TR-86-26

SYSTEMATIC NUMERICAL EXPERIMENTS IN NONLINEAR FILTERING WITH AUTOMATIC FORTRAN CODE GENERATION

by

Francois LeGland

Sophia-Antipolis

Antoine Gondel

Submitted to the 25th. CDC, Athens, Dec.10-12, 1986

SYSTEMATIC NUMERICAL EXPERIMENTS IN NONLINEAR FILTERING WITH AUTOMATIC FORTRAN CODE GENERATION

François LeGland *
Systems Research Center
University of Maryland
COLLEGE PARK, MD 20742
and
INRIA Sophia-Antipolis
Route des Lucioles
F-06560 VALBONNE

Antoine Gondel
ENST, 46 rue Barrault
F-75634 PARIS CEDEX 13

Abstract

This paper describes a system, still under development, whose purpose is to make numerical experiments in nonlinear filtering easy. Given a description, in terms of a list of keywords and pieces of data, of both the nonlinear filtering problem and the algorithm to solve it, the system, which is written itself in Macsyma, produces a Fortran program. Emphasis has been put on graphic output, using GKS. There is also an interactive mode, in which the system helps the user building this list of keywords and pieces of data.

* Currently visiting Systems Research Center

1. Motivation

Nonlinear filtering is an estimation problem whose exact solution involves, in general, a stochastic partial differential equation (Zakaï equation). Therefore it has the same limitation as any partial differential equation concerning the space variable dimension. A recent attempt to compute the exact optimal filter with no reference to any partial differential equation, thus circumventing the abovementioned limitation, has proved rather unsuccessful (F.LeGland [4]). On the other hand, a large number of so-called "approximate" nonlinear filters have been designed. The point is that, apart from a very few works (e.g. J.Picard [7],[9]), the approximate nature of those filters is not proved, while they are commonly used in practice.

As a consequence, one would like to be able to test – for any kind of model, by means of numerical simulations computing any kind of conditional statistics – any "approximate" nonlinear filter, old, new or to-be-found, against the exact filter given by Zakaï equation or an approximation to it. This means that Zakaï equation is to be our reference method and that we must be able to rely on an efficient and accurate algorithm to solve it.

Therefore, it has been considered worth building a system with the following two main purposes:

1) help designing efficient and accurate algorithms for the solution of Zakaï equation, with extensive numerical experiments
2) help testing any other filter and compare it to the reference filter

One feature is that, since numerical experiments are involved, our system should generate Fortran code. In particular this would make possible the use of general mathematical libraries (Nag, Linpack,...) wherever they are available. On the other hand, Macsyma has already proved able to take care of the generation of Fortran code (see C.Gomez et al. [1]), and is certainly exactly suited to do the few symbolic computations involved, so that our system is actually written in Macsyma, with parts in Lisp as well.

The rest of the paper is organized as follows. Coming next is section 2 which shows how the system works. Section 3 contains a short presentation of nonlinear filtering. In section 4 we discuss some possible algorithms for the solution of Zakaï equation we would like to make our system test, and the kind of information to feed it with. Section 5 discusses implementation issues and future development.

2. How the system works

Basically, our system can be ran in two different modes, either interactive or batch. In batch mode, it is feeded with a list of keywords and pieces of data describing our problem (those are presented in section 4, and from now on will appear in boldface characters). In interactive mode, the system actually builds itself this list of keywords and pieces of data by asking the user appropriate questions. At this point, it should be stressed that in both modes, the user is supposed to have a minimum knowledge of nonlinear filtering theory (for a short presentation, see next section).

The next step is the system to analyze this list and accordingly to build a new big list whose innermost elements are instructions in an intermediate language, known as macrofort for macro-Fortran and originally written in Macsyma by J.P.Quadrat (INRIA-Rocquencourt). When the process of building this second list is completed, it is finally translated into Fortran 77 program.

One additional feature is that whenever a formal constant (actually an atom) is found in any of the symbolic expressions that describe our model and are given in the original list, it will automatically be considered by the system as a parameter. According to a given switch, the numerical value of this parameter will be either prompted for interactively, or read in a data file, at Fortran runtime.

Moreover, there is some kind of bookkeeping in the sense that some informations concerning the Fortran program actually generated will be made available in symbolic form, in a Macsyma batchable file. The main interest of this is to make easy the modification of the data file.

