Spatialisation du modèle agronomique Stics à l'aide de Open-PALM Parasol

Florent Duchaine[1], Florence Habets[2*], Marie Launay[3], Thierry Morel[1], Wilfried Queyrel[2], Dominique Ripoche[3].

[1]Cerfacs, Toulouse, France
[2]UMR Sisyphe/ Mines-Paristech, Paris, France
[3]INRA-Agroclim, Avignon, France
*Auteur correspondant : florence.habets@mines-paristech.fr

1 Objectifs

L'objectif du travail est d'être capable d'appliquer le modèle agronomique Stics actuellement conçu pour être utilisé sur une parcelle sur un bassin versant sur lequel peut se trouver plusieurs centaines de parcelles.

Ce travail avait déjà été mené avec une version précédente de Stics (version 3.0), mais, les modifications apportées aux codes étaient nombreuses, ce qui a limité la capacité évolutive du système. Ainsi, il devenait nécessaire de reprendre ce travail de façon plus évolutive afin de pouvoir bénéficier de l'ensemble des avancées de la nouvelle version de Stics (modulo-Stics ou Stics version 7.0).

De plus, contrairement à ce qui avait été fait avec l'ancienne version, on souhaite pouvoir gérer un couplage étroit entre le modèle agronomique spatialisé et le modèle hydrologique, notamment pour pouvoir gérer de façon interactive les besoins en irrigation et l'impact des prélèvements sur le bassin. Pour cela, il est nécessaire de permettre une interaction au pas de temps journalier entre l'ensemble des parcelles Stics et le modèle hydrologique,

Pour pouvoir répondre à ces deux contraintes, on se base sur le coupleur PALM du Cerfacs (http://www.cerfacs.fr/globc/PALM_WEB/), maintenant devenu open-source (ce qui garanti un libre accès au coupleur) et renommé OpenPALM. Au sein de OpenPALM, le module Parasol est développé pour permettre l'appel simultané en parallèle d'un même programme (ici Stics).

Ce rapport présente donc le principe du module Parasol, ainsi que son application pour la spatialisation de Stics.

2 Présentation du contexte

Le modèle agronomique Stics (Brisson et al., 1998, http://www.avignon.inra.fr/agroclim_stics) permet de simuler la croissance des cultures en prenant en compte l'ensemble des pratiques agricoles. Les nombreux travaux menés par les développeurs/utilisateurs de Stics lui permettent d'aborder l'impact de ces pratiques à la fois sur les rendements et sur l'environnement.

Ainsi, Stics a été utilisé sur le bassin de la Seine, en utilisant les bases de données agricoles développées par l'INRA SAD (Mignolet et al., 2007) pour estimer des flux de nitrates. Ces flux ont ensuite été utilisés pour estimer la contamination des nappes par les nitrates avec le modèle hydrologique Modcou-Newsam (Viennot et al., 2009), et plus récemment, avec la nouvelle version de Modcou développée dans le cadre d'Eau-dyssée (Philippe et al., 2012).

Un couplage fin entre Stics et Modcou dans Eau-dyssée a été réalisé pour la gestion journalière de l'irrigation, avec une application de Stics limitée à une parcelle représentative (Habets et al., 2011).

Ce couplage nécessite une interaction au pas de temps journalier entre les deux modèles (figure 1): en haut, l'ancienne approche consistait à faire tourner les parcelles Stics sur l'ensemble de la période en séquentiel, et ensuite, de transférer les flux au modèle hydrologique. La nouvelle approche consiste à faire tourner chaque jour l'ensemble des parcelles Stics pour un jour, et de transférer ensuite les informations au modèle hydrologique. Ce couplage, réalisé avec OpenPALM-Parasol permet d'interagir sur les simulations agronomiques en fonction des conditions hydrologiques. Il est facilité par le fait que les calculs journaliers de Stics sont réalisés au sein d'un grand module nommé Stics_Jour.

