Momentum Budget Evaluation: update for ASTE & ECCOv4 extensions

Helen Pillar [*1]

[1]Oden Institute for Computational Engineering and Sciences, UT Austin, TX

January 25, 2022

1 Introduction

We provided notes on closing the momentum budget and assessing contributions from different forcings to accompany Release 1 of the Arctic Subpolar gyre sTate Estimate (ASTE_R1), configured in MITgcm checkpoint 65q. This recipe also holds for ECCOv4r4. ASTE and ECCOv4 are currently undergoing further development, including an upgrade of the code base to a newer MITgcm checkpoint (**c67w** for ASTE, **c68d** for ECCOv4r5), necessitating correction of our method for budget closure. The purpose of these notes is to outline this correction and try to keep track of where additional sources may contribute for different CPP options to help keep the budget closed in the future.

2 Code changes relevant for momentum budget evaluation

(1) **Pressure Gradient Force**: Jean-Michel introduced a new diagnostic `[U,V]m_Phi[X,Y]` containing the tendency from the full pressure gradient force. Originally, we combined the existing diagnostic for the hydrostatic pressure gradient & explicitly-computed surface pressure gradient (`[U,V]m_dPHd[x,y]`) with our diagnostic for the implicitly-computed surface pressure gradient (`[U,V]m_dPsd[x,y]`) offline. Now `[U,V]m_dPHd[x,y]` has been retired and `[U,V]m_dPsd[x,y]` is not needed, we can just use `[U,V]m_Phi[X,Y]`. We don't need to make any further mods to this part of the code to close the momentum budget offline. Note that if you run non-hydrostatic MITgcm, the non-hydrostatic PGF is also contained within the new diagnostic `[U,V]m_Phi[X,Y]`. Below in green we give a summary of code updates (not writing out all DO loops etc. this time):

*helen.pillar@utexas.edu

```
forward_step
|- - -dynamics
|       |
|       |START BY COMPUTING HYDROSTATIC AND EXPLICIT SFC PGF. THESE ARE APPLIED TO ACCELERATE THE FLOW MUCH LATER IN S/R timestep.F
|       |Arrays for hydrostatic PGF [dPhiHydX(i,j), dPhiHydY(i,j)] and surface PGF [PhiSurfX(i,j), PhiSurfY(i,j)] all initialized to 0.
|       |If computing surface PGF explicitly (NOT TRUE FOR ASTE extension OR ECCOv4r5) do this here:
|       |IF (implicSurfPress.NE.1.)  THEN
|       |- - - calc_grad_phi_surf  explicit part of ∇_h p_sfc, output = [phiSurfX,phiSurfY]
|       |ENDIF
|       |START OF DYNAMICS LOOP OVER VERTICAL LEVELS
|       |DO k = 1,Nr
|       |- - - calc_phi_hyd  integrate hydrostatic balance for level k
|       |       |- -calc_grad_phi_hyd  compute ∇_h p_hyd for level k, output=[dPhyHydX,dPhiHydY]
|       |       o
|       |Diagnostic for hydrostatic PGF no longer filled here (and diagnostic has been retired).
|       |~~- - - DIAGNOSTIC_SCALE_FILL( dPhiHydX, tmpFac, 1,'Um_dPHdx',k,1,2,bi,bj,myThid )~~
|       |~~- - - DIAGNOSTIC_SCALE_FILL( dPhiHydY, tmpFac, 1,'Vm_dPHdy',k,1,2,bi,bj,myThid )~~
|       |
|       |EXPLICIT PGF IS IN ARRAYS [dPhiHydX,dPhiHydY] AND [phiSurfX,phiSurfY]. Only the hydrostatic PGF ≠ 0 here for ASTE extension & ECCOv4r5? - check
|       |
|       |- - - timestep  Compute remaining tendency from external forcing & implicit terms and accelerate the flow
|       |       |       input = guDissip, gvDissip, dPhiHydX, dPhyHydY, phiSurfX, phiSurfY, and advective tendency [gU,gV] via #include DYNVARS.h
|       |       | gUdPx(i,j) = 0.   _d 0  initialize u-mom tendency from TOTAL PGF
|       |       | gVdPy(i,j) = 0.   _d 0  initialize v-mom tendency from TOTAL PGF
|       |       | IF ( staggerTimeStep .OR. implicitIntGravWave ) THEN  we're here for ASTE and ECCOv4r5
|       |       |   gUdPx(i,j) = -phFac*dPhiHydX(i,j) - psFac*phiSurfX(i,j)  total explicit zonal PGF
|       |       |   gVdPy(i,j) = -phFac*dPhiHydY(i,j) - psFac*phiSurfY(i,j)  total explicit meridional PGF
|       |       |  ELSEIF ( implicSurfPress.NE.oneRL ) THEN
|       |       |   ...
|       |       | ENDIF
|       |       |  If using NH pressure (not in ASTE or ECCOv4r5), extra contribution added from explicit NH PGF to [gUdPx, gVdPy].
|       |       |  Now accelerate the flow by applying inertia + Coriolis + wind + explicit dissipation forcing and total explicit PGF
|       |       |  Note after this loop, [gU,gV] arrays contain velocity not acceleration
|       |       |  gU(i,j,k,bi,bj) = uVel(i,j,k,bi,bj) +
|       |       |                    deltaTMom*[gUtmp(i,j) + gUdPdx(i,j)]*_maskW(i,j,k,bi,bj)
|       |       |  gV(i,j,k,bi,bj) = vVel(i,j,k,bi,bj) +
|       |       |                    deltaTMom*[gVtmp(i,j) + gVdPdy(i,j)]*_maskS(i,j,k,bi,bj)
|       |       |  IF ( staggerTimeStep .OR. implicitIntGravWave ) THEN  we're here for ASTE and ECCOv4r5
|       |       |   gUdPx(i,j) = gUdPx(i,j)*_maskW(i,j,k,bi,bj)
|       |       |   gVdPy(i,j) = gVdPy(i,j)*_maskS(i,j,k,bi,bj)
|       |       |  ELSE
|       |       |   ...
|       |       |  ENDIF
|       |       |  First call to DIAGNOSTIC_FILL stores explicit PGF in [U,V]m_dPhi[X,Y] = hydrostatic part only for ASTE & ECCOv4r5?
|       |       |- - - DIAGNOSTICS_FILL( gUdPx,'Um_dPhiX',k,1,2,bi,bj,myThid)
|       |       |- - - DIAGNOSTICS_FILL( gVdPy,'Vm_dPhiY',k,1,2,bi,bj,myThid)
|       |       |
|       |       o
```

```
|       |ENDDO  end of dynamics k loop (1:Nr)
|       |
|      o
|
|- - -solve_for_pressure  solve elliptic eq. for p and update free surface displacement. Input/output include etaN via DYNVARS.h
|- - -momentum_correction_step  correct divergence in flow field with sfc pressure term from updated (just) surface displacement
|       |- - -calc_grad_phi_surf  Compute the PGF from updated free surface displacement, input = etaN, output = [phiSurfX,phiSurfY]
|       |- - -correction_step  Correct [u,v] with PGF from updated (implicit) eta inc. NH part (if chosen), input = [phiSurfX,phiSurfY]
|       |       | Do k = 1,Nr
|       |       |  #ifdef ALLOW_SOLVE4_PS_AND_DRAG  (need to compile with this option if using implicit bottom drag)
|       |       |   IF ( selectImplicitDrag.EQ.2 ) THEN  NOT ENTERING HERE: selectImplicitDrag=0 in ASTE and ECCOv4r5
|       |       |    ...this is where implicit PGF would be added to [U,V]m_dPhid[X,Y] if we ran with implicit bottom drag
|       |       |   ENDIF
|       |       |  #endif /* ALLOW_SOLVE4_PS_AND_DRAG */
|       |       |
|       |       |  Set sacling factor psFac, think this = 1 for us (not checked)
|       |       |  psFac = pfFacMom*implicSurfPress*recip_deepFacC(k)*recip_rhoFacC(k)
|       |       |  IF ( use3Dsolver ) THEN  NOT ENTERING HERE: use3Dsolver=F in ASTE and ECCOv4r5; performing 2D pressure solve
|       |       |   ...this is where implicit zonal PGF would be added to Um_dPhiX if using 3D p solver. NH contribution to zonal PGF would be added here too.
|       |       |  ELSE  ENTERING HERE! in ASTE and ECCOv4r5
|       |       |   gU_dpx(i,j) = -psFac*phiSurfX(i,j)*_maskW(i,j,k,bi,bj)
|       |       |  ENDIF
|       |       |  IF ( use3Dsolver ) THEN  NOT ENTERING HERE: use3Dsolver=F in ASTE and ECCOv4r5; performing 2D pressure solve
|       |       |   ...this is where implicit merid. PGF would be added to Vm_dPhiY if using 3D p solver. NH contribution to merid. PGF would be added here too.
|       |       |  ELSE  ENTERING HERE! in ASTE and ECCOv4r5
|       |       |   gV_dpy(i,j) = -psFac*phiSurfY(i,j)*_maskS(i,j,k,bi,bj)
|       |       |  ENDIF
|       |       |
|       |       |   Now write implicit surface PGF (including correction for non-divergence) to [U,V]m_dPhi[X,Y].
|       |       |   Note this is the 2nd call to fill these diagnostics; they already contain the hydrostatic PGF
|       |       |   As a result, we make this call with bibjFld = -2 to add in the implicit surface PGF without increasing the counter
|       |       |   The 2D surface PGF is added to all model levels:
|       |       |- - - DIAGNOSTICS_FILL( gU_dpx,'Um_dPhiX', k, 1, -2, bi, bj, myThid )
|       |       |- - - DIAGNOSTICS_FILL( gV_dpy,'Vm_dPhiY', k, 1, -2, bi, bj, myThid )
|       |       |   Here [u,v] are accelerated with deltaT * implicit (+NH) PGF
|       |       | ENDDO  end of k loop over vertical levels
|       |      o
|       |
|       |  We no longer need to output the implicit sfc PGF here. Skip requesting [U,V]m_dPsd[x,y] in data.diagnostics
|       |~~gU_eta(i,j) = (-1. _d 0)*phiSurfX(i,j)~~
|       |~~gV_eta(i,j) = (-1. _d 0)*phiSurfY(i,j)~~
|       |~~- - - DIAGNOSTICS_FILL(gU_eta,'Um_dPsdx',1,1,2,bi,bj,myThid)~~
|       |~~- - - DIAGNOSTICS_FILL(gV_eta,'Vm_dPsdy',1,1,2,bi,bj,myThid)~~
|      o
|
|  FINISHED TENDENCY CALCULATION, FINISHED VELOCITY UPDATE.
|
o
```

