The energy balance interface between CABLE and the UM within ACCESS

Ian Harman 2-12-2016

Background

Some initial comparisons of CABLE simulations have indicated that there are differences in the behaviour of CABLE when run offline and when coupled to the UM that relate to the soil moisture dynamics. These differences have been assessed via a modelling comparison study where by output from a coupled simulation (CABLE-UM) has been used as meteorological forcing for a subsequent offline CABLE simulation. There are also concerns expressed from the CoE that CABLE-ACCESS results in excess evapotranspiration. This document potentially touches both issues.

The following is based on viewing (but not necessarily understanding) the primary references –the JULES technical documents (Essery et al. 2001, Best et al. 2009) and CABLE technical document (Kowalcyk et al. 2006) – alongside the publicly available code from the CABLE repository (R3651), the boundary-layer routines of the UM (ACCESS1.4, v8.5 and v10.x) and the JULES repository (v4.x). The differences between code versions necessitates that any modifications to the coupling will need to be adjusted to match the parent version of the UM (and JULES and CABLE).

Coupling basics

The fundamental premise behind the coupling of CABLE to the UM is that CABLE provides the surface fluxes of momentum, energy, water and carbon given mean atmospheric forcing as provided by the UM. The UM handles the evolution of the boundary-layer and CABLE the evolution of the land surface, including soil, vegetation, snow and ice-covered land. In this sense CABLE is directly replacing the UK community model JULES.

However considerations within the atmospheric code mean that implicit numerical schemes are used within the boundary layer scheme (i.e. rates of change are determined by conditions at the end of the time step and both the fluxes and state variables are solved for simultaneously). This imposes a similar implicit structure on JULES and its formulation. This implicit method is different to how CABLE operates in uncoupled mode.

Briefly the differences between ACCESS-coupled and offline are that (non-exclusively)

- CABLE is called multiple times for each time step when used within ACCESS.
- On the first (explicit) call the (prognostic) soil states are *not* updated and the (diagnostic) fluxes and surface states from CABLE are utilised primarily to determine the turbulent transfer properties within the UM's boundary-layer.
- On subsequent (implicit) calls updated meteorology is used to recalculate the energy balance as a surrogate for the implicit solution of JULES. The soil state[1] is updated and the revised surface turbulent fluxes are then used to evolve the boundary-layer.
- Correction terms to the energy and water balances are applied.

[1] If CASA is enabled the carbon pools are also updated only on the second calls. Again consideration needs to be given to whether and how the update applies with multiple calls to CABLE of the second type.

Updating the meteorological forcing

Other than the calls to the soil and CASA schemes (and the correction terms) the difference between the first and second calls to CABLE is the updated meteorological forcing. Specifically in the code at the start of cable_implicit_driver there is a section

```
:
CALL um2cable_rr( (LS_RAIN+CON_RAIN)*um1%TIMESTEP, met%precip)
CALL um2cable_rr( (LS_SNOW+CONV_SNOW)*um1%TIMESTEP, met%precip_sn)
CALL um2cable_rr( dtl_1, dtlc)
CALL um2cable_rr( dqw_1, dqwc)
met%precip   =  met%precip + met%precip_sn
met%tk = met%tk + dtlc
met%qv = met%qv + dqwc
:
:
CALL cbm(TIMESTEP, air, bgc, canopy, met, bal,  &
         rad, rough, soil, ssnow, sum_flux, veg)
```

i.e. this section converts required variables onto the CABLE grid – specifically precipitation, dtl_1 and dqw_1 – and then updates the meteorological forcing (met%precip, met%tk and met%qv) for CABLE accordingly.

As the concerns identified apply in conditions without rainfall, for the moment the precipitation variables are discounted as an issue. This note concentrates on the increment variables dtlc and dqwc, their basis and usage. Prior to that, however, note that the increments

1. Must result in physically realisable meteorological conditions.
2. Ideally should be small and result in meteorological conditions that lie in between the meteorology at the start of the time step and at the start of the next time step.

The origins of the increments

This section follows the working of Essery et al. (2001) and the bdy_impl routines of the UM.

