An Adaptive Local Grid Refinement Method for Nonlinear Filtering

Zhiqiang Cai, François Le Gland, Huilong Zhang

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An Adaptive Local Grid Refinement Method for Nonlinear Filtering

Zhiqiang Cai, Francois Le Gland, Huilong Zhang

N° 2679

Octobre 1995

PROGRAMME 5

ISSN 0249-6399

An Adaptive Local Grid Refinement Method for Nonlinear Filtering

Zhiqiang Cai*, Francois Le Gland**, Huilong Zhang***

Programme 5 — Traitement du signal, automatique et productique
Projet AS

Rapport de recherche n° 2679 — Octobre 1995 — 23 pages

Abstract: Numerical solution of the Zakai equation usually leads to large systems of equations which have to be solved at each time step. An algorithm for the numerical approximation of the Zakai equation is presented, based on discretization schemes provided by Kushner, and by LeGland. We use an *a posteriori* criterion based on truncation error, to localize refinement regions. We apply the fast adaptive composite grid (FAC) method introduced by McCormick for solving the resulting linear systems. Numerical tests are presented, which show the efficiency of the FAC method.

Key-words: diffusion processes, nonlinear filtering, Zakai equation, stochastic PDE, multigrid methods, FAC.

(Résumé : tsvp)

This work was partially supported by the Commission of the European Communities, under the SCIENCE project *SPDE's and Random Fields*, project number SC1–CT91–0627, and by the Army Research Office, under grant DAAH04–95–1–0164.

*Center for Applied Mathematical Sciences, University of Southern California, Los Angeles, CA 90089–1113, USA — zcai@cams.usc.edu
**IRISA / INRIA, Campus de Beaulieu, 35042 Rennes Cédex, France — legland@irisa.fr
***MAB, ERS CNRS 123, Université de Bordeaux I, 351 cours de la Libération, 33405 Talence Cédex, France — hzhang@math.u-bordeaux.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Une Méthode de Raffinement Local de Maillage pour le Filtrage Non–Linéaire

Résumé : La résolution numérique de l'équation de Zakai conduit généralement à des systèmes d'équations qui doivent être inversés à chaque pas de temps. Un algorithme pour l'approximation numérique de l'équation de Zakai est présenté, qui repose sur des schémas de discrétisation proposés par Kushner, et par LeGland. Nous utilisons un critère *a posteriori* fondé sur l'erreur de troncature, pour déterminer les régions à raffiner. Nous appliquons la méthode FAC (Fast Adaptive Composite) introduite par McCormick pour résoudre le système linéaire obtenu. Des tests numériques sont présentés, qui montrent l'efficacité de la méthode FAC.

Mots-clé : processus de diffusion, filtrage non–linéaire, équation de Zakai, EDP stochastique, méthode multigrille, FAC.

1 Introduction

Nonlinear filtering is about estimating the state of a noisy dynamical system, using noisy observations — see the model given by equations (1) and (2) below. It is well known that the Kalman filter, or the extended Kalman filter, provides in a recursive manner the expected value of the state of the system, given all the past available observations, see e.g. Jazwinski [6]. Numerically, this filter is simple to compute and therefore is widely used. On the other hand, the solution of the Zakai equation provides the complete conditional probability density of the state, given the observations. It is more general and in some cases, gives much better estimates of the state of the system than the extended Kalman filter. But question on how to solve efficiently the large linear systems of equations arising from the discretization of the Zakai equation, so as to satisfy as much as possible real–time constraints, still remains, in particular when the state space is multidimensional.

The basic remark which we are going to exploit, is that in most nonlinear filtering problems, and in particular when the observation noise is small, the conditional density is well–localized in some *small* region of the state space. This is our motivation to use local refinement techniques. One specific feature of our problem however, is that the conditional density, which is the solution of the Zakai equation, is designed to track the solution of a noisy state equation. For this reason, the adaptive refinement regions may have arbitrary shape, and generally could not be predicted in advance. We propose to use the `C` programing language and convenient data structure, in order to overcome the resulting programing difficulties, and to optimise the software efficiency.

In Section 2, we present the Zakai equation, and its discretization with respect to time and space. In particular, we propose a finite difference approximation scheme on a composite grid, which ensures positivity of the solution of the resulting approximate equation. In Section 3, we present our FAC algorithm for the Zakai equation, we describe the data structure which has been used in the implementation, and we give some numerical results.

2 The Zakai equation and its discretization

We are interested in the following model, with state equation

$$dX_t = b(X_t)\,dt + \sigma(X_t)\,dW_t \ , \tag{1}$$

and observation process

$$dY_t = g(X_t)\,dt + dV_t \ , \tag{2}$$

Here, $\{X_t\ ,\ t \geq 0\}$ and $\{Y_t\ ,\ t \geq 0\}$ take values in $\mathbf{R}^m$ and $\mathbf{R}^d$ respectively and X_0 has probability distribution $p_0(x)\,dx$. $\{W_t\ ,\ t \geq 0\}$ and $\{V_t\ ,\ t \geq 0\}$ are independent Wiener processes with the appropriate dimensions, and non–singular covariance matrix I (identity) and R respectively. Let $\mathcal{Y}_t = \sigma(Y_s\ ,\ 0 \leq s \leq t)$ denote the σ–field generated by the observations up to time t. The problem is to compute at each time t, the conditional density p_t of the state

X_t, given the past available observations $\mathcal{Y}_t$, so as to compute

$$\mathbf{E}[\phi(X_t) \mid \mathcal{Y}_t] = c_t \cdot \int_{\mathbf{R}^m} \phi(x)\, p_t(x)\, dx \ ,$$

for any test function ϕ (here c_t is a normalization constant). Under broad assumptions, the conditional density p_t is the unique solution of the following stochastic PDE, called the Zakai equation [14]

$$dp_t = L^*\, p_t\, dt + p_t\, g^*\, R^{-1}\, dY_t \ , \tag{3}$$

where L is the infinitesimal generator associated with the diffusion process $\{X_t\ ,\ t \geq 0\}$, i.e.

$$L = \tfrac{1}{2} \sum_{i,j=1}^{m} a_{i,j}(\cdot)\, \frac{\partial^2}{\partial x_i \partial x_j} + \sum_{i=1}^{m} b_i(\cdot)\, \frac{\partial}{\partial x_i} \ , \tag{4}$$

with $a = (a_{i,j}) = \sigma\, \sigma^*$. For basic references about nonlinear filtering and the Zakai equation, see Jazwinski [6], Pardoux [11], and Rozovskii [12].

