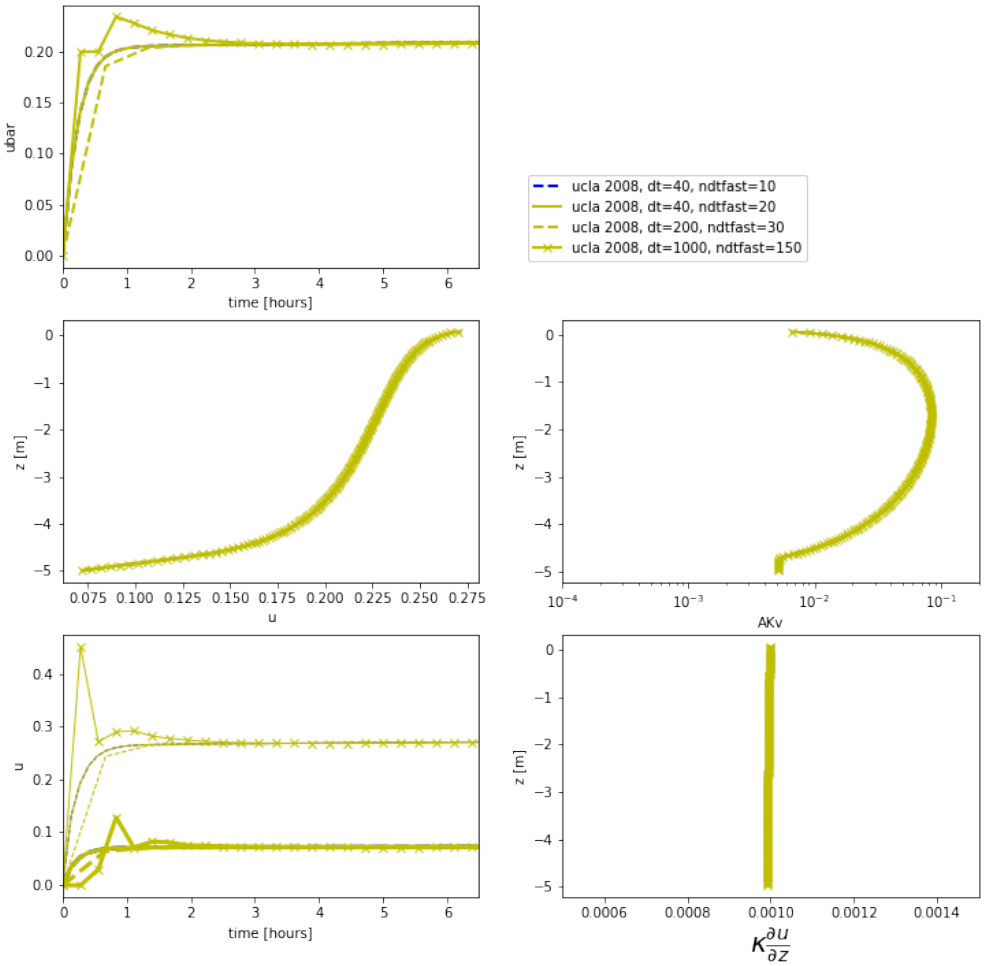

Figure 4: Same than Fig. 1 for UCLA 2008 with different time-steps.

```
#define CORR_COUPLED_MODE
c-# define IMPLICIT_BOTTOM_DRAG
# define IMPLCT_NO_SLIP_BTTM_BC
```

The option IMPLCT_NO_SLIP_BTTM_BC is equivalent to the option NO_SLIP_BOTTOM in the ROMS 2008 version. If defined, the bottom stress will be included in the implicit vertical viscosity solver. If not defined, the bottom stress will be

computed explicitly in step3d_uv1.F (if IMPLICIT_BOTTOM_DRAG is also undefined) or it will only be included in the barotropic equation, but not in the 3D equations (if IMPLICIT_BOTTOM_DRAG is defined).