The increments to the atmospheric temperature, δT_k, due to turbulent processes are given in the UM as

$$\delta T_k = \frac{g\,\Delta t}{\Delta p_k}\,[F_T(k+1) - F_T(k)]\; + \delta T_{k\mathrm{NT}} \qquad\qquad k = 1, \ldots, N \tag{1}$$

where *k* is the atmospheric model level with $k = 1$ the surface level, *N* is the number of layers in the boundary layer, *g* is gravitational acceleration, Δt the time step, $\Delta p_k < 0$ is the increment in pressure over the level and $F_T = H/c_p$ is the turbulent flux into the *kth* layer from below given by

$$F_T(k) = -RK_H(k)\left[\frac{T_k - T_{k-1}}{\Delta z_{k-\frac{1}{2}}} + \frac{g}{c_p}\right] \tag{2}$$

with RK_H the turbulent diffusivity on level *k*, T_k the temperature on level *k* and $\Delta z_{k-1/2}$ the vertical distance between the heights for levels *k* and *k-1*. (1) and (2) are the appropriate discretization of the 1st order closure diffusion equation

$$\frac{\partial \theta}{\partial t} = -\frac{\partial}{\partial z}\left[K_H \frac{\partial \theta}{\partial z}\right] + \left.\frac{\partial \theta}{\partial t}\right|_{\mathrm{NON-TURBULENT}}$$

when expressed in a form that uses temperature and a mixed *p-z* vertical coordinate system. Equivalent equation sets are set up for the turbulent increments to the humidity (mixing ratio *q*) and wind vector.

As stated earlier the solution for δT_k is numerically sensitive (and continually evolving in the UM for efficiency, accuracy and stability). In practice, as per the evolution of the soil state, the boundary-layer requires an implicit-in-time solution. Letting subscript (n) indicate known initial values, the substitution

$$T_k = T_k^{(n)} + \gamma_k \delta T_k \tag{3}$$

with γ_k a weighting factor, is used to derive a matrix equation

$$\begin{array}{lllll} B_{TN}\,\delta T_N & + & C_{TN}\delta T_{N-1} & & = \delta T_{Nex} \\ A_{Tk}\delta T_{k-1} & + & B_{Tk}\,\delta T_k & + \; C_{Tk}\delta T_{k-1} & = \delta T_{kex} \\ & & A_{T2}\delta T_2 & + \; B_{T1}\,\delta T_1 & = \delta T_{1ex} - (g\Delta t/\Delta p_1)F_T^{(n+1)} \end{array}$$

For levels *k=2,…,N* δT_{kex} is the known increment that would be obtained if the known, explicit, temperatures were used to determine the fluxes, i.e. $T_k = T_k^{(n)}$ in equations (2) and (1) **PLUS the any other, non-turbulent, increments**. These increments are physically realistic as they balance a rate of change with realisable flux divergence.

However for level *k=1* the implicit surface flux $F_T^{(n+1)}$ is retained in the equations and is unknown. δT_{1ex} at this stage is given by

$$\delta T_{1ex} = (g\Delta t/\Delta p_1)F_T^{(n)} + \text{non} - \text{turbulent terms} \tag{4}$$

δT_{1ex} is known at the start of the time step but is not a realistic increment as it is the result of an incomplete balance. The rate of change due to non-turbulent terms and due to the sensible heat flux from the layer above are incorporated into δT_{1ex}; the flux from the layer below (i.e. the surface) is not included.

Onward solution of the matrix equation results in

$$\begin{array}{lllll} \delta T_N & + & C'_{TN}\delta T_{N-1} & & = \delta T'_{Nex} \\ & & \delta T_k & + \; C'_{Tk}\delta T_{k-1} & = \delta T'_{kex} \\ & & & \delta T_1 & = \delta T'_{1ex} - \beta_T(g\Delta t/\Delta p_1)F_T^{(n+1)} \end{array}$$

with dashes indicating updated variables and β_T a function of the *A, B* and *C* variables. All $\delta T'_{kex}$ can be calculated but only for levels *k=2,…,N* are they plausible increments to the meteorology.

$\delta T'_{1ex}$, and its partner humidity variable, $\delta q'_{1ex}$, correspond to the coded variables dtl1 and dqw1 (in imp_solver and bdy_impl), dtl1_1 and dqw1_1 (in surf_couple_implicit and sf_impl) and dtl_1 and dqw_1 in cable_implicit_driver – i.e. these are the increments used.

