Tutoriel création de module avec Vsoil

denitrification simple

Éric Aivayan, Nicolas Moitrier

2024-02-29

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.

Pour ce tutoriel, nous allons utiliser un modèle simple de dénitrification dépendant de la température, de l'humidité et de la concentration en nitrate.

Pour un point du profil de sol, la vitesse de dénitrification est donnée par

$$R_{\text{Denitrification}} = D_{Max} \times f(T) \times g(H) \times k(C) \tag{1}$$

où

— $R_{\text{Denitrification}}$ est la vitesse de dénitrification en $kg.m^{-3}.s^{-1}$,
— D_{Max} est la vitesse de dénitrification maximale pour un horizon de sol donné en $kg.m^{-3}.s^{-1}$,
— $f(T)$ est le coefficient correspondant à la dépendance en température et suivant une loi du q10 (représenté sur la figure 1),
— $g(H)$ est le coefficient correspondant à la dépendance à l'humidité et suivant une loi abiotique (représenté sur la figure 2).
— $k(C)$ est le coefficient correspondant à la dépendance à la concentration en nitrate donné par le rapport de Michaelis (représenté sur la figure 3).

La production de N_2O, en un point du profil de sol, est calculée simplement par multiplication de la vitesse de dénitrification par l'abondance relative de N_2O dans la production des gaz azotés.

Pour obtenir la production totale, il suffit de faire une intégration sur l'ensemble des points du profil de sol.

2 Ce que vous allez découvrir.

Dans ce deuxième tutoriel, vous allez approfondir vos connaissances sur la manipulation des entrées et des sorties, en particulier lorsqu'elles sont de type *vector*. En plus de ce premier point vous allez :

— Sélectionner les colonnes d'une sortie de type « vector » dans l'onglet « Inputs/Outputs ».
— Définir et utiliser des variables globales.
— Déterminer le numéro de colonne correspondant à une donnée particulière d'une entrée de type « vector ».
— Utiliser des paramètres qui sont dépendant des couches (*Layer Dependent*).
— Travailler sur le profil.

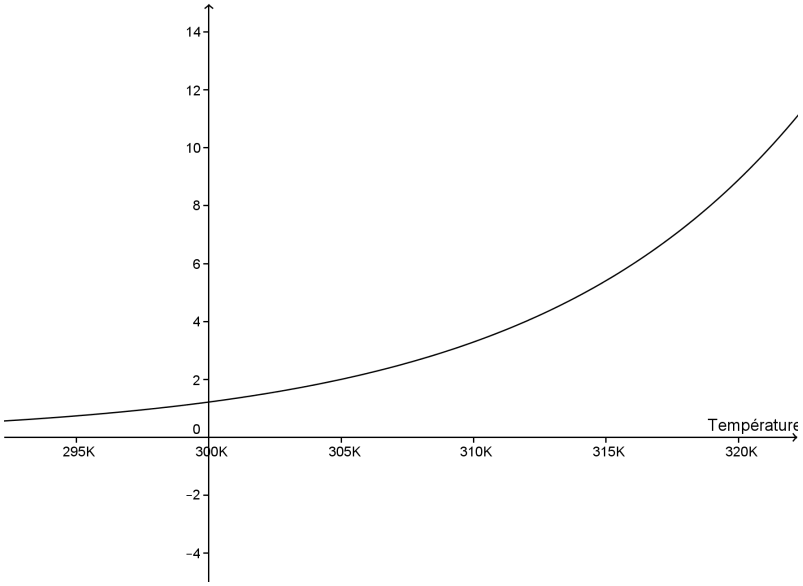


FIGURE 1 – Allure de la loi du q10 pour une température de référence de 298K et un coefficent de 2,7.

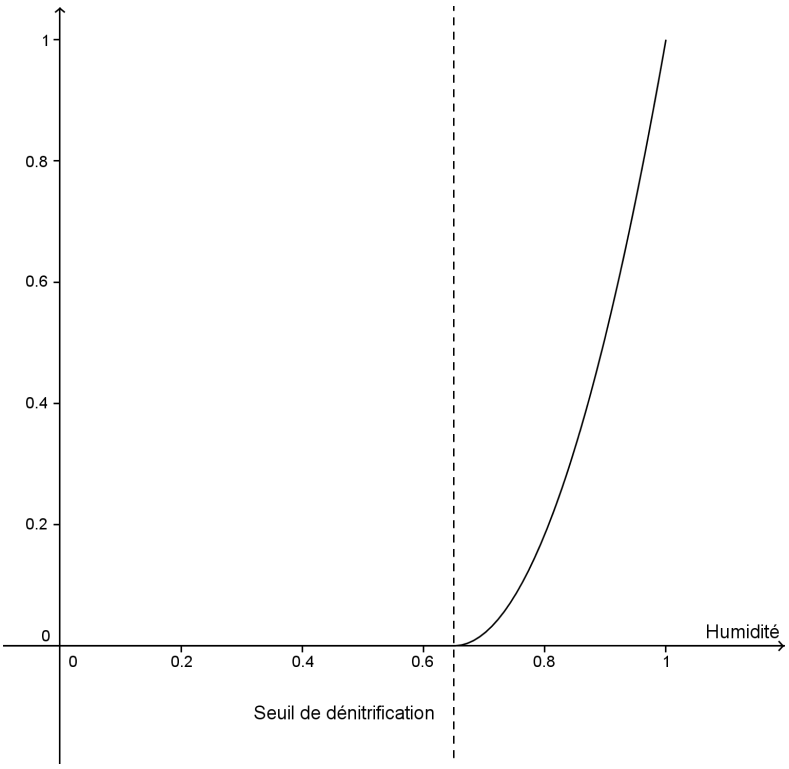


FIGURE 2 – Allure de la loi abiotique pour une puissance de 2 et un seuil de dénitrification de 65%.

Nous passerons très rapidement sur les points ayant été déjà vus dans le premier tutoriel et ne présentant pas de difficulté particulière.