Altogether, the system produces as its output three different files:

- a Fortran source file (the program itself)
- a Fortran data file
- a Macsyma file, for bookkeeping.

3. A short presentation of nonlinear filtering

We will discuss here the problem of estimating the state of a diffusion process when only a white-noise perturbed observation of it is available. This is the situation our system is able to deal with now. Other situations will perhaps be included progressively.

3.1 The model

Let us consider two stochastic processes, one known as the signal and the other as the observation:

$$dX_t = b(X_t)\,dt + \sigma(X_t)\,dW_t$$

$$dY_t = h(X_t)\,dt + d\bar{W}_t$$

Here $(W_t;\ t \geq 0)$ and $(\bar{W}_t;\ t \geq 0)$ are independent Wiener processes, with covariance I and Q respectively (for the sake of simplicity we will assume $Q = I$ from now on). The random variable X_0 is independent of $(\bar{W}_t;\ t \geq 0)$ and its law is absolutely continuous with respect to Lebesgue measure: let $p_0(.)$ denote its density.
The pieces of data describing our model are therefore:

$p_0(.)$	**initial_density**
$b(.)$	**drift**
$a(.) = \sigma\sigma^*(.)$	**diffusion**
$h(.)$	**sensor**

Another quantity of interest for the rest of this section is the infinitesimal generator of the signal process:

$$L = \frac{1}{2} a_{ij}(.) \frac{\partial^2}{\partial x_i \partial x_j} + b_i(.) \frac{\partial}{\partial x_i}$$

Remark: As far as the solution of Zakaï equation (see below) is concerned, the dimension of the signal has to be small. At the present time, our system is only able to deal with dimension 1. In the future, we plan to allow dimension 2 and perhaps 3 as well. On the other hand, as far as testing "approximate" nonlinear filters is concerned, there is not such kind of limitation, although no comparison will be possible with our reference method in case the dimension is large.

3.2 The problem and its solution

What we are interested in is to compute such quantities as: $E(f(X_t)|\mathcal{Y}_t)$ for any bounded measurable function $f(.)$, where $\mathcal{Y}_t = \sigma(Y_s;\ 0 \leq s \leq t)$, i.e. to compute the conditional density of the signal X_t given $\mathcal{Y}_t$.

In the rest of this section, we will just state Zakaï equation, which actually gives the solution to the problem under consideration. Define:

$$Z_t = \exp\left(\int_0^t h(X_s)\,dY_s - \frac{1}{2}\int_0^t h^2(X_s)\,ds\right)$$

Provided for instance that $h(.)$ is bounded, the process $(Z_t^{-1};\ t \geq 0)$ is a $(\mathcal{G}_t;\ t \geq 0)$ martingale under P, where $\mathcal{G}_t = \sigma(X_s, Y_s;\ 0 \leq s \leq t)$. One can therefore define a

new probability measure $\overset{o}{P}$, absolutely continuous with respect to the original probability measure P and such that:

$$\frac{d\overset{o}{P}}{dP}\Big|_{\mathcal{G}_t} = Z_t^{-1}$$

The conditional expectations given $\mathcal{Y}_t$, under P and $\overset{o}{P}$ are related by the Kallianpur-Striebel formula:

$$E(f(X_t)|\mathcal{Y}_t) = \frac{\overset{o}{E}\left(f(X_t)Z_t|\mathcal{Y}_t\right)}{\overset{o}{E}\left(Z_t|\mathcal{Y}_t\right)}$$

The following equation, known as Zakaï equation:

$$\begin{cases} dp_t = L^\star p_t\, dt + h p_t\, dY_t \\ p_0 = p_0(.) \end{cases}$$

where $L^\star$ is the adjoint operator of the infinitesimal generator of the signal process, has a unique solution (E.Pardoux [5]). This solution satisfies:

$$(p_t, f) = \overset{o}{E}\left(f(X_t)Z_t|\mathcal{Y}_t\right)$$

for every bounded measurable $f(.)$, which means that p_t is the (unnormalized) conditional density of the signal X_t given $\mathcal{Y}_t$.

In the next section, we are going to discuss some possible algorithms to solve this equation. In particular, we will describe the keywords and pieces of data related to them. Since the system is still under development, it might very well happen that some of those keywords change in the next future.