Within JULES $\delta T'_{1ex}$ and $\delta q'_{1ex}$ are used (im_sf_pt2) as part of the solution for the multi-tile energy balance. The physical interpretation of these quantities is that they reflect the increment to the level 1 variables due to mixing of heat and water from the layers aloft. These quantities are important stepping-stone variables in the UM-JULES solution methodology. However as there is no influence of the surface forcing these variables are largely meaningless outside of that scheme. **The use of $T_1 + \delta T'_{1ex}$ in CABLE as feasible meteorological forcing for the second calls within the time step therefore risks obtaining an incorrect and inappropriate energy balance. However CABLE is unlikely to crash as a result.**

Implications

As an illustration of the impact of this mismatch in meaning – consider a daytime condition case with a unstable atmosphere ($\mathrm{d}\theta/\mathrm{d}z \leq 0$, $\mathrm{d}q/\mathrm{d}z \leq 0$, $H > 0$, $\lambda E > 0$) in near steady-state. In such situations $F_T(1) \approx F_T(2) > 0$ and $F_q(1) \approx F_q(2) > 0$ and we would expect the turbulence to result in zero change to both θ and q. However

$$\delta T'_{1ex} = \frac{g\,\Delta t}{\Delta p_1} F_T^{(n)}(2) < 0$$

and

$$\delta q'_{1ex} = \frac{g\,\Delta t}{\Delta p_1} F_q^{(n)}(2) < 0$$

So instead of forcing CABLE on the second call with meteorological conditions that are the same (or close) to that of the first call – CABLE would be forced with colder and drier conditions. To first order this second call would be expected to lead to larger values for the canopy and soil turbulent fluxes of sensible and latent heat than the meteorology, including radiation, and surface states would indicate. Note the impact is larger in magnitude when the surface fluxes themselves are large[2].

Rectification

If the above is correct then CABLE needs to perform an additional calculation to account for the influence of the surface forcing on the meteorological increments before applying them in the second (implicit) call(s). There are a number of potential alternate increments that could be applied: Two options have been tested further (option 1 has been discounted).

Option 2

One possible formulation[3] of the revised increments to the forcing are

$$\delta T_1^{(n)} = -\Delta t\; c_p^{-1} \frac{H_2^{(n)} - H_1^{(n)}}{\Delta z_1} + \Delta T_1^{(NT)}$$

$$\delta q_1^{(n)} = -\Delta t\; \frac{E_2^{(n)} - E_1^{(n)}}{\Delta z_1} + \Delta q_1^{(NT)}$$

These increments are those that would be obtained if the rate of change were established by conditions at the start of the time step and then held constant. The impacts of turbulent mixing is only partially captured.

[2] More precise estimates of the impact can be obtained if details of the model set up are known. However as an indication of how bad this could be – if $H_1^{(n)} = H_2^{(n)} = H_2^{(n+1)} = 100\mathrm{Wm}^{-2}$, the time step is 1800 seconds and the surface atmospheric layer is 20m deep, then simple energy balances for the atmospheric layer using coincident (n) fluxes gives that $\delta\theta = 0\mathrm{K}$ whereas $\delta T'_{1ex}$ could be as large as 7.5K!!

[3] These formulations need to be adjusted for spherical geometry.

Option 3

An alternate form for appropriate known increments are

$$\delta T_1 \quad = \quad \delta T'_{1ex} - \beta_T H^{(n)}/c_p$$

$$\delta q_1 \quad = \quad \delta q'_{1ex} - \beta_q E^{(n)}$$

These increments are those that would be obtained using the fully implicit numerical scheme from the boundary layer scheme but where the surface fluxes are held constant at the values taken at the start of the time step (i.e. from the first call to CABLE). Note in the above $H^{(n)}$ and $E^{(n)}$ are the grid-cell averages not the tile fluxes.

Either option could be implemented by either calculating the increments within the UM and then passing to CABLE, or calculating the increments at the start of cable_implicit_driver after passing additional information (β_T, β_q, vertical grid information, the grid-cell averaged $H^{(n)}$ and $E^{(n)}$) from the UM to CABLE. The net effect would not be an equivalent system to JULES (there would still be explicit-implicit, linearised-nonlinear and diagnostic-prognostic differences) but it would result in a physically realistic energy balance.