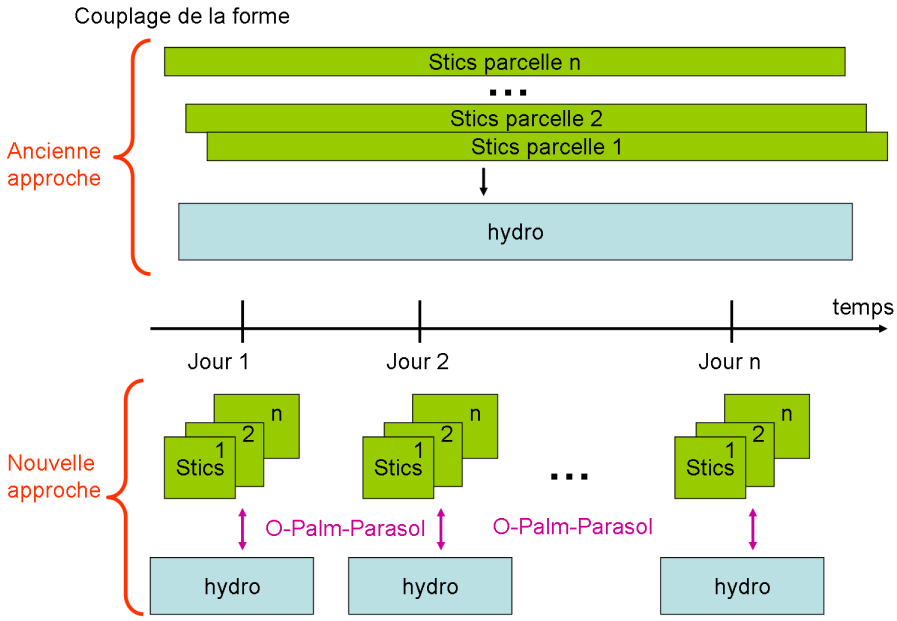


Figure 1 : Principe du couplage entre les modèles agronomique et hydrologique

Figure 2 : visualisation des échanges via Prépalm. Les points en haut des modules correspondent à des variables en entrée, et ceux en bas à des variables en sortie. Les connexions entre modules sont indiquées par des courbes

3 OpenPALM-Parasol

3.1 Présentation

PALM_PARASOL est un utilitaire permettant de lancer automatiquement en parallèle un nombre quelconque d'instances d'un même code de calcul. Son but n'est pas de paralléliser automatiquement un code de calcul via une décomposition de domaine, son intérêt est plutôt d'exécuter n instances d'un même code avec des entrées (et par voie de conséquence des sorties) différentes. Il a été développé pour répondre au besoin spécifique d'une application de couplage faisant intervenir le modèle agronomique STICS dans la plateforme intégrée EauDyssée. Dans ce projet, l'idée est de "spatialiser" le code STICS en le lançant sur un ensemble de parcelles constituant un bassin versant. PALM_PARASOL est fortement contraint par l'utilisation du coupleur OpenPALM (CERFACS/ONERA) car il s'appuie sur lui pour créer automatiquement du code source constituant une unité PALM. Bien que développé pour paralléliser les sols (d'où son nom) PALM_PARASOL est applicable à toute application nécessitant de lancer en parallèle un nombre conséquent d'instances du même code, on peut donc y trouver un intérêt sur les problèmes basés sur des plans d'expériences ou certaines méthodes d'assimilation de données de type Blue.

3.2 Principe

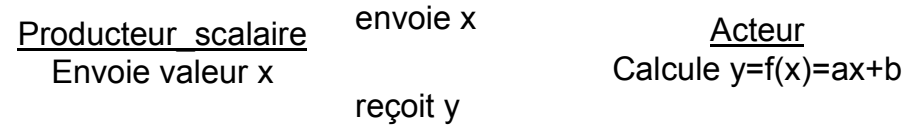
Lancer en parallèle plusieurs instances d'un programme n'est pas difficile en soi. Même si le code n'a jamais vu de près ou de loin le calcul parallèle, il suffit d'un travail minime pour l'adapter à la librairie MPI qui sait très bien faire cela. Le but de PARASOL consiste à faire travailler ces différentes instances de code sur des données différentes et de collecter les résultats dans le bon ordre. Au passage on cherchera un cadre générique, c'est à dire facilement paramétrable quant au nombre de processeurs, pour équilibrer les calculs sur des machines différentes dont les ressources en processeurs sont très différentes. L'utilisation d'OpenPALM offre cette souplesse.