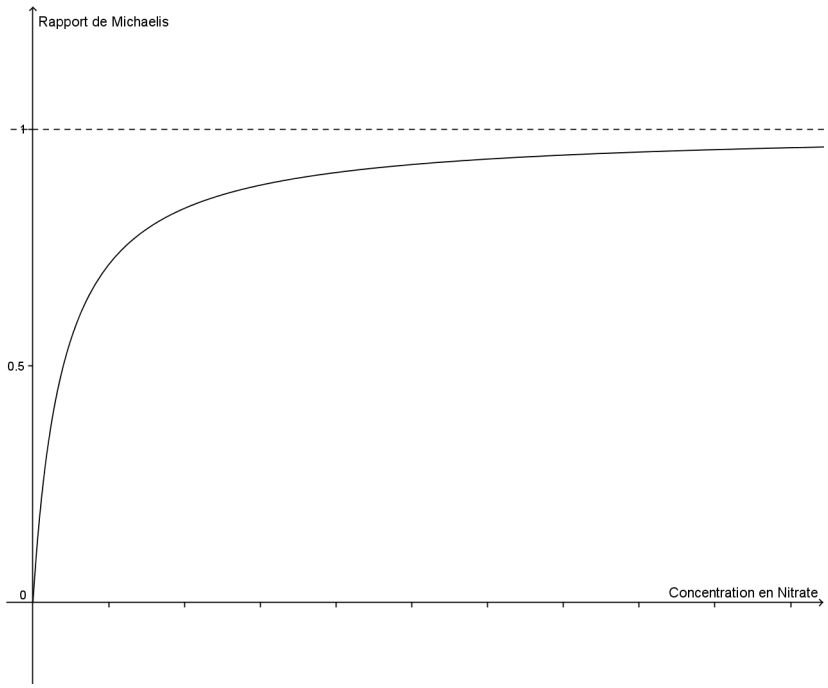


FIGURE 3 – Allure du rapport de Michaelis.

3 En avant !

3.1 Choix du processus simulé

Le processus utilisé dans ce module est *denitrification*. Avant de vous lancer dans la création du module vérifiez que votre version du processus contient au minimum en entrée

— soil bulk density, kg.m-3, profile, scalar
— soil temperature, K, profile, scalar
— soil saturation index, ratio0–1, profile, scalar
— soil mobile solution concentration, kg.m-3, profile, vector

et en sortie

— soil organic matter gas production, mol.m-3.s-1, profile vector
— soil solution denitrification rate, mol.m-3.s-1, profile, scalar
— soil solution denitrification rate profile cumulated, kg.m-2.-s1, none, scalar.

Si cela n'est pas le cas vous devez modifier votre processus pour qu'il ait toutes ces entrées et sorties.

3.2 Caractérisation du module

Nous allons construire le module *denit_simple*. La dimension spatiale de ce module sera de 1.

Vous pouvez mettre en description, par exemple, *Denitrification is calculated as the product of a maximum denitrification rate with simple relationships describing the dependence on temperature, nitrate availability and saturation index.*

3.3 Sélection des entrées/sorties

Comme vous vous en doutez déjà, nous allons sélectionner les entrées suivantes :

— soil bulk density,
— soil temperature,
— soil saturation index,
— soil mobile solution concentration,
et les sorties suivantes :
— soil organic matter gas production,
— soil solution denitrification rate,
— soil solution denitrification rate profile cumulated.

Vous remarquez sur la ligne de la sortie *soil organic matter gas production* le symbole car elle est du type *vector*. Nous allons choisir maintenant quelles seront les composantes de cette variable.

Cliquez dessus et dans la fenêtre qui s'ouvre sélectionnez N2 et N2O.

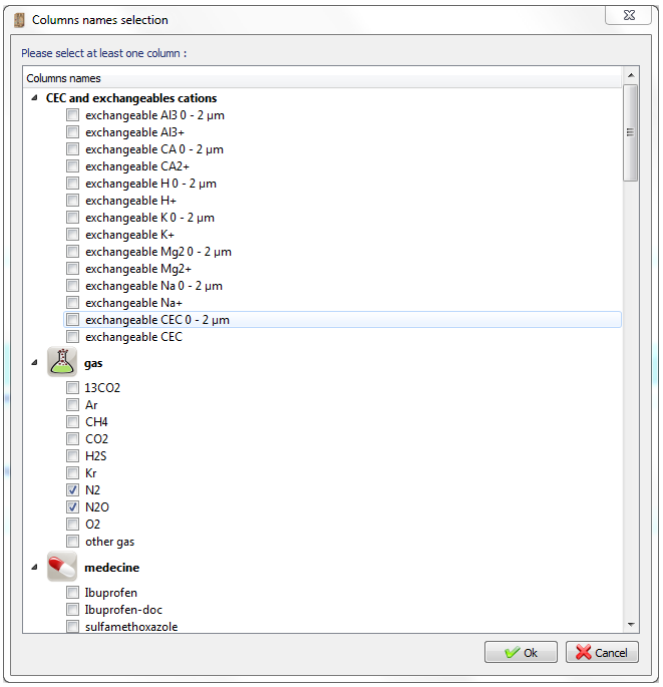


FIGURE 4 – Sélection des noms de colonnes.

Nous avons, par cette manipulation, sélectionné les colonnes de la sortie. L'ordre d'affichage donne les entêtes du tableau correspondant. N2 correspond à la première colonne et N2O correspond à la seconde. Cet ordre est utilisé dans l'onglet « Code Editor ».

Si vous modifiez votre module en ajoutant des colonnes (au maximum 10), il faudra reporter toutes les modifications d'ordre dans le programme !

3.4 Définition des paramètres

Pour ce module, nous avons besoin de paramètres caractérisant les couches de sol ou le processus de dénitrification lui même. Ainsi certains paramètres seront *Layer dependent* et d'autres non. Pour l'instant, nous allons entrer les caractéristiques de ces paramètres sans chercher à trop comprendre le problème de la dépendance en couche.

La capture d'écran 5, vous donne une idée du résultat que nous allons obtenir.

Comme pour le premier tutoriel, nous donnons un tableau récapitulatif de tous les paramètres.

