Créer des modules avec Vsoil

Tutoriel 4 : Matière organique premier ordre

Éric Aivayan, Nicolas Moitrier, Cédric Nouguier

2022-05-16

Table des matières

Table des figures

Listings

Cet ensemble de tutoriels a pour objectif de montrer par des exemples concrets la création de modules. Les différents modules présentés permettent d'aborder les possibles difficultés de réalisation de manière graduelle. Ainsi nous allons partir d'un module indépendant du temps et ne nécessitant pas de données représentées sous forme de tableaux pour arriver à un module dépendant du temps avec tous les types de données possibles.

Pour pouvoir aborder ces tutoriels, vous devez savoir :

— créer un processus,
— modifier un processus,
— créer une entrée/sortie,
— modifier une entrée/sortie.

Le codage des modules étant réalisé en Fortran, vous devez avoir quelques notions de programmation dans ce langage.

Sur toutes les captures d'écran, les textes encadrés en rouge sont présents à titre indicatif. Ils ne s'affichent pas lors de l'utilisation de la plateforme.

1 Présentation scientifique.

Dans ce tutoriel, nous allons simuler la variation de matière organique dans le sol en utilisant une loi différentielle de variation du premier ordre. Nous vous invitons à vous reporter au tutoriel précédent pour la présentation de la résolution numérique.

Pour une source de matière organique donnée nous avons la relation :

$$pool(t + dt, p) = \frac{pool(t, p)}{1 + dt \times f(p) \times VantHoff \times Andren} \tag{1}$$

où

— t est le temps,
— p est la profondeur,
— $VantHoff$ est un facteur d'activité dépendant de la température,
— $Andren$ est le potentiel matriciel dépendant de la profondeur.

Le facteur d'activité est de la forme $e^{Q(T)(T-T_{ref})}$ comme représenté à la figure 1.

Le potentiel matriciel suit la loi représentée à la figure 2.

2 Ce que vous allez découvrir.

Dans ce tutoriel vous allez assembler les notions vues dans les autres tutoriels et utiliser la section *validate* de l'éditeur de code pour calculer une grandeur cumulée sur le temps.

En effet, du fait du mécanisme de détermination du pas de temps approprié par la plateforme, le simple cumul d'une grandeur peut conduire à des incohérences.

Prenons un exemple. Supposons que nous disposions d'un modèle contenant les modules A et B tels que le module B soit exécuté après le module A. Durant l'exécution, le module A vérifie que le pas de temps est correct pour ses calculs avant le module B et, surtout, les effectue avant. Si, dans une boucle de temps, le module B vient à refuser le pas de temps, l'ensemble des modules en amonts recommencent leurs calculs pour le nouveau pas de temps. Si une variable accumule les valeurs d'une sortie, elle aura reçu deux valeurs pour le même pas de temps.

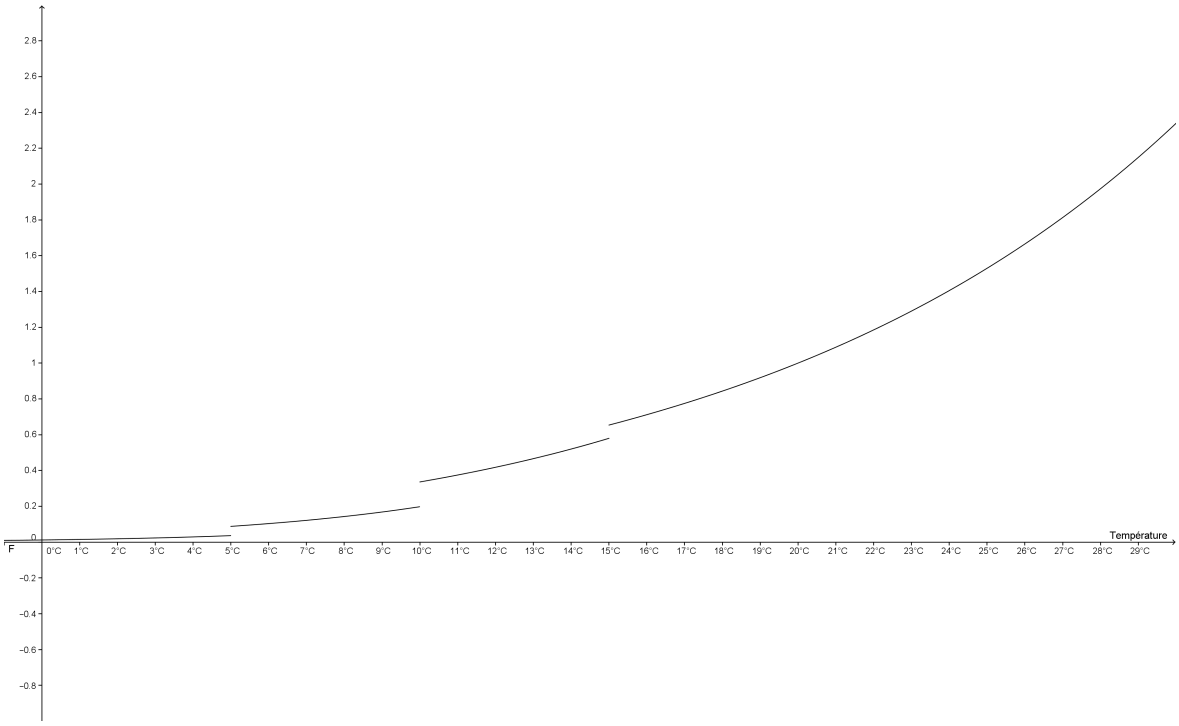

FIGURE 1 –
Loi de VantHoff avec

$$Q(T) = \begin{cases} 0,221 \text{ pour } T < 5 \\ 0,162 \text{ pour } 5 \leq T < 10 \\ 0,109 \text{ pour } 10 \leq T < 15 \\ 0,085 \text{ pour } 15 \leq T \end{cases}$$