2.1 Time discretization

We introduce a uniform partition of the time interval $[0, +\infty)$

$$0 = t_0 < t_1 < \cdots < t_n < \cdots$$

with constant time step $\Delta = t_{n+1} - t_n$. We use the splitting–up algorithm studied in Le-Gland [9], which combines an implicit Euler scheme for the prediction step, and a Bayes formula based on sampled observations :

$$z_n^\Delta = \frac{1}{\Delta}\, [Y_{t_n} - Y_{t_{n-1}}] = \frac{1}{\Delta} \int_{t_{n-1}}^{t_n} h(X_s)\, ds + \frac{1}{\Delta}\, [V_{t_n} - V_{t_{n-1}}] \ ,$$

for the correction step. Following [9], the conditional density p_{t_n} is approximated by p_n^Δ, where the transition from p_{n-1}^Δ to p_n^Δ is divided into the following two steps :

- In the *prediction* step, we solve the following Fokker–Planck equation, between times t_{n-1} and t_n

  $$\frac{\partial p_t^n}{\partial t} = L^*\, p_t^n \ ,$$

  with initial condition $p_{t_{n-1}}^n = p_{n-1}^\Delta$ at time $t = t_{n-1}$. The final value at time $t = t_n$ gives the prior estimate $p_{n-1/2}^\Delta = p_{t_n}^n$. The Fokker–Planck equation is further discretized by the Euler implicit scheme, which is unconditionally stable, i.e. we solve

  $$[I - \Delta\, L]^*\, p_{n-1/2}^\Delta = p_{n-1}^\Delta \ . \tag{5}$$

- In the *correction* step, we use the new observation z_n^Δ and the Bayes formula, to update the prior estimate $p_{n-1/2}^\Delta$, i.e. we compute

$$p_n^\Delta = c_n \; \Psi_n^\Delta \; p_{n-1/2}^\Delta \; , \tag{6}$$

where

$$\Psi_n^\Delta(x) = \exp \left\{ \; -\tfrac{1}{2} \, \Delta \, |z_n^\Delta - g(x)\,|_R^2 \; \right\} \; ,$$

and c_n is a normalization constant. Here, the notation $|\cdot|_R$ denotes the norm in $\mathbf{R}^d$ associated with the definite positive matrix R^{-1}, i.e. $|u|_R^2 = u^* \, R^{-1} \, u$.

2.2 Finite difference on a regular grid

On a given m–dimensional regular grid Ω^h with mesh $h = (h_1, \cdots, h_m)$, we use the upwind finite difference scheme introduced by Kushner [7], see also Kushner and Dupuis [8], to approximate the partial differential operator L defined in (4). In this scheme, the first order derivatives are approximated as

$$\frac{\partial \phi}{\partial x_i}(x) \simeq \begin{cases} \dfrac{\phi\,(x + e_i h_i) - \phi(x)}{h_i} \; , & \text{if } b_i(x) \geq 0 \\ \dfrac{\phi(x) - \phi\,(x - e_i h_i)}{h_i} \; , & \text{if } b_i(x) < 0 \end{cases}$$

where e_i denotes the unit vector in the i–th coordinate direction. The central second order derivatives are approximated as

$$\frac{\partial^2 \phi}{\partial x_i^2}(x) \simeq \frac{\phi\,(x + e_i\, h_i) - 2\phi(x) + \phi\,(x - e_i\, h_i)}{h_i^2} \; ,$$

and the mixed second order derivatives as

$$\frac{\partial^2 \phi}{\partial x_i \partial x_j}(x) \simeq \begin{cases} \dfrac{1}{2h_i} \left[\dfrac{\phi\,(x + e_i h_i + e_j h_j) - \phi\,(x + e_i h_i)}{h_j} - \dfrac{\phi\,(x + e_j h_j) - \phi(x)}{h_j} \right. \\ \left. + \dfrac{\phi(x) - \phi\,(x - e_j h_j)}{h_j} - \dfrac{\phi\,(x - e_i h_i) - \phi\,(x - e_i h_i - e_j h_j)}{h_j} \right] \; , \\ \qquad\qquad \text{if } a_{i,j}(x) \geq 0 \\ \dfrac{1}{2h_i} \left[\dfrac{\phi\,(x + e_i h_i) - \phi\,(x + e_i h_i - e_j h_j)}{h_j} - \dfrac{\phi(x) - \phi\,(x - e_j h_j)}{h_j} \right. \\ \left. + \dfrac{\phi(x + e_j h_j) - \phi(x)}{h_j} - \dfrac{\phi(x - e_i h_i + e_j h_j) - \phi(x - e_i h_i)}{h_j} \right] \; , \\ \qquad\qquad \text{if } a_{i,j}(x) < 0 \end{cases}$$

As a result, the infinitesimal generator L is approximated as

$$L\,\phi(x) \simeq L^h\,\phi(x) = \sum_{y \in A^h(x)} L^h(x,y)\,\phi(y) \ ,$$

where for all $x \in \Omega^h$, $A^h(x) \subset N^h(x)$ denotes the set of points in the grid Ω^h which are *accessible* from x, i.e.

$$A^h(x) = \{ x + \varepsilon_i e_i h_i + \varepsilon_j e_j h_j \ , \text{ for all } \varepsilon_i, \varepsilon_j \in \{0, \pm 1\} \ , \ i \neq j \ \} \ ,$$

and $N^h(x)$ denotes the set of *nearest* neighbours of x, including x itself, i.e.

$$N^h(x) = \{ x + \varepsilon_1 e_1 h_1 + \cdots + \varepsilon_m e_m h_m \ , \text{ for all } \varepsilon_1, \cdots, \varepsilon_m \in \{0, \pm 1\} \ \} \ .$$

Notice that $\#A^h(x) = 1 + 2m + 4\frac{1}{2}m(m-1) = 1 + 2m^2$, and $\#N^h(x) = 3^m$, respectively.

By identifying the coefficients of the matrix L^h, we have for all $x \in \Omega^h$

$$\begin{aligned}
L^h(x,x) &= -\sum_{i=1}^m \Big[\ \frac{1}{h_i^2} a_{i,i}(x) - \sum_{j:j\neq i} \frac{1}{2h_ih_j} |a_{i,j}(x)| \ \Big] - \sum_{i=1}^m \frac{1}{h_i} |b_i(x)| \ , \\
L^h(x, x \pm e_i h_i) &= \frac{1}{2h_i^2} a_{i,i}(x) - \sum_{j:j\neq i} \frac{1}{2h_ih_j} |a_{i,j}(x)| + \frac{1}{h_i} b_i^{\pm}(x) \ , \\
L^h(x, x + e_i h_i \pm e_j h_j) &= \frac{1}{2h_ih_j} a_{i,j}^{\pm}(x) \ , \\
L^h(x, x - e_i h_i \mp e_j h_j) &= \frac{1}{2h_ih_j} a_{i,j}^{\pm}(x) \ , \\
L^h(x,y) &= 0 \ , \qquad \text{otherwise,}
\end{aligned}$$

for all $i, j = 1, \cdots, m$, $i \neq j$. Let

$$\bar{\Omega}^h = \bigcup_{x \in \Omega^h} N^h(x) \qquad \text{and} \qquad \partial\Omega^h = \bar{\Omega}^h \setminus \Omega^h \ .$$

For a boundary point $x \in \partial\Omega^h$, the definition of the finite difference approximation depends on the boundary condition, see [7]. In any case, one can check that

$$\forall x \in \bar{\Omega}^h \ , \quad \sum_{y \in A^h(x)} L^h(x,y) = 0 \ ,$$

and under the additional assumption

$$\forall x \in \mathbf{R}^m \ , \quad \frac{1}{h_i^2} a_{i,i}(x) - \sum_{j:j\neq i} \frac{1}{2h_ih_j} |a_{i,j}(x)| \geq 0 \ , \qquad i = 1, \cdots, m \tag{7}$$

the matrix L^h satisfies

$$\forall x \in \bar{\Omega}^h \ , \quad L^h(x,x) \leq 0 \quad \text{and} \quad L^h(x,y) \geq 0 \quad \text{for each } y \in A^h(x) \ , \ y \neq x \ .$$

Therefore, under the additional assumption (7), the finite difference matrix L^h obtained by this method can be interpreted as the infinitesimal generator of a pure jump Markov process taking values in the discretization grid Ω^h. In addition, the matrix $[I - \Delta\, L^h]$ is a M–matrix. As a consequence, the resulting approximation to the Fokker–Planck equation will automatically be a discrete probability distribution (i.e. non–negative everywhere, and adding up to one).