(2) **Bottom & Ice Shelf Drag**: Jean-Michel added the option for bottom drag to be computed implicitly by setting `selectImplicitDrag=2` in `data`. In ASTE and ECCOv4r5 this flag is unspecified and the default explicit computation (`selectImplicitDrag=0`) is performed. The original 3D diagnostics for momentum tendency from bottom drag (`[U,V]BotDrag`) have been removed and replaced with 2D tendencies for bottom stress (`botTau[X,Y]`, units = Nm^{-2}). The `botDrag[U,V]` arrays in the code have been appropriated to carry the bottom stress around (bit confusing at first!). If bottom drag is computed explicitly, it is still included in the original diagnostics for momentum tendency from all explicit dissipation (`[U,V]m_Diss`), but we are missing a diagnostic for this contribution, so our decomposition of (`[U,V]m_Diss`) no longer holds. If it is computed implicitly, it will be included in Jean-Michel's new diagnostics for implicit dissipation (`[U,V]m_ImplD`). Whether we choose an implicit or explicit calculation, it could be useful to isolate the forcing from bottom drag, so we're adding the diagnostic back in. We also add diagnostics for drag exerted by ice shelves now this is included in ECCOv4r5. As for bottom drag, this is included in `[U,V]m_Diss` if `selectImplicitDrag=0`, else it will be in `[U,V]m_ImplD`. Below in green we give a summary of code updates and in red we give a summary of further code mods for outputting the momentum tendency from bottom drag (not writing out all DO loops etc. this time). See original doc for more detail.

```
forward_step
|- - -dynamics
|       |
|       |botDragU(i,j,bi,bj) = 0.   _d 0  Initialize 2D arrays for carrying bottom stress
|       |botDragV(i,j,bi,bj) = 0.   _d 0  These are carried around in common block FFIELDS.h
|       |
|       |START OF DYNAMICS LOOP OVER VERTICAL LEVELS
|       |DO k = 1,Nr
|       |Call wrapper for momentum tendency explicit calculations:
|       |- - - mom_vecinv    inputs/outputs include arrays botDragU,botDragV via common block FFIELDS.h.
|       |       |                outputs also include gUDiss, gVDiss
|       |       |
|       |       |Initialize local arrays to hold total explicit dissipation tendency (name mismatch below ok - not case sensitive):
|       |       |gUDiss(i,j)=0.
|       |       |gVDiss(i,j)=0.
|       |       |And initialize arrays to hold explicit bottom drag tendency and ice shelf drag tendency:
|       |       |tmpDragU(i,j)=0.
|       |       |tmpDragV(i,j)=0.
|       |       |tmpDragU2(i,j)=0.
|       |       |tmpDragV2(i,j)=0.
|       |       |Switch on calculation of bottom drag in this routine, only if we chose to compute it explicitly:
|       |       |We enter here for ECCOv4r5 and ASTE as we run with explicit drag (selectImplicitDrag=0) and no slip bottom bcs (no_slip_bottom = .TRUE.)
|       |       |IF (selectImplicitDrag.EQ.0 .AND. ( no_slip_bottom .OR.
|       |       |   selectBotDragQuadr.GE.0 .OR. bottomDragLinear.NE.0.  )  )  THEN
|       |       |   bottomDragTerms=.TRUE.
|       |       |ENDIF
|       |       |
|       |       |Now calculate explicit contributions to dissipation, starting with harmonic and biharmonic viscosity:
|       |       |- - - mom_vi_hdissip      output = updated guDiss, gvDiss = tendency from harmonic & biharmonic viscosity
|       |       |
```

Find all other explicit contributions to zonal momentum dissipation and continue accumulating in gUDiss:

```
--- mom_u_sidedrag        output = vF = tendency from lateral drag
gUDiss(i,j)=gUDiss(i,j)+vF(i,j)
```

The call to the original mom_*_bottomdrag.F S/R has been replaced with a call to new S/R mom_*_botdrag_coeff.F:

```
---- mom_u_bottom_drag
     |---- DIAGNOSTICS_FILL( uDragTerms, 'UBotDrag',k,1,2,bi,bk,myThid)
     o
```

If we decided above that we are still using explicit bottom drag, compute drag coefficient for zonal momentum here and add to Um_Diss.

(For an implicit calculation, the active call is later in mom_*_implicit_r.F)

```
IF ( bottomDragTerms ) THEN
--- mom_u_botdrag_coeff        input = uFld, vFld, kappaRU, output = cDrag (2D array, units m/s)
     |                                  where: -cDrag * U_bot * rUnit2mass = taux_bot
     o
gUdiss(i,j) = gUdiss(i,j) -
          cDrag(i,j)*uFld(i,j)*_recip_hFacW(i,j,k,bi,bj)*recip_drF(k)
```

Here botDragU(i,j)=0, now we update it:

```
botDragU(i,j) = botDragU(i,j) - cDrag(i,j)*uFld(i,j)*rUnit2mass
```

Bit confusing: The botDrag* arrays are now used to hold bottom stress: TAUX = -cDrag*U_bot*rUnit2mass, and passed back out in FFIELDS.h

This is why - right at the end of dynamics.F - we make a call to fill botTauX with botDragU

Assigning the momentum tendency from bottom drag to a new tmp array (currently containing zeros) so we can call to fill diagnostic:

cDrag (m/s) * uFld (m/s) * recip_drF (1/m) = ms^{-2}:

```
tmpDragU(i,j) = tmpDragU(i,j)
           - cDrag(i,j)*uFld(i,j)* recip_hFacW(i,j,k,bi,bj)*recip_drF(k)
--- DIAGNOSTICS_FILL( tmpDragU,'UBotDrag',k,1,2,bi,bj,myThid)
ENDIF
```

Drag from shelf ice gets added here if we are using it (true for ECCOv4r5, not for ASTE)

```
IF ( useShelfIce .AND. selectImplicitDrag.EQ.0 )
--- shelfice_u_drag_coeff        input = uFld, vFld, kappaRU, KE, output = cDrag (2D array, units m/s)
     |                                  where: -cDrag * U_bot * rUnit2mass = taux_bot
     o
gUdiss(i,j) = gUdiss(i,j) -
          cDrag(i,j)*uFld(i,j)*_recip_hFacW(i,j,k,bi,bj)*recip_drF(k)
```

Assigning the momentum tendency from ice shelf drag to a new tmp array (currently containing zeros) so we can call to fill diagnostic:

cDrag (m/s) * uFld (m/s) * recip_drF (1/m) = ms^{-2}:

```
tmpDragU2(i,j) = tmpDragU2(i,j)
           - cDrag(i,j)*uFld(i,j)* recip_hFacW(i,j,k,bi,bj)*recip_drF(k)
--- DIAGNOSTICS_FILL( tmpDragU2,'UShIDrag',k,1,2,bi,bj,myThid)
ENDIF
```

Find all other explicit contributions to meridional momentum dissipation and accumulate in gVDiss.

```
--- mom_v_sidedrag        output = vF = tendency from lateral drag
gVDiss(i,j)=gVDiss(i,j)+vF(i,j)
```

The call to the original mom_*_bottomdrag.F S/R has been replaced with a call to new S/R mom_*_botdrag_coeff.F:

```
---- mom_v_bottom_drag
     |---- DIAGNOSTICS_FILL( vDragTerms, 'VBotDrag',k,1,2,bi,bj,myThid)
     o
```

