Real time architectures for the Zakai equation and applications

John S. Baras
Electrical Engineering Department
and Systems Research Center
University of Maryland at College Park

Abstract

We examine in detail real-time architectures for the sequential detection and/or estimation problems for diffusion type signals. We demonstrate the fundamental role played by the Zakai equation in defining candidate architectures. For scalar and two dimensional state models an architecture based on systolic arrays is derived. For higher dimensional problems a multilayer architecture based on an asychronous parallel implementation of the Multigrid algorithm is derived. Properties of the architectures and practical hardware implementation results are also reported.

1. Introduction

One of the basic activities of electrical engineering today is the processing of signals, be they in the nature of speech, radar, images, or of electromechanical or biological origin. By "processing" we generally mean conversion of the signals into some more acceptable format for analysis. Examples could be the reduction of noise content, parameter estimation, bandpass filtering, or the enhancement of contrast, as required for imaging systems. In feedback control systems analysis and synthesis, signal processing problems such as sequential detection and estimation are fundamental. The signal theorist develops algorithms for performing these functions by constructing mathematical models of signals and the operations conducted on them. The result of intensive research in this area over the last

ISBN 0-12-481005-5

twenty years has been a rather sophisticated theory, which utilizes advanced concepts from stochastic processes, differential equations and algebraic system theory. For a survey of such work the reader is referred to [2], where it can be seen that researchers have gone far beyond the classic work of Doob [3] and Wong [4].

One of the main reasons for the lack of impact of the more theoretical work on the engineering applications fields has been the failure to meet economic as well as real-time processing constraints imposed by the problems engineers face. The fundamental issue to be resolved for at least a wide class of signal processing problems involves meeting the time constraints implicit in the design. The problem is that such techniques will have much greater demands for their numerical analysis. As this translates to mean a greater number of arithmetic operations per second, meeting real-time processing conditions will be all the more difficult.

This is indeed the trend throughout much of signal processing: a greater volume of signals must be processed in a lesser amount of time, in addition to requiring more sophisticated analysis and relatively inexpensive electronics packaged on a small scale. To better understand these issues, we should examine in more detail the nature of some of these advanced techniques of signal processing. This type of analysis will become critical in the future as designers begin to assess model accuracy on device performance, cost of production and other factors.

A typical problem considered in this paper is described below. There are two hypotheses H_1, H_2 each representing that the observed data $y(t)$ originate from two different models. The decision maker receives the data $y(t)$ and has to decide which of the two hypotheses is valid. This problem is generic to a plethora of digital and analog signal processing problems, such as: pulse amplitude modulation, delta modulation, adaptive delta modulation, speech processing, direction finding receivers, digital phase lock loops, adaptive sonar and radar arrays, simultaneous detection and estimation.

As a matter of fact almost any sequential detection problem can be formulated in a similar manner. The underlying mathematical models can be diverse: diffusion processes, point processes, mixed processes, Markov chains etc. In this paper we shall concentrate on diffusion process models. That is to say, under each hypothesis the

model for the observed data is

$$\begin{aligned} dx^i(t) &= f^i(x^i(t))dt + g^i(x^i(t))dw^i(t) \\ dy(t) &= h^i(x^i(t))dt + dv(t) \end{aligned} \tag{1.1}$$

where $i = 1, 2$ correspond to hypotheses H_1 or H_2. If we let

$$\hat{h}_i(t) = E_i \left\{ h^i(x^i(t)) | \mathcal{F}_t^y \right\} \tag{1.2}$$

the likelihood ratio for the problem is

$$\begin{aligned} \Lambda_t = \exp &\left(\int_0^t (\hat{h}_1(s) - \hat{h}_2(s) \right)^T dy(s) \\ &- \frac{1}{2} \int_0^t \left(\|\hat{h}_1(s)\|^2 - \|\hat{h}_2(s)\|^2 \right) ds \end{aligned} \tag{1.3}$$

In [7] we showed that the optimal sequential detector utilizes threshold policies under both Neyman-Pearson and Bayes formulations and the likehood ratio Λ_t.