Dans un premier temps, notons **n** le nombre d'instances de code à faire tourner, dans l'exemple de STICS, n est le nombre de parcelles du bassin à traiter, on appellera n par la suite "taille du problème". On se propose

de traiter ce problème en s'aidant grandement des paradigmes mis à disposition par le coupleur dynamique de codes parallèles OpenPALM. L'idée est d'externaliser "l'alimentation" des codes en données via des communications par mémoire "à la PALM" et de collecter les résultats par le même mécanisme. Pour l'application, données (IN) et résultats (OUT) seront stockés quelque part en mémoire dans des tableaux multidimensionnels dont l'une des dimensions est la taille du problème. Remarquons que le code de calcul à "PARASOLiser" ne voit jamais la taille n du problème, c'est un paramètre de l'application couplée et non du code, ce qui permet d'être un minimum intrusif dans ce code.

Pour fixer les idées prenons un exemple simpliste où le code résout l'équation du 1er degré ax+b=0 (cas a de la figure 2).
Ce qu'on cherche, c'est, sans modifier ce code, résoudre en parallèle n équations du premier degré : Y(i)=aX(i) +b, pour i allant de 1 à n. Pour cela, Parasol génère un sous programme encapsulant la subroutine d'origine acteur qui reste inchangée. Ce sous programme est nommé Master_acteur (cas b de la figure 2). Master_acteur gère deux tableaux d'entrée de taille n pour X et Y, et intègre également une dimension pour les variables a et b décrivant le système, afin de permettre d'adapter ces paramètres à chaque cas.
Pour plus de souplesse, et parce qu'on ne dispose pas toujours de n processeurs mais plutôt de **m** (m<n) Parasol traite non pas n opérations en parallèle mais seulement m, quitte à boucler n/m fois pour lancer tous les calculs.

a) Programme d'origine



b) Programme sous parasol

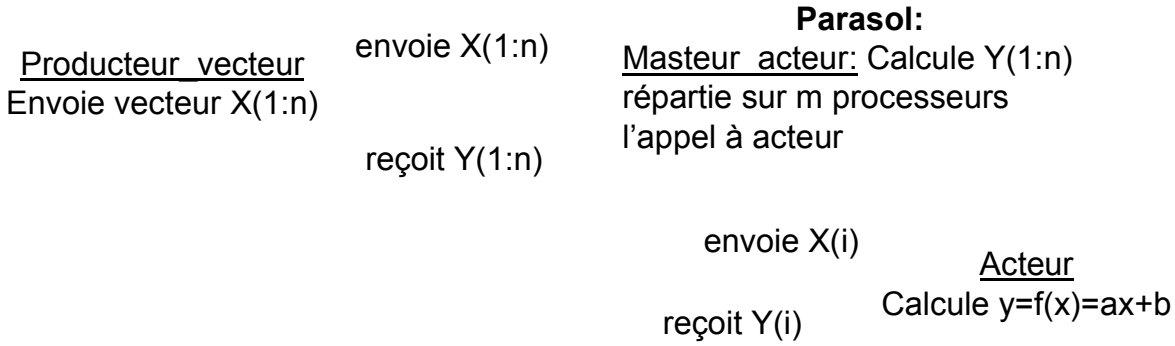


Figure 2 : Principe de l'usage de Parasol

PALM_PARASOL est basé sur un mode maitre/ouvrier disponible dans la norme MPI_2. Un programme "master" va recevoir sous forme de PALM_Get les tableaux de taille n, puis dans une boucle de taille n/m il va lancer dynamiquement (commande MPI_Comm_spawn) m instances du code auxquelles le master va envoyer les données dispatchées, le programme Master se charge également de collecter les résultats de tous les modèles lancés pour finalement les regrouper dans un tableau de taille n. Au total, au lancera donc n instances du code. Bien qu'il soit tout à fait possible d'optimiser les lancements en faisant une boucle interne dans le code, un peu comme on peut le faire dans PALM en ajoutant un block autour d'une boucle, cette solution n'a pas été retenue car elle suppose que le code soit adapté à ce mode de fonctionnement. Il faudrait en effet être certain que tous les fichiers ouverts par le code soient fermés et que toutes les variables allouées dynamiquement soient désallouées, c'est en général loin d'être le cas pour la majorité des codes de calcul.

La génération du code Master, de la mécanique MPI pour envoyer/recevoir les tableaux, et l'encapsulation du code à lancer est faite automatiquement et une fois pour toute (si l'interface du code ne change pas) par PALM_PARASOL. La taille n du problème et le nombre de processeurs disponibles deviennent paramétrables dans l'interface graphique PrePALM car ces données apparaissent sous forme d'entrées de l'unité OpenPALM générée. L'utilisateur n'a besoin de remplir qu'un descriptif des entrées et sorties du

codes, de remplacer le programme principal par une subroutine Fortran ou une fonction C qui fait remonter dans sa liste d'appel (et en cohérence avec le descriptif) les paramètres IN, OUT ou INOUT que l'on veut faire varier dans le code.
Cette approche permet ainsi de spatialiser un code sans intervention très intrusive, ce qui favorise l'intégration de l'évolution du code.

