Résolution d'équations différentielles spatialisées à l'aide de la quantification de variables et de Cell-DEVS

David Versmisse, Raphaël Duboz et Éric Ramat
Laboratoire d'Informatique du Littoral
Maison de la Recherche Blaise Pascal
50, rue Ferdinand Buisson - BP 719
62228 Calais Cedex
E-Mail: versmisse@lil.univ-littoral.fr
majecstic@lil.univ-littoral.fr

Résumé : Le but de cet article est de démontrer la capacité des méthodes de type quantification de variables et l'extension Cell-DEVS pour la spécification DEVS des automates cellulaires pour la résolution numérique d'équations différentielles aux dérivées partielles. L'intérêt est triple : gagner du temps de calcul sous certaines conditions, obtenir un découpage de l'espace en hypercubes totalement autonomes et développer des modèles dans le cadre formel de multi-modélisation.
Mots-clés : Simulation, équations aux dérivées partielles, Quantized State System, DEVS, Cell-DEVS.

1 INTRODUCTION

Afin de simuler un système complexe couplant plusieurs modèles (équations différentielles, réseau de Petri, agents,...), il est possible d'utiliser le formalisme DEVS qui permet de spécifier chaque modèle dans le même cadre formel. Lorsqu'il s'agit de traiter des problèmes où la division en cellules autonomes est naturelle (jeu de la vie, propagation d'un feu de forêt,...) une extension de DEVS existe : Cell-DEVS [WAI01].
Pour la résolution d'équations différentielles du premier ordre, Kofman propose dans le même esprit deux méthodes événementielles compatibles DEVS : QSS1 et QSS2. Ces méthodes reposent sur la discrétisation des valeurs prises par les fonctions intégrées.
L'objectif de cet article est alors double :

- pouvons-nous étendre le champ d'application de QSS1 ou QSS2 aux équations différentielles aux dérivées partielles, et donc proposer une méthode événementielle compatible DEVS pour la résolution de ces équations ? Cette méthode pourra s'intégrer par la suite au sein du *framework* VLE développé par le laboratoire LIL de l'Université du Littoral [RAM03].
- quels sont les gains de cette approche par rapport aux méthodes classiques ?

Des travaux sur la quantification et les équations différentielles aux dérivées partielles ont déjà été réalisés [MUZ02] mais pas sous cette optique d'une utilisation conjointe de Cell-DEVS et de QSS1/QSS2.

Après un rappel rapide de DEVS et de QSS1, QSS2, nous verrons comment adapter ces méthodes aux équations différentielles aux dérivées partielles et enfin nous traiterons un exemple pour apprécier les apports.

2 PRÉSENTATION SUCCINCTE DU FORMALISME DEVS

Afin de garantir une homogénéité au sein des échanges entre les différents composants d'un système complexe, il est nécessaire de disposer d'un langage commun. Le formalisme DEVS [ZEI00] permet dans une certaine mesure de répondre à cette préoccupation. DEVS définit le "dialogue" entre les différents composants par l'échange d'événements discrets estampillés.

La spécification d'un modèle DEVS se fait via la définition d'une structure :

$M =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, Ta >$ où :

- X est l'ensemble des types d'événements externes et Y est l'ensemble des types d'événements de sortie
- S est l'ensemble des états séquentiels
- δ_{int} (resp. δ_{ext}) est la fonction de transition interne (resp. externe) définissant les changements d'état dus à des événements internes (resp. externes)
- λ est la fonction de sortie
- Ta est la fonction définissant la durée de vie des états

Typiquement, les événements émis en sortie d'un modèle perturbent les modèles qui sont connectés à cette sortie. La perturbation est spécifiée par la fonction de transition externe. La fonction de transition interne est, quant à elle, évaluée si la durée de vie d'un état notée Ta est atteinte. La fonction de sortie est activée lors des transitions internes.