Following initial testing of the two options above option 2 appears not to be robust (or possibly implemented incorrectly) whereas option 3 passes initial short run qualitative and quantitative tests. The following section documents the implementation method within ACCESS1.4 and within v8.5 of the UM. The implementation methodology chosen has been to perform the calculations within the UM and pass the increments to CABLE rather than pass the necessary UM variables to CABLE. This way 'insures' that the need to calculate these increments is evident when changes are made to the UM boundary layer scheme and reduces the number of variables being passed through the various intermediary subroutines.

Option 3 for ACCESS1.4 (code and computation within UM)

Initial implementation and testing undertaken in ACCESS1.4 so that the different options could be tested – options 1 and 2 required code within the UM.

Aim: the revised increments for T and q are calculated in im_bl_pt1, passed through the UM to sf_impl and hence to CABLE.

In the UM: within imp_solver

- new additional workspace variables dtl_exp(:,:), dqw_exp(:,:) created as needed to hold the revised increments and pass around the code.
- l_cable, dtl_exp and dqw_exp included in the arguments to subroutine bdy_impl1
- l_cable, dtl_exp and dqw_exp included in the arguments to sf_impl

within bdy_impl1

- l_cable, dtl_exp, dqw_exp included in the calling declaration

- variables l_cable(logistic flag, intent IN), dtl_exp, dqw_exp created (intent OUT)
- l_cable, dtl_exp and dqw_exp included in the arguments to subroutine im_bl_pt1

within im_bl_pt1

- l_cable, dtl_exp, dqw_exp included in the calling declaration
- variables l_cable(logistic flag, intent IN) and dtl_exp, dqw_exp created (intent OUT)
- After section 3.3 (calculates the entries of the bottom row of the matrix equation) addition code included within a double for loop and condition statement set by l_cable to calculate the new increments

```
IF (l_cable) THEN
  DO J= 1,rows
    DO I=1, row_length
      DTL_EXP(I,J) = DTL(I,J,1)-CT_CTQ(I,J,1)*FTL(I,J,1)
      DQW_EXP(I,J) = DQW(I,J,1)-CT_CTQ(I,J,1)*FQW(I,J,1)
    ENDDO
  ENDDO
ENDIF
```

Note that the variable ct_ctq=β includes all the spherical geometry considerations and FTL=H/c_p.

within sf_impl

- dtl_exp, dqw_exp included in the calling declaration
- variables dtl_exp and dqw_exp created (intent IN)
- dtl_exp and dqw_exp included in the call to cable_implicit_driver

In CABLE: within cable_common

- a new switch (l_revised_coupling) included within the kbl_user_switches section with a default value of .false.

within cable_implicit_driver

- dtl_exp, dqw_exp included in the calling declaration
- variables dtl_exp and dqw_exp created (intent IN)
- At the point where the original mapping from UM variable dtl_1 to CABLE variable dtlc occurs included a conditional remapping as set by the value of l_revised_coupling

```
if (cable_user%l_revised_coupling) then
    ! revised forms
    CALL um2cable_rr( DQW_EXP, dqwc)
    CALL um2cable_rr( DTL_EXP, dtlc)
  else
    ! ORIGINAL forms
    CALL um2cable_rr( dtl_1, dtlc)
    CALL um2cable_rr( dqw_1, dqwc)
  endif
```

Option 3 for UMv8.5-CABLE (code and computation within CABLE)

Following discussion with the Met Office (November 2016) it became evident that for long term success coding should remain, where ever possible, on the CABLE-JULES side of the model. This necessitates changes to the implementation.

Aim: The coupling coefficient ct_ctq is passed to CABLE and then used to revise the increments for T and q within CABLE.

IN JULES: within sf_impl2

- variable rhokh_tile is multiplied by 0.0 in the call to im_sf_pt2
- ctctq1 included in the call to cable_control7
- cable%im%ctctq1 included in the call to cable_implicit_driver in both locations

within im_sf_pt2

- the USE cable_data_mod statement is not required and commented out.
- around line 434 the existing if (cable% um % l_cable) oncdition is commented out.
- around line 592, the calculation of dtstar_tile is placed within an if statement.

See later for further details as to why the changes to im_sf_pt2 are required.