First it is clear that the detector has to select two things. A time τ, to stop collecting data, and a decision δ which declares one of the two hypotheses. Given the miss and false alarm probabilities α, β one computes thresholds A, B [7] and then the optimal detection strategy is given by

$$\tau^* = \inf \{ t \geq 0 | \Lambda_t \notin (A, B) \} \tag{1.4}$$

$$\delta^* = \begin{cases} 1, & \Lambda_{\tau^*} \geq B \\ 2, & \Lambda_{\tau^*} \leq A. \end{cases} \tag{1.5}$$

It is therefore clear that real time implementation of this rule is based on our ability to compute $\hat{h}_i(t)$ in real time. This is related to the Zakai equation of nonlinear filtering [1]. This is so because

$$\hat{h}^i(t) = \frac{\int u^i(x,t) h^i(x) dx}{\int u^i(x,t) dx} \tag{1.6}$$

where $u^i(x,t)$ is the unnormalized conditional probability density of $x(t)$ given $y(s), s \leq t$, under each model 1, or 2. This density satisfies

the stochastic partial differential equation

$$
\begin{aligned}
du^i(x,t) &= L_i^* u^i(x,t)dt + u^i(x,t)h^{i^T}(x)dy(t) \\
u^i(x,0) &= p_0^i(x) \\
L_i^* u^i(x,t) &= \sum_{k,l} \frac{\partial^2}{\partial x_k \partial x_l}(\sigma_{k\,l}^i u^i(x,t)) - \\
&\quad - \sum_k \frac{\partial}{\partial x_k}(f_k^i(x)u^i(x,t)) \\
\sigma^i(x) &= \frac{1}{2}g^i(x)g^{i^T}(x).
\end{aligned}
\tag{1.7}
$$

It can be shown that the likelihood ratio (1.3) can be represented as

$$
\Lambda_t = \frac{\int u^1(x,t)dx}{\int u^2(x,t)dx}. \tag{1.8}
$$

As a consequence the real-time implementation issue, is reduced to the real time implementation of (1.7), (1.8), by a digital or analog circuit.

The Zakai equation (1.7), first indroduced by Moshe Zakai in [1], plays a fundamental role regarding the resolution of the real-time implementation problem for such algorithms, primarily due to its linearity! As is also well known the real-time implementation of the Zakai equation holds the keys to the solution of partially observed stochastic control problems [5], since it provides the natural state for the equivalent fully observed problem, which the feedback controller utilizes.

In section 4 we describe a special architecture, which can achieve real-time operation for many applications, utilizing systolic arrays, as first demonstrated in [7]. We emphasize that this architecture solves the problem for dimensions of x^i, less than or equal to 2. The higher dimensional problems were not addressed in [7]. In section 6 we provide a solution to the higher dimensional problem based on the so called multigrid method applied to (1.7).

The primary objective of the paper is to demonstrate that Moshe Zakai's fundamental contribution in [1], in addition to its well known theoretical value has paramount implications on the practical feasibility of real-time implementation of any sequential detector and/or estimator as well as feedback controller.

2. Review of Basic Sequential Detection Problems

The fundamental problem (1.1) can be easily reduced to two simpler problems of the following type. We are given a vector-valued signal $x_t \in R^n$ which satisfies the stochastic differential equation

$$\begin{aligned} dx_t &= f(x_t)\,dt + g(x_t)\,dw_t \\ x_0 &= \nu \end{aligned} \tag{2.1}$$

where w_t is a vector standard Brownian motion. Unfortunately, we cannot observe x_t directly, instead we only observe y_t, a vector-valued stochastic process $y_t \in R^p$. Under each hypothesis the observed data is the output of a stochastic differential equation, i.e.,

$$\begin{aligned} &\text{Under } H_1: \quad dy_t = h(x_t)\,dt + dv_t \\ &\text{Under } H_0: \quad dy_t = dv_t \end{aligned} \tag{2.2}$$

where v_t is another standard Brownian motion which is independent of w_t. Notice that if $f(\cdot)$ and $h(\cdot)$ are linear and $g(\cdot)$ is constant then this becomes a standard problem which can be solved by the Kalman filter.