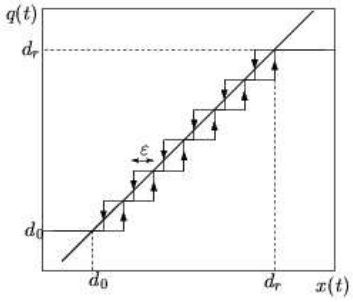


FIG. 1 – Quantification avec hystérésis d'une fonction (tirée de [KOF01])

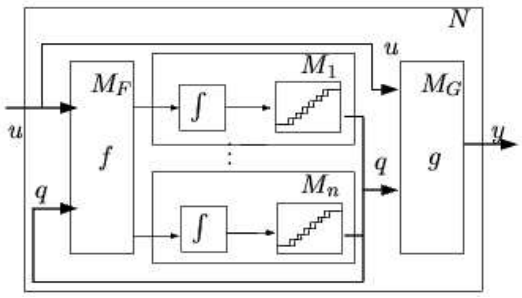


FIG. 2 – Modèle DEVS de QSS1 tiré de [KOF01]

3 INTÉGRATEURS PAR QUANTIFICATION

Parallèlement aux méthodes classiques sont apparues de nouvelles techniques de résolution numérique d'équations différentielles basées sur la quantification des valeurs de sortie plutôt que sur la discrétisation du temps. Ainsi, par exemple, Kofman propose à travers QSS1 [KOF01] et QSS2 [KOF02] deux méthodes de résolution numérique d'équations différentielles du premier ordre. Pour la présentation de ces deux méthodes, considérons le système suivant :

$$\begin{cases} \dot{x}(t) &= f(x(t), u(t)) \\ y(t) &= g(x(t), u(t)) \end{cases} \quad (1)$$

On peut remarquer la dépendance en t de f qui se fait via une fonction u, ce qui est une façon d'intégrer la gestion du temps (ou d'une fonction extérieure) dans le formalisme DEVS. La fonction y est, quant à elle, la sortie désirée. Voyons maintenant ces deux techniques de résolution plus en détail.

3.1 QSS1

3.1.1 Présentation

QSS1 repose sur deux principes :

- la quantification avec hystérésis de la fonction x
- un calcul du pas de temps en cohérence avec $\dot{x}(t)$ et la quantification

La quantification avec hystérésis consiste à remplacer une fonction à valeurs réelles par une fonction constante par morceaux en utilisant une discrétisation de $\mathbb{R}$ (qui peut ne pas être à pas constants) mais en incorporant de l'hystérésis dans le processus. Par exemple : soit ...,d_{-1}, d_0, d_1,...une discrétisation de $\mathbb{R}$ (donc $d_i < d_{i+1}$), supposons que $x(t_0)$ soit légèrement supérieur à d_4 et que x soit croissante alors l'approximation de x sera d_4 et dès que x atteindra d_5, l'approximation sera d_5, etc...
Mais si x est décroissante, alors il faut attendre que x atteigne la valeur $d_4 - \epsilon$ pour que l'approximation devienne d_3. Kofman appelle ϵ la fenêtre d'hystérésis. Tout ceci se résume sur la figure 1 où q est l'approximation de x et où les flèches représentent le sens de variation de x.
Cette façon d'approximer x empêche que d'éventuelles oscillations très rapides de x autour d'une valeur d_i se répercutent par une approximation de x qui ne cesse de basculer entre d_{i-1} et d_i. Cela fournit de bonnes propriétés à la méthode qui est alors assurée (sous certaines conditions) de converger en un temps fini.
Si maintenant la fonction x est à valeurs dans $\mathbb{R}^n$, chaque composante de x est traitée par une machine DEVS différente qui va s'occuper de fournir sa discrétisation au reste du système. Le calcul du pas de temps pour lequel cette approximation reste valable est le suivant : la machine DEVS M_i responsable de x_i reçoit de la part de f $\dot{x}_i(t_0)$. Elle peut ainsi en approximant $x_i(t)$ par $x_i(t_0) + \dot{x}_i(t_0) \times (t - t_0)$ calculer le temps nécessaire à x_i pour atteindre d_{i+1} ou $d_i - \epsilon$.

3.1.2 Le modèle DEVS

Le schéma de la figure 2 représente le système dans sa globalité. La machine DEVS M_F s'occupe de calculer les $\dot{x}_i$ qu'elle envoie aux modèles DEVS M_i. L'approximation est alors calculée et renvoyée à M_F ainsi qu'à M_G pour le calcul de la sortie globale.
Commençons par voir le modèle DEVS de M_i :

