

User manual for iterations between modules

François Lafolie, Nicolas Beudez, Nathalie Moitrier

2024-02-29



Table of contents

1	Introduction	4
2	Specify of a module handling iterations	5
3	Allow iterations between two modules	9

List of figures

1	Example of a model containing two coupled modules: <i>balance_batch_deg</i> and <i>first_order_degradation</i>	4
2	Tag a module input as <i>Iterate with</i> (<i>Inputs/Outputs</i> tab of <i>vsoil-modules</i> application)	6
3	Definition of parameters <i>epsiter</i> and <i>nitermax</i> for <i>first_order_degradation</i> module (<i>Parameters</i> tab of <i>vsoil-modules</i> application)	6
4	Definition of the maximum number of iterations: common value to all modules involved in a process of iterations between modules (<i>Initialization</i> tab of <i>vsoil-modules</i> and <i>vsoil-models</i> applications)	7
5	<i>Compute section</i> of module <i>first_order_degradation</i> : use of boolean variables <i>iteration_enabled</i> and <i>iteration_succeeded</i> (red frames), module parameters <i>epsiter</i> and <i>nitermax</i> (blue frame) and variable dedicated to the reduction of time step <i>reduce_dt</i> (green frame) (<i>Code editor</i> tab of <i>vsoil-modules</i> application) . . .	8
6	List of modules to be ordered	9
7	Unordered list of modules	10
8	Ordered list of modules	10
9	Changing the order of the list of modules	10
10	Warning appearing in the ordered list of modules	11
11	Selection of the input to be used for the process of iterations between modules .	11

1 Introduction

Two variables A and B calculated by two different modules MA and MB are said **coupled** when variable A is calculated using variable B and variable B is calculated using variable A. In this situation, one has the choice between **linearization** (for example module MA is executed first and variable A is then used to calculate variable B in module MB) and **iterations** between the two modules.

A typical situation is presented in the model of figure 1. This model is the official model *example_for_modules_iterations* available in *vsoil-models* application. It is composed of three modules:

- *steady_water_flow*: a module providing a steady water content (implements process *water flow and balance*);
- *balance_batch_deg*: a module carrying out the balance for solute species subjected to degradation (implements process *transport in liquid phase*);
- *first_order_degradation*: a module calculating the degradation rate using the concentrations of solute species (implements process *molecules degradation*).

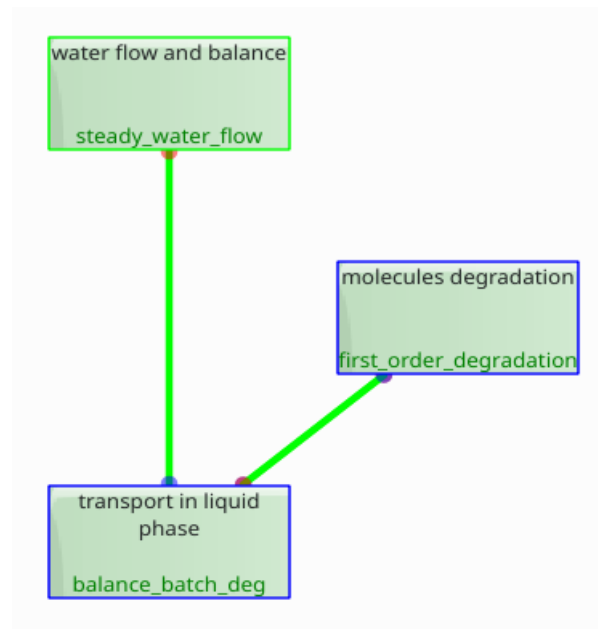


Figure 1: Example of a model containing two coupled modules: *balance_batch_deg* and *first_order_degradation*

The inputs/outputs of the modules composing this model are:

- module *steady_water_flow*:
 - inputs: none;
 - outputs: *soil water volumetric content* (used by module *balance_batch_deg*);
- module *balance_batch_deg*:
 - inputs: *soil solution solutes degradation rate* (provided by module *first_order_degradation*) and *soil water volumetric content* (provided by module *steady_water_flow*);
 - outputs: *soil solution solutes mass concentration* (used by module *first_order_degradation*);
- module *first_order_degradation*:

- inputs: *soil solution solutes mass concentration* (provided by module *balance_batch_deg*);
- outputs: *soil solution solutes degradation rate* (used by module *balance_batch_deg*).