2.3 Finite difference on a composite grid

For simplicity, we consider the situation with two meshes : a fine mesh $h = (h_1, \cdots, h_m)$, and a coarse mesh $H = 2h$. We assume that a composite grid $\Omega^{\underline{h}}$ is given as

$$\Omega^{\underline{h}} = \Omega^H \cup \Omega^h_{\rm loc} \qquad \text{where} \qquad \Omega^h_{\rm loc} = \bigcup_{x \in \Omega^H, \text{ refined}} N^h(x) \ .$$

This defines a first partition of the composite grid $\Omega^{\underline{h}}$ into the set $\Omega^h_{\rm loc}$ of fine grid points, and the set $\Omega^H_{\rm loc} = \Omega^{\underline{h}} \setminus \Omega^h_{\rm loc}$ of coarse grid points, see Figure 1.

A fine grid point $x \in \Omega^h_{\rm loc}$ is said *regular* if $A^h(x) \subset \Omega^h_{\rm loc}$. Similarly, a coarse grid point $x \in \Omega^H_{\rm loc}$ is said *regular* if $A^h(x) \cap \Omega^h_{\rm loc} = \emptyset$. This defines a second partition of the composite grid $\Omega^{\underline{h}}$ into the set $\Omega^{\underline{h}}_C$ of regular coarse grid points, the set $\Omega^{\underline{h}}_F$ of regular fine grid points, and the set $\Omega^{\underline{h}}_I = \Omega^{\underline{h}} \setminus \Omega^{\underline{h}}_R$ of *interface* points, where $\Omega^{\underline{h}}_R = \Omega^{\underline{h}}_C \cup \Omega^{\underline{h}}_F$, see Figure 2.

For any point $x \in \Omega^{\underline{h}}$, we need to specify the set $A^{\underline{h}}(x)$ of points in the composite grid $\Omega^{\underline{h}}$ which are *accessible* from x, and to specify the corresponding finite difference coefficients $L^{\underline{h}}(x,y)$, $y \in A^{\underline{h}}(x)$.

At a regular point $x \in \Omega^{\underline{h}}_R$, we use the original finite difference scheme : at a regular coarse grid point $x \in \Omega^{\underline{h}}_C$, we use $A^{\underline{h}}(x) = A^H(x)$ and $L^{\underline{h}}(x,y) = L^H(x,y)$ for all $y \in A^H(x)$, and at a regular fine grid point $x \in \Omega^{\underline{h}}_F$, we use $A^{\underline{h}}(x) = A^h(x)$ and $L^{\underline{h}}(x,y) = L^h(x,y)$ for all $y \in A^h(x)$.

At an interface point $x \in \Omega^{\underline{h}}_I$, a special technique is used : (i) First, as depicted in Figures 3 and 5, we consider the set $A^h(x)$ of points in the grid Ω^h which are *accessible* from x, by adding some slave points (+) if necessary, and we define the corresponding finite difference coefficients $L^h(x,y)$, $y \in A^h(x)$, using the expressions given in Section 2.2 above. (ii) Then, we redistribute the finite difference coefficients of each slave point to points that actually exist in the composite grid, by linear interpolation.

We get in the end a composite scheme :

$$L^{\underline{h}}\phi(x) = \sum_{y \in A^{\underline{h}}(x)} L^{\underline{h}}(x,y)\, \phi(y) \ ,$$

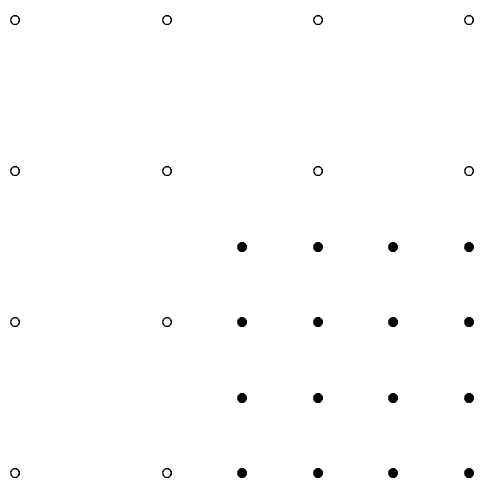

Figure 1: Coarse grid $\Omega^H_{\rm loc}(\circ)$ and fine grid $\Omega^h_{\rm loc}(\bullet)$

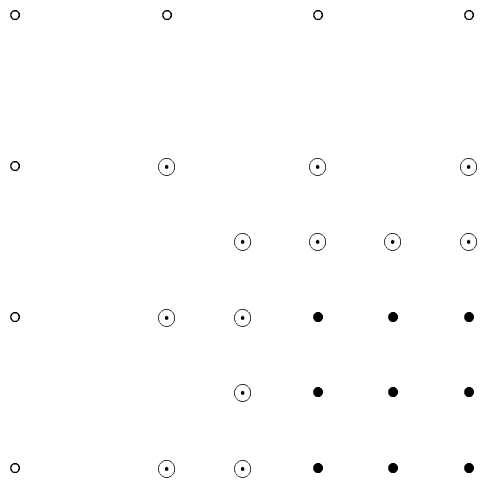

Figure 2: Regular coarse grid $\Omega^h_C(\circ)$, regular fine grid $\Omega^h_F(\bullet)$, and interface $\Omega^h_I(\odot)$

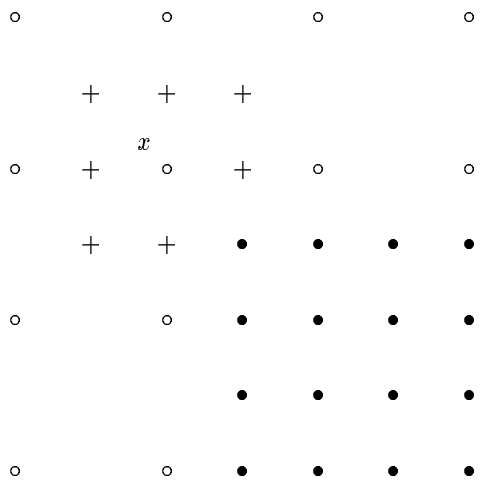

Figure 3: Coarse grid $\Omega^H_{\rm loc}(\circ)$, fine grid $\Omega^h_{\rm loc}(\bullet)$, and slave $(+)$ for interface coarse grid point x