4 Utilisation de Parasol pour la spatialisation de Stics

4.1 Principe

Pour spatialiser le modèle agronomique Stics, tout en permettant un échange d'information au pas de temps journalier entre l'ensemble des parcelles agricoles et le modèle agronomique, nous avons donc séparé le module journalier Stics_Jour du module d'initialisation et de gestion des simulations que nous avons nommé Stics_Gere. Dans ce dernier module, une dimension spatiale a été rajoutée à l'ensemble des variables Stics, afin de travailler sur un domaine spatial cohérent avec une application hydrologique. Par contre, le module STICS_JOUR est lui très peu modifié. Le principe de cette spatialisation est présenté figure 3.

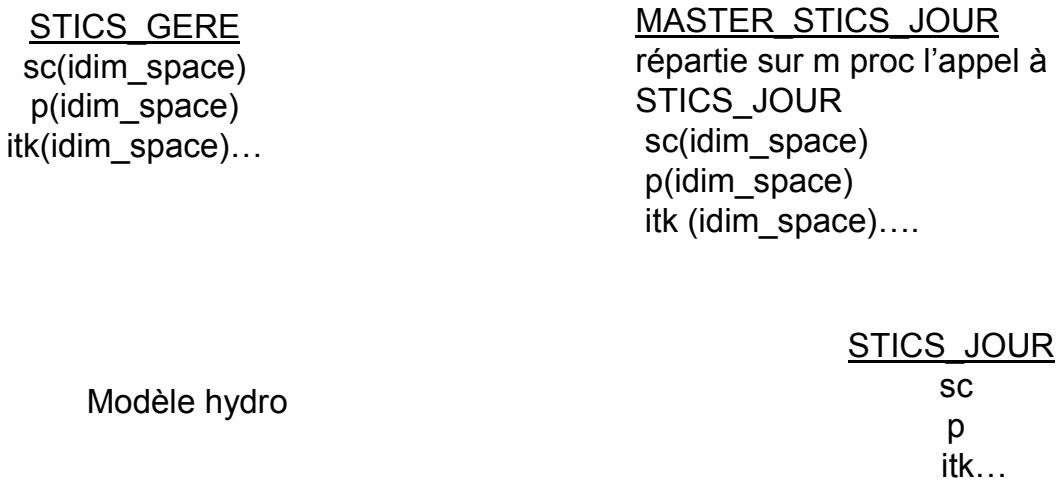


Figure 3 : Principe de la spatialisation de Stics

4.2 Ajout d'une dimension spatiale

Afin d'échanger les informations entre Stics_Gere et Master_Stics_Jour sur l'ensemble sur l'ensemble des parcelles, il a fallu rajouter une dimension spatiale aux variables Stics. Or, les variables Stics sont contenues dans des structures, réunissant un grand nombre de variables. Ainsi, l'ensemble des variables décrivant les paramètres et les variables prognostiques de la plante sont contenus dans la structure plante noté « p », ceux décrivant les itinéraires techniques sont inclus dans la structure « itk », ceux décrivant les variables génériques dans « stics_commun » ou « sc ». Huit structures sont ainsi gérées par Stics, auquel il a donc fallu rajouter une dimension dans les subroutines d'initialisation de la partie Stics gère. Seules deux subroutines de la partie Stics_Gere sont affectées par ces modifications à ce jour (cf annexe).

4.3 Insertion des communications par OpenPALM

La gestion des communications se fait assez simplement dans OpenPALM, puisque seules les 8 structures sont transférées. Les structures sont définies dans OpenPALM comme un tableau d'entier dont la dimension correspond à celle de la structure (cf figure 4).