Data are observed continuously starting at an initial time which is taken for convenience to be zero. We let $\mathcal{F}_t^y$ represent the information collected up to time t. At each time t, the decision-maker can either stop and declare one of the hypotheses to be true or can continue collecting data. We let τ represent the termination time and δ represent the decision. The decision-maker selects his decision based on the current information ,$\mathcal{F}_t^y$, so as to minimize an appropriate cost function. More precisely, an *admissible decision policy* is any pair $u = (\tau, \delta)$ of RV's where τ is an $\mathcal{F}_t^y$ stopping time and δ is a $\{0,1\}$-valued $\mathcal{F}_\tau^y$ measurable RV. An admissible policy $u = (\tau, \delta)$ is a *threshold policy* if there exist constants A and B, with $0 < A \leq 1 \leq B < \infty$ and $A \neq B$, such that

$$\tau = \inf(t \geq 0 \mid \Lambda_t \notin (A, B)) \tag{2.3}$$

$$\delta = \begin{cases} 1, & \Lambda_\tau \geq B \\ 0, & \Lambda_\tau \leq A \end{cases} \tag{2.4}$$

Here Λ_t is the likelihood ratio associated with this problem, namely

$$\Lambda_t = \exp\left(\int_0^t \hat{h}_s^T \, dy_s - \frac{1}{2}\int_0^t \|\hat{h}_s\|^2 \, ds\right), \tag{2.5}$$

and

$$\hat{h}_t = E_1(\, h(x) \mid \mathcal{F}_t^y \,). \tag{2.6}$$

Using Girsanov's theorem, it can be shown [6] that for threshold policies

$$P_0(\delta = 1) = \frac{1 - A}{B - A} \qquad P_1(\delta = 0) = \frac{A(B-1)}{B - A}.$$

Some algebra gives the following result.

Let u be a threshold policy with A and B defined by

$$A = \frac{\beta}{1 - \alpha} \qquad B = \frac{1 - \beta}{\alpha} \tag{2.7}$$

where $\alpha + \beta < 1$, then

$$P_0(\delta = 1) = \alpha \qquad P_1(\delta = 0) = \beta. \tag{2.8}$$

Hence, given desired false alarm and miss probabilities, α, β, it is possible to find thresholds, (A, B), so that the resulting threshold policy has the required probabilities.

Given $0 < \alpha, \beta < 1$ with $\alpha + \beta < 1$, let $\mathcal{U}(\alpha, \beta)$ be the set of all admissible policies u such that

$$P_0(\delta = 1) \leq \alpha \qquad P_1(\delta = 0) \leq \beta. \tag{2.9}$$

The fixed probability of error formulation to the sequential hypothesis testing problem requires the solution of the following.

Problem ($\mathcal{P}_F$): *Find u^* in $\mathcal{U}(\alpha, \beta)$ such that for all u in $\mathcal{U}(\alpha, \beta)$,*

$$E_i\left(\int_0^{\tau} \|\hat{h}_s\|^2 \, ds\,\right) \geq E_i\left(\int_0^{\tau^*} \|\hat{h}_s\|^2 \, ds\right), \quad i = 0, 1. \tag{2.10}$$

The term in the expectation above represents the expected signal energy present. Usually, the observation time is minimized, subject to the error probability constraints (2.9). However, in this problem we cannot always be sure that we receive "good" data because $h(x_t)$ is itself a random process. It is clear that the longer we observe y the more energy signal we receive. Therefore, trying to decide "faster" is related to trying to decide while receiving the "minimum" signal energy. This intuitive idea is captured in (2.10). We can now state [7, 8]:

Theorem 1. *If u^* is the threshold policy with constants (A^*, B^*) defined by*

$$A^* = \frac{\beta}{1-\alpha}, \qquad B^* = \frac{1-\beta}{\alpha}, \tag{2.11}$$

then u^* solves problem $(\mathcal{P}_F)$.