If both options (IMPLICIT_BOTTOM_DRAG and IMPLCT_NO_SLIP_BTTM_BC) are defined (in correction-coupled mode), the computation of the drag is similar to ROMS 2008. So we have an implicit computation of the stress at the corrector stage (step3d_uv1.F), before the barotropic time-step (step2d.F) and finally only the 2d/3d coupling in step3d_uv2.F.

However the results are slightly different than with ROMS 2008 (Fig. 6). In particular the results are more sensitive to the choice of the baroclinic and (more suprisingly) the barotropic time-steps, and are less stable for large time-steps (see Fig. 5). The code is able to run with $dt = 1000$ s as with ROMS 2008, but giving different answers than the runs with smaller time-steps.

An interesting point is that the vertical momentum flux $\kappa \frac{\partial u}{\partial z}$ is not constant at equilibrium. This term is balanced by the effect of the 2d/3d coupling in the code.

If the option IMPLICIT_BOTTOM_DRAG is undefined (which is currently the default choice), the implicit vertical viscosity solver is not done in step3d_uv1.F before the barotropic step, but it is done after, in step3d_uv2.F. Thus, the bottom stress is computed explicitly in step3d_uv1.F, but only to be used in step2d.F to solve the barotropic equation:

```
      rufrc(i,j)=ru(i,j,1) +ru(i,j,N) +dn_u(i,j)*dm_u(i,j)*(
     &                                  0.5*(sustr(i-1,j)+sustr(i,j))
     &                     -0.5*(r_D(i-1,j)+r_D(i,j))*u(i,j,1,nstp) )
```

Then, it is recomputed implicitly in the final stage (step3d_uv2.F) inside the implicit vertical viscosity solver for the 3d equation.

Results are very close to the version with IMPLICIT_BOTTOM_DRAG for time-steps lower than about 200s (Fig. 6). However the code is not as stable and cannot be run with $dt = 400$ s, presumably due to splitting errors.

This solution is supposed to lead to splitting errors (as the 2d and 3d equations do not see the same bottom stress), and so I do not understand why it is the default choice.