Nom	Description	Type	Unit	Pref. min	Pref. max	default	Vector(array)	LD
denitmax	Maximum denitrification rate under non limiting nitrate and carbon conditions.	Real(sc)	kg.kg-1.s-1					☑
isdenit	Saturation index at which denitrification starts	Real(sc)	NA			7e-01		☑
power	Exposant of the power law describing denitrification dependence on saturation index f(is) = is**power	Real(sc)	NA			2		☑
michaelisdenit	Michaelis-Menten constant for denitrification dependence on nitrate concentation.	Real(sc)	kg.m-3					☑
q10den	Q10 coefficient for temperature dependence	Real(sc)	NA					
temp_inf	Lower temperature for denitrification	Real(sc)	K			2,83e+02		
Temp_opt	OPtimum temperature for denitrification	Real(sc)	K			2,98e+02		
rN2Oratio	Ratio N2O/(N2O+N2) of the denitrification. Used to calculate production of N2O	Real(sc)	NA			2,0e-01		

Comme vous pouvez le remarquer, tous les paramètres n'ont pas de valeurs par défaut. Lors de l'utilisation du module, on verra apparaître les valeurs par défaut dans les champs de saisie des paramètres ayant une valeur par défaut, tandis que l'utilisateur devra fournir des valeurs pour les paramètres qui n'ont pas de valeurs par défaut.

Module parameters and external data files

The module parameters are constant values given to the module for the whole simulation. At the beginning of the simulation they are initialized by the user, and can be used in the module code by the module developer.

The module parameters are meant to be used for computing, rather than for initialization of the module.

	Name	Descript°	Type	Unit	Min.	Max.	Pref. min	Pref. max	Default	Vector (array)	LD
1	denitmax	...	Real (scientific)	kg.kg-1.s-1	-infinite	+infinite	0,00000000e+00 kg	0,00000000e+00 kg	0,00000000e+00 kg	size 2	☑
2	isdenit	...	Real (scientific)	NA	-infinite	+infinite	0,00000000e+00	0,00000000e+00	☑ 7,00000000e-01	size 2	☑
3	power	...	Real (decimal)	NA	-infinite	+infinite	0,00000000000000(	0,00000000000000(	☑ 2,00000000000000(	size 2	☑
4	michaelisdenit	...	Real (scientific)	kg.m-3	-infinite	+infinite	0,00000000e+00 kg	0,00000000e+00 kg	0,00000000e+00 kg	size 2	☑
5	q10den	...	Real (scientific)	NA	-infinite	+infinite	0,00000000e+00	0,00000000e+00	0,00000000e+00	size 2	☐
6	temp_inf	...	Real (scientific)	K	-infinite	+infinite	0,00000000e+00 K	0,00000000e+00 K	☑ 2,83000000e+02 K	size 2	☐
7	Temp_opt	...	Real (scientific)	K	-infinite	+infinite	0,00000000e+00 K	0,00000000e+00 K	☑ 2,98000000e+02 K	size 2	☐
8	rN2Oratio	...	Real (scientific)	NA	-infinite	+infinite	0,00000000e+00	0,00000000e+00	☑ 2,00000000e-01	size 2	☐

FIGURE 5 – Capture d'écran du composant de plateforme vsoil-module sur l'onglet « Parameters ». Vue des paramètres créés pour le module de dénitrification simple.

3.5 Le code informatique du module

Dans cette section, nous allons détailler les principaux points de ce tutoriel.

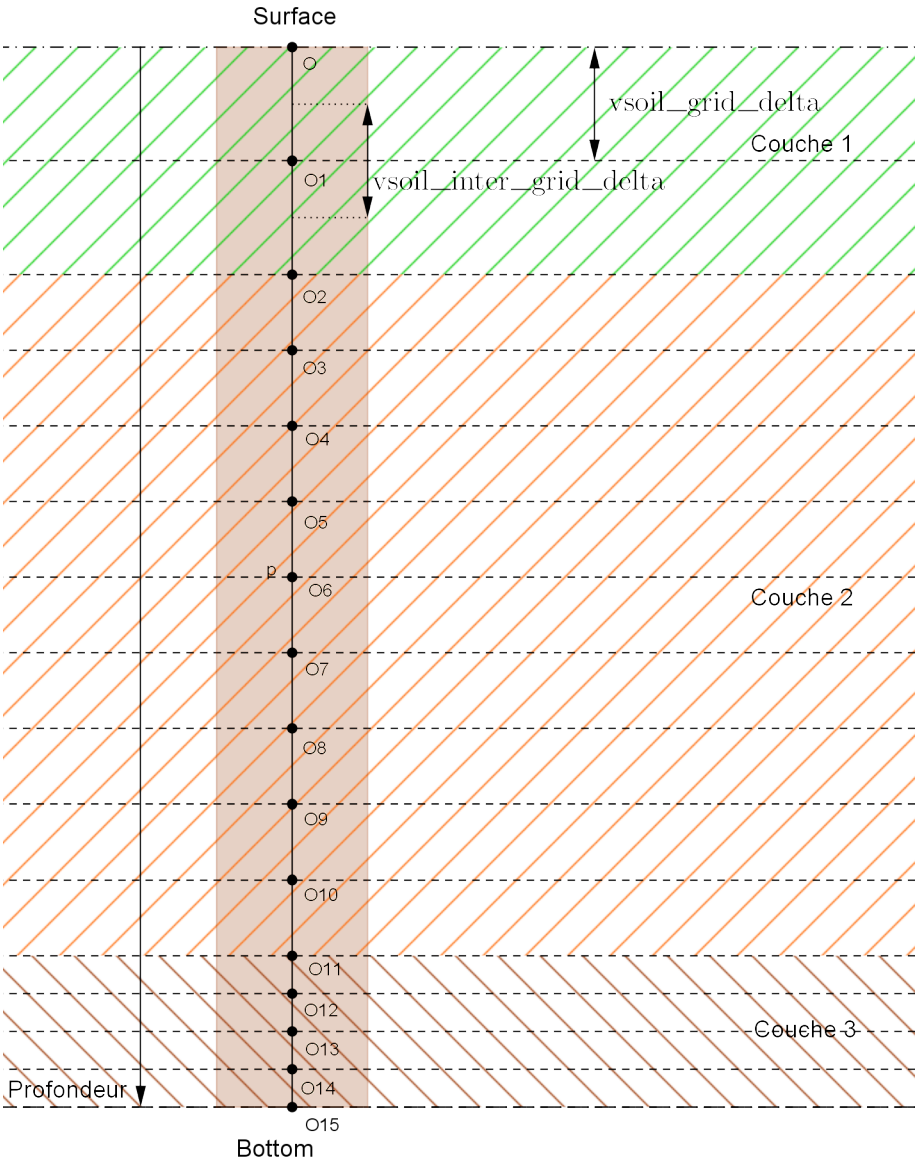
Dans la section « Initialization section », nous allons regarder de plus près comment on récupère et on produit des données vectorielles. Deux mécanismes sont présentés.