For the Bayesian formulation, let H be a $\{0,1\}$-valued RV indicating the true hypothesis. By φ we denote the *a priori* probability that hypothesis H_1 is true.

We shall assume, two costs are incurred. The first cost is for observation and is accrued according to $k\int_0^t \|\hat{h}_s\|^2\,ds$, where $k > 0$ and $\{\hat{h}_t,\ t \geq 0\}$ is defined by (2.6). The second cost is associated with the final decision δ and is given by

$$C(H,\delta) = \begin{cases} c_1, & \text{when } H = 1 \text{ and } \delta = 0; \\ c_2, & \text{when } H = 0 \text{ and } \delta = 1; \\ 0, & \text{otherwise,} \end{cases} \tag{2.12}$$

where $c_1 > 0$ and $c_2 > 0$.

We are interested in minimizing the expected cost. If $u = (\tau, \delta)$ is any admissible policy, then the corresponding expected cost is

$$J(u) = E(\, k\int_0^\tau \|\hat{h}_s\|^2\,ds + C(H,\delta)). \tag{2.13}$$

The Bayesian approach to sequential detection seeks to find the solution to the following problem.

Problem $(\mathcal{P}_B)$: *Given $\varphi \in (0,1)$, find u^* such that,*

$$J(u^*) = \inf_{u \in \mathcal{U}} J(u). \tag{2.14}$$

It can be shown that to any admissible policy u there corresponds a threshold policy which has no greater cost, $J(u)$, therefore, the infimum in (2.14) need only be computed over threshold policies. In fact, it can be shown [7, 8] that the infimum is obtained and the following theorem results.

Theorem 2. *There exists an admissible threshold policy u^* that solves problem $(\mathcal{P}_B)$. The optimal thresholds $0 < A^* \leq 1 \leq B^* < \infty$ with $A^* \neq B^*$, are given by the relations*

$$A^* = \left(\frac{1-\varphi}{\varphi}\right)\left(\frac{a^*}{1-a^*}\right), \qquad B^* = \left(\frac{1-\varphi}{\varphi}\right)\left(\frac{b^*}{1-b^*}\right).$$

where a^* and b^* are the unique solutions of the transcendental equations

$$c_2 + c_1 = k(\Psi'(a^*) - \Psi'(b^*))$$

$$c_2(1-b^*) = c_1 a^* + (b^* - a^*)(c_1 - k\Psi'(a^*)) + k(\Psi(b^*) - \Psi(a^*)),$$

with

$$\Psi(x) = (1-2x)\log\frac{x}{1-x}.$$

satisfying $0 < a^* < b^* < 1$.

Here again the thresholds are unique functions of the cost parameters.

These results hold for the hypothesis testing problem described in (1.1) with the small modification of using Λ_t as defined by (1.3).

3. Numerical Solution for Scalar x

In this section we discuss the numerical method used to approximate $\hat{h}(t)$, Λ_t. As mentioned earlier, it can be shown that $\Lambda_t = \int_{R^n} u(x,t)\,dx$ where $u(x,t)$ is the solution to the *Zakai equation* (1.7). Our strategy will be to find an approximation to $u(x,t)$ which results in a good approximation to Λ_t.