Another difference, in the ROMS 2017 version, is that a second computation

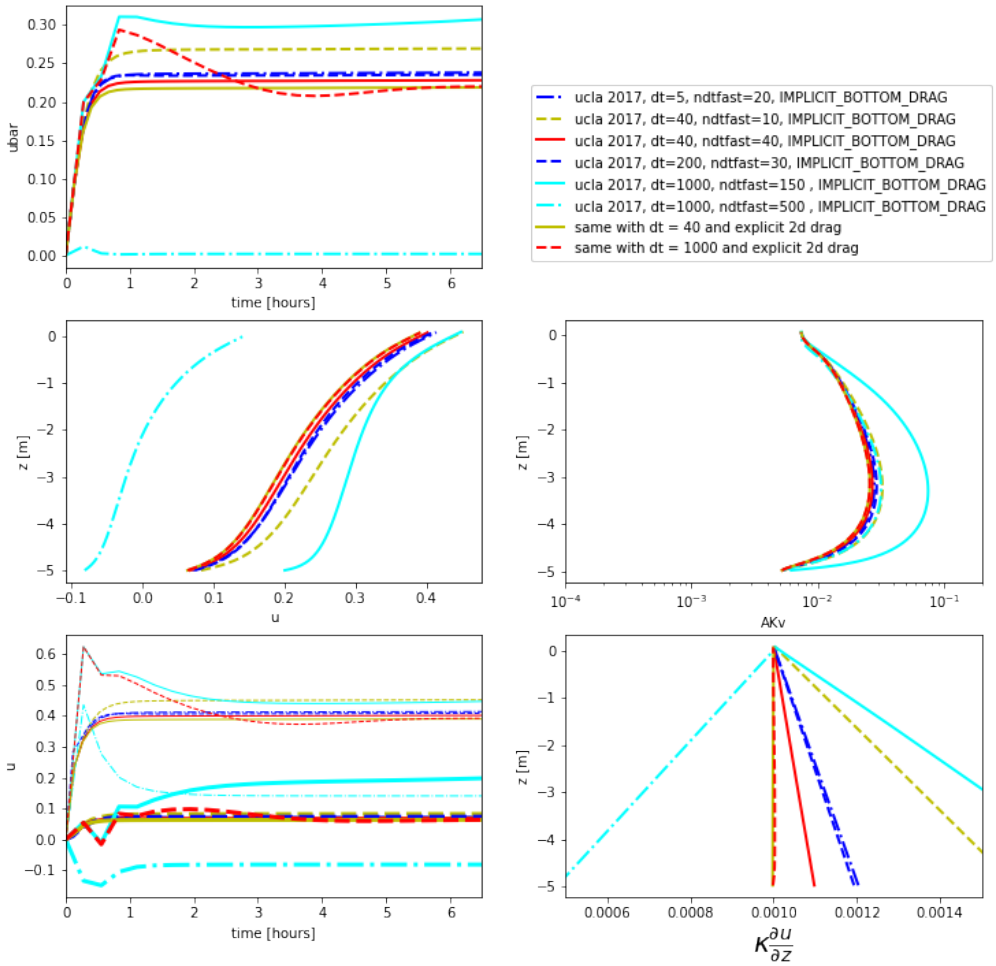

Figure 5: Same than Fig. 1 for UCLA 2017 with fully implicit drag.

of the implicit vertical viscosity solver (including bottom stress) is included in the predictor (pre_step3d.F). The bottom drag coefficient r_d computed at the predictor stage (based on velocities at u^n) is also used at the corrector stage if the key RECOMPUTE_RD is undefined (default choice). Tests with or without RECOMPUTE_RD, and tests using u^n or $u^{n+1/2}$ for r_d computation in step3d_uv1.F were similar for the shelf test case (with dt=40).