If we decided above that we are still using explicit bottom drag, compute drag coefficient for meridional momentum here and add to Vm_Diss:

```
|       |       |IF ( bottomDragTerms ) THEN
|       |       |--- mom_v_botdrag_coeff        input = uFld, vFld, kappaRV, output = cDrag (2D array, units m/s)
|       |       |      |                                where: -cDrag * V_bot * rUnit2mass = tauy_bot
|       |       |      o
|       |       |gVdiss(i,j) = gVdiss(i,j) -
|       |       |           cDrag(i,j)*vFld(i,j)*_recip_hFacS(i,j,k,bi,bj)*recip_drF(k)
|       |       |Here botDragV(i,j)=0, now we update it:
|       |       |botDragV(i,j) = botDragV(i,j) - cDrag(i,j)*vFld(i,j)*rUnit2mass
|       |       |           cDrag(i,j)*vFld(i,j)*_recip_hFacS(i,j,k,bi,bj)*recip_drF(k)
|       |       |Bit confusing: The botDrag* arrays are now used to hold bottom stress: TAUY = -cDrag*V_bot*rUnit2mass, and passed back out in FFIELDS.h
|       |       |This is why - right at the end of dynamics.F - we make a call to fill botTauY with botDragV
|       |       |Assigning the momentum tendency from bottom drag to a new tmp array (currently containing zeros) so we can call to fill diagnostic:
|       |       |tmpDragV(i,j) = tmpDragV(i,j)
|       |       |           - cDrag(i,j)*vFld(i,j)*recip_hFacS(i,j,k,bi,bj)*recip_drF(k)
|       |       |--- DIAGNOSTICS_FILL( tmpDragV,'VBotDrag',k,1,2,bi,bj,myThid)
|       |       |ENDIF
|       |       |Drag from shelf ice gets added here if we are using it.
|       |       |
|       |       |Drag from shelf ice gets added here if we are using it (true for ECCOv4r5, not for ASTE)
|       |       |IF ( useShelfIce .AND. selectImplicitDrag.EQ.0 )
|       |       |--- shelfice_v_drag_coeff        input = uFld, vFld, kappaRU, KE, output = cDrag (2D array, units m/s)
|       |       |      |                                where: -cDrag * U_bot * rUnit2mass = taux_bot
|       |       |      o
|       |       |gVdiss(i,j) = gVdiss(i,j) -
|       |       |           cDrag(i,j)*vFld(i,j)*_recip_hFacS(i,j,k,bi,bj)*recip_drF(k)
|       |       |Assigning the momentum tendency from ice shelf drag to a new tmp array (currently containing zeros) so we can call to fill diagnostic:
|       |       |cDrag (m/s) * vFld (m/s) * recip_drF (1/m) = ms^-2:
|       |       |tmpDragV2(i,j) = tmpDragV2(i,j)
|       |       |            - cDrag(i,j)*vFld(i,j)* recip_hFacS(i,j,k,bi,bj)*recip_drF(k)
|       |       |--- DIAGNOSTICS_FILL( tmpDragV2,'VShIDrag',k,1,2,bi,bj,myThid)
|       |       |ENDIF
|       |       |
|       |       |Arrays [gUDiss,gVDiss holding total dissipation are passed back out as [gUDissip,gVDissip] ]
|       |       |If we are running with implicit bottom drag (selectImplicitDrag<2), the bottom drag is contained in these arrays on exiting.
|       |       |Otherwise, the implicit bottom drag gets stored in the total implicit dissipation diagnostic [U,V]m_ImplD (See below).
|       |       o
|       |
|       |--- timestep  Compute remaining tendency from external forcing and accelerate the flow at each level
|       |       |         inputs include arrays [gUDissip, gvDissip] = [gUDiss, gvDiss] output from mom_vecinv.F, holding tendency from explicit dissipation
|       |       |
|       |       |Acceleration from dissipation is accumulated with other terms in gUtmp,gVtmp arrays and then update velocity in gU,gV arrays
|       |       |Before calling to fill diagnostic for total explicit dissipation:
|       |       |--- DIAGNOSTICS_FILL(guDissip,'Um_Diss ',k,1,2,bi,bj,myThid)
|       |       |--- DIAGNOSTICS_FILL(gvDissip,'Vm_Diss ',k,1,2,bi,bj,myThid)
|       |       |
|       |       o
|       |
|       |ENDDO  end of dynamics k loop (1:Nr)
```

ADD FORCING FROM IMPLICIT TERMS AND CORRECT FOR DIVERGENCE

NB: after leaving S/R timestep, [gU,gV] arrays hold updated velocity NOT acceleration.

[uvel,vvel] arrays contain old velocity, prior to acceleration from explicit terms

~~If (momImplVertAdv) THEN~~ FALSE in ASTE (and as default) so we never used to enter the following S/R

But the following condition is true, because we have compiled with #define INCLUDE_IMPLVERTADV_CODE in CPP_OPTIONS.h

and we also have ALLOW_MOM_COMMON and implicitViscosity

```
If (defined (INCLUDE_IMPLVERTADV_CODE) && If (ALLOW_MOM_COMMON) && IF (
  momImplVertAdv .OR. implicitViscosity .OR. selectImplicitDrag.GE.1 ) THEN
```

--- mom_u_implicit_r.F apply forcing from implicit vertical advection and viscosity using kappaRU computed way earlier in calc_viscosity.F

input/output includes botDragU,botDragV available via common block FFIELDS.h

inputs also include uvel,vvel via common block DYNVARS.h

Initialize array for 2D drag coeff and local copy of zonal velocity value on entering S/R:

```
cDrag(i,j) = 0.   _d 0
```

Make copy of velocity: gU = zonal velocity on exiting timestep.F, available here via #include DYNVARS.h.

This is needed below for us to back out the total acceleration from the implicit forcing terms.

```
IF ( useDiagnostics .AND.
     ( implicitViscosity .OR. selectImplicitDrag.GE.1 ) ) THEN
   delUdt(i,j,k) = gU(i,j,k,bi,bj)
ENDIF

IF ( selectImplicitDrag.GE.1 ) THEN
   IF ( no_slip_bottom .OR. selectBotDragQuadr.GE.0 .OR.
           bottomDragLinear.NE.0.   )  THEN
```

Here's the call to compute bottom drag coefficient cDrag for zonal velocity if we didn't choose to compute it explicitly earlier on

I am not sure why the drag coefficient is computed with the oldest velocity (uvel, vvel) (i.e., prior to acceleration by explicit terms)

But later on in correction_step.F we recompute it again...I think after accelerating the flow with the other implicit terms.

-cDrag * U_bot * rUnit2mass = taux_bot

```
      DO k = 1,Nr
```

--- mom_u_botdrag_coeff inputs = [uvel,vvel], output = 2D array cDrag

Add drag coefficient to main diagonal "c5d" of tridiagonal matrix (i.e, multiplies local u(i,j)?) to solve implicit viscosity and drag:

```
         DO j=jMin,jMax
            DO i=iMin,iMax
              c5d(i,j,k) = c5d(i,j,k) +
                cDrag(i,j)*deltaTMom*_recip_hFacW(i,j,k,bi,bj)*recip_drF(k)
            ENDDO
         ENDDO
```

cDrag has units m/s, rUnit2mass has units kg/m^3, so botDragU has units N/m^2/ms^{-1} here?

```
      DO j=jMin,jMax
       DO i=iMin,iMax
         botDragU(i,j,bi,bj)=botDragU(i,j,bi,bj)+cDrag(i,j)*rUnit2mass
       ENDDO
      ENDDO
```

Contribution from ice shelf drag also added to c5d here if implicit, and cDrag is overwritten

Solve for the new flow and back out the total implicit forcing as the difference of the updated and original velocity:

We enter this S/R with [gU,gV] = velocity predictions accelerated by the explicit forcings: $u^* = u^n + \Delta t G_u^{n+1/2}$

The velocity at the next time step (n+1) is equal to u^* plus any forcing exerted by implicit terms (dependent on the future velocity u^{n+1}):

$u^{n+1} - \Delta t\ f(u^{n+1}) = u^*$, where f includes implicit drag, viscosity etc.

Following the manual, consider a simple case where we only have implicit vertical diffusion, then $f(u^{n+1}) = \partial_z \kappa \partial_z u^{n+1}$

giving $u^{n+1} = L_u^{-1}(u^*)$, where $L_u = [1 + \Delta t \partial_z \kappa \partial_z]$

This inversion is what we now solve below:

--- solve_tridiagonal inputs = coeffs for L_u^{-1}, output = gU = updated velocity (i.e., u^{n+1}).

..although final adjustments will be made in correction_step.F below to correct for divergence

```
IF ( useDiagnostics ) .AND.
     (implicitViscosity .OR. selectImplicitDrag.GE.1 ) ) THEN
```

Here's the back out the total implicit forcing from the updated velocity and write it to diagnostic:

```
     diagName = 'Um_ImplD'
```

loop over all grid cells i,j,k and find acceleration due to implicit terms = $(u^{n+1} - u^{n+1/2})/\Delta t$

```
     delUdt(i,j,k) = ( gU(i,j,k,bi,bj)-delUdt(i,j,k) )*recip_dT
--- DIAGNOSTICS_FILL(delUdt,diagName, 0,Nr, 2,bi,bj, myThid)
ENDIF
```

We just aggregated all implicit dissipation into one diagnostic. But we might want to examine how different terms contribute to the total implicit forcing.