In this section we consider in detail the case when x is scalar. Then in the *Zakai equation* (1.7)

$$\begin{aligned} L^* u(x,t) &= \frac{1}{2}\frac{d^2}{dx^2}\left[g^2(x)\,u(x,t)\right] - \frac{d}{dx}\left[f(x)\,u(x,t)\right] \\ &= a(x)\,u_{xx}(x,t) + b(x)\,u_x(x,t) + c(x)u(x,t) \\ &= A^* u(x,t) + c(x)u(x,t) \end{aligned} \tag{3.1}$$

and

$$\begin{aligned} a(x) &= \frac{1}{2}g^2(x) \\ b(x) &= g(x)\,g'(x) - f(x) \\ c(x) &= g(x)\,g''(x) + (g'(x))^2 + g(x)\,g'(x) - f'(x) \end{aligned} \tag{3.2}$$

with $p_0(x)$ the initial density of x.

In general, it is not possible to explicitly solve (1.7), (3.1). Therefore, we will approximate its solution using finite difference methods. Since (1.7) is a linear parabolic equation we use an implicit discretization for x-derivatives, in order to maintain stability. The general vector valued case is treated in [8, 9].

The solution is approximated on the interval $D = (a, b)$. Let $\Delta x > 0$ and define $x_k = a + k\,\Delta x$ and n such that $x_n \leq b$. Consider the collection of points $\{x_k\}_0^n$ in D. Let $\Delta t > 0$ and define $t_k = k\,\Delta t$.

The value of $u(x,t)$ at the grid point (x_i, t_k) is represented by v_i^k. We replace the x-derivatives in A^* with implicit finite difference approximations. To this end,

$$\begin{aligned} a(x)\,u_{xx}(x,t) \quad &\sim \quad a(x_i)\left(\frac{v_{i+1}^{k+1} - 2v_i^{k+1} + v_{i-1}^{k+1}}{(\Delta x)^2}\right) \\ b(x)\,u_x(x,t) \quad &\sim \quad b(x_i)\left(\frac{v_{i+1}^{k+1} - v_i^{k+1}}{\Delta x}\right) \text{ if } b(x_i) > 0 \\ &\qquad\quad b(x_i)\left(\frac{v_i^{k+1} - v_{i-1}^{k+1}}{\Delta x}\right) \text{ if } b(x_i) < 0 \\ u_t(x,t) \quad &\sim \quad \frac{v_i^{k+1} - v_i^k}{\Delta t} \end{aligned} \tag{3.3}$$

Let V^k represent the vector of mesh points at time $k\,\Delta t$. Then the above approximations result in a matrix A_n which approximates A^*. The approximation takes the form

$$(I - \Delta t\,A_n)V^{k+1} = V^k + \text{ other terms.}$$

Note that $a(x_i)$ is always positive and the special way of choosing the first derivative approximation guarantees that the matrix A_n is diagonally dominant and of the form

$$A_n = \begin{pmatrix} - & + & & \\ + & \ddots & \ddots & \\ & \ddots & \ddots & + \\ & & + & - \end{pmatrix}. \tag{3.4}$$

Therefore $(I - \Delta t\, A_n)$ is strictly diagonally dominant. In fact, it is also inverse positive, i.e., every element of the inverse is positive [10, Corollary 1.6b, p. 221].

The final step is to approximate the solution of (3.1) assuming $A^* = 0$. This leads to the matrix

$$D_k = \operatorname{diag}(e^{h(x_i)\,\Delta y_k + (c(x_i) - \frac{1}{2}h(x_i)^2)\,\Delta t}) \tag{3.5}$$

with $\Delta y_k = y_{(k+1)\Delta t} - y_{k\,\Delta t}$.

The overall approximation is

$$(I - \Delta t\, A_n)V^{k+1} = D_k\, V^k. \tag{3.6}$$

This approximation can be shown to be convergent [8, 9, 11].

A nice property of the above approximation is that it is positivity preserving. Regardless of the relationship of Δt and Δx, the solution V^k is always positive. This is important since we are approximating a probability density which we know can never be negative.