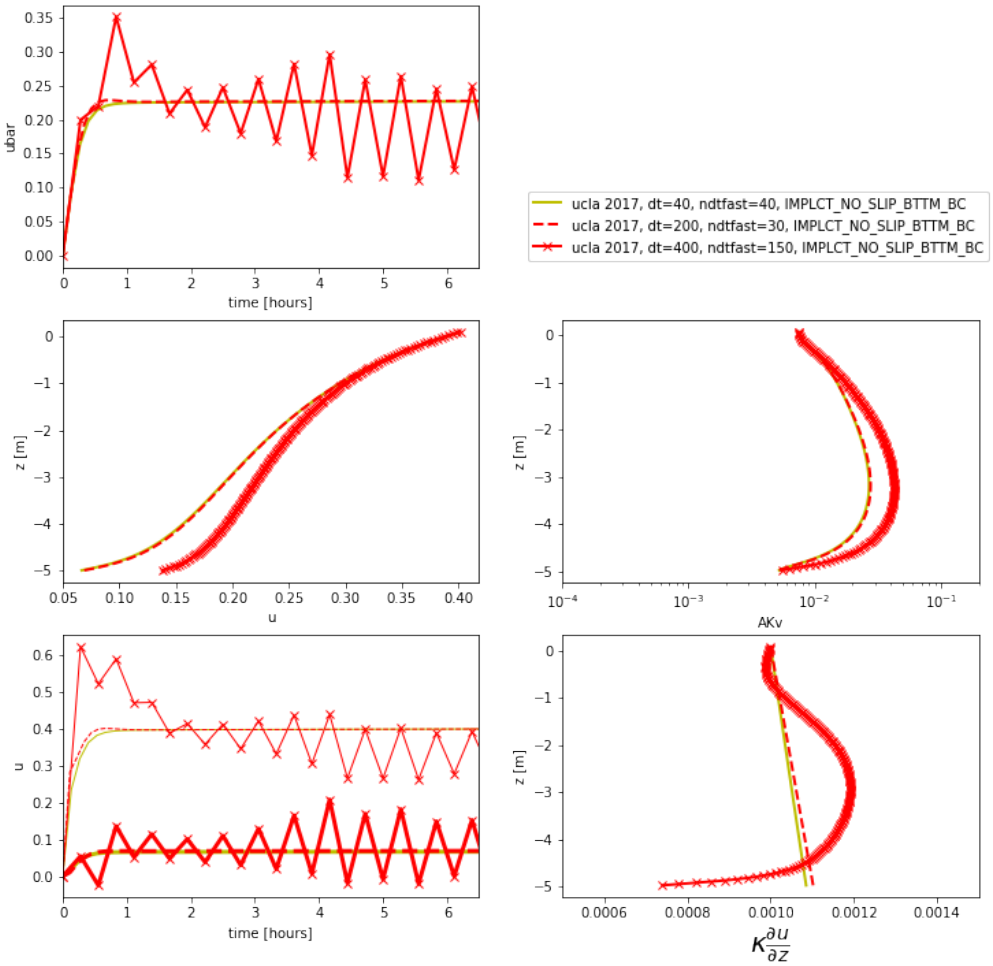

Figure 6: Same than Fig. 1 for UCLA 2017 with undef IMPLICIT_BOTTOM_DRAG and define IMPLCT_NO_SLIP_BTTM_BC.

I do not know what are the reasons for adding vertical mixing at both the predictor and corrector stages?

1.3.1 barotropic solver

The barotropic solver also has a few differences between versions concerning the drag. For all versions the 3D terms are integrated (including surface and bottom stresses) to be used in the barotropic equation as a forcing term (rufrc). However, how the drag is included in the 2d equations may vary.

In ROMS 2008 the drag is explicitly computed for each fast time-step using coefficients r_d:

```
      rubar(i,j)=rubar(i,j) - 0.5*(r_D(i,j)+r_D(i-1,j))
&                     *dm_u(i,j)*dn_u(i,j)*ubar(i,j,kstp)
```

Note that we do rufrc = rufrc - rubar at the first fast time-step so that the drag is not included twice (As 3d and 2d drags). And at the end of each time-step:

```
      DUnew=( (Dstp(i,j)+Dstp(i-1,j))*ubar(i,j,kstp)
&       +cff*(pm(i,j)+pm(i-1,j))*(pn(i,j)+pn(i-1,j))
&                                  *(rubar(i,j)+rufrc(i,j))

      ubar(i,j,knew)=DUnew/(Dnew(i,j)+Dnew(i-1,j))
```

However in ROMS 2017, the drag is not included in rubar, but instead it is included in an implicit way at the end of the computation by doing:

```
      DUnew=( (Dstp(i,j)+Dstp(i-1,j))*ubar(i,j,kstp)
&       +cff*(pm(i,j)+pm(i-1,j))*(pn(i,j)+pn(i-1,j))
&                                  *(rubar(i,j)+rufrc(i,j))

      ubar(i,j,knew)=DUnew/( Dnew(i,j)+Dnew(i-1,j)
&                           +dtfast*(r_D(i,j)+r_D(i-1,j)))
```

This difference is the reason why the 3d vertical mixing flux is not constant. If we go back to the explicit 2d drag in step2d.F, we get a constant vertical mixing flux and a much more stable code (Fig. 5).

In CROCO it seems that 2d drag is included in the equation only if a linear or a quadratic drag is used. *(to be investigated).*

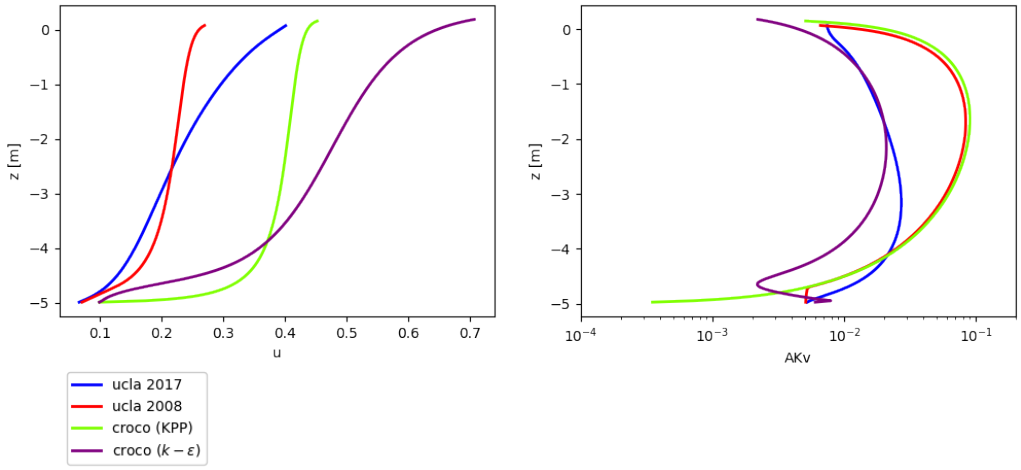

Figure 7: Vertical profiles of velocity (left) and vertical mixing coefficient K_v (right) at the end of the simulations.