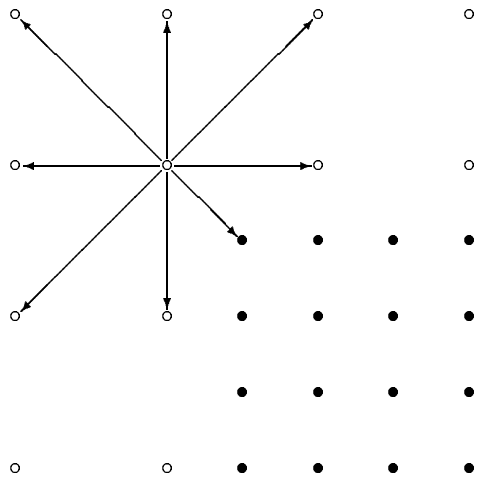

Figure 4: Coarse grid $\Omega^H_{\rm loc}(\circ)$, fine grid $\Omega^h_{\rm loc}(\bullet)$, and pointers $(\longrightarrow)$ to the set $A^{\underline{h}}(x)$

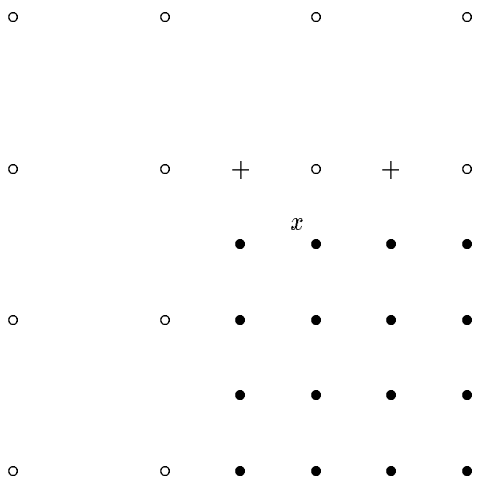

Figure 5: Coarse grid $\Omega^H_{\rm loc}(\circ)$, fine grid $\Omega^h_{\rm loc}(\bullet)$ and slave $(+)$ for interface fine grid point x

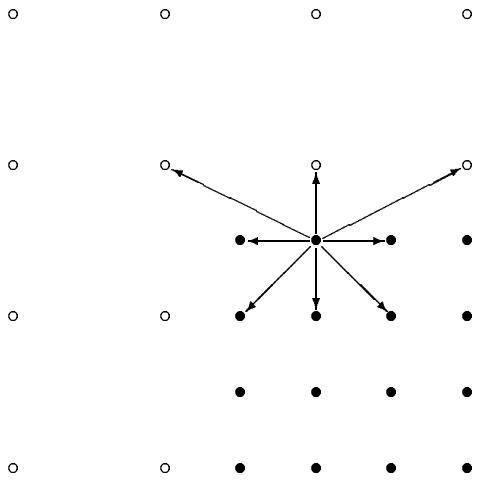

Figure 6: Coarse grid $\Omega^H_{\rm loc}(\circ)$, fine grid $\Omega^h_{\rm loc}(\bullet)$, and pointers $(\longrightarrow)$ to the set $A^{\underline{h}}(x)$

where $A^{\underline{h}}(x)$ is the set of points in the composite grid $\Omega^{\underline{h}}$ which are *accessible* from x, see Figures 4 and 6.

With this technique, we can easily prove that the resulting finite difference matrix $L^{\underline{h}}$ on the composite grid $\Omega^{\underline{h}}$, is such that the matrix $[I - \Delta\, L^{\underline{h}}]$ is a M–matrix. In general however, this scheme is not locally consistent at interface points, but this does not prevent convergence to occur, provided there are not too many interface points, see Theorem 5.3 in [8, Chapter 10].

2.4 Algorithm for the filtering problem

For simplicity, we present our algorithm in the simple case of two meshes : a fine mesh $h = (h_1, \cdots, h_m)$, and a coarse mesh $H = 2h$.

To obtain our approximation $p_n^{\Delta,\underline{h}}$ at time t_n, we proceed in three steps.

□ We first consider the following linear system on the coarse grid Ω^H

$$[I - \Delta\, L^H]^*\; p_{n-1/2}^{\Delta,H} = p_{n-1}^{\Delta,H} \;\; .$$

The right hand side vector $p_{n-1}^{\Delta,H}$ is just the restriction to Ω^H of the fine grid approximation $p_{n-1}^{\Delta,h}$ obtained at the previous time t_{n-1}. To solve this linear system, we use a direct solver.

Based on this rough approximation $p_{n-1/2}^{\Delta,H}$ of the solution $p_{n-1/2}$, we have then to decide where to refine the grid. From a theoretical point of view, the determination of the refinement region is a difficult task, for which there have been several approaches discussed in the literature, see e.g. Babuška and Rheinboldt [1] and Bieterman [2]. Here, we propose a widely used heuristic *a posteriori* criterion, based on the magnitude of the discretization truncation error : at any coarse grid point $x \in \Omega^H$, we compute an approximation of the local truncation error

$$\rho(x) = \Big\{ \sum_{i=1}^{m} \Big| \frac{\partial^2 p_{n-1/2}^{\Delta}}{\partial x_i^2}(x) \Big|^2 \Big\}^{1/2} \;\; ,$$

using central finite differences, i.e.

$$\rho(x) \simeq \Big\{ \sum_{i=1}^{m} \frac{1}{h_i^2} \, |p_{n-1/2}^{\Delta,H}(x + e_i h_i) - 2p_{n-1/2}^{\Delta,H}(x) + p_{n-1/2}^{\Delta,H}(x - e_i h_i)\,|^2 \Big\}^{1/2} \;\; .$$

This is a local computation, which takes little time. Using a simple threshold decision rule, we decide whether this point is going to be refined or not. As a result, we obtain a composite grid

$$\Omega^{\underline{h}} = \Omega^H \cup \Omega^h_{\rm loc} \qquad \text{where} \qquad \Omega^h_{\rm loc} = \bigcup_{x \in \Omega^H,\ \text{refined}} N^h(x) \;\; .$$

Notice that the nested iteration of the multigrid method, offers a numerically cheap way to estimate the local truncation error. In practice, this simple criterion appeared to be quite efficient and reliable.

□ In the second step we construct a new linear system

$$[I - \Delta\, L^{\underline{h}}]^*\; p_{n-1/2}^{\Delta,\underline{h}} = p_{n-1}^{\Delta,\underline{h}}\ , \tag{8}$$

on the composite grid. The matrix $L^{\underline{h}}$ is defined as in Section 2.3 above. The right hand side vector $p_{n-1}^{\Delta,\underline{h}}$ is again the restriction to $\Omega^{\underline{h}}$ of the fine grid approximation $p_{n-1}^{\Delta,h}$ obtained at the previous time t_{n-1}. To solve this system, we use the FAC method.