It can be seen that modules *balance_batch_deg* and *first_order_degradation* are coupled: indeed variable *soil solution solutes mass concentration* is calculated using variable *soil solution solutes degradation rate* and variable *soil solution solutes degradation rate* is calculated using variable *soil solution solutes mass concentration*. Very often the problem is linearized: the degradation rate (variable *soil solution solutes degradation rate*) is calculated first (it means that module *first_order_degradation* is called first) and is used by the module carrying out the balance (module *balance_batch_deg*). This means that the concentration (*soil solution solutes mass concentration*) at time t is used to calculate the sink term applied at time $t + dt$ in module *balance_batch_deg*. A more precise solution is obtained when iterations are set up. This situation appears quite frequently.

2 Specificity of a module handling iterations

To set up iterations between two modules, you must first select the module able to handle iterations. **A module able to handle iterations is detected by the platform because it has at least one input variable that is tagged as: *Iterate with*.** This choice is done when developing the module.

Edit your module in *vsoil-modules* application. In *Inputs/Outputs* tab, some check boxes are displayed in the *Iterate with* column of the *Module inputs* part of the window (see figure 2). The developer of the module can use these check boxes to indicate that the module can be involved in an iterative loop using the checked input variables. In this figure, the input variable *soil solution solutes mass concentration* of module *first_order_degradation* is tagged as *Iterate with*, so that module *first_order_degradation* will have the ability to handle a process of iterations with the module that produces this variable, that is module *balance_batch_deg* (see figure 1 and the list of inputs/outputs of modules *first_order_degradation* and *balance_batch_deg* presented above).

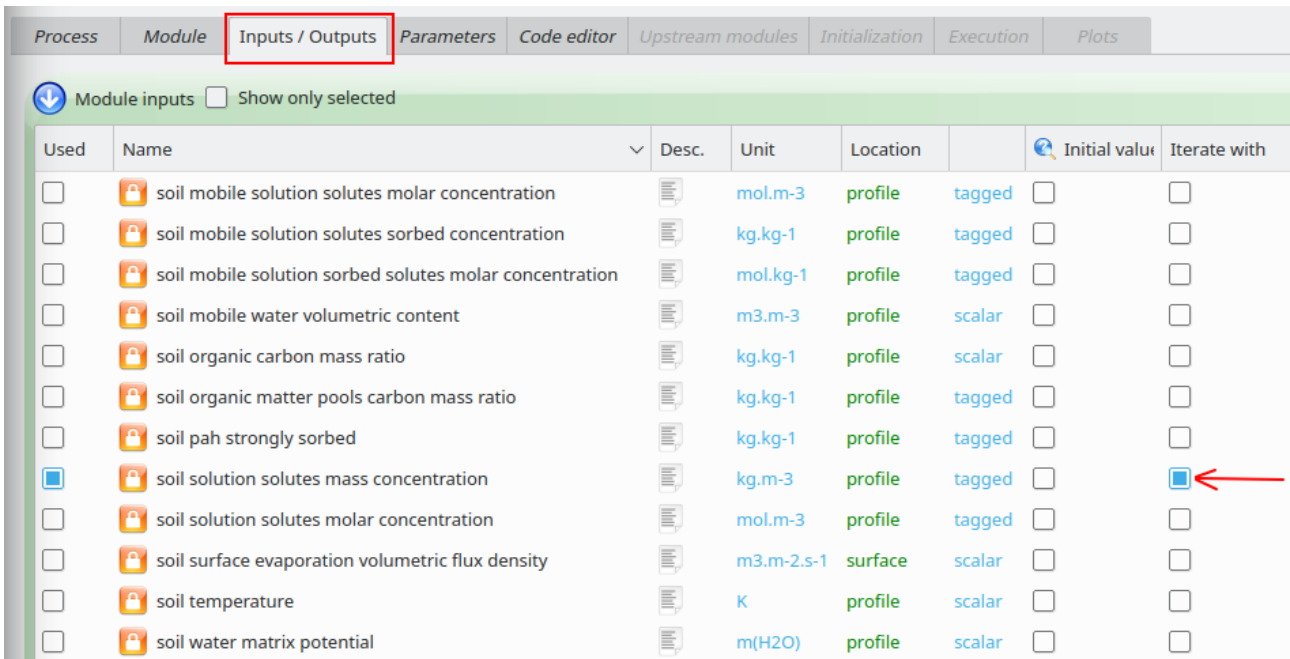


Figure 2: Tag a module input as *Iterate with* (*Inputs/Outputs* tab of *vsOil-modules* application)

Remark: instead of tagging *soil solution solutes mass concentration* as *Iterate with*, we could have tagged variable *soil solution solutes degradation rate* that is an input of module *balance_batch_deg*. In this case, module *balance_batch_deg* would have been able to handle the process of iterations with module *first_order_degradation* that produces this variable (see figure 1 and the list of inputs/outputs of modules *first_order_degradation* and *balance_batch_deg* presented above).

