Chapitre 1

Description de l'algorithme de simulation

1.1 Forme générale du modèle

L'environnement permet de simuler des modèles paramétrés de la forme :

$$\begin{cases} \partial_t U(t,x) & = & \Delta D(x,\Theta)U(t,x) + F(\Theta, U(t,x)) & , \text{ pour } x \in \Omega,\ t \in]0,T] \\ U(0,x) & = & U_0(x,\Theta) & , \text{ pour } x \in \Omega \\ \vec{\nabla} D(x,\Theta)U(t,x).\vec{n} & = & 0 & , \text{pour } x \in \delta\Omega,\ t \in]0,T] \end{cases} \tag{1.1}$$

avec :

$\Theta \in \mathbb{R}^p$

$U \in (t,\Omega) \mapsto \mathbb{R}^m$

1.2 Résolution numérique, simulation du système (1.1).

La simulation est basée sur une approche par éléments finis mise en oeuvre avec Freefem. La discrétisation temporelle est une θ-méthode, les non linéarités sous gérées par un algorithme de Newton-Raphson. Un pas de temps adaptatif permet d'accélérer les simulations.

1.2.1 Algorithme d'avancée en temps

Dans le cas du schéma implicite, on cherche à résoudre $\alpha(U_{t+\delta_t} - U_t) = \Delta D(x,\Theta)U_{t+\delta_t} + F(t+\delta_t, U_{t+\delta_t})$, avec $\alpha = 1/\delta_t$. En cas de non linéarité de F, on a recours à un algorithme de Newton Raphson. On note u_k l'itéré

courant, on cherche alors u_{k+1} l'itéré suivant comme solution du système linéarisé suivant :

$$\alpha U_{k+1} - \Delta D U_{k+1} - \partial_U F(t+\delta_t, U_k) U_{k+1} = F(t+\delta_t, U_k) - \partial_U F(t+\delta_t, U_k) U_k + \alpha U_k$$

Puis on écrit la formulation variationnelle de ce problème linéaire, on est alors amené à résoudre un système linéaire de la forme :

$$MX = Y \text{ avec } M_{ij} = \int_\Omega (\alpha - \partial_U F(t+\delta_t, U_k))\phi_i.\phi_j + \vec{\nabla} D\phi_i.\vec{\nabla}\phi_j dx \quad (1.2)$$

1.2.2 Ajustement de la taille du pas de temps

Un algorithme d'ajustement de la valeur de δ_t permet de réduire le temps de simulation. Pour cela, on est amené à préciser une tolérance représentant l'erreur numérique acceptable.

Chapitre 2

Couplage du simulateur au algorithme d'optimisation

2.1 Recherche du paramètre optimal

Lorsque nous cherchons les paramètres optimaux d'un modèle on peut choisir d'utiliser des méthodes de quasi-Newton. Ces méthodes nécessitent peu d'appels au simulateur pour converger vers un optimum local, en pratique quelques dizaines d'appels suffisent. Ces méthodes ont besoin d'évaluer, de manière précise, les valeurs de la fonction objectif et ses variations. Pour cela, MSE propose des algorithmes de différenciation permettant d'évaluer les variations de la fonction objectif comme solution d'EDP. Cette section montre que la mise oeuvre de ces algorithmes ne modifie pas significativement les performances du simulateur.

2.1.1 Calcul des gradients pour l'optimisation de système

On peut voir le modèle comme une fonction du paramètre Θ :

$$\begin{cases} S & : & \mathbb{R}^p \mapsto ((t, \Omega) \mapsto \mathbb{R}^m) \\ & & \Theta \to S(\Theta), \text{ la solution de (1.1) pour } \Theta. \end{cases} \tag{2.1}$$

Nommons, $\mathcal{L}$, la fonction associée au modèle (vraisemblance, log-vraisemblance, ou tout autre critére) :

$$\begin{cases} \mathcal{L} & : & ((t, \Omega) \mapsto \mathbb{R}^m) \mapsto \mathbb{R} \\ & & U \to \mathcal{L}(U), \text{ la vraisemblance associée à } U. \end{cases} \tag{2.2}$$