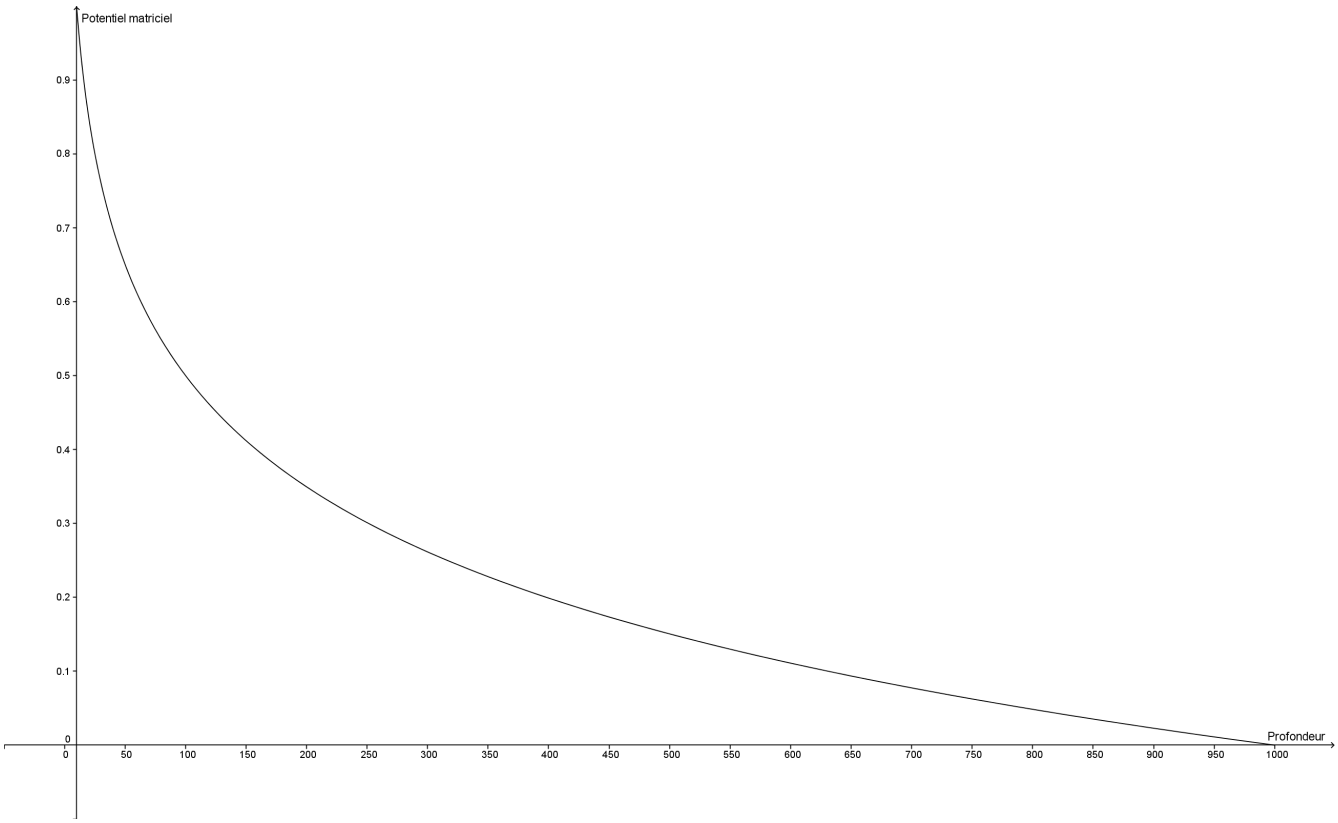

FIGURE 2 – Potentiel matriciel

3 En avant !

3.1 La définition générale du module.

Nous indiquons ici les points indispensables pour la construction du module. Nous vous renvoyons aux tutoriels précédent pour vous remémorer l'ordre de saisie et l'emplacement de ces informations. Les renommages se feront lors de l'édition du code.

Le module *OM_firstorder* est une simulation du processus *organic matter dynamics* en *fortran*. C'est un module de dimension spatiale 1D.

Les entrées choisies sont *soil profile matrix potential* qui sera renommée *psi* et *soil temperature* qui sera simplement renommée *temp*.

Les sorties choisies sont *organic matter pools*,*organicmatter solutes production* et *organicmatter solutes consumption*. Elles seront renommées respectivement *pools*, *sink* et *source*. La première sortie devra être initialisée (*initial value required*). Pour les deux autres, il faudra sélectionner les colonnes *NH4+* et *NO3-*.

Les paramètres utilisés sont ceux présentés dans le tableau suivant. Seul le paramètre *tref* sera renommé en *tempref*.

La dépendance au temps se fait seulement par le pas de temps *dt*.

Nom	Description	Type	Unit	Pref. min	Pref. max	default	Vector(array)	LD
decrate	Time constant for decomposition of the organic matter pools	Real(Sc)	NA	0,00000000e+00		1,00000000e-03	10	☑
pfmin	minimum pf for decomposition of organic matter	Real(Sc)	NA	1,00000000e+00		1,00000000e+00		
pfmax	maximum pf for decomposition of organic matter	Real(Sc)	NA	3,00000000e+00		4,00000000e+00		
tref	reference temperature for biological processes	Real(Sc)	NA			2,88000000e+02		
qvanhoff	coefficients for vanhoff law for temperature dependence of biological processes	Real(Sc)	NA			1,15000000e-01	9	