It is the responsibility of the module developer to define and code in the compute section of the module (see below), the algorithm controlling the convergence of the iterations. So, the module should have a parameter defining the convergence criterion (parameter *epsiter* of module *first_order_degradation*, see figure 3).

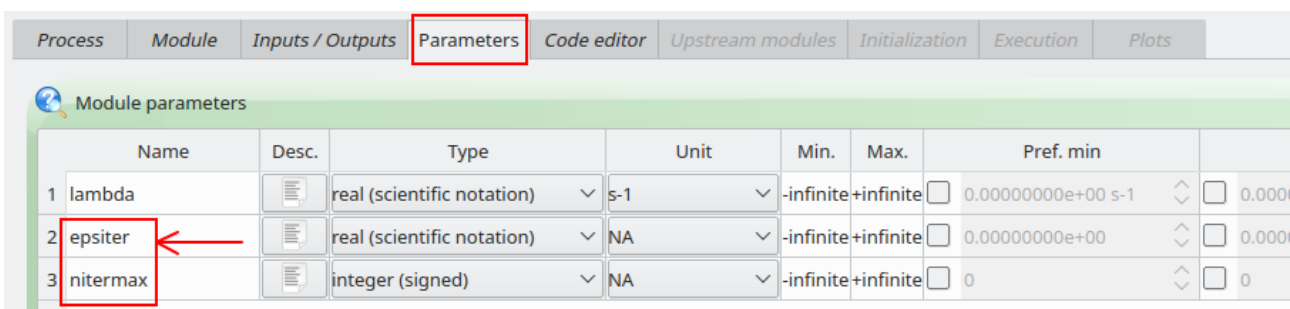


Figure 3: Definition of parameters *epsiter* and *nitermax* for *first_order_degradation* module (*Parameters* tab of *vsOil-modules* application)

Although a maximum number of iterations is available in *Global* sub-tab of *Initialization* tab in *vsOil-models* application (see figure 4, let's name it *iterateWithMaxIterations* for the rest of this documentation), this number being common to all modules composing a model, it is strongly recommended to have a module parameter defining a maximum number of iterations (parameter *nitermax* of module *first_order_degradation*, see figure 3): this offers the possibility

to build a model composed of modules having their own (and potentially different) maximum number of iterations.