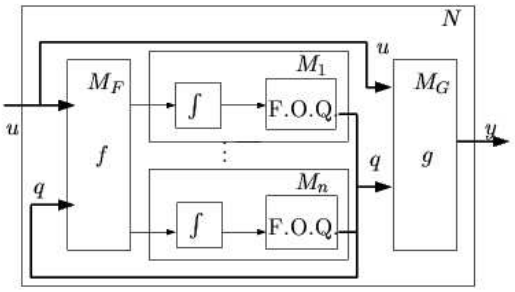
$M_i =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$ où :
$X = Y = \mathbb{R}$
$S = \mathbb{R} \times \mathbb{R} \times \mathbb{Z} \times \mathbb{R}^+ \cup \infty$
$\delta_{int}(x, u, i, \sigma) = (x + \sigma.u, u, i + sgn(u), \sigma')$
$\delta_{ext}(x, u, i, \sigma, e, v) = (x + e.u, v, i, \sigma'')$
$\lambda(x, u, i, \sigma) = d_{i+sgn(u)}$
$ta(x, u, i, \sigma) = \sigma$
avec

$$\sigma' = \begin{cases} \dfrac{d_{i+2} - (x + \sigma.u)}{u} & \text{si } u > 0 \\ \dfrac{x + \sigma.u - d_{i-1} + \epsilon}{|u|} & \text{si } u < 0 \end{cases}$$

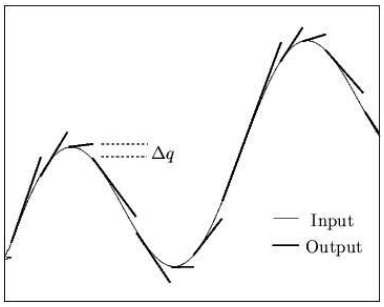
$$\sigma'' = \begin{cases} \dfrac{d_{i+1} - (x + e.u)}{v} & \text{si } v > 0 \\ \dfrac{x + e.u - d_i + \epsilon}{|v|} & \text{si } v < 0 \\ \infty & \text{si } v = 0 \end{cases}$$

Quelques explications :

- i représente via d_i l'approximation courante fournie au reste du système.
- x et u représentent respectivement une approximation de $x_i(t)$ et $\dot{x}_i(t)$. u est fourni par f à travers la variable v.
- $sgn(u)$ est la fonction signe.



(a) Modèle DEVS de QSS2



(b) Quantification d'ordre 1

FIG. 3 – QSS2 tiré de [KOF02]

Après calculs, on constate que les valeurs de σ sont bien en cohérence avec le modèle.
Le modèle de f, M_F se contente de calculer les différents $\dot{x_i}$ et de les envoyer aux M_i. Kofman a rajouté dans son modèle [KOF01] une mémoire des valeurs envoyées pour ne pas déranger les M_i avec des valeurs qui n'ont pas changées.

3.2 QSS2

L'idée tirée de [KOF02] est de garder les concepts présentés dans QSS1, mais en passant à un degré supérieur : on n'approxime plus la sortie par une constante mais par une fonction affine. Pour réaliser cela, f nous fournit la dérivée de x_i et également sa dérivée seconde, le modèle conservant la même architecture (cf figure 3(a)). La machine DEVS M_i réalise donc une approximation de la variable $x_i(t)$ à l'aide de x_i, $\dot{x_i}$ et $\ddot{x_i}$ via la formule de Taylor. Ainsi, sa représentation de $x_i(t)$ est une parabole. Elle calcule ensuite σ de sorte que la différence entre la parabole et son approximation par une fonction affine soit inférieure à un certain Δq. On retrouve de l'hystérésis à travers le calcul de σ. (cf figure 3(b)).

3.2.1 Le modèle DEVS

Commençons par la machine DEVS M_i :

$M_i =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$ où :
$X = \mathbb{R} \times \mathbb{R} \times \{\text{inports}\} = \mathbb{R} \times \mathbb{R} \times \{1\}$
$S = \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R} \times \mathbb{R}^+ \cup \infty$
$Y = \mathbb{R} \times \mathbb{R} \times \{\text{outports}\} = \mathbb{R} \times \mathbb{R} \times \{1\}$
$\delta_{int}(u, m_u, x, q, m_q, \sigma) =$
$\quad (u + m_u.\sigma, m_u, x + u.\sigma + \frac{m_u}{2}\sigma^2,$
$\quad x + u.\sigma + \frac{m_u}{2}\sigma^2, u + m_u.\sigma, \sigma')$
$\delta_{ext}(u, m_u, x, q, m_q, \sigma, e, v, m_v, port) =$
$\quad (v, m_v, x + u.e + \frac{m_u}{2}e^2, q + m_q.e, m_q, \sigma'')$
$\lambda(u, m_u, x, q, m_q, \sigma) =$
$\quad (x + u.\sigma + \frac{m_u}{2}\sigma^2, u + m_u.\sigma, 1)$
$ta(u, m_u, x, q, m_q, \sigma) = \sigma$
avec

$$\sigma' = \begin{cases} \sqrt{\frac{2\Delta q}{|m_u|}} & \text{si } m_u \neq 0 \\ \infty & sinon \end{cases}$$

et σ'' la plus petite solution positive de :
$|x + u.e + \frac{m_u}{2}e^2 + v.\sigma'' + \frac{m_v}{2}\sigma''^2 - (q + m_q.e + m_q\sigma'')| = \Delta q$