Schemes similar to (3.6) have been discussed in [9, 11]. Furthermore, numerical studies have been performed in [12] using these methods which have produced satisfactory results for approximations to $\hat{h}_t$.

Using V^k defined in (3.6) it is easy to construct a convergent approximation to the likelihood ratio via:

$$\Lambda_t^n = \int_a^b u_n(x,t)\,dx = \sum_{i=0}^{n} v_i^k\,\Delta x$$

Then Λ_t^n is a convergent approximation to Λ_t [8].

4. Real Time Architectures for Scalar and Two Dimensional x

The finite difference scheme used to approximate the solution of the Zakai equation involves solving the linear equation

$$(I - \Delta t A_n)\, V^{k+1} = D_k\, V^k, \tag{4.1}$$

for each time $t = k\Delta t$. Here D_k is a data dependent diagonal matrix. Our goal is to design a multiprocessor to efficiently solve (4.1). This means that:

(1) the time necessary to compute V^{k+1}, given V^k, A, and y_k, should be below a problem dependent threshold; and

(2) the control structure should be simple and regular. We have chosen the *systolic array architecture* of [13] to implement this scheme.

We are interested in systolic processors which perform linear algebra operations. The basic component of these arrays is the *inner product processor* (IPP). At each clock pulse the IPP takes the inputs x, y and a and computes $ax + y$ (the inner product step). This value is output on the y-output line, and the x and a values pass through to their respective output lines untouched. The number of processors in a systolic array solving the system in (4.1) depends only on the bandwidth of the matrix and not on its dimension. This makes a systolic design ideal for implementing *finite difference* approximations, where the number of mesh points is not known a priori but where the maximum bandwidth is determined by the specific scheme.

To solve (4.1), we use Gaussian elimination without pivoting. This method, which is stable since $(I - \Delta t\, A_n)$ is strictly diagonally dominant, results in matrices L and U such that

$$LU = (I - \Delta t\, A_n) \tag{4.2}$$

where U is an upper triangular matrix and L is a unit lower triangular matrix.

The matrices L and U will be bi-diagonal since Gaussian elimination without pivoting is used. As is standard with Gaussian elimination, once the factors L and U are found, (4.1) is solved by simple

back substitution. It is well known that back substitution can be accomplished by systolic arrays [13]. The algorithm takes $2n + 2$ time units to operate. Here n is the dimension of the matrix. Hence we are able to solve (4.1) in $5n + 4$ time units. The extra n units result from a necessary buffering of the data between the two back substitution operations.

Notice that the matrix, $(I - \Delta t\, A_n)$, is independent of the received data, y_t, therefore, the LU factorization only has to be done at the time of filter design.

We have completed various implementations of the resulting processor using board level designs and employing current signal processing chips, with an IBM PC AT as the host. With current technology, such boards are capable of processing data at a rate of 20 kHz. We have also completed a special purpose VLSI chip design, which we named the *Zakai Chip* in honor of Moshe Zakai. We have recently completed a much improved special purpose VLSI chip design which we named the *Zakai II Chip.* Preliminary performance evaluation has been very good.

This programmable architecture solves a long standing problem for scalar state and observation models. It can be easily extended to vector observations of a scalar state. We have also shown that essentially the same architecture works for two dimensional state process x. However, the discovery of appropriate architectures for observations of vector states is much harder. For higher state dimensions, a different architecture is needed. In the rest of the paper we develop such multi-level architectures using numerical schemes based on multigrid methods.

5. Multigrid Algorithms for the Zakai Equation.

In this section we show how multigrid algorithms [14, 15] can be developed for the Zakai equation (1.7) in a systematic manner. As before we employ an implicit full discretization scheme that provides consistent time and space discretizations of the Zakai equation, in the sense that for each choice of discretization mesh, the problem can be interpreted as a nonlinear filtering problem for a discrete time, discrete state, hidden Markov chain.

