

UNIVERSITÉ LIBRE DE BRUXELLES
Faculté des Sciences
Département d'Informatique

**Approches non Probabilistes pour la Simulation de
Systèmes Dynamiques en Présence d'Incertitude :
Etude Expérimentale.**

Frédéric Branger

Mémoire présenté en vue de
l'obtention du diplôme de licencié
en informatique

Année académique 2002–2003

Table des matières

Introduction	4
I Etat de l’art de la simulation en présence d’incertitude	8
1 La simulation de systèmes dynamiques continus en absence d’incertitude	9
1.1 Les équations différentielles ordinaires	9
1.1.1 Introduction au concept	9
1.1.2 Résolution numérique des ODE	11
1.2 Les équations différentielles partielles	13
1.2.1 Introduction au concept	13
1.2.2 Résolution numérique des PDE	15
2 Représentation de l’incertitude	17
2.1 Représentation par les probabilités	18
2.2 Représentation par intervalles	18
2.2.1 Intervalles classiques	19
2.2.2 Nombres affins	19
2.2.3 Intervalle de confiance	20
2.3 Représentation par les nombres flous	20
2.3.1 Théorie des ensembles flous	20
2.3.2 Principe d’extension et opérations	23
3 La simulation des systèmes en présence d’incertitude	24
3.1 Simulation avec probabilités	25
3.1.1 La modélisation	25
3.1.2 La résolution du modèle	25
3.1.3 Simulation par Monte-Carlo	26
3.2 Simulation par arithmétique d’intervalles	27
3.2.1 Intervalles classiques	27
3.2.2 Nombre affins	28

3.3	Simulation par les nombres flous	29
3.3.1	Equation différentielle floue	29
3.3.2	Simulation possibiliste vs. simulation probabiliste	30
4	Simulation qualitative avec l'approche Qua.Si. (Qualitative Simulator)	32
4.1	Recherche des extrema en tant que problème d'optimisation	32
4.2	L'algorithme	33
4.3	Note sur l'implémentation	36
4.4	Vue critique	36
4.4.1	Aspects positifs	36
4.4.2	Aspects négatifs	36
II	Apports au domaine	38
5	Extension de l'approche Qua.Si. avec des méthodes stochastiques d'optimisation	39
5.1	Monte-Carlo	39
5.2	Simulated Annealing	43
5.2.1	Introduction à Simulated Annealing	43
5.2.2	Application à notre problème	45
6	Simulation qualitative des PDE, avec l'approche Qua.Si.	54
6.1	Résolution des PDE déterministes	55
6.1.1	Résolution d'équations similaires à l'équation de Laplace	55
6.2	Application aux PDE floues	57
6.2.1	Les PDE floues	57
6.2.2	L'approche Qua.Si. pour la résolution de PDE floues	58
6.2.3	Remarques et considérations	59
7	Simulations réalisées	61
7.1	Modèle de la température de transformateurs de courant	61
7.1.1	Description du problème	61
7.1.2	Le modèle mathématique	62
7.1.3	L'expérience	63
7.2	Modèle de Lotka-Volterra	75
7.2.1	Le modèle	75
7.2.2	L'expérience	75
7.3	Equation de Laplace	90
7.3.1	Discussion des résultats	90

Conclusion	96
A Méthodes numériques	98
A.1 Méthode d'Euler [MF99]	98
A.2 Résolution de systèmes linéaires	99
A.2.1 Méthode de Jacobi [MF99]	99
A.2.2 Méthode de Gauss-Seidel [MF99]	99
A.3 Méthode de Kuhn-Tucker	100
B Les programmes Matlab développés	101
B.1 Qua.Si+MC et Qua.Si+SA	101
B.2 Résolution de l'équation de Laplace floue	104
Bibliographie	107

Introduction

Modélisation et simulation

La simulation est une des composantes du processus de la modélisation et de la prévision du comportement de systèmes dynamiques. Il est possible de représenter ce processus en différentes phases.

Nous disposons au départ d'un phénomène naturel que nous désirons analyser et dont nous voulons prévoir le comportement.

La première phase du processus est la modélisation conceptuelle. Elle va consister à spécifier les interactions existant entre les composantes physiques du phénomène ainsi qu'avec son environnement. S'il est désiré de considérer le non-déterminisme d'événements qui pourraient se produire, il faut également spécifier quels éléments seront traités de façon non-déterministe.

Il faut maintenant définir un modèle mathématique du phénomène à partir du modèle conceptuel. Cette phase inclura la spécification complète des équations, les conditions initiales ou les conditions aux bords et le choix de la représentation formelle de la partie non-déterministe. Dans notre cas, nous obtiendrons une représentation sous forme de variables et d'un système d'équations différentielles et un formalisme non-déterministe. Ces équations modéliseront le phénomène que l'on désire étudier. Le but de la simulation sera alors de représenter, de la façon la plus appropriée, l'évolution du modèle du système au cours du temps [Zei76].

Dans un nombre important de cas, les équations ne sont pas linéaires, ce qui entraîne une grande difficulté, voire même l'impossibilité de les résoudre de façon analytique. Il sera alors nécessaire d'utiliser des méthodes numériques pour résoudre ces systèmes. La troisième phase consistera alors en le choix des méthodes numériques pour la résolution du système et également la discrétisation du système qui était représenté de façon continue. Il nous faudra également déterminer la façon par laquelle simuler le non-déterminisme.

Vient ensuite l'encodage sur l'ordinateur du modèle discret obtenu lors de la phase précédente.

Il nous faut par après lancer les calculs pour obtenir la solution numérique du système.

Enfin viendra la représentation de la solution numérique. Cette phase finale va consister en la représentation des valeurs numériques en termes du phénomène que l'on voulait simuler. Il faudra reconstituer la continuité de certains paramètres, apposer les ordres de grandeurs et unités de mesure appropriées, . . . A partir de cela, il sera possible d'interpréter les résultats et en

tirer les conclusions.

Incertitude

Nous avons parlé de composantes non-déterministes dans la section précédente. En effet, des incertitudes peuvent apparaître pour diverses raisons :

- La connaissance du système n'est pas complète car certaines parties ne peuvent être observées.
- La connaissance du système n'est pas complète car certaines parties sont toujours en développement.
- Le modèle du système est connu mais trop complexe pour être représenté de façon détaillée.
- Les paramètres du système peuvent changer au cours du temps, de façon imprévisible.
- Les paramètres fournis proviennent de données ou prédictions soumises à une certaine imprécision.
- ...

La simulation avec incertitude consistera donc à simuler des systèmes dont le modèle est imprécis [ATMVdlR00]. La difficulté principale sera de représenter et propager au cours du temps l'incertitude contenant la solution du système.

L'approche traditionnelle pour représenter cette incertitude est probabiliste. Dans ce cas, la représentation des données nécessite l'utilisation de variables aléatoires et de distribution de probabilités. Cependant, dans certains cas, utiliser les probabilités est inapproprié :

1. Les experts du système ne sont pas toujours certains de la représentation de la connaissance imprécise des variables en termes de distributions de probabilités à cause de leur connaissance qualitative du système.
2. Les experts du système ne sont pas toujours certains de la représentation de la connaissance imprécise des variables en termes de distributions de probabilités à cause d'une quantité insuffisante de données pour pouvoir estimer les distributions de probabilités.
3. Pour adopter des méthodes probabilistes, les ingénieurs font souvent la supposition de la normalité et de l'indépendance statistique des variables. Les résultats expérimentaux montrent souvent que ces suppositions ne sont pas confirmées par les évidences empiriques.

En plus de méthodes probabilistes de représentation de l'incertitude, il existe donc des méthodes non-probabilistes pour modéliser cette incertitude, comme la théorie de la mathématique d'intervalles [Moo66], la théorie des possibilités (basée sur la théorie des ensemble flous) [Zad78] et la théorie de l'évidence [Sha76]. Les méthodes par mathématique d'intervalles et par les nombres flous feront l'objet d'une étude expérimentale dans ce document.

La simulation avec incertitude peut être appliquée à tous les domaines dans lesquels la complexité ou l'imprécision vient jouer un rôle important. Parmi ces domaines, l'industrie du

nucléaire [BB94], la production d'électricité [GIVV02], les prévisions météorologiques, la détection de poches de gaz naturel ou de pétrole [DK01],...

En particulier, l'étude expérimentale se focalisera sur l'étude d'un modèle mathématique utilisé dans la production d'électricité. Ce modèle a pour but la prévision de la température des transformateurs des sous-stations d'approvisionnement en courant [GIVV02].

Plan du document

Ce document est divisé en deux parties. La première donne un aperçu de l'état de l'art dans la simulation avec incertitude.

- Dans le premier chapitre, nous évoquerons la simulation numérique en absence d'incertitude. Nous y rappellerons donc les notions d'équation différentielle ordinaire et d'équation différentielle partielle. Nous discuterons également de méthodes numériques pour la résolution de ces équations.
- Dans le deuxième, nous allons parler de différents formalismes de représentation de l'incertitude. En plus d'évoquer l'approche probabiliste, il sera question de certaines méthodes non-probabilistes, à savoir les représentations par intervalles, par arithmétique affine et enfin par les nombres flous.
- Dans le troisième chapitre sera fait un exposé sur l'utilisation de ces méthodes de représentation d'incertitude pour la simulation avec incertitude.
- Dans le quatrième chapitre, nous allons parler de l'approche Qua.Si (Qualitative Simulator) [Bon96]. L'intérêt de cette méthode est de permettre la simulation de systèmes dont les paramètres et/ou les conditions initiales prennent des valeurs floues. La particularité de cette méthode est de considérer la propagation de l'incertitude au cours du temps comme un problème d'optimisation basé sur les gradients des équations à résoudre.

La seconde partie constituera en notre contribution innovante au domaine.

- Nous proposerons dans le chapitre cinq une extension à l'approche Qua.Si. par des techniques d'optimisation stochastiques. Le but sera de précéder l'optimisation basée sur les gradients par une méthode d'optimisation stochastique, non basée sur les gradients des équations. Le but de ces extensions est d'obtenir des résultats plus précis et/ou en un temps plus réduit que l'approche Qua.Si. standard.
- En six, nous étudierons et proposerons à une extension possible à Qua.Si. pour les équations différentielles partielles. En particulier, nous nous attarderons sur les équations du type de l'équation de Laplace
- Dans le septième et dernier chapitre, nous étudierons de façon expérimentale les techniques évoquées dans ce document. Nous étudierons et comparerons l'application de ces techniques à différents problèmes.

Le premier sera un problème réel dans le domaine de la production d'électricité. Cet

exemple permet l'étude, pour un modèle réaliste, du comportement des techniques de simulations avec un nombre relativement important de variables.

Le deuxième traitera de l'étude du modèle de Lotka-Volterra, afin d'étudier les résultats obtenus en cas de non-linéarité du système. Ces deux expériences nous permettront de discuter des qualités et défauts des méthodes dans le cas d'équations différentielles ordinaires.

Enfin, nous allons comparer l'application de l'approche Qua.Si. à l'équation de Laplace avec l'utilisation de Monte-Carlo pour la simulation de ce modèle. Nous y verrons ainsi les performances respectives des deux méthodes dans le cas d'un problème simple d'équations différentielles partielles.

Première partie

Etat de l'art de la simulation en
présence d'incertitude

Chapitre 1

La simulation de systèmes dynamiques continus en absence d'incertitude

La formalisation mathématique d'un système continu consiste en l'utilisation de variables réelles définissant l'**état** et un système d'équations différentielles définissant le **modèle**. Simuler le système dynamique revient alors à résoudre le système d'équations différentielles. Ce qui nous donne alors l'évolution des variables d'état au cours du temps, définissant le **comportement** du système dynamique [Lue79].

On peut distinguer deux types d'équations différentielles : les équations différentielles ordinaires et les équations différentielles partielles. Un rappel de la notion d'équation différentielle ordinaire va être fait afin de mieux voir le parallèle avec les équations différentielles partielles.

1.1 Les équations différentielles ordinaires

1.1.1 Introduction au concept

Définition 1.1 - Equations différentielles ordinaires

Les équations différentielles ordinaires (que nous appellerons **ODE** par la suite) sont des équations de la forme [Doi99]) :

$$F(y^{n/}, y^{n-1/}, \dots, y', t) = 0 \quad (1.1)$$

où F est une fonction donnée de $n + 2$ arguments, t est la **variable indépendante**, y est la **variable dépendante**, c'est à dire que $t \mapsto y = y(t)$ est une fonction et $y^{i/}$ désigne la dérivée i -ème de cette fonction. L'**ordre** de cette fonction est le plus grand n tel que la dérivée n -ème apparaisse explicitement dans l'équation.

Résoudre l'équation consistera donc à en trouver toutes les **fonctions solutions**, c'est-à-dire toutes les fonctions $t \mapsto y(t)$ qui satisfont (1.1) ; ces fonctions seront supposées n fois dérivables sur leurs domaines. Le terme **ordinaire** porte sur le fait que les fonctions inconnues sont fonctions d'une seule variable.

Une équation de la forme

$$y^{n/} = G(y^{n/}, y^{n-1/}, \dots, y', y, t) \quad (1.2)$$

est une équation en **forme normale**. Si l'on introduit les variables dépendantes $y_1 = y$, $y_2 = y'$, \dots , $y_n = y^{(n-1)/}$, nous obtiendrons le système d'ODE du premier ordre équivalent

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ &\dots \\ y_n' &= G(y_n, y_{n-1}, \dots, y_2, y_1, t) \end{aligned}$$

Par exemple, dans le cas d'un problème d'ordre 2 de la forme

$$\frac{d^2 y}{dt^2}(t) = f\left(t, y(t), \frac{dy}{dt}(t)\right) \quad (1.3)$$

il suffira d'appliquer la substitution

$$y(t) = y_1(t) \quad \text{et} \quad \frac{dy}{dt}(t) = y_2(t). \quad (1.4)$$

Ce qui nous donnera le système

$$\begin{cases} \frac{dy_1}{dt} = y_2 \\ \frac{dy_2}{dt} = f(t, y_1, y_2) \end{cases}$$

Nous aurons donc plusieurs fonctions à résoudre, celles-ci pouvant dépendre les unes des autres. Nous appellerons donc **fonction solution** les fonctions à résoudre à l'intérieur du système d'équations différentielles. Celles-ci seront dénotées dans la plupart des équations par y_i $i = 1, \dots, n$. La plupart des systèmes dynamiques évoluant au cours du temps, la variable indépendante du système sera donc souvent le temps, dénoté par t . Les équations seront donc de la forme

$$\frac{dy_i}{dt} = \dots$$

Pour alléger les écritures, nous dénoterons, sauf cas particulier de formulation, $\frac{dy_i}{dt}$ par \dot{y}_i .

Dans notre cas, les système d'ODE à résoudre seront sous forme de problèmes aux conditions initiales (aussi appelés problèmes de Cauchy). Il nous faut nous assurer que les équations soient résolubles et admettent une solution unique. Un problème de Cauchy est de la forme :

$$\begin{cases} \dot{y} = f(t, y(t)) \\ y(t_0) = y_0 \end{cases}$$

où $y_0 \in \mathbb{R}^n$ représente la condition initiale du vecteur y .

L'existence et l'unicité de la solution est garantie par le théorème suivant (énoncé dans le cas où $n = 1$):

Théorème 1.1

Pour tout $t \in \text{domaine de } t$ et pour tout $y \in \mathbb{R}$ qui satisfait la condition de Lipschitz [MF99], alors le problème de Cauchy

$$\begin{cases} \dot{y} &= f(t,y(t)) \\ y(t_0) &= y_0 \end{cases}$$

a une solution unique $y(t) = y(t,y_0) \in \mathcal{C}^1$ pour tout $t \in \text{domaine de } t$.

Un exemple de problème de Cauchy (pour $n = 2$) est le système différentiel oscillant suivant :

$$\begin{aligned} \dot{y}_1 &= y_2 & y_1(0) &= 9 \\ \dot{y}_2 &= -y_1 & y_2(0) &= 4 \\ & & 0 \leq t &\leq 10 \end{aligned} \tag{1.5}$$

dont la solution des deux fonctions solutions est donnée par les graphes de la figures 1.1 et 1.2.

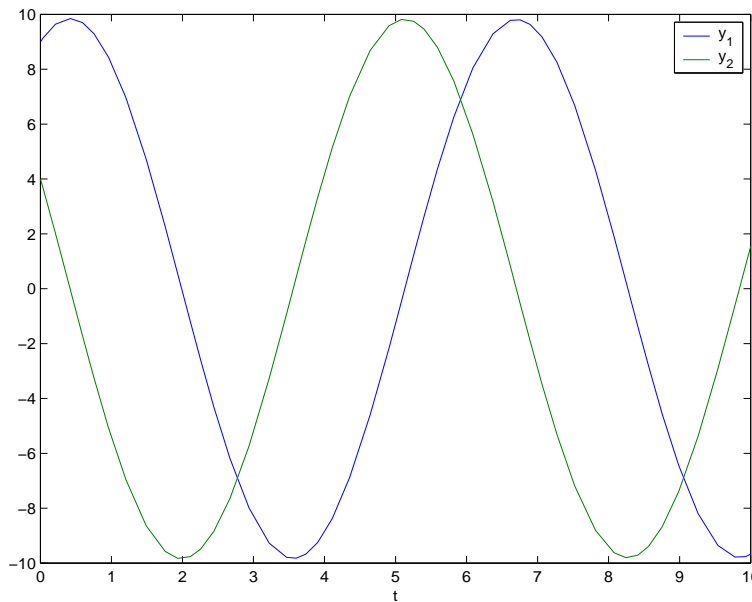


FIG. 1.1 – Solution du système oscillant (1.5) (y_1 et y_2 en fonction de t)

1.1.2 Résolution numérique des ODE

Il est rare d'avoir à sa disposition un moteur très performant de calcul symbolique pour résoudre ces systèmes en un temps acceptable. De plus, certaines équations ne sont pas encore résoluble avec cet outils. La complexité de calcul que provoque la résolution de systèmes d'équations différentielles nécessite donc l'utilisation de méthodes numériques.

Si le problème est posé sous forme d'un système d'ODE, il suffira d'appliquer la méthode à chacune des ODE composant le système.

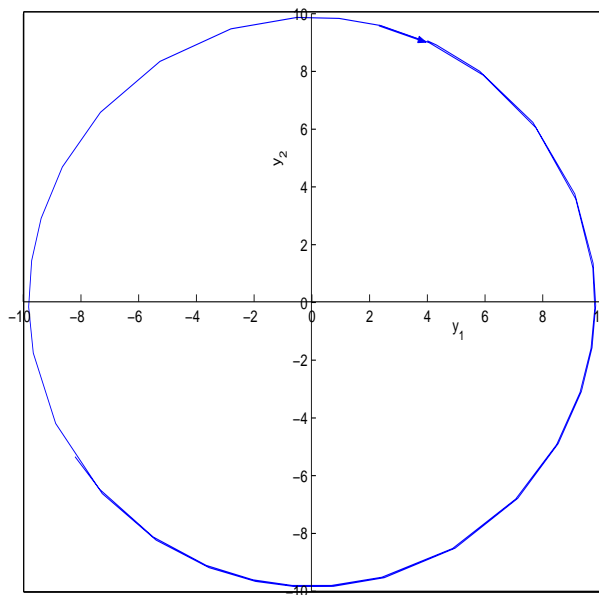


FIG. 1.2 – Solution du système oscillant (1.5) dans l'espace d'état

Nous allons introduire dans cette section deux des méthodes les plus connues, à savoir la méthode d'Euler et les méthodes de Runge–Kutta [MF99]. Il est intéressant de décrire ces deux méthodes car la méthode d'Euler sera utilisée pour la résolution par arithmétique affine (voir la section 3.2.2) et les méthodes de Runge-Kutta sont souvent considérées comme un bon compromis entre précision et rapidité du calcul.

La méthode d'Euler

Cette méthode est la plus simple à implémenter et permet de mieux comprendre les méthodes plus avancées mais possède comme désavantage que l'erreur générée augmente au fur et à mesure de l'avancement de l'algorithme. Elle consiste en la génération d'un ensemble de points, utilisés pour approximer la solution de l'équation. Cette méthode est décrite plus en détail dans l'annexe A.1. Voici un résumé dans le cas d'un ODE d'ordre $n = 1$

Soit $[a, b]$ l'intervalle sur lequel nous voulons résoudre la solution au problème $\dot{y} = f(t, y)$ avec $y(a) = y_0$. Divisons ensuite l'intervalle $[a, b]$ en M sous-intervalles égaux et prenons les points

$$t_k = a + kh \quad \text{pour } k = 0, 1, \dots, M \quad \text{où } h = \frac{b - a}{M}. \quad (1.6)$$

La valeur h est appelée le **pas**. Il faut ensuite résoudre approximativement

$$\dot{y} = f(t, y) \quad \text{sur } [t_0, t_M] \quad \text{avec } y(t_0) = y_0. \quad (1.7)$$

Il suffira d'itérer la formule suivante pour obtenir les points approxinant la courbe de la fonction

solution :

$$t_{k+1} = t_k + h, \quad y_{k+1} = y_k + hf(t_k, y_k) \quad \text{pour } k = 0, 1, \dots, M - 1. \quad (1.8)$$

Les méthodes de Runge–Kutta

La particularité des méthodes de Runge–Kutta (RK) est d'utiliser plusieurs évaluations de la fonction $f(t, y)$ et d'augmenter sa précision si on augmente le nombre d'évaluations de f . Ces méthodes se basent également sur le développement en séries de Taylor de la fonction solution. On dira qu'une méthode de RK est d'**ordre** o lorsque l'on fait en sorte que les coefficients, jusqu'à l'ordre o du développement de Taylor de la fonction à résoudre sont égaux à ceux de la fonction approchée [Fau99]. La méthode d'ordre 4 en est une des plus populaires.

La méthode RK d'ordre 4 consistera à utiliser l'approximation suivante à chaque pas d'itération :

$$y_{k+1} = y_k + \frac{h(f_1 + 2f_2 + 2f_3 + f_4)}{6} \quad (1.9)$$

avec

$$f_1 = f(t_k, y_k) \quad (1.10)$$

$$f_2 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}f_1\right) \quad (1.11)$$

$$f_3 = f\left(t_k + \frac{h}{2}, y_k + \frac{h}{2}f_2\right) \quad (1.12)$$

$$f_4 = f(t_k + h, y_k + hf_3) \quad (1.13)$$

1.2 Les équations différentielles partielles

1.2.1 Introduction au concept

Définition 1.2

Les équations différentielles partielles (appelées par la suite **PDE**) sont, à l'opposé des ODE, des équations dont les fonctions ont plus d'une variable indépendante. De façon similaire aux ODE, les équations sont de la forme

$$F\left(\frac{\partial^n \phi}{\partial x_n^n}, \frac{\partial^n \phi}{\partial x_n^{n-1} \partial x_{n-1}}, \dots, \frac{\partial^{n-1} \phi}{\partial x_n^{n-2} \partial x_{n-3}}, \dots, \phi, x_1, \dots, x_n\right) = 0 \quad (1.14)$$

où ϕ est la **variable dépendante**, $\{x_i\}_{i=1..n}$ est l'ensemble des **variables indépendantes**. L'**ordre** de cette fonction est le plus grand n tel que la dérivée n -ème partielle apparaisse explicitement dans l'équation.

Résoudre l'équation consistera donc à en trouver toutes les **fonctions solutions**, c'est-à-dire toutes les fonctions $(x_1, \dots, x_n) \mapsto \phi(x_1, \dots, x_n)$ qui satisfont l'équation (1.14) ; ces fonctions seront supposées n fois dérivables sur leurs domaines.

On peut distinguer trois classes de PDE : les PDE linéaires, non-linéaires et enfin les PDE quasi-linéaires.

Définition 1.3

Une PDE **linéaire** est une équation de la forme

$$f_1(x_1 \dots x_n) \frac{\partial^n \phi}{\partial x_n^n} + \dots + f_{k-1}(x_1 \dots x_n) \phi + f_k(x_1 \dots x_n) = 0 \quad (1.15)$$

où les fonctions f_i ne dépendent que des variables indépendantes.

Définition 1.4

Une PDE **non-linéaire** est une équation de la forme

$$f_1 \left(\frac{\partial^n \phi}{\partial x_n^n}, \dots, \phi, x_1 \dots x_n \right) + \dots + f_{k-1}(\phi, x_1 \dots x_n) + f_k(x_1 \dots x_n) = 0 \quad (1.16)$$

avec les fonctions f_i qui dépendent de la variable dépendante.

Définition 1.5

Une PDE **quasi-linéaire** est une équation non-linéaire où les fonctions f_i ne dépendent que des variables indépendantes et de dérivées d'ordre inférieur à l'ordre de la PDE.

Nous nous intéresserons dans ce document aux PDE quasi-linéaires du second ordre, qui seront donc de la forme

$$a \frac{\partial^2 \phi}{\partial x_1^2} + b \frac{\partial^2 \phi}{\partial x_1 x_2} + c \frac{\partial^2 \phi}{\partial x_2^2} = f \left(x_1, x_2, \phi, \frac{\partial \phi}{\partial x_1}, \frac{\partial \phi}{\partial x_2} \right) \quad (1.17)$$

où a , b et c sont des constantes. On peut les classer en trois types :

si $b^2 - 4ac < 0$, l'équation est qualifiée d'équation elliptique (1.18)

si $b^2 - 4ac = 0$, l'équation est qualifiée d'équation parabolique (1.19)

si $b^2 - 4ac > 0$, l'équation est qualifiée d'équation hyperbolique (1.20)

Pour alléger les notations, les PDE seront dénotées à l'avenir par $\phi_{x_i \dots x_k}$ qui signifiera la dérivée partielle de ϕ par les variables x_i à x_k .

Comme exemples de PDE elliptique, nous avons les équations de Laplace, Poisson et Helmholtz. Rappelons que le Laplacien d'une fonction $u(x,y)$ est :

$$\nabla^2 u = u_{xx} + u_{yy} \quad (1.21)$$

et nous avons donc les équations suivantes [MF99] :

$$\nabla^2 u = 0 \quad \text{pour l'équation de Laplace} \quad (1.22)$$

$$\nabla^2 u = g(x, y) \quad \text{pour l'équation de Poisson} \quad (1.23)$$

$$\nabla^2 u + f(x, y) = g(x, y) \quad \text{pour l'équation de Helmholtz} \quad (1.24)$$

Pour les PDE paraboliques, nous avons l'équation de la chaleur à une dimension [MF99] :

$$u_t(x, t) = c^2 u_{xx}(c, t) \quad (1.25)$$

Un exemple de PDE hyperbolique est l'équation de l'onde [MF99] :

$$u_{tt}(x, t) = c^2 y_{xx}(x, t) \quad (1.26)$$

1.2.2 Résolution numérique des PDE

On peut distinguer comme méthode de résolution numérique des PDE les méthodes par éléments finis [Fau99] et les méthodes par différence finie. Nous ne parlerons ici que de ces dernières.

Méthode par différence finie

Pour décrire cette méthode, nous avons tout d'abord besoin d'introduire la notion de différence-quotient qui consiste à approximer la dérivée d'une fonction. En analyse mathématique, il est connu que :

$$f'(x) = \lim_{h \rightarrow \infty} \frac{f(x+h) - f(x)}{h} \quad (1.27)$$

Nous pouvons donc en déduire une approximation :

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad (1.28)$$

Cependant cette dernière ne donne pas de très bon résultats. Nous pouvons alors imaginer de prendre deux points pour approximer la dérivée – un à sa gauche et un autre à sa droite. Nous obtiendrons alors des formules **centrées**. L'**ordre** de ces formules sera alors l'ordre de grandeur en $\mathcal{O}(g(h))$ de l'erreur induite par l'approximation.

Théorème 1.2 - Formules centrées d'ordre $\mathcal{O}(h^2)$ [MF99]

Si $f \in \mathcal{C}^3[a, b]$ (appartient aux fonctions dérivables au moins trois fois dans $[a, b]$) et que $x - h$ et $x + h \in [a, b]$ alors

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \quad (1.29)$$

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \quad (1.30)$$

avec une erreur d'ordre $\mathcal{O}(h^2)$

Cette notion désormais précisée, revenons à nos PDE. Les problèmes portant sur des PDE seront des problèmes dont nous connaissons les conditions aux bords. Il nous faudra alors ré-écrire les équations sous forme de différences-quotients, diviser l'espace comprenant nos variables en un quadrillage dont les points correspondent aux termes obtenus. Un exemple détaillé est donné dans le chapitre 6.

Nous aurons alors un système d'équations linéaires qu'il nous faudra résoudre pour obtenir les valeurs correspondant aux points. Pour résoudre ces systèmes d'équations linéaires, nous avons utilisé les méthodes de Gauss-Seidel, Jacobi ainsi qu'une méthode itérative par point fixe qui se base sur une reformulation de la différence-quotient d'ordre $n = 2$. Les méthodes de Gauss-Seidel et Jacobi seront expliquées en profondeur dans les sections A.2.1 et A.2.2.

Chapitre 2

Représentation de l'incertitude

Lorsque nous simulerons de manière numérique les systèmes, nous serons confrontés à des erreurs dans les calculs. On peut classer ces erreurs en diverses catégories [BdSSdW03] :

- Erreurs de modélisation : ce sont les erreurs qui vont être générées lorsque l'on utilise des modèles mathématiques approximatifs pour représenter un phénomène. Ce sera donc les erreurs dont la cause repose sur la façon de poser le problème.
- Erreurs de mesure : ce sont les erreurs dont la cause vient de la présence dans le modèle de valeurs provenant de mesures expérimentales. Ces valeurs ne seront alors connues qu'approximativement à cause de l'imprécision des méthodes de mesure.
- Erreurs d'approximation ou de troncature : ce sont les erreurs associées aux processus infinis en analyse mathématique, comme les séries numériques. Nous ne pourrions pas calculer de manière infinie ces processus et nous serons obligés de prendre une approximation.
- Erreurs d'arrondi : un ordinateur ne peut prendre en considération qu'un certain nombre de chiffres. Par exemple un nombre rationnel peut comporter un nombre infini de chiffres à droite de la virgule.
- Erreurs de propagation et génération : ce sont les erreurs obtenues dans les résultats des opérations comme conséquence des erreurs des opérandes. En effet, nous ferons des calculs sur des nombres approchés et les résultats auront une approximation qui sera la combinaison des erreurs des données de départ.

Les deux premiers types d'erreur sont regroupés sous le nom d'**erreurs de modélisation**, les trois derniers sous le nom d'**erreurs numériques**.

Dans ce document, nous nous focaliserons sur l'incertitude dérivant des erreurs de modélisation. Ce chapitre va traiter des méthodes existant pour représenter cette incertitude et l'intégrer dans la simulation numérique.

2.1 Représentation par les probabilités

L'approche probabiliste pour la représentation de l'incertitude repose sur les notions de variable aléatoire et de distribution de probabilités [Pap91]. Les domaines des variables peuvent être discrets ou continus. Un exemple de variable aléatoire discrète peut être le résultat d'un tirage de dés. Comme exemples de variables aléatoires continues, nous pouvons prendre la température environnementale ou encore le pourcentage d'erreurs de fabrications dans une chaîne de production. Nous aurons des variables aléatoires (discrètes ou continues, selon le problème) qui modéliseront les valeurs incertaines.

Si nous connaissons la distribution de probabilités de la variable aléatoire représentant les données sur lesquelles nous avons de l'incertitude, il nous sera alors possible de traiter cette incertitude à l'aide d'opérations sur les probabilités fournies par la distribution. Dans le cas où nous ne disposons pas de la distribution, mais seulement d'un échantillonnage des données, nous pourrions alors utiliser les méthodes d'inférence statistique pour estimer les informations manquantes. Il existe deux grands types de méthodes d'estimation : l'**estimation paramétrique** et l'**estimation non-paramétrique** [Bon03a].

Dans le cas où nous connaissons la forme de la distribution (p. ex. normale) et un échantillon de données mais certains paramètres de la distribution sont inconnus, il conviendra de procéder à l'estimation paramétrique.

Si nous ne connaissons pas la forme de la distribution de la variable aléatoire, les méthodes à utiliser seront les méthodes d'estimation non-paramétrique.

2.2 Représentation par intervalles

Lorsque nous n'avons aucune information au sujet de la distribution des valeurs possibles mais que nous disposons de bornes pour ces mêmes valeurs, l'incertitude peut être représentée par un intervalle sur lequel nous avons la garantie qu'il contiendra la vraie valeur [Moo66]. Cette garantie est formalisée par la notion d'**invariant fondamental de domaine pour la mathématique d'intervalles** [FS97] :

Propriété 2.1

Pour toute opération $f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ satisfaisant la condition de Lipschitz, nous pouvons définir une extension correspondante, sous forme d'intervalle $F : \mathfrak{R}^m \rightarrow \mathfrak{R}^n$, où \mathfrak{R} est l'extension aux intervalles de l'ensemble des nombres réels, telle que si le vecteur d'entrée (x_1, \dots, x_m) est inclus dans l'intervalle déterminé par les valeurs approximatives (X_1, \dots, X_m) , alors les quantités $(z_1, \dots, z_n) = f(x_1, \dots, x_m)$ sont sûrement comprises dans l'intervalle défini par les valeurs approximatives $(Z_1, \dots, Z_n) = f(X_1, \dots, X_m)$.

2.2.1 Intervalles classiques

La représentation par intervalle est une méthode simple et peu coûteuse en temps de calcul de représentation de l'incertitude. Nous aurons donc des valeurs de la forme

$$\bar{x} = [x_{inf}, x_{sup}] \quad (2.1)$$

où $x_{inf} \in \mathbb{R}$ et $x_{sup} \in \mathbb{R}$.