1.4 KPP

The 3 models exhibit differences in the vertical profile of vertical mixing coefficient coming from either KPP or $k-\epsilon$ (Fig. 7). Differences between the different KPP versions are discussed in this section. These include differences for surface KPP, bottom KPP, as well as the rules used to merge them when surface and bottom boundary layers overlap (which is the case all the time in the shelf example).

1.4.1 surface KPP

First Kv is initialised as the result of background and Richardson mixing (if defined) as $Kv = Kv_b$. Then the mixed-layer depth h_{bl} and the turbulent velocity scale: w_b are computed (h_{bl} is always equal to the full depth of the water in the shelf example presented here). Computations are similar for all code versions at this stage. Some differences come from the choice of INT_AT_RHO_POINTS (default in ROMS 2017) or INT_AT_W_POINTS (default in ROMS 2008 / CROCO), but this is not sensitive in our example.

Minor differences also include values for the Critical bulk Richardson number ($Ric = 0.15$ or $Ric = 0.45$ depending on the choices of keys for CROCO and ROMS 2008, $Ric = 0.45$ for all cases in ROMS 2017 and $Ric = 0.15$ for all cases in ROMS 2022) and the constant for computing stabilization term due to Coriolis force ($C_{Ek} = 258$ for CROCO and ROMS 2017, $C_{Ek} = 215$ for ROMS 2008).

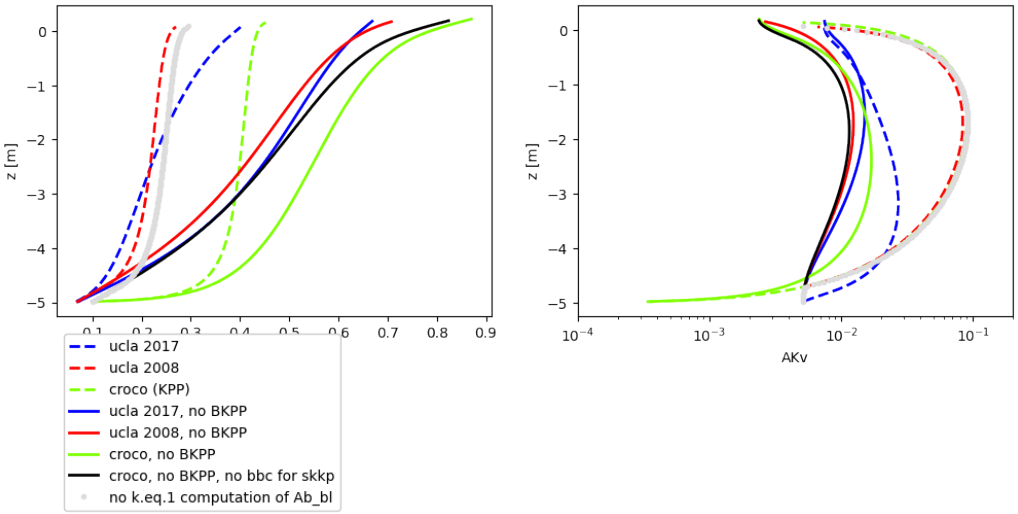

Figure 8: Vertical profiles of velocity (left) and vertical mixing coefficient K_v (right) at the end of the simulations when bottom KPP (BKPP) is on (dashed lines) or off (plain lines).

But the main differences seen in Fig. 7 are due to the choice of vertical functions used for the computation of the vertical mixing coefficients, and the boundary conditions used at the top and bottom. We present briefly the different methods in the next subsections.