Dans la section « Compute section », nous allons travailler avec les données dépendant des couches. Pour éclaircir le propos, nous allons commencer par définir les termes de couches, de points de calcul, etc. dans la prochaine sous section.

Nous allons écrire un module en *Fortran*.

3.5.1 Où l'on découpe le profil de sol

Un schéma valant plus qu'un long discourt, nous avons représenté un profil de sol avec les différentes informations que nous devons comprendre.



Sur ce profil de sol, nous avons appliqué un maillage de 16 points (de O à O15). Ces seize points appartiennent à des couches différentes soit à la couche 1 soit à la couche 2 soit à la

couche 3.

Par convention le point de maillage à une interface entre deux couches appartient à la couche inférieure. Ainsi le point de maillage O2 appartient à la couche 2 et le point de maillage O11 appartient à la couche 3.

Par conséquent la couche 1 contient deux points, la couche 2 contient 9 points et la couche 3 contient 4 points. Comme cette couche est la dernière, elle contient aussi le dernier point de maillage.

Dans les variables fournies par la plateforme, *vsoil_grid_n* vaudra 16. Le tableau *vsoil_grid_nodes_hor* associera chaque point du maillage à une couche. Ainsi lorsque l'on demande vsoil_grid_nodes_horizon_in (4) on obtiendra 2 puisque le quatrième point du maillage appartient à la couche 2.

La dépendance en couche (*Layer Dependent*) implique que pour chacune des couches, les valeurs d'un paramètre peuvent changer et doivent donc être récupérées pour chaque point de calcul.

3.5.2 Global variables section

Les variables définies dans cette section sont accessibles dans toutes les autres section de code. Elles peuvent être des informations que l'on veut déterminer une seule fois et réutiliser à de nombreuses reprises.

Dans notre cas, nous avons une entrée et une sortie de type *vector*. Il faut donc déterminer la place des données que l'on veut récupérer ou fournir.

Nous allons donc créer trois variables globales qui contiendront la position dans le vecteur des éléments que l'on désire utiliser. Ce qui se traduit par la section de code donnée dans le listing 1

```
integer , private :: &
ino3,   & ! Nitrate index in the concentrations vector
in2o,   & ! N2O index in the gaz production vector
in2       ! N2 index in the gaz production vector
```

Listing 1 – Définition des variables globales

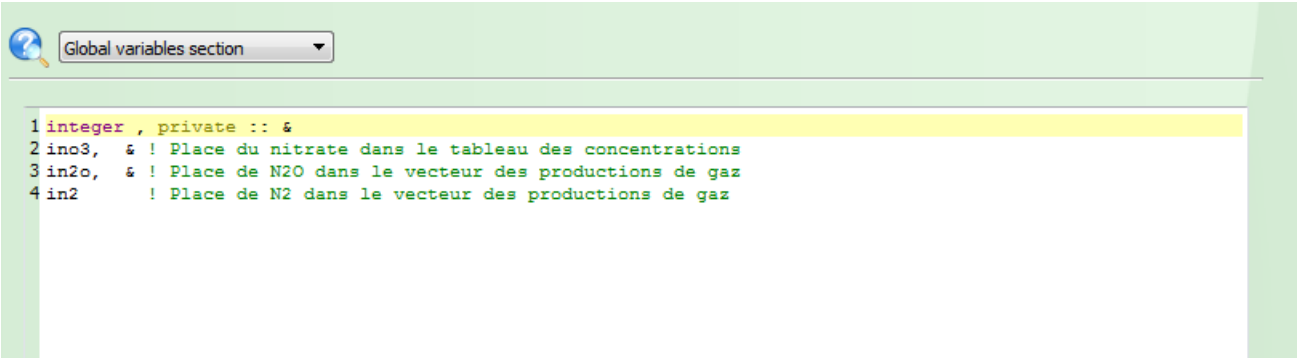


FIGURE 6 – Capture d'écran du composant de plateforme vsoil_modules sur l'onglet de l'éditeur de code. Il est représenté la partie droite lorsque la section de définition des variables globales est sélectionnée.

3.5.3 On renomme les entrées/sorties

Comme dans le premier tutoriel, nous allons renommer les entrées et les sorties. Pour cela nous vous renvoyons au premier tutoriel. Le tableau suivant résume l'ensemble des changements