□ The last step consists of the correction step

$$p_n^{\Delta,h} = c_n\; \Psi_n^{\Delta}\; I_{\underline{h}}^{h}\; p_{n-1/2}^{\Delta,\underline{h}}\ , \tag{9}$$

on the global fine grid Ω^h. Here c_n is a normalization constant, to make sure that the vector $p_n^{\Delta,h}$ is a discrete probability distribution (i.e. non–negative everywhere, and adding up to one).

Notice that what is actually needed in the next interation are the right hand side vectors

$$p_n^{\Delta,H} = I_h^H\ [\, c_n\; \Psi_n^{\Delta}\; I_{\underline{h}}^{h}\; p_{n-1/2}^{\Delta,\underline{h}}\,] = c_n\ [\, I_h^H\; \Psi_n^{\Delta}\; I_{\underline{h}}^{h}\,]\; p_{n-1/2}^{\Delta,\underline{h}}\ ,$$

and

$$p_n^{\Delta,\underline{h}} = I_{\underline{h}}^{\underline{h}}\,[\, c_n\; \Psi_n^{\Delta}\; I_{\underline{h}}^{h}\; p_{n-1/2}^{\Delta,\underline{h}}\,] = c_n\ [\, I_{\underline{h}}^{\underline{h}}\; \Psi_n^{\Delta}\; I_{\underline{h}}^{h}\,]\; p_{n-1/2}^{\Delta,\underline{h}}\ ,$$

which can be computed with only *local* computations, and no global computation on the fine grid Ω^h. Only for visualization purpose does one actually use (9).

The grid transfer operator $I_{\underline{h}}^{h}$, $I_{\underline{h}}^{\underline{h}}$, and I_h^H are defined in an obvious way : notice however that they differ from the operators described in [10, page 88], in the sense that they involve the global fine grid Ω^h.

3 FAC method and numerical results

The FAC method introduced in McCormick [10] is a natural extension of the conventional multigrid methods — see Hackbush [5] — to problems discretized on composite grids.

In the case of a two level composite grid $\Omega^{\underline{h}}$, the FAC algorithm for the solution of equation (8) above, i.e.

$$M^{\underline{h}}\, u^{\underline{h}} = f^{\underline{h}}\ ,$$

with

$$M^{\underline{h}} = [I - \Delta\, L^{\underline{h}}]^* \qquad \text{and} \qquad f^{\underline{h}} = p_{n-1}^{\Delta,\underline{h}} = I_{\underline{h}}^{\underline{h}}\, p_{n-1}^{\Delta,h}\ .$$

is represented by $u^{\underline{h}} \longleftarrow \mathsf{FAC}^{\underline{h}}(u^{\underline{h}}, f^{\underline{h}})$, and is defined in Figure 7.

The grid transfer operators $I_{\underline{h}}^{2h}$, $I_{2h}^{\underline{h}}$, $I_{\underline{h}}^{h,\mathrm{loc}}$ and $I_{\underline{h},\mathrm{loc}}^{\underline{h}}$ are the operators described in [10, page 88]. Notice that the direct solver of the subgrid equation is usually replaced by some iteration method.

begin

f^{2h}	=	$I_{\underline{h}}^{2h}\ (f^{\underline{h}} - M^{\underline{h}}\ u^{\underline{h}})$	residual restriction to the coarse grid Ω^{2h}
u^{2h}	=	$(M^{2h})^{-1}\ f^{2h}$	exact solution of the coarse grid problem
$u^{\underline{h}}$	=	$u^{\underline{h}} + I_{2h}^{\underline{h}}\ u^{2h}$	global correction
$f_{\rm loc}^{h}$	=	$I_{\underline{h}}^{h,{\rm loc}}\ (f^{\underline{h}} - M^{\underline{h}}\ u^{\underline{h}})$	residual restriction to the local fine grid $\Omega_{\rm loc}^{h}$
$u_{\rm loc}^{h}$	=	$(M_{\rm loc}^{h})^{-1}\ f_{\rm loc}^{h}$	exact solution of the local fine grid problem
$u^{\underline{h}}$	=	$u^{\underline{h}} + I_{h,{\rm loc}}^{\underline{h}}\ u_{\rm loc}^{h}$	local correction

end

Figure 7: Two–level FAC algorithm

A full multigrid method is used in our algorithm. As we have already pointed out, this approach gives us not only a good initial value for the FAC iteration, but also a cheap way to find the refined regions. We propose the following algorithm for nonlinear filtering problems, as defined in Figure 8, where $h_l = H/2^{l-1}$ denotes the grid mesh at level l, and $h_1 = H$ and $h_L = h$ denote the coarse grid mesh, and the fine grid mesh respectively.

begin

$l = 1$, $\underline{h}_l = H$

$f^{\underline{h}_l} = I_h^{\underline{h}_l}\ p_{n-1}^{\Delta,h}$

until ($l = L$) {

$u^{\underline{h}_l} \longleftarrow \mathsf{FAC}^{\underline{h}_l}(u^{\underline{h}_l}, f^{\underline{h}_l})$

$l = l + 1$

$\underline{h}_l = (h_l, h_{l-1}, \cdots, h_1 = H)$ find local refined region in $\Omega_{\mathrm{loc}}^{\underline{h}_{l-1}}$

$f^{\underline{h}_l} = I_h^{\underline{h}_l}\ p_{n-1}^{\Delta,h}$

$u^{\underline{h}_l} = I_{\underline{h}_{l-1}}^{\underline{h}_l}\ u^{\underline{h}_{l-1}}$

}

$u^{\underline{h}_L} \longleftarrow \mathsf{FAC}^{\underline{h}_L}(u^{\underline{h}_L}, f^{\underline{h}_L})$ end of prediction step

$p_n^{\Delta,h} = c_n\ \Psi_n^{\Delta}\ I_{\underline{h}_L}^{h}\ u^{\underline{h}_L}$ correction step

end

Figure 8: Multi–level FAC algorithm for the Zakai equation (from t_{n-1} to t_n)

3.1 Data structure

The `FORTRAN 77` programming language is difficult to use for this kind of problem, in particular in terms of memory management. In contrast, the `C` — and recently the `FORTRAN 90` — programming language offers dynamic memory allocation and specific data structure by means of `pointer` and `struct`, see Darnell and Margolis [4], Brainerd, Goldberg and Adams [3]. This can reduce significantly the software complexity. Let us now give some indications on the data structure, which plays an important role in our implementation.