Étape 1 : Association processus-module

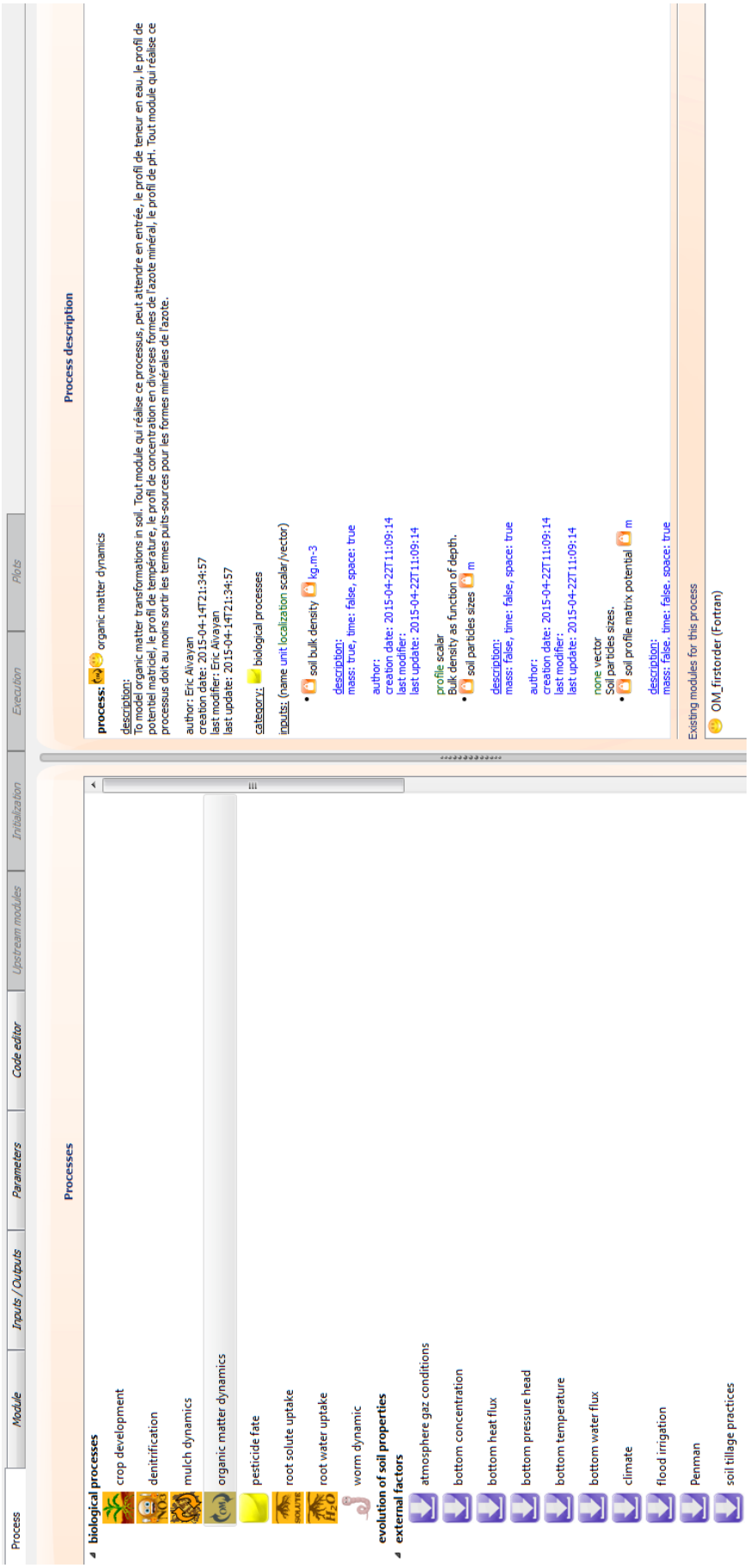

FIGURE 3 – Capture d'écran du composant de plateforme « vsoil-modules ». Il est représenté l'onglet des processus. Pour de plus amples renseignement se reporter au tutoriel *root water uptake*