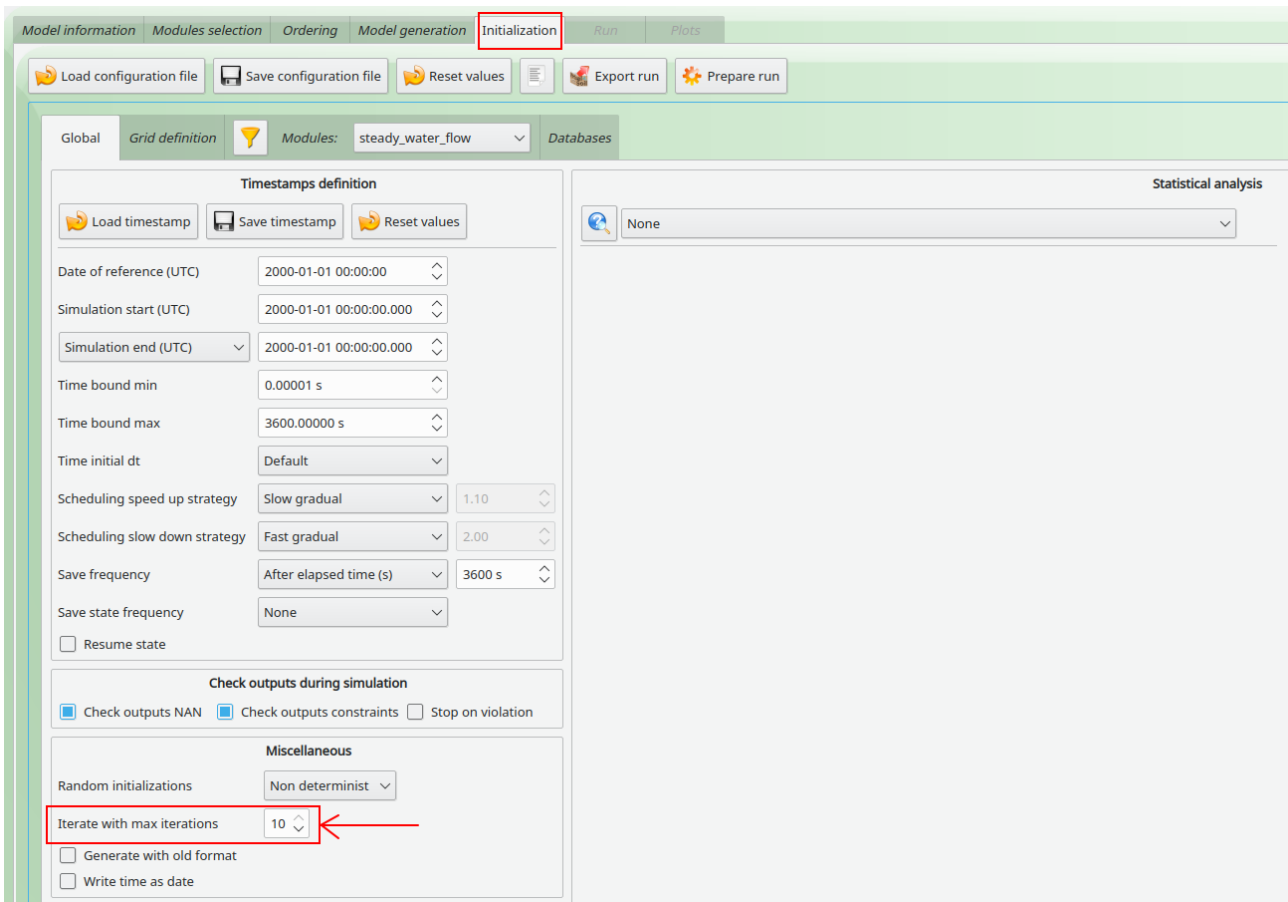


Figure 4: Definition of the maximum number of iterations: common value to all modules involved in a process of iterations between modules (*Initialization* tab of *vsoil-modules* and *vsoil-models* applications)

A module handling iterations is supposed to do the following tasks: (1) verify the convergence of the iterative process, (2) control the progress of the iterations in relation to the maximum number of iterations and (3) inform the coupler whether the iterations succeeded or not.

Note: the step (2) has to be done only in the case the module has its own maximum number of iterations *nitermax*.

When at least one input variable is tagged as *Iterate with* (see figure 2), the two following boolean variables are made available in the *Compute* section of the *Code editor* tab of the concerned module: *iteration_enabled* and *iteration_succeeded* (see the red frames in figure 5). The first (*iteration_enabled*) informs the *Compute* section that the iterations are possible. Note that **iterations are a possibility and not an obligation** for the developer of the module. The second variable (*iteration_succeeded*) must be used by the developer to inform the coupler of the progress of the iterations. By default, *iteration_succeeded* is set to **true**. This means that the coupler will consider that the iterations are successful. In this case the coupler does not call again the module providing the coupling variable (here module *balance_batch_deg* that provides variable *soil solution solutes mass concentration*) and continues the execution of the model for the current time increment. So if the developer wants to iterate, he must initially

set *iteration_succeeded* to **false** until he considers that the iterations have succeeded. When *iteration_succeeded* is false, this indicates the coupler to call again the module providing the coupling variable. Note that the outputs of the module managing the iterations are calculated each time its compute section is called.

```

1
2 do j = 1,nbdeg
3   deg(1:vsoil_grid_size,j) = -lambda(j)*csol(1:vsoil_grid_size,j)
4 end do
5
6 if(iteration_enabled.eqv..true.) then
7   iteration_succeeded = .false.
8   if(iter.eq.0) then
9     iter = iter + 1
10    prec_deg(1:vsoil_grid_size,1:nbdeg) = deg(1:vsoil_grid_size,1:nbdeg)
11  else
12    iter = iter + 1
13    do j=1,nbdeg
14      dif(1:vsoil_grid_size,j) = deg(1:vsoil_grid_size,j)-prec_deg(1:vsoil_grid_size,j)
15      call absarray(dif(1:vsoil_grid_size,j),abs_dif,vsoil_grid_size)
16      delta(j) = maxval(abs_dif,1)
17    end do
18    deltamax = maxval(delta,1)
19
20    if(deltamax.le.epsiter) then
21      iteration_succeeded = .true.
22      iter = 0
23    else
24      if(iter.gt.nitermax) then
25        iteration_succeeded = .false.
26        reduce_dt = .true.
27        iter = 0
28        return
29      else
30        prec_deg(1:vsoil_grid_size,1:nbdeg) = deg(1:vsoil_grid_size,1:nbdeg)
31      end if
32    end if
33  end if
34 end if
35