The Figure 9 presents the definition of the structure associated with a point. For any point $x = (x_1, x_2)$ in the composite grid, the eight pointers `SW, S, SE, W, E, NW, N` and

`NE` contain the address of its eight nearest neighbours, each of which is represented itself by same type of structure. If one or several of these neighbours does not exist (e.g. at the boundary), then the corresponding pointers will be set to `NULL`. The other eight pointers `SWS, SES, WSW, ESE, WNW, ENE, NWN` and `NEN` are used only at the interface points, they contain the address of points in the composite grid which are accessible from x but are not nearest neighbours. In the next subsection, we can see by some examples how it works. The double precision variable `u` and `f` represent the solution and right hand side at the grid point respectively. Finally, the remaining double precision variables represent the coefficients of the matrix $M\underline{^h} = [I - \Delta\, L\underline{^h}]^*$.

```
struct POINT {
  double x1, x2;                    /* coordinates of a point */
  POINT         *NWN,    *NEN,      /* pointers to neighbours */
         *WNW, *NW, *N, *NE, *ENE,
               *W,      *E,
         *WSW, *SW, *S, *SE, *ESE,
               *SWS,    *SES;

  double u, f;

  double       nwn,   nen,          /* matrix coefficients */
         wnw, nw, n, ne, ene,
              w,  c, e,
         wsw, sw, s, se, ese,
              sws,   ses;

  POINT  *next;
  .....
}
```

Figure 9: Data structure for a composite grid point

We store the different types of points $\Omega\underline{^h_C}(\circ)$, $\Omega\underline{^h_I}(\odot)$ and $\Omega\underline{^h_F}(\bullet)$ into three *lists*. The pointer `next` is so reserved to the construction of these lists. This is used mainly as a counter for iteration.

There is another programming language `C++`, an extension of `C`, which provides an efficient implementation of object oriented programming techniques, see Stroustrup [13]. An important notion of this language is `derive class` and `virtual function`. One can for instance define the class of regular points as `Basic Class`, so that the different types of interface points will be defined as derived classes of `Basic Class`. With the notion of `virtual function`, various functions such as relaxation can be programmed seperatly according to

the class. These functions will share the same name, and in the execution of the program, each class will use its own function automatically.

3.2 Numerical results

We consider as an example the case of a two–dimensional state equation, with coefficients

$$b(x) = \begin{pmatrix} -x_2 \\ x_1 \end{pmatrix} \qquad \text{and} \qquad \sigma(x) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} ,$$

where $x = (x_1, x_2)$. This is a random linear oscillator, or in phase space, a randomly perturbed circular motion around the origin.

We consider the following different cases of observation functions, in dimension either $d = 1$ or $d = 2$:

Case A $d = 2$ — direct observations, i.e. $g(x) = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$.

Case B $d = 1$ — distance observations only, i.e. $g(x) = \sqrt{x_1^2 + x_2^2}$.

Case C $d = 1$ — angle observations only, i.e. $g(x) = \arctan\left(\dfrac{x_2}{x_1}\right)$.

Case D $d = 2$ — with $g(x) = \begin{pmatrix} |x_1| \\ |x_2| \end{pmatrix}$.

In all four cases, the covariance matrix R of the observation noise is small. As a result, it is clear that at each time t_n the conditional probability distribution p_n^Δ is *practically negligible* outside a small neighbourhood of the subset $g^{-1}(z_n^\Delta) = \{x \in \mathbf{R}^2 \, : \, g(x) = z_n^\Delta\}$. Whether this conditional probability distribution is actually concentrated on a smaller region of the state space, would depend on the way the observations and the prior information (carried by the state equation) complement each other.

For instance in Case B, a zero mean Gaussian probability distribution is used as the initial condition of the Zakai equation. In this situation, the state equation does not help : at each time t_n, the conditional probability distribution is concentrated around the annulus $g^{-1}(z_n^\Delta)$, see Figures 12 and 13.

In Case D, a similar initial condition is used for the Zakai equation. In this situation, the state equation partly helps : at each time t_n, the conditional probability distribution is concentrated around two out of the four points in $g^{-1}(z_n^\Delta)$, see Figures 16 and 17.

The results are given in Figures 10 to 17. Five grid levels are used : the coarse grid is a uniform 5×5 grid on the square $[-6.0\,,\,6.0] \times [-6.0\,,\,6.0]$, and four levels of refinement have been used when necessary. We represent any point x in the composite grid $\Omega^{\underline{h}}$ by its position $(\cdot)$ and pointers $(\longrightarrow)$ to its accessible neighbours. The solution is computed on the composite grid using the FAC method, and then *for visualization purpose only* is

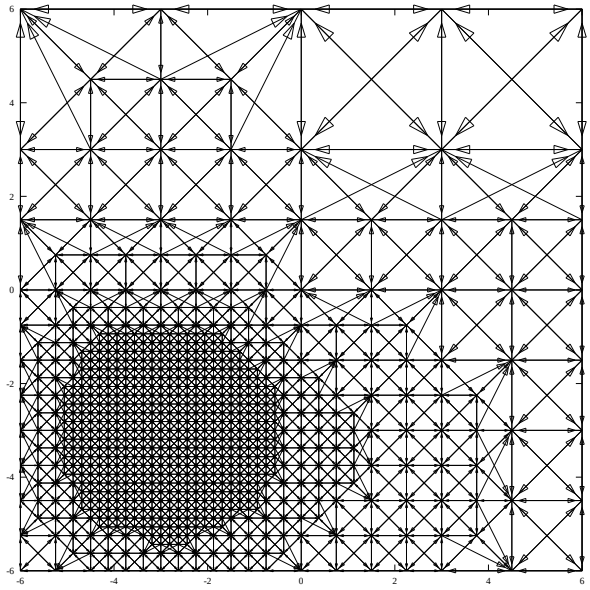

Figure 10: Case A — composite grid

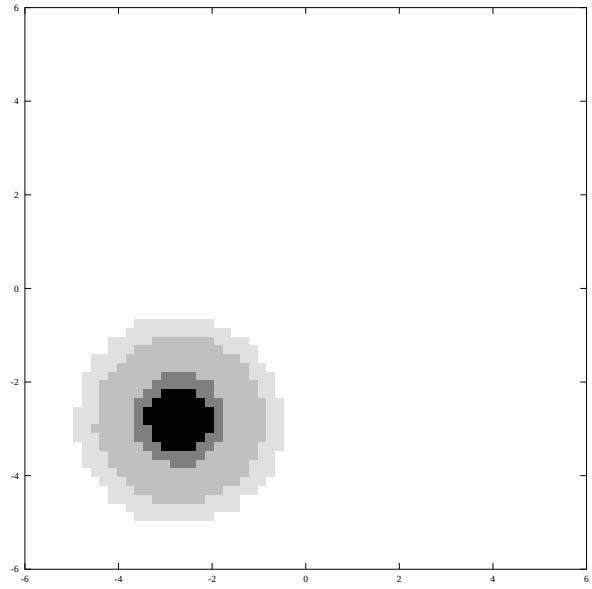

Figure 11: Case A — conditional density (confidence regions)