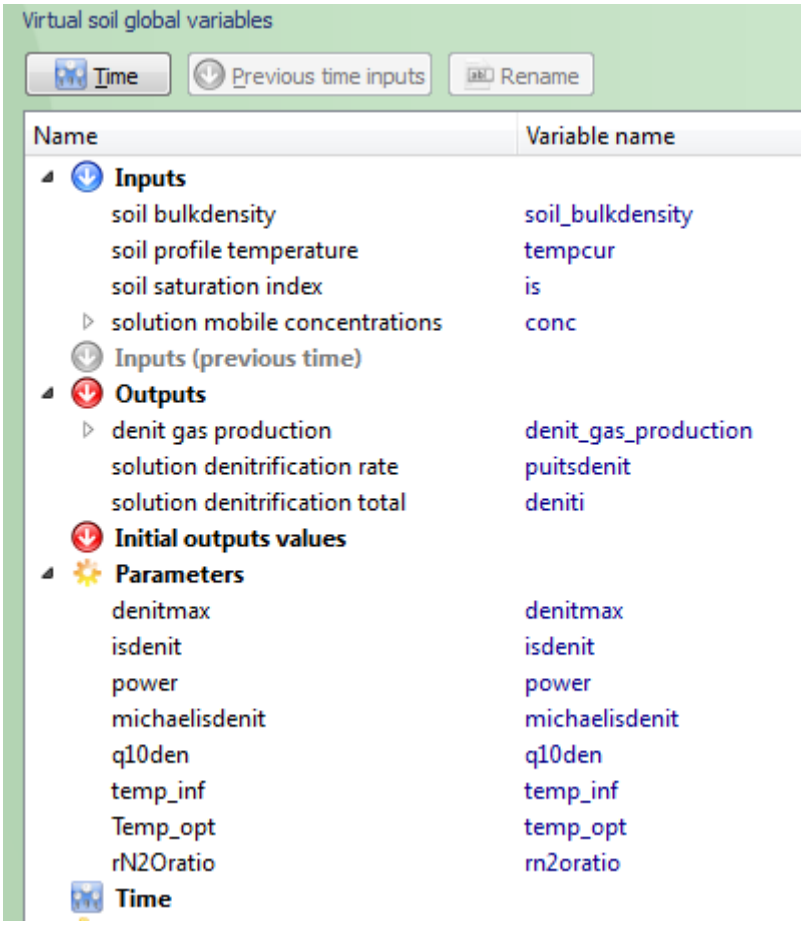


FIGURE 7 – Capture d'écran du composant de plateforme vsoil_modules sur l'onglet de l'éditeur de code. Il est représenté la partie gauche montrant l'ensemble des entrées, des sorties et des paramètres avec leur nom d'origine et leur nouveau nom.

de noms sur les entrées et les sorties.

Entrée	Nouveau nom
soil bulk density	soil_bulkdensity
soil temperature	tempcur
soil saturation index	is
soil mobile solution concentration	conc

Sortie	Nouveau nom
soil organic matter gas production	denit_gas_production
soil solution denitrification rate	puitsdenit
soil solution denitrification rate profile cumulated	deniti

3.5.4 Initialization section

Pour créer ce module, nous allons avoir un gros travail d'initialisation. Nous allons successivement voir :

— Comment initialiser une variable globale pour quelle fournisse le numéro de colonne d'un vecteur de sortie.

— Comment initialiser une variable globale pour quelle fournisse le numéro de colonne d'un vecteur d'entrée.

— Comment indiquer à la plateforme qu'une étape essentielle a échoué et qu'elle ne doit pas poursuivre son exécution.
— Comment initialiser les sorties définies sur le profil de sol.

Numéro de colonne d'un vecteur de sortie. Dans la section Global variables section page 12, nous avons défini des variables qui allaient contenir la position d'informations dans les vecteurs d'entrée et de sortie. Il faut donc leur affecter une valeur. La section d'initialisation est le lieu pour cela.

Pour la sortie, nous avons explicitement choisi les données que nous allons fournir : N_2 et N_2O dans cet ordre. Ainsi la première colonne de notre vecteur correspond au taux de production de diazote et la seconde à celui de protoxyde d'azote. Il est ainsi aisé d'affecter une valeur aux variables globales in2 et in2o.

Le code d'initialisation est ainsi très simple :

```
! Initialization
```

Listing 2 – Initialisation des numéros de colonnes de la sortie.

Attention ! Il faudra se souvenir que le choix des colonnes du vecteur de sortie dans l'onglet *Inputs/Outputs* a un impact direct sur la partie initialisation.

Numéro de colonne d'un vecteur d'entrée. Pour l'entrée, déterminer la colonne qui correspond au nitrate n'est pas aussi immédiat. En effet, rien ne présuppose de la place du nitrate dans le vecteur d'entrée *soil mobile solution concentration*. D'ailleurs, il n'est même pas certain que le module qui fournit cette entrée, contienne une information sur la concentration de nitrate !

Ainsi, il faudra chercher dans le vecteur d'entrée si une colonne porte le nom « NO3- ». Si on trouve cette colonne alors on enregistre sa position dans la variable ino3 que nous avons défini dans la section « Global variables section ». Dans le cas où on ne trouve pas cette colonne il faut informer le programme qu'il ne peut pas continuer : des données essentielles pour le mécanisme du module n'existent pas.

La plateforme met à disposition des variables secondaires de l'entrée que nous allons utiliser. Il s'agit de *soil mobile solution concentration vector cols nb* et de *soil mobile solution concentration vector cols names* qui ont été renommées en *conc_vector_cols_nb* et *conc_vector_cols_names*. Elles apparaissent dans la partie de gauche de l'onglet « Code Editor » comme le montre la figure 8. Ces deux variables contiennent respectivement le nombre de colonnes du vecteur et la liste ordonnée des noms de colonnes sous forme d'un tableau.