```

Figure 5: *Compute section* of module *first_order_degradation*: use of boolean variables *iteration_enabled* and *iteration_succeeded* (red frames), module parameters *epsiter* and *nitermax* (blue frame) and variable dedicated to the reduction of time step *reduce_dt* (green frame) (*Code editor* tab of *vssoil-modules* application)


If convergence is reached (see first blue frame in figure 5), *iteration_succeeded* is set to **true**. On the contrary, if convergence is not reached within the maximum number of iterations allowed in the module (see second blue frame in figure 5), *iteration_succeeded* is set to **false** and the *reduce_dt* variable must be set to **true** (see green frame in figure 5). At this point, the coupler behaves as usually when a module returns the *reduce_dt* variable set at **true**.

Notes:

- If the value of *nitermax* (maximum number of iterations associated with module *first_order_degradation*; see figure 3 and second blue frame in figure 5) is greater than or equal to the value of *iterateWithMaxIterations* (maximum number of iterations common to all modules involved in a process of iterations between modules, see figure 4), then *iterateWithMaxIterations* will be the maximum number of possible iterations between modules *first_order_degradation* and *balance_batch_deg*; it means in this case that the value of *nitermax* is ignored.
- The maximum value that can be entered for *iterateWithMaxIterations* is by default equal to 10 (see figure 4). If you need to use more iterations, contact the VSoil support using the following email: vsoil-support@inrae.fr.

3 Allow iterations between two modules

In order to allow iterations between two modules, run the *vsoil-models* application and then:

1. Open/create a model containing a module handling iterations. In our example, we open the model *example_for_modules_iterations* presented in figure 1.
2. In *Ordering* tab, the list of modules to be manually ordered is displayed (see figure 6). In upper part of this tab, the  icon appears in front of variable *soil solution solutes mass concentration* because this variable, which is produced by module *balance_batch_deg* and is an input of module *first_order_degradation*, has been previously tagged as *Iterate with* in *vsoil-models* application (see figure 2).