Au niveau du calcul, les intervalles sont additionnés, soustraits, multipliés de façon à ce qu'on ait la garantie que l'intervalle qui sera le résultat de l'opération contiendra bien la valeur inconnue qui est représentée. Ainsi, pour l'addition et la soustraction de deux intervalles :

$$\bar{x} + \bar{y} = [x_{inf} + y_{inf}, x_{sup} + y_{sup}] \quad (2.2)$$

$$\bar{x} - \bar{y} = [x_{inf} - y_{sup}, x_{sup} - y_{inf}] \quad (2.3)$$

Pour plus d'informations, [FS97] contient la récapitulation de l'adaptation aux intervalles de toutes les opérations classiques en mathématique. Cependant, il convient de modérer cette représentation par le fait qu'à chaque opération, l'intervalle obtenu contiendra plus que les valeurs calculées. Nous aurons donc un accroissement de l'imprécision qui deviendra important au fil des calculs, risquant alors de donner des valeurs calculées qui ne sont plus d'aucune utilité.

2.2.2 Nombres affins

Une autre façon de représenter les intervalles est le formalisme des nombres affins [ACS94, FS97]. Ceci est utile lorsque l'on dispose de sources multiples d'incertitude pour nos données. De plus, il a été montré [ACS94, FS97] que cette méthode donne de meilleurs résultats au niveau de l'accroissement de l'imprécision, la rendant donc plus adaptée pour les longues séries de calculs. Il convient de préciser que cela se fera au prix de calculs plus complexes. Avec ce formalisme, une valeur incertaine ξ sera représentée sous forme d'un polynôme :

$$\xi = x_0 + \sum_{i=1}^s x_i \varepsilon_i \quad (2.4)$$

où s est le nombre de sources d'incertitudes, les ε_i sont des variables symboliques inconnues qui prennent leur valeur dans l'intervalle $[-1, 1]$ et les x_i des coefficients réels connus. Chaque ε_i correspondra à une source d'incertitude indépendante et les x_i donneront l'importance de cette incertitude. On appelle x_0 la **valeur centrale**, x_i les **déviations partielles** et les ε_i les **symboles de bruit** [ACS94, FS97]. Nous noterons l'extension de \mathbb{R} en mathématique affine \mathbb{R} .

L'intervalle correspondant au nombre affiné sera donc un intervalle centré sur la valeur centrale, avec une taille qui vaudra le double de la somme des déviations partielles. En règle générale, lorsqu'une opération en mathématique affine est faite, des calculs sont effectués entre les

déviations partielles qui agissent sur les mêmes symboles d'erreur. Par exemple pour l'addition-soustraction de deux nombres affins \hat{x} et \hat{y} :

$$\hat{x} \pm \hat{y} = x_0 \pm y_0 + \sum_{i=1}^s (x_i \pm y_i) \varepsilon_i \quad (2.5)$$

Si les nombres sur lesquels on travaille n'ont pas les mêmes sources d'erreurs ou bien que les opérateurs utilisés nécessitent l'introduction de nouveaux symboles d'erreur, on aura, comme résultat de chaque opération, un nombre affin avec plus de composantes que le précédent, ce qui entraînera donc lors de l'implémentation des problèmes d'espace mémoire et de temps de calcul. On peut alors diminuer le nombre de symboles d'erreurs en les fusionnant [FS97].

2.2.3 Intervalle de confiance

Notons enfin que les notions d'intervalle et de probabilité ne sont pas forcément en opposition. Il est également possible de combiner les probabilités et les intervalles grâce à la notion d'intervalle de confiance : on va associer à l'intervalle une probabilité qui correspondra à la probabilité que la variable sur laquelle nous avons de l'incertitude est contenue dans l'intervalle. Les opérations sur les intervalles de confiance seront une combinaison des opérations sur les probabilités et sur le formalisme (classique ou affiné) utilisé pour représenter l'intervalle.

2.3 Représentation par les nombres flous

La théorie des **ensembles flous** (en anglais fuzzy set) a pour but de représenter la notion de connaissance imprécise. Par exemple dans le cas d'une bonbonne de gaz, nous pourrions exprimer la phrase *La pression à l'intérieur de la bonbonne est d'à peu près 5 bar* par le formalisme $p = \mu$ ou μ est un ensemble flou centré sur 5.

Introduisons maintenant les fondements théoriques de la notion d'ensemble flou [Zad65].

2.3.1 Théorie des ensembles flous

Un ensemble flou est une extension du concept d'ensemble booléen. Soit Y un ensemble classique d'objets, appelé l'**univers** dans lequel les éléments génériques sont dénotés par y . L'appartenance dans un sous-ensemble booléen A de Y peut être vu comme une fonction caractéristique, ou **fonction d'appartenance**

$$\mu_A : Y \rightarrow \{0,1\} \quad \text{telle que} \quad (2.6)$$

$$\mu_A(y) = 1 \quad \text{si } y \in A \quad (2.7)$$

$$\mu_A(y) = 0 \quad \text{si } y \notin A \quad (2.8)$$

$\{0,1\}$ est appelé l'**ensemble de valuation**. Si l'ensemble de valuation peut être l'intervalle réel $[0, 1]$, A est appelé un **ensemble flou**.

Définition 2.1

Une α -**coupure** de niveau h ($0 \leq h \leq 1$) d'un ensemble flou A est l'ensemble précis

$$A_h = \{y \in A, \mu_A \geq h\}. \quad (2.9)$$

Définition 2.2

Un ensemble flou est **normalisé** si

$$\exists y | \mu_A(y) = 1 \quad (2.10)$$

Définition 2.3

Un **nombre flou** est un sous-ensemble flou, convexe et normalisé du domaine réel \mathbb{R} (figure 2.1).

Notons que μ_y peut également s'écrire en *m.f.y* (pour membership function). Les deux notations sont rencontrées dans les figures de ce document.

Le concept de nombre flou est donc une extension de la notion de nombre réel: il permet l'encodage de données quantitatives approximatives [KG85].

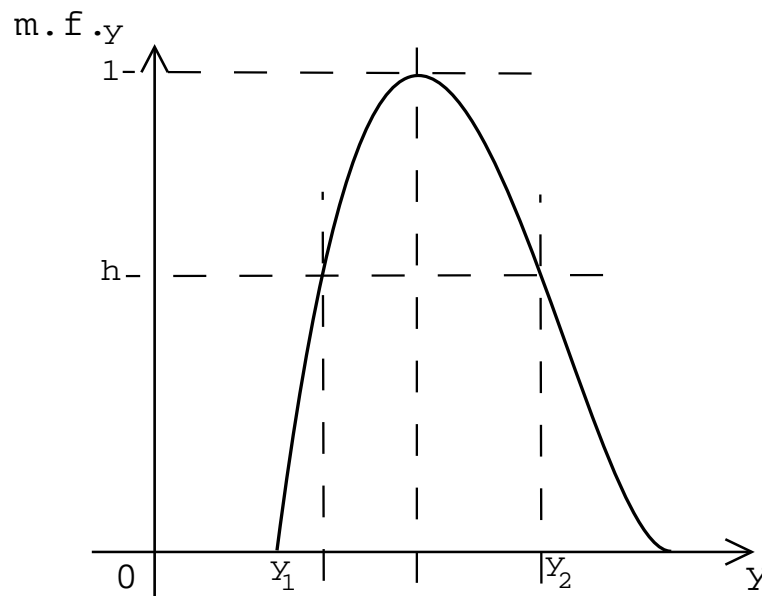


FIG. 2.1 – Un nombre flou dans le domaine Y

Introduisons maintenant la notion de **possibilité**. Par exemple, $\mu_{grand}(h)$ quantifie l'appartenance d'une personne de taille h à l'ensemble des personnes de grande taille. La valeur $\pi_{grand}(h)$ quantifie la possibilité que si un homme mesure une taille h , il appartienne à l'ensemble des personnes de grande taille. Alors, [Zad78]

$$\mu_{grand}(h) = \pi_{grand}(h) \quad \text{pour tout } h \in H \quad (2.11)$$

où $H = [0, \infty]$ est l'ensemble des tailles. De façon générale, π_A est appelé la **distribution de possibilités** associée à l'ensemble A et le **principe possibiliste de Zadeh** dit que

$$\mu_A(y) = \pi_A(y) \quad \text{pour tout } A, \text{ pour tout } y \in A \quad (2.12)$$

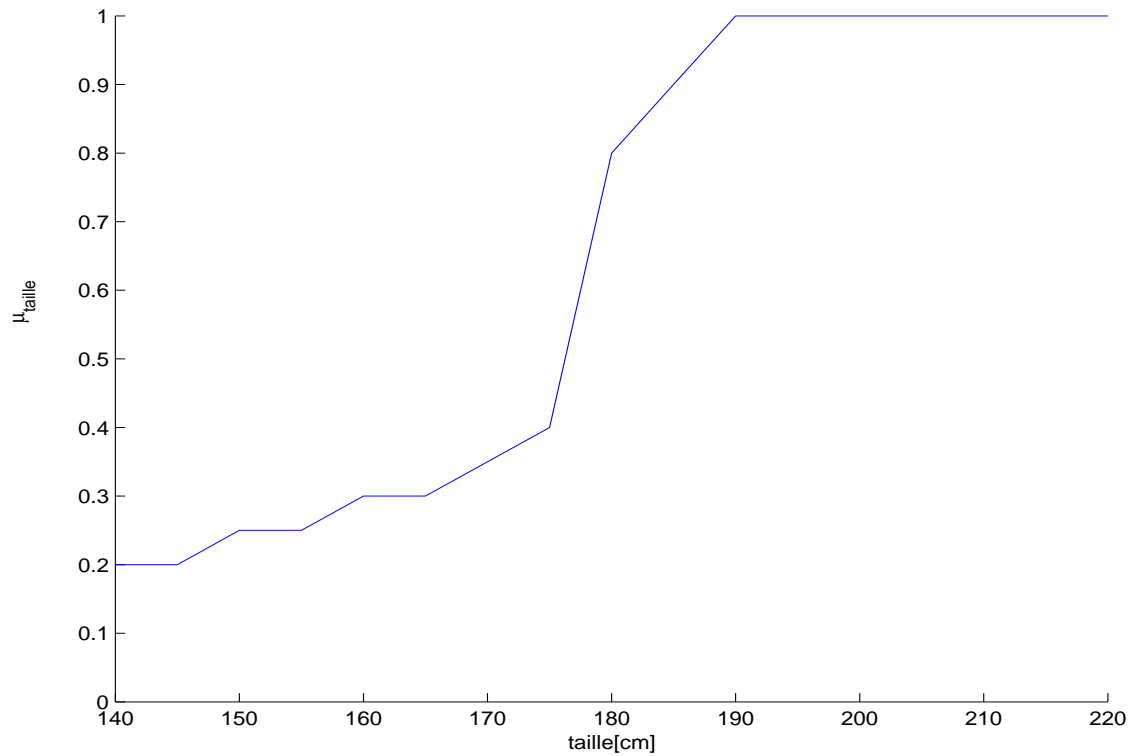


FIG. 2.2 – *Notion floue de taille*

Différentes notions peuvent être utilisées pour interpréter la fonction $\mu_A(y)$:

1. **α -coupure :** Les lignes horizontales de hauteur α correspondent au niveau de possibilité α . La projection de leurs intersections avec la courbe donne l'intervalle de variation au niveau de possibilité α .
2. **Degré de possibilité :** Pour toute valeur y , la hauteur de la courbe est le degré de la possibilité que le paramètre donné prenne la valeur y .
3. **Logique multivaluée :** Le degré de possibilité peut être interprété comme la valeur de vérité de l'assertion *La valeur du paramètre est y*.

L'approche des nombres, et donc des ensembles flous peut donc être judicieusement utilisée dans le cas où une approche probabiliste n'est pas envisageable, comme dans les cas suivants :

1. Les fluctuations des données ne sont pas de nature stochastique mais liées à une connaissance imprécise.
2. On ne dispose d'aucune information statistique sur les données.

3. La structure d'un modèle probabiliste ne peut être justifiée avec les informations dont on dispose.
4. L'estimation (paramétrique ou non-paramétrique) est jugée trop coûteuse ou peu fiable.

2.3.2 Principe d'extension et opérations

Nous avons besoins de certaines notions théoriques pour définir les opérations sur les nombres flous. La mathématique sur les nombres se base sur le **principe d'extension de Zadeh** [Zad65]. Ce principe nous donne un méthode générale pour étendre la mathématique classique aux nombres flous [DP80].

Soit $\varphi : X_1 \times X_2 \times \dots \times X_n \rightarrow Y$ une fonction réelle de $X_1 \times X_2 \times \dots \times X_n$ vers l'univers Y de façon à ce que $y = \varphi(x_1, x_2, \dots, x_n)$. Le principe d'extension permet d'induire de n ensembles flous \bar{x}_i sur X_i , un ensemble flou \bar{y} sur Y par l'intermédiaire de φ de façon à ce que :

$$\mu_{\bar{y}}(t) = \sup_{t=\varphi(s_1, \dots, s_n)} \min(\mu_{\bar{x}_1}(s_1), \dots, \mu_{\bar{x}_n}(s_n)) \quad (2.13)$$

ou

$$\mu_{\bar{y}}(t) = 0 \text{ si } \varphi^{-1} = \emptyset \quad (2.14)$$

où $\varphi^{-1} = \emptyset$ est l'image inverse de t , et $\mu_{\bar{x}_i}$ est la fonction d'appartenance de x_i ($i = 1, \dots, n$).

Le principe d'extension nous fournira donc une méthode pour calculer la valeur floue d'une fonction floue, mais cela n'est pas possible en pratique à cause du nombre infini de calculs qui seraient nécessaires.

Nous considérerons ici le calcul flou comme une extension de la mathématique des intervalles à l'aide de la notion d' α -coupure. [Ngu78] a prouvé que calculer la valeur d'une fonction floue appliquant une α -coupure d'un niveau générique h ($0 \leq h \leq 1$) sur son image dépend seulement des α -coupures de niveaux h des arguments.

$$\bar{y}_h = \varphi(\bar{x}_1, \dots, \bar{x}_n)_h = \varphi(\bar{x}_{1h}, \dots, \bar{x}_{nh}) \quad (2.15)$$

Nous pourrions alors décomposer une opération floue en plusieurs opérations sur des intervalles, chacune ayant comme arguments les intervalles correspondant aux α -coupures de même niveau. Les résultats seront alors les α -coupures du résultat de l'opération floue. Nous pouvons également en déduire qu'un nombre flou rectangulaire peut être équivalent à un intervalle.

Nous appellerons cette méthode basée sur la discrétisation par α -coupures des arguments flous, la **méthode par α -coupures**. Nous ramenons ainsi le problème du calcul flou à du calcul d'intervalles.

Chapitre 3

La simulation des systèmes en présence d'incertitude

Dans le cas d'une équation différentielle où tous les paramètres sont connus d'une manière déterministe, p. ex. si les conditions du problème de Cauchy sont satisfaites, nous obtiendrons comme solution, l'évolution du système sous forme d'une trajectoire. Par exemple, dans le cas du système (1.1), la solution peut être représentée graphiquement par la trajectoire de la figure 3.1.

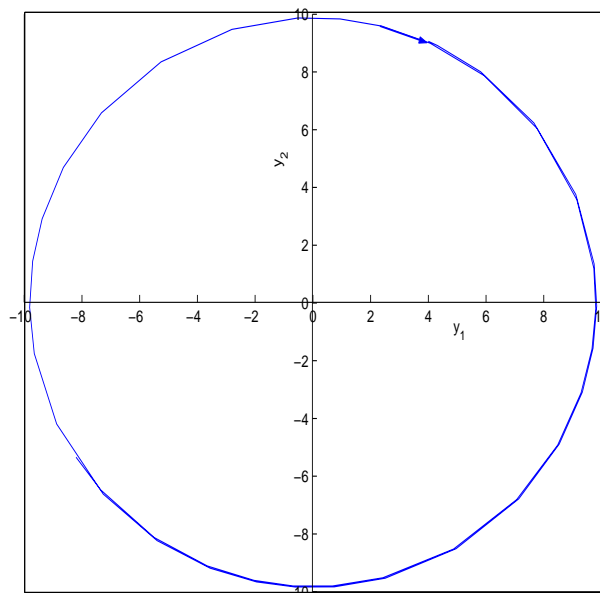


FIG. 3.1 – *Solution d'un système oscillant (trajectoire de y_1 en fonction de y_2)*

Le but de la simulation avec imprécision sera de simuler l'évolution de la distribution des diverses trajectoires. Malheureusement, trouver la distribution qui tienne compte d'exactement

toutes les trajectoires est parfois irréaliste si le système est fort complexe (p. ex. équation non-linéaire). Il est alors envisageable de définir des méthodes qui fournissent comme solution une bonne approximation de la distribution.

On désirera également des méthodes limitant la propagation des imprécisions au cours du temps sur lequel on simule le système.

3.1 Simulation avec probabilités

3.1.1 La modélisation

Si l'on représente l'incertitude par des probabilités, on parlera alors de systèmes d'**équations différentielles stochastiques (SDE)** pour les modèles mathématiques composés d'ODE auxquelles on a ajouté des éléments aléatoires. Cependant, la définition de SDE dépend de ce que l'on entend par dérivées d'un processus stochastique. Une notion importante pour cela est la **régularité** [Sob90] des fonctions aléatoires intervenant dans les équation différentielles.

Si l'on se retrouve avec un processus aléatoire très irrégulier, nous avons ce qui est appelé **une équation différentielle stochastique d'Ito**. Dans ces équations, l'incertitude des variables et/ou des paramètres est représentée en ajoutant à leur valeur déterministe l'influence d'un processus aléatoire du type du bruit blanc et s'exprime comme suit. Considérons un paramètre $U(t)$ variant en fonction du temps sujet à certaines actions aléatoires. Selon Ito, cette incertitude est modélisée par

$$U(t) = D(t) + \varepsilon(t) \quad (3.1)$$

où $D(t)$ est la composante déterministe et où la distribution de probabilités du processus stochastique $\varepsilon(t)$ est connue.

Si le processus aléatoire rencontré dans les équations est suffisamment régulier on va alors parler d'**équation différentielle stochastique régulière**. Dans ce cas, le paramètre incertain U est représenté par une variable aléatoire dont la distribution de probabilités est connue. Les SDE dont nous allons parler ensuite seront régulières. Le formalisme d'Ito est typiquement utilisé pour la représentation de systèmes dynamiques sujets à une excitation aléatoire qui varie rapidement au cours du temps, ce dont nous ne traitons pas dans cet ouvrage. De plus la théorie sur les SDE d'Ito est plus complexe que celle des SDE régulières qui en représentent un cas particulier [Gar85].

3.1.2 La résolution du modèle

Considérons la SDE régulière

$$\dot{y} = F(y) \quad \text{avec} \quad y \in \mathbb{R}_n \quad \text{et} \quad y(t_0) \sim \mathbf{y}_0 \quad (3.2)$$

où \mathbf{y}_0 est une variable aléatoire connue. La solution $y(t, y_0)$ du problème sera encore un processus stochastique de densité de probabilités $p(y(t, y_0))$. Une façon très usitée de résoudre les SDE

régulières est donnée par l'équation de Liouville [Gar85, Sob90] qui nous dit que la densité de probabilités de la solution satisfait :

$$\frac{\partial p(y,t)}{\partial t} + \sum_{i=1}^n \frac{\partial}{\partial y_i} [p(y,t) F_i(y)] = 0 \quad (3.3)$$

Cela nécessite alors de résoudre une PDE le long d'une trajectoire du système. Ce problème peut être contourné en transformant l'équation en une ODE

$$\frac{dp(y,t)}{dt} = -p(y,t) \operatorname{div} F(y) \quad (3.4)$$

$$\operatorname{div} F(y) = \sum_{i=1}^n \frac{\partial F_i}{\partial y_i} \quad (3.5)$$

Où div est la **divergence** du système.

Soit $\Omega(t)$ un volume arbitraire de l'espace contenant les solutions tel que :

$$\Omega(t + dt) = \{y(t + dt, y(t)) : y(t) \in \Omega(t), \dot{y} = F(y)\} \quad (3.6)$$

On peut alors interpréter la densité de probabilités d'un point dans l'espace comprenant la solution par la densité de probabilités des trajectoires du système dynamique passant en ce point. De plus, au fil du temps, la densité des trajectoires dépend de la divergence, de même que la taille du volume $\Omega(t)$. Dans le cas où la divergence est nulle, le système est dit **conservatif**. Autrement, il est dit **dissipatif**.

La solution d'une SDE sera alors un processus stochastique satisfaisant à la fois l'équation et ses propriétés probabilistes. La résolution nécessiterait alors une résolution numérique qui serait coûteuse en temps de calcul.

Un autre façon courante de résolution est la méthode de Monte-Carlo. C'est celle qui a été employée durant les expériences décrites dans cet ouvrage.

3.1.3 Simulation par Monte-Carlo

La simulation par Monte-Carlo (**MC**) d'une SDE régulière est constituée de la répétition de ces trois étapes [KK68] :

1. Un échantillonnage aléatoire de la densité de probabilités dans la distribution des conditions initiales,
2. Une série de simulations déterministes utilisant l'échantillon comme conditions initiales,
3. La description statistique de l'évolution de la densité de probabilités au moyen des résultats des différentes simulations.

3.2 Simulation par arithmétique d'intervalles

3.2.1 Intervalles classiques

Lorsque l'incertitude est représentée par un intervalle, la simulation par arithmétique d'intervalle nécessitera la résolution d'une **équation différentielle sur intervalle (IDE)**. Une IDE consiste en une ODE dont la valeur des paramètres ou des conditions initiales appartiennent à un intervalle. Soit une ODE paramétrique d'ordre m .

$$\dot{x} = F(x, k_1, \dots, k_p), \quad x \in \mathfrak{R}^n, \quad x(t_0) \in [x_{0,inf}, x_{0,sup}], \quad k_j \in \mathfrak{R}, \quad k_j \in [k_{j,inf}, k_{j,sup}], \quad j = 1, \dots, p. \quad (3.7)$$

où les conditions initiales appartiennent à l'intervalle $[x_{0,inf}, x_{0,sup}]$ et les paramètres incertains prennent leur dans les intervalles $[k_{j,inf}, k_{j,sup}]$. Soit $n = m + p$, le système (3.7) peut alors être réécrit sous la forme

$$\dot{y} = F(y), \quad y \in \mathfrak{R}^n, \quad y(t_0) \in [y_{inf}(t_0), y_{sup}(t_0)]. \quad (3.8)$$

En développant, nous obtenons

$$\begin{cases} \dot{y}_i = F_i(y) & y_i(t_0) \in [y_{i,inf}(t_0), y_{i,sup}(t_0)] & x_j = y_j & i = 1, \dots, m \\ \dot{y}_{m+j} = 0 & y_{m+j}(t_0) \in [k_{j,inf}(t_0), k_{j,sup}(t_0)] & y_{m+j} = k_j & j = 1, \dots, p \end{cases}$$

où

Nous obtenons alors un système de la forme (3.8) dont les paramètres sont précis et où l'incertitude affecte seulement les conditions initiales.

Une solution intervalle de l'IDE (3.8) sera définie par l'ensemble des solutions des ODE dont les conditions initiales appartiennent à $[y_{inf}(t_0), y_{sup}(t_0)]$.

De façon graphique, l'ensemble des intervalles définissant les conditions initiales et les paramètres formera un hypercube de dimension n qui sera appelé **région d'incertitude** du système au moment $t = t_0$. La simulation en utilisant les intervalles consistera à calculer au cours du temps l'évolution de cette région.

Une approche naïve est de résoudre numériquement le système d'IDE, d'une façon analogue à la résolution d'ODE mais où l'on remplace les opérateurs sur les nombres réels rencontrés dans la méthode numérique (p. ex. Euler). par les opérateurs de la mathématique d'intervalles. Cette technique est simple à implémenter et très peu coûteuse en temps de calcul. Malheureusement, l'erreur propagée à cause de la nécessité d'avoir les résultats réels des opérations compris dans les intervalles calculés croît très rapidement.

Le défaut de la mathématique d'intervalles dérive du fait qu'elle ne sait pas tenir compte de l'interaction qui est établie entre les variables du système d'équations différentielles. Voici ce que nous entendons par interaction entre les variables. Dans le cas $n = 2$, soient deux variables y_1 et y_2 qui prennent leur valeur respectivement dans les intervalles I_1 et I_2 , $I_1 = [A, B]$ et $I_2 = [C, D]$. Si x_1 n'est lié à aucune valeur comprise dans I_2 , alors nous disons que les variables sont **non-interactives**. Dans le cas contraire où $x_1 = a$ implique que x_2 prend sa valeur dans un intervalle compris dans I_2 , nous disons que les variables sont **interactives**.

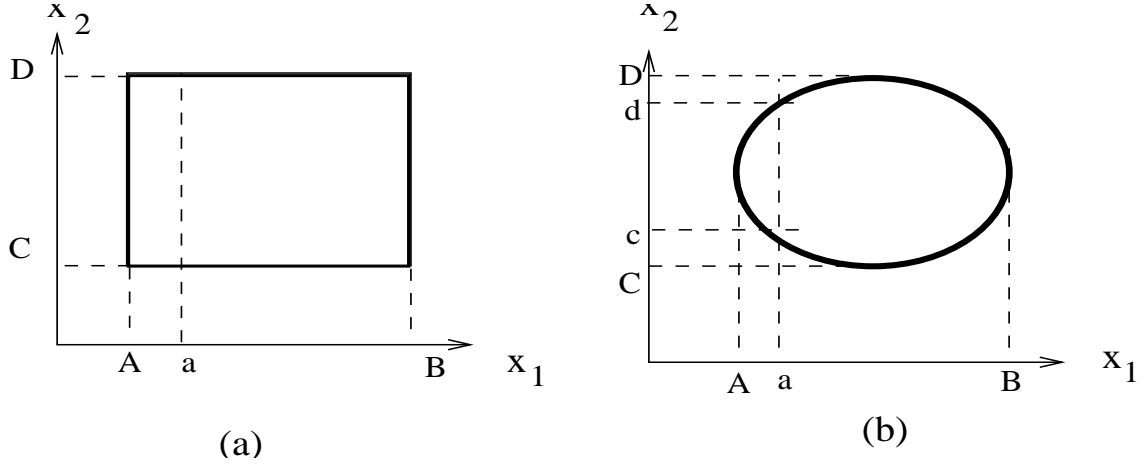


FIG. 3.2 – Relation entre deux variables $x_1 \in I_1$ et $x_2 \in I_2$. (a) variables non interactives (b) variables interactives

Prenons comme exemple

$$y \in I = [-1, 2] \quad (3.9)$$

si l'on ne tient pas compte de l'interaction

$$y - y \in [-1, 2] - [-1, 2] = [-3, 3] \quad (3.10)$$

alors que la solution exacte est

$$y - y = 0 = [0, 0] \quad (3.11)$$

Donc, si l'on ne tient pas compte de l'interaction des valeurs lors des calculs, cela entraîne l'introduction de trajectoires erronées qui ne correspondent à aucune solution correspondant aux solutions initiales. La figure 3.3 illustre bien le problème, représentant l'évolution de l'espace des solutions sur un système oscillant à deux dimensions dans laquelle les régions noires contiennent des trajectoires erronées.

3.2.2 Nombre affins

Pour diminuer la propagation de l'erreur décrite dans la section précédente, il peut être envisagé d'employer l'arithmétique affine. En employant la mathématique affine plutôt que la mathématique d'intervalles, les systèmes à résoudre seront de la forme

$$\dot{y} = F(y), \quad y \in \mathbb{R}^n, \quad y(t_0) = y_0(t_0) + \sum_{i=1}^s \varepsilon_i y_i(t_0) \quad (3.12)$$

où \mathbb{R} est l'extension des nombres réels à l'arithmétique affine. En développant, nous obtenons

$$\begin{cases} \dot{y}_i = F_i(y) & y_i(t_0) = y_i^0(t_0) + \sum_{ii=1}^s \varepsilon_{ii} y_i^{ii}(t_0) & i = 1, \dots, m \\ \dot{y}_{m+j} = 0 & y_{m+j}(t_0) = y_{m+j}^0(t_0) + \sum_{ii=1}^s \varepsilon_{ii} y_{m+j}^{ii}(t_0) & j = 1, \dots, p \end{cases}$$

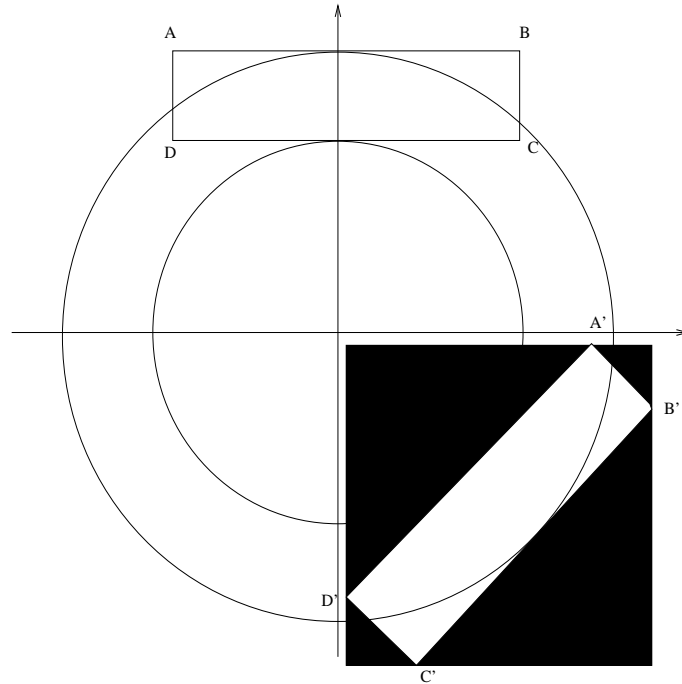


FIG. 3.3 – Introduction de trajectoires erronées dues à la non-interaction des variables, dans un système oscillant

Résoudre numériquement le système d'équations différentielles dont l'incertitude est sous forme de nombre affins consistera, de façon similaire aux IDE, à résoudre l'ODE de façon numérique mais en remplaçant les opérateurs des mathématiques sur les réels par leurs homologues en mathématique affine.

Les simulations effectuées dans l'étude expérimentale ont été réalisées en employant la méthode d'Euler pour la résolution du système d'équations différentielles.

3.3 Simulation par les nombres flous

3.3.1 Equation différentielle floue

Considérons maintenant une équation différentielle floue

$$\dot{y} = F(y) \quad y \in \mathbf{R}^n \quad y(t_0) \sim \mu_y(t_0) \equiv \mu_0 \quad (3.13)$$

où μ_0 est la condition initiale floue définie sur le domaine à n dimensions Y et \mathbf{R} est l'extension aux nombres flous de l'ensemble des réels.

L'équation (3.13) sera interprétée comme l'extension de la notion d'ODE conformément au principe (2.13). Et donc, une **équation différentielle floue** (FDE) sera définie comme une équation différentielle, où certaines conditions initiales ou paramètres sont incertains et sont

représentés par des nombres flous. La solution d'une FDE sera alors l'évolution au cours du temps de la distribution de possibilité par laquelle nous avons représenté l'incertitude.

Comme vu dans la section 2.3.2, le calcul sur les nombres flous peut être décomposé en plusieurs calculs sur une discrétisation des valeurs floues en leurs α -coupures, obtenant ainsi les α -coupures des résultats, ramenant le problème des nombres flous à un problème de mathématique d'intervalles. Nous ferons de même pour la résolution numérique des FDE : nous calculerons l'évolution au cours du temps des α -coupures des nombres flous constituant les conditions initiales et les paramètres, obtenant ainsi une succession d'IDE à résoudre. La figure 3.4 nous donne un exemple de résolution de FDE pour deux α -coupures d'un nombre flou y .

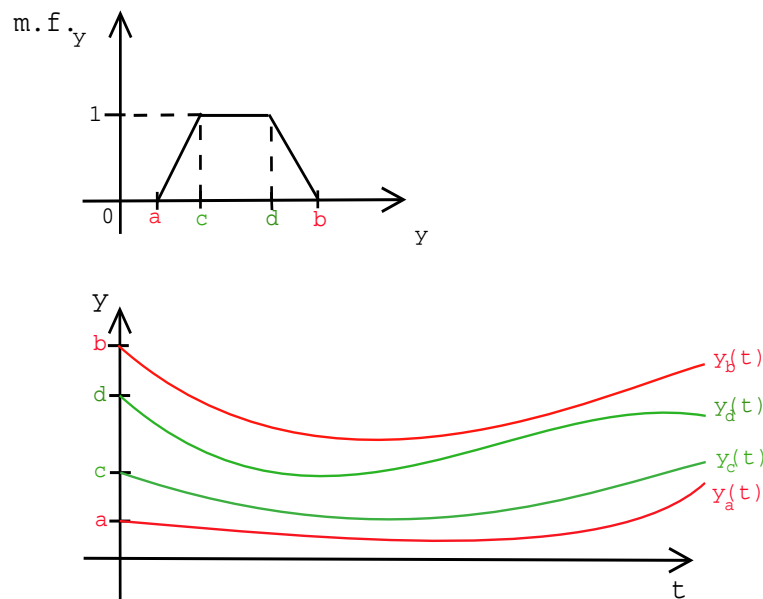


FIG. 3.4 – Résolution de $y(t)$ pour deux α -coupures de y

3.3.2 Simulation possibiliste vs. simulation probabiliste

La simulation par MC est l'approche la plus commune dans le cas où l'on utilise un modèle probabiliste. Cependant, dans le cas où l'on ne dispose que d'informations non-probabilistes sur le système à simuler, on peut remarquer que la tendance à utiliser également MC est grande [Fis91].