Étape 2 : Informations du module

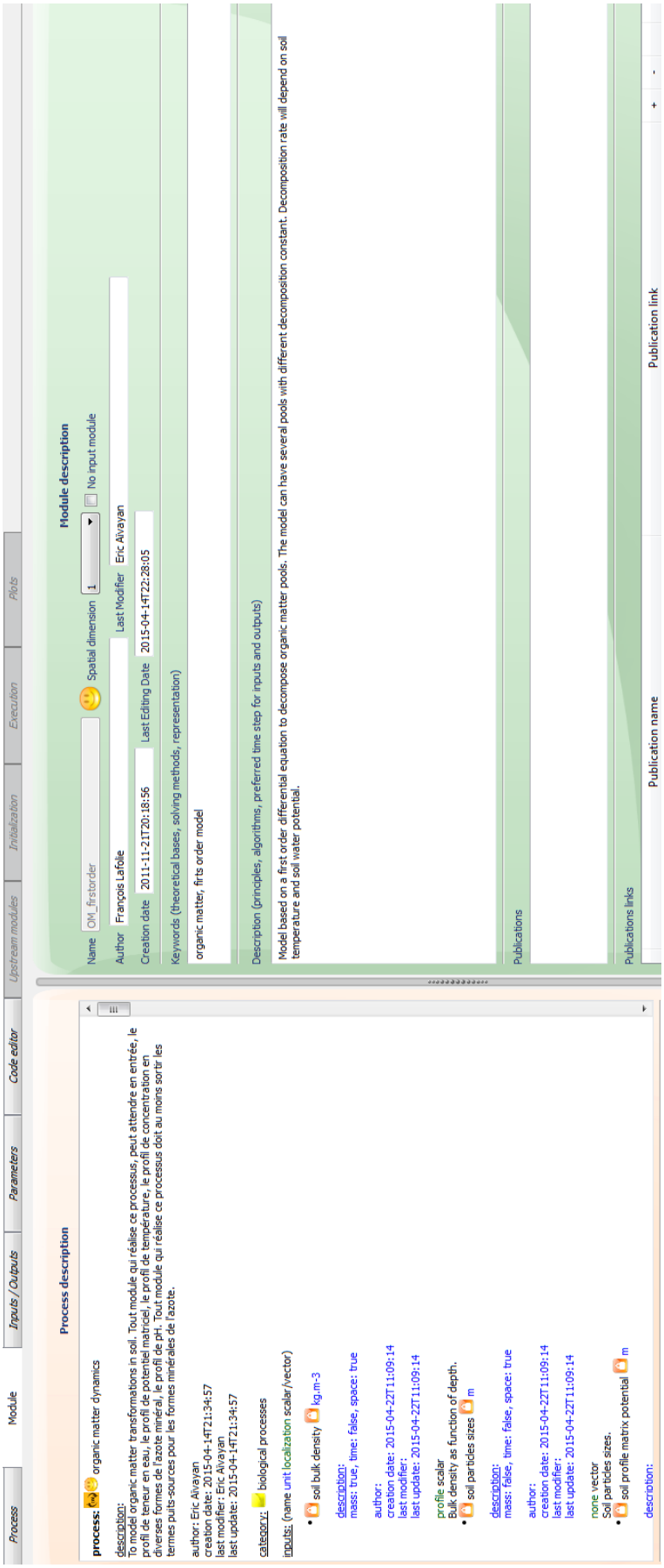

FIGURE 4 – Capture d'écran du composant de plateforme « vsoil-modules ». Il est représenté l'onglet module. Pour de plus amples renseignement se reporter au tutoriel *root water uptake*

Étape 3 : Sélection des entrées/sorties

Module inputs

Used	Name	Descript°	Unit	Localization	
☐	crop residues	...	kg.m-2	surface	vector
☐	crop residues depth	...	m	none	scalar
☐	crop residues gcgms	...	kg.kg-1	surface	vector
☐	crop residues n c	...	undetermined	surface	vector
☐	mulchcontact decomposition solutesink	...	kg.m-3.s-1	profile	vector
☐	mulchcontact decomposition solutesource	...	kg.m-3.s-1	profile	vector
☐	organicfert incorporation depth	...	m	none	scalar
☐	organicfert pools	...	kg.m-2	surface	vector
☐	organicfert pools gcgms	...	NA	surface	vector
☐	organicfert pools n c	...	undetermined	surface	vector
☐	organicmatter pools change	...	kg.kg-1	profile	vector
☐	root biomass	...	kg.m-3	profile	scalar
☐	soil bulk density	...	kg.m-3	profile	scalar
☐	soil particles sizes	...	m	none	vector
☑	soil profile matrix potential	...	m	profile	scalar
☑	soil profile temperature	...	K	profile	scalar
☐	soil profile water content	...	m3.m-3	profile	scalar
☐	solution mobile concentrations	...	kg.m-3	profile	vector
☐	solution pH	...	undetermined	profile	scalar

Module outputs

Used	Name	Descript°	Unit	Localization			Columns names	Initial
☐	organicmatter imobilization cumproftime	...	kg.m-2	none	scalar			☐
☐	organicmatter imobilization rate	...	kg.m-3.s-1	profile	scalar			☐
☐	organicmatter mineralization cumprof	...	kg.m-2.s-1	none	scalar			☐
☐	organicmatter mineralization cumproftime	...	kg.m-2	none	scalar			☐
☐	organicmatter mineralization rate	...	kg.m-3.s-1	profile	scalar			☐
☐	organicmatter nitrification cumprof	...	kg.m-2.s-1	none	scalar			☐
☐	organicmatter nitrification cumproftime	...	kg.m-2	none	scalar			☐
☐	organicmatter nitrification rate	...	kg.m-3.s-1	profile	scalar			☐
☐	organicmatter pools c13	...	undetermined	profile	vector			☐
☐	organicmatter pools gcgms	...	NA	profile	scalar			☐
☐	organicmatter pools n c	...	undetermined	profile	vector			☐
☐	organicmatter pools n15	...	undetermined	profile	vector			☐
☐	organicmatter pools rate	...	kg.kg-1.s-1	profile	vector			☐
☑	organicmatter solutes consumption	...	kg.m-3.s-1	profile	vector		NH4+,NO3-	☐
☐	organicmatter solutes consumptioncumprof	...	kg.m-2.s-1	none	vector			☐
☐	organicmatter solutes consumptioncumproftime	...	kg.m-2	none	vector			☐
☑	organicmatter solutes production	...	kg.m-3.s-1	profile	vector		NH4+,NO3-	☐
☐	organicmatter solutes productioncumprof	...	kg.m-2.s-1	none	vector			☐
☐	organicmatter solutes productioncumproftime	...	kg.m-2	none	vector			☐

FIGURE 5 – Capture d'écran du composant de plateforme « vsoil-modules ». Il est représenté l'onglet de sélection des entrées sorties. Pour de plus amples renseignement se reporter au tutoriel *root water uptake*.La sortie *organicmatter pool* n'est pas visible sur la capture présentée.

Étape 4 : Définition des paramètres

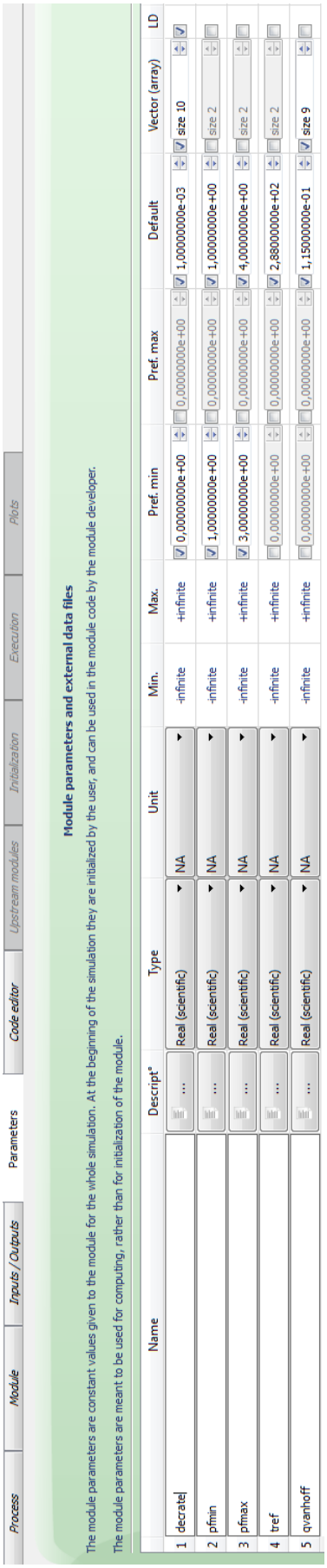

FIGURE 6 – Capture d'écran du composant de plateforme « vsoil-modules ». Il est représenté l'onglet des paramètres. Pour de plus amples renseignement se reporter au tutoriel *root water uptake*

Étape 5 : Choix du langage et renommage des variables

Après avoir sélectionné le *fortran* comme langage de programmation, n'oubliez pas de renommer les variables !