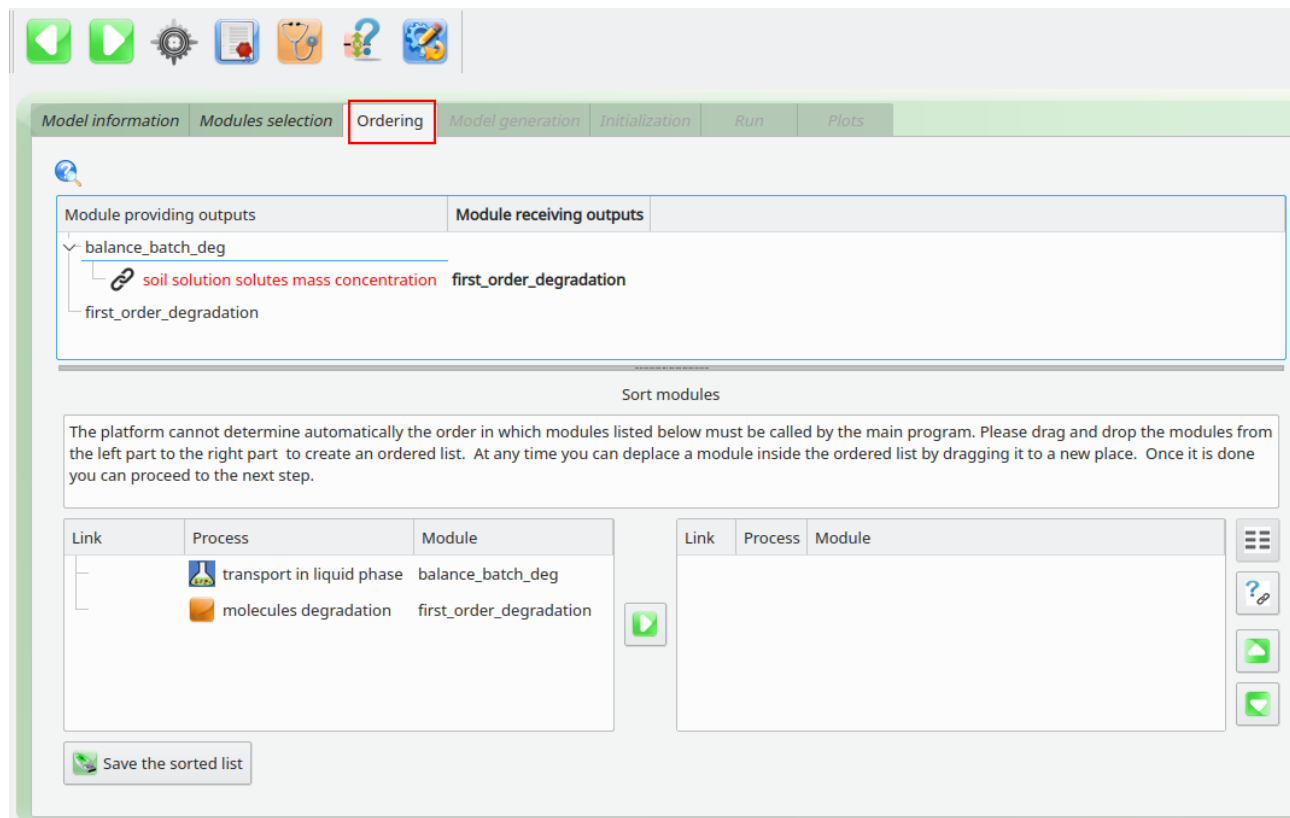



Figure 6: List of modules to be ordered

- In the bottom part of *Ordering* tab (*Sort modules* panel), drag and drop the modules from the left part to the right part of the panel (or use the  button) to create the desired ordered list (see figures 7 and 8):