Malheureusement une simulation par MC pose le risque d'obtenir comme solution une distribution qui ne tient pas compte des trajectoires qui correspondent à des conditions initiales et des valeurs de paramètres qui sont peu probables mais tout à fait possibles. Il y a donc le risque avec MC de rater certaines trajectoires qui pourraient s'avérer importantes [Bon03b].

Ceci est particulièrement important dans le problème de détection de défauts ou bien la vérification qu'un système répond bien à certaines conditions critiques. Il peut s'avérer que la

probabilité de rencontrer un défaut ou bien de se retrouver dans des conditions critiques soit proche de zéro mais que la possibilité de les rencontrer soit significativement supérieure à zéro.

Chapitre 4

Simulation qualitative avec l'approche Qua.Si. (Qualitative Simulator)

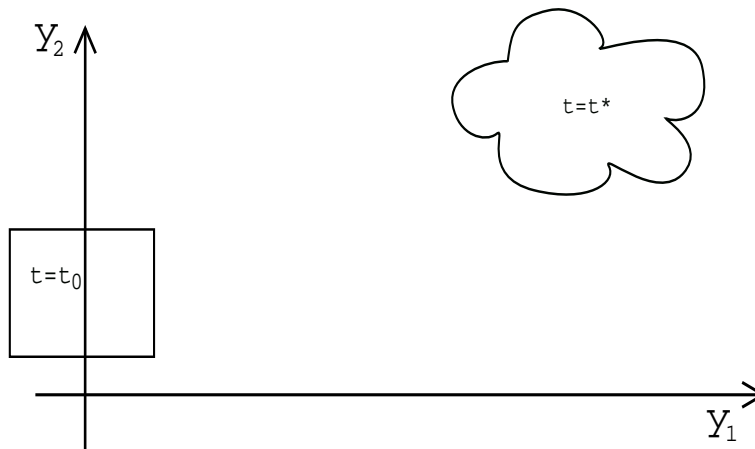
Nous allons parler ici de l'approche Qua.Si. (Qua.Si. tient pour Qualitative Simulator) [BB94, Bon96] qui va servir de base aux méthodes proposées dans le chapitre 5.

Qua.Si est une méthode pour la simulation d'équations différentielles floues basée sur la méthode des α -coupures. Cependant, comme vu dans la section 3.2.1, l'approche par intervalles pourrait entraîner l'explosion de l'incertitude de la solution. Ce problème de propagation et d'accroissement d'incertitude a été évité en le traitant comme un problème d'optimisation sur plusieurs variables, avec contraintes. La qualité des résultats fournis par cette méthode est donc alors comparable avec la qualité des résultats d'une méthode d'optimisation continue pour un problème non linéaire multivarié.

4.1 Recherche des extrema en tant que problème d'optimisation

Soit μ_0 les conditions initiales floues rectangulaires d'une FDE. Introduisons la notion de région d'incertitude. Si $n = 2$, la région d'incertitude initiale (pour $t = t_0$) sera définie graphiquement par le rectangle déterminé par les points dont les coordonnées sont obtenues en prenant les extrémités des deux intervalles constituant les conditions initiales rectangulaires. Pour $n = 3$, la région d'incertitude initiale sera définie graphiquement par un parallélépipède. En règle générale, la région d'incertitude initiale sera définie graphiquement par un hypercube de dimension n . Pour $t = t^*$, la région d'incertitude sera définie par la région limitée par les points solutions en $t = t^*$ de la FDE aux conditions initiales floues rectangulaires. La figure 4.1 illustre cette notion pour $n = 2$.

Il a été prouvé dans [BB94] qu'il suffit de calculer à chaque instant les points constituant la surface externe de la région d'incertitude pour obtenir l'évolution au cours du temps de cette région. Il nous reste à trouver les points constituant cette surface externe. L'idée proposée dans [Bon96] que le problème peut être amené à un problème d'optimisation continue. Trouver la

FIG. 4.1 – Région d'incertitude en t_0 et en t^*

solution numérique de notre équation consistera à trouver pour chaque instant t^* le minimum et le maximum des variables d'état y_i . Nous devons alors minimiser et maximiser à chaque instant t^* les fonctions $y_i(t^*, y_0)$. Un algorithme d'optimisation multivariée avec contraintes pourra alors être utilisé pour trouver les extrema des fonctions $y_i(t^*, y_0)$ où t^* est fixé et y_0 est la valeur sur laquelle l'algorithme d'optimisation va jouer. La valeur y_0 sera compris dans les limites de la surface externe de la région incertaine initiale.

Il est intéressant d'ajouter que dans le cas d'ODE linéaires, il est suffisant de connaître l'évolution des extrémités de la région d'incertitude initiale pour connaître les extrémités de l'évolution temporelle de cette région [Coo92].

4.2 L'algorithme

Les performances d'un algorithme d'optimisation multivariable avec contraintes peuvent être améliorées en donnant comme input additionnel le gradient de la fonction à minimiser (nous entendons ici par fonction la combinaison des fonctions solutions du système). Dans notre cas, il faudra fournir les dérivées au temps t^* de y_i en considération de y_0 . Obtenir ces dérivées est possible grâce à la notion de **matrice de connection** exposée dans [Moo66] décrite comme suit.

Soit $y(t, y_0)$ le vecteur solution du système d'équations

$$\dot{y} = F(y), \quad y \in \mathfrak{R}^n, \quad y(t_0) \in [y_{inf}(t_0), y_{sup}(t_0)]. \quad (4.1)$$

où y est une α -coupure de la fonction floue à simuler. La matrice de connection pour la solution $y(t, y)$ est définie comme suit par la matrice $C(t, y_0)$ avec les éléments :

$$C_{ij} = \left. \frac{\partial y_i(t, y)}{\partial y_j} \right|_{y=y_0} \quad (4.2)$$

Soit $J(t, y_0)$ la matrice Jacobienne du vecteur fonction évalué en $y(t, y_0)$ avec les éléments :

$$J_{ij}(t, y_0) = \left. \frac{\partial F_i(y)}{\partial y_j} \right|_{y=y_0} \quad (4.3)$$

[Moo66] a démontré que

$$\frac{\partial C(t, y_0)}{\partial t} = J(t, y_0)C(t, y_0) \quad (4.4)$$

avec $C(t_0, y_0)$ est la matrice identité. La matrice de connection va nous donner la sensibilité des composantes de la solution en fonction de faibles changements des conditions initiales. En associant le système d'équations avec l'ensemble des équations provenant de la matrice de connection, nous obtenons un système augmenté donnant les relations à chaque instant t^* entre les y_i et les valeurs des dérivées en fonction de la condition initiale. La combinaison donne alors la valeur ainsi que le gradient de la fonction à minimiser (ou maximiser).

La procédure de minimisation (ou maximisation) fonctionnera comme suit :

1. La routine d'optimisation choisit un y_0 de la surface externe de la région initiale d'incertitude.
2. La routine de résolution numérique prend y_0 comme condition initiale et retourne la valeur de $y(t^*, y_0)$ ainsi que du gradient.
3. L'algorithme s'arrête si la routine d'optimisation considère $y(t^*, y_0)$ comme un minimum (ou maximum), sinon une nouvelle valeur de y_0 est choisie en fonction des valeurs de $y(t^*, y_0)$ et du gradient.

La figure 4.2 donne un exemple de l'évolution de la surface extérieure de la région d'incertitude. On y voit bien l'utilité de la recherche des extrema. La condition initiale donnant le minimum pour une composante peut très bien se retrouver au milieu de la composante de la surface.

La table 4.1 nous donne l'algorithme Qua.Si. en pseudo-code.

Soit n le nombre de variables d'état

$\forall t^*, t^*$ allant de l'instant de départ à l'instant final

\forall variable y_i

$$y_{i \min}(t^*) = \infty$$

$$y_{i \max}(t^*) = -\infty$$

\forall composante y_{0j} de la surface externe de la région initiale, $j \in \{1 \dots 2^n\}$

$$y_{ij \min}(t^*) = \min(\{y_i(t^*, y_0)\} \mid y_0 \in y_{0j} \text{ et } y_i(t^*, y_0) \text{ est une solution du système}$$

$$y_{ij \max}(t^*) = \max(\{y_i(t^*, y_0)\} \mid y_0 \in y_{0j} \text{ et } y_i(t^*, y_0) \text{ est une solution du système}$$

$$y_{i \min}(t^*) = \min(y_{i \min}(t^*), y_{ij \min})$$

$$y_{i \max}(t^*) = \max(y_{i \max}(t^*), y_{ij \max})$$

TAB. 4.1 – Qua.Si. en pseudo-code

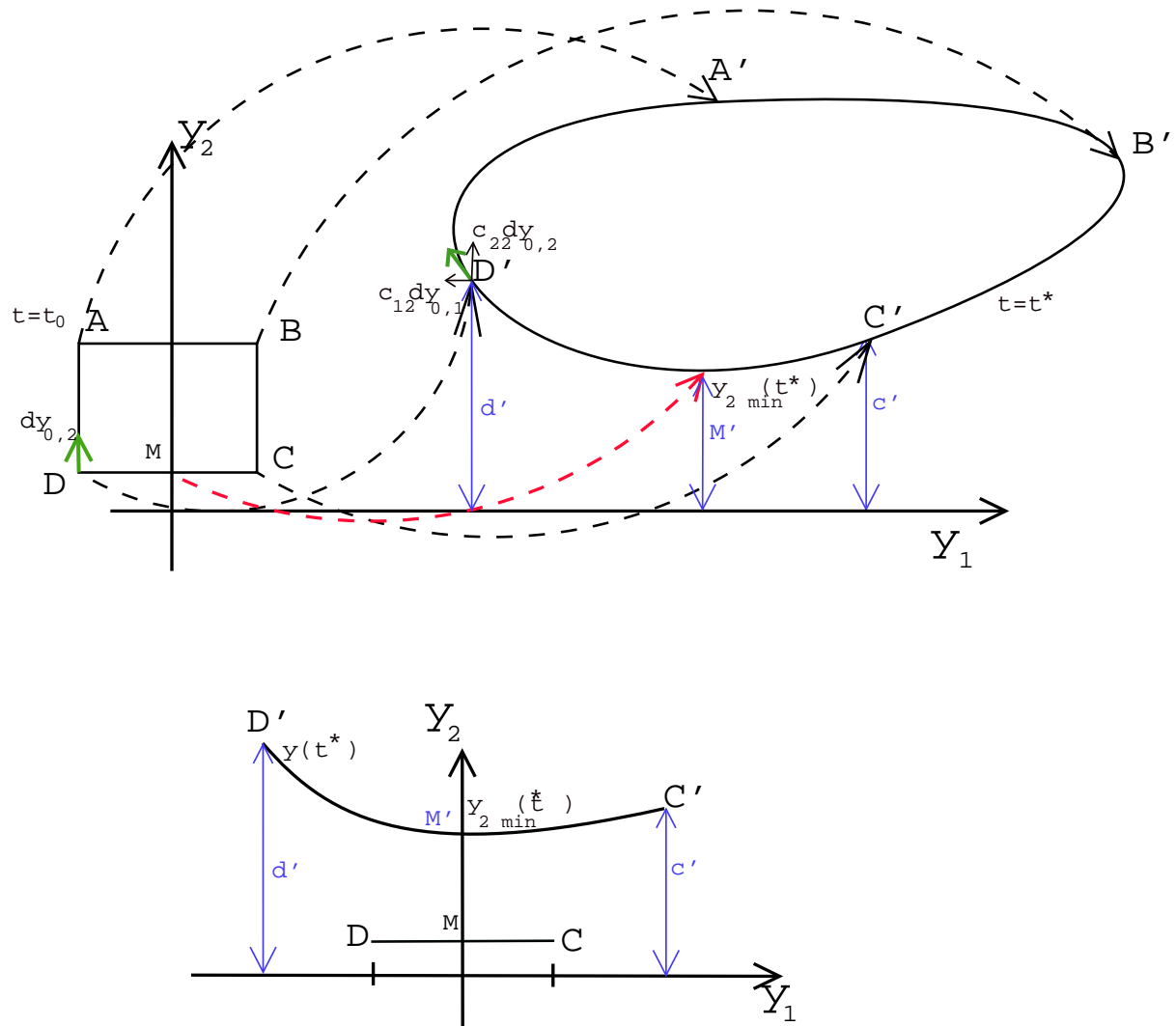


FIG. 4.2 – Evolution de la surface extérieure de la région d'incertitude

4.3 Note sur l'implémentation

L'algorithme décrivant l'approche Qua.Si. a été implémenté en Matlab¹ [Bon96].

Le programme utilise comme procédure de résolution de système d'ODE, la fonction *ode23* de Matlab qui correspond à l'application d'une méthode de Runge-Kutta.

La routine d'optimisation est la fonction d'optimisation avec contraintes *constr* fournie avec Matlab. Si la taille du problème le permet, le programme qui implémente l'algorithme offre la possibilité de diminuer les risques de trouver un extremum local éloigné de l'extremum global en lançant une ou trois fois par étape du programme la routine d'optimisation. Cela peut s'avérer utile lorsque les valeurs extrêmes sont difficiles à trouver, comme nous le verrons dans la section 7.2.1 ou encore dans le cas où des ODE du système sont linéaires.

Dans le cas où un seul appel à *constr* est employé, la routine recevra comme point de départ pour les indices d'échantillonnages de la condition initiale le centre de la composante y_{0j} courante. Si trois appels sont effectués, après avoir trouvé un résultat avec le centre de la composante externe, *constr* recevra, comme point de départ pour les indices d'échantillonnages de la condition initiale, chacune des extrémités de la composante y_{0j} courante à chacun des deux appels restants.

Nous dénoterons respectivement par **un appel** et **3 appels** ces deux possibilités d'exécution de l'algorithme.

4.4 Vue critique

4.4.1 Aspects positifs

- L'algorithme tient compte de l'interaction entre les variables (section 4.1, 3.2.1), la région solution ne contiendra aucune trajectoire ne correspondant pas à une solution du système d'ODE
- Dans le cas où les valeurs trouvées par l'algorithme d'optimisation sont les extrema globaux, nous obtenons les bonnes valeurs pour déterminer les points de la surface externe de la région d'incertitude.
- L'erreur par rapport aux vraies solutions ne dépend plus de la méthode mais des outils numériques employés

4.4.2 Aspects négatifs

- La qualité des solutions dépend de celle de la routine d'optimisation
- Dans le cas où les valeurs trouvées par l'algorithme d'optimisation sont des extrema locaux, la région trouvée comme solution ne contiendra pas toutes les trajectoires.

1. <http://iridia.ulb.ac.be/~gbonte/software/Quasi/Quasi.html>

- La méthode est très lourde en temps de calcul : elle nécessite une quantité exponentielle en le nombre de variable d'états d'appels à la routine d'optimisation, qui va elle-même appeler plusieurs fois la procédure de résolution d'ODE. De plus ces appels seront faits à chaque instant t^* , donc au plus l'algorithme progressera d'une étape sur l'intervalle de temps sur lequel on fait le calcul, plus la résolution d'ODE prendra du temps et donc aussi l'optimisation.

Deuxième partie

Apports au domaine

Chapitre 5

Extension de l'approche Qua.Si. avec des méthodes stochastiques d'optimisation

Ce chapitre propose d'améliorer Qua.Si en ajoutant des méthodes stochastiques d'optimisation à l'algorithme, de façon à poursuivre deux buts :

1. Faire en sorte que l'algorithme d'optimisation trouve des valeurs plus proches des extrema globaux en lui fournissant des meilleures valeurs de départ.
2. Accélérer l'exécution en évitant de parcourir un nombre exponentiel de composantes de la surface externe de la région d'incertitude.

De façon résumée, la procédure se fera en deux grandes parties

1. On recherche de bonnes conditions initiales de départ en lançant une méthode stochastique de simulation, non basée sur les gradients des équations.
2. Pour chaque instant, pour chaque variable d'état, on minimise et maximise la résolution du système en donnant, comme valeur de départ, à la routine d'optimisation basée sur les gradients les conditions initiales trouvées à l'étape précédente.

Les méthodes d'optimisation stochastiques envisagées ici sont une recherche aléatoire implémentée par Monte-Carlo et une autre par Simulated Annealing. De plus, le canevas utilisé pourrait être réutilisé en employant d'autres méthodes. On peut envisager d'employer un algorithme inspiré d'autres heuristiques (recherche taboue, recherche locale, recherche guidée par méthodes d'apprentissage, . . .).

5.1 Monte-Carlo

Dans cette section, nous allons effectuer une recherche aléatoire par MC avant de faire la simulation avec optimisation, nous appellerons la combinaison des deux parties **Qua.Si.+MC**.

La partie MC consistera à la recherche des conditions initiales, pour chaque t^* , qui nous donnent, pour chacune de variables d'état, des conditions initiales proches de celles qui peuvent fournir les valeurs extrêmes. Nous échantillonnerons alors, de façon uniforme dans l'intervalle qui les contient, les conditions initiales y_0 . A chaque échantillonnage, nous fournirons les valeurs qui ont été tirées au hasard à la routine de résolution numérique d'ODE. Nous allons ensuite comparer, pour chaque variable y_i , chacun des $y_i(t^*, y_0)$ constituant la trajectoire calculée avec $\hat{y}_i(t^*, \hat{y}_{t^*, i}^0)$ donnant le minimum courant trouvé pour t^* et avec $\check{y}_i(t^*, \check{y}_{t^*, i}^0)$ donnant le maximum courant trouvé pour t^* . Si nous obtenons des valeurs améliorant respectivement le minimum ou bien le maximum de la variable y_i , nous les sauverons respectivement comme nouveau minimum ou maximum courant, de même que la condition initiale correspondante. Pour borner inférieurement la qualité des conditions initiales obtenues, avant de procéder aux échantillonnages, nous allons résoudre l'ODE en prenant de façon déterministe les conditions initiales : nous fournirons à la procédure de résolution les extrémités des 2^n composantes de la surface d'incertitude initiale. De plus, cela nous permettra d'obtenir les valeurs exactes si le système simulé contient des équations linéaires [Coo92].

En résumé, la méthode consistera à fusionner entre elles les trajectoires qui ont été trouvées. La région d'incertitude qui constitue la solution est déterminée \hat{y}_i et \check{y}_i .

A la fin de la partie MC, nous obtiendrons alors pour chacun des t^* , pour chacune des variables d'état y_i , des valeurs de départ à fournir à l'algorithme d'optimisation qui seront relativement proches des extrema globaux. De plus, ces conditions initiales se trouvent déjà dans la bonne composante de la surface d'incertitude. Nous n'aurons alors plus besoin d'effectuer les 2^n appels par variable d'état, par instant, à la procédure d'optimisation. La tables 5.1 et 5.2 montrent respectivement le pseudo-code de l'algorithme pour la partie MC et la partie optimisation basée sur les gradients.

Nous aurons également besoin d'un paramètre supplémentaire à considérer : le nombre d'échantillonnages qui va être effectué. Il va falloir le choisir de façon à avoir un bon rapport qualité/vitesse d'exécution pour le calcul des nouveaux points de départ de l'optimisation.

Soit n le nombre de variables d'état

$\forall y_i \forall t^*$

$$\hat{y}_i(t^*, \hat{y}_{t^*, i}^0) = \infty$$

$$\check{y}_i(t^*, \check{y}_{t^*, i}^0) = -\infty$$

\forall extrémité y_0 de la surface initiale

$\forall y_i$

$y_i(t^*, y_0) \leftarrow$ la solution numérique du système pour y_0

$\forall t^*$

si $y_i(t^*, y_0) < \hat{y}_i(t^*, \hat{y}_{t^*, i}^0)$ (*On améliore le minimum*)

$$\hat{y}_i(t^*, \hat{y}_{t^*, i}^0) \leftarrow y_i(t^*, y_0)$$

$$\hat{y}_{t^*, i}^0 \leftarrow y_0$$

sinon si $y_i(t^*, y_0) > \check{y}_i(t^*, \check{y}_{t^*, i}^0)$ (*On améliore le maximum*)

$$\check{y}_i(t^*, \check{y}_{t^*, i}^0) \leftarrow y_i(t^*, y_0)$$

$$\check{y}_{t^*, i}^0 \leftarrow y_0$$

répéter

tirer y_0 de façon aléatoire uniforme dans la région d'incertitude

$\forall y_i$

$y_i(t^*, y_0) \leftarrow$ la solution numérique du système pour y_0

$\forall t^*$

si $y_i(t^*, y_0) < \hat{y}_i(t^*, \hat{y}_{t^*, i}^0)$ (*On améliore le minimum*)

$$\hat{y}_i(t^*, \hat{y}_{t^*, i}^0) \leftarrow y_i(t^*, y_0)$$

$$\hat{y}_{t^*, i}^0 \leftarrow y_0$$

sinon si $y_i(t^*, y_0) > \check{y}_i(t^*, \check{y}_{t^*, i}^0)$ (*On améliore le maximum*)

$$\check{y}_i(t^*, \check{y}_{t^*, i}^0) \leftarrow y_i(t^*, y_0)$$

$$\check{y}_{t^*, i}^0 \leftarrow y_0$$

renvoyer $\{\hat{y}^0\}$ et $\{\check{y}^0\}$

TAB. 5.1 – Pseudo-code de la partie MC

Soit n le nombre de variables d'état

Soient $\{\hat{y}^0\}$ et $\{\check{y}^0\}$ fournis par l'optimisation non basée sur les gradients

$\forall t^*, t^*$ allant de l'instant de départ à l'instant final

\forall variable y_i

$$y_{i \min}(t^*) = \infty$$

$$y_{i \max}(t^*) = -\infty$$

Donner $\hat{y}_{t^*,i}^0$ comme point de départ de la routine d'optimisation

$$y_{ij \min}(t^*) = \min(\{y_i(t^*, y_0)\}) \mid y_0 \in y_{0j} \text{ et } y_i(t^*, y_0) \text{ est une solution du système}$$

Donner $\check{y}_{t^*,i}^0$ comme point de départ de la routine d'optimisation

$$y_{ij \max}(t^*) = \max(\{y_i(t^*, y_0)\}) \mid y_0 \in y_{0j} \text{ et } y_i(t^*, y_0) \text{ est une solution du système}$$

$$y_{i \min}(t^*) = \min(y_{i \min}(t^*), y_{ij \min})$$

$$y_{i \max}(t^*) = \max(y_{i \max}(t^*), y_{ij \max})$$

TAB. 5.2 – Pseudo-code de la partie d'optimisation basée sur les gradients

5.2 Simulated Annealing

Plutôt que d'employer une méthode de recherche aléatoire pour trouver des valeurs approchant les extrema des fonctions solutions, nous proposons employer une méthode conçue pour trouver les extrema des fonctions, d'où l'idée d'employer une méthode stochastique spécialisée dans la minimisation de fonctions. Supposons que cette dernière fonctionne bien et que la routine d'optimisation, basée sur les gradients, de la seconde partie est fort sensible à la qualité des conditions initiales qui lui sont fournies comme point de départ. Nous pouvons espérer accélérer le temps de calcul de l'optimisation et nous aurons une meilleure qualité des valeurs trouvées. Comme cela, nous aurons une couverture plus grande des trajectoires. Parmi de telles méthodes stochastiques, on nous focaliserons sur la méta-heuristique **Simulated Annealing** (que nous dénoterons **SA**).

5.2.1 Introduction à Simulated Annealing

Le nom de cette méta-heuristique provient de la technique de recuit en physique des matériaux [MRR⁺53]. Le but de cette technique est de trouver les états d'énergie minimale d'objets qui sont passés de l'état liquide à solide. En phase liquide, les atomes sont répartis de façon aléatoire, mais si l'on passe rapidement de l'état liquide à solide, les atomes resteront dans la même configuration. Or une fois solidifié, on risque d'être dans une configuration à haute énergie, donnant des tensions à l'intérieur de l'objet qui pourront être sources de défauts. Pour remédier à cela, l'état d'énergie minimale va être trouvé en recuisant l'objet puis en le laissant refroidir très lentement, de façon à ce que les atomes aient le temps de se retrouver dans une configuration d'énergie minimale [MRR⁺53].

Il est montré dans [Cer85, KGV83] que l'on peut faire l'analogie à l'optimisation de fonctions. Les différents états de l'objet représentent les solutions acceptables du problème et l'énergie correspond à la fonction à minimiser. L'algorithme général de SA pourrait être décrit comme suit :

1. On applique un algorithme de descente.
2. Une fois arrivé à un minimum local, on essaye d'en sortir en prenant une solution au hasard, selon une distribution de Boltzmann.
3. On garde une valeur appelée **Température** (T).
4. A intervalles réguliers (de taille L), on multiplie la température par un coefficient (α) de décroissance.
5. On répète le processus.

Voici quelques détails supplémentaires. La **distribution de Boltzmann** est donnée par :

$$e^{-\frac{\Delta F}{T}} \quad (5.1)$$

où ΔF vaut la variation de la fonction solution entre deux itérations. Comme vu dans le point 4, T va suivre une décroissance géométrique par paliers. Ces paliers auront une taille L . Nous

aurons une température initiale $T = T_0$ pendant L paliers, et à chaque palier, on modifie T selon $T \leftarrow \alpha T$ ($0 < \alpha < 1$). Il nous reste à parler des paramètres à fournir. Il faut choisir α et L de façon judicieuse en fonction de la taille du problème. T_0 est trouvé de façon à ce que $e^{-\frac{\langle \Delta F \rangle}{T_0}} = \tau$. $\langle \Delta F \rangle$ vaut la variation moyenne de F obtenue par simulation, et τ est le taux d'acceptation moyen initial des solutions plus mauvaises. Enfin, il nous faut une bonne taille pour le voisinage sur lequel échantillonner les valeurs. Le tout nous donne le pseudo-code de la table 5.3.

```
 $T \leftarrow t_0$ 
Soit une valeur initiale  $x^*$ 
 $\hat{F} = \infty$ 
répéter
  Soit le compteur  $cpt$ ,  $cpt \leftarrow 0$ 
  répéter
    Prendre de façon aléatoire  $x$  dans le voisinage de  $x^*$ 
     $\Delta F \leftarrow F(x) - F(x^*)$ 
    si  $\Delta F < 0$  (On améliore la trajectoire précédente)
       $x^* \leftarrow x$  (On met à jour le voisinage)
      si  $F(x) < \hat{F}$  alors  $\hat{F} \leftarrow F(x)$  (On améliore la fonction à minimiser)
    sinon si  $\text{random}(0,1) < e^{-\frac{\Delta F}{T}}$  alors  $x^* \leftarrow x$  (On met à jour le voisinage)
     $cpt \leftarrow cpt + 1$ 
  tant que  $cpt < L$ 
   $T \leftarrow \alpha T$ 
tant que le critère d'arrêt n'est pas rencontré
```

TAB. 5.3 – Pseudo-code général pour SA pour la minimisation de fonctions

Il nous reste à parler de la condition d'arrêt : on peut imaginer d'arrêter l'algorithme après un certain nombre d'échantillonnages, lorsque l'amélioration de la fonction est inférieure à une certaine quantité ou bien encore, plus spécifiquement à SA, lorsque l'on n'a pas amélioré la fonction durant un certain nombre de paliers (le système est alors **gelé**).

Une caractéristique intéressante est que l'algorithme SA peut être vu comme un ensemble de chaînes de Markov homogènes pouvant être ramenée à une seule chaîne de Markov inhomogène qui converge, sous certaines conditions, asymptotiquement vers le minimum global de la fonction à minimiser [MRSV86].

5.2.2 Application à notre problème

L'algorithme de SA, dans sa forme générale, ne nous convient pas pour remplacer MC comme partie préliminaire à la partie d'optimisation. Nous allons donc nous inspirer de SA de façon à obtenir, de façon économique en temps de calcul, des conditions initiales proches. En effet, le but ici n'est pas de remplacer la routine d'optimisation mais de trouver une heuristique qui de façon plus intéressante que MC. Le pseudo-code de la démarche expliquée dans cette section se trouve dans les tables 5.5 et 5.6.

Tout d'abord, similairement à MC, pour borner inférieurement la qualité des résultats et pour obtenir directement les bonnes valeurs en cas d'équations linéaires, nous allons résoudre les équations en prenant comme conditions initiales les extrémités des composantes externes de la région initiale d'incertitude [Coo92].

Nous allons ensuite allier à l'idée de fusion des trajectoires la gestion des voisinages.

Nous allons effectuer des comparaisons pour chaque variable d'état, à chaque instant, similairement à MC. comme l'heuristique développée ici s'inspire de SA, il va falloir effectuer en plus les tests sur la distribution de Boltzmann. Les comparaisons et les tests se font à l'aide de conditions initiales tirées au sort dans le voisinage d'un point dans la région d'incertitude initiale. Le problème va se poser sur la gestion de ce voisinage. En effet, dans le canevas général de SA, trois cas sont possibles à l'issue des comparaisons et des test :

1. La valeur x tirée au sort donne une valeur de $F(x)$ meilleure que $F(x^*)$ où x^* est le centre du voisinage courant ; lors de la prochaine itération, cette même valeur x va devenir le centre du nouveau voisinage.
2. La valeur tirée au sort donne un moins bon résultat et le test de Boltzmann nous dit de prendre cette valeur comme centre du voisinage de l'itération suivante.
3. La valeur tirée au sort donne un moins bon résultat et le test de Boltzmann nous dit de garder le même voisinage.

Dans le problème qui nous intéresse, après avoir résolu l'équation différentielle avec la condition initiale tirée au sort, on peut se retrouver dans différentes situation pour la trajectoire qui a été calculée et la condition initiale correspondante :

1. La trajectoire $y_i(t)$ en certains points peut être meilleure que la trajectoire précédente $\hat{y}_i(t)$. Ces points feront alors partie de la trajectoire $\hat{y}_i(t)$ lors de la prochaine itération. Il va falloir prendre comme nouveau centre de voisinage la condition initiale correspondant à cette trajectoire.
2. La trajectoire générée $y_i(t)$ est en certains points moins bonne que la trajectoire précédente $\hat{y}_i(t)$ mais le test de Boltzmann nous dit de prendre malgré tout la condition initiale correspondante comme prochain centre de voisinage.
3. La trajectoire générée $y_i(t)$ est en certains points moins bonne que la trajectoire précédente $\hat{y}_i(t)$ mais le test de Boltzmann nous dit de garder l'ancien voisinage.

Pour certains points de la trajectoire, il nous faudra donc changer de voisinage, et pour d'autres nous devons le conserver.

Pour résoudre ce problème et rester le plus proche possible de l'algorithme de base de SA, il nous faudrait alors garder pour chaque variable d'état y_i , pour chaque instant t^* , un voisinage et effectuer un tirage au sort et une résolution d'équation différentielle pour chacun. Cela serait alors dispendieux en temps et nécessiterait le stockage des informations sur chacun de ces intervalles. Pour éviter cela, l'heuristique présentée ici va se contenter de conserver pour chaque variable d'état les moments où l'on a changé de voisinage et les instants où l'on a gardé le voisinage. Cela sera représenté dans le pseudo-code par les ensembles \tilde{t}_i et \bar{t}_i qui correspondent respectivement aux instants de la variable d'état y_i pour lesquels on va prendre un nouveau voisinage et les instants de la variable d'état y_i pour lesquels on va garder le même voisinage. A chaque itération de l'algorithme, nous aurons alors deux voisinages par variable d'état. L'ancien voisinage dont le centre est dénoté par \bar{y}_0^i et le nouveau dont le centre est dénoté par \tilde{y}_0^i .

Nous avons deux voisinages. Nous obtiendrons donc deux trajectoires après la résolution numérique d'équation différentielle. Cependant, nous ne devons comparer qu'une seule trajectoire avec la trajectoire $\hat{y}_i(t)$ obtenue à l'itération précédente. Nous devons donc reconstituer cette trajectoire à partir de la résolution du système d'ODE pour chacune des deux conditions initiales $\{\tilde{y}_0\}$ et $\{\bar{y}_0\}$. Nous prendrons alors les points correspondant aux instants tels qu'ils sont dans les ensembles \tilde{t}_i et \bar{t}_i .

Il est à noter que nous n'aurons jamais deux voisinages qui coïncident, comme le centre du voisinage à l'itération suivante est la condition initiale de l'itération courante. Cependant, nous pouvons nous retrouver, par exemple dans le cas d'équations linéaires, avec un ensemble \tilde{t}_i ou \bar{t}_i vide. Il ne sera alors nécessaire que d'effectuer une seule résolution d'équation différentielle.

Prenons un exemple avec une seule variable pour illustrer le concept. Supposons que nous devons traiter les instants $t = 1 \dots 8$. Les comparaisons et les tests nous ont donné la situation suivante : on va prendre la valeur qui a été tirée au sort pour les instants 1, 3 et 7 et on va garder l'ancien centre de voisinage pour les instants 2, 4, 5, 6 et 8. Nous aurons donc $\tilde{t} = \{1,3,7\}$ et $\bar{t} = \{2,4,5,6,8\}$. Cette situation est illustrée par la figure 5.1 dans le cas d'une recherche du minimum, avec une seule variable. Notons que sur l'exemple présenté, $\hat{y}(t) = \tilde{y}(t)$ pour $t = 5$ et $t = 6$ et que l'on a trouvé une valeur meilleure que le minimum courant $\hat{y}(t)$ en $t = 7$ (le minimum courant va alors être mis à jour).

Passons à l'itération suivante. Une fois $\{\tilde{y}\}$ et $\{\bar{y}\}$ calculés, la trajectoire va être reconstituée avec les points de $\{\tilde{y}\}$ correspondant aux instants 1, 3 et 7 (l'ensemble \tilde{t}) et les points de $\{\bar{y}\}$ correspondant aux instants 2, 4, 5, 6 et 8 (l'ensemble \bar{t}). Cette situation est illustrée par la figure 5.2. Notons que, comme $y(t) = \bar{y}(t)$, entre autres, pour les instants consécutifs 4,5 et 6, les deux courbes sont confondues sur l'intervalle de temps [4, 6].