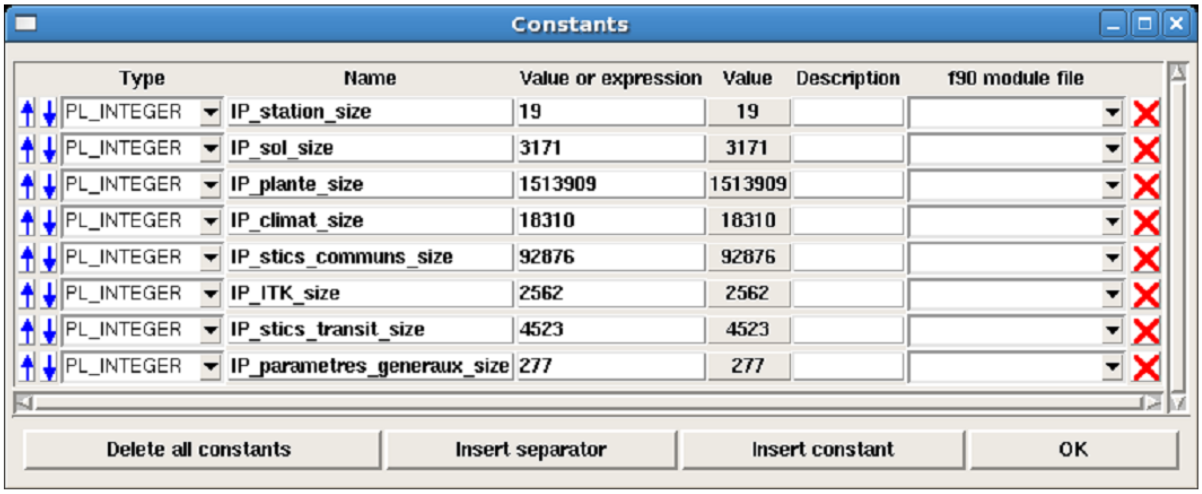


Figure 4 : définition dans OpenPALM des tailles des tableaux contenant les structures Stics

Une des principales modifications introduites par OpenPALM consiste à ne plus gérer l'appel direct à la subroutine Stics, mais, à transférer les données via des Palm_put/Palm_get vers Stics_Jour, comme cela est présenté figure 5. L'appel de la subroutine Stics_Jour est en commentaire, car cet appel est maintenant effectué par master_stics_jour. La subroutine Stics_Jour n'est ainsi pas modifiée par l'introduction de OpenPALM-Parasol.

➔On échange les données sur l'ensemble des parcelles par des palm_put/palm_get

➔L'appel à Stics_Jour se fait par Palm

Figure 5 : Modification de la subroutine Stics_Boucle_année pour échanger les variables de Stics via OpenPALM.

4.4 Modification des entrées-sorties

Afin de gérer les entrées et sorties sur plusieurs parcelles, un minimum de modifications ont du être apportées au code. A ce stade, les formats des fichiers d'entrée et de sortie n'ont pas du tout été changés, et des modifications supplémentaires seront à prévoir pour lire les paramètres de simulation compatible à partir des bases de données spatialisées de l'INRA SAD.

Par contre, une légère modification a été nécessaire pour pouvoir écrire un fichier de sortie pour chaque parcelle. Ainsi, le nom du fichier a été modifié pour introduire le numéro de la parcelle correspondante. Afin de réaliser cela sans rajouter une nouvelle variable à une structure type (ce qui aurait nécessité de recalculer la dimension du type et de modifier les constantes dans PrePALM), la variable sc%sys a été réaffectée (cette variable semblait inutilisée). Les subroutines modifiées pour cela sont indiquées dans l'annexe.

De nouveau, il convient de noter que la structure des fichiers de sorties n'est pas optimale pour une simulation spatialisée, et que des fichiers de sorties contenant les informations nécessaires en 2D seraient plus performants en termes d'accès ainsi qu'en temps de calcul et en place mémoire.

4.5 Structure de l'application visualisée par PrePALM

La figure 6 présente l'application Stics spatialisée par Parasol telle que visualisée par l'outil PrePALM.

Le nombre de processeurs et le nombre de parcelles doivent être spécifiés avant la compilation.

La branche master_stics_jour inclut une boucle temporelle, puisque la subroutine stics_jour est appelée au pas de temps journalier.

Figure 6 : Visualisation par PrePALM de l'application Stics spatialisée

Cette structure a permis de faire tourner 350 parcelles sur une année et sur 8 processeurs en 36 minutes. (donner un mot sur les performances sans Parasol ?) Cependant, le nombre de parcelle est limité par la place mémoire. Des modifications supplémentaires ont été réalisées dans Stics pour limiter cette place mémoire.