To discretize (1.7), we choose a time step Δ and a space discretization mesh of size ϵ which determines a "grid" in R^d, the space where we wish to solve the Zakai equation. In other words $x \in R^d$. Using results on estimates of the tail behavior of $u(x,t)$ as $\|x\| \to \infty$ we can actually select a rectangular domain in R^d, D, where we are primarily interested in solving (1.7). Let us suppose that there are $n(\epsilon)$ points on each dimension of the grid of size ϵ. Let $G_d(\epsilon)$ denote the hypercube generated in R^d by the spatial mesh of size ϵ. We assume for simplicity here uniform grid spacing.

Given this set-up we can construct following the methods of Kushner [9] a matrix $A(\epsilon)$ which approximates the operator L^* in (1.7), in the sense that $A(\epsilon)$ defines an approximating Markov chain to the diffusion (1.1). Let

$$\begin{aligned} \Delta y(k) &= y((k+1)\Delta) - y(k\ \Delta) \\ H_i(\epsilon) &= h(x_i(\epsilon)) \end{aligned} \tag{5.1}$$

where $x_i(\epsilon)$ is a generic point on the grid $G_d(\epsilon)$. Finally let $\mathbf{V}^{k+1}(\epsilon)$ be the vector of samples $u(x_i(\epsilon),(k+1)\Delta)$ of the unnormalized conditional density over the grid $G_d(\epsilon)$. In [8] the following was proved, using semigroup techniques.

Theorem 3: Let

$$\mathbf{D}(\epsilon,k) = \text{ diag } \{\exp(H_i^T(\epsilon)\Delta y(k) - \frac{1}{2}\|H_i(\epsilon)\|^2\Delta)\}$$

and consider the implicit iteration

$$\begin{aligned} \frac{(I - \Delta A(\epsilon))\mathbf{V}^{k+1}}{(\epsilon)} &= D(\epsilon,k)V^k(\epsilon) \\ V^o(\epsilon) &= \{p_0(x_i(\epsilon))\}. \end{aligned} \tag{5.2}$$

Then as $k\Delta \to t$, with $\epsilon \to 0$ (along some sequence)

$$\lim_{\epsilon \to 0} \sup_{i \in G_d(\epsilon)} |V_i^k(\epsilon) - u(x_i(\epsilon), k\Delta)| = 0. \tag{5.3}$$

In other words Theorem 3, provides a uniformly convergent scheme. Once we have this the likelihood ratio (1.8) can be easily approximated since

$$\int u(x,t)dx \sim \sum_{i \in G_d} V_i^k(\epsilon)\Delta x(\epsilon); \text{ for } k\Delta \le t < (k+1)\Delta \tag{5.4}$$

where $\Delta x(\epsilon)$ is the approximation to the volume element in $G_d(\epsilon)$.

Therefore the real-time solution of (1.7) has been reduced to the analysis of the real-time computation of (5.2). We consider the matrix $I - \Delta A(\epsilon)$ on $G_d(\epsilon)$ and $G_{d+1}(\epsilon)$. There is a convenient way to label the states of the resulting Markov chain so as to have some recursion between these two representations. Indeed let the two matrices be denoted as Γ_d and Γ_{d+1} respectively. Then [16]

$$\Gamma_{d+1} = \begin{bmatrix} \Gamma_d & T & 0 & \cdots & 0 \\ T & \Gamma_d & T & \ddots & 0 \\ 0 & T & \ddots & \Gamma_d & T \\ 0 & \cdots & 0 & T & \Gamma_d. \end{bmatrix} \tag{5.5}$$

where T is a tridiagonal matrix with positive entries. Γ_{d+1} is an $n \times n$ block matrix. Furthermore it is straightforward to establish [16] that for any d, Γ_d is strongly diagonally dominant. Furthermore $I - A(\epsilon)\Delta$ has finite bandwidth [16]. The strong diagonal dominance of $I - \Delta A(\epsilon)$ implies that we will need no pivoting. As we shall see this property will help also in the selection of the relaxation scheme in the multigrid iteration.