Quelques explications :
- x est la valeur courante de x_i.
- u et m_u représentent respectivement $\dot{x_i}$ et $\ddot{x_i}$. Ils sont fournis par f.
- q et m_q forment l'approximation courante donnée par M_i au reste du modèle : $x_i(t) \simeq q + m_q \times t$.

Il est facile de voir que les calculs de σ dans δ_{int} et δ_{ext} engendrent un événement dès que l'écart entre la sortie courante $q + m_q.t$ et la représentation interne de $x_i(t)$ par $x + u.t + \frac{m_u}{2}.t^2$ est supérieur à Δq.
Voyons maintenant le modèle DEVS de la fonction f. Pour simplifier, on peut considérer qu'elle ne dépend que de q variables $z_1, z_2, \ldots, z_q$ en oubliant la séparation entre la partie x et la partie u. Toujours pour simplifier, nous allons modéliser f à travers ses p composantes f_i. On obtient alors le modèle suivant :

$F_i =< X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta >$, où
$X = (\mathbb{R} \times \mathbb{R}) \times \{\text{inports}\}$
$\quad = (\mathbb{R} \times \mathbb{R}) \times \{1, \ldots, q\}$
$S = (\mathbb{R} \times \mathbb{R} \times \mathbb{R})^q \times \mathbb{R}^+ \cup \infty$
$Y = (\mathbb{R} \times \mathbb{R}) \times \{outports\} = (\mathbb{R} \times \mathbb{R}) \times \{1\}$
$\delta_{int}((z_1, m_{z_1}, c_1), \ldots, (z_q, m_{z_q}, c_q), \sigma) =$
$\quad ((z_1, m_{z_1}, c_1), \ldots, (z_q, m_{z_q}, c_q), \infty)$
$\delta_{ext}((z_1, m_{z_1}, c_1), \ldots, (z_q, m_{z_q}, c_q), \sigma,$
$\quad e, v, m_v, port) =$
$\quad ((z'_1, m'_{z_1}, c'_1), \ldots, (z'_q, m'_{z_q}, c'_q), 0)$
$\lambda((z_1, m_{z_1}, c_1), \ldots, (z_q, m_{z_q}, c_q), \sigma) =$
$\quad (f_i(z_1, \ldots, z_q), c_1 m_{z_1} + \cdots + c_q m_{z_q}, 1)$
$ta((z_1, m_{z_1}, c_1), \ldots, (z_q, m_{z_q}, c_q), \sigma) = \sigma$
avec :

$$z'_j = \begin{cases} v & \text{si } j = port \\ z_j + m_{z_j} e & \text{sinon} \end{cases}$$

$$m'_{z_j} = \begin{cases} m_v & \text{si } j = port \\ m_{z_j} & \text{sinon} \end{cases}$$

Méthodes	Nombre de pas
QSS1	301
Euler	150000
Runge-Kutta	90000
Runge-Kutta 4-5	30000
Adams-Bashforth-Moulton	60000
Matlab's ode15s	81

TAB. 1 – Comparaison de différentes méthodes sur le système (2)

$$c'_j = \begin{cases} \dfrac{f_i(z + m_z e) - f_i(z')}{z_j + m_{z_j} e - z'_j} & \text{si } j = port \text{ et si} \\ & z_j + m_{z_j} e - z'_j \neq 0 \\ c_j & \text{sinon} \end{cases}$$

z_j et m_{z_j} sont une sauvegarde de l'approximation affine courante de la variable $z_j(t) : z_j(t) \simeq z_j + m_{z_j} \times t$. Ces deux constantes sont fournies par les M_i, dans le cas des x_i, via δ_{ext} et les variables v et m_v. Ainsi $m_{z_j} \simeq \dot{z_j}(t)$. Enfin, on peut montrer que $f_i(z_1, \ldots, z_q)$ fournit $\dot{x_i}$ et $c_1 m_{z_1} + \cdots + c_q m_{z_q}$ fournit $\ddot{x_i}$.