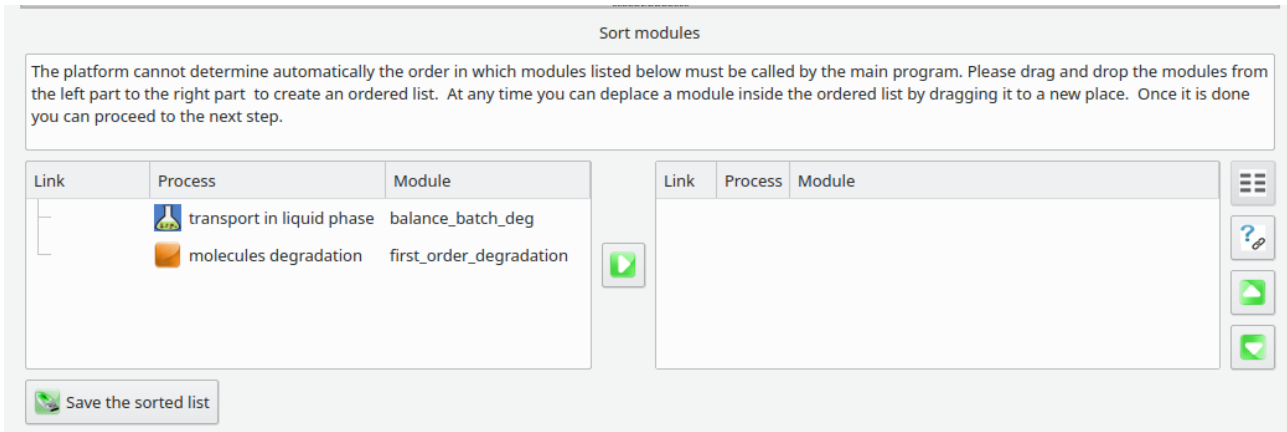


Figure 7: Unordered list of modules

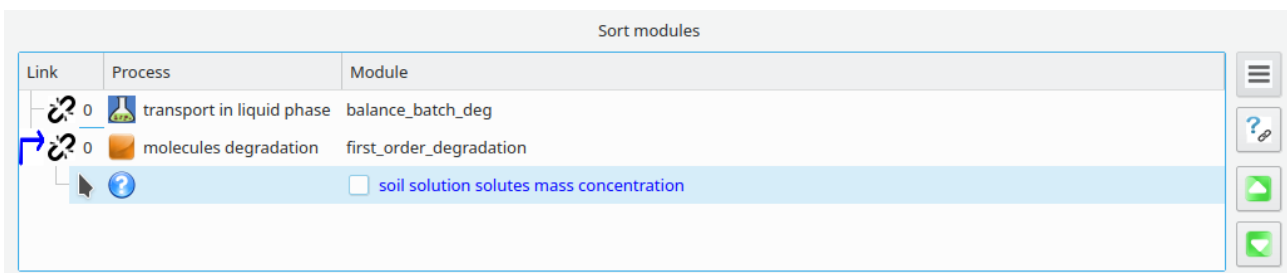




Figure 8: Ordered list of modules

If you want to modify the order of the list of modules, you can select a module and then drag and drop it to the desired position or use the  and  buttons (see figure 9).

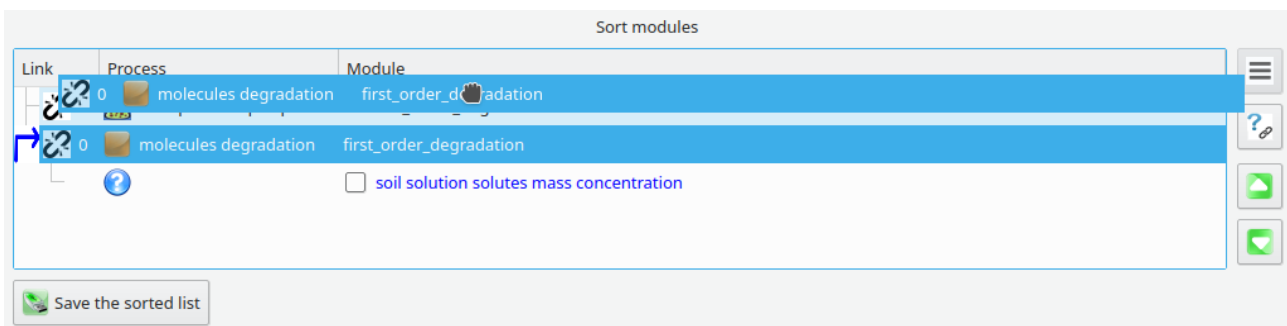



Figure 9: Changing the order of the list of modules

Note: the  icon appears in front of the variable tagged as *Iterate with* if the module receiving these variable is executed before the module providing it (in figure 10, module *first_order_degradation* is executed before module *balance_batch_deg*). This means that module *first_order_degradation* will start its iterations using the outputs of module *balance_batch_deg* calculated at the previous time step.