Nous pouvons enfin effectuer les comparaisons et les tests pour trouver une meilleure solution et établir le nouveau voisinage (cf. figure 5.3).

Nous allons comparer les points de la trajectoire y_i avec les points de \hat{y}_i (la trajectoire

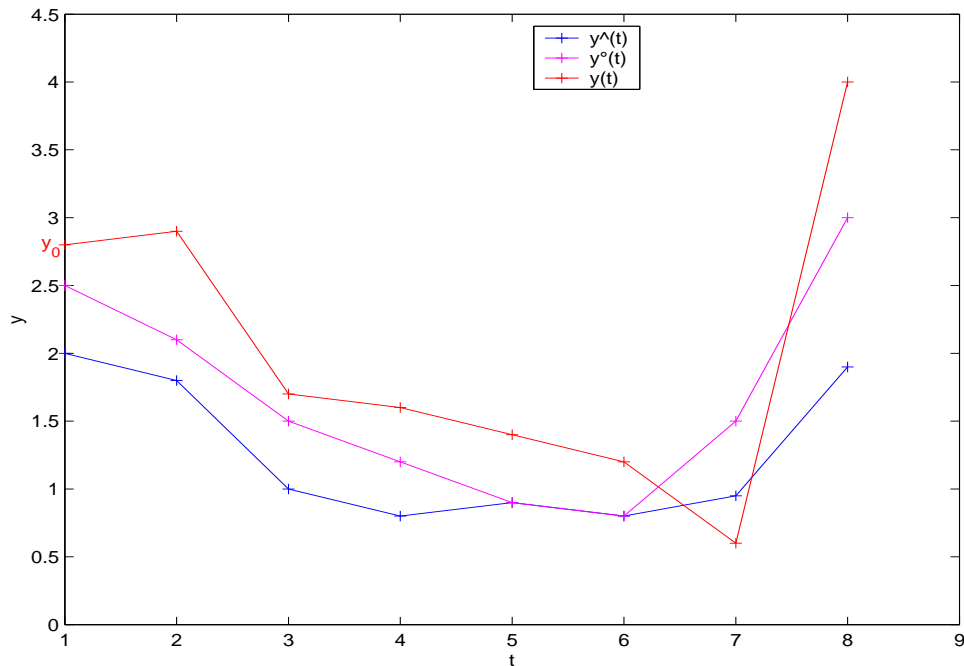


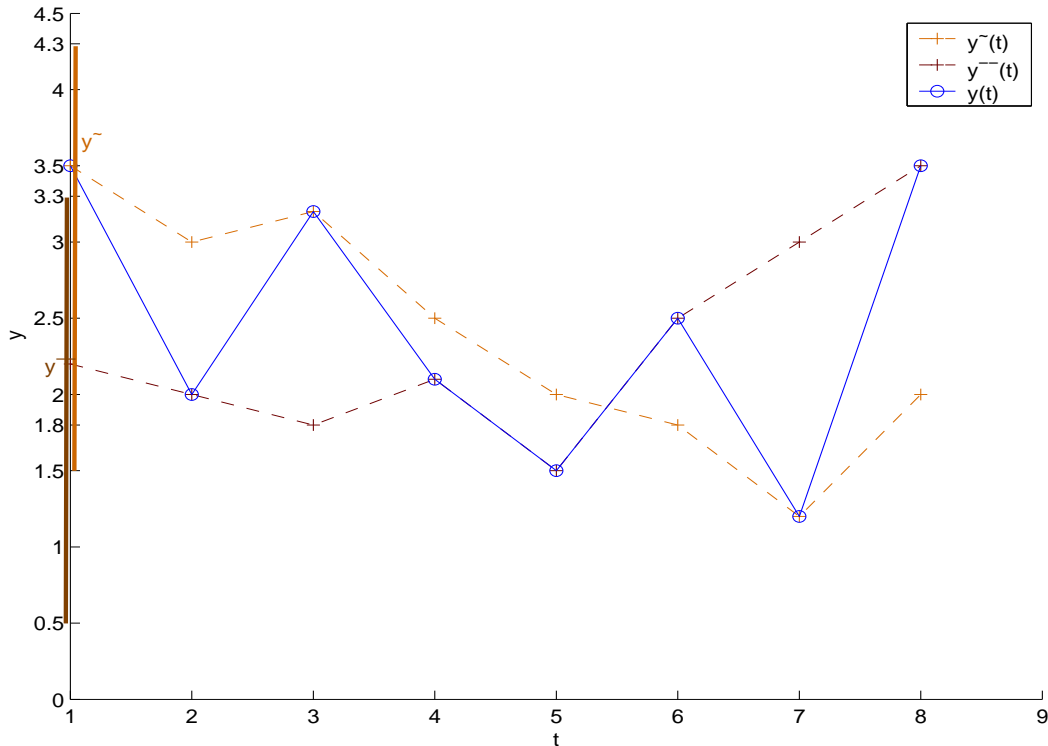
FIG. 5.1 – Trajectoire courante $y(t)$, trajectoire précédente $\hat{y}(t)$ et courbe du minimum courant $\tilde{y}(t)$, à l'itération k

trouvée lors de l'itération précédente) et effectuer la fusion de ces deux trajectoires. Pour chacun des nouveaux points de la nouvelle trajectoire \hat{y}_i , il nous faut vérifier si ceux-ci n'améliorent pas les meilleures valeurs trouvées (\hat{y}_i pour la recherche du minimum, et \tilde{y}_i pour la recherche du maximum). Pour les points qui donnaient une valeur moins bonne que celle trouvée à l'itération précédente, nous allons effectuer les tests avec la distribution Boltzmann. Nous pourrions ainsi établir pour quels points garder l'ancien voisinage et pour lesquels prendre le nouveau voisinage.

La solution de l'équation qui va être trouvée sera ici encore la région d'incertitude délimitée par \hat{y}_i et \tilde{y}_i .

Il reste à donner quelques précisions sur le pseudo-code :

- Lors de l'initialisation des variables précédant les itérations des recherches du minimum et du maximum, pour avoir un résultat cohérent, il faut que pour tout y_i , l'un des deux ensembles soit vide : on ne peut avoir d'instant t^* appartenant aux deux ensembles, et la trajectoire à considérer lors de la première itération ne proviendra que d'un seul voisinage.
- Les $\hat{y}_i(t^*)$ correspondent à la valeur de la variable d'état y_i à l'instant t^* lors de l'itération précédente.
- T est la température courante du système. Toutes les L itérations, sa valeur sera multipliée par le paramètre de simulation α .

FIG. 5.2 – Reconstitution de la trajectoire à comparer, à l'itération $k + 1$.

- Similairement à MC (section 5.1), les $\hat{y}_i(t^*)$ et les $\check{y}_i(t^*)$ correspondent respectivement aux minima et maxima trouvés.
- Similairement à MC (section 5.1), les \hat{y}^0 et les \check{y}^0 correspondent respectivement aux conditions initiales donnant les minima et les maxima.
- La recherche des conditions initiales à fournir à l'algorithme d'optimisation de la seconde partie du processus se fera de la façon résumée dans la table 5.2.
- Le raisonnement va nous donner le pseudo-code des tables 5.4, 5.5 et 5.6 qui correspondent respectivement au pseudo-code de l'initialisation de conditions initiales aux extrémités de la surface, à la recherche d'un minimum acceptable et la recherche du maximum acceptable. L'utilité du code de la table 5.4 sera, comme pour MC, de borner inférieurement la qualité des conditions initiales qui seront fournies à l'optimisation. Il suffira par la suite d'appliquer le code de la table 5.2

La condition d'arrêt serait ici l'accomplissement d'un certain nombre d'échantillonnages ou bien de paliers, l'objectif n'est pas de trouver le minimum mais une valeur relativement proche. Nous aurons donc, comme paramètres à fournir, la température initiale pour le minimum, la température initiale pour le maximum, le coefficient de décroissance, la taille des paliers, la taille du voisinage dans lequel échantillonner et enfin le nombre d'échantillonnages à effectuer. Dans le cas où la simulation du minimum se comporte différemment de la simulation du maximum

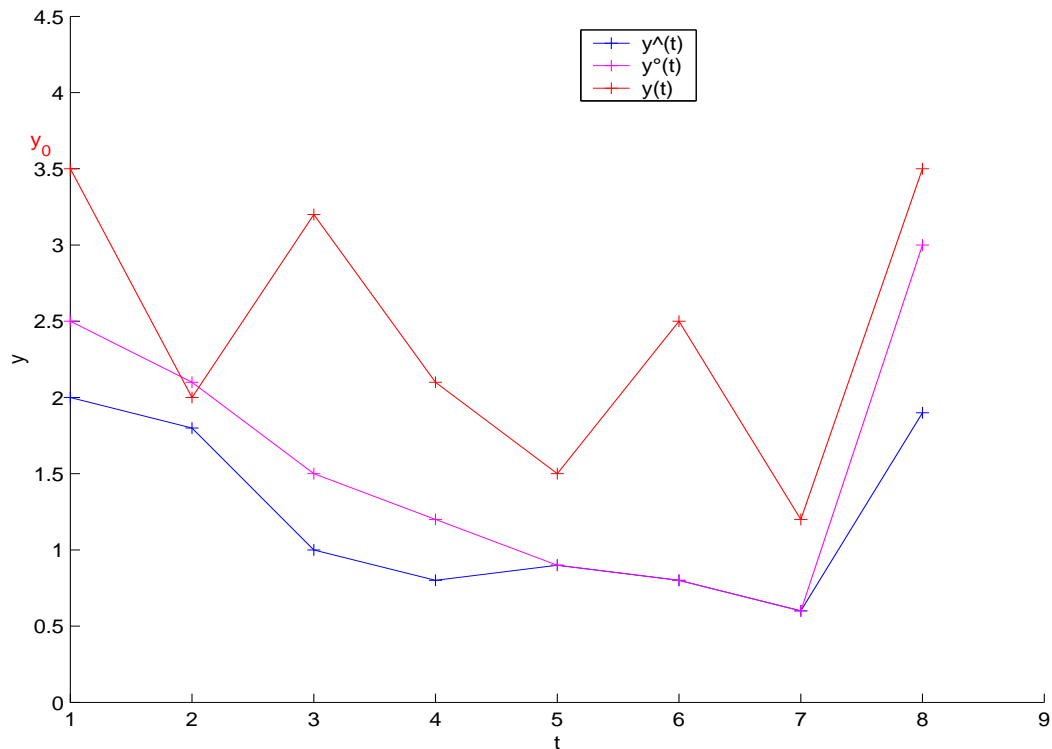


FIG. 5.3 – Trajectoire courante $y(t)$, trajectoire précédente $\hat{y}(t)$ et courbe du minimum courant $\hat{y}(t)$, à l'itération $k + 1$

en termes de qualité de solution, on pourrait envisager de prendre des paramètres différents pour chacune des deux parties. Nous avons donc un nombre élevé de paramètres à fournir qui peuvent influencer sur la proximité entre les conditions initiales trouvées par l'algorithme et celles donnant les extrema des fonctions. Le réglage de ces paramètres rend donc délicate l'utilisation de l'algorithme, ce qui contraste fort avec l'utilisation de MC où l'on ne doit se soucier que du temps que va prendre la simulation. Il faut également ajouter que pour avoir la température initiale du système, on a besoin de connaître la variation moyenne de la fonction. Il conviendra donc de faire une simulation préalable avec une autre méthode (MC par exemple) si on ne connaît pas cette valeur.

Soit n le nombre de variables d'état

$\forall y_i \forall t^*$

$$\hat{y}_i(t^*, \hat{y}_{t^*, i}^0) = \infty$$

$$\check{y}_i(t^*, \check{y}_{t^*, i}^0) = -\infty$$

\forall extrémité y_0 de la surface initiale

$\forall y_i$

$y_i(t^*, y_0) \leftarrow$ la solution numérique du système pour y_0

$\forall t^*$

si $y_i(t^*, y_0) < \hat{y}_i(t^*, \hat{y}_{t^*, i}^0)$ (*On améliore le minimum*)

$$\hat{y}_i(t^*, \hat{y}_{t^*, i}^0) \leftarrow y_i(t^*, y_0)$$

$$\hat{y}_{t^*, i}^0 \leftarrow y_0$$

sinon si $y_i(t^*, y_0) > \check{y}_i(t^*, \check{y}_{t^*, i}^0)$ (*On améliore le maximum*)

$$\check{y}_i(t^*, \check{y}_{t^*, i}^0) \leftarrow y_i(t^*, y_0)$$

$$\check{y}_{t^*, i}^0 \leftarrow y_0$$

TAB. 5.4 – Pseudo-code de l'initialisation précédant l'optimisation par SA

$\forall y_i, \forall t^*, \hat{y}_i(t^*) = \infty, \dot{y}_i(t^*) = \infty$
 $\forall y_i$ On choisit aléatoirement \tilde{y}_i dans l'intervalle et $\bar{y}_i \leftarrow \tilde{y}_i$
 $\forall y_i, \forall t^*, t^* \in \tilde{t}_i$ et $t \notin \bar{t}_i$
 Répéter
 \forall variable y_i du système
 choisir \tilde{y}_0^i aléatoirement dans le voisinage de \tilde{y}_0^i
 choisir \bar{y}_0^i aléatoirement dans le voisinage de \bar{y}_0^i
 $\{\tilde{y}_i\}_{\forall i} \leftarrow$ la solution numérique du système pour $\{\tilde{y}_0^i\}_{\forall i}$
 $\{\bar{y}_i\}_{\forall i} \leftarrow$ la solution numérique du système pour $\{\bar{y}_0^i\}_{\forall i}$
 \forall variable y_i du système
 $y_i \leftarrow \{y_i(t) | t \in \tilde{t}_i\} \cup \{\bar{y}_i(t) | t \in \bar{t}_i\}$
 $\tilde{t} \leftarrow \emptyset$ et $\bar{t} \leftarrow \emptyset$
 \forall variable y_i du système
 $\forall t^*$
 si $y_i(t^*) < \dot{y}_i(t^*)$ (*On améliore la trajectoire précédente*)
 $\dot{y}_i(t^*) = y_i(t^*)$ (*On met à jour la trajectoire courante*)
 $\hat{y}_i(t^*) \leftarrow \min(y_i(t^*), \dot{y}_i(t^*))$ (*On améliore le minimum courant*)
 $\tilde{t}_i \leftarrow t^*$ (*L'instant t^* doit être utilisé avec le nouveau voisinage*)
 sinon
 On tire q au hasard entre 0 et 1
 $\Delta y \leftarrow |y_i(t^*) - \dot{y}_i(t^*)|$
 si $q \leq e^{-\frac{|\Delta y|}{T}}$ alors $\tilde{t}_i \leftarrow t^*$ (*L'instant t^* doit être utilisé avec le nouveau voisinage*)
 sinon alors $\bar{t}_i \leftarrow t^*$ (*L'instant t^* doit être utilisé avec le voisinage qui ne change pas*)
 $\tilde{y}_0^i \leftarrow y_0^i$ (*On met à jour le nouveau voisinage*)
 On met T à jour *multiplication par α toutes les L itérations*
 Arrêt si on rencontre les critères de terminaison
 renvoyer $\{\hat{y}^0\}$

TABLEAU 5.5 – Pseudo-code pour la recherche du minimum

$\forall y_i, \forall t^*, \check{y}_i(t^*) = -\infty, \hat{y}_i(t^*) = -\infty$

$\forall y_i$ On choisit aléatoirement \tilde{y}_i dans l'intervalle et $\bar{y}_i \leftarrow \tilde{y}_i$

$\forall y_i, \forall t^*, t^* \in \tilde{t}_i$ et $t \notin \bar{t}_i$

Répéter

\forall variable y_i du système

choisir \tilde{y}_0^i aléatoirement dans le voisinage de \tilde{y}_0^i

choisir \bar{y}_0^i aléatoirement dans le voisinage de \bar{y}_0^i

$\{\tilde{y}\}_{\forall i} \leftarrow$ la solution numérique du système pour $\{\tilde{y}_0^i\}_{\forall i}$

$\{\bar{y}\}_{\forall i} \leftarrow$ la solution numérique du système pour $\{\bar{y}_0^i\}_{\forall i}$

\forall variable y_i du système

$y_i \leftarrow \{y_i(t)|t \in \tilde{t}\} \cup \{\bar{y}_i(t)|t \in \bar{t}\}$

$\tilde{t} \leftarrow \emptyset$ et $\bar{t} \leftarrow \emptyset$

\forall variable y_i du système

$\forall t^*$

si $y_i(t^*) > \hat{y}_i(t^*)$ (*On améliore la trajectoire précédente*)

$\hat{y}_i(t^*) = y_i(t^*)$ (*On met à jour la trajectoire courante*)

$\check{y}_i(t^*) \leftarrow \max(y_i(t^*), \check{y}_i(t^*))$ (*On améliore le maximum courant*)

$\tilde{t}_i \leftarrow t^*$ (*L'instant t^* doit être utilisé avec le nouveau voisinage*)

sinon

On tire q au hasard entre 0 et 1

$\Delta y \leftarrow |y_i(t^*) - \hat{y}_i(t^*)|$

si $q \leq e^{-\frac{|\Delta y|}{T}}$ alors $\tilde{t}_i \leftarrow t^*$ (*L'instant t^* doit être utilisé avec le nouveau voisinage*)

sinon alors $\bar{t}_i \leftarrow t^*$ (*L'instant t^* doit être utilisé avec le voisinage qui ne change pas*)

$\tilde{y}_0^i \leftarrow y_0^i$ (*On met à jour le nouveau voisinage*)

On met T à jour *multiplication par α toutes les L itérations*

Arrêt si on rencontre les critères de terminaison

renvoyer $\{\check{y}^0\}$

TAB. 5.6 – Pseudo-code pour la recherche du maximum

$\hat{y}_i(t^*)$	Le minimum courant de la variable d'état y_i , à l'instant t^*
$\check{y}_i(t^*)$	Le maximum courant de la variable d'état y_i , à l'instant t^*
$\hat{y}_{t^*,i}^0$	La condition initiale minimisant la variable d'état y_i , à l'instant t^*
$\check{y}_{t^*,i}^0$	La condition initiale maximisant la variable d'état y_i , à l'instant t^*
\tilde{y}_0^i	Le centre du voisinage courant, pour la variable d'état y_i
\bar{y}_0^i	Le centre du voisinage précédent, pour la variable d'état y_i
$\tilde{y}_0^{i'}$	La condition initiale tirée au sort dans le voisinage courant, pour la variable d'état y_i
$\bar{y}_0^{i'}$	La condition initiale tirée au sort dans le voisinage précédent, pour la variable d'état y_i
\tilde{t}_i	L'ensemble des instants t^* pour lesquels on doit prendre le nouveau voisinage, pour la variable d'état y_i
\bar{t}_i	L'ensemble des instants t^* pour lesquels on doit prendre le voisinage précédent, pour la variable d'état y_i
$\dot{y}_i(t^*)$	La valeur de la variable d'état y_i sur la trajectoire précédente, au point correspondant à l'instant t^*

TAB. 5.7 – Légende pour les algorithmes du chapitre 5

Chapitre 6

Simulation qualitative des PDE, avec l'approche Qua.Si.

Ce chapitre a pour but d'étendre l'approche Qua.Si. aux PDE où les paramètres et/ou les conditions aux bords sont exprimés par des nombres flous.

La résolution déterministe des PDE déterministes est plus complexe que la résolution des ODE pour les raisons suivantes :

- La présence de plusieurs variables indépendantes va provoquer une diversité de types d'équations auxquels correspondront des techniques particulières de résolution numérique.
- Les problèmes aux conditions initiales des ODE sont remplacés par des problèmes avec des conditions au bord, comme nous allons travailler sur au moins deux dimensions.

En ajoutant de l'incertitude, sous forme de nombres flous, aux PDE, nous allons nous retrouver avec des problèmes dont chacune des conditions aux bords est représentée par un nombre flou. Pour pouvoir résoudre ces PDE floues, il nous faudra exprimer ces valeurs floues sous forme d' α -coupure afin de pouvoir calculer les α -coupures des solutions de la PDE.

Appliquer l'approche Qua.Si. aux PDE va consister à, en plus de gérer les variables d'état composant la PDE, utiliser la routine d'optimisation non plus pour chaque instant mais pour chaque variable indépendante de l'équation à résoudre. Comme nous allons utiliser une méthode de différence-finie pour effectuer la résolution numérique, en pratique nous allons devoir effectuer la minimisation et la maximisation, pour chacune des variables d'état constituant le modèle du problème, pour chacun des points constituant le quadrillage par lequel le domaine des variables indépendantes a été discrétisé.

En raison de la diversité de formes que peuvent prendre les ODE, nous allons nous focaliser dans ce chapitre sur les PDE elliptiques, et en particulier sur les équations comportant l'opérateur ∇ , à savoir les équations de Laplace, Poisson et Helmholtz.

6.1 Résolution des PDE déterministes

Pour rappel (section 1.2.1) :

$$\nabla^2 u = 0 \quad \text{pour l'équation de Laplace} \quad (6.1)$$

$$\nabla^2 u = g(x,y) \quad \text{pour l'équation de Poisson} \quad (6.2)$$

$$\nabla^2 u + f(x,y) = g(x,y) \quad \text{pour l'équation de Helmholtz} \quad (6.3)$$

avec

$$\nabla^2 u = u_{xx} + u_{yy} \quad (6.4)$$

Nous pourrions employer la méthode de différence-finie dans le cas où toutes les conditions initiales aux bords sont connues. Dans le cas où nous avons deux variables indépendantes, les bords peuvent être représentés par un rectangle délimité en abscisses par l'intervalle $[x_{min}, x_{max}]$ et en ordonnées par l'intervalle $[y_{min}, y_{max}]$. Connaître les conditions aux bords consistera alors à connaître la fonction $u(x,y)$ (et les fonctions $f(x,y)$ et/ou $g(x,y)$ pour Poisson et/ou Helmholtz) sur les côtés du rectangle délimité par les points :

$$(x_{min}, y_{min}) \quad , \quad (x_{min}, y_{max}) \quad \text{à gauche}$$

$$(x_{min}, y_{min}) \quad , \quad (x_{max}, y_{min}) \quad \text{en bas}$$

$$(x_{max}, y_{min}) \quad , \quad (x_{max}, y_{max}) \quad \text{à droite}$$

$$(x_{min}, y_{max}) \quad , \quad (x_{max}, y_{max}) \quad \text{en haut}$$

6.1.1 Résolution d'équations similaires à l'équation de Laplace

Nous allons utiliser des différences-quotients d'ordre 2 pour approximer les dérivées secondes partielles en x et y .

De la section 1.2.2, nous avons

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \quad (6.5)$$

et donc

$$u_{xx} \approx \frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{h^2} \quad (6.6)$$

$$u_{yy} \approx \frac{u(x,y+h) - 2u(x,y) + u(x,y-h)}{h^2} \quad (6.7)$$

$$\nabla^2 \approx \frac{u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) - 4u(x,y)}{h^2} \quad (6.8)$$

Pour résoudre nos équations, il nous faudra alors

$$\frac{u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) - 4u(x,y)}{h^2} = 0 \quad (6.9)$$

pour l'équation de Laplace

$$\frac{u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) - 4u(x,y)}{h^2} = g(x,y) \quad (6.10)$$

pour l'équation de Poisson

$$\frac{u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) - 4u(x,y)}{h^2} = g(x,y) - f(x,y)u \quad (6.11)$$

pour l'équation de Helmholtz

Traçons maintenant la grille dans le rectangle auquel appartiennent x et y , dont les points sont répartis uniformément :

$$- x_{i+1} = x_i + h \text{ pour } i = 1, \dots, n-1 \text{ et } x_{i-1} = x_i - h \text{ pour } i = 2, \dots, n$$

$$- y_{j+1} = y_j + h \text{ pour } j = 1, \dots, m-1 \text{ et } y_{j-1} = y_j - h \text{ pour } j = 2, \dots, m$$

tels qu'un des deux côtés du rectangle soit de longueur mh et le second de taille nh . Utilisons la notation $u_{i,j}$ pour $u(x_i, y_j)$. Nous obtenons alors

$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = 0 \quad (6.12)$$

pour l'équation de Laplace

$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = g_{i,j} \quad (6.13)$$

pour l'équation de Poisson

$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = g_{i,j} - f_{i,j}u_{i,j} \quad (6.14)$$

pour l'équation de Helmholtz

qui nous donnent des **formules de différence à 5 points**. Nous pouvons en déduire la relation qui existent entre la valeur $u_{i,j}$, avec ses quatre voisins et éventuellement les fonctions f et g .

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = 0 \quad (6.15)$$

pour l'équation de Laplace

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = (g_{i,j})h^2 \quad (6.16)$$

pour l'équation de Poisson

$$u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = (g_{i,j} - f_{i,j}u_{i,j})h^2 \quad (6.17)$$

pour l'équation de Helmholtz

Supposons que nous connaissons les valeurs $u(x,y)$ sont connues aux bords de la grille du rectangle.

$$u(x_1, y_j) = u_{1,j} \text{ pour } 2 \leq j \leq m-1 \text{ (à gauche)} \quad (6.18)$$

$$u(x_i, y_1) = u_{i,1} \text{ pour } 2 \leq i \leq n-1 \text{ (en bas)} \quad (6.19)$$

$$u(x_n, y_j) = u_{n,j} \text{ pour } 2 \leq j \leq m-1 \text{ (à droite)} \quad (6.20)$$

$$u(x_i, y_m) = u_{i,m} \text{ pour } 2 \leq i \leq n-1 \text{ (en haut)} \quad (6.21)$$

Si on applique les formules à chacun des points intérieurs de la grille, nous obtenons un système linéaire de $(n - 2) \times (m - 2)$ équations à $(n - 2) \times (m - 2)$ inconnues qu'il nous suffira de résoudre pour trouver les valeurs des points intérieurs. Il sera possible de résoudre le système en utilisant les méthodes itératives de Gauss-Seidel et de Jacobi (cf. sections A.2.1 et A.2.2).

Pour éviter le stockage du système d'équations, il existe une méthode itérative inspirée de la résolution d'équations par point fixe. On pourrait ré-écrire les formules de différence de façon itérative. Nous obtiendrons alors les équations sous la forme itérative

$$u_{i,j} = u_{i,j} + r_{i,j} \quad (6.22)$$

avec

$$r_{i,j} = \frac{u_{i+1,j} + u_{x-1,y} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{4} \quad \text{pour l'équation de Laplace} \quad (6.23)$$

$$r_{i,j} = \frac{u_{i+1,j} + u_{x-1,y} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} - h^2 g_{i,j}}{4} \quad \text{pour l'équation de Poisson} \quad (6.24)$$

$$r_{i,j} = \frac{u_{i+1,j} + u_{x-1,y} + u_{i,j+1} + u_{i,j-1} - (4 - h^2 f_{i,j})u_{i,j} - h^2 g_{i,j}}{4 - h^2 f_{i,j}} \quad \text{pour l'équation de Helmholtz} \quad (6.25)$$

Comme nous avons une formule itérative, il nous faut attribuer une valeur de départ aux valeurs de u pour les points intérieurs du rectangle. Elle sera donnée en prenant la moyenne arithmétique des $2n + 2m - 4$ conditions sur les bords du rectangle [MF99].

6.2 Application aux PDE floues

Dans cette section, nous nous attarderons sur les PDE floues et discuterons de l'utilisation de l'approche Qua.Si. dans le cas spécifique évoqué en 6.1.

6.2.1 Les PDE floues

Considérons une équation différentielle partielle floue

$$F \left(\frac{\partial^m \phi}{\partial x_1^m}, \frac{\partial^m \phi}{\partial x_2^m}, \dots, \frac{\partial^{m-1} \phi}{\partial x_1^{m-1} \partial x_2}, \dots, \phi, x_1, \dots, x_m \right) = 0 \quad (6.26)$$

$$\{x_1, \dots, x_m\} \in \mathfrak{D} \in \mathbb{R}^m \quad \phi \in \mathbb{R}^n$$

$$\phi_{bord}(x_1, \dots, x_m) \sim \mu_\phi(f_{bord}(x_1, \dots, x_m)) \equiv \mu_{bord}(x_1, \dots, x_m) \quad \forall \text{ bord de } \mathfrak{D}$$

où

- La région délimitée par les domaines des variables dépendantes x_1, \dots, x_2 est dénotée par \mathfrak{D} .
- $\{\mu_{bord}(x_1, \dots, x_m)\}_{\forall bord}$ est l'ensemble des conditions floues sur les bords de \mathfrak{D} .
- Les composantes de $\{\mu_{bord}\}_{\forall bord}$ sont chacune définies sur le domaine à n dimensions de ϕ

L'équation (6.26) sera interprétée comme l'extension de la notion de PDE conformément au principe (2.13). Et donc, une **équation différentielle partielle floue** sera définie comme une PDE, où certaines conditions aux bords ou paramètres sont incertains et sont représentés par

des nombres flous. La solution de la PDE floue sera alors l'évolution dans \mathfrak{D} des distributions de possibilités sur les bords de \mathfrak{D} .

Comme vu en 2.3.2, nous pourrons décomposer les calculs nécessaires à la résolution des PDE floues en la succession des résolutions de la PDE en prenant les α -coupures des distributions de possibilités sur les bords de \mathfrak{D} . Pour résoudre numériquement les PDE floues en utilisant une méthode de différence-finie, il faudra prendre les α -coupures des distributions de possibilités sur les bords, à chacun des points qui vont être utilisés pour discrétiser \mathfrak{D} . Nous obtenons ainsi, pour toute α -coupure, un intervalle associé à chacun des points utilisés pour discrétiser les bords de \mathfrak{D} .

Revenons à notre exemple. Si nous prenons des conditions aux bords floues pour l'équation de Laplace, nous obtiendrons

$$\nabla^2 u = 0 \quad u \in \mathbb{R} \quad (6.27)$$

$$u(x,y) \sim \mu_u(f_{gauche}(x,y)) \equiv \mu_{gauche}(x,y) \quad \text{pour} \quad x = x_{min} \text{ et } y_{min} \leq y_{max} \quad (6.28)$$

$$u(x,y) \sim \mu_u(f_{bas}(x,y)) \equiv \mu_{bas}(x,y) \quad \text{pour} \quad x_{min} \leq x_{max} \text{ et } y = y_{min} \quad (6.29)$$

$$u(x,y) \sim \mu_u(f_{droite}(x,y)) \equiv \mu_{droite}(x,y) \quad \text{pour} \quad x = x_{max} \text{ et } y_{min} \leq y_{max} \quad (6.30)$$

$$u(x,y) \sim \mu_u(f_{haut}(x,y)) \equiv \mu_{haut}(x,y) \quad \text{pour} \quad x_{min} \leq x_{max} \text{ et } y = y_{max} \quad (6.31)$$

où \mathfrak{D} est défini par le rectangle délimité par les points (x_{min}, y_{min}) , (x_{max}, y_{min}) , (x_{min}, y_{max}) et (x_{max}, y_{max}) .

Comme nous allons résoudre l'équation par une méthode différence-finie, nous allons discrétiser le rectangle \mathfrak{D} en un quadrillage de la façon exposée en 6.1.1. Nous connaissons les conditions floues aux bords de \mathfrak{D} .

$$u(x_1, y_j) \sim \mu_u(x_1, y_j) \equiv \mu_{1,j} \quad \text{pour } 2 \leq j \leq m-1 \quad (\text{à gauche}) \quad (6.32)$$

$$u(x_i, y_1) \sim \mu_u(x_i, y_1) \equiv \mu_{i,1} \quad \text{pour } 2 \leq i \leq n-1 \quad (\text{en bas}) \quad (6.33)$$

$$u(x_n, y_j) \sim \mu_u(x_n, y_j) \equiv \mu_{n,j} \quad \text{pour } 2 \leq j \leq m-1 \quad (\text{à droite}) \quad (6.34)$$

$$u(x_i, y_m) \sim \mu_u(x_i, y_m) \equiv \mu_{i,m} \quad \text{pour } 2 \leq i \leq n-1 \quad (\text{en haut}) \quad (6.35)$$

où $\mu_{i,j}$ est la valeur floue de u aux points x_i et y_j .

Prenons maintenant une α -coupure de niveau l . Résoudre u pour le niveau l consistera à trouver les α -coupures de niveau l des points intérieurs de \mathfrak{D} , en fonction des α -coupures de niveau l des conditions floues aux bords de \mathfrak{D} . Dans la section 6.2.2, nous proposerons une méthode permettant de trouver les α -coupures des points intérieurs de \mathfrak{D} .

6.2.2 L'approche Qua.Si. pour la résolution de PDE floues

Supposons que nous connaissons les conditions aux bords de \mathfrak{D} , pour une PDE du second ordre. Nommons u la variable dépendante à résoudre et x et y les variables indépendantes. Soient m et n les deux dimensions du quadrillage utilisé pour la résolution par différence-finie.

En prenant les α -coupures de ces conditions aux bords ou bien dans le cas de nombres flous rectangulaires, nous obtiendrons des intervalles correspondant aux valeurs que peut prendre u sur les bords de \mathcal{D} . Notre but sera de trouver les intervalles correspondant aux valeurs que peut prendre u dans les points intérieurs de \mathcal{D} .

Pour trouver les extrémités des intervalles associés aux points de u , il nous faudra faire la recherche du minimum et du maximum de la fonction u pour chacun des points du quadrillage du rectangle \mathcal{D} . Il ne nous faudra donc plus résoudre l'équation pour tout t^* comme pour les ODE, mais bien pour tout x_i^* avec $i = 1, \dots, n$ et pour tout y_j^* avec $j = 1, \dots, m$. L'algorithme d'optimisation utilisé recherchera alors les extrema de u . En pratique, il cherchera à minimiser et maximiser pour tout x_i^* et pour tout y_j^* la valeur en (x_i^*, y_j^*) de la fonction résolue en utilisant les méthodes exposées en 6.1.1.