3.3 Performances

Les capacités de ces méthodes sont très encourageantes, nous les avons testées sur de nombreux problèmes (linéaires, non linéaires,...) toujours avec le même succès. En guise d'exemple, nous citerons un test de Kofman [KOF01] où il a comparé QSS1 avec d'autres méthodes connues : Il est parti du principe que pour résoudre le système (2), QSS1 a obtenu une précision de 10^{-2}, puis il a cherché comment obtenir la même précision avec les autres méthodes. Il a ensuite comparé le nombre de pas de calcul nécessaires.
Il a obtenu les résultats du tableau 1.

$$\begin{cases} \dot{x_1}(t) = \frac{1}{L} x_2(t) & \text{avec } x_1(0) = 0 \\ \dot{x_2}(t) = U - \frac{1}{C} x_1(t) - \frac{R}{L} x_2(t) & \text{avec } x_2(0) = 0 \\ y(t) = \frac{1}{L} x_2(t) \end{cases} \tag{2}$$

où $R = 100.01, L = 0.01, C = 0.01, U = 100$.
On peut remarquer que pour toutes les méthodes le nombre de pas de calcul est bien supérieur à celui de QSS1. Seule la méthode Matlab's ode15s réalise une meilleure performance. Mais selon Kofman, c'est au prix d'un calcul par pas très complexe (méthode d'ordre 5, inversion de matrice,...) sans comparaison avec la simplicité de QSS1. Zeigler trouve dans [ZEI98] des résultats tout aussi encourageants en soulignant de surcroit que pour QSS1, les messages qui transitent au sein du système sont +1 ou -1 (pour d_{i+1} ou d_{i-1}) et tiennent donc sur 1 bit.

3.4 Cell-DEVS

L'extension Cell-DEVS est née de la constatation suivante : de nombreux modèles font intervenir des espaces discrets et utilisent des formalismes tels que les automates cellulaires. Wainer et Giambiasi dans [WAI01] développent l'extension Cell-DEVS. Cette extension doit pouvoir décrire et simuler des modèles à base d'automates cellulaires multi-dimensionnels et à événements discrets. La dynamique des cellules est temporisée c'est-à-dire que l'état d'une cellule sera modifié en fonction de l'état de son voisinage, mais il sera connu des cellules voisines qu'après un certain délai. L'idée de base est de fournir un mécanisme simple de définition de la synchronisation des cellules. Comme pour toute proposition d'extension, les auteurs offrent à la fois l'extension du formalisme qui se résume à l'ajout de variables supplémentaires et de leur sémantique et le simulateur abstrait.
Un modèle Cell-DEVS est basé sur la même structure qu'un modèle DEVS classique. Nous allons retrouver les modèles atomiques pour les cellules, les éléments de base d'un réseau, et les modèles couplés pour les réseaux eux-mêmes. Néanmoins, la structure proposée par Zeigler [ZEI00] se voit augmentée d'attributs. Le modèle DEVS d'une cellule est défini par la structure suivante :
TDC = (X,Y,I,S,N,$delai$,d,δ_{int},δ_{ext},τ,λ,t_a)

où

- X et Y représentent l'ensemble des ports et des valeurs d'entrée et de sortie,
- I est l'interface de la cellule,
- S l'ensemble des états de la cellule,
- N l'ensemble des états des cellules voisines,
- $delai$ le type de délai utilisé,
- d la durée du délai,
- δ_{ext} (resp. δ_{int}) est la fonction de transition externe (resp. interne),
- τ la fonction locale de calcul,
- λ la fonction de sortie,
- ta la fonction d'avancement du temps.

L'interface I de la cellule définit le voisinage de la cellule ainsi que les connexions en terme de ports d'entrée et de sortie entre la cellule et ses voisines. Il y a autant de ports d'entrée que de voisines. La fonction de calcul τ modélise la fonction de calcul de l'état de la cellule en fonction de l'état du voisinage. L'état de la cellule est effectif pour les cellules voisines qu'au bout du délai d'attente d.
La définition des cellules est complétée par la définition d'un modèle couplé. Le modèle d'une cellule définit l'interface qu'elle offre à l'extérieur mais ne précise pas la forme des connexions. Le rôle du modèle couplé est donc de définir les connexions entre les cellules (à l'aide d'un *pattern* - N, C et B), la taille $\{t_1, ..., t_n\}$ et la dimension n du réseau et l'interface I avec d'éventuels modèles DEVS (de type Cell-DEVS ou non).