4. Keywords

In section 3, we have already found four pieces of data which describe the model: **initial_density, drift, diffusion, sensor.** In the list our system is to be fed with, each of those keywords must appear, followed by its symbolic expression. As was said above, whenever a formal constant is found in such a symbolic expression, it is considered as a parameter, a model-parameter actually. There are other parameters, which are related to algorithms (e.g. time step, grid size, ...) We will not list them here since the user has not to take care of them. There is just a special keyword that should appear into the list, either **prompted_for** or **read_on_file**, specifying how the Fortran program will manage to assign a numerical value to each of the parameters.

Observation can either come from a **record** i.e. actual measurements previously stored on a file, or be generated by **simulation.** In the latter case, one must choose an appropriate discretization scheme. At this point there is not much flexibility in our system, since only Milshtein's scheme is used. If needed, other schemes could be included (see E.Pardoux-D.Talay [6]). What quantities are to be found on the file or to be simulated, depend on the sampling procedure (see below).

Another point of interest is the purpose of the specific program the system is going to generate. It can be either to:

- just compute the conditional density (the default)
- test an algorithm on the basis of a single observation (**single_trajectory_test**). For instance one can plot the density and look at its deformation as time runs.
- compute some conditional statistics for a whole set of observations and then perform some Monte-Carlo analysis (**monte_carlo_analysis**). The symbolic expression of those conditional statistics is another piece of data to be found in the input list just after the keyword **statistics.** Most likely in this case are the observations to be generated by simulation, so the corresponding keyword is expected. One can plot the rms error vs. the number of Monte-Carlo runs, i.e. the number of observations simulated. Another option is **sensitivity_analysis:** here one lets a parameter vary within the Monte-Carlo analysis and plot the rms error vs. this particular parameter. What is needed is just to introduce a loop at the appropriate place in the Fortran program being generated. For this particular parameter, one needs a whole range of numerical values. Moreover it can be either a model-parameter (the most difficult case since every step is concerned, including simulation), an algorithm-parameter or a parameter of the conditional statistics itself (the easiest case).

In the next future, it will be possible to compare some (let us say 2) algorithms for either **single_trajectory_test** or **monte_carlo_analysis** (with or without additional **sensitivity_analysis**), with plots to help comparison.

Let us now describe some of the various algorithms we have in mind for the approximation of Zakaï equation.

4.1 Time discretization

We follow the approach of H.Korezlioglu-G.Mazziotto [2], since it provides easy probabilistic interpretation for the numerical schemes involved. Three steps are considered: sampling of the observation, discretization and approximation of the signal.

a. Sampling

This is the process by which the information contained in the whole trajectory $(Y_s;\ 0 \leq s \leq t)$ is replaced by some simpler information provided by a finite collection of random variables. The sampling time step, denoted by k, is a parameter associated to sampling. There are different sampling strategies depending on the finite collection of random variables considered. If one uses only the increments of the observation process on time intervals of length k, one gets **simple_sampling** (the default). If one adds quantities like $\frac{1}{k}\int_{kn}^{k(n+1)} s\, dY_s$ and/or $\frac{1}{k}\int_{kn}^{k(n+1)} Y_s\, ds$ one gets **second_order_sampling**. Higher order sampling strategies can also be defined, but will not be considered here.

b. Discretization

Here one replaces the original signal process by a piecewise constant process. In the next future, it will be possible to consider discretization intervals smaller than sampling intervals: this will be **refined_discretization**, the default being **simple_discretization**. In the former case, the ratio of those interval lengths will be a parameter asssociated to discretization. The system will be able to decide whether to use a refined discretization or not. This is one of the few points where a kind of expertise could be found in the system.

On a discretization interval, the signal can be replaced by its value at either the **left_point** or the **right_point** of this interval (the latter choice has been considered in J.Picard [8]). This gives the two possible schemes:

$$p^k_{n+1} = P^\star_k\ \Psi^k_n\ p^k_n$$

$$p^k_{n+1} = \Psi^k_n\ P^\star_k\ p^k_n$$

where $(P^\star_t;\ t \geq 0)$ is the semigroup with infinitesimal generator $L^\star$ and:

$$\Psi^k_n = \exp(h(.)\Delta Y^k_n - \frac{k}{2}h^2(.))\ ,\ \Delta Y^k_n = Y_{k(n+1)} - Y_{kn}$$

c. Approximation

Here one replaces the signal by an approximating Markov chain. This can be done by discretizing the signal equation, as in [2]. We follow here a different approach, which is to approximate the underlying semigroup. The only scheme available now in the system is **euler_implicit**, which corresponds to approximating the semigroup by its resolvent, but we plan to introduce higher order schemes as well.