Even if we don't care about this, it would be good to write the tendency from implicit bottom drag...

to be consistent with us allowing it to be written for explicit calculation (in mom_vecinv.F) We will do this in correction_step.F

--- mom_v_implicit_r.F apply v forcing from implicit vertical advection and viscosity using kappaRU computed way earlier in calc_viscosity.F

input/output includes botDragU,botDragV available via common block FFIELDS.h

inputs also include uvel,vvel via common block DYNVARS.h

Not writing this all our here...analogous calculation as shown above for u ...eventually get to fill diagnostic for total acceleration of v by implicit forcing:

```
IF ( useDiagnostics ) .AND.
     (implicitViscosity .OR. selectImplicitDrag.GE.1 ) ) THEN
```

Here's the back out the total implicit forcing from the updated velocity and write it to diagnostic:

```
     diagName = 'Vm_ImplD'
```

loop over all grid cells i,j,k and find acceleration due to implicit terms = $(v^{n+1} - v^{n+1/2})/\Delta t$

```
     delVdt(i,j,k) = ( gV(i,j,k,bi,bj)-delVdt(i,j,k) )*recip_dT
--- DIAGNOSTICS_FILL(delVdt,diagName, 0,Nr, 2,bi,bj, myThid)
ENDIF
```

Or if we didn't compile with INCLUDE_IMPLVERTADV_CODE we call impldiff.F instead and avoid the tridiagonal solver (we always used to call this S/R)

We do not enter here for ASTE and I think not for ECCOv4r5 either

```
ELSE
  IF ( implicitViscosity ) THEN
```

apply forcing from implicit vertical viscosity using [kappaRU,kappaRV] computed way earlier in calc_viscosity.F

Here's where we used to output the implicit viscosity, but this is no longer reached

--- impldiff.F inputs = kappaRU and gU (u velocity accelerated by explicit terms)

```
     --- DIAGNOSTICS_FILL(df,'Um_Impl ',k,1,2,bi,bj,myThid)
```

```
|      |--- impldiff.F  inputs = kappaRV and gV (v velocity accelerated by explicit terms)
|      |      |--- DIAGNOSTICS_FILL(df,'Vm_Impl ',k,1,2,bi,bj,myThid)
|      |      o
|      |ENDIF
|      |
|      o
|
|--solve_for_pressure  solve elliptic eq. for p and update free surface displacement
|
|--momentum_correction_step  correct divergence in flow field with sfc pressure term from updated (just) surface displacement
|      |---correction_step      Inputs include botDragU, botDragV via common block FFIELDS.h
|      |      |
|      |      |Here's where we should initalize local tmp arrays for 3D cDrag for both u and v momentum
|      |      |And also initalize local tmp arrays for 3D momentum tendency from implicit bottom drag for both u and v
|      |      | Do k = 1,Nr
|      |      |  #ifdef ALLOW_SOLVE4_PS_AND_DRAG  (need to compile with this option if using implicit bottom drag)
|      |      |   IF ( selectImplicitDrag.EQ.2 ) THEN  NOT ENTERING HERE: selectImplicitDrag=0 in ASTE and ECCOv4r5
|      |      |    ...if we arrive here we have already stored implicit bottom drag in [U,V]m_ImplD,
|      |      |    but we're about to apply a pressure correction needed to ensure non-divergence of the new velocity (uvel,vvel)
|      |      |    So we'll have to ammend the implicit tendencies too.
|      |      |    If I follow correctly, we left mom_*_botdrag_coeff with botDrag[U,V] containing drag with units of Nm^-2/ms^-1...
|      |      |    So when we multiply by the *new* velocity here, botDrag[U,V] contain a stress with units Nm^-2
|      |      |    DO j=jMin,jMax
|      |      |     DO i=iMin,iMax
|      |      |     k = MAX( 1, MIN( kLowC(i-1,j,bi,bj), kLowC(i,j,bi,bk)) )
|      |      |     botDragU(i,j,bi,bj)=-botDragU(i,j,bi,bj)*uvel(i,j,k,bi,bj)
|      |      |        Here's where we should make a 3D store of cDrag from 2D botDragU
|      |      |     ENDDO
|      |      |    ENDDO
|      |      |    Write the bottom stress diagnostic:
|      |      |    DIAGNOSTICS_FILL( botDragU, 'botTauX', 0, 1, 1, bi, bj, myThid )
|      |      |     Here's where we should compute the momentum tendency from implicit bottom drag and call to fill the UBotDrag diagnostic
|      |      |     Perform an analogous calculation for the bottom stress acting on the meridional *new* bottom velocity:
|      |      |    DO j=jMin,jMax
|      |      |     DO i=iMin,iMax
|      |      |     k = MAX( 1, MIN( kLowC(i,j-1,bi,bj), kLowC(i,j,bi,bk)) )
|      |      |     botDragV(i,j,bi,bj)=-botDragV(i,j,bi,bj)*vvel(i,j,k,bi,bj)
|      |      |        Here's where we should make a 3D store of cDrag from 2D botDragV
|      |      |     ENDDO
|      |      |    ENDDO
|      |      |    Write the bottom stress diagnostic:
|      |      |    DIAGNOSTICS_FILL( botDragV, 'botTauY', 0, 1, 1, bi, bj, myThid )
|      |      |     Here's where we should compute the momentum tendency from implicit bottom drag and call to fill the VBotDrag diagnostic
|      |      |    ENDIF
|      |      |  #endif /* ALLOW_SOLVE4_PS_AND_DRAG */
|      |      | ENDDO  end of k loop over vertical levels
|      |      | shelf drag diagnostics are computed here too if using. Then exit.
|      |      o
```

| |
| o
o

3 MITgcm Required Diagnostic Output

A closed momentum budget at [u,v] points on the k-th model level of ASTE configured in checkpoint 67w or ECCOv4r5 configured in checkpoint 68d is given by:

$$\begin{aligned}\frac{\texttt{TOTUTEND(i,j,k)}}{86400} = \texttt{Um\_Cori(i,j,k)} &+ \Big(\texttt{Um\_Advec(i,j,k)} - \texttt{Um\_Cori(i,j,k)}\Big) \\ &+ \texttt{Um\_Diss(i,j,k)} + \texttt{Um\_Ext(i,j,k)} + \texttt{Um\_dPhiX(i,j,k)} \\ &+ \texttt{AB\_gU(i,j,k)} + \texttt{Um\_ImplD(i,j,k)}\end{aligned} \tag{1}$$

$$\begin{aligned}\frac{\texttt{TOTVTEND(i,j,k)}}{86400} = \texttt{Vm\_Cori(i,j,k)} &+ \Big(\texttt{Vm\_Advec(i,j,k)} - \texttt{Vm\_Cori(i,j,k)}\Big) \\ &+ \texttt{Vm\_Diss(i,j,k)} + \texttt{Vm\_Ext(i,j,k)} + \texttt{Vm\_dPhiY(i,j,k)} \\ &+ \texttt{AB\_gV(i,j,k)} + \texttt{Vm\_ImplD(i,j,k)}\end{aligned} \tag{2}$$

Updated terms are shown in blue. Momentum closure has been validated in a 3 month test run by Ou Wang outputting 5-day mean diagnostics; below we plot the momentum budget on 20 February 1992 to demonstrate closure. Figures 1 & 2 show terms contributing to a closed zonal momentum budget (Eq.(1)) at depths of 5m and 300m, respectively. This closure is verified in Fig. 3. In Figs. 4 & 5 we show terms contributing to a closed meridional momentum budget (Eq.(1)) at depths of 5m and 300m, respectively. This closure is verified in Fig. 6. Note that all diagnostics are output on the correct grid (i.e., the [u,v]-momentum tendencies are output at the [u,v]-velocity points, respectively). For ECCOv4r5 and ASTE, `[U,V]m_Ext` still describes momentum tendency from the wind alone. We also define additional diagnostics to allow the decomposition of advection and explicit dissipation into dynamically distinct contributions:

$$\texttt{Um\_Advec} = \texttt{Um\_Cori} + \texttt{Um\_AdvZ3} + \texttt{Um\_AdvRe} + \texttt{Um\_dKEdx} \tag{3}$$

$$\texttt{Um\_Diss} = \texttt{Um\_Diss2} + \texttt{Um\_Diss4} + \texttt{UBotDrag} + \texttt{USidDrag} + \texttt{UShIDrag} \tag{4}$$

where `UShIDrag` is the ice shelf drag. An analogous decomposition can be written for v-momentum tendencies. Figures 7 & 8 show these decomposition; confirmation that they are correct is given in Fig. 9. I suggest most folks using these diagnostics for dynamical investigation may like to combine the Adams Bashforth (`AB_g[U,V]`) and Eulerian (`TOT[U,V]TEND`) tendency in their offline analyses. They may also like to combine dissipation computed explicitly (`[U,V]m_Diss`) and implicitly (`[U,V]m_ImplD`) although their dynamical distinction is interesting. In ASTE the implicit term describes momentum tendency from vertical viscosity. I believe this is also still the case for ECCOv4r5; this could be verified by comparing to our original `[U,V]m_Impl` diagnostic.