5 Optimisation de l'application Stics spatialisée

Le modèle Stics ayant été conçu pour fonctionner sur une seule parcelle, il n'a pas été optimisé pour limiter sa taille mémoire. Ainsi, Stics gère de nombreux tableaux, dont les dimensions sont définies en fonction par exemple d'un nombre de jours fixé par défaut à 730 et d'un nombre de couches de sol fixé à 1000. L'échange entre les processeurs étant limité à 2 giga octets, la taille de ces tableaux dans les structures Stics limite le nombre de parcelle que l'on peut simuler.

Afin de réduire la taille de ces structures, nous avons modifié quelques subroutines de Stics pour imposer

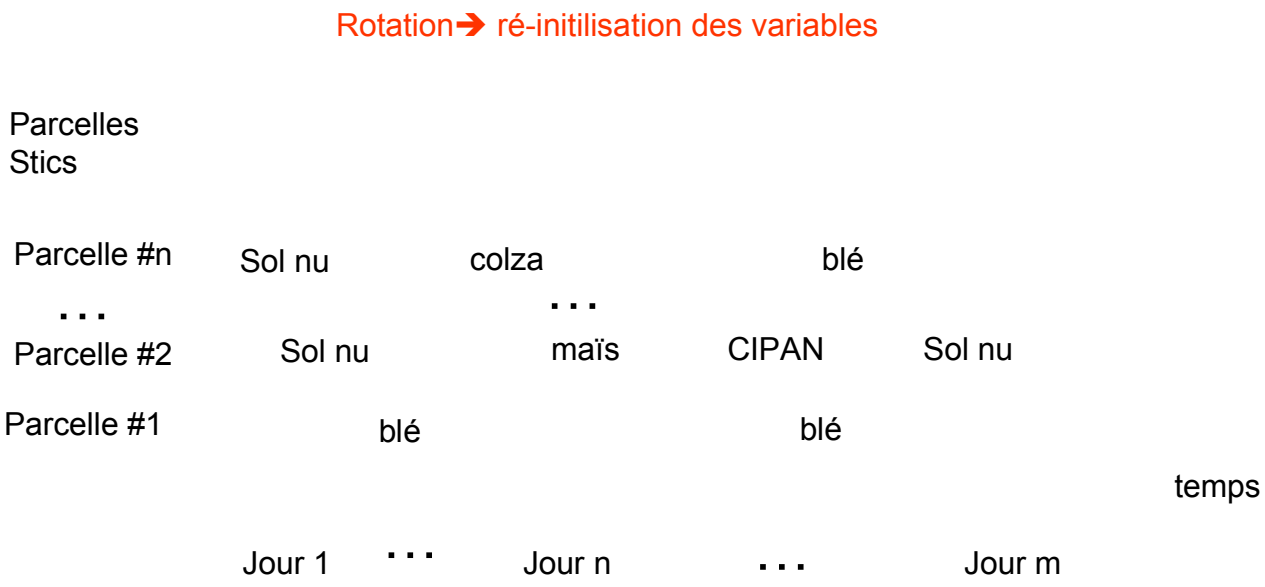
une profondeur de sol plus réaliste. Pour cela, on introduit la variable IPROF_SOL, qui est définit dans la subroutine Stics_user_param. Les cultures existantes sur le bassin de la Seine ont une profondeur de sol de 200cm, ce qui permet de réduire fortement la taille des tableaux. Ainsi, une simulation de 1500 parcelles a pu être réalisée en 1h30 sur un an.

Des optimisations supplémentaires de la place mémoire des structures de Stics devront être faites afin de traiter un plus grand nombre de parcelles, l'application sur la Seine réunissant un peu plus de 9000 parcelles.

6 Conclusion et prospectives

Les travaux menés jusqu'ici ont permis de spatialiser la dernière version de Stics sur plusieurs processeurs, en permettant un échange des valeurs au pas de temps journalier permettant un couplage interactif avec un modèle hydrologique, tout en conservant une efficience de calcul.

Les travaux vont se poursuivre cette année afin d'étendre le nombre de parcelles, avec l'objectif d'atteindre au moins 10 000 parcelles afin de pouvoir simuler le bassin de la Seine. Pour cela, la réduction des tailles des tableaux de Stics sont nécessaires, ce qui pourrait se réaliser en limitant la taille des tableaux temporels, définis par défaut sur 2 ans, alors que la gestion du code se fait maintenant au pas de temps journalier. Une meilleure compréhension de la gestion des tableaux temporels est de toute façon nécessaire pour être capable de gérer les rotations de culture qui se déroulent à différent moment selon les parcelles (figure 7), et nécessiteront donc une ré-initialisation de certaines variables.