$$GCC = (X_{list}, Y_{list}, I, X, Y, n, \{t_1, ..., t_n\}, N, C, B, Z)$$

X_{list} et Y_{list} définissent la liste des cellules du réseau possédant des ports d'entrée et des ports de sortie non connectés en interne et par conséquent disponibles pour une connexion avec un autre modèle. En terminologie

DEVS, cet ensemble est l'ensemble des connexions entre les ports d'entrée et des ports de sortie avec les ports du modèle couplé. L'interface I du réseau complète la définition en réunissant au sein d'une même structure les éléments de définition de l'interface du réseau vers l'extérieur. Cette définition intègre l'ensemble Z qui met en relation les ports de sortie appartenant à Y_{list} d'un réseau et les ports d'entrée appartenant à X_{list} d'un autre réseau. X et Y représentent l'ensemble des événements d'entrée et de sortie. De manière à simplifier la définition des connexions entre les cellules du réseau (communément notées connexions internes dans les modèles couplés) un *pattern* de voisinage, noté N, est défini. Ce *pattern* spécifie pour toute cellule n'appartenant pas à la bordure B la position relative de ses voisins. Avec Cell-DEVS, nous disposons d'un outil de spécification DEVS pour les automates cellulaires. Il faut noter que Cell-DEVS ne s'arrête pas à un langage de spécification mais offre aussi les simulateurs abstraits afin de préciser le comportement d'un modèle.

4 APPLICATION AUX ÉQUATIONS AUX DÉRIVÉES PARTIELLES

4.1 Principe

Si maintenant, on essaie d'appliquer directement les méthodes QSS1 et QSS2 sur des équations différentielles manipulant des fonctions qui ne dépendent plus seulement du temps, on tombe sur une impossibilité logique. En effet, QSS1 et QSS2 reposent sur la notion d'événements discrets ce qui implique une dépendance des processus vis à vis du temps. Par contre, il est possible de réaliser une méthode hybride où la gestion du temps se ferait par un intégrateur type QSS et la gestion des dépendances spatiales par des méthodes classiques d'approximation de dérivée.

4.2 Un exemple d'application

Afin d'illustrer notre propos, considérons l'exemple de la diffusion de la température sur une barre. Celle-ci est gérée [FLE00] par l'équation suivante :

$$\frac{\partial T(t,x)}{\partial t} = K\frac{\partial^2 T(t,x)}{\partial x^2} \tag{3}$$

où K est le coefficient de diffusion.
On approxime alors $T(t,x)$ par ses valeurs en différents points de la barre après avoir choisi un pas spatial Δx. En notant $T_i(t)$ ces valeurs, l'équation 3 devient :

$$\frac{\partial T_i(t)}{\partial t} = K\frac{T_{i-1}(t) - 2T_i(t) + T_{i+1}(t)}{(\Delta x)^2} \tag{4}$$

pour i allant de 1 à N, en utilisant, par exemple, une approximation de la dérivée seconde en trois points.
Une des approches classiques (la méthode explicite) consisterait maintenant à approximer $\frac{\partial T_i(t)}{\partial t}$ par $\frac{T_i(t+\Delta t) - T_i(t)}{\Delta t}$. Mais les équations forment à ce niveau un système d'équations différentielles du premier ordre en $T_i(t)$. Il est donc possible de les résoudre à l'aide d'une méthode type QSS.

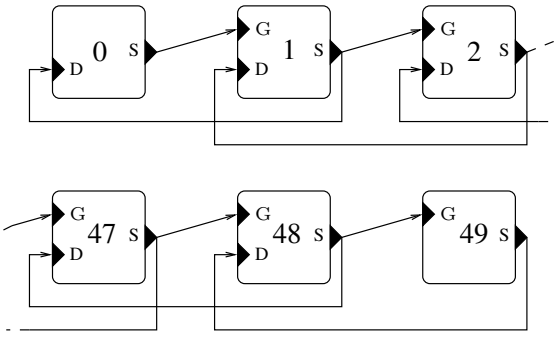


FIG. 4 – Le modèle Cell-DEVS

4.3 La formalisation DEVS

On peut modéliser le problème par 50 cellules Cell-DEVS interconnectées (cf figure 4). Une cellule possède un port de sortie S qui envoie la nouvelle température avant l'évaluation de sa fonction de transition interne à ses voisines. Elle possède également deux ports d'entrée G et D pour recevoir la température des voisines par activation de sa fonction de transition externe. Dans notre cas, le délai d'envoi de l'état de la cellule est nul. Les voisins sont donc immédiatement informés du changement d'état.