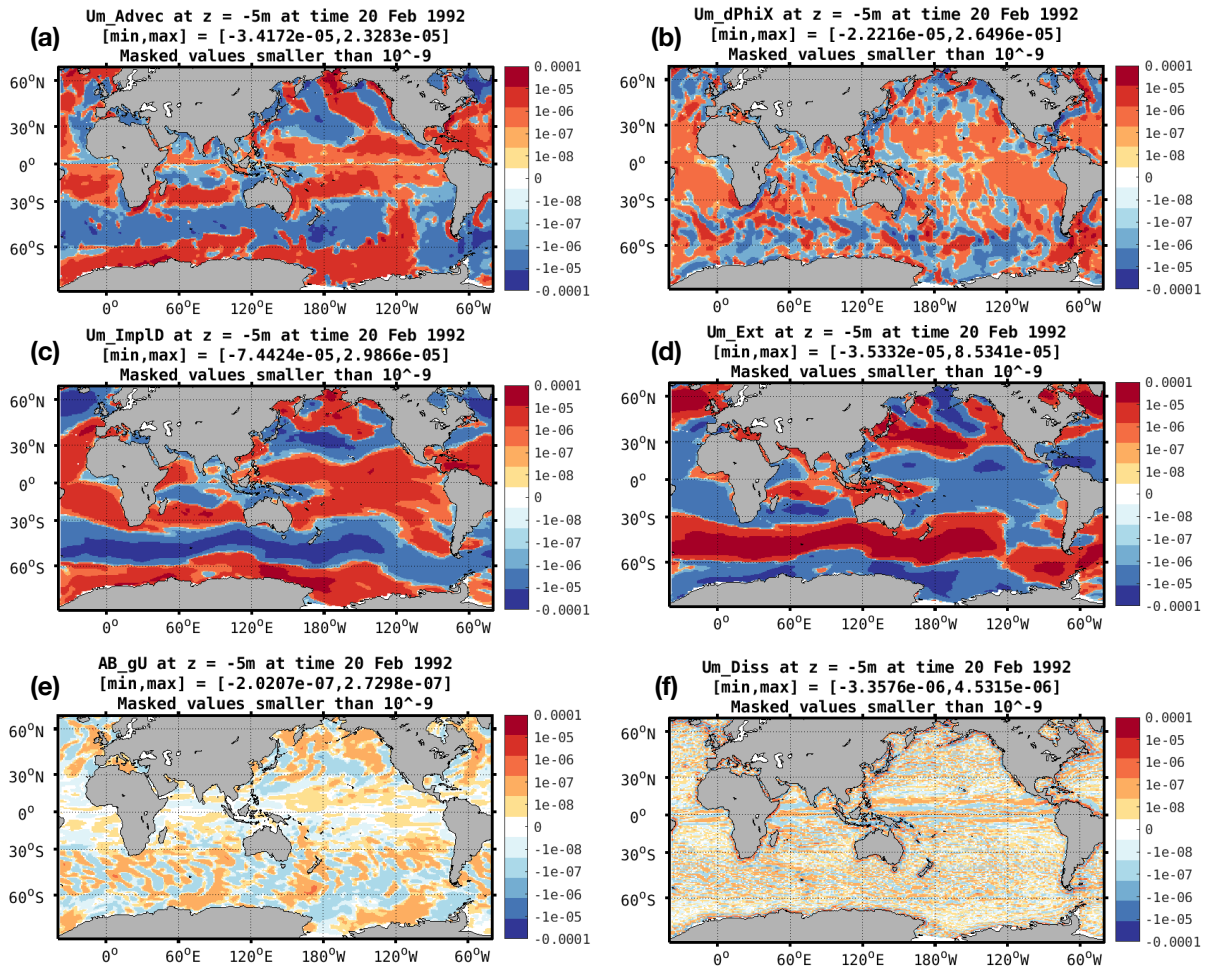

Figure 1: Terms contributing to the ECCOv4r5 u-momentum tendency (ms^{-2}) at level 1 (depth = 5 m). Their sum (RHS of Eq. (1)) explains the Eulerian tendency of the zonal velocity at this depth level (see Fig. 3). In this configuration, only wind forcing contributes to `[Um_Ext,Vm_Ext]`, so these terms are zero for all depths but the surface level.

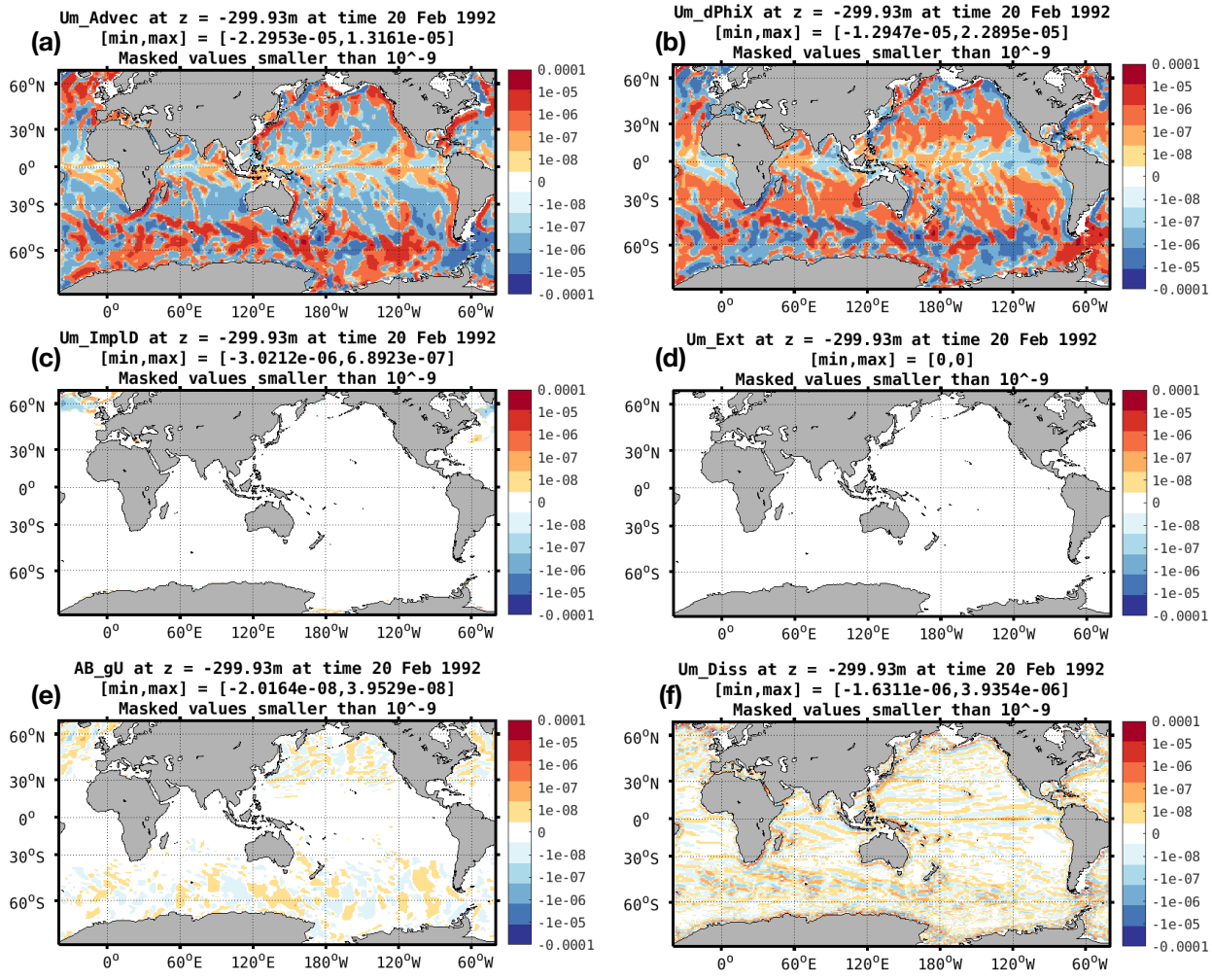

Figure 2: As for Fig. 1 but showing terms contributing to the ECCOv4r5 u-momentum tendency (ms^{-2}) at level 20 (depth = 299.9 m). Their sum (RHS of Eq. (1)) explains the Eulerian tendency of the zonal velocity at this depth level (see Fig. 3). In this configuration, only wind forcing contributes to `[Um_Ext,Vm_Ext]`, so these terms are zero for all depths but the surface level.