En parallèle, la limite actuelle de 1600 parcelles étant suffisante pour commencer les premières applications sur le bassin versant de l'Orgeval, on pourra tester l'application de la dernière version de Stics sur ce bassin. Cela nécessitera de développer des liens entre les bases de données de l'INRA SAD (Nicola et Schott , 2011) décrivant les pratiques culturales du bassin et Stics-parasolisé, en lien avec les nouveaux formats des fichiers décrivant les plantes et les sols de Stics v7.

7 Références

Brisson, N., Mary, B., Ripoche, D., Jeuffroy, M. H., Ruget, F., Nicoullaud, B., Gate, P., Devienne-Barret, F., Antonioletti, R., Durr, C., Richard, G., Beaudoin, N., Recous, S., Tayot, X., Plenet, D., Cellier, P., Machet, J. M., Meynard, J. M., and Delecolle, R. (1998). STICS: a generic model for the simulation of crops and their water and nitrogen balances. I. Theory and parameterization applied to wheat and corn. Agronomie 18, 311-346

Habets F., N. Flipo , P. Goblet, E. Ledoux, C. Monteil, E. Philippe, W. Queyrel, F. Saleh, O. Souhar, A. Stoul, P. Viennot, C. David, A. Bacchi, H. Blanchoud, E. Moreau-Guigon, M. Launay, D. Ripoche, B. Mary, P.-A. Jayet, E. Martin, T. Morel, J. Tournebize, 2010, Le développement du modèle intégré

des hydrosystèmes Eau-dyssée, rapport du Piren Seine synthèse 2007-2010, 46 p. http://www.sisyphe.upmc.fr/piren/webfm_send/960

Mignolet C., C. Schott, and M. Benoit. Spatial dynamics of farming practices in the Seine basin : Methods for agronomic approaches on a regional scale. Science of the Total Environment, 375(1-3), 2007.

Morel Thierry et Florent Duchaine, 2011, PALM_PARASOL version 1.0.0 Manuel utilisateur, 13p

Nicola L. and C. Schott. Évolution des traitements phytosanitaires en vue de simuler leurs impacts sur la qualité de l'eau : synthèse sur l'Orgeval. 2011, Rapport d'activité du PIREN-Seine 2010, http://www.sisyphe.upmc.fr/piren/webfm_send/961

Philippe Elodie, Florence Habets, Emmanuel Ledoux, Patrick Goblet, Pascal Viennot, Bruno Mary Amélioration du transfert des nitrates dans l'hydrosystème Seine: impact sur l'évolution de la contamination des nappes à l'actuel et sous scénario, Rapport d'activité du Piren-Seine 2012.

Viennot P., E. Ledoux, J.M. Monget, C. Schott, C. Garnier, N. Beaudoin, La pollution du basin de la Seine par les nitrates, plaquette du Piren Seine.43p.

Annexe

Le programme Stics contient 172 subroutines. La spatialisation de Stics par O-palm-Parasol a nécessité des modifications dans 9 d'entre elle. Ces modifications étant légères, cela laisse la possibilité d'intégrer les évolutions du code.

Subroutines modifiées pour ajouter une dimension spatiale :

- Stics_boucle_annees
- Main_stics_space

Subroutines modifiées pour gérer les accès aux fichiers d'entrée et sorties :

- Stics_jour_after . le numéro de la parcelle est inclus dans le nom du fichier de sortie. De plus, le fichier de sortie est ouvert dans la subroutine en mode "append" ce qui assure que le numéro du fichier correspond bien à la parcelle en cours même lorsque l'application est distribuée sur plusieurs processeurs.
- Climat_lecture : Rajout d'une fermeture du fichier climat après lecture, afin de pouvoir utiliser la subroutine pour lire l'information sur un autre parcelle.

Subroutines modifiées pour modifier la taille des tableaux ayant pour dimension la profondeur du sol :

- Rajout de la subroutine stics_user_param, qui contient la valeur de la profondeur du sol.
- insertion de la variable IPROF_SOL pour dimension des tableaux du type stics_commun dans les subroutines Stics.f90, Plante.f90 et Sol.f90.
- insertion de la variable IPROF_SOL pour effectuer les tests et dimension des variables par défaut dans les subroutines croira.f90 et init_sol.f90