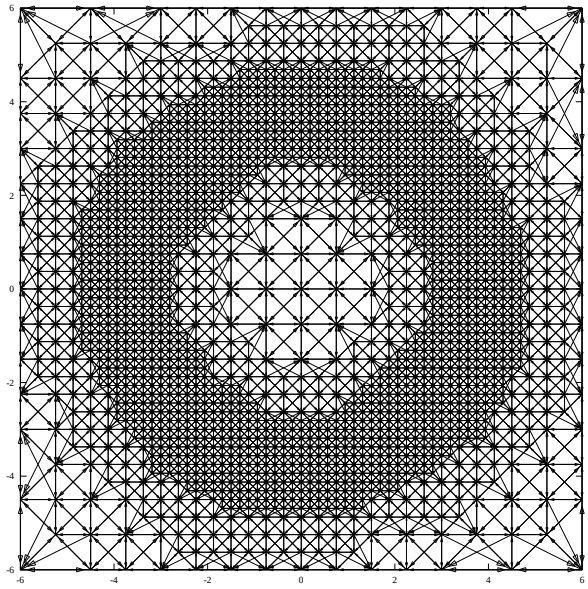

Figure 12: Case B — composite grid

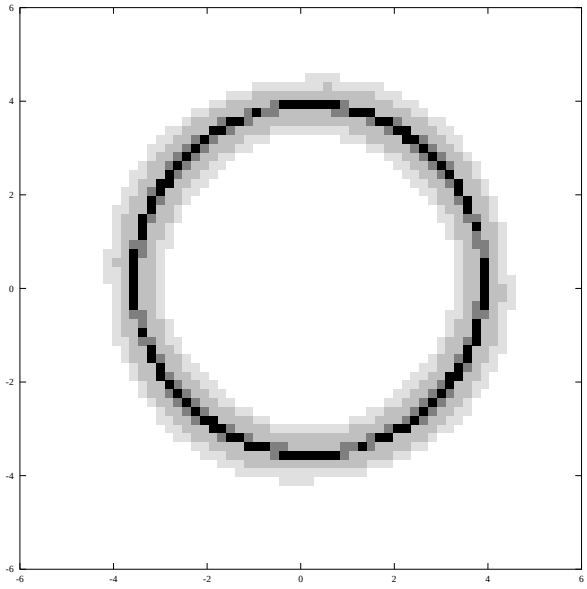

Figure 13: Case B — conditional density (confidence regions)

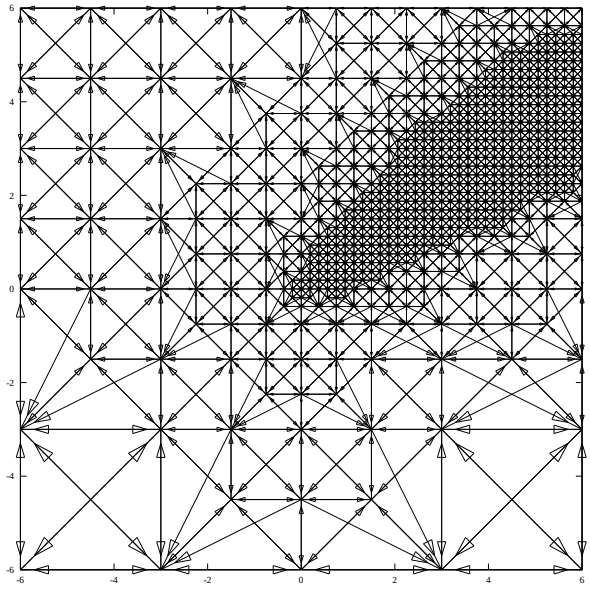

Figure 14: Case C — composite grid

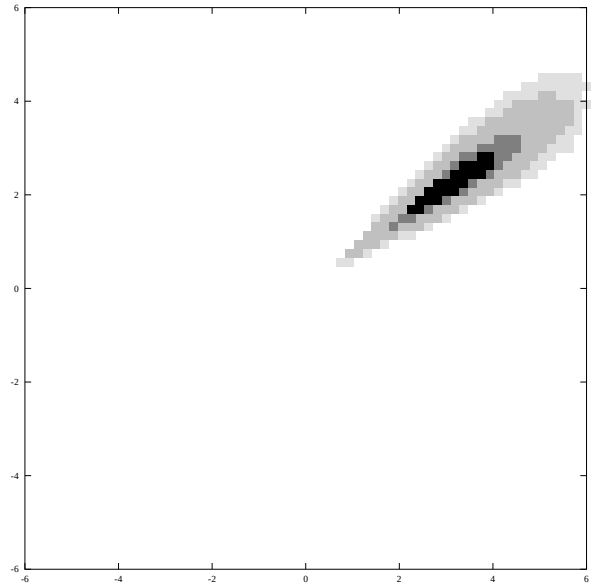

Figure 15: Case C — conditional density (confidence regions)

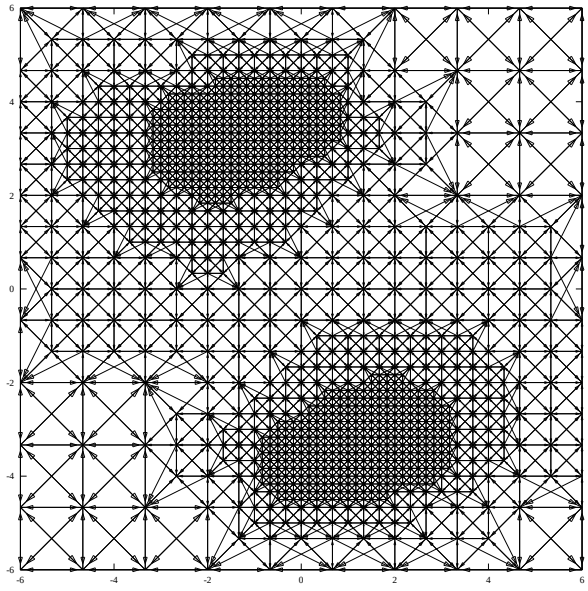

Figure 16: Case D — composite grid

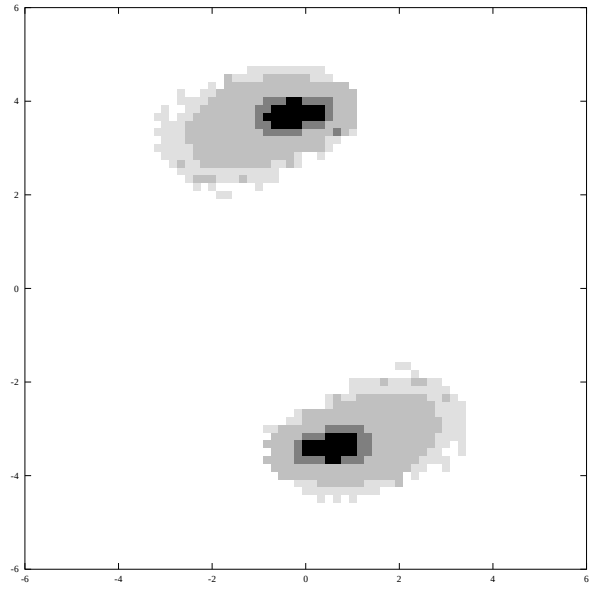

Figure 17: Case D — conditional density (confidence regions)

interpolated on a global uniform fine grid. Confidence regions are represented for the density, at four different levels : 0.5, 0.75, 0.99 and 0.9999 respectively.