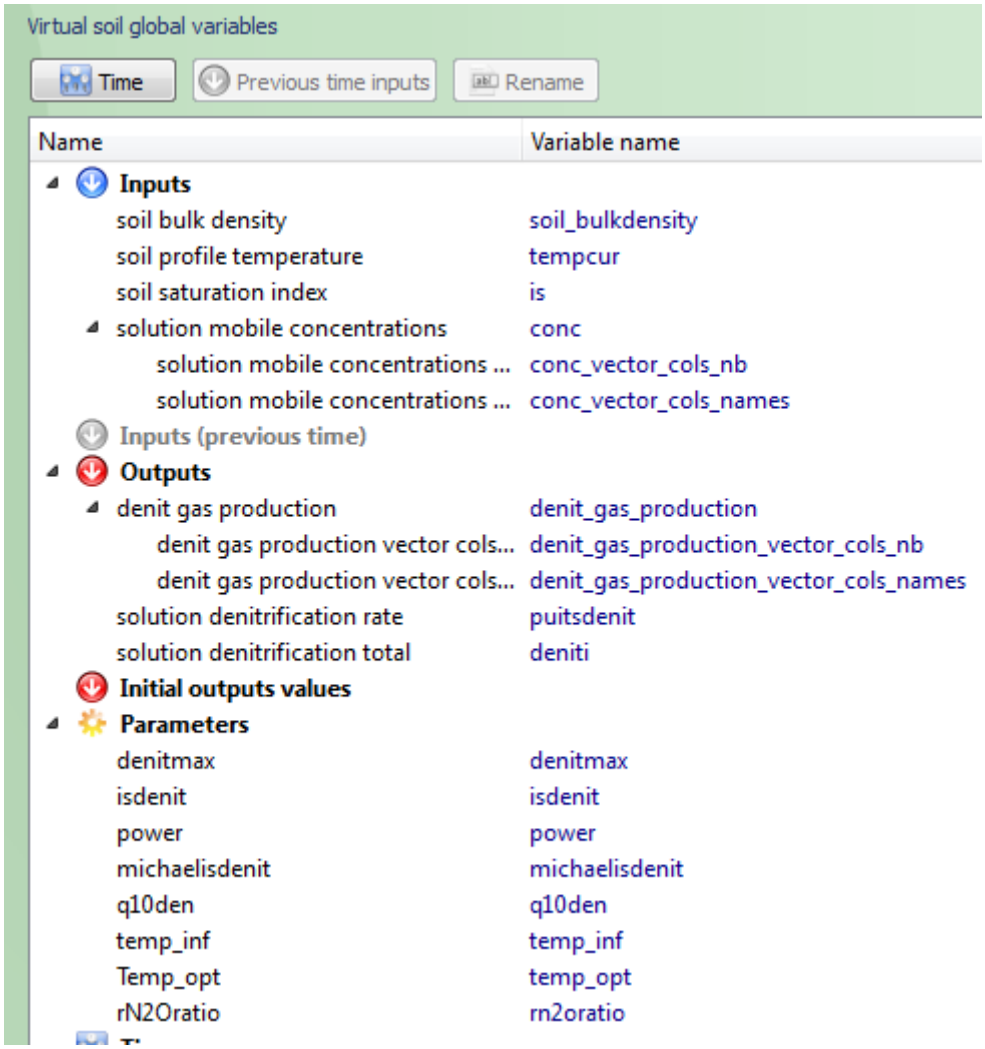


FIGURE 8 – Capture d'écran du composant de plateforme vsoil_modules sur l'onglet de l'éditeur de code. Il est présenté la partie gauche affichant l'ensemble des variables.

L'algorithme de recherche est ainsi simplifié et ressemblera à ce qui est proposé dans le listing 3.

```
Entree : Entier : nombreDeColonnes
Entree : Tableau de chaines de caracteres : nomDesColonnes
Entree : Chaine de caracteres : colonneCherchee

Sortie : Entier : position
Sortie : Booleen : trouve?

Variable : Entier : indice, initialise a 0

DebutFonction
 position <- -1
 trouve? <- faux

 Pour indice allant de 0 A nombreDeColonnes par pas de 1 faire
        Si nomDesColonnes[ indice ] === colonneCherchee
        alors
                position <- indice
                trouve? <- vrai
        finsi
```

```
 finpour

 retourner position
 retourner trouve?
FinFonction
```

Listing 3 – Algorithme de recherche du numéro de colonne

Cela est traduit en *Fortran* par le listing suivant :

```
found = .false.
i = 1
do while((found.eqv..false.).and.(i.le.conc_vector_cols_nb))
   if(trim(conc_vector_cols_names(i)).eq.'NO3-') then
      found = .true.
      ino3=i
      exit
   else
      i = i+1
   end if
end do
```

Listing 4 – Implémentation en Fortran de la recherche d'une colonne

Nous remarquons immédiatement qu'il faut déclarer des variables, cette fois-ci, locales pour pouvoir exécuter ce code, ce que nous avons vu dans le précédent tutoriel. Il faudra donc écrire dans la partie « Declare Local Variables » de cette section les ligne suivantes :

```
integer i
logical found
```

Listing 5 – Déclaration des variables locales de la section d'initialisation

Il reste maintenant à signaler à la plateforme ce qu'elle doit faire si la colonne n'a pas été trouvée dans l'entrée et à initialiser les valeurs des sorties.

Indiquer à la plateforme qu'une étape essentielle a échoué. La recherche de la colonne contenant les concentrations en nitrate est une étape essentielle. Ne pas avoir ces concentrations rend caduque tous les calculs qui suivront. Il faut indiquer à la plateforme que le module n'est pas capable de poursuivre son travail et que, par conséquence, il ne pourra pas fournir la production de N_2 et de N_2O comme annoncé.

Cette information se fait simplement en utilisant la variable *Subroutine return status* que l'on trouve dans la section *utility* du panneau de gauche de l'onglet « Code Editor ».

On test si la recherche a été fructueuse. Si ce n'est pas le cas on place cette variable dans l'état faux. En faisant un glisser–déposer du nom de la variable depuis le panneau gauche jusqu'au panneau droit, la variable s'écrit automatiquement dans l'état d'échec.

On peut aussi informer l'utilisateur de la raison de l'échec et afficher un message. C'est ce qui est fait dans ce module.

```
if(found.eqv..false.) then
   !print *,' Denit_simple module.'
   !print *,' NO3 could not be found in the array of solutes '
   !print *,' Please check the species selected for the solute transport and balance
    module'
```

```
write ( vsoil_fortranlogger_message , * ) 'NO3 could not be found in the array of
 solutes. Please check the species selected for the solute transport and balance module
 .'
call vsoil_fortranlogger_fatal()
subroutine_succeeded = .false.
```

Listing 6 – Arrêt du module sur l'échec d'une étape essentielle.

Initialiser les sorties définies sur le profil de sol. Nous allons, pour terminer la partie initialisation, affecter une valeur de départ à toutes nos sorties. Par défaut, nous voulons que les valeurs soient nulles au début de l'exécution du module.

Pour la sortie qui est de type *scalar* et dont la localisation est *none*, il suffit de lui affecter directement la valeur 0.

Pour la sortie qui est de type *scalar* et dont la localisation est *profile*, il faut lui affecter la valeur 0 pour chaque point de calcul du maillage.

Pour la sortie qui est de type *vector* et dont la localisation est *profile*, il faut lui affecter la valeur 0 pour chaque point de calcul du maillage et pour chaque colonne du vecteur.

C'est ce qui est réalisé en *Fortran* dans cet extrait de code.

```
in2 = 1

deniti = 0.d0
```

Listing 7 – Initialisation des sorties

On remarquera l'utilisation de *vsoil_grid_n* qui donne le nombre de points du maillage du profil de sol et de *denit_gas_production_vector_cols_nb* qui donne le nombre de colonnes de notre sortie.