La table 6.1 nous donne le canevas de la résolution d'une α -coupure d'une PDE floue du second ordre, ou la résolution d'une PDE floue rectangulaire du second ordre, avec une seule variable dépendante u .

$\forall x_i^*, i = 1, \dots, n$ $\forall y_j^*, j = 1, \dots, m$ $u_{i^*, j^*}^{min} = \infty$ $u_{i^*, j^*}^{max} = -\infty$ $u_{i^*, j^*}^{min} = \min (\{u_{i^*, j^*}(u_{1,j}, u_{i,1}, u_{n,j}, u_{i,m})\}) \mid u_{i^*, j^*}(u_{1,j}, u_{i,1}, u_{n,j}, u_{i,m}) \text{ est une solution de la PDE}$ $u_{i^*, j^*}^{max} = \max (\{u_{i^*, j^*}(u_{1,j}, u_{i,1}, u_{n,j}, u_{i,m})\}) \mid u_{i^*, j^*}(u_{1,j}, u_{i,1}, u_{n,j}, u_{i,m}) \text{ est une solution de la PDE}$
--

TAB. 6.1 – Qua.Si. pour les PDE

6.2.3 Remarques et considérations

L'extension proposée ici est assez restrictive. En effet, nous ne voyons ici que l'application de la méthode Qua.Si pour un problème simple composé d'une seule équation, avec une seule variable.

La résolution par différences-finies employées ici est limitée à un seul type de PDE. En effet, résoudre des PDE plus complexes ou bien des systèmes de PDE aurait nécessité le développement d'un algorithme plus général de résolution. Le développement d'une telle méthode nécessite des connaissances avancées en calcul numérique. En effet, l'utilisation de différences-finies nécessite la résolution d'un système linéaire d'équations. Or, certaines méthodes de résolution de systèmes linéaires risquent de ne pas bien fonctionner ou bien d'être inadaptées à cause de la forme de la matrice définissant le système.

La complexité engendrée par l'utilisation des PDE et leur résolution par différences-finies

nous a empêché de prendre en considération les gradients. L'utilisation de fonctions à plus d'une variable pose problème au niveau des dérivées partielles du gradient et l'emploi des différences-finies complique la compréhension de la forme du gradient. L'utilisation d'une méthode d'optimisation basée sur les gradients aurait alors permis une plus grande rapidité et une plus grande précision.

Toujours à propos du gradient, la notion de matrice de connection utilisée en 4.2 devra être modifiée pour gérer l'ajout de dimensions qu'entraîne la résolution de PDE

Une autre difficulté entraînée par l'utilisation de fonctions à plusieurs variables se situe au niveau de la surface extérieure d'incertitude. Le problème réside dans le cas où le nombre de variables d'état est supérieur à 1. Il faut en effet étudier la nécessité de minimiser et maximiser la fonction sur toutes les composantes de la surface extérieure d'incertitude. Un autre problème réside dans l'implémentation s'il y a plus d'une variable d'état. Si toutes les composantes doivent être considérées, il faut en prendre en considération pour toute variable d'état, chaque variable indépendante de la PDE.

Chapitre 7

Simulations réalisées

Plusieurs systèmes dynamiques où les conditions initiales et/ou paramètres sont incertains et décrits de manière non-probabiliste ont été simulés avec les outils décrits jusqu'ici. Cela a permis la comparaison des performances ainsi que la validation éventuelle des améliorations apportées à Qua.Si. dans le cadre des ODE. En particulier, nous avons comparé les résultats obtenus par l'utilisation des méthodes de simulation par

- arithmétique affine
- MC
- Qua.Si.
- Qua.Si+MC
- Qua.Si+SA

. La méthode de simulation par arithmétique affine utilise la méthode d'Euler pour résoudre numériquement l'équation décrivant le système. Une méthode de Runge-Kutta a été employée pour la résolution numérique utilisées dans les autres méthodes de simulation d'ODE (MC, Qua.Si, Qua.Si+MC et Qua.Si+SA) étudiées dans ce chapitre.

Pour la résolution des systèmes modélisés par des équations différentielles partielles, une comparaison a été effectuée entre l'utilisation des méthodes de MC et de celle décrite dans 6.1 pour l'équation de Laplace

7.1 Modèle de la température de transformateurs de courant

Le premier modèle qui va être étudié est celui de la température des transformateurs à bain d'huile. Plus d'informations ainsi que des références à sujet sont dans [Pro99].

7.1.1 Description du problème

La protection thermique des transformateurs à bain d'huile minérale des sous-stations d'alimentation en courant est d'importance critique pour les systèmes électriques. En effet, en

cas de panne, les conséquences seraient importantes tant au niveau économique que dans l'image de l'entreprise victime de la panne, vis à vis du consommateur. Pour cela, il est nécessaire d'estimer la capacité de chargement des transformateurs. Cela nécessite de pouvoir évaluer de façon correcte les températures des régions chaudes du centre ou du sommet des bobines ainsi que de celle de la surface de l'huile, afin de vérifier si elles sont plus basses qu'un seuil maximal acceptable de température. Intéressons nous à celle des zones chaudes. Elle peut être évaluée selon le modèle en 7.1.2 à partir des hausses ultimes de température de la surface d'huile et des régions chaudes émergées. Celles-ci sont respectivement calculables grâce à la hausse évaluée de température au-delà de la température ambiante ainsi que le taux de perte de charge aux branchements et la hausse évaluée de température des régions chaudes émergées. De plus, ces valeurs sont valables pour un certain profil de charge du transformateur, pondérées par des constantes de temps et sont également sous l'influence de la méthode de refroidissement utilisée et des changements de résistance lors des variations de charge [Pro99].

Le problème est que certaines de ces valeurs sont incertaines. Cette incertitude ayant pour source la surcharge, le vieillissement et enfin la tolérance aux contraintes du système [GIVV02]. De plus, cette incertitude est connue de façon non-probabiliste.

7.1.2 Le modèle mathématique

Selon [Pro99], la température des zones chaudes au centre ou au sommet des bobines peut être modélisée par le système suivant :

$$\begin{cases} \tau_{TO} \frac{d\Theta_{TO}}{dt} &= [\Delta_{TO,U} + \Theta_A] - \Theta_{TO} \\ \tau_H \frac{d\Delta\Theta_H}{dt} &= \Delta\Theta_{H,U} - \Delta\Theta_H \\ \Delta\Theta_{TO,U} &= \Delta\Theta_{TO,R} \left[\frac{I_L^2 R + 1}{R + 1} \right]^q \\ \Delta\Theta_{H,U} &= \Delta\Theta_{H,R} I_L^{2m} \end{cases}$$

où :

- $\Delta\Theta_{TO,U}$ = la hausse ultime de température à la surface de l'huile,
- $\Delta\Theta_{TO,R}$ = la hausse évaluée de température à la surface de l'huile par rapport à la température ambiante,
- $\Delta\Theta_{H,U}$ = la hausse ultime de température des zones chaudes émergées,
- $\Delta\Theta_{H,R}$ = la hausse évaluée de température des zones chaudes émergées,
- τ_{TO} = une constante de temps pour pour l'augmentation en surface de l'huile,
- τ_H = une constante de temps pour pour l'augmentation aux zones chaudes,
- I_L = la charge courante normalisée à celle évaluée,
- R = la proportion de charge évaluée perdue aux branchements,
- m = un exposant dérivé empiriquement, dépendant de la méthode de refroidissement,
- q = un exposant dérivé empiriquement, dépendant des changements de résistance lors des variations de charge.

La quantité que nous recherchons (la température des zones chaudes) est donnée par la somme $\Theta_{TO} + \Delta\Theta_H$.

7.1.3 L'expérience

La simulation a été effectuée pour les deux profils de I_L , utilisés dans [GIVV02]. le premier correspond à la charge du transformateur en semaine et le second à celle des week-end et jours fériés. Certains paramètres sont constants et sont repris dans la table 7.1 ; les conditions initiales pour les simulations sont reprises dans la table 7.2. Les ε_1 , ε_2 et ε_3 , des valeurs affines (cf. section 2.2) correspondent respectivement aux sources d'incertitudes suivantes : la surcharge possible, le vieillissement et la résistance aux contraintes du système.

Paramètre	valeur	unité
Θ_A	27	°C
m	1,6	nombre réel
q	0,8	nombre réel

TAB. 7.1 – Paramètres constants

Paramètre	valeur affine	cond. rectangulaire floue	unité
Θ_{TO}	30	[30 30]	°C
$\Delta\Theta_H$	0	[0 0]	°C
τ_H	$8 + 3\varepsilon_1 + 0,16\varepsilon_2 + 0,24\varepsilon_3$	[4,6 11,4]	min
$\Delta\Theta_{H,R}$	$4,75 + 0,75\varepsilon_1 + 0,14\varepsilon_2 + 0,095\varepsilon_3$	[3,763 5,737]	°C
τ_{TO}	$2,5 + 1\varepsilon_1 + 0,07\varepsilon_2 + 0,05\varepsilon_3$	[1,375 3,625]	h
$\Delta\Theta_{TO,R}$	$37,5 + 2,5\varepsilon_1 + 0,09\varepsilon_2 + 0,135\varepsilon_3$	[33,87 41,12]	°C
R	$4,5 + 0,5\varepsilon_1 + 0,09\varepsilon_2 + 0,135\varepsilon_3$	[3,78 5,22]	nombre réel

TAB. 7.2 – Conditions initiales pour la simulation

Discussions des résultats

Les figures présentées dans cette section montrent les résultats de la simulation du modèle de température, sur un intervalle de temps allant de 0 à 24 heures, pour les deux profils de charge du transformateur.

Les figures 7.1 et 7.2 montrent les résultats de la simulation avec les méthodes de mathématique affine, Monte-Carlo avec 20 000 échantillonnages et Qua.Si. La dernière courbe (valeurs nominales) montre la résolution du système sans incertitude.

Les figures 7.3 et 7.4 montrent les résultats trouvés, pour les deux profils de charge, de la simulation par Monte-Carlo avec 20 000 et 200 échantillonnages

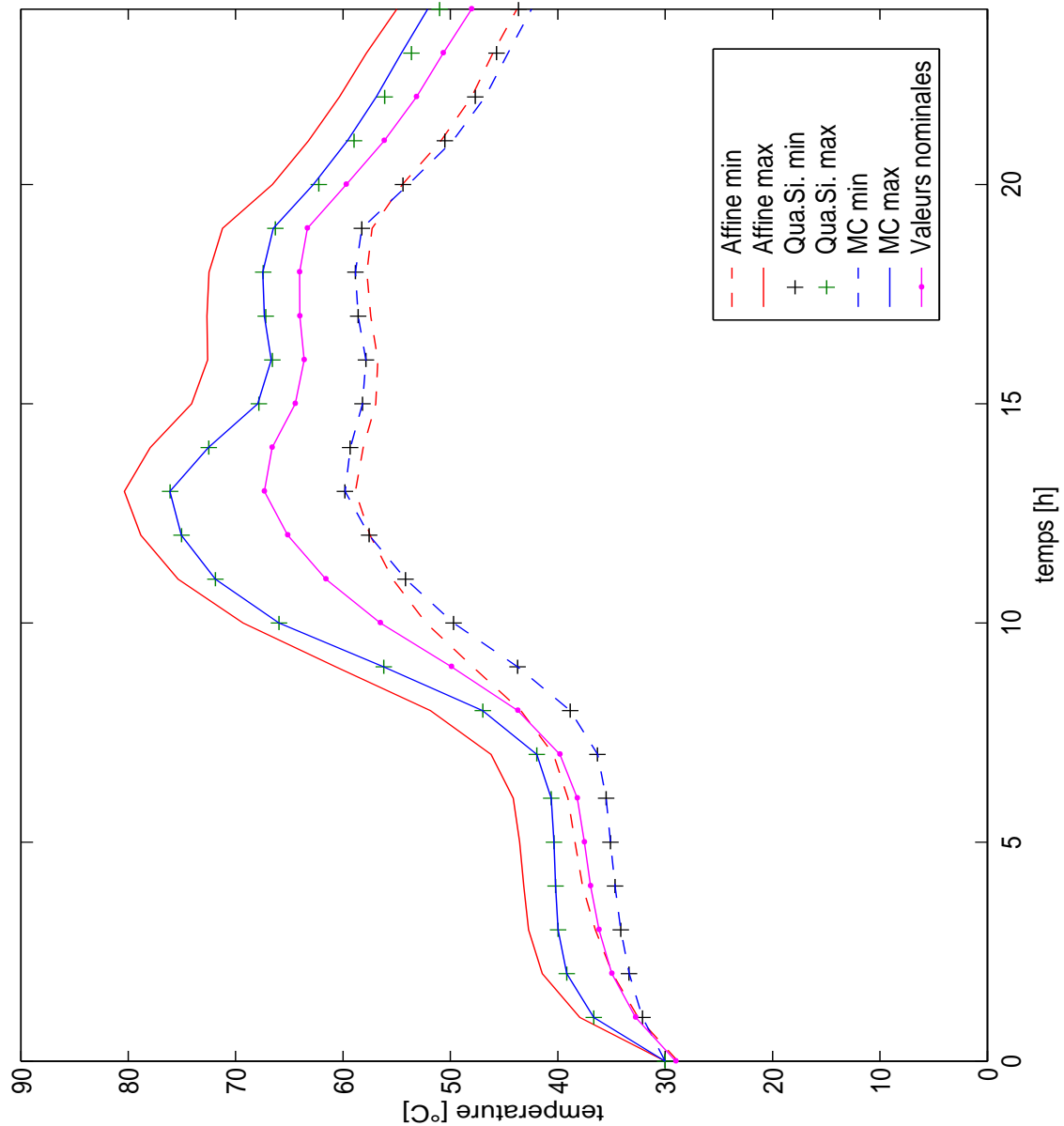


FIG. 7.1 – *Modèle du transformateur – comparaison des intervalles trouvés pour le profil des jours de semaine*

Les figures 7.5 et 7.6 montrent la simulation par Qua.Si. et Qua.Si+MC pour le profil des jours de semaine, les figures 7.7 et 7.8 font de même pour le profil des week-end et jours fériés. 100 et 200 échantillonnages ont été effectués pour la partie non-basée sur les gradients

Les figures 7.9 et 7.10 montrent la comparaison entre MC. et Qua.Si+MC pour le profil des jours de semaine, les figures 7.11 et 7.12 font de même pour le profil des week-end et jours fériés.

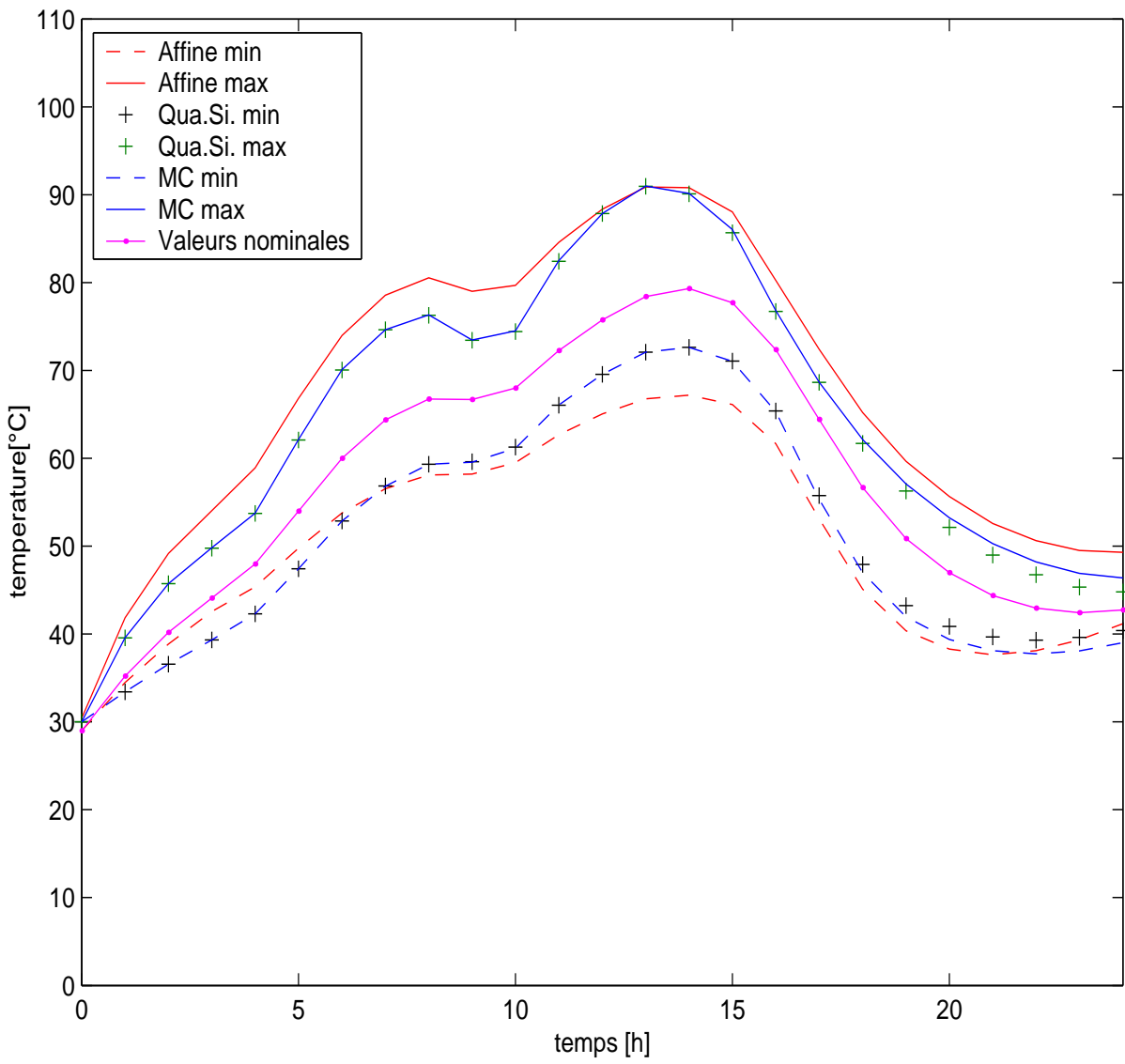


FIG. 7.2 – Modèle du transformateur – comparaison des intervalles trouvés pour le profil des week-end et jours fériés

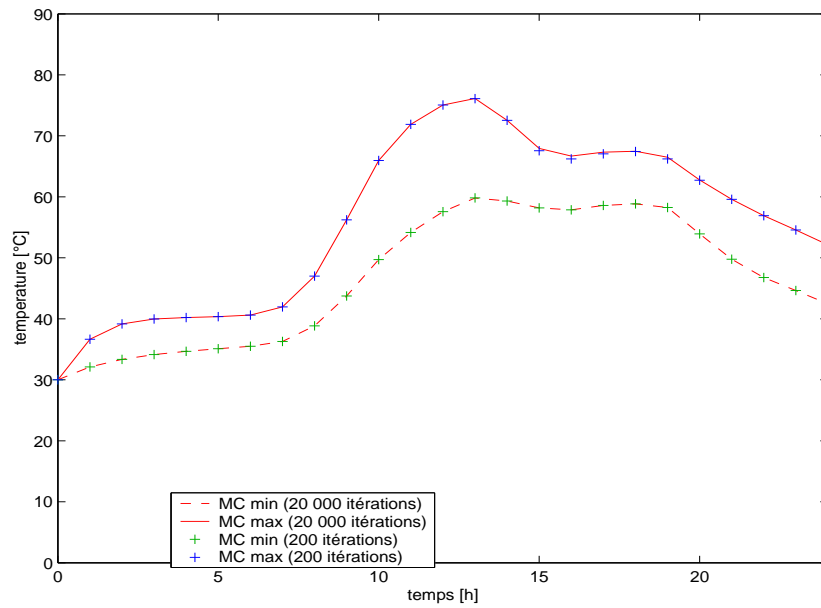


FIG. 7.3 – *Modèle du transformateur – comparaison des intervalles trouvés avec MC pour le profil des jours de semaine*

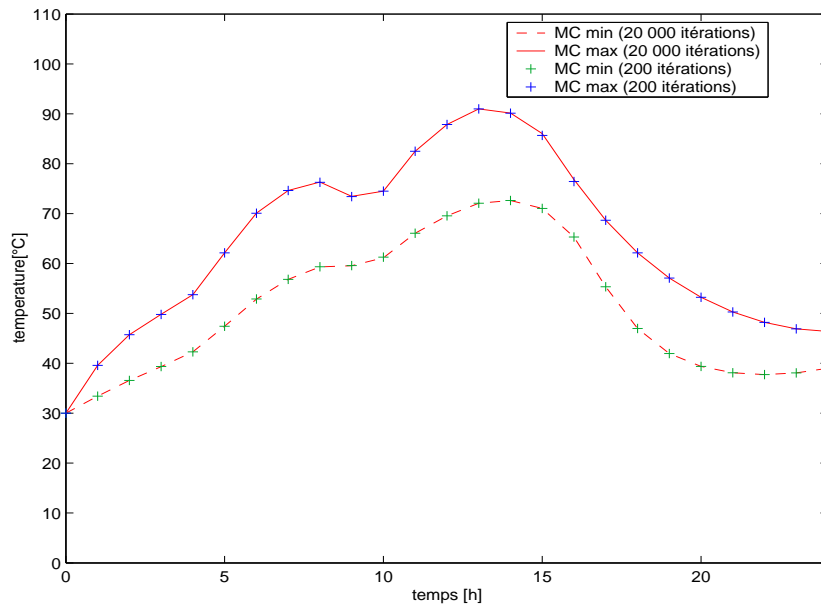


FIG. 7.4 – *Modèle du transformateur – comparaison des intervalles trouvés avec MC pour le profil des week-end et jours fériés*

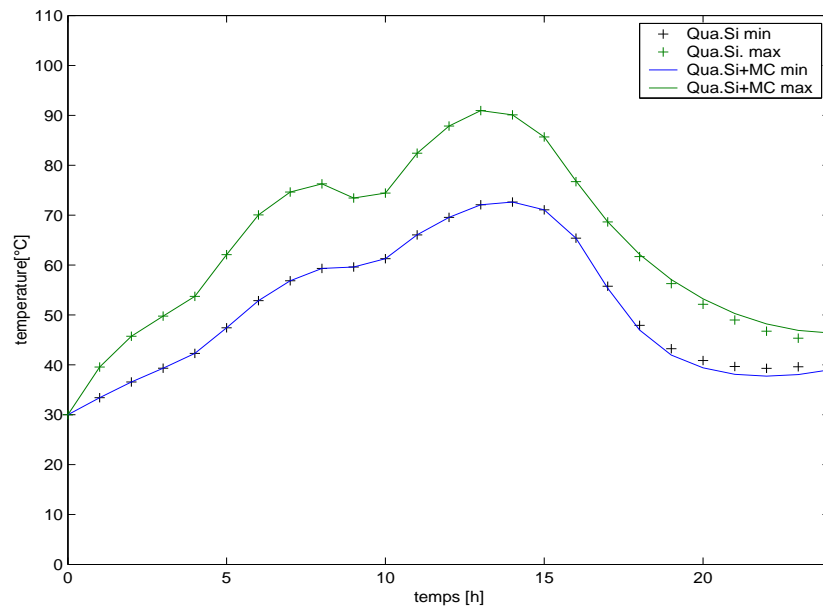


FIG. 7.5 – Modèle du transformateur – comparaison Qua.Si. et Qua.Si.+MC(100 it.) pour le profil des jours de semaine

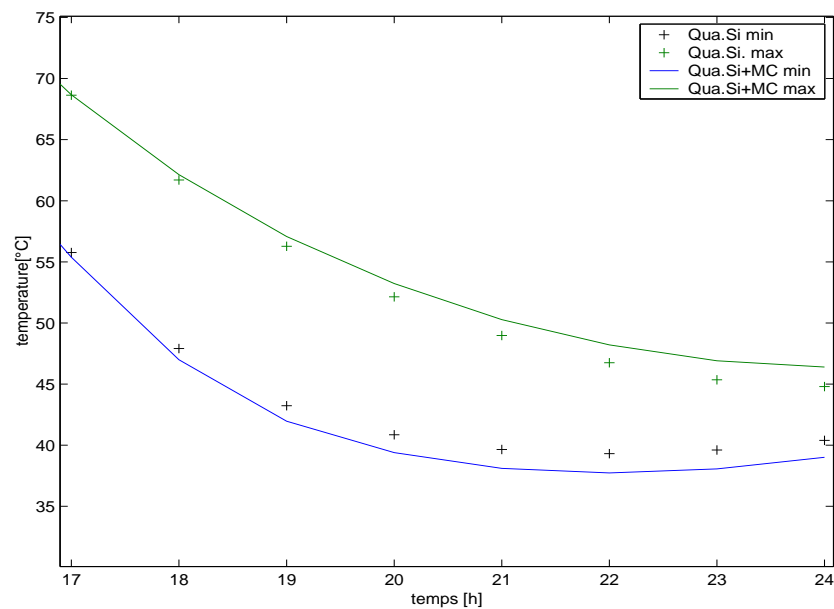


FIG. 7.6 – Modèle du transformateur – comparaison Qua.Si. et Qua.Si.+MC(100 it.) pour le profil des jours de semaine (zoom)

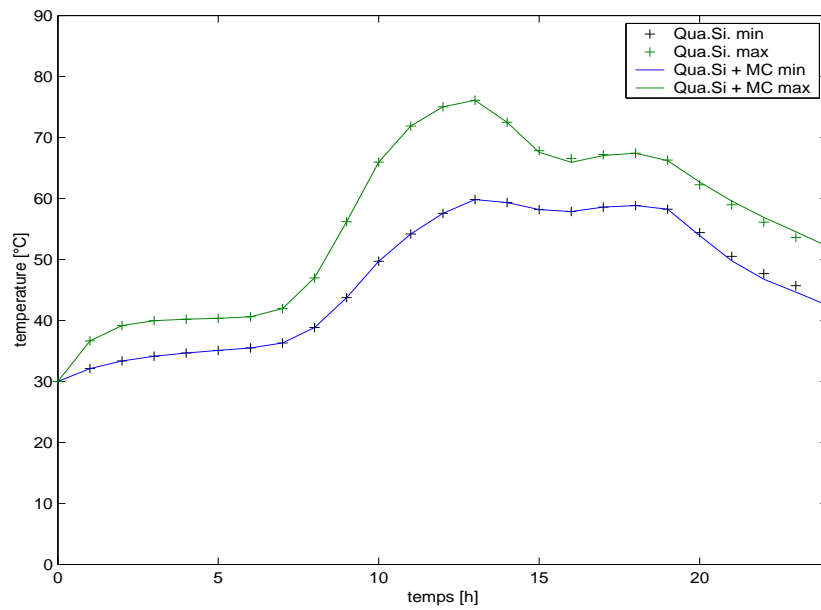


FIG. 7.7 – Modèle du transformateur – comparaison Qua.Si. et Qua.Si.+MC(100 it.) pour le profil des week-end et jours fériés

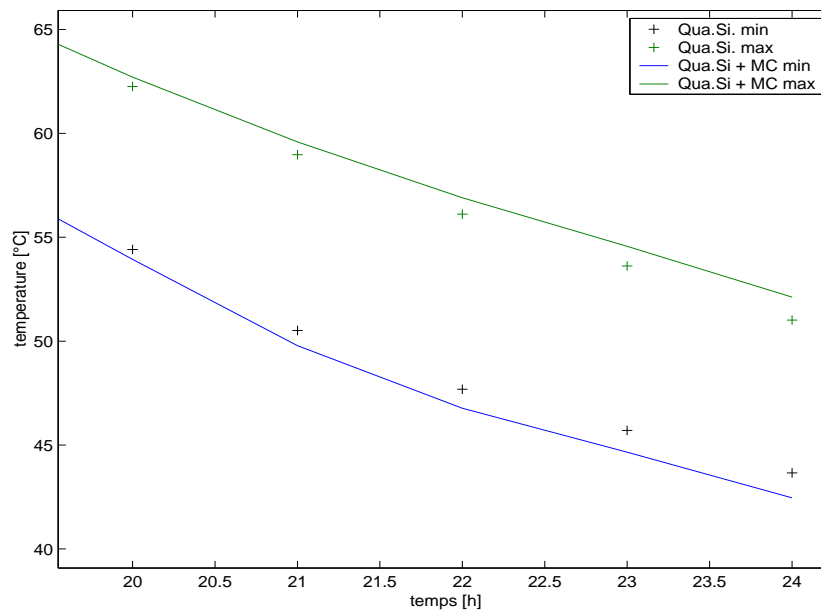


FIG. 7.8 – Modèle du transformateur – comparaison Qua.Si. et Qua.Si.+MC pour le profil des week-end et jours fériés (zoom)

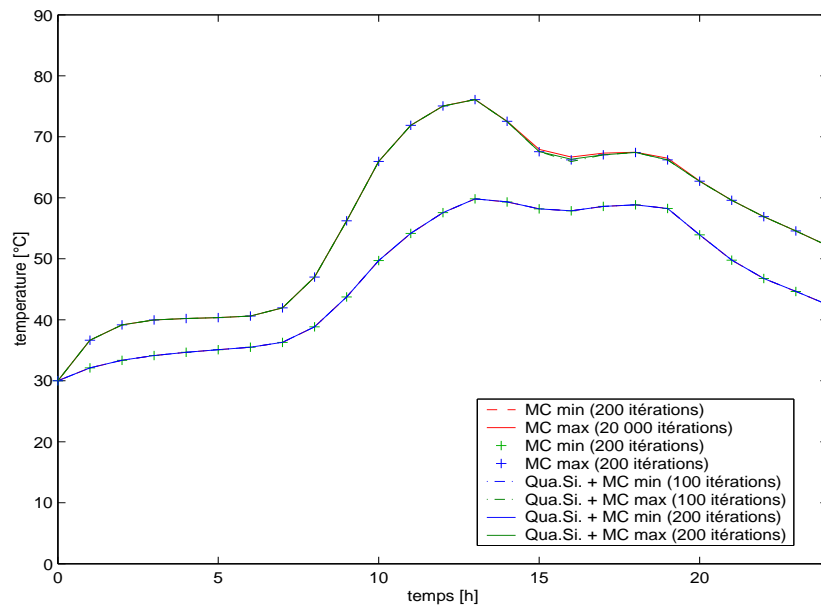


FIG. 7.9 – Modèle du transformateur – comparaison MC et Qua.Si+MC pour le profil des jours de semaine

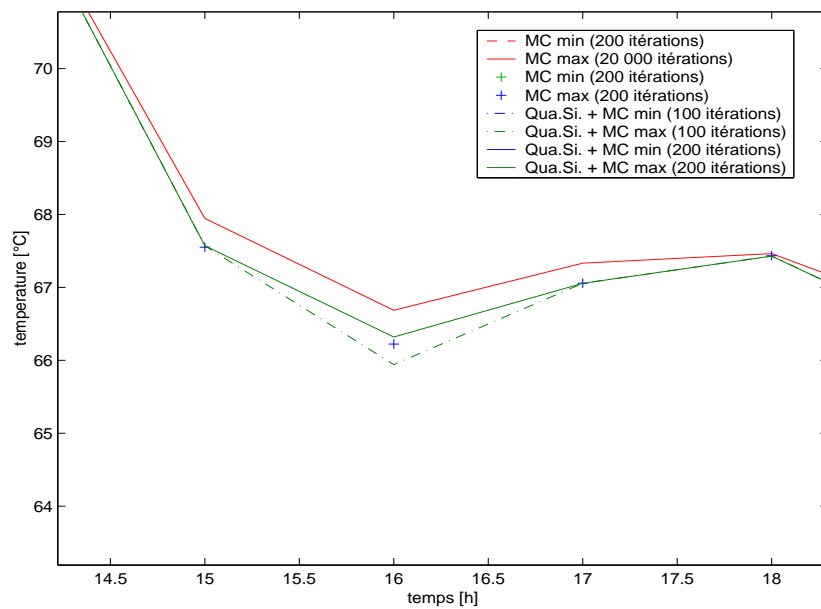


FIG. 7.10 – Modèle du transformateur – comparaison MC et Qua.Si+MC pour le profil des jours de semaine (zoom)

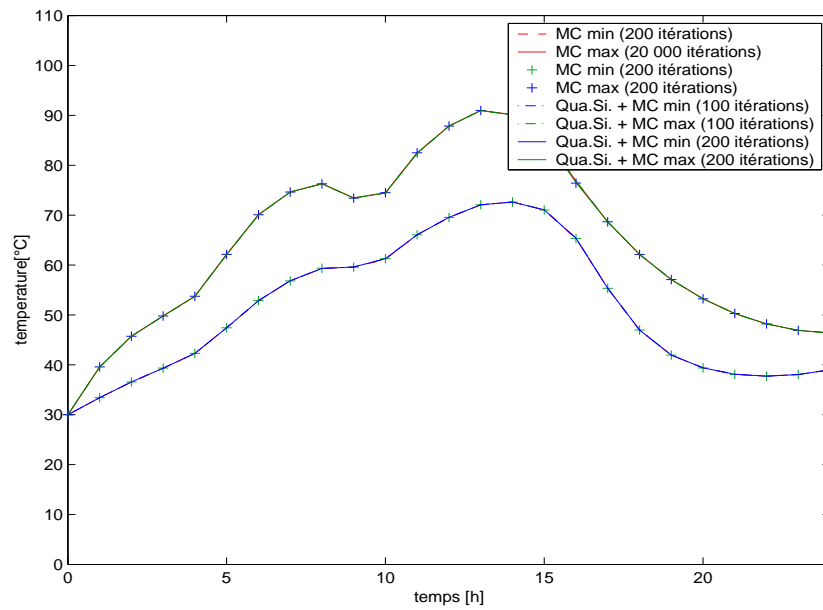


FIG. 7.11 – *Modèle du transformateur – comparaison MC et Qua.Si.+MC pour le profil des week-end et jours fériés*

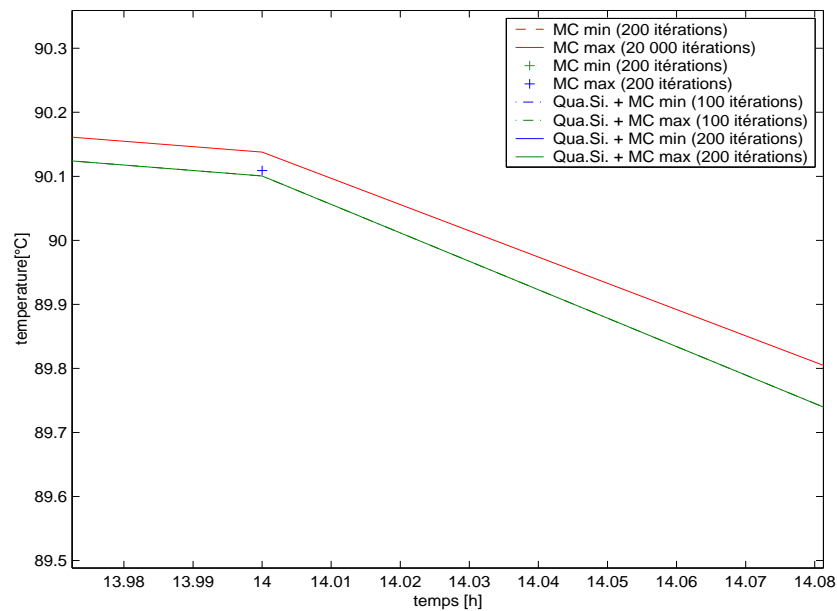


FIG. 7.12 – *Modèle du transformateur – comparaison MC et Qua.Si.+MC pour le profil des week-end et jours fériés (zoom)*

Les figures 7.13 et 7.14 montrent la comparaison des résultats de la simulation par Qua.Si+MC(100 it.) et Qua.Si+SA pour les deux profils de charge.