Name	Variable name
Inputs	
soil profile matrix potential	psi
soil profile temperature	temp
Inputs (previous time)	
Outputs	
organic matter pools	pools
organicmatter solutes consumption	sink
organicmatter solutes production	source
Outputs (previous time)	
valid_organic matter pools	valid_pools
valid_organicmatter solutes consumption	valid_sink
valid_organicmatter solutes production	valid_source
Initial outputs values	
organic matter pools	pools_first_state
Parameters	
decrate	decrate
pfmin	pfmin
pfmax	pfmax
tref	tempref
qvanhoff	qvanhoff
Time	
dt	dt

FIGURE 7 – Capture d'écran du composant de plateforme « vsoil-modules ». Il est représenté la partie gauche de l'onglet de l'éditeur de code

3.2 Le code informatique du module

3.2.1 Global variables section

Nous allons définir plusieurs variables globales.

— *npool* qui correspondra au nombre de pools de matière organique. Elle sera initialisée par la suite.

— *perte(1000,10)* qui correspond à la quantité d'azote perdue sur un pas de temps particulier. C'est un tableau de 1000 lignes, pour les nœuds de la grille de calcul, et de 10 colonnes, pour les pools de matière organique.

— *perte_cum(10)* qui correspond à la quantité d'azote perdue sur l'ensemble du profil par pool à un temps t. C'est cette variable que l'on va calculer dans la section « validate ».
Cette variable n'est pas une sortie du processus. On pourrait la rajouter au processus pour s'en servir par la suite dans un autre module au sein d'un modèle. Le but ici est de comprendre l'utilisation de la section « validate ».

```
integer , private :: &
 npool        ! Nombre de pools de MO


double precision , private :: &
perte(VSOIL_MAX_GRID_SIZE,VSOIL_MAX_TAGGED_SIZE),  & ! Quantités de
    N perdues sur le pas de temps pour chaque pool à chaque noeud
perte_cum(VSOIL_MAX_TAGGED_SIZE)        ! Quantités de N perdues sur le pas
    de temps pour chaque pool intégrée sur le profil
```

Listing 1 – Définition de variables globales

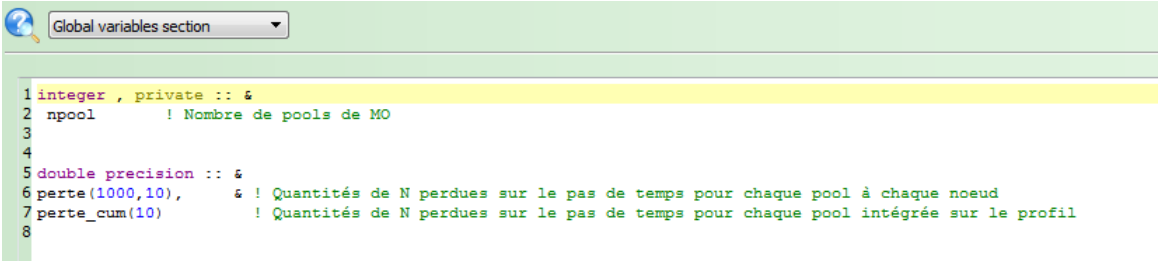

FIGURE 8 – Capture du composant de plateforme « vsoil-modules » sur l'onglet *code editor*. Il est représenté la partie édition du code sur la partie *Global variables section*.

3.2.2 Initialization section.

Dans un premier temps, nous déclarons trois variables locales comme indiqué dans le listing 2.

```
integer ipool
double precision pool_loc(VSOIL_MAX_GRID_SIZE),pool_calc(
    VSOIL_MAX_GRID_SIZE)
```

Listing 2 – Déclaration des variables locales de la section d'initialisation des variables.

Puis on initialise les deux sorties qui n'ont pas été déclarées *initial value required*. De même on initialise les variables globales que l'on a déclaré à la section précédente.

```
source = 0.d0
sink = 0.d0
perte = 0.d0
perte_cum = 0.d0    ! Attention ici initialisation de la variable ajoutée pour le
    validate

! Initialize organic pool at grid nodes

npool = pools_vector_cols_nb

do ipool = 1,npool
```

Listing 3 – Initialisation des variables.

Pour continuer, nous devons calculer les valeurs des différentes pools sur les nœuds de calcul de notre grille. En effet, à l'initialisation les pools de matière organique ont pu être définies sur une grille différente de celle utilisée pour l'exécution du module.

C'est pourquoi nous itérons sur toutes les pools de matière organique les opérations suivante :

1. On sauve dans la variable locale l'état initial d'une pool dans la variable *pool_loc*. C'est la première opération de la boucle.
2. On appelle la fonction d'interpolation linéaire agissant sur des tableaux de la plateforme vsoil_interpolation_linear_vector.