4.4 Les premiers résultats

Nous avons réalisé ce premier test avec l'équation de la diffusion (cf 3) et les paramètres suivants :

- une barre graduée de 0 à 1
- un coefficient de diffusion de : $K = \frac{1}{\pi^2}$
- des valeurs limites aux bords fixées à 0 : $T(t,0) = T(t,1) = 0\ \forall t$
- une fonction d'initialisation à $t = 0$ égale à $T(0,x) = sin(\pi x)$

Ainsi initialisé, la solution exacte du système est : $T(t,x) = e^{-t}sin(\pi x)$. Pour la discrétisation spatiale, la barre est découpée en 50 cases, les cases aux extrémités étant fixées à une température de 0. La méthode explicite (en faisant varier le pas de temps) et les méthodes QSS1 et QSS2 (en faisant varier la précision de quantification) ont été testées. Les résultats obtenus sont présentés dans les tableaux 2, 3 et 4. T correspond au temps simulé, le temps est la durée d'exécution, la précision donne la plus grande erreur obtenue sur la barre et sur la durée de l'expérience (les relevés étant réalisés toutes les 0.1 secondes) et enfin les itérations comptent le nombre d'itérations pour toutes les cases, celles-ci correspondant à un cycle complet de calcul pour une case. On peut préciser ici que la méthode explicite calcule à chaque itération les valeurs pour l'ensemble des cases de la barre. Afin de comparer les résultats, le nombre d'itérations de la méthode explicite est multiplié par le nombre de cases. Ceci explique le nombre important d'itérations par rapport à QSS1 avec, malgré tout, un temps d'exécution relativement comparable.

Time step	$T = 10s$		
	temps	précision	itérations
0.001	< 1s	0.00089	480 000
0.0005	1s	0.00046	960 000
0.0001	7s	0.00012	4 800 000
0.00005	13s	0.00013	9 600 000

Time step	$T = 30s$		
	temps	précision	itérations
0.001	2s	0.00089	1 440 000
0.0005	4s	0.00046	2 880 000
0.0001	20s	0.00012	14 400 000
0.00005	40s	0.00013	28 800 000

TAB. 2 – Mesures réalisées avec la méthode explicite

Précision	$T = 10s$		
	temps	précision	itérations
0.1	< 1s	0.095	6213
0.05	< 1s	0.05	9233
0.01	< 1s	0.0098	16 567
0.005	< 1s	0.0053	20 463
0.001	1s	0.0010	45 565
0.0005	1s	0.00052	76 771
0.0001	4s	0.00019	326 147
0.00005	7s	0.00016	637 452

Précision	$T = 30s$		
	temps	précision	itérations
0.1	< 1s	0.095	6212
0.05	< 1s	0.050	9232
0.01	< 1s	0.0098	16 566
0.005	< 1s	0.0053	20 462
0.001	1s	0.0010	45 564
0.0005	1s	0.00052	76 770
0.0001	4s	0.00019	326 146
0.00005	8s	0.00016	637 884

TAB. 3 – Mesures réalisées avec QSS1

Précision	$T = 10s$		
	temps	précision	itérations
0.1	2s	0.13	201 080
0.05	2s	0.061	202 560
0.01	2s	0.013	206 961
0.005	2s	0.006	207 933
0.001	2s	0.0012	209 107
0.0005	2s	0.00064	209 478
0.0001	2s	0.00023	209 362
0.00005	2s	0.00017	209 926

Précision	$T = 30s$		
	temps	précision	itérations
0.1	5s	0.13	621 627
0.05	5s	0.061	623 578
0.01	5s	0.013	627 057
0.005	5s	0.006	628 504
0.001	5s	0.0012	630 084
0.0005	5s	0.00064	629 039
0.0001	5s	0.00023	628 794
0.00005	5s	0.00017	630 505

TAB. 4 – Mesures réalisées avec QSS2

4.5 Interprétations

Retenons d'abord qu'il a été nécessaire de démarrer les tests sur la méthode explicite à partir d'un pas de temps de 0.001. En effet, la méthode est stable dès que l'on vérifie $K\frac{\Delta t}{(\Delta x)^2} < \frac{1}{2}$ (ce qui donne ici $\Delta t < 0.002$). Par contre QSS1 et QSS2 semblent stables pour toutes les précisions. Nous avons notamment testé QSS1 sur d'autres problèmes avec toujours le même succès en ce qui concerne la stabilité.