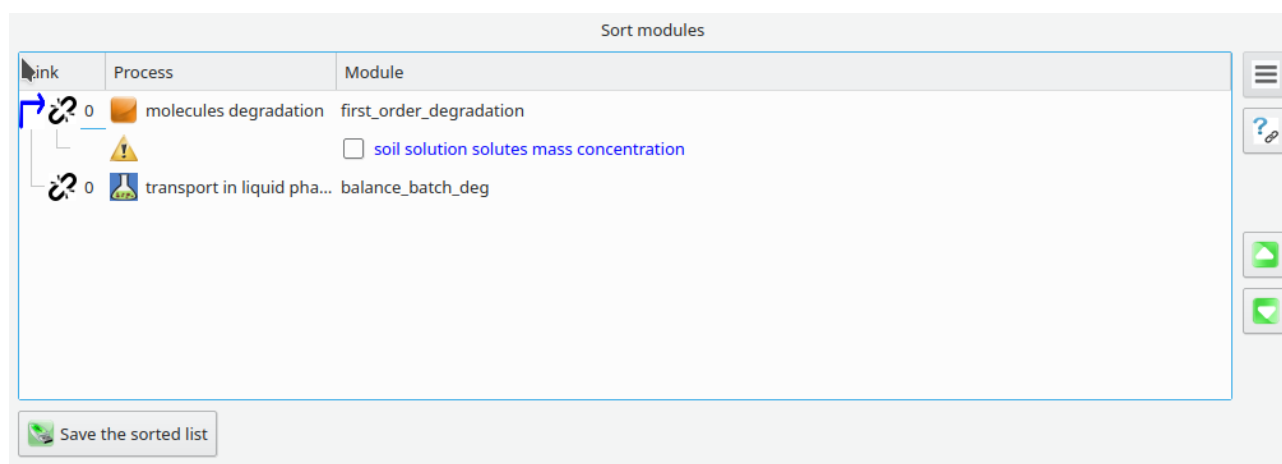








Figure 10: Warning appearing in the ordered list of modules

- Once the modules have been ordered (see figure 8), a check box then appears in front of variable *soil solution solutes mass concentration* because this variable is tagged as *Iterate with* (it is marked with the  icon in upper part of *Ordering* tab, see figure 6). The  icon specifies which module receives the variable tagged as *Iterate with*: here it is module *first_order_degradation*. The  icon (open loop) indicates that iterations between two modules are available but not active; the number associated to this icon (0 here, ) identifies the couple of modules that can iterate (case of several couples of modules that can iterate in a model).
- To activate the iterations between the two modules, select the variable tagged as *Iterate with* (see figure 11). Once the variable is selected, the symbolic representation changes: the  icons (open loop) are replaced by the  icons (closed loop) which indicates that the process of iterations between the two concerned modules is active.

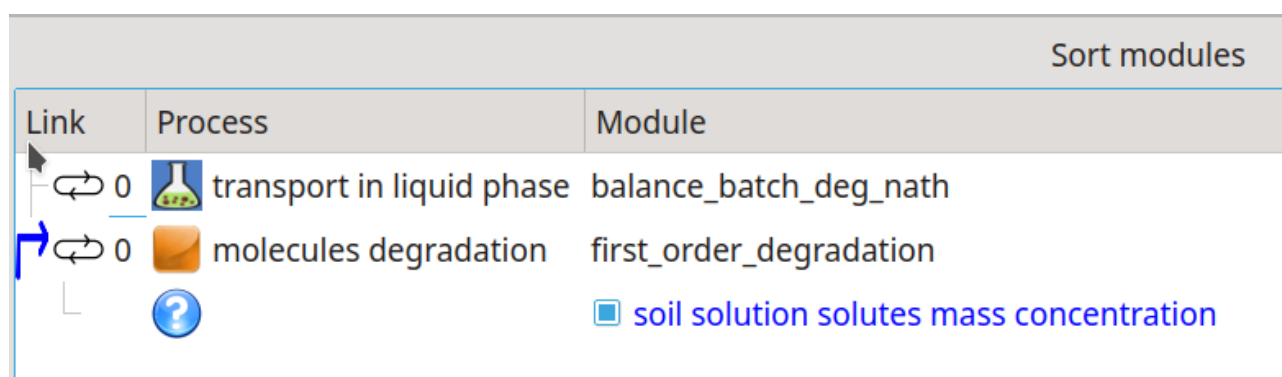





Figure 11: Selection of the input to be used for the process of iterations between modules

Notes:

- When the main program iterates between two modules, all the modules that are in between in the execution flow are also evaluated.
- There is no constraint on the order in which the modules are called by the coupler. For example, if MA, the module providing the inputs, is called after module MB, then MB

will start its iterations using the outputs of MA at the previous time step (it is the reason of the presence of  icon on figure 10, see explanation above). If MA is called before MB, then MB will start its iterations with the current outputs of MA.

- The number placed after  and  icons is a link identifier between modules (0 in figure 8). Indeed, different iterations between different modules can be activated. Thus, this identifier allows to know which modules are involved in a given process of iterations between modules. The only limitation is that processes of iterations between modules cannot overlap. It means, if we consider iterations between modules MA and MB on the one hand, and modules MC and MD on the other hand, that the two following execution flows are allowed:



and









but the following execution flow is forbidden:



- In *Sort modules* part of *Ordering* tab in *vsOil-models* application (see figures 7 and 8), it is possible that you cannot move a module in a desired position because the platform detects automatically overlapping iterations loops and forbids them (see the forbidden execution flow in previous item).

Summary of icons appearing in a process of iterations between modules:

- : indicates that iterations between two modules are available but not activated;
- : indicates that iterations between two modules are activated;
-  and  : 0 is the link identifier between two modules (case of several processes of iterations between different couples of modules);
- : specifies the module that receives the variable tagged as *Iterate with*;
- : appears in front of a variable tagged as *Iterate with* if the module receiving this variable (MB) is executed before the module providing it (MA). In this case, MB will start its iterations using the outputs of MA calculated at the previous time step.