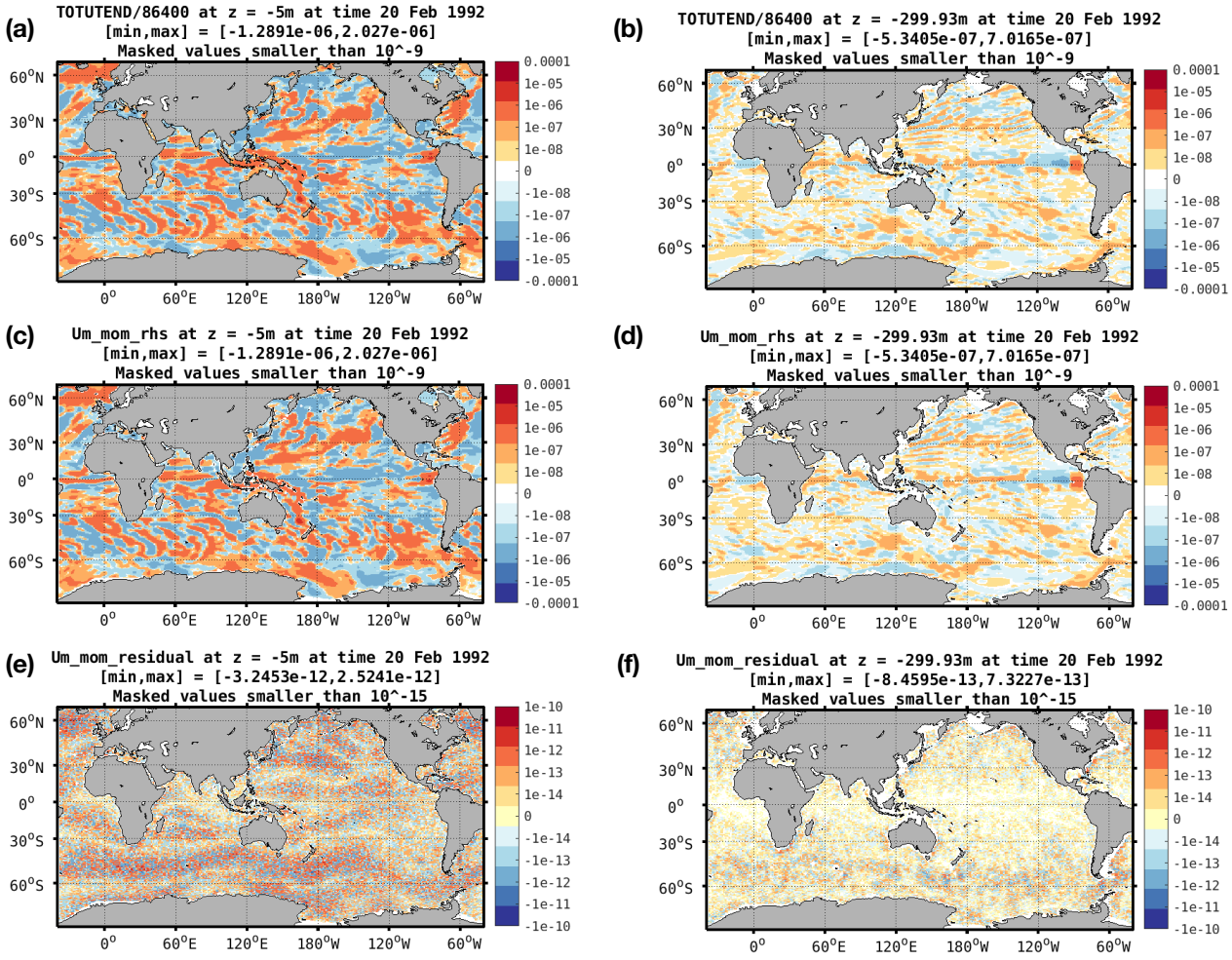

Figure 3: u-momentum (ms^{-2}) closure at (left) level 1 (depth = 5 m) and (right) level 20 (depth = 299.93 m). Closure is determined by subtracting (c,d) the sum of all RHS tendencies as shown in Figs. 1 & 2 from (a,b) `TOTUTEND` = the Eulerian tendency of the zonal velocity. The residual is shown in (e,f) to be 6 orders of magnitude smaller than `TOTUTEND`. Note a different colorbar is used for the residuals.

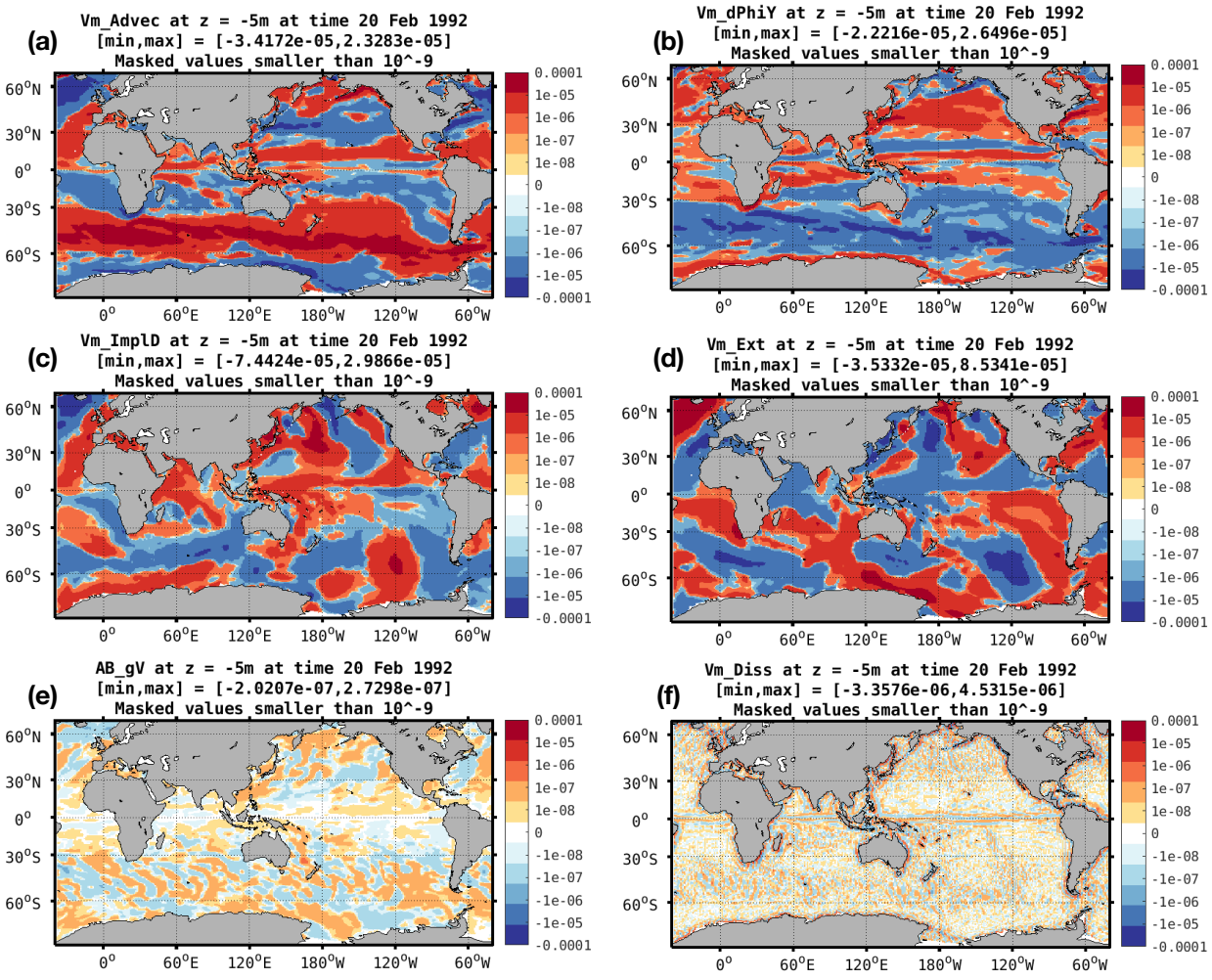

Figure 4: As for Fig. 1 but showing terms contributing to the ECCOv4r5 v-momentum tendency (ms^{-2}) at level 1 (depth = 5 m). Their sum (RHS of Eq. (1)) explains the Eulerian tendency of the meridional velocity at this depth level (see Fig. 6). In this configuration, only wind forcing contributes to `[Um_Ext,Vm_Ext]`, so these terms are zero for all depths but the surface level.