   On lui fournit dans l'ordre, le tableau des profondeurs correspondant aux valeurs initiales des pools, le tableau des valeurs de la pool pour les différentes profondeurs, le nombre de lignes des tableaux précédent, le tableau des valeurs des profondeurs correspondant à la grille de calcul, le tableau où sauvegarder les valeurs de la pool pour ces profondeurs et le nombre de lignes de ces deux derniers tableaux.
3. On sauve le résultat de cette interpolation dans le tableau correspondant.

```
    pool_loc(1:pools_depth_values_nb_first_state) = pools_first_state(1:
     pools_depth_values_nb_first_state,ipool)

    call vsoil_interpolation_linear_vector( pools_depth_first_state , &
          pool_loc , pools_depth_values_nb_first_state , &
          vsoil_grid_nodes , pool_calc , vsoil_grid_size )

    pools(1:vsoil_grid_size,ipool) = pool_calc(1:vsoil_grid_size)

end do
```

Listing 4 – Interpolation linéaire des valeurs des pools sur la grille de calcul

On termine en faisant une conversion de la température de référence de kelvin à celsius.

```
! Convert tempref to Celsius
tempref = tempref - TZEROKELVIN
```

Listing 5 – Conversion de la température de référence.

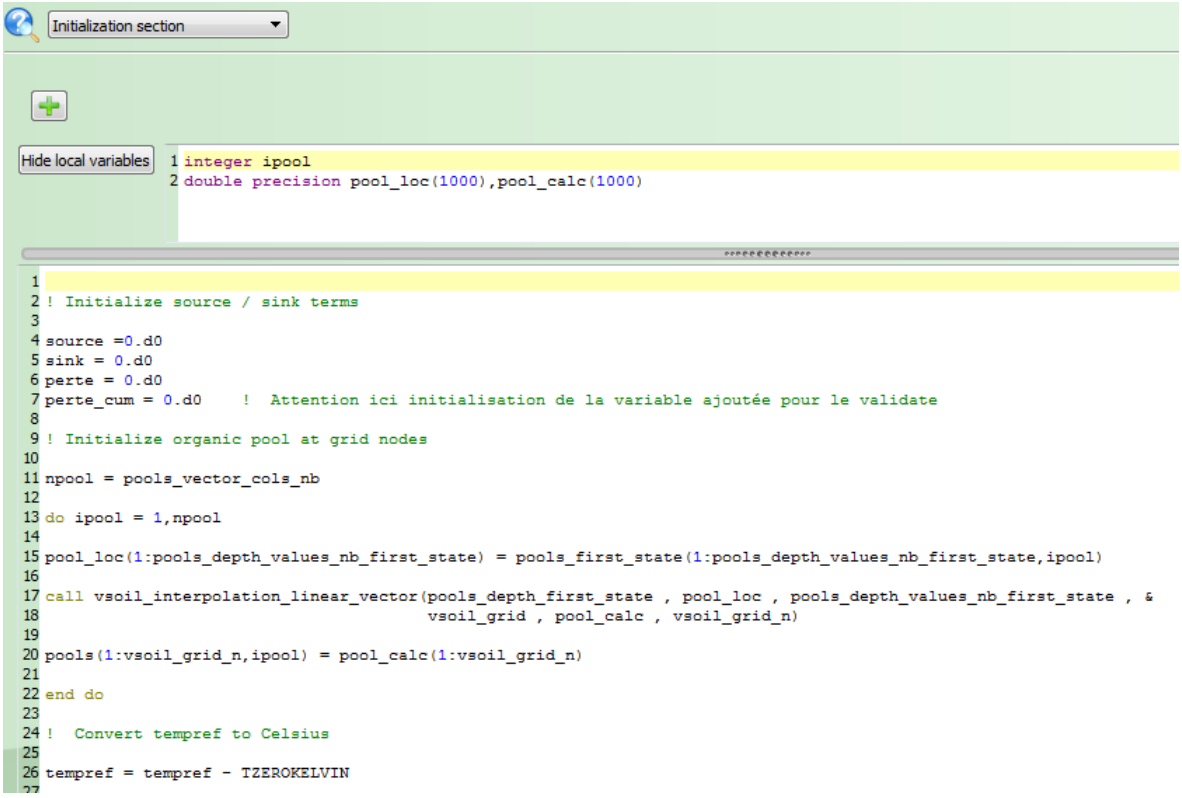

FIGURE 9 – Capture d'écran du composant de plateforme « vsoil-modules ». Il est représenté la partie de l'éditeur de code dans la section d'initialisation des variables.

3.2.3 Compute section

Dans cette partie, après avoir déclaré des variables de calculs intermédiaires, on applique la résolution de l'équation différentielle du premier ordre pour tous les points de la grille de calcul (première boucle) sur toutes les pools de matière organique (boucle la plus interne).

La valeur de la pool au temps précédent est représentée par valid_pools(i,j). Cette valeur n'est pas modifiée tant que la plateforme n'est pas passé au temps suivant.

```
integer i , j
double precision xt,xpsi,aux
```

Listing 6 – Déclaration des variables locales de la section de calcul du module.

```
! Loop over grid nodes to calculate source term
! Psi is multiplied by 100 because andren function works with centimeters
! Temperatures are converted to Celsius because vanthoff function works with Celsius
  .
! Tempref already converted to Celsius.

do i=1,vsoil_grid_size
    xt=vsoil_vanthoff(temp(i)-TZEROKELVIN!,tempref,qvanhoff)
    xpsi=vsoil_andren(psi(i)*100,pfmin,pfmax)
    do j = 1 , npool
        aux=dt*decrate(vsoil_grid_nodes_horizon_index(i),j)*xt*xpsi
        source(i,2) = source(i,2)+aux*valid_pools(i,j)
        pools(i,j)=valid_pools(i,j)/(1+aux)
        perte(i,j) = pools(i,j) - valid_pools(i,j)
    end do
end do
```

Listing 7 – Section de calcul du module.