1.4.2 ROMS 2008

The vertical mixing profile in the surface mixed layer in ROMS 2008 is computed as:

$$Kv = w_m \; h_{bl} \; G(z) \tag{1}$$

with the turbulent velocity scale w_m, the mixed-layer depth h_{bl}, and the shape function:

$$G(z) = \sigma_0(1-\sigma)^2/2 + \sigma(1 + \sigma\,(a_1 + a_2\; G_1 + a_3\; dG_1 dS)) \tag{2}$$

with $\sigma_0 = 0.07$ and $\sigma = \frac{z_s - z}{h_{bl}}$ is a function equal to 0 at the surface and 1 at the bottom of the mixed-layer.

The various coefficients are:

$$\begin{aligned} a_1 &= \sigma - 2. \\ a_2 &= 3 - 2\sigma \\ a_3 &= \sigma - 1. \\ G_1 &= Av_{bl}/(h_{bl}\, w_m + \epsilon) \\ dG_1 dS &= \min(0., -dAv_{bl}/(w_m + \epsilon)) \end{aligned}$$

where Av_{bl} and dAv_{bl} are coefficients used to match values and vertical derivatives of interior mixing coefficients at h_{bl}. If Kv_b is constant in the vertical, it simply gives: $Av_{bl} = Kv_b/(w_m\ h_{bl})$, and $dAv_{bl} = 0$, such that $Kv_{(z=h_{bl})} = Kv_b$. The generic profile is shown in Fig. 9.

In most cases the mixing coefficient at h_{bl} will be equal to the background value. When the surface mixed-layer extends down to the bottom, the value used for the boundary condition will be equal to the value at one level above bottom. In the shelf case this value is much higher than the background value because Richardson mixing (LMD_MIXING) is active there. This is why the mixing coefficient is so large a the bottom (not because of bottom KPP).

1.4.3 CROCO

The surface KPP in CROCO (lmd_skpp2005.F) is pretty close to the ROMS 2008 version with minor differences in the discretization. The vertical shape is also slightly different:

$$G(z) = \begin{cases} 0.5(\sigma - \sigma_0)^2/\sigma_0 + \sigma(1 + \sigma\,(a_1 + a_2\ G_1 + a_3\ dG_1 dS)), & \text{if } \sigma < \sigma_0 \\ \sigma(1 + \sigma\,(a_1 + a_2\ G_1 + a_3\ dG_1 dS)) & \text{otherwise} \end{cases} \tag{3}$$

which corresponds to the key KPP_PATCH (undefined by default) in ROMS 2008.

Another difference, clearly visible in Fig. 7, is the way the bottom condition is handled in surface KPP, where it is assumed that if the surface boundary layer extends to the bottom, the neutral boundary layer similarity theory should hold and we should have $K_v = \kappa\, u^* z$ at the bottom level ($k = 1$). Because of this condition, K_v is reaching much smaller values close to the bottom in CROCO than in other ROMS versions, where the condition used

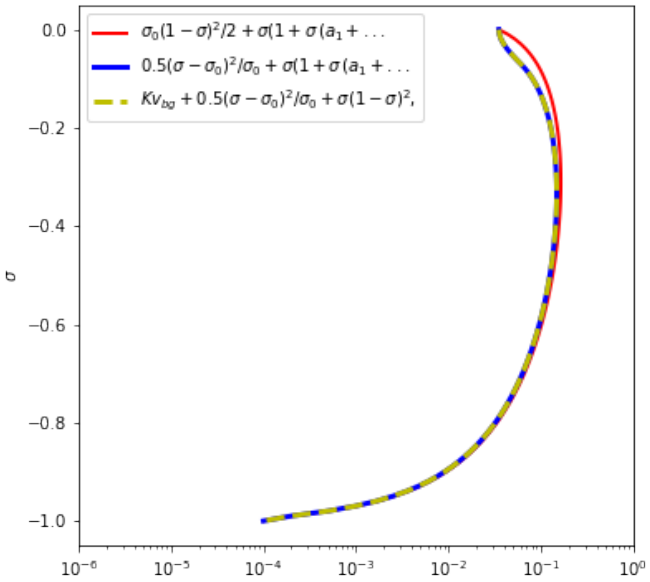

Figure 9: Shape functions used in KPP.

is continuity with the background value (including RIMIX). If we relax this condition (case "croco, no BKPP, no bbc for skkp" in Fig. 8), the result in CROCO and ROMS 2008 is very close.