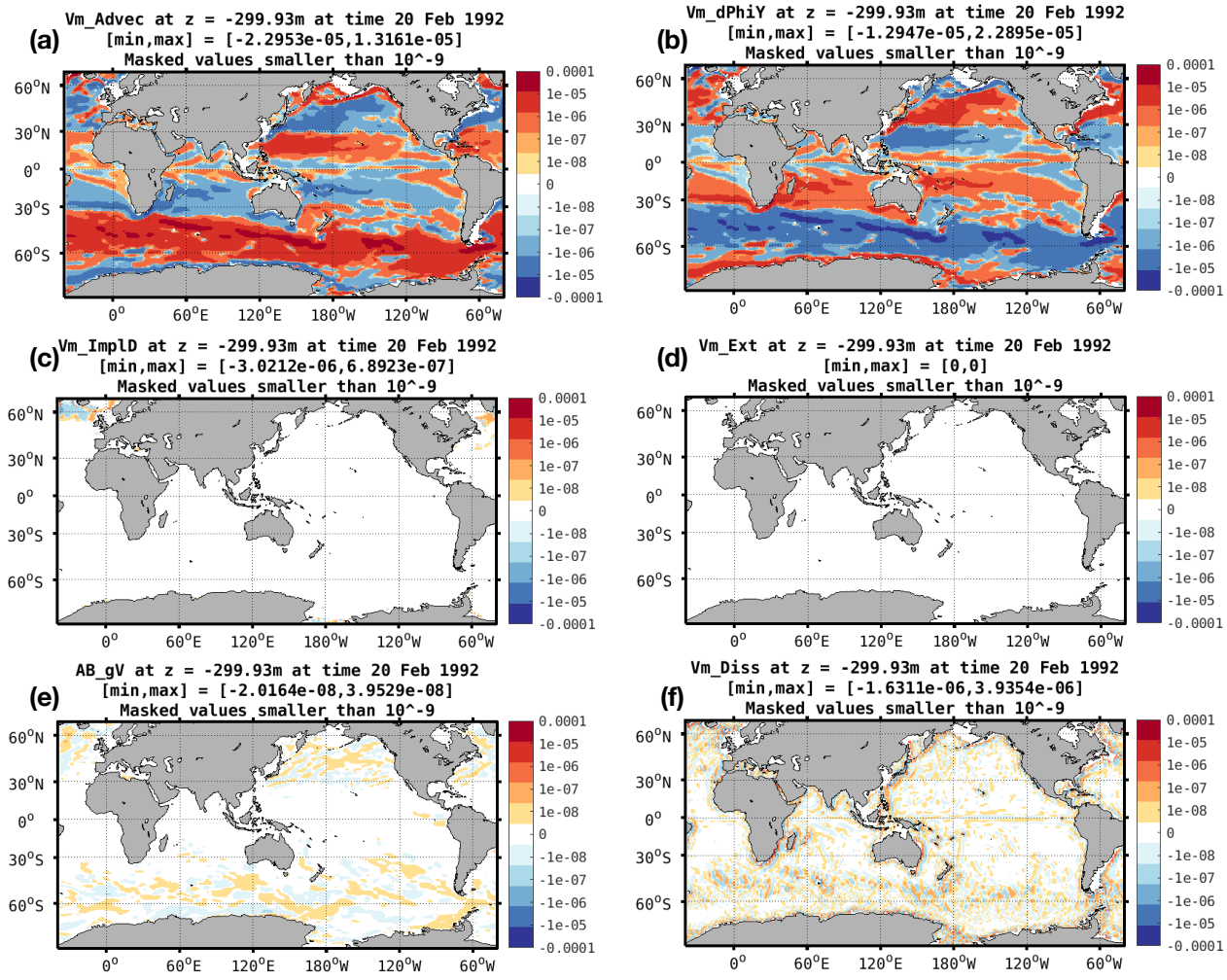

Figure 5: As for Fig. 2 but showing terms contributing to the ECCOv4r5 v-momentum tendency (ms^{-2}) at level 20 (depth = 299.9 m). Their sum (RHS of Eq. (1)) explains the Eulerian tendency of the meridional velocity at this depth level (see Fig. 6). In this configuration, only wind forcing contributes to `[Um_Ext,Vm_Ext]`, so these terms are zero for all depths but the surface level.

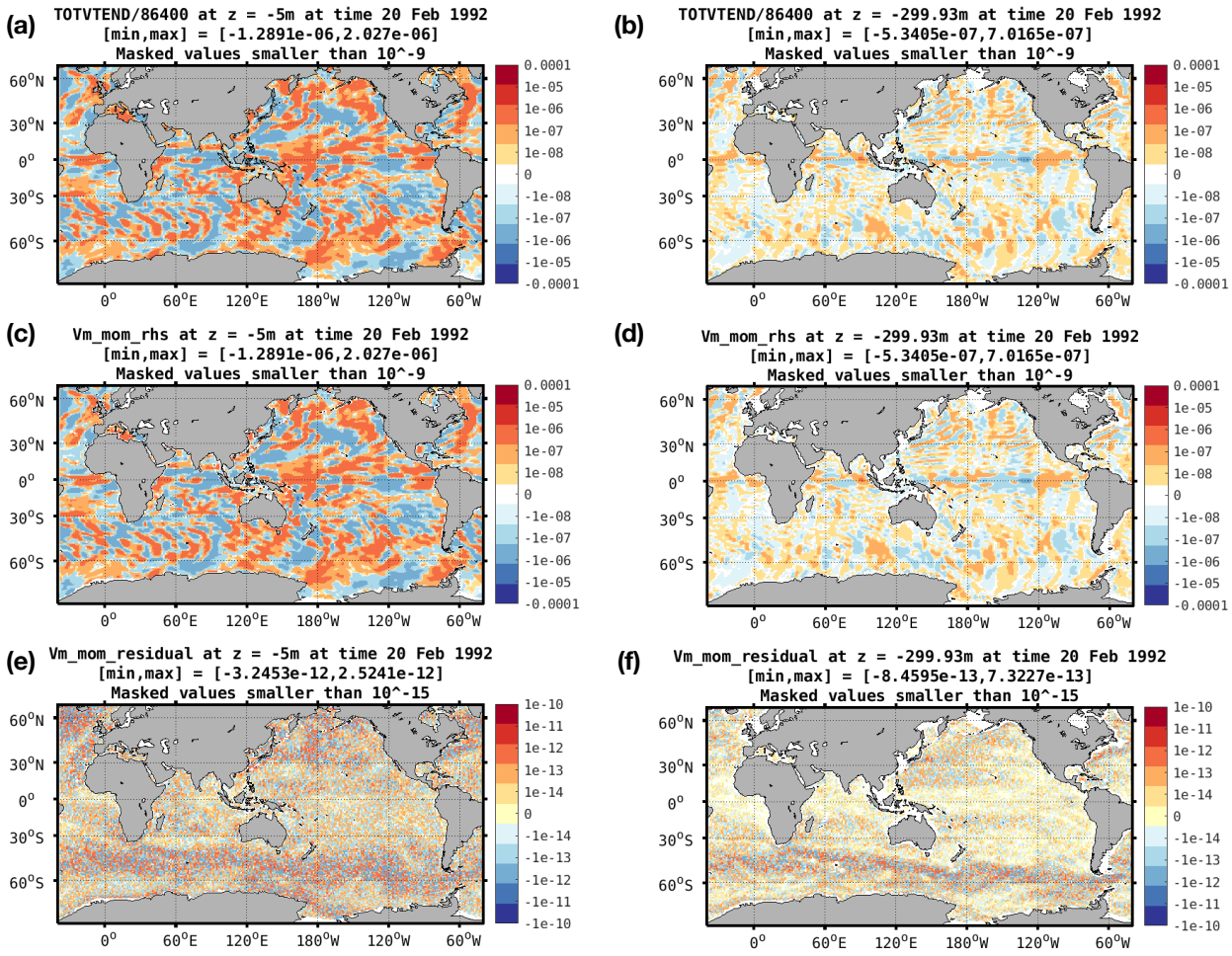

Figure 6: v-momentum (ms^{-2}) closure at (left) level 1 (depth = 5 m) and (right) level 20 (depth = 299.93 m). Closure is determined by subtracting (c,d) the sum of all RHS tendencies as shown in Figs. 1 & 2 from (a,b) `TOTVTEND` = the Eulerian tendency of the meridional velocity. The residual is shown in (e,f) to be 6 orders of magnitude smaller than `TOTVTEND`. Note a different colorbar is used for the residuals.