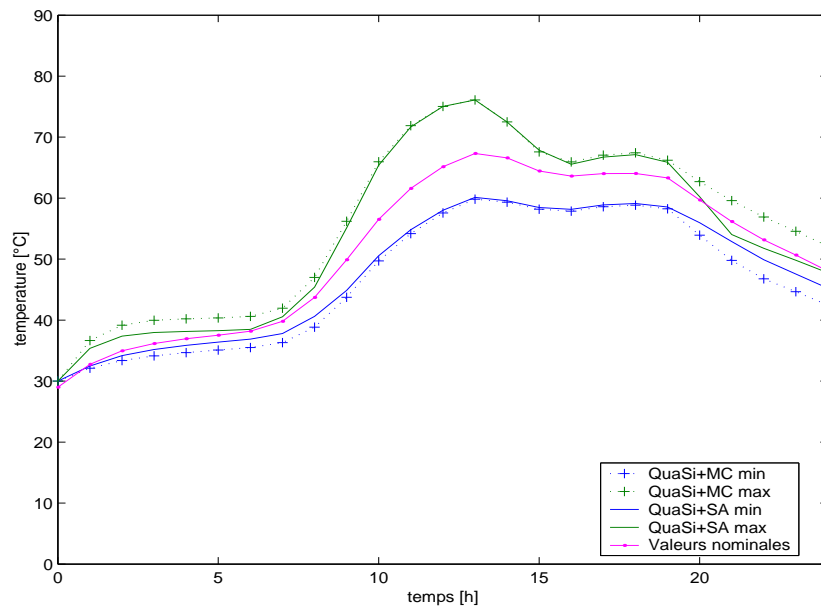


FIG. 7.13 – *Modèle du transformateur – comparaison Qua.Si+MC(100 it.) et Qua.Si+SA pour le profil des week-end et jours fériés*

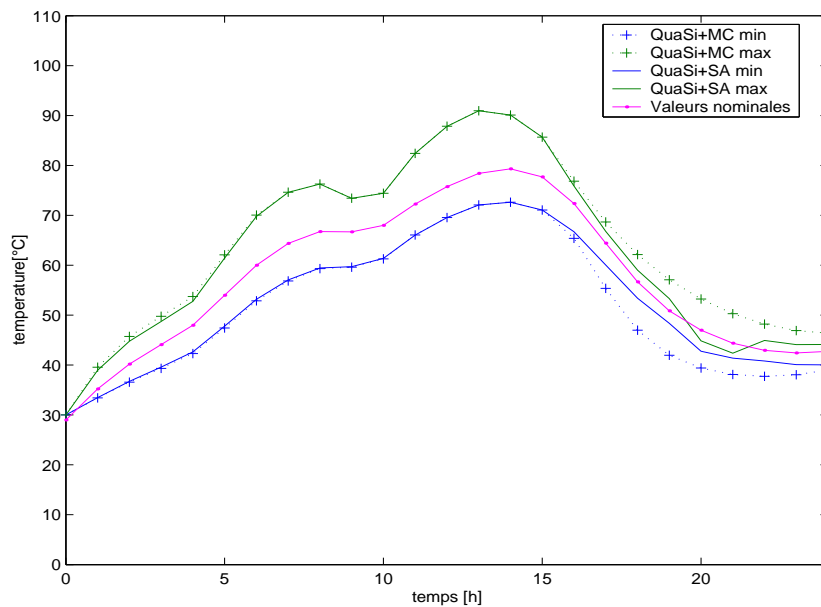


FIG. 7.14 – *Modèle du transformateur – comparaison Qua.Si+MC(100 it.) et Qua.Si+SA pour le profil des week-end et jours fériés*

Discutons maintenant des résultats obtenus. Nous savons déjà que les simulations par Qua.Si., Qua.Si+MC Qua.Si+SA et MC ne généreront jamais de trajectoires en dehors de la vraie région d'incertitude.

- Faisons l'hypothèse que MC avec 20 000 échantillonnages nous fournit la solution la plus proche de la solution exacte. En effet, on a l'assurance de ne pas avoir de trajectoires erronées et les figures 7.1, 7.2, 7.3, 7.4, 7.10 et 7.12 montrent que cette méthode nous donne la solution la plus large sans trajectoire erronée.
- Les figures 7.3 et 7.4 donnent que les résultats obtenus avec MC pour 200 et 20 000 échantillonnages sont pratiquement identiques. Cela montre que la simulation par MC se prête bien au type de système d'équations utilisé ici. En effet, on n'a aucune amélioration notable de l'espace des solutions malgré deux ordres de grandeurs de différence pour le nombre d'échantillonnages, cela nous conforte dans l'hypothèse que l'on obtient avec MC une solution très proche de la meilleure possible.
- Pour l'arithmétique affine, les figures 7.1 et 7.2 montrent que les courbes donnant les valeurs maximales se situent au-dessus des autres. Cela vient probablement de trajectoires erronées qui sont incluses dans la surface de résolution, à cause de la propagation d'erreur inhérente à cette méthode. Pour les courbes correspondant au minimum avec la même méthode, on peut constater que certaines trouvées avec MC ne sont pas reprises dans le résultat par arithmétique affine. De plus, pour certaines valeurs de temps, même les valeurs sans incertitude ne sont pas reprises dans la solution. Nous nous retrouvons donc avec une méthode qui génère des trajectoires superflues et qui n'en contient pas certaines. On peut donc en déduire que la simulation par arithmétique affine donne des résultats peu précis.
- Les figures 7.1 et 7.2 montrent que pour Qua.Si., la surface couverte par la solution donne les mêmes résultats que MC avec 20 000 échantillons puis devient un peu plus étroite sur la fin. La simulation par Qua.Si. semble donner des résultats d'une précision meilleure qu'avec l'arithmétique affine. On a également la confirmation que Qua.Si. sous-évalue l'étendue de la surface contenant les solutions, mais d'une façon faible et régulière.
- On peut voir sur les graphiques des figures 7.5 et 7.7 qu'utiliser l'amélioration Qua.Si+MC donne des résultats presque identiques à ceux donnés par Qua.Si. Cependant, si on regarde d'un peu plus près, les figures 7.6 et 7.8, nous renseignent que donner comme point de départ les conditions initiales trouvées par la partie MC permet d'obtenir une surface couvrant plus de solutions que Qua.Si., malgré un faible nombre de trajectoires calculées par MC (100 et 200 en comptant les extrémités). De même, si on fait la comparaison entre Qua.Si+MC (100 it. et 200 it.) et les deux autres simulations par MC (200 et 20 000) (voir figures 7.9 et 7.11), on obtient une différence très faible pour certains points (figures 7.10 et 7.12), visible seulement en zoomant sur les figures. Nous pouvons donc dire qu'au niveau précision, la méthode Qua.Si+MC est meilleure que Qua.Si. mais moins bonne que MC.
- Regardons l'évolution au cours du temps de Qua.Si+SA. On peut voir sur les graphiques des figures 7.13 et 7.14 que les solutions qui ont été trouvées avec Qua.Si+SA englobent

moins de trajectoires que celles trouvées avec Qua.Si+MC, malgré un nombre d'itérations identique. De plus, à l'instar de la simulation par arithmétique affine, certains points de la simulation sans incertitude, qui devraient se trouver dans la surface trouvée lors de la simulation, n'apparaissent pas dans l'intervalle. On peut cependant remarquer que Qua.Si+SA ne produit pas de trajectoire erronée.

La qualité des résultats est cependant pondérée par le temps de calcul : la résolution par mathématique affine est presque instantanée. Pour MC, cela ne dépend que du nombre d'échantillonnages que l'on veut effectuer, l'algorithme de résolution d'ODE de Matlab étant assez rapide. Par contre, pour Qua.Si., cela peut durer plusieurs heures, voire quelques jours sur la machine *Aster* du Centre de Calcul de l'ULB, en ne faisant qu'un seul appel à la routine d'optimisation. Cela montre bien la nécessité d'utiliser les améliorations proposées pour diminuer le temps de calcul.

Utilisons un critère plus objectif que le temps de calcul qui peut varier selon la machine et donnons le nombre d'opérations en virgule flottante qu'ont nécessité les simulations.

Décrivons les tables 7.3 et 7.4 :

1. La colonne SA/MC nous donne le nombre de flops pour la partie MC ou SA (non-basée sur les gradients) de la simulation.
2. La colonne résolution nous donne le nombre de flops pour la partie utilisant la routine d'optimisation (basée sur les gradients).
3. La dernière nous donne le total des deux autres.

méthode	SA/MC	résolution	total
Qua.Si	N.A.	39 Gflops	39 Gflops
MC(200)	45 Mflops	N.A.	45 Mflops
MC(20 000)	1,7 Gflops	N.A.	1,7 Gflops
Affine	N.A.	0,1 Mflops	0,1 Mflops
Qua.Si+MC(100)	24 Mflops	7 Gflops	7,02 Gflops
Qua.Si+SA	0,2 Gflops	4,8 Gflops	5 Gflops

TAB. 7.3 – Comparaison du nombre de flops pour le transformateur, profil jours de semaines

Les tables 7.3 et 7.4 nous confirment ce qui avait été pressenti avec l'estimation approximative du temps de calcul sur *Aster* :

- La simulation par mathématique affine est beaucoup plus rapide que les autres, l'utilisateur emploiera cette méthode s'il désire obtenir des résultats immédiats.
- La deuxième méthode en rapidité d'exécution est MC avec 200 échantillonnages.
- En troisième place vient la méthode par MC avec 20 000 échantillonnages. On peut remarquer que la rapidité d'exécution est linéaire en le nombre d'échantillonnages.

méthode	SA/MC	résolution	total
Qua.Si	N.A.	49 Gflops	49 Gflops
MC(200)	44,8 Mflops	N.A.	44,8 Mflops
MC(20 000)	1,7 Gflops	N.A.	1,7 Gflops
Affine	N.A.	0,1 Mflops	0,1 Mflops
Qua.Si+MC(100)	22 Mflops	7,76 Gflops	7,79 Gflops
Qua.Si+SA	0,2 Gflops	8,5 Gflops	8,7 Gflops

TAB. 7.4 – Comparaison du nombre de flops pour le transformateur, profil week-end et jours fériés

- Il est difficile de trancher au niveau rapidité d'exécution entre Qua.Si+MC et Qua.Si+SA. Suivant le profil de charge du transformateur, le nombre d'opérations peut être deux fois moins grand avec Qua.Si+SA que Qua.Si+MC, ou bien supérieur d'environ 1 Gflops. On peut cependant noter que le nombre d'opérations effectuées avec Qua.Si+MC est fort semblable avec les deux profils de charge. On peut aussi remarquer que la partie SA nécessite un nombre d'opérations plus grand de deux ordres de grandeurs que la partie MC. Cela est dû à plusieurs raisons : la partie SA nécessite plus de tirages aléatoires par itération que MC, ainsi que deux résolutions du système d'ODE contre deux pour MC. On peut également voir pour ces deux méthodes que le nombre d'opérations de seconde partie de l'algorithme est plus petit de deux ordres de grandeurs que Qua.Si. L'objectif d'avoir une méthode sensiblement plus rapide que Qua.Si. est donc bien atteint.
- Le nombre d'opérations effectuées par la partie d'optimisation basée sur les gradients de Qua.Si+MC et Qua.Si+SA ont des valeurs similaires. On peut supposer que la vitesse d'exécution de la routine d'optimisation est peu sensible aux conditions initiales que l'on lui donne comme point de départ pour la recherche d'extrema.
- En dernier vient la version originale de la méthode Qua.Si. Le nombre important d'opérations vient du fait que l'algorithme effectue un nombre exponentiel en le nombre de variables d'état de fois la routine d'optimisation.

7.2 Modèle de Lotka-Volterra

7.2.1 Le modèle

Le modèle de Lotka-Volterra [Jac91] est un modèle mathématique qui modélise le système proie – prédateur. Il consiste en deux espèces animales dont l’une est le prédateur et l’autre la proie, qui dispose elle-même d’une quantité inextinguible de nourriture. Le système d’ODE qui le représente est :

$$\begin{cases} \dot{y}_1 = k_1 y_1 - k_2 y_1 y_2 \\ \dot{y}_2 = k_3 y_1 y_2 - k_4 y_2 \end{cases}$$

où

- y_1 représente la densité de population de la proie,
- y_2 représente la densité de population du prédateur,
- k_1 représente le taux de natalité de la proie,
- k_2 représente le taux de capture de la proie,
- k_3 représente le taux de natalité du prédateur,
- k_4 représente le taux de mortalité du prédateur.

7.2.2 L’expérience

Nous allons maintenant étudier le comportement des méthodes sur le système de Lotka-Volterra, avec les conditions initiales floues suivantes [Bon03b].

$$\begin{cases} 1 \leq y_1(0) \leq 50 & k_1 = k_4 = 1 \\ 2 \leq y_2(0) \leq 60 & k_2 = k_3 = 0,01 \\ 0 \leq t \leq 8 \end{cases}$$

Discussion des résultats

Les figures de cette section montrent les résultats de la simulation du modèle de Lotka-Volterra. On distinguera les figures représentant la simulation de la variable y_1 et celles de la simulation de y_2 .

En plus de Qua.Si, Deux simulations pour MC ont été effectuées, une avec 200 échantillonnages l’autre avec 20 000. Pour les améliorations Qua.Si+MC et Qua.Si+SA, 200 échantillonnages ont été effectués.

La figure 7.15 montre l’évolution temporelle du maximum de la variable d’état y_1 en faisant appel une fois à la routine d’optimisation, par passage dans la boucle, en prenant comme point de départ le milieu de la composante courante de la surface ; par rapport à trois appels, avec comme points de départ le milieu de la composante courante et les extrémités de la composante courante de la surface extérieure. Pour rappel, la notion de nombre d’appels à la routine d’optimisation est expliquée en 4.3. On peut voir sur la figure 7.15 que faire un seul appel par tour de boucle

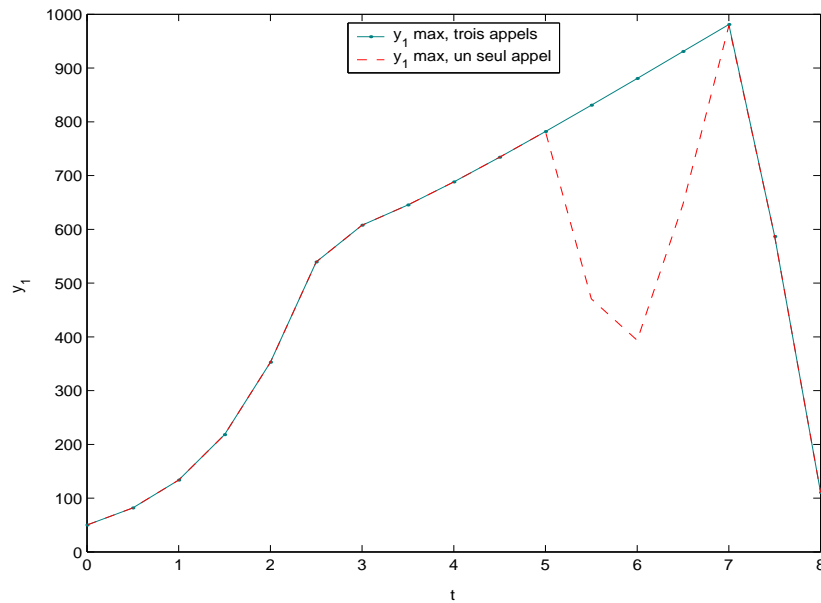


FIG. 7.15 – *Evaluation de l'évolution du maximum de $y_1(t)$ pour le système de Lotka-Volterra, avec les deux types d'exécution de Qua.Si*

est insuffisant. Les comparaisons avec Qua.Si se feront donc avec la simulation en appelant trois fois la routine d'optimisation.

Regardons si c'est le cas avec les Qua.Si+MC et Qua.Si+SA.

Regardons d'abord Qua.Si+MC (figures 7.16, 7.17, 7.18 et 7.19). Les figures 7.16, 7.17, 7.18 et 7.19 nous montrent qu'il est superflu d'effectuer plus d'un appel à la routine d'optimisation après avoir trouvé des conditions initiales de départ avec la partie MC de la procédure, les résultats trouvés sont identiques.

Effectuons la même comparaison avec l'amélioration Qua.Si+SA (figures 7.20, 7.21, 7.22 et 7.23). On peut se rendre compte que dans le cas de Qua.Si+SA, il vaut mieux effectuer trois fois la procédure d'optimisation.

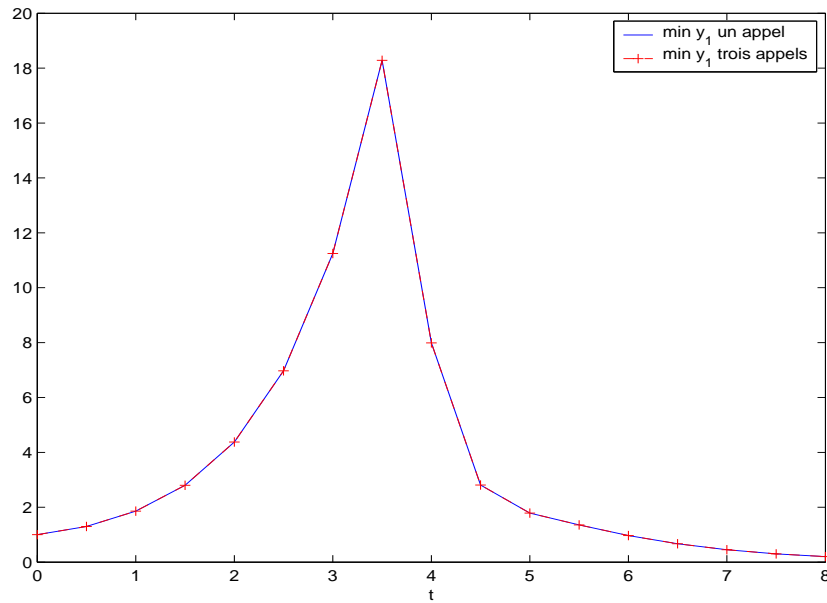


FIG. 7.16 – Comparaison Qua.Si+MC avec 1 et trois 3 appels à la routine d'optimisation (minimum de y_1)

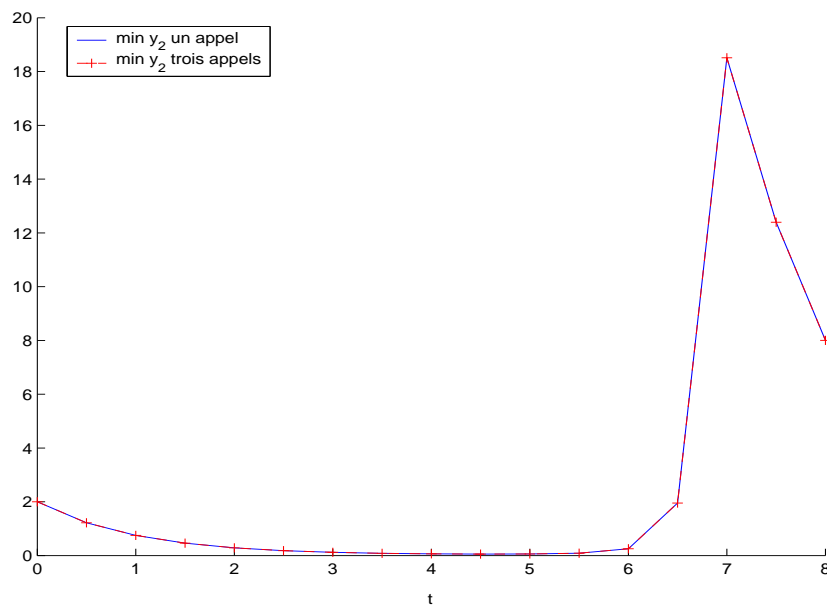


FIG. 7.17 – Comparaison Qua.Si+MC avec 1 et trois 3 appels à la routine d'optimisation (minimum de y_2)

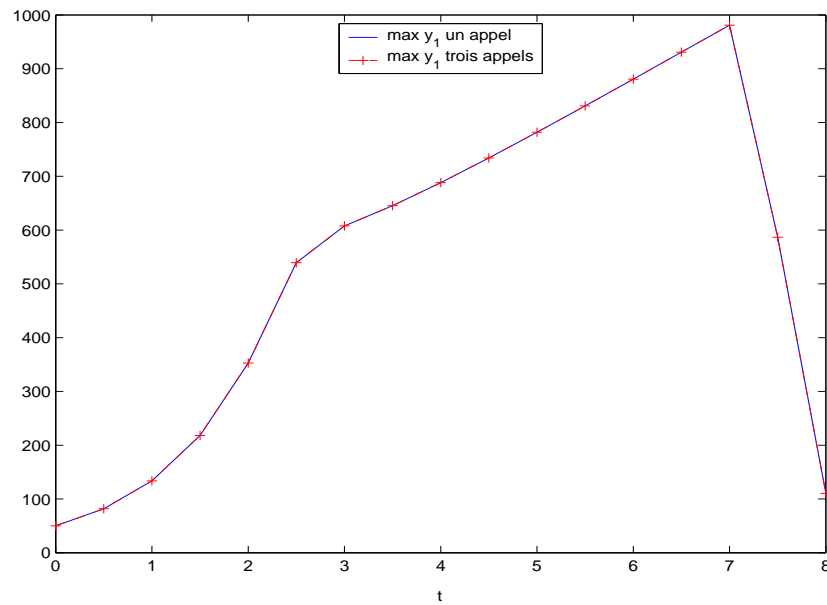


FIG. 7.18 – Comparaison Qua.Si+MC avec 1 et trois 3 appels à la routine d'optimisation (maximum de y_1)

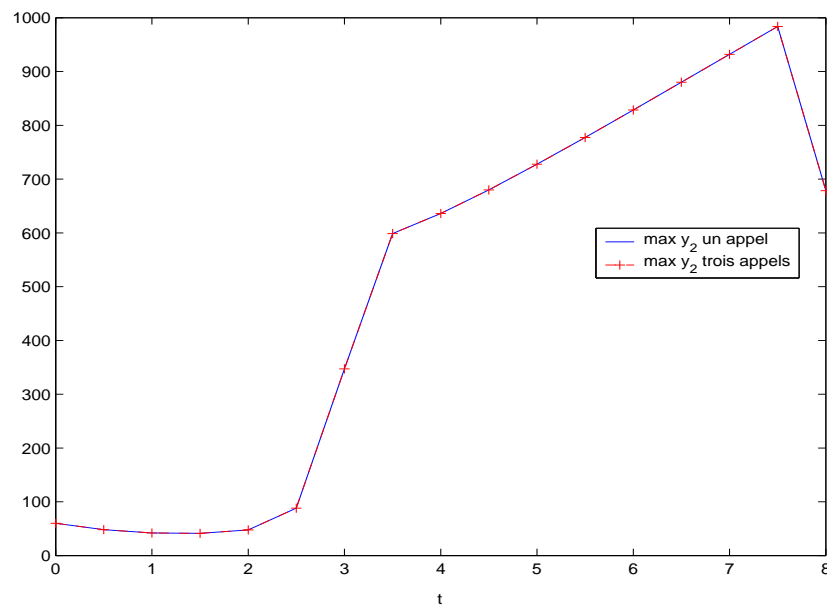


FIG. 7.19 – Comparaison Qua.Si+MC avec 1 et trois 3 appels à la routine d'optimisation (maximum de y_2)

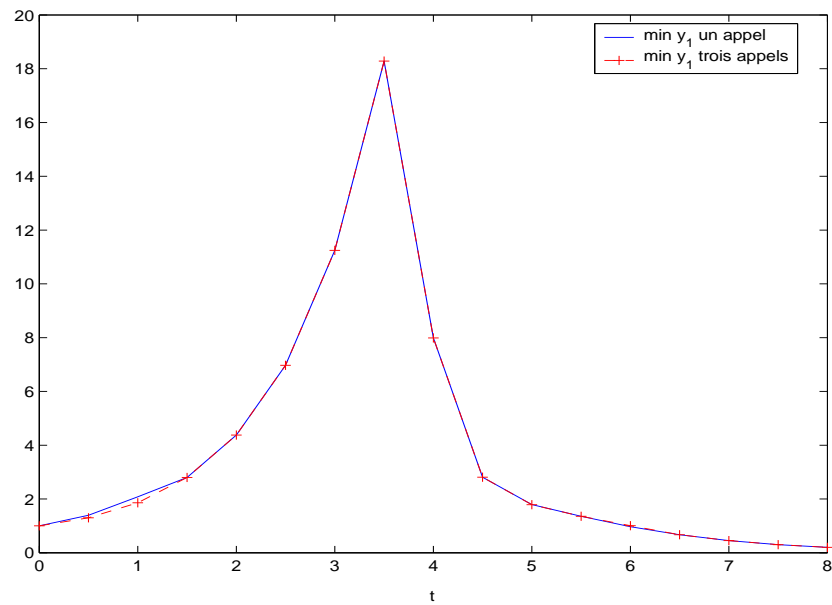


FIG. 7.20 – Comparaison Qua.Si+SA avec 1 et trois 3 appels à la routine d'optimisation (minimum de y_1)

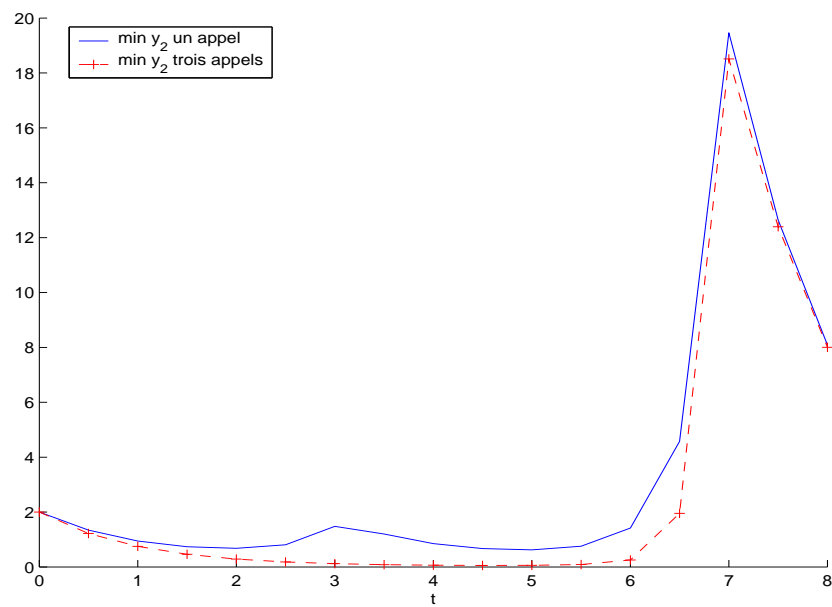


FIG. 7.21 – Comparaison Qua.Si+SA avec 1 et trois 3 appels à la routine d'optimisation (minimum de y_2)

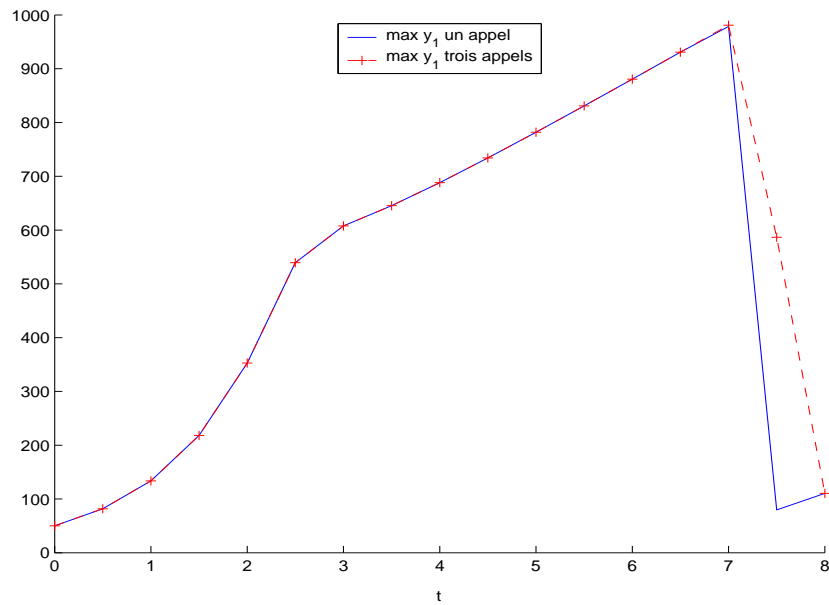


FIG. 7.22 – Comparaison Qua.Si+SA avec 1 et trois 3 appels à la routine d'optimisation (maximum de y_1)

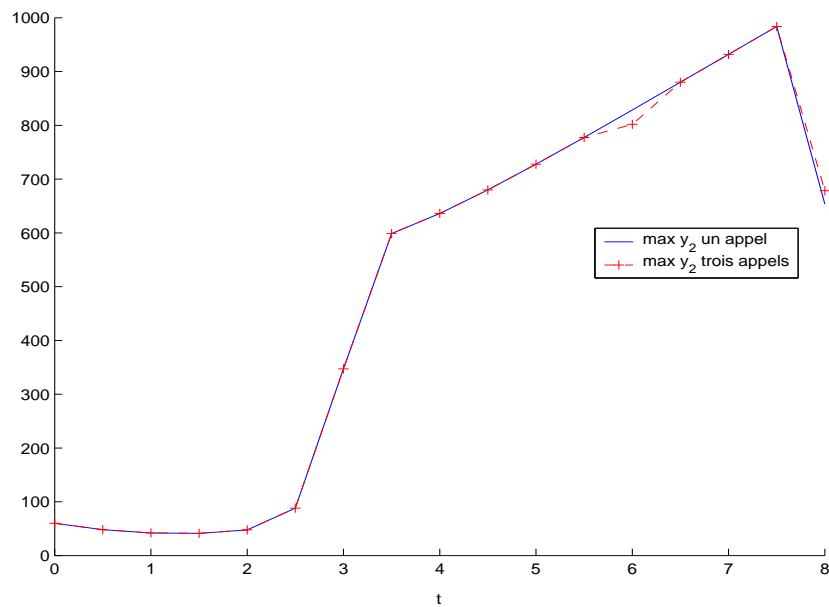


FIG. 7.23 – Comparaison Qua.Si+SA avec 1 et trois 3 appels à la routine d'optimisation (maximum de y_2)

Maintenant que nous savons comment employer Qua.Si, Qua.Si+MC et Qua.Si+SA, comparons maintenant la précision des différentes par rapport aux autres.

Les figures 7.24 et 7.25 montrent respectivement les résultats de la simulation de y_1 et y_2 en utilisant la mathématique affine.