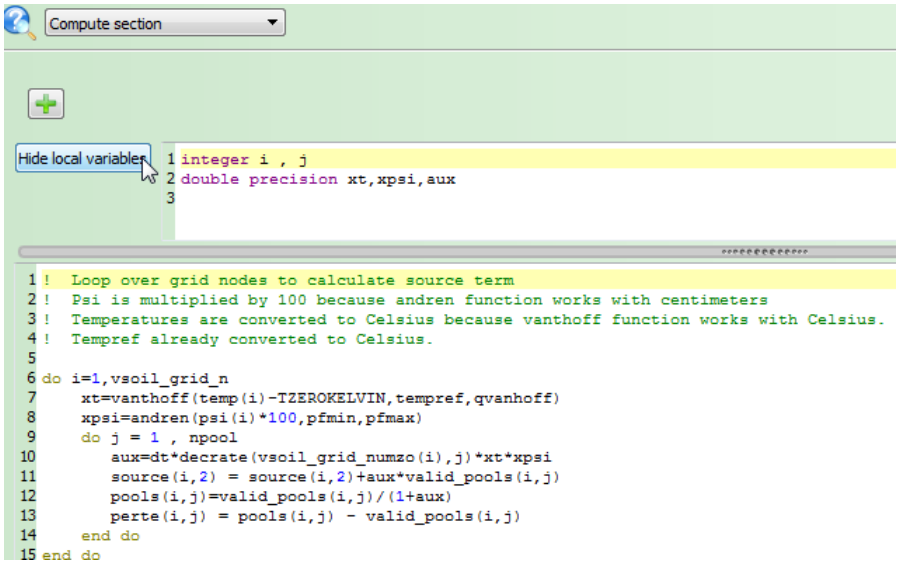

FIGURE 10 – Capture d'écran du composant de plateforme « vsoil-modules ». Il est représenté la partie de l'éditeur de code dans la section de calcul.

3.2.4 Internal functions section

Nous donnons à titre informatif, le code des fonctions de Vant'Hoff et de Andren.

```
!-----------------------------------------------------------------------
!     FUNCTION VANTHOFF
!-----------------------------------------------------------------------
!     Use the Vant'Hoff exponential function to model temperature
!     dependence of mineralization and nitrification processes.
!
!     F(T)=exp(Q(T-Tref))
!
!     T is temperature (C)
!     Tref is the temperature (C) corresponding to the nominal biological
   transformation rate
!     Q (T-1) is Vant'Hoff parameter for the corresponding process.
!
!-----------------------------------------------------------------------
!
      double precision function vanthoff(temp,tempref,qvanthoff)
!
      double precision,intent(in) :: temp,tempref,qvanthoff(9)

      double precision deltat,fact
      integer ntemp,i
!
      fact = 0.d0
      if(temp.gt.0.d0) then
        fact = 1.d0
        ntemp=int(1.d0+temp/5.d0)
        if(ntemp.gt.9) ntemp=9
        if(temp.ge.tempref) then
          do 13 i=4,ntemp
            deltat = 5.d0
            if(i.eq.ntemp) deltat = temp-5.d0*dble(i-1)
            fact = fact * dexp(qvanthoff(i)*deltat)
 13       continue
        else
          do 17 i=3,ntemp,-1
            deltat = -5.d0
            if(i.eq.ntemp) deltat = temp-5.d0*dble(i)
            fact = fact * dexp(qvanthoff(i)*deltat)
 17       continue
        end if
      end if
!
      vanthoff=fact
!
      return
```

```
      end function vanthoff

!-----------------------------------------------------------------------
!     FUNCTION ANDREN
!-----------------------------------------------------------------------
!     Use the Andren's model to account for mineralization and nitrification
!     processes dependence on soil water potential
!
!     F(Psi)=(log10(psi_min_bio)-log10(psi))/(log10(psi_min_bio)-log10(psi_opt_bio
    ))
!
!     PSI is soi water potential in centimeter.
!
!-----------------------------------------------------------------------
!
      double precision function andren(psi,pfmin,pfmax)
!
      double precision,intent(in) :: psi,pfmin,pfmax

      double precision facw,pf

      facw = 1.d0
      if(psi.lt.0.d0) then
        pf = dlog10(-psi)
        if(pf.le.pfmax) then
          if(pf.gt.pfmin) then
            facw = (pfmax-pf)/(pfmax-pfmin)
          end if
        else
          facw = 0.d0
        end if
      end if
!
      andren=facw
!
      return
    end function andren
```

Listing 8 – Section des fonctions internes (Vant'Hoff et de Andren).