L'ensemble mis bout à bout. La réunion de tous ces éléments donne le listing de code suivant,

```
! Determines the column of NO3 in the array of solutes concentations

found = .false.
i = 1
do while((found.eqv..false.).and.(i.le.conc_vector_cols_nb))
   if(trim(conc_vector_cols_names(i)).eq.'NO3-') then
      found = .true.
      ino3=i
      exit
   else
      i = i+1
   end if
end do

if(found.eqv..false.) then
   !print *,' Denit_simple module.'
   !print *,' NO3 could not be found in the array of solutes'
   !print *,' Please check the species selected for the solute transport and balance
    module'
```

```
    write ( vsoil_fortranlogger_message , * ) 'NO3 could not be found in the array of
     solutes. Please check the species selected for the solute transport and balance module
     .'
    call vsoil_fortranlogger_fatal()
    subroutine_succeeded = .false.
    return
end if

! Initialization

in2o = 2   ! See the inputs/outputs tab
in2 = 1

deniti = 0.d0
puitsdenit(1:vsoil_grid_size)=0.d0  ! sink term for solute (NO3-)
denit_gas_production(1:vsoil_grid_size , 1: denit_gas_production_vector_cols_nb) = 0.d0
```

Listing 8 – Listing de la section d'initialisation hors déclaration de variables locales

comme représenté sur la capture d'écran suivante.

3.5.5 Compute section

Le programme de simulation va répéter une série de calculs sur l'ensemble des points de calculs du maillage. Nous allons calculer la pondération de la dénitrification due à la température (loi du Q10), puis celle due à l'humidité (loi abiotique), le rapport de Michaelis et enfin donner le taux de dénitrification et les vitesses de production des différents gaz azoté.

Lorsque les calculs sur l'ensemble des points du maillage auront été effectués, nous intégrerons pour avoir le taux de dénitrification total.

Dans ces calculs, nous utilisons des paramètres définissant le taux maximum de dénitrification, le taux de saturation à partir duquel la dénitrification commence etc. Ces paramètres sont définis pour des compositions de sol particulières qui constituent les couches de notre profil de sol. Avant de lancer un calcul sur un point du maillage, il convient de déterminer à quelle couche du sol il appartient. C'est le rôle de la variable *vsoil_grid_nodes_horizon_index* que nous avons vu au paragraphe 3.5.1 page 12.

Déclaration de variables locales. Pour faciliter la rédaction du programme de simulation, nous créerons trois fonctions internes qui correspondront aux différentes lois utilisées dans ce module. De même nous allons créer des variables locales qui nous permettrons de stocker des étapes intermédiaires des calculs.

La partie « Déclare Local Variables » sera la suivante :

```
double precision coef,ratio_michaelis
integer i, ilay
```

Listing 9 – Variables locales de la « Compute Section »

Les deux premières variables permettrons de stocker le résultat du calcul de la pondération de la dénitrification due à l'humidité et à la température ainsi que le rapport de Michaelis du à la concentration en nitrate.

La variable ilay stockera le numéro de couche correspondant au point de calcul considéré et la dernière variable servira dans la réalisation d'une boucle.

Initialization section

Hide local variables

```
1 integer i
2 logical found
3
```

```
1 ! Determines the column of NO3 in the array of solutes concentations
2
3   found = .false.
4   i = 1
5   do while((found.eqv..false.).and.(i.le.conc_vector_cols_nb))
6      if(trim(conc_vector_cols_names(i)).eq.'NO3-') then
7         found = .true.
8         ino3=i
9         exit
10     else
11        i = i+1
12     end if
13  end do
14
15 if(found.eqv..false.) then
16    print *,' Denit_simple module.'
17    print *,' NO3 could not be found in the array of solutes'
18    print *,' Please check the species selected for the solute transport and balance module'
19    subroutine_succeeded = .false.
20    return
21 end if
22
23 ! Initialization
24
25 in2o = 2    ! See the inputs/outputs tab
26 in2 = 1
27
28 puitsdenit(1:vsoil_grid_n)=0.d0  ! sink term for solute (NO3-)
29
30 deniti = 0.d0
31 denit_gas_production(1:vsoil_grid_n , 1: denit_gas_production_vector_cols_nb) = 0.d0
32
```

FIGURE 9 – Capture d'écran du composant de plateforme vsoil_modules sur l'onglet de l'éditeur de code. Il est présenté la partie droite représentant la section d'initialisation des variables avec la présentation des variables locales.

Corps principal du programme. Comme annoncé, nous effectuons une boucle sur l'ensemble des points de calcul du maillage. L'étape suivante est de déterminer dans quelle couche on se trouve ce qui est réalisé par l'instruction ilay = vsoil_grid_nodes_horizon_index(i). Puis on calcule l'ensemble des quantités intervenant dans la dénitrification. Enfin, après la boucle, nous intégrons sur l'ensemble du profil pour obtenir la dénitrification totale.

Nous remarquerons la présence d'une détermination d'un maximum dans l'instruction puitsdenit(i)=**max**(ratio_michaelis*coef,0.d0) alors qu'il semble qu'aucune des grandeurs ne puisse être négative. Il faut se souvenir que nous parlons d'une simulation numérique : lors de calculs avec des valeurs proches de zéro, nous pouvons obtenir des résultats très légèrement négatifs à cause des approximations numériques. Pour éviter des données non significatives, nous préférons nous assurer que les très petites valeurs ne sont pas négatives et dans ce cas sont traitées comme nulles.

L'ensemble du code du programme principal est reproduit ci-dessous suivi d'une capture d'écran associée.

```
do i=1,vsoil_grid_size

    ilay  = vsoil_grid_nodes_horizon_index(i)

    coef=fdenit(tempcur(i),is(i),soil_bulkdensity(i), isdenit(ilay),power(ilay),denitmax(
  ilay))

    ratio_michaelis=conc(i,ino3)/(conc(i,ino3)+michaelisdenit(ilay))

    puitsdenit(i)=max(ratio_michaelis*coef,0.d0)

    denit_gas_production(i,in2o) = puitsdenit(i)*rn2oratio*1.d+03/62.d0      ! Division by
  62 to convert grams into moles
    denit_gas_production(i,in2) = puitsdenit(i)*(1-rn2oratio)*1.d+03/62.d0
end do

call vsoil_generic_integration(puitsdenit,vsoil_grid_nodes_delta,vsoil_grid_size,deniti)

deniti=deniti*surfplot
```