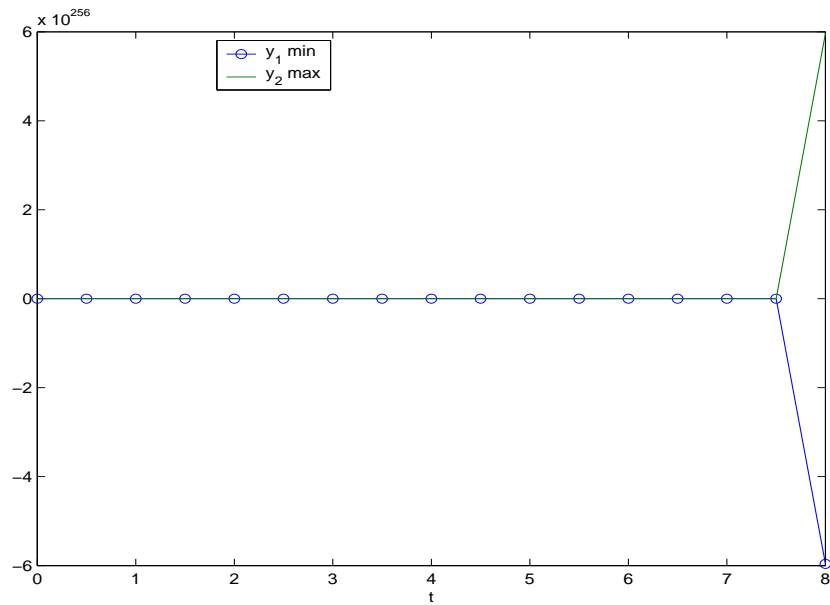


FIG. 7.24 – Lotka-Volterra – résultats par mathématique affine pour y_1

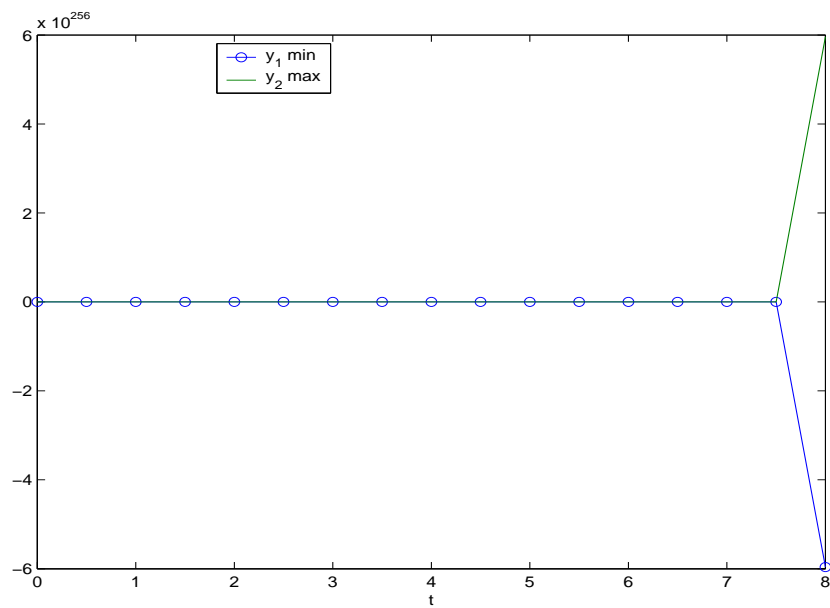


FIG. 7.25 – Lotka-Volterra – résultats par mathématique affine pour y_2

Les figures 7.26, 7.27, 7.28 et 7.29 montrent la simulation du minimum de y_1 et y_2 avec MC pour 200 et 20 000 échantillonnages et Qua.Si.

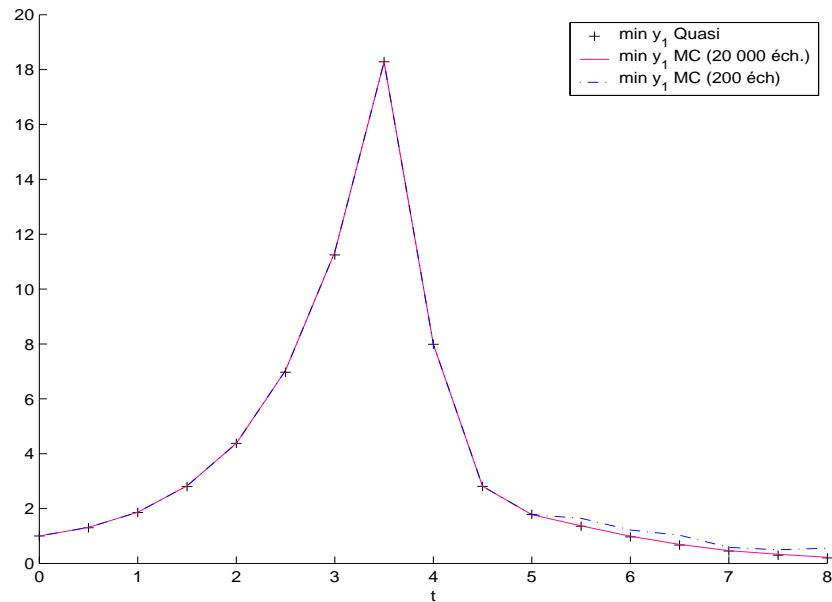


FIG. 7.26 – Comparaison Qua.Si. – Monte-Carlo pour Lotka-Volterra (minimum de y_1)

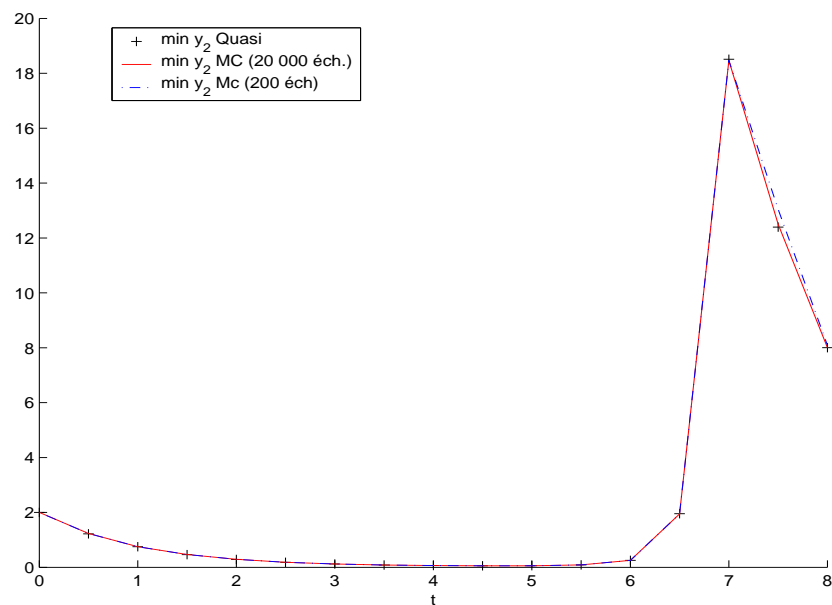
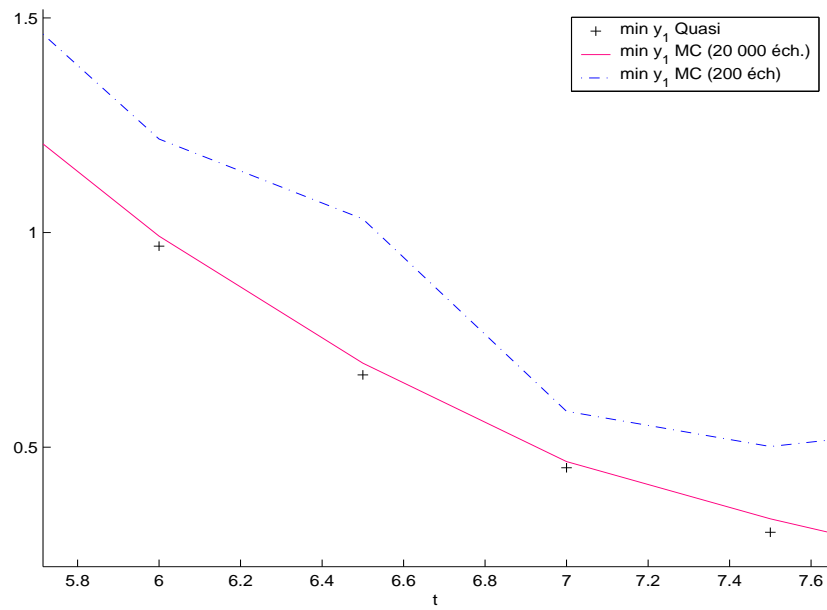
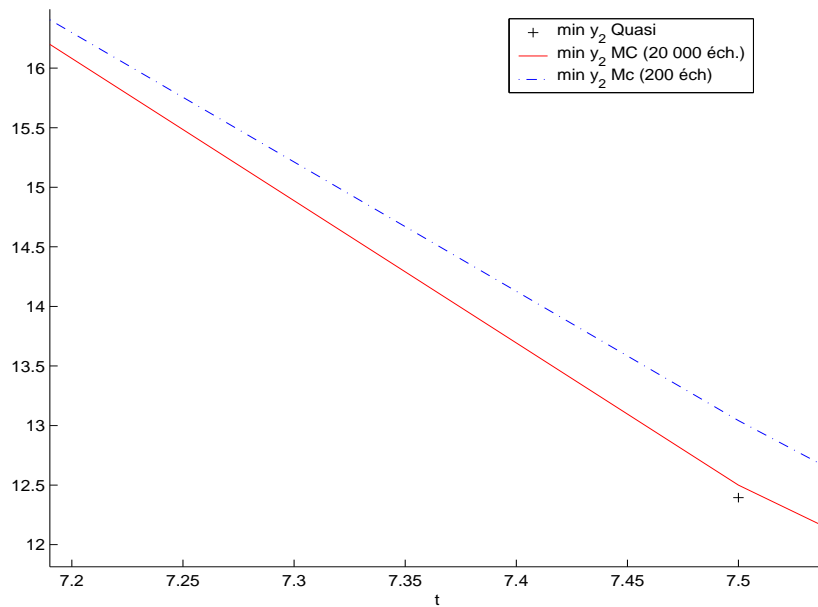
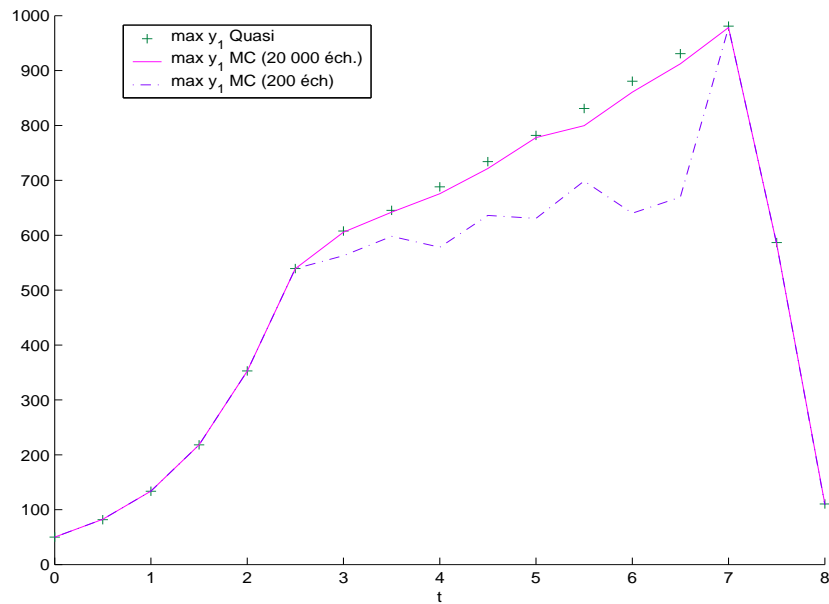
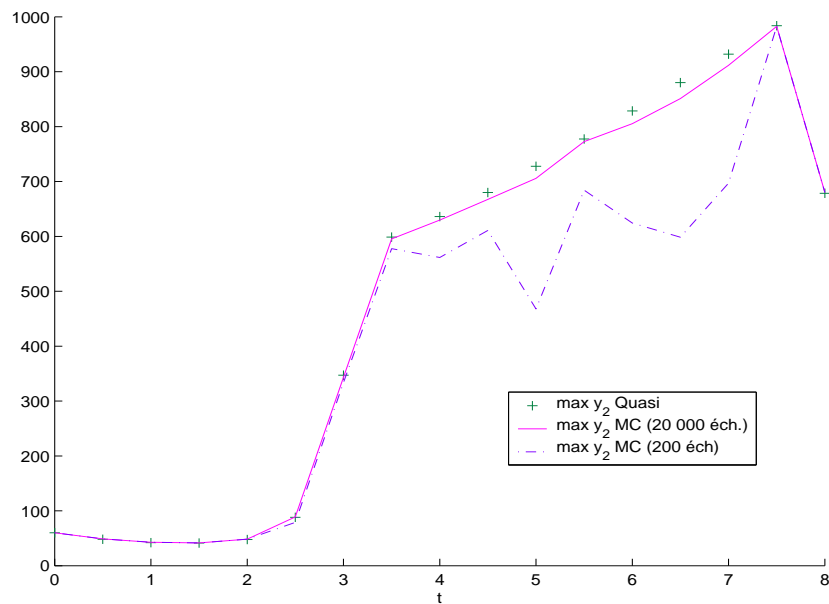


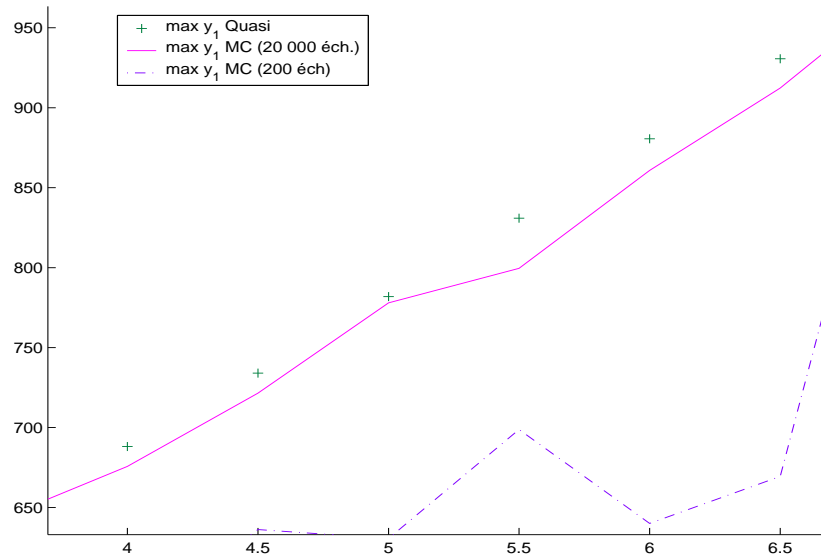
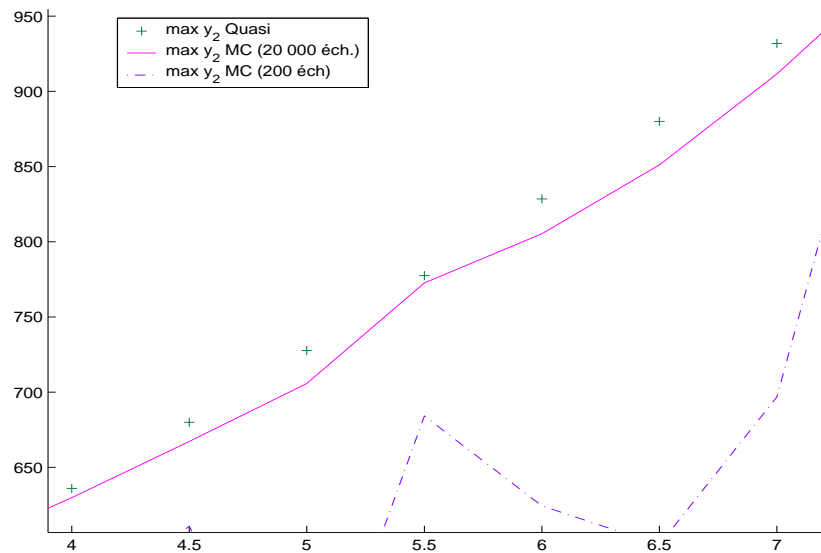
FIG. 7.27 – Comparaison Qua.Si. – Monte-Carlo pour Lotka-Volterra (minimum de y_2)

Les figures 7.30, 7.31, 7.32 et 7.33 montrent la simulation du minimum de y_1 et y_2 avec MC pour 200 et 20 000 échantillonnages et Qua.Si.

FIG. 7.28 – *Comparaison Qua.Si. – Monte-Carlo pour Lotka-Volterra (minimum de y_1 , zoom)*FIG. 7.29 – *Comparaison Qua.Si. – Monte-Carlo pour Lotka-Volterra (minimum de y_2 , zoom)*

Les figures 7.34, 7.35, 7.36 et 7.37 montrent la comparaison entre Qua.Si, Qua.Si+MC et Qua.Si+SA

FIG. 7.30 – Comparaison Qua.Si. – Monte-Carlo pour Lotka-Volterra (maximum de y_1)FIG. 7.31 – Comparaison Qua.Si. – Monte-Carlo pour Lotka-Volterra (maximum de y_2)

FIG. 7.32 – Comparaison Qua.Si. – Monte-Carlo pour Lotka-Volterra (maximum de y_1 , zoom)FIG. 7.33 – Comparaison Qua.Si. – Monte-Carlo pour Lotka-Volterra (maximum de y_2 , zoom)

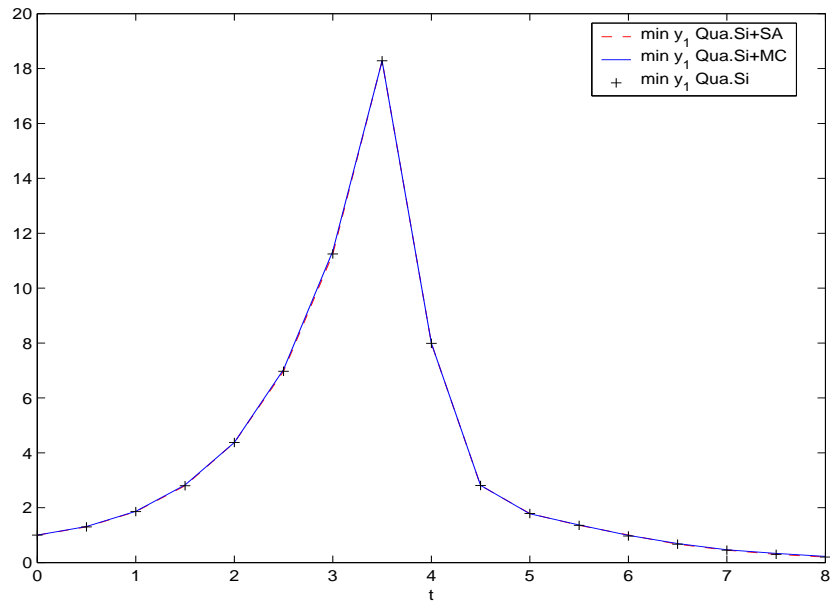


FIG. 7.34 – Comparaison Qua.Si, Qua.Si+MC et Qua.Si+SA (minimum de y_1)

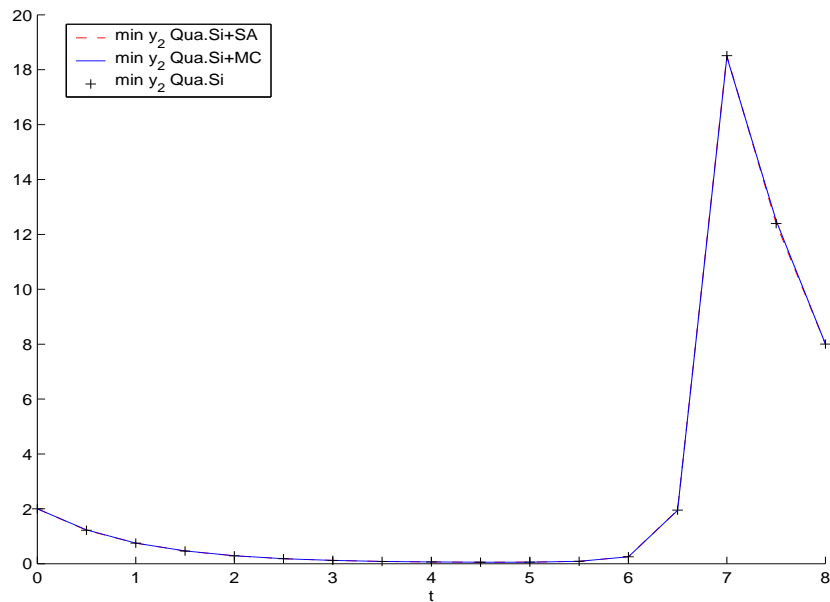
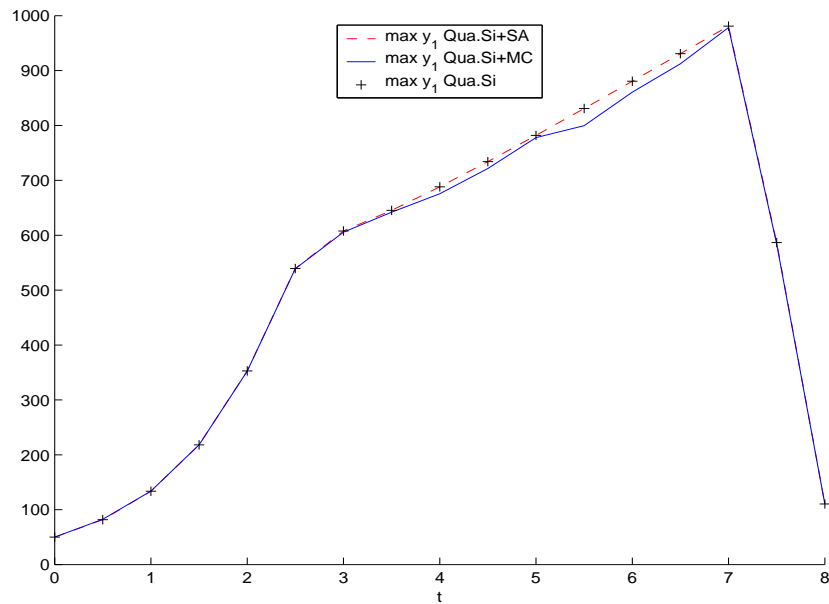
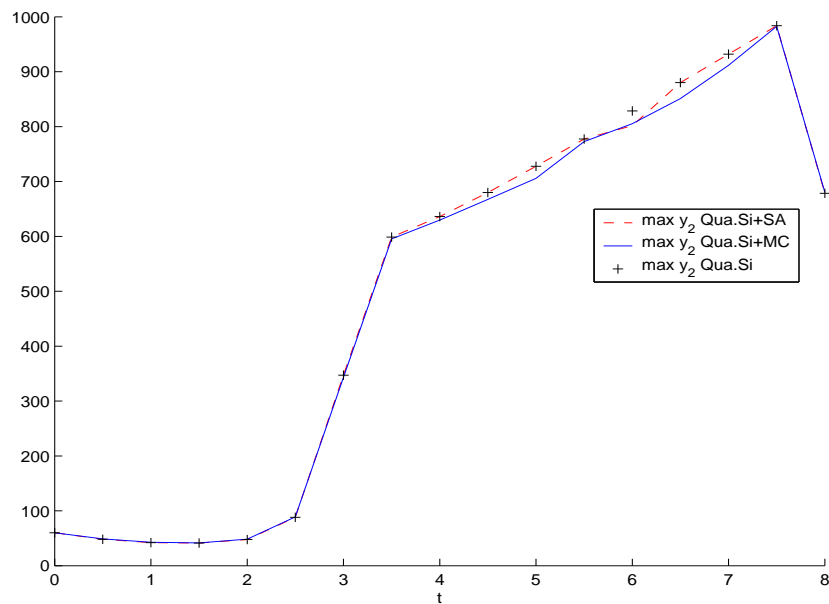


FIG. 7.35 – Comparaison Qua.Si, Qua.Si+MC et Qua.Si+SA (minimum de y_2)

FIG. 7.36 – Comparaison Qua.Si, Qua.Si+MC et Qua.Si+SA (maximum de y_1)FIG. 7.37 – Comparaison Qua.Si, Qua.Si+MC et Qua.Si+SA (maximum de y_2)

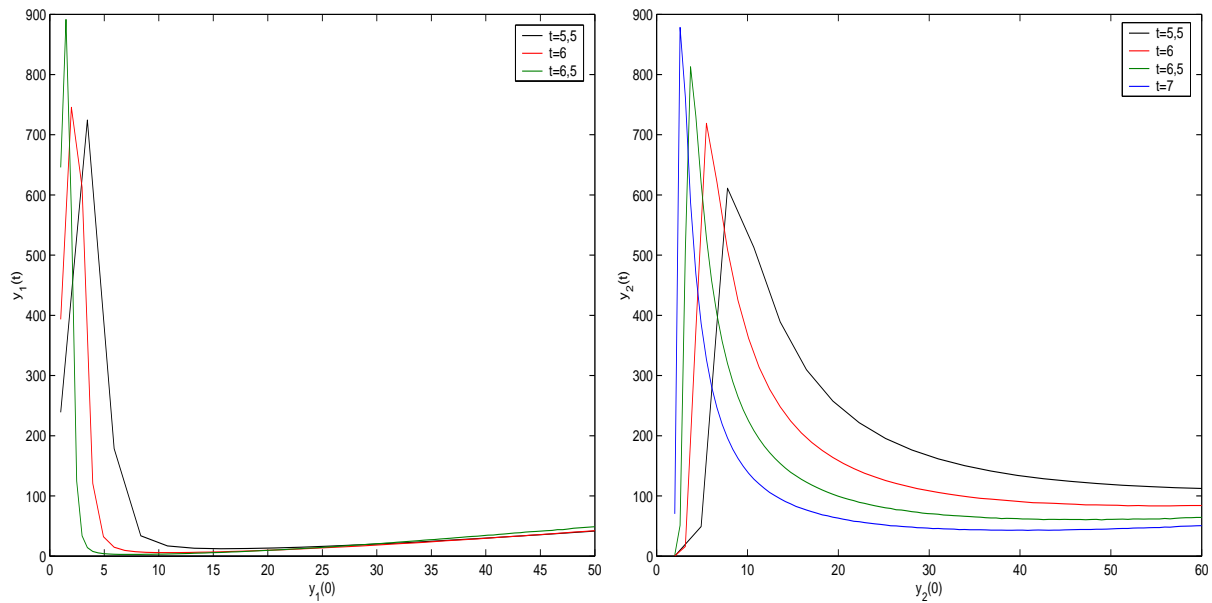
Discutons maintenant de la précision des résultats obtenus. Comme précédemment (section 7.1.3), nous savons déjà que les simulations par Qua.Si., Qua.Si+MC Qua.Si+SA et MC ne généreront jamais de région contenant des trajectoires ne correspondant à aucune solution.

- Faisons l’hypothèse que Qua.Si avec trois appels à la routine d’optimisation nous donne la solution la plus proche de la solution exacte. On peut avancer cela car on a de ne pas avoir de trajectoires erronées et les figures 7.15, 7.26, 7.27, 7.28, 7.29, 7.30, 7.31, 7.32, 7.33 7.34, 7.35, 7.36 et 7.37 montrent que Qua.Si fournit la solution la plus étendue sans trajectoire erronée.
- Nous pouvons voir sur les figures 7.24 et 7.25 que la surface obtenue en résolvant le système par arithmétique affine est totalement erronée. Cela vient du fait que la simulation par arithmétique affine ne tient pas suffisamment compte de l’interaction entre les variables. Nous allons donc obtenir une erreur qui va grandir de façon exponentielle. On peut déduire des simulations pour le transformateur et pour Lotka-Volterra que la simulation par arithmétique affine ne convient pas pour les systèmes dont l’interaction entre les variables est grande. En effet, malgré la complexité apparente du modèle du transformateur, ce n’était pas le cas.
- Il y a une nette différence entre la précision de la simulation par MC suivant que l’on prenne 200 ou 20 000 échantillonnages. Cependant même avec 20 000 échantillonnages, on obtient une solution plus étroite que celle de Qua.Si (figures 7.26, 7.27, 7.28 et 7.29).
- Les figures 7.34, 7.35, 7.36 et 7.37 permettent de dire qu’il y a peu de différence entre les résultats obtenus par Qua.Si, Qua.Si+MC et Qua.Si+SA (en effectuant les trois appels par tour de boucle à la routine d’optimisation) Cependant, sur les figures 7.36 et 7.37, on peut voir que Qua.Si+SA donne un solution plus précise que Qua.Si+MC, mais moins que Qua.Si.
- Il est nécessaire de faire trois appels à la routine d’optimisation pour Qua.Si. et Qua.Si+SA mais cela n’est pas nécessaire pour Qua.Si+MC (figures 7.15, 7.16, 7.17, 7.18, 7.19, 7.20, 7.21, 7.22 et 7.23).
- La difficulté de trouver de bonnes solutions à certaines valeurs de t peut s’expliquer grâce à la figure 7.38. On peut y voir que pour certaines valeurs de t , pour y_1 , les conditions initiales donnant les solutions maximales sont concentrées sur une région très réduite. On peut encore mieux se rendre compte de la difficulté de trouver ces conditions initiales en sachant qu’il a fallu calculer la valeur de $y(t)$ pour 100 points appartenant aux intervalles pour trouver les pics.

Nous allons maintenant comparer le nombre d’opérations en virgule flottante pour les simulations effectuées, à l’exception de celle par mathématique affine. Les résultats donnés par cette dernière n’étant pas assez bons en comparaison avec ceux des autres simulations.

Discutons des valeurs de la table 7.5 :

- La simulation par MC pour 200 échantillonnages est la plus rapide des méthodes qui

FIG. 7.38 – Lotka-Volterra – variation de la solution $y(t)$ pour certaines valeurs fixées de t

méthode	SA/MC	résolution	total
Qua.Si (1 appel)	N.A.	30,9 Mflops	30,9 Mflops
Qua.Si (3 appels)	N.A.	83,9 Mflops	83,9 Mflops
MC(200)	1,2 Mflops	N.A.	1,2 Mflops
MC(20 000)	127 Mflops	N.A.	127 Mflops
Qua.Si+MC, 1 appel	1,3 Mflops	11 Mflops	12,5 Mflops
Qua.Si+MC, 3 appels	1,2 Mflops	33 Mflops	34 Mflops
Qua.Si+SA, 1 appel	5,1 Mflops	18 Mflops	23 Mflops
Qua.Si+SA, 3 appels	5,3 Mflops	40 Mflops	45 Mflops

TAB. 7.5 – Comparaison du nombre de flops pour Lotka-Volterra

donnent des résultats cohérents.

- La deuxième méthode en rapidité d'exécution est Qua.Si+MC, avec un seul appel à la routine d'optimisation par tour de la boucle de la partie optimisation.
- En troisième vient Qua.Si+SA. Cela peut s'expliquer par le nombre de tirages aléatoires, de résolutions d'équations différentielles et de comparaisons effectuées par la partie SA.
- Qua.Si. avec un appel à la routine d'optimisation, par composante de la surface extérieure occupe la quatrième place en terme de rapidité d'exécution.
- Les trois méthodes suivantes, par ordre, sont Qua.Si+MC, Qua.Si+SA et Qua.Si. avec trois appels à la fonction d'optimisation. Nous obtenons alors un nombre d'opérations entre deux et trois fois plus grand que leurs homologues avec un seul appel.

- En dernier vient MC avec 20 000 échantillonnages.

7.3 Equation de Laplace

L'équation de Laplace est utilisée, entre autres, en électrostatique pour modéliser le potentiel électrique d'une région de l'espace ne contenant pas de charge [NAV03]. Son équation est donnée par

$$u_{xx} + u_{yy} = 0 \quad (7.1)$$

La simulation a été faite avec ces conditions aux bords :

$$\begin{aligned} u(x,0) \in [10, 30] \quad \text{et} \quad u(x,4) \in [170, 190] \quad \text{pour} \quad 0 < x < 4 \\ u(0,y) \in [70, 90] \quad \text{et} \quad u(4,y) \in [-10, 10] \quad \text{pour} \quad 0 < y < 4 \end{aligned} \quad (7.2)$$

Une grille de 5×5 a été employée comme quadrillage du carré contenant les ordonnées et les abscisses des points à calculer.

7.3.1 Discussion des résultats

La PDE a été résolue en utilisant Gauss-Seidel, Jacobi et la méthode itérative par point fixe. La simulation a été effectuée en utilisant Qua.Si adapté aux PDE, MC avec 200 échantillonnages et MC avec 20 000 échantillonnages.

Regardons tout d'abord les valeurs, sans incertitude, prenons les valeurs précises telles que les intervalles définissant l'incertitude sont centrés en la valeur sans incertitude. La figure 7.39 montre le profil de u en fonction de x et y . Les tables 7.6, 7.7 et 7.8 nous donnent les valeurs de u calculées sans incertitude respectivement avec Gauss-Seidel, Jacobi et la méthode itérative par point fixe, pour les conditions aux bords déterministes suivantes :

$$\begin{aligned} u(x,0) = 20 \quad \text{et} \quad u(x,4) = 180 \quad \text{pour} \quad 0 < x < 4 \\ u(0,y) = 80 \quad \text{et} \quad u(4,y) = 0 \quad \text{pour} \quad 0 < y < 4 \end{aligned} \quad (7.3)$$

	1	2	3	4	5
1	130.0000	180.0000	180.0000	180.0000	90.0000
2	80.0000	112.8129	111.7415	84.2636	0
3	80.0000	79.5544	69.9115	45.3129	0
4	80.0000	55.6258	43.1258	27.0986	0
5	50.0000	20.0000	20.0000	20.0000	10.0000

TAB. 7.6 – Résolution déterministe avec Gauss-Seidel

Les tables 7.9 et 7.10 nous donnent certaines valeurs de u par la méthode de Gauss-Seidel.