IN CABLE: within cable_common (as for ACCESS1.4)

- a new switch (l_revised_coupling) included within the kbl_user_switches section with a default value of .false.

within cable_implicit_driver

- CTCTQ1 included in the calling declaration
- variable CTCTQ1 created (intent IN)
- working variables ctctq1_c, ftl_1c, fqw_1c, dimension(mp) created.
- working pointer variable CAPP included
- At the point where the original mapping from UM variable dtl_1 to CABLE variable dtlc occurs included a conditional remapping as set by the value of l_revised_coupling

```
if (cable_user%l_revised_coupling) then
    ! revised forms
    CAPP => PHYS%CAPP
    CALL um2cable_rr( CTCTQ1,ctctq1c)
    CALL um2cable_rr( FTL_1, ftl_1c)
    CALL um2cable_rr( FQW_1, fqw_1c)
    CALL um2cable_rr( dtl_1, dtlc)
    CALL um2cable_rr( dqw_1, dqwc)
    !NB FTL_1 is in W/m2 here so divide through by CAPP
    dtlc = dtlc - ctctq1c*ftl_1c/CAPP
    dqwc = dqwc - ctctq1c*fqw_1c
```

```
else
      ! ORIGINAL forms
      CALL um2cable_rr( dtl_1, dtlc)
      CALL um2cable_rr( dqw_1, dqwc)
endif
```

This code converts the um grid 2D variables ctctq1, FTL_1, FQW_1, dtl_1 and dqw_1 to CABLE 1D variables, then performs the Option 3 calculations.

in cable_data_module

- added ctctq1 to the impl_params 2D array pointer section
- included ctctq1 in the calling declaration for cable_control7
- variable ctctq1 created in the 2D array target section
- cable%im%ctctq1 pointed to the relevant variable

in cable_data_module_standalone (as for cable_data_module)

- added ctctq1 to the impl_params 2D array pointer section
- included ctctq1 in the calling declaration for cable_control7
- variable ctctq1 created in the 2D array target section
- cable%im%ctctq1 pointed to the relevant variable

Coastal grid cells (pt1)

Currently within im_sf_pt2 the cable specific code sets variable gamma_1=0 if CABLE is active. This is a simple, if non-transparent, means whereby the implicit stage increment to the surface fluxes that JULES routinely performs can be set to zero when CABLE operates.

However ACCESS wishes to retain the full JULES calculations for sea and sea-ice. gamma_1=0 negates the implicit stage increment for all surfaces within a particular grid cell. Hence for grid cells with mixed land/sea/sea-ice, the current implementation incorrectly removes the implicit stage increments. An alternate implementation instead sets the land tile variable rhokh_tile=0 within this part of the code only.

The precise details of how to implement this needs further discussion – the current, *deliberately bad**, implementation has been to multiply rhokh_tile by 0.0 within the call to im_sf_pt2.

The changes above are included in code set:

cable_common_INH1.3.F90,
cable_data_module_standalone_INH1.3.F90,
cable_data_module_UM_INH1.3.F90,
cable_implicit_driver_INH1.3.F90,
im_sf_pt2_jls_INH1.3.F90 and
sf_impl2_jls_INH1.3.F90
*deliberately bad as the best way to do this will require conversation with the Met Office in the long run.

Coastal grid cells (pt2)

The current implementation of cable_implicit_driver within sf_impl2 calls CABLE after im_sf_pt2 but, as stated above, CABLE tiles are not incremented within im_sf_pt2. This implementation uses CABLE explicit stage fluxes as part of the algorithm (for coastal grid cells). However the intent of im_sf_pt2 is to increment all the tile fluxes within a grid cell to their values at the end of the time step given the changes in all the other tile fluxes. Hence to retain the intent of im_sf_pt2 the CABLE tile fluxes should reflect their values at the end of the time step. In practice this requires that cable_implicit_driver is called prior to im_sf_pt2.

This 'simple' change to the code sequencing requires a substantial amount of code changes within sf_impl2. These are briefly

- Immediately prior to the call to im_sf_pt2 a new CABLE specific section is included. If l_correct is .false. this code
    1. converts FTL from $K/m^2/s$ to W/m^2 as required for CABLE
    2. initialises the melt_tile variable
    3. calls cable_control_7 and cable_implicit driver as before
    4. converts FTL from W/m^2 back to $K/m^2/s$.
- im_sf_pt2 is called with rhokh_tile=0 as described above
- In the next section, labelled 6.1, the initial FTL conversions are retained but all subsequent cable code is removed (including the initialisation of melt_tile).

This change in sequencing is included within sf_impl2_jls_INH1.3b.F90