1.4.4 ROMS 2017

The shape functions have been simplified in the ROMS 2017 version. The computation is now simply

$$Kv = Kv_{bg} + w_m \; h_{bl} \; G(z) \tag{4}$$

with

$$G(z) = \begin{cases} 0.5(\sigma-\sigma_0)^2/\sigma_0 + \sigma(1-\sigma)^2, & \text{if } \sigma < \sigma_0 \\ \sigma(1-\sigma)^2, & \text{otherwise} \end{cases} \tag{5}$$

with $\sigma_0 = 0.07$ and where $\sigma = \frac{z_s - z}{h_{bl}}$ is a function equal to 0 at the surface and 1 at the bottom of the mixed-layer.

The computation is much simpler and gives a very similar shape than other versions.

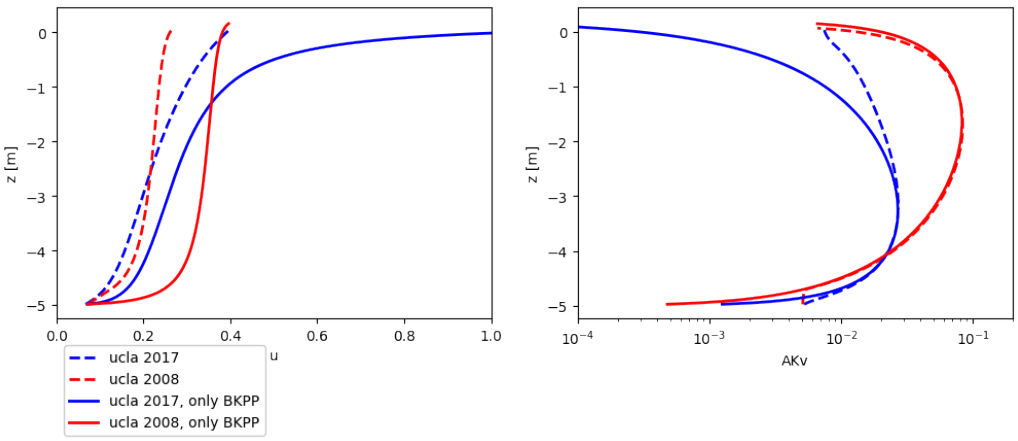

Figure 10: Vertical profiles of velocity (left) and vertical mixing coefficient K_v (right) at the end of the simulations when bottom KPP (BKPP) is on (dashed lines) or off (plain lines).

1.5 Bottom KPP

The K_v profiles are mostly driven by bottom KPP in our example.

If we plot K_v values from BKPP only, we get the same profiles in CROCO and ROMS 2008 (see Fig. 10). But the BKPP profile in ROMS 2017 is less intense, especially close to the surface.

Bottom KPP uses more or less the same shape functions than the ones used in the corresponding surface KPP for all code versions.

1.5.1 ROMS 2008

In ROMS 2008 it is:

$$Kv = w_{mb}\, h_{bbl}\, \sigma(1 + \sigma\,(a_1 + a_2\, G_1 + a_3\, dG_1 dS))$$

with the turbulent velocity scale: $w_{mb} = \kappa u^*$, the turbulent friction velocity $u^* = \sqrt{\|\tau_b\|}$, and $\sigma = \frac{z - z_b + Zob}{h_{bbl}}$ a function equal to $Zob/(h_{bbl})$ at the bottom and 1 at the top of the bbl.

Coefficients are:

$$\begin{aligned} a_1 &= \sigma - 2. \\ a_2 &= 3 - 2\sigma \\ a_3 &= \sigma - 1. \\ G_1 &= Av_b l/(h_{bbl}\, w_{mb} + \epsilon) \\ dG_1 dS &= \min(0., -dAv_b l/(w_{mb} + \epsilon)) \end{aligned}$$

where Av_{bl} and dAv_{bl} are coefficients used to match values and vertical derivatives of mixing coefficients (and which may have been updated by the surface KPP part of the code) at h_{bbl}. So when surface and bottom mixed layers overlap, the surface condition comes from the surface KPP profile.