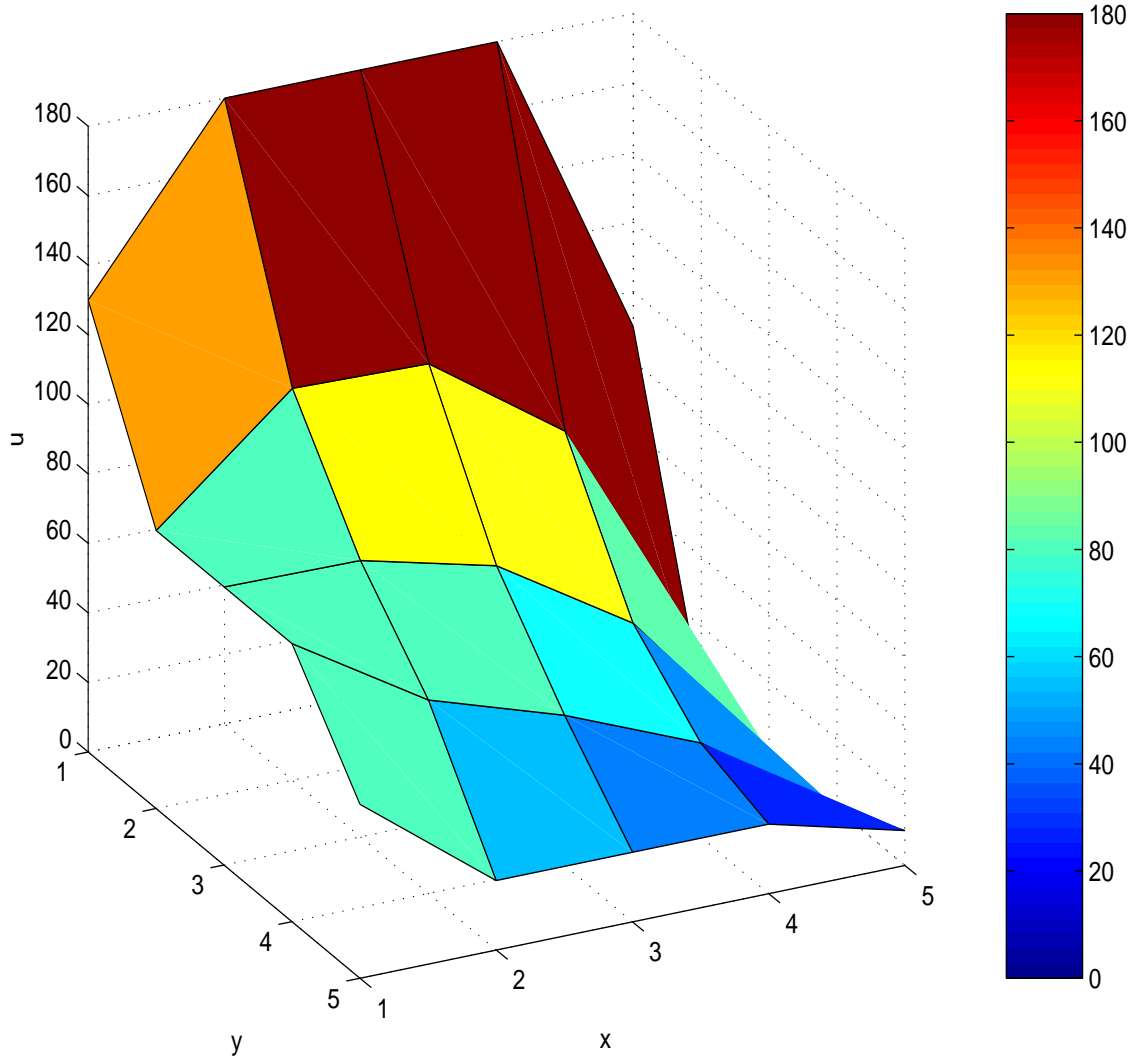


FIG. 7.39 – graphique de la solution de la PDE pour les conditions initiales (7.4)

	1	2	3	4	5
1	130.0000	180.0000	180.0000	180.0000	90.0000
2	80.0000	112.7546	111.6490	84.1832	0
3	80.0000	79.5061	69.7949	45.2204	0
4	80.0000	55.6118	43.0776	27.0403	0
5	50.0000	20.0000	20.0000	20.0000	10.0000

TAB. 7.7 – Résolution déterministe avec Jacobi

	1	2	3	4	5
1	130.0000	180.0000	180.0000	180.0000	90.0000
2	80.0000	112.8568	111.7854	84.2855	0
3	80.0000	79.6422	69.9993	45.3568	0
4	80.0000	55.7136	43.2136	27.1425	0
5	50.0000	20.0000	20.0000	20.0000	10.0000

TAB. 7.8 – Résolution déterministe avec la méthode par point fixe

	1	2	3	4	5
1	120.0000	170.0000	170.0000	170.0000	80.0000
2	70.0000	101.6568	98.6210	76.6766	-10.0000
3	70.0000	68.0457	56.1707	38.0853	-10.0000
4	70.0000	44.4742	30.0099	19.5139	-10.0000
5	40.0000	10.0000	10.0000	10.0000	0

TAB. 7.9 – Minimum pour u , avec Gauss-Seidel, avec Qua.Si.

	1	2	3	4	5
3	79.9749	79.5255	69.8771	45.2948	-0.0251
4	79.9749	55.5978	43.0929	27.0796	-0.0251

TAB. 7.10 – Maximum pour u , avec Gauss-Seidel(extrait), avec Qua.Si.

	1	2	3	4	5
1	120.0000	170.0000	170.0000	170.0000	80.0000
2	70.0000	101.5890	98.4946	76.5890	-10.0000
3	70.0000	67.9590	56.0352	37.9590	-10.0000
4	70.0000	44.4462	29.9233	19.4462	-10.0000
5	40.0000	10.0000	10.0000	10.0000	0

TAB. 7.11 – Minimum pour u , pour Jacobi, avec Qua.Si.

	1	2	3	4	5
3	79.9749	79.4773	69.7605	45.2024	-0.0251
4	79.9749	55.5838	43.0448	27.0214	-0.0251

TAB. 7.12 – Maximum pour u , pour Jacobi (extrait), avec Qua.Si.

	1	2	3	4	5
1	120.0000	170.0000	170.0000	170.0000	80.0000
2	70.0000	102.8568	101.7854	74.2856	-10.0000
3	70.0000	69.6422	59.9994	35.3568	-10.0000
4	70.0000	45.7137	33.2137	17.1425	-10.0000
5	40.0000	10.0000	10.0000	10.0000	0

TAB. 7.13 – Minimum pour u , pour la méthode par point fixe, avec Qua.Si.

	1	2	3	4	5
1	140.0000	190.0000	190.0000	190.0000	100.0000
2	90.0000	122.8568	121.7853	94.2855	10.0000
3	90.0000	89.6421	79.9992	55.3568	10.0000
4	90.0000	65.7135	53.2135	37.1425	10.0000
5	60.0000	30.0000	30.0000	30.0000	20.0000

TAB. 7.14 – Maximum pour u , pour la méthode par point fixe, avec Qua.Si.

Les tables 7.11 et 7.12 nous donnent certaines valeurs de u par la méthode de Jacobi.

Les tables 7.13 et 7.14 nous donnent les valeurs de u par la méthode par point fixe

Les tables 7.15 et 7.16 nous donnent les valeurs de u en utilisant MC pour 200 échantillonnages avec la méthode par point fixe.

	1	2	3	4	5
1	120.0000	170.0000	170.0000	170.0000	80.0000
2	70.0000	102.8568	101.7854	74.2856	-10.0000
3	70.0000	69.6422	59.9994	35.3568	-10.0000
4	70.0000	45.7137	33.2137	17.1425	-10.0000
5	40.0000	10.0000	10.0000	10.0000	0

TAB. 7.15 – Minimum pour u , pour la méthode par point fixe, avec MC pour 200 échantillonnages

Les tables 7.17 et 7.18 nous donnent les valeurs de u en utilisant MC pour 20 000 échantillonnages avec la méthode par point fixe.

On peut remarquer, à la lecture des tables

- Lorsque l'on utilise les trois méthodes avec les conditions aux bords sans incertitude, les tables 7.6, 7.7 et 7.8 montrent qu'il y a peu de différence à utiliser l'une des trois méthodes.
- Pour la recherche du minimum, comparer les valeurs contenues dans les tables 7.6, 7.7 et 7.8 et 7.9, 7.11 et 7.13 nous donnent également des différences minimales entre les méthodes.

	1	2	3	4	5
1	140.0000	190.0000	190.0000	190.0000	100.0000
2	90.0000	122.8571	121.7857	94.2857	10.0000
3	90.0000	89.6428	80.0000	55.3571	10.0000
4	90.0000	65.7142	53.2142	37.1428	10.0000
5	60.0000	30.0000	30.0000	30.0000	20.0000

TAB. 7.16 – Maximum pour u , pour la méthode par point fixe, avec MC pour 200 échantillonnages

	1	2	3	4	5
1	120.0000	170.0000	170.0000	170.0000	80.0000
2	70.0000	102.8568	101.7854	74.2856	-10.0000
3	70.0000	69.6422	59.9994	35.3568	-10.0000
4	70.0000	45.7137	33.2137	17.1425	-10.0000
5	40.0000	10.0000	10.0000	10.0000	0

TAB. 7.17 – Minimum pour u , pour la méthode par point fixe, avec MC pour 20 000 échantillonnages

	1	2	3	4	5
1	140.0000	190.0000	190.0000	190.0000	100.0000
2	90.0000	122.8571	121.7857	94.2857	10.0000
3	90.0000	89.6428	80.0000	55.3571	10.0000
4	90.0000	65.7142	53.2142	37.1428	10.0000
5	60.0000	30.0000	30.0000	30.0000	20.0000

TAB. 7.18 – Maximum pour u , pour la méthode par point fixe, avec MC pour 20 000 échantillonnages

- Pour la recherche du maximum, comparer les valeurs contenues dans les tables 7.6, 7.7 et 7.8 et 7.10, 7.12 et 7.14 permet de dire que les méthodes de Gauss-Seidel et Jacobi ne conviennent pas pour les résoudre l'équation afin de trouver le maximum. En effet, nous nous retrouvons avec des valeurs très proches, parfois inférieurs à celles trouvées sans incertitude pour les méthodes de Gauss-Seidel et Jacobi mais la méthode par point fixe nous donne des valeurs de u partout supérieures. Cela peut venir du fait que pour les méthodes de Gauss-Seidel et Jacobi, qu'une condition suffisante de convergence est d'avoir une matrice d'entrée dont les éléments de la diagonale sont strictement supérieurs à la somme des autres éléments, pour chaque ligne. Cela n'est pas le cas dans le cadre de notre PDE.

- En raison du point précédent, nous allons donc uniquement considérer la méthode par point fixe pour résoudre l'équation avec Qua.Si.
- En termes de précision de solution pour le minimum, la méthode Qua.Si est équivalente à quelques décimales près à MC avec 200 et 20 000 échantillonnages (tables 7.13, 7.15 et 7.17).
- En termes de précision de solution pour le maximum, la méthode Qua.Si est équivalente à quelques décimales près à MC avec 200 et 20 000 échantillonnages (tables 7.13, 7.15 et 7.18).

La différence de quelques décimales qui existe entre la simulation par Qua.Si. et MC peut s'expliquer de deux façons, qui peuvent être conjuguées :

1. Nous n'utilisons pas le gradient pour la résolution par Qua.Si, ce qui risque d'entraîner une précision moins grande qu'avec une méthode basée sur les gradients.
2. La PDE qui a été simulée est une équation quasi-linéaire très simple. Il se peut donc que l'équation ait des propriétés proches des PDE linéaires au niveau des conditions aux bords minimisant et maximisant la fonction. Or nous considérons en premier lieu pour la simulation par MC, les valeurs correspondant aux extrémités des intervalles. Cela expliquerait alors les résultats identiques avec 200 et 20 000 échantillonnages : les conditions aux bords donnant les extrema de la fonction sont celles correspondant aux extrémités des intervalles.

Au niveau du temps de calcul, il est intéressant de remarquer que c'est avec Qua.Si. que l'on obtient la résolution la plus rapide : 8,1 Mflops avec Qua.Si. contre 13,1 Mflops pour MC avec 200 échantillonnages et 1,3 Gflops pour MC avec 20 000 échantillonnages.

Conclusion

Dans ce document, nous avons donné un aperçu général des méthodes de simulation avec incertitude, nous nous sommes penchés avec plus d'attention sur l'approche Qua.Si. et proposé des extensions à cette approche et étudié leur comportement en comparaison avec des méthodes existantes.

Nous avons tiré certains enseignements des expériences réalisées :

- La complexité des systèmes à simuler risque de rendre l'approche Qua.Si. peu compétitive si le but recherché est d'obtenir des résultats dans un temps très court.
- L'approche par mathématique affine, malgré sa rapidité, donne des résultats assez imprécis et est à proscrire dans le cas où des variables interagissent entre elles.
- La non-linéarité du système rend la résolution de celui-ci plus difficile et risque de nécessiter un surcroît de calculs. De plus, cela rend problématique la résolution par des méthodes stochastiques, comme MC, si l'incertitude est représentée de manière non-probabiliste.
- L'utilisation de méthodes d'optimisation stochastiques en extension à l'approche Qua.Si. nous permet de donner une bonne précision des résultat en un temps raisonnable dans le cas de systèmes non-linéaires.
- Il peut être plus intéressant d'utiliser une méthode d'optimisation stochastique simple, avec peu de paramètres à gérer qu'une méthode plus compliquée, dont la précision dépend fortement des paramètres.
- Dans le cas où les équations sont linéaires ou ont des caractéristiques qui les rendent presque linéaires, la simulation par Monte-Carlo peut s'avérer intéressante au vu du temps de calcul et de la précision
- Dans le cas des équations différentielles partielles, on a pu voir qu'il vaut mieux employer une méthode de résolution spécifique à la forme du problème qu'une méthode générale.
- Toujours pour les équations différentielles partielles, la linéarité des équations est un facteur important pour la facilité de résolution.

En résumé, on peut dire qu'il n'y a pas de recette miracle qui donne des résultats précis à tous les coups. Il faut également établir le compromis entre la vitesse d'exécution et la précision des résultats. Chaque problème nécessite une étude spécifique pour savoir quelle méthode employer pour atteindre l'objectif voulu (précision/rapidité).

L'utilisation des méthodes d'optimisation stochastique en apport aux méthodes existantes peut être un sujet d'étude intéressant. N'oublions pas que l'adaptation de ces méthodes à la simulation avec incertitude n'est pas triviale. Pour terminer, l'étude de la simulation avec imprécision sur les équations différentielles partielles mérite une attention particulière à cause de la complexité introduite par l'utilisation de plusieurs variables indépendantes.

Annexe A

Méthodes numériques

A.1 Méthode d'Euler [MF99]

Soit $[a, b]$ l'intervalle sur lequel nous voulons résoudre la solution au problème $\dot{y} = f(t, y)$ avec $y(a) = y_0$. Divisons ensuite l'intervalle $[a, b]$ en M sous-intervalles égaux et prenons les points

$$t_k = a + kh \quad \text{pour } k = 0, 1, \dots, M \quad \text{où } h = \frac{b - a}{M}. \quad (\text{A.1})$$

La valeur h est appelée le **pas**. Il faut ensuite résoudre approximativement

$$\dot{x} = f(t, y) \quad \text{sur } [t_0, t_M] \quad \text{avec } y(t_0) = y_0. \quad (\text{A.2})$$

Supposons que $y(t)$, $\dot{y}(t)$ et $\ddot{y}(t)$ soient continues et utilisons le développement de Taylor de $y(t)$ en prenant $t = t_0$. Pour toute valeur t il existe une valeur c_1 entre t_0 et t telle que

$$y(t) = y(t_0) + \dot{y}(t_0)(t - t_0) + \frac{\ddot{y}(c_1)(t - t_0)^2}{2}. \quad (\text{A.3})$$

Substituons dans cette équation $\dot{y}(t_0) = f(t_0, y(t_0))$ et $h = t_1 - t_0$, nous obtenons comme résultat pour $y(t_1)$:

$$y(t_1) = y(t_0) + hf(t_0, y(t_0)) + \ddot{y}(c_1) \frac{h^2}{2}. \quad (\text{A.4})$$

Si le pas h est choisi suffisamment petit, on peut négliger la composante de second ordre, la saute commence à prendre et nous obtenons

$$y_1 = y_0 + hf(t_0, y_0) \quad (\text{A.5})$$

appelée **approximation d'Euler**.

Il ne nous reste plus qu'à itérer le processus pour générer la suite de points approximant la courbe de la fonction objectif. La forme générale d'un pas d'itération de la méthode d'Euler est donc

$$t_{k+1} = t_k + h, \quad y_{k+1} = y_k + hf(t_k, y_k) \quad \text{pour } k = 0, 1, \dots, M - 1. \quad (\text{A.6})$$

A.3 Méthode de Kuhn-Tucker

La méthode de Kuhn-Tucker est la méthode utilisée dans la toolbox d'optimisation de Matlab pour l'optimisation continue avec contraintes [Mat].

Elle se base sur la résolution des équations de Kuhn-Tucker : Soit $f(x)$ la fonction à minimiser et $G_i(x), i = 1, \dots, n$ la matrice décrivant des contraintes sur x Les équation de Kuhn-Tucker sont décrites comme suit :

$$\nabla f(x^x) + \sum_{i=1}^n \lambda_i^* \nabla G_i(x^*) = 0 \quad (\text{A.12})$$

$$\lambda_i^* \nabla G_i(x^*) = 0 \quad \text{pour } i = 1, \dots, n \quad (\text{A.13})$$

$$\lambda_i^* \geq 0 \quad \text{pour } i = n_e + 1, \dots, n \quad (\text{A.14})$$

où x^* est un point solution, $i = n_e + 1, \dots, n$ sont les indices correspondant aux contraintes d'inégalités pour x et les λ_i^* sont des coefficients de Lagrange pour le point solution x^* .

Pour résoudre ces équations, la toolbox va faire appel à un algorithme de programmation non-linéaire séquentielle quadratique.

Annexe B

Les programmes Matlab développés

Cette annexe a pour but une description un peu plus technique des programmes Matlab qui ont été développés dans le cadre de ce mémoire, pour la résolution d'équations différentielles floues.

B.1 Qua.Si+MC et Qua.Si+SA

L'application fonctionne comme suit : après avoir rédigé les fichiers `expe.m` contenant les paramètres de simulation et les deux fichiers correspondant au modèle à simuler (ici `lotka.m` et `lotka_sto.m`), l'utilisateur peut lancer l'exécution du script principal `main_fast.m`, qui va d'abord lancer la partie stochastique non-basée sur les gradients. Celle-ci va fournir les conditions initiales de départ qui vont être utilisées par la partie basée sur les gradients. La routine `constr.m` fournie avec Matlab va trouver ensuite les extrema de la résolution d'ODE effectuée en appliquant le script `ode23.m` fourni avec Matlab au système augmenté fourni par `differx.m`, en tenant compte des gradients donnés par `differc.m`. Le modèle à simuler (ici `lotka.m`) étant fourni en paramètres à ces deux derniers scripts.

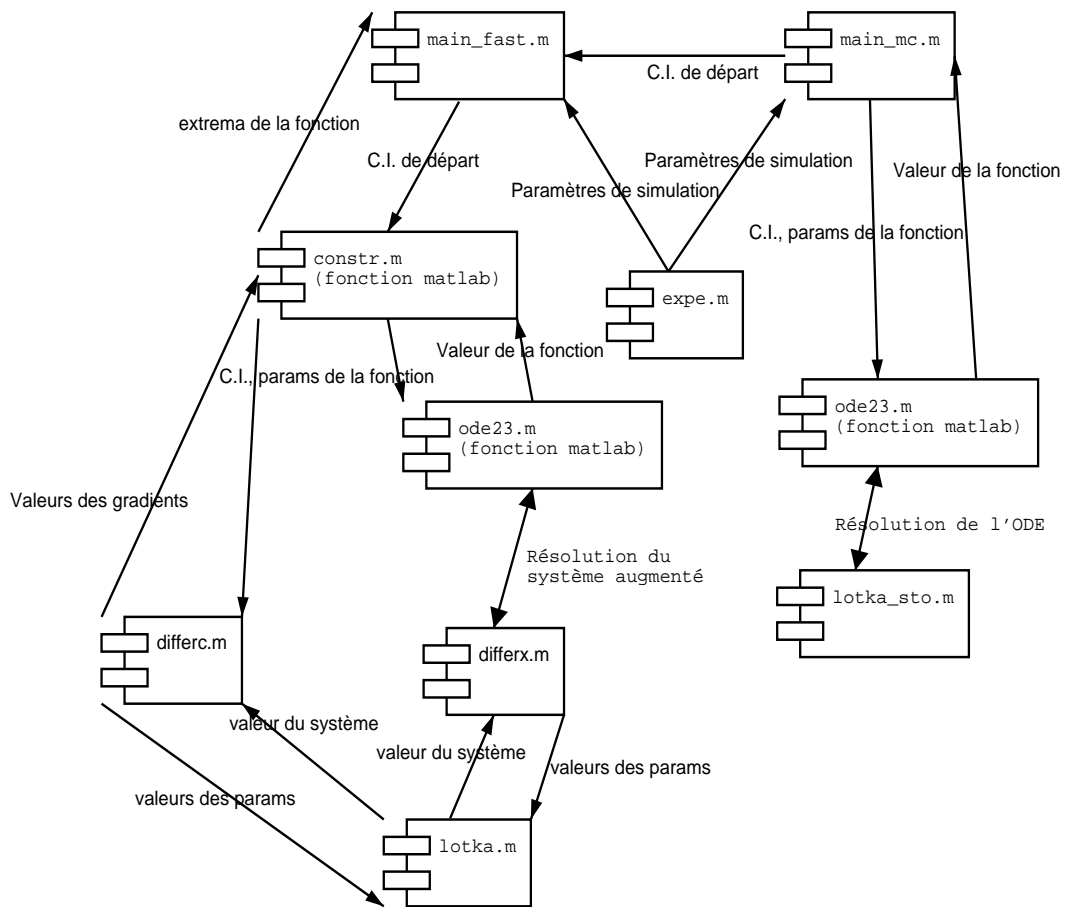


FIG. B.1 – Diagramme des composants de Qua.Si.+MC

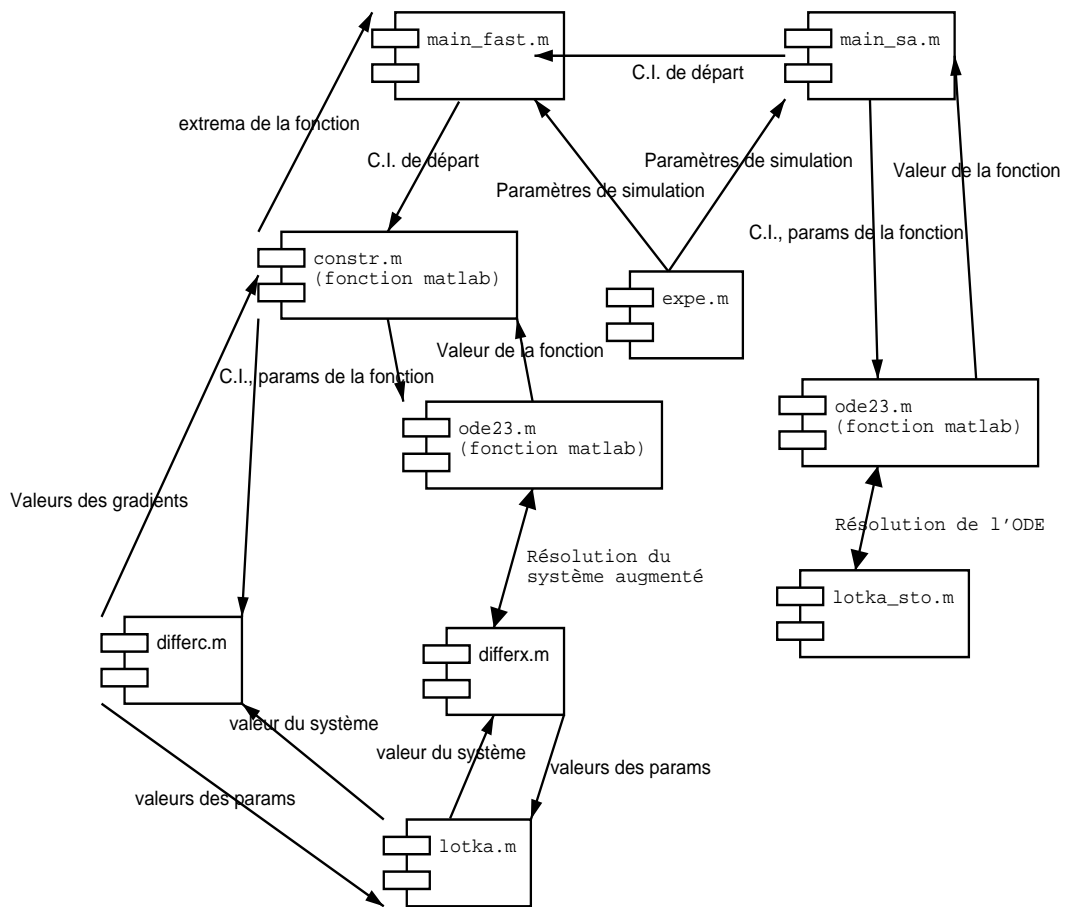


FIG. B.2 – Diagramme des composants de Qua.Si.+SA

B.2 Résolution de l'équation de Laplace floue

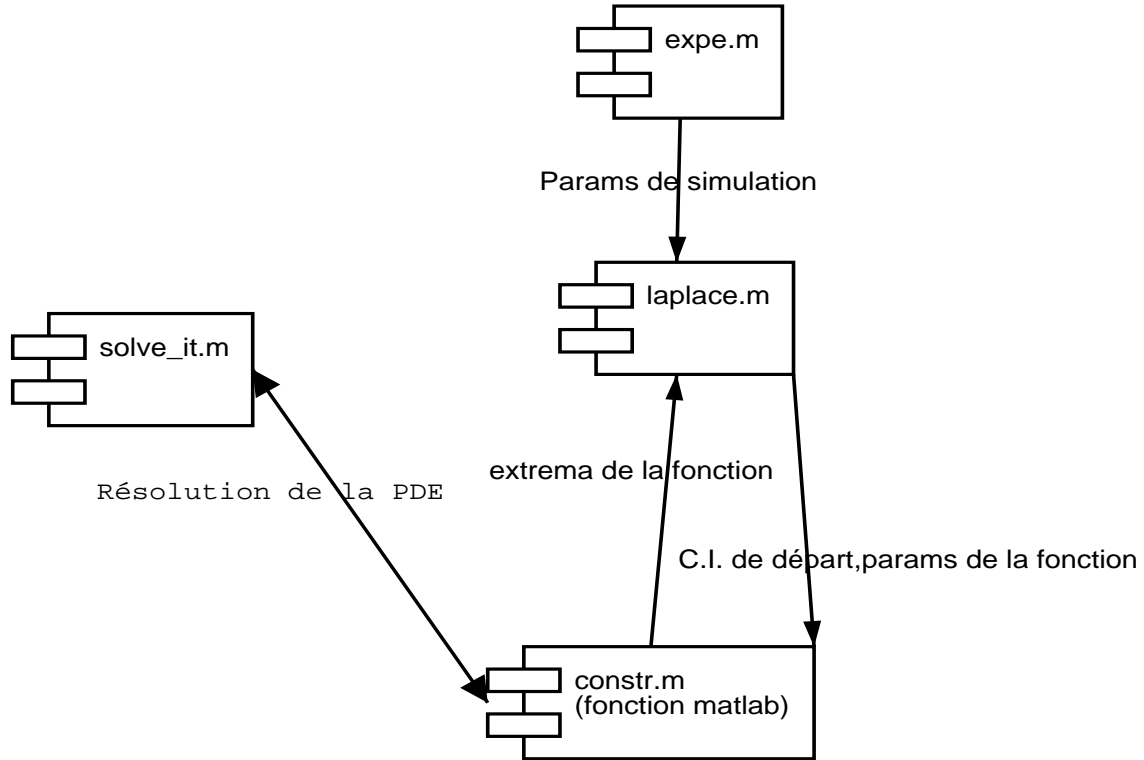


FIG. B.3 – Diagramme des composants de la résolution de l'équation floue de Laplace

L'application fonctionne de manière similaire : l'utilisateur doit rédiger le fichier **expe.m** qui contiendra les contraintes de l'équation ainsi que les paramètres de simulation. L'utilisateur va ensuite lancer le fichier principal **laplace.m** qui va rechercher les extrema de l'équation de Laplace en utilisant la fonction **constr.m** fournie avec Matlab pour trouver les extrema de la solution de la PDE. Cette solution est calculée à l'aide du script **solve_it.m**

Bibliographie

- [ACS94] M.V.A. Andrade, J.L.D. Comba, and J. Stolfi. Affine arithmetic. In *Abstracts of the International Conf. on Interval and Computer-Algebraic Methods in Science and Engineering (INTERVAL/94)*, pages 36–40, S. Petersburg, Russia, March 1994.
- [ATMVdlR00] J. Armengol, L. Travé-Massuyès, J. Vehì, and J.L. de la Rosa. A survey on interval model simulators and their properties related to fault detection. *Annual Reviews in Control*, 24(1):31–39, 2000.
- [BB94] A. Bonarini and G. Bontempi. A qualitative simulation approach for fuzzy dynamical models. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 4(4), 1994.
- [BdSSdW03] G. Bontempi, A. da Silva Soares, and M. de Wulf. Cours de calcul formet et numérique. Syllabus de Deuxième Candidature en Informatique – Université Libre de Bruxelles, 2003.
- [Bon96] G. Bontempi. Qua.si. III: A software tool for simulation of fuzzy dynamical systems. In A.Javor, A. Lehmann, and I. Molnar, editors, *Modeling and Simulation ESM 96 (Proceedings European Simulation Multiconference 1996)*, pages 615–619, Ghent, Belgium, 1996. SCS International.
- [Bon03a] G. Bontempi. Cours de modèles sotchastiques II. Syllabus de Deuxième Licence en Informatique – Université libre de Bruxelles, 2003.
- [Bon03b] G. Bontempi. Simulating continuous dynamical systems with uncertainty: the probability and the possibility approaches. in "Fuzzy Partial Differential Equations and Relational Equations", Editors: Masoud Nikraves and Lotfi A. Zadeh, Series Studies in Fuzziness and Soft Computing, Physica-Verlag, Springer, 2003. To appear.
- [Cer85] V. Cerny. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Application*, pages 41–51, 1985.
- [Coo92] Arnold Cook. *Ordinary differential equations*. Springer-Verlag, Berlin, 1992.

- [DK01] R.V. Demicco and G.J. Klir. Stratigraphic simulations using fuzzy logic to model sediment dispersal. *Journal of Petroleum Science and Engineering*, 31:135–155, 2001.
- [Doi99] J.P. Doignon. *Mathématique générale. Syllabus de Première Candidature en Informatique – Université Libre de Bruxelles*, 1999.
- [DP80] D. Dubois and H. Prade. *Fuzzy Sets and Systems*. Academic Press, New York, 1980.
- [Fau99] L. Fausett. *Applied Numerical Analysis Using MATLAB*. Prentice Hall, 1999.
- [Fis91] P. A. Fishwick. Fuzzy simulation: Specifying and identifying qualitative models. *International Journal of General Systems*, 19:295–316, 1991.
- [FS97] L.H. Figueiredo and J. Stolfi. Self-validated numerical methods and applications. In *Brazilian Mathematics Colloquium Monograph*, Rio de Janeiro, Brazil, 1997. IMPA.
- [Gar85] C. W. Gardiner. *Handbook of stochastic methods*. Springer–Verlag, Berlin, 1985.
- [GIVV02] V. Galdi, L. Ippolito, A. Vaccaro, and D. Villacci. The use of affine arithmetic for thermal state estimation of substation distribution transformers. *The International Journal for Computation and Mathematics in Electrical and Electronic Engineering (COMPEL)*, 2002.
- [Jac91] E. A. Jackson. *Perspective of nonlinear dynamics*. Cambridge University Press, 1991.
- [KG85] A. Kaufmann and M. Gupta. *Introduction to fuzzy arithmetic: theory and applications*. Van Nostran and Reinhold, New York, 1985.
- [KGV83] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, pages 671–680, 1983.
- [KK68] G. A. Korn and T. M. Korn. *Mathematical handbook for scientists and engineers*. McGraw-Hill, 1968.
- [Lue79] D.G. Luenberger. *Introduction to Dynamic Systems: Theory, Models, and Applications*. John Wiley and Son, November 1979.
- [Mat] Optimization toolbox. in Matlab Documentation The Mathworks Inc. <http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml>.
- [MF99] J.H. Mathews and K.D. Fink. *Numerical Methods Using MATLAB*. Prentice Hall, 3 edition, 1999.
- [Moo66] R. E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, 1966.
- [MRR⁺53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

-
- [MRSV86] D. Mitra, F. Romeo, and A.L. Sangiovanni-Vincentelli. Convergence of finite-time behaviour of simulated annealing. *Advances in Applied Probability*, 18:747–771, 1986.
- [NAV03] C.R. NAVE. Laplace’s and poisson’s equations. in hyperphysics :<http://hyperphysics.phy-astr.gsu.edu>, 2003.
- [Ngu78] H. T. Nguyen. A note on the extension principle for fuzzy sets. *Journal of Mathematical Analysis Applications*, 64(2):369–380, 1978.
- [Pap91] Papoulis. *Probability, Random Variables and Stochastic Proces*. McGRaw-Hill, 1991.
- [Pro99] Working Group K3 (Transformer Thermal Overload Protection). Adaptive transformer thermal overload protection. Technical report, IEEE, January 1999.
- [Sha76] G. Shafer. *A mathematical theory of evidence*. Princeton University Press, Princeton, 1976.
- [Sob90] K. Sobczyk. *Stochastic Differential Equations with applications to physics and engineering*. Kluwer Academic Publishers, Dordrecht, 1990.
- [Zad65] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.
- [Zad78] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1(1):3–28, 1978.
- [Zei76] B. Zeigler. *Theory of Modeling and Simulation*. Wiley, New-York, 1976.