On peut maintenant écrire la fonction objectif, $\mathcal{O}$, que l'on cherche à optimiser :

$$\left\{ \begin{array}{lll} \mathcal{O} & : & (\mathbb{R}^p) \mapsto \mathbb{R} \\ & & \Theta \to \mathcal{O}(\Theta) = \mathcal{L}(S(\Theta)), \text{ la vraisemblance associée à } \Theta. \end{array} \right. \tag{2.3}$$

Les algorithmes de Quasi-Newton construisent une suite Θ_k qui convergent vers un extremum local de $\mathcal{O}$. Le calcule de Θ_{k+1} utilise $(\mathcal{O}(\Theta_l), \nabla\mathcal{O}(\Theta_l))$ pour $l \leq k$.

2.2 Expression de $\nabla\mathcal{O}(\Theta)$

$\nabla\mathcal{O}(\Theta) = (\partial_{\Theta_1}\mathcal{O}(\Theta), .., \partial_{\Theta_m}\mathcal{O}(\Theta))^t \in \mathbb{R}^p$, avec $\partial_{\Theta_i}\mathcal{O}(\Theta) = \mathcal{L}'(S(\Theta))(\partial_{\Theta_i}S(\Theta))$. on a :

* $\partial_{\Theta_i}S(\Theta)$ est une fonction $(t, \Omega) \mapsto \mathbb{R}^m$
* $\mathcal{L}'(S(\Theta))$ est la linéarisé de la vraisemblance en $S(\Theta)$.

2.2.1 calcul de $\partial_{\Theta_i}S(\Theta)$

On écrit le taux d'accroissement. On note $U = S(\Theta + \epsilon e_i)$ et $V = S(\Theta)$, on a $\partial_{\Theta_i}S(\Theta) = \lim_{\epsilon \to 0} \frac{U-V}{\epsilon}$, avec :

$$\left\{ \begin{array}{lll} \partial_t U & = & \Delta D(x, \Theta + \epsilon e_i)U(t,x) + F(\Theta + \epsilon e_i, U) \\ \partial_t V & = & \Delta D(x, \Theta)V(t,x) + F(\Theta, V) \\ U(0,x) & = & U_0(x, \Theta) \\ V(0,x) & = & U_0(x, \Theta + \epsilon e_i) \end{array} \right. \tag{2.4}$$

On fait un développement limité au première ordre en ϵ :

$$F(\Theta + \epsilon e_i, U) = F(\Theta, U) + \partial_{\Theta_i}F(\Theta, U)\epsilon + O(\epsilon^2)$$

et un en $(U - V)$:

$$F(\Theta, U) = F(\Theta, V) + [\partial_U F(\Theta, V)](U - V) + O(U - V)^2$$

pour la diffusion :

$$D(x, \Theta + \epsilon e_i) = D(x, \Theta) + \epsilon\partial_{\Theta_i}D(x, \Theta) + O(\epsilon^2)$$

puis avec $W = \frac{U-V}{\epsilon}$, on a

$$\partial_t W = \Delta D(x, \Theta)W + \partial_U F(\Theta, V)W + \partial_{\Theta_i}F(\Theta, V) + O(\epsilon) + O(U - V)$$

en passant à la limite, on obtient $\partial_{\Theta_i}S(\Theta)$ comme solution du système linéaire :

$$\left\{ \begin{array}{l} \partial_t W = \Delta D(x, \Theta)W + \partial_U F(\Theta, V)W + \partial_{\Theta_i}F(\Theta, V) \\ W(0,x) = \partial_{\Theta_i}U_0(x, \Theta) \end{array} \right. \tag{2.5}$$

Remarque importante : les systèmes (2.5) et (1.1) ont la même partie linéaire, donc la mise en oeuvre de l'algorithme de Newton-Raphton conduit à la résolution de deux systèmes linéaires dont seul le second membre diffère. C'est pourquoi le calcul des variations de la fonction à optimiser ne modifie pas significativement les performances du simulateur.