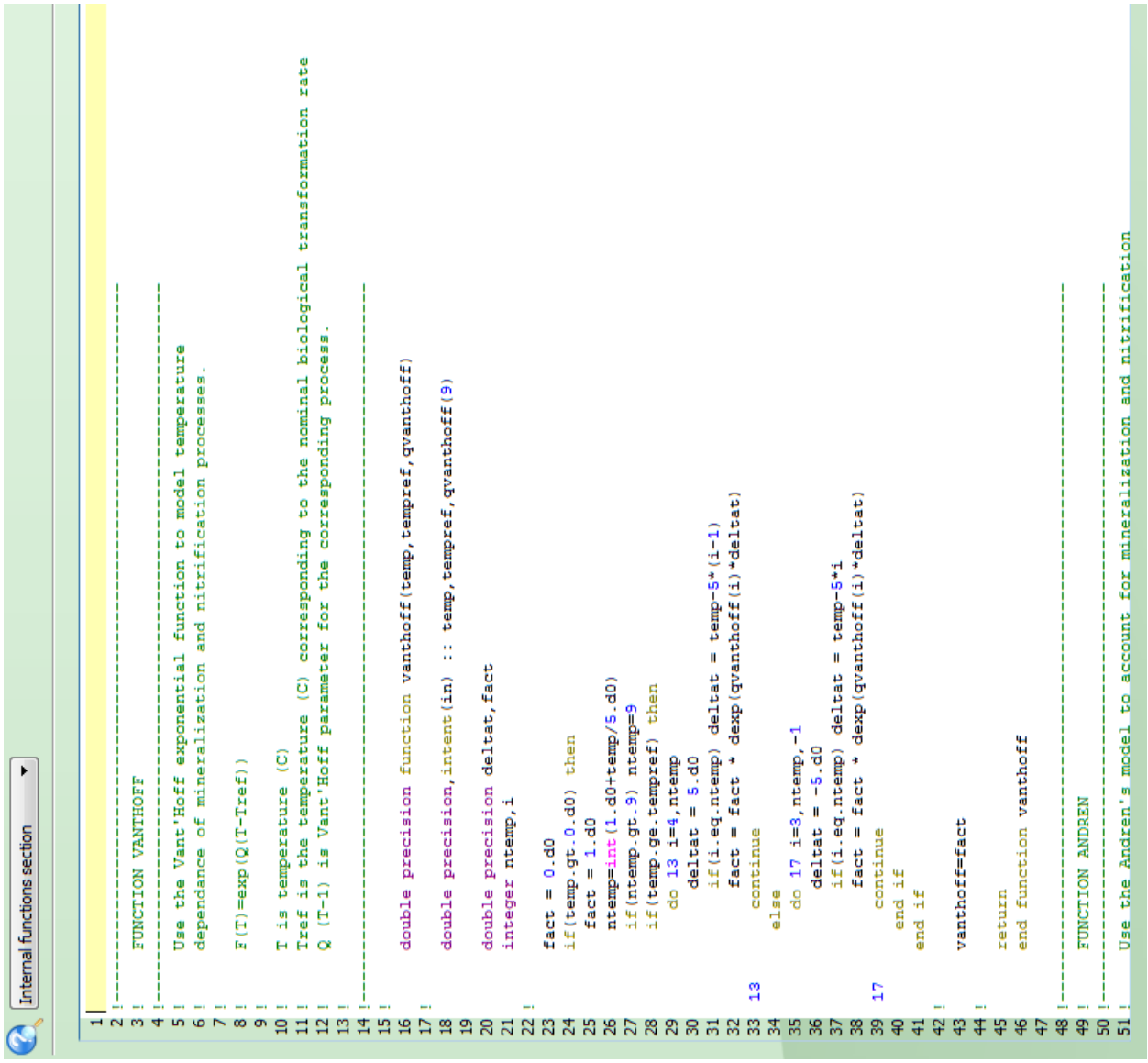

FIGURE 11 – Capture d'écran du composant de plateforme « vsoil-modules ». Il est représenté la partie de l'éditeur de code dans la section des fonctions internes

3.2.5 Validate section

Cette section n'est exécuté que lorsque tous les modules ont accepté le pas de temps proposé par la plateforme. Les valeurs des sorties sont donc fixées. On peut ajouter les résultats à une variable d'accumulation.

Dans la partie des variables globales, nous avions déclaré la variable perte_cum qui correspond à la perte cumulée de matière organique au cours du temps.

Cette variable ne correspond pas à une sortie. Pour se faire, il faut créer une entrée–sortie du processus correspondant à la perte cumulée au cours du temps.

```
double precision aux(VSOIL_MAX_GRID_SIZE),veccum
integer ipool
```

Listing 9 – Déclaration des variables locales de la section validate du module.

Après avoir déclaré les variables locales nécessaires (voir le listing 9), nous copions la valeur des pertes pour une pool dans un tableau.

Nous appelons la méthode d'intégration sur le profil de la plateforme et stockons le résultat dans la variable veccum.

Enfin nous cumulons cette perte avec celles déjà calculées dans la variable perte_cum. Comme elle se présente sous forme de tableau, elle permet de contenir les pertes cumulées de chaque pool.

```
do ipool = 1,npool
    aux (1:vsoil_grid_size) = perte(1:vsoil_grid_size , ipool)
    call vsoil_generic_integration(aux, vsoil_grid_nodes_delta , vsoil_grid_size ,
    veccum)
        perte_cum (ipool) = perte_cum (ipool) + veccum
end do
```

Listing 10 – Section validate du module.

Dans le cas où une moyenne temporelle devrait être effectuée, c'est dans cette partie que nous le ferions à l'image de cette perte cumulée.

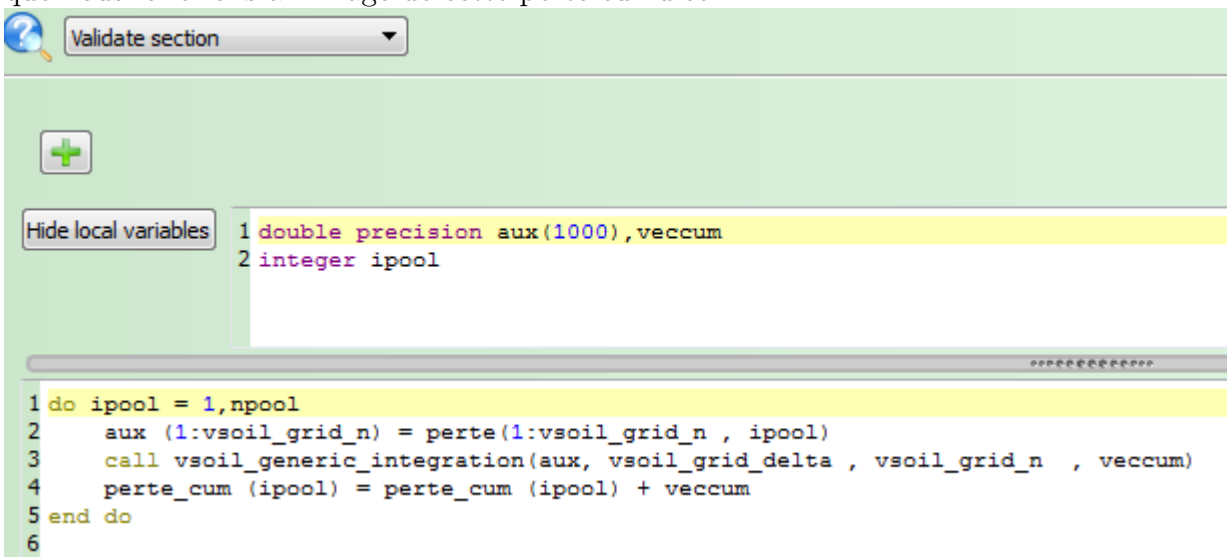

FIGURE 12 – Capture d'écran du composant de plateforme « vsoil-modules ». Il est représenté la partie de l'éditeur de code dans la section validate.