Listing 10 – Compute Section

Compute section

Hide local variables

```
1 double precision coef,ratio_michaelis
2 integer i,ilay
3
```

```
1   do i=1,vsoil_grid_n
2
3      ilay = vsoil_grid_numzo(i)
4
5      coef=fdenit(tempcur(i),is(i),soil_bulkdensity(i),isdenit(ilay),power(ilay),denitmax(ilay))
6
7      ratio_michaelis=conc(i,ino3)/(conc(i,ino3)+michaelisdenit(ilay))
8
9      puitsdenit(i)=max(ratio_michaelis*coef,0.d0)
10
11     denit_gas_production(i,in2o) = puitsdenit(i)*rn2oratio*1.d+03/62.d0      ! Division by 62 to c
12     denit_gas_production(i,in2) = puitsdenit(i)*(1-rn2oratio)*1.d+03/62.d0
13  end do
14
15  call vsoil_generic_integration(puitsdenit,vsoil_grid_delta,vsoil_grid_n,deniti)
16
17  deniti=deniti*surfplot
```

FIGURE 10 – Capture d'écran du composant de plateforme vsoil_modules sur l'onglet « Code Editor ». Il est représenté la partie de droite sur la partie « Compute Section » avec l'affichage des variables locales.

3.5.6 Internal functions section

Les fonctions internes ne présentent pas de difficultés particulières. Elles modélisent les différentes lois présentées dans la section Présentation scientifique et qui sont représentés sur les figures 1, 2 et 3.

Nous donnons donc seulement le listing correspondant à ces fonctions et une capture d'écran partielle pour situer l'emplacement de ce code.

```
!-----------------------------------------------------------------
!       FUNCTION FDENIT
!-----------------------------------------------------------------
!       Purpose: To calculate  denitrification  rate and the weighting factor
!                function of humidity and temperature.
!
!-----------------------------------------------------------------

      double precision function fdenit(temp,is,densit,isdenit,power,denitmax)

      double precision,intent(in) :: temp     ! Temperature(K)
      double precision,intent(in) :: densit   ! Bulk density
      double precision,intent(in) :: is       ! Saturation index
      double precision,intent(in) :: isdenit  ! Threshold for saturation index
      double precision,intent(in) :: denitmax! Max denitrification rate
      double precision,intent(in) :: power    ! exponent of the power law for water
    dependence

      double precision w_temp,w_theta

      if((temp.le.temp_inf).or.(is.lt.isdenit)) then
            fdenit=0.d0
      else
        w_temp=vsoil_q10gen(temp,temp_inf,temp_opt)
        w_theta=abiot(is,isdenit,power)
        fdenit=denitmax*w_theta*w_temp
        fdenit=fdenit*densit        ! Convert from kg/kg/s to kg/m3/s
      end if

    end function fdenit

!-----------------------------------------------------------------
!       FUNCTION ABIOT(IS,ISDEN,POWER)
!-----------------------------------------------------------------
!       Purpose: To control the  effect  of  soil  humidity on  biological
!                processes.
!-----------------------------------------------------------------

      double precision function abiot(is,isden,power)

      double precision,intent(in) :: is,isden,power
      double precision xnum,denom
      xnum=max(0.d+00,is-isden)
      denom=1-isden
      abiot=(xnum/denom)**power

    end function abiot

!-----------------------------------------------------------------
```

```
!      FUNCTION Q10GEN(TEMP)
!-----------------------------------------------------------------
!      Purpose: To calculate the response function of a biological process
!               to temperature variations according to a q10 law.
!
!      TEMP     : Temperature
!      TPROCESS : Temperature below which the process is inactive
!      TREF     : Process temperature reference.
!      QFAC     : Rate multiplied by QFAC for a deviation of 10 degrees relative to the
    reference temperature.
!                                 In case of temperature beneath the reference temperate,
    we divide by QFAC.
!
!-----------------------------------------------------------------
!
!!$       double precision function q10gen(temp,temp_inf,temp_opt)
!!$
!!$       double precision ,intent(in) :: temp,temp_inf,temp_opt
!!$       double precision dif
!!$
!!$       if(temp.lt.temp_inf) then
!!$          q10gen=0.d0
!!$       else
!!$          dif=temp-temp_opt
!!$          q10gen=q10den**dif
!!$       end if
!!$
!!$     end function q10gen
```

Listing 11 – Internal functions section

4 Ce que vous avez appris.

Vous avez vu plus en détail les notions de maillage de sol et de couches de sol.

Vous avez appris comment utiliser les données vectorielles lorsque vous les utilisez en sortie et lorsque vous les utilisez en entrée.

Internal functions section

```
!-----------------------------------------------------------------------
!      FUNCTION FDENIT
!-----------------------------------------------------------------------
!      Purpose: To calculate denitrification rate and the weighting factor
!               function of humidity and temperature.
!
!-----------------------------------------------------------------------

      double precision function fdenit(temp,is,densit,isdenit,power,denitmax)

      double precision,intent(in) :: temp    ! Temperature(K)
      double precision,intent(in) :: densit  ! Bulk density
      double precision,intent(in) :: is      ! Saturation index
      double precision,intent(in) :: isdenit ! Threshold for saturation index
      double precision,intent(in) :: denitmax! Max denitrification rate
      double precision,intent(in) :: power   ! exponent of the power law for water dependence

      double precision w_temp,w_theta

      if((temp.le.temp_inf).or.(is.lt.isdenit)) then
            fdenit=0.d0
      else
        w_temp=q10gen(temp,temp_inf,temp_opt)
        w_theta=abiot(is,isdenit,power)
        fdenit=denitmax*w_theta*w_temp
        fdenit=fdenit*densit      ! Convertit de kg/kg/s à kg/m3/s
      end if

    end function fdenit

!-----------------------------------------------------------------------
!     FUNCTION ABIOT(IS,ISDEN,POWER)
!-----------------------------------------------------------------------
!     Purpose: To control the effect of soil humidity on biological
!              processes.
!-----------------------------------------------------------------------

      double precision function abiot(is,isden,power)

      double precision,intent(in) :: is,isden,power
      double precision xnum,denom
      xnum=max(0.d+00,is-isden)
      denom=1-isden
      abiot=(xnum/denom)**power

    end function abiot

!-----------------------------------------------------------------------
!     FUNCTION Q10GEN(TEMP)
!-----------------------------------------------------------------------
```

FIGURE 11 – Capture d'écran du composant de plateforme vsoil_modules sur l'onglet « Code Editor ». Il est représenté la partie de droite sur un morceau de la partie « Internal functions section ».