As we can see, the composite grids show a good agreement with the computed density. In particular, Figure 16 shows that our method can easily handle the case where the refinement region has several disjoint connected components.

3.3 Performance evaluation

To illustrate the performance of our algorithm, we present below some results for the solution of the time discretized Fokker–Planck equation (5), i.e.

$$[I - \Delta\, L]^*\; p^{\Delta}_{n-1/2} = p^{\Delta}_{n-1}\ ,$$

on a time interval of length Δ.

In Table 1, two methods are compared : the classical full multigrid method (FMG) in which the refinement at each level is global, and the fast adaptive composite grid method (FAC) in which the refinement at each step is local. These two methods are applied with $L = 1, 2, \cdots, 6$ levels respectively and a $V(2,1)$ multigrid cycle is used. For each of the two methods, the table gives the ratio of the CPU time over the CPU time used by coarse grid direct solver. The table gives also the number of points in the global fine grid Ω^h for the FMG method, and the number of points in the composite grid $\Omega^{\underline{h}}$ for the FAC method. These numbers represent the computational memory used by each of the two methods. The gain displayed in the last column is the ratio of the CPU time for the FMG method over the CPU time for the FAC method. This ratio varies according to number of levels and the number of point in the coarse grid. For example, this gain is only 1.6 when we use 3 levels with a 5×5 coarse grid, and it is 67.12 when we use 6 levels with a 10×10 coarse grid. Obviously, the efficiency of the FAC method becomes significant only when the number of levels is large enough.

levels	FMG		FAC			FMG		FAC		
L	Ω^h	CPU	$\Omega^{\underline{h}}$	CPU	gain	Ω^h	CPU	$\Omega^{\underline{h}}$	CPU	gain
1	5×5	1.0				10×10	1.0			
2	9×9	4.2	72	3.1	1.35	19×19	5.3	281	4.0	1.33
3	17×17	10.0	146	6.0	1.60	37×37	23.0	571	11.0	2.09
4	33×33	38.0	251	12.0	3.16	73×73	162.0	1087	21.0	7.71
5	65×65	220.0	408	26.0	8.46	145×145	1472.01	2005	68.0	21.65
6	129×129	2012.0	682	56.0	35.93	289×289	11776.1	4932	175.5	67.12

Table 1: CPU time comparison between FMG and FAC for the Fokker–Planck equation

In Table 2, the Euclidean norms of the composite grid residuals are displayed, using various number of levels, and various coarse grids Ω^H. Also included are the convergence factors geometrically averaged over five cycles.

	L = 5				L = 6			
	$\Omega^H = 5 \times 5$		$\Omega^H = 10 \times 10$		$\Omega^H = 5 \times 5$		$\Omega^H = 10 \times 10$	
FAC cycle	residual	factor	residual	factor	residual	factor	residual	factor
cycle 1	1.835E-2		2.250E-2		1.715E-2		1.904E-2	
cycle 2	1.400E-3	0.076	2.714E-3	0.121	2.589E-3	0.151	4.116E-3	0.216
cycle 3	1.029E-4	0.073	3.336E-4	0.122	3.974E-4	0.153	9.272E-4	0.225
cycle 4	6.939E-6	0.067	3.933E-5	0.117	6.010E-5	0.151	2.091E-4	0.225
cycle 5	4.209E-7	0.060	4.360E-6	0.111	8.790E-6	0.146	4.631E-5	0.221
average factor		0.069		0.118		0.150		0.222

Table 2: Convergence history of FAC for the Fokker–Planck equation

4 Conclusion

The FAC method has shown its efficiency for solving filtering problem, in providing accurate solution while saving computational time and memory. Future work will be to increase its range of application. It will be quite easy to extend our results to higher dimensions. Another application is to solve the Zakai equation in the whole state space, in which case the discretization space should be determined by a posteriori error estimates. There are also several questions which have not been addressed here : parallelization and reliable meaningful criterion for local grid refinement. Our last remark is that data strutures have been very useful to handle the case of refinement regions with general arbitrary shape and/or made of disjoint connected components.

Acknowledgement

The last two authors gratefully acknowledge Etienne Pardoux for suggesting the problem, and Harold Kushner for stimulating discussions about earlier versions of this paper.

References

[1] I. BABUŠKA and W.C. RHEINBOLDT. A posteriori error analysis of finite element solutions of one–dimensional problems. *SIAM Journal on Numerical Analysis*, 18:565–589, 1981.

[2] M.B. BIETERMAN. A posteriori error estimation and adaptive finite element grids for parabolic equations. In I. Babuška, J. Chandra, and J.E. Flaherty, editors, *Adaptive Computational Methods for Partial Differential Equations, College Park–1983*, pages 123–143, SIAM, Philadelphia, 1983.

[3] S.W. BRAINERD, C.H. GOLDBERG, and J.C. ADAMS. *Programmer's Guide to* `FORTRAN 90`. McGraw–Hill, New York, 1990.

[4] P.A. DARNELL and P.E. MARGOLIS. *Software Engineering in* `C`. *Springer Books on Professional Computing*, Springer Verlag, Berlin, 1988.

[5] W. HACKBUSCH. *Multi–Grid Methods and Applications*. Volume 4 of *Springer Series in Computational Mathematics*, Springer Verlag, Berlin, 1985.

[6] A.H. JAZWINSKI. *Stochastic Processes and Filtering Theory*. Volume 64 of *Mathematics in Science and Engineering*, Academic Press, New York, 1970.

[7] H.J. KUSHNER. *Probability Methods for Approximations in Stochastic Control and for Elliptic Equations*. Volume 129 of *Mathematics in Science and Engineering*, Academic Press, New York, 1977.

[8] H.J. KUSHNER and P. DUPUIS. *Numerical Methods for Stochastic Control Problems in Continuous Time*. Volume 24 of *Applications of Mathemathics*, Springer Verlag, New York, 1992.

[9] F. LE GLAND. Time discretization of nonlinear filtering equations. In *Proceedings of the 28th IEEE Conference on Decision and Control, Tampa 1989*, pages 2601–2606, IEEE–CSS, December 1989.

[10] S.F. McCORMICK. *Multilevel Adaptive Methods for Partial Differential Equations*. Volume 6 of *Frontiers in Applied Mathematics*, SIAM, Philadelphia, 1989.

[11] E. PARDOUX. Filtrage non linéaire et équations aux dérivées partielles stochastiques associées. In P.L. Hennequin, editor, *Ecole d'Eté de Probabilités de Saint–Flour XIX—1989*, pages 67–163, Springer Verlag, Berlin, 1991.

[12] B.L. ROZOVSKII. *Stochastic Evolution Systems*. Volume 35 of *Mathematics and its Applications (Soviet Series)*, Kluwer Academic Publishers, Dordrecht, 1990.

[13] B. STROUSTRUP. *The* `C++` *Programming Language*. Addison Wesley, Reading, MA, second edition, 1991.

[14] M. ZAKAI. On the optimal filtering of diffusion processes. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete*, 11(3):230–243, 1969.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399