Ensuite, on peut remarquer que QSS1 et QSS2 génèrent nettement moins d'itérations que la méthode classique, et la différence devient écrasante pour $T = 30s$. Ceci s'explique par le maximum de la fonction $T(x)$ qui vaut $Max(t) = e^{-t}$ et pour, par exemple, $t = 15s$ $Max(15) \simeq 3.06 \times 10^{-7}$; les méthodes QSS produisant une itération après chaque franchissement d'une barrière de quantification, elles ne sont plus nécessaires à partir d'un certain t donné. On peut constater cette adaptation de la charge de calcul sur le graphique 5 qui montre le nombre d'itérations par 0.1 seconde avec la méthode QSS1.

Remarquons également que la précision des méthodes QSS1 et QSS2 est de l'ordre de grandeur de la précision choisie, et enfin, que QSS2 semble avoir davantage de difficultés vis à vis de QSS1 pour adapter sa charge de calcul, son nombre d'itérations étant presque constant.

5 CONCLUSION

Au vu de ces premiers résultats, on peut conclure qu'il est possible d'intégrer des méthodes du type QSS pour la gestion des équations différentielles aux dérivées partielles. Le gain en terme de vitesse d'exécution est malheureusement inexistant sauf dans le cas où le système

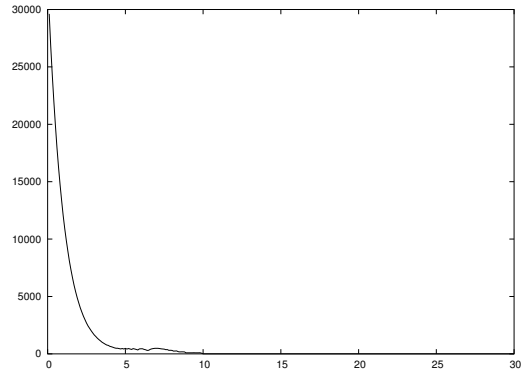


FIG. 5 – Nombre d'itérations par 0.1 seconde en utilisant la méthode QSS1

admet une solution qui tend vers une fonction constante (indépendante du temps), auquel cas la méthode adapte sa charge de calcul en fonction de ses besoins et la vitesse d'exécution est alors beaucoup plus grande. Par contre, le gain en terme d'intégration au sein d'un système DEVS, comme le *framework* VLE développé au laboratoire du LIL, est total puisque la méthode est naturellement, sans artifice, événementielle et compatible DEVS.

Il faut maintenant essayer de comprendre plus finement les calculs effectués par ces méthodes afin de les améliorer pour que leurs performances atteignent et dépassent sur tous les tests les méthodes classiques. Nous pensons, par exemple, à un test d'arrêt plus efficace, pour accentuer l'effet d'adaptation de la charge de calcul, peut-être également un travail sur l'espace de recherche des solutions. . . ?

BIBLIOGRAPHIE

[FLE00] C.A.J. Fletcher, "*Computational Techniques for Fluid Dynamics Volume 1 and 2*". Springer 2nd Edition, 2000

[GIA98] N. Giambiasi, "*Abstractions à événements discrets de systèmes dynamiques*". APII-Jesa. Vol. 32, No. 3, pp. 275-311, 1998

[KOF01] E. Kofman et S. Junco, "*Quantized-State Systems. A DEVS Approach for Continuous System Simulation*". Transactions of SCS, 18(3), pp. 123–132, 2001

[KOF02] E. Kofman, "*A Second Order Approximation for DEVS Simulation of Continuous Systems*". Transactions of SCS, 78(2), pp. 76–89, 2002

[MUZ02] A. Muzy, E. Innocenti, A. Aiello, J. F. Santucci et G. Wainer, "*Cell-Devs Quantization Techniques in a Fire Spreading Application*". Proceedings of the 2002 Winter Simulation Conference, pp. 542–549, 2002

[RAM03] E. Ramat et P. Preux, "*"Virtual laboratory environment" (VLE) : a software environment oriented agent and object for modelling and simulation of complex systems*". Simulation Modelling Practice and Theory, pp. 45-55, vol 11, 2003

[WAI01] Wainer G.A. et Giambiasi N., "*Application of the Cell-DEVS Paradigm for Cell Spaces Modelling and Simulation*". Simulation 76 :1, pp.22-39, 2001.

[ZEI98] B.P. Zeigler, "*DEVS Theory of Quantized Systems*". Juin 1998

[ZEI00] B. P. Zeigler, H. Praehofer, Tag Gon Kim, "*Theory of Modeling and Simulation, Integrating Discrete Event and Continuous Complex Dynamic Systems*". Seconde Edition, 2000