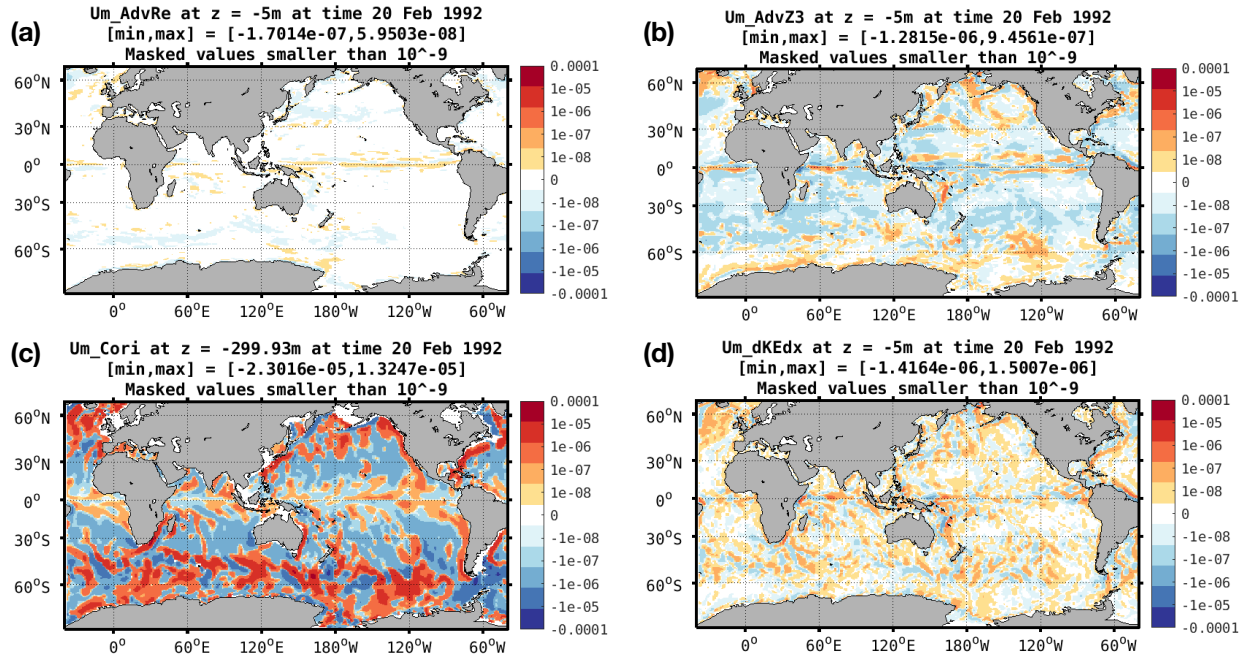

Figure 7: Decomposition of the u-momentum tendency (ms^{-2}) from advection (`Um_Advec`) shown in Fig. 1a (z = 5 m) into separate contributions (a) `Um_AdvRe`, (b) `Um_AdvZ3`, (c) `Um_Cori`, and (d) `Um_dKEdx`, as given in Eq. (3).

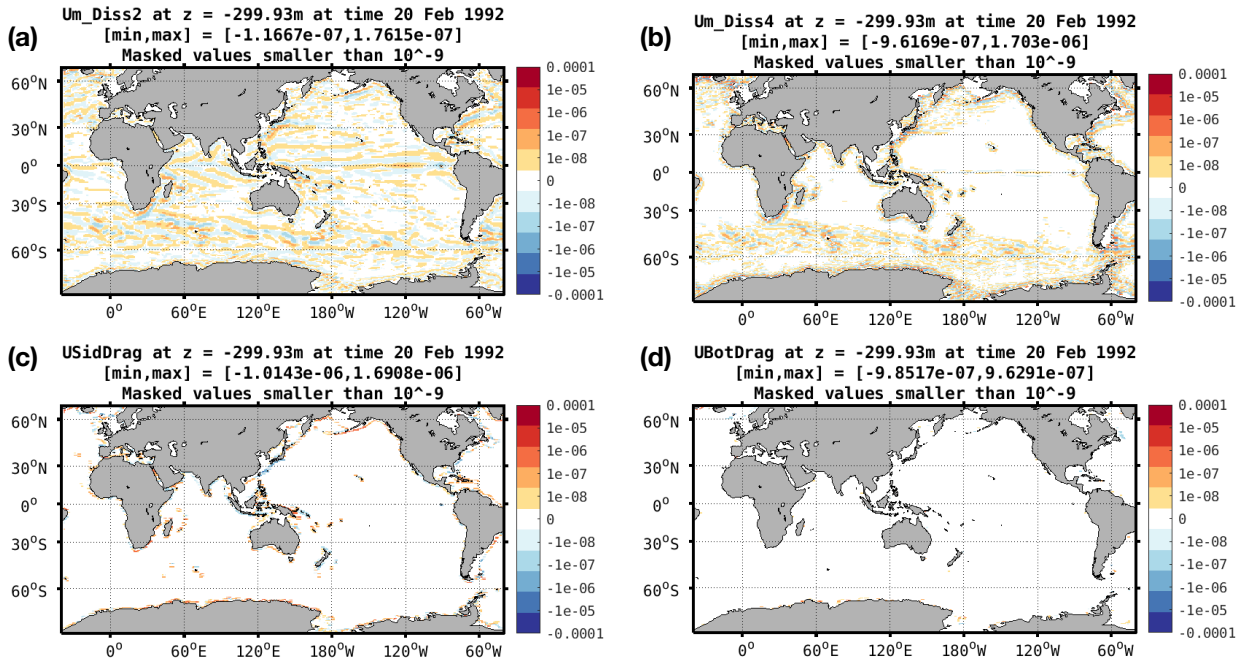

Figure 8: Decomposition of the u-momentum tendency (ms^{-2}) from explicit dissipation (`Um_Diss`) shown in Fig. 1f (z = 5 m) into separate contributions (a) `Um_Diss2`, (b) `Um_Diss4`, (c) `USidDrag`, and (d) `UBotDrag`, as given in Eq. (4). `UShIDrag` is missing here.

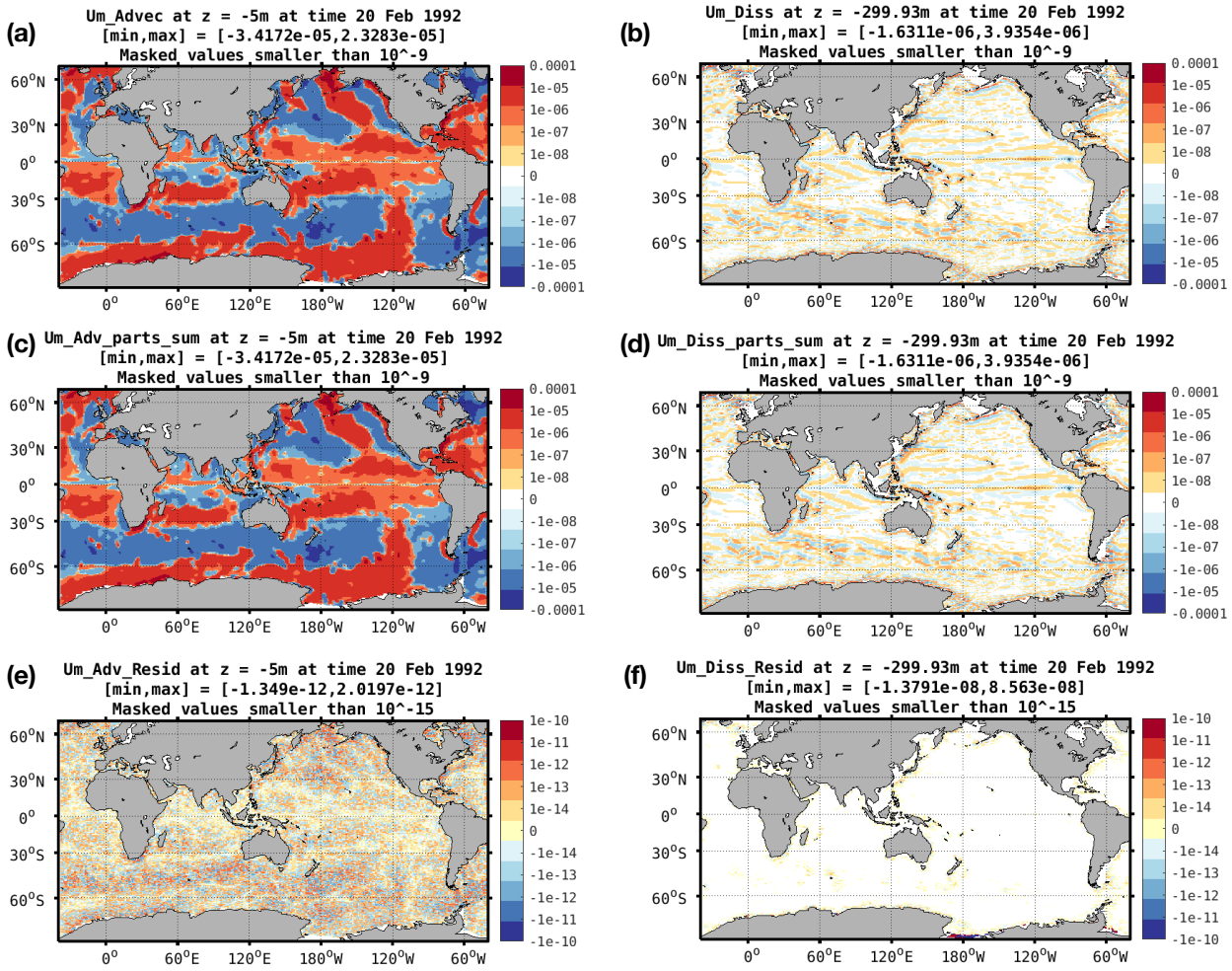

Figure 9: Confirmation the the tendencies (ms^{-2}) for (left) advection are accurately decomposed using Eqs. (3) & (4). The total tendencies from (a) advection and (b) explicit dissipation are equal to (c,d) the sum of their parts, shown in Figs. 7 & 8, respectively. The residuals for the advective decomposition is shown in (e) to be 7 orders of magnitude smaller than `Um_Advec`. The decomposition for `Um_Diss` does not hold because `UShIDrag` is missing here (see big residuals around Antarctica in (f)). Note a different colorbar is used for the residuals.