What is possible now is to refine the time step for approximation: this will give **refined_approximation** as opposed to **simple_approximation** (the default). Here again the system is able to make the choice of a refined approximation. One possible scheme would now be:

$$(I - kL)^\star\ p^k_{n+1} = \Psi^k_n\ p^k_n$$

A detailed presentation of these refined and higher order time-discretization schemes will appear elsewhere.

4.2 Space discretization

At the present time, this point is less developed than time discretization, but will be paid much attention in the next future. As a matter of fact, the system is only able to deal with dimension 1. The first step here is to restrict the whole state space to a fixed bounded subregion (unless the state space was already bounded itself). In the one-dimensional case, this is just an interval, whose characteristics (location and width) are just considered as parameters. In higher dimension, one will have to describe more explicitly the shape of this subregion. The boundary conditions can be of either Dirichlet type (**stopped**) or Neumann type (**reflected**), where both keywords refer to the behaviour of the underlying signal process with respect to the boundary.

Then one has to design a grid, whose mesh size denoted by ε is again a parameter. Following H.J.Kushner [3], we approximate on this grid the infinitesimal generator L of the original process by a finite difference operator L_ε that can be interpreted as the infinitesimal generator of an approximating Markov process with countable state space. One possible scheme would now be:

$$(I - kL_\varepsilon)^\star \; p_{n+1}^{k,\varepsilon} = \Psi_n^k \; p_n^{k,\varepsilon}$$

What we plan to introduce in the next future concerning space-discretization includes adaptive selection of the grid and the bounded subregion, and multigrid methods to solve the linear problems involved.

5. Implementation issues and future development

A preliminary version of the system has been written by the second author during a 2-months stay at INRIA-Sophia-Antipolis. This version runs under Multics operating system, i.e. it uses the MacLisp version of Macsyma. The Fortran program generated also uses some routines of the Nag library. Another system-dependent point is that it uses the Multics random number generator for the simulations. It would not be difficult to make our system generate Fortran program that would take advantage of the software currently available on any site. However the software we plan to consider are: Nag or Linpack for some linear algebra routines, and GKS or possibly Plot10 for graphic output.

Next we will have to consider what was called the second main purpose of this system, i.e. to introduce "approximate" nonlinear filters and make it easy to compare them to the reference filter, whatever approximation scheme we choose for it.

In the long run, it would be possible also to include some parameter estimation problems with partial information, since these problems are closely related to nonlinear filtering. In particular, the likelihood ratio would be nothing but the denominator of the Kallianpur-Striebel formula introduced in section 3.2.

References

[1] C.Gomez, J.P.Quadrat, A.Sulem, G.L.Blankenship, P.Kumar, A.LaVigna, D.C.McEnany, K.Paul and I.Yan: An expert system for control and signal processing with automatic Fortran code generation, *Proceedings 23rd. CDC* (Las Vegas 1984), 716–723

[2] H.Korezlioglu and G.Mazziotto: Approximations of the nonlinear filter by periodic sampling and quantization, *Analysis and Optimization of Systems* (Nice 1984), 553–567, (l.n.Control Info.Sci. #62), Springer-Verlag, 1984

[3] H.J.Kushner: Probability methods for approximations in stochastic control and for elliptic equations, Academic Press, 1977

[4] F.LeGland: Monte-Carlo methods in nonlinear filtering and importance sampling, *Proceedings 23rd. CDC* (Las Vegas 1984), 31–32

[5] E.Pardoux: Stochastic partial differential equations and filtering of diffusion processes, *Stochastics* **3**,2,127–167 (1979)

[6] E.Pardoux and D.Talay: Discretization and simulation of stochastic differential equations, *Acta Applicandae Mathematicae* **3**,1,23–47 (1985)

[7] J.Picard: Nonlinear filtering of one-dimensional diffusions in the case of a high signal-to-noise ratio, *to appear in SIAM J.Appl.Math.*

[8] J.Picard: An estimate of the error in time discretization of nonlinear filtering problems, *Proceedings 7th. MTNS Symposium* (Stockholm 1985)

[9] J.Picard: Filtrage de diffusions vectorielles faiblement bruitees, *Analysis and Optimization of Systems* (Antibes 1986)