In case of an overlap between surface and bottom layers, the maximum value is used: $Kv = \max(Kv_{skpp}, Kv_{bkpp})$. But one potential issue with this method is that the computation of BKPP depends on the SKPP computation (especially the surface condition).

1.5.2 ROMS 2017

$$Kv = w_{mb}\, h_{bbl}\, \sigma(1-\sigma)^2 \tag{6}$$

with the turbulent velocity scale: $w_{mb} = \kappa u^*$, the turbulent friction velocity $u^* = \sqrt{\|\tau_b\|}$, and $\sigma = \frac{z - z_b + Zob}{h_{bbl} + Zob}$ a function equal to $Zob/(h_{bbl} + Zob)$ at the bottom and 1 at the top of the bbl.

In this case, bottom and surface conditions are giving very small K_v values compared to the other BKPP versions.

Note that a κ seems to be missing in the 2017 and 2022 versions for the computation of w_{mb}:

```
      wmb=vonKar*sqrt( 0.333333333333*(
     &                 u(i,j,1,nstp)**2 +u(i+1,j,1,nstp)**2
     &                    +u(i,j,1,nstp)*u(i+1,j,1,nstp)
     &                +v(i,j,1,nstp)**2 +v(i,j+1,1,nstp)**2
     &                    +v(i,j,1,nstp)*v(i,j+1,1,nstp)
     &                                                  ) )
     &                   / log(1.+0.5*Hz(i,j,1)/Zob)
```

because $\kappa * \|u_b\| \frac{1}{\log(1+\frac{0.5Hz}{Zob})} = u^* \neq \kappa u^*$.

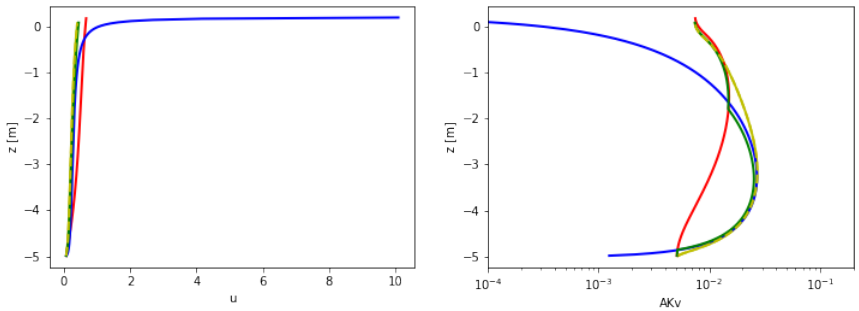

Figure 11: Vertical profiles of velocity (left) and vertical mixing coefficient K_v (right) at the end of the simulations for ROMS 2017 with or without BKPP, or with a different way to merge results from SKPP and BKPP.

One difference is the way coefficients are computed in case of an overlap between surface and bottom layers: $Kv = \sqrt{Kv_{skpp}^2 + Kv_{bkpp}^2}$.

Another difference is a time memory for hbbl (computed as the mean between the previous value and the new one):

```
# ifdef LMD_BKPP
      bbl(i,j)=0.5*bbl(i,j) + 0.5*hbbl(i,j,nstp)
# endif
```

as well as for K coefficients:

```
      Akv(i,j,k)= 0.5*Akv(i,j,k) + 0.5*Kv(i,j,k)
```

Overall this looks like a cleaner and more natural way to compute surface and bottom KPP profiles, and merge them together (with or without overlap), as seen in Fig. 11. Especially compared to the older versions where the overall shape was strongly influenced by the choices of Kv values at the boundaries.

1.5.3 CROCO

CROCO has 2 different BKPP versions, the lmd_bkpp1994.F (used by default) seems similar to the ROMS 2008 version. There is also a lmd_bkpp2005.F that resembles the ROMS 2017, but which seems incomplete *(to be checked and cleaned up)*.