The fundamental idea of multigrid (MG) algorithms is relatively easy to understand [14, 15]. The primary reasons for using MG methods are as follows. Direct solvers of discretized p.d.e's have computation time that grows linearly with n, the width of the finest grid, while MG methods can actually do much better than this as we shall see. As for relaxation schemes, slow convergence is a typical problem, although they are perfectly suited for parallel implementation, as they rely only on "local" information when a sweep is performed. Thus, relaxation schemes have a computation time that is independent of the size of the grid. Because of its naturally parallel properties, it turns out that the Multigrid method has a computation time that is essentially independent of the dimension of the problem. Because we wish to compute in real-time, such a numerical method is an ideal candidate for investigation.

We now described a one-cycle full Multigrid algorithm program. Let there be K point-grids which we will denote by $G_1, G_2, \ldots, G_K$ with the finest being G_K and the coarsest being G_1.

The finest grid contains the problem:

$$L^K U^K = f^K. \tag{5.6}$$

Smoothing Part I:

Given an initial approximation to the problem in (5.7), smooth j_1 times to obtain u^K.

Coarse-grid correction:

Compute the residual $d^K = f^K - L^K u^K$.

Inject the residual into the coarser grid G_{K-1},

$$d^{K-1} = I_K^{K-1} d^K.$$

Compute the approximate solution $\tilde{v}^{K-1}$ to the residual equation on G_{K-1}:

$$L^{K-1}\, v^{K-1} = d^{K-1}, \tag{5.7}$$

by performing $c \geq 1$ iterations of the Multigrid method, but this time we will be using the grids $G_{K-1}, G_{K-2}, \ldots, G_1$ applied to equation (5.7).

Interpolate the correction $\tilde{v}^K = I_{K-1}^K\, \tilde{v}^{K-1}$.

Compute the corrected approximation on G^K,

$$u^K + \tilde{v}^K. \tag{5.8}$$

Smoothing Part II:

Compute a new approximation to U^K by applying relaxation sweeps to $u^K + \tilde{v}^K$.

The recursive structure of the algorithm entering just after eq. (5.7) is apparent. Here the algorithm simply repeats itself, so in the case of $c = 1$ we have initial smoothing, computation of residual equation, injection to coarser grid, all until the coarsest grid is reached, where the equation is directly solved. Then we have interpolation upward through the grids, offering each finer grid an approximation for relaxation. This would be a "V-shape" structure as opposed to a "W-shape struture [16]. Only if $c > 1$, would we have a "W shape" structure. Of course it is clear that there can be many variants of the algorithm, depending on the number of interpolation and injection operations.

The convergence of the multigrid method is based on the following representation of the algorithm. For the general grid G_k we will have the equation

$$L^k U_k = f_k \quad , k = 1, \ldots, K. \tag{5.9}$$

The formula for obtaining a new approximation to the solution U_k from the old one u_k can be written as

$$\begin{aligned} \tilde{u}_k &= (I - ML^k)u_k + Mf_k \\ &= S_k u_k. \end{aligned} \tag{5.10}$$

Here S_k is the smoothing operation on the grid G_k, and we assume that M is invertible and the smoother is consistent. With this notation S_k^j denotes the smoother that uses j relaxation sweeps, or is applied j times.

As examples of smoothers, define D to be the matrix whose diagonal entries are equal to those of L^k, and which is zero everywhere else. Then $m = \omega D^{-1}$ is the modified Jacobi method. If T is the "upper triangular part" of L^k, and zero elsewhere, then we have the Gauss-Seidel method by setting $M = T^{-1}$. In fact, M is usually some approximation to the inverse of L^k, which forces $\rho(I - ML^k)$ to be close to zero.

By constructing the "Multigrid operator" we can show that, like any other iterative process, convergence is guaranteed under certain conditions.

Given u^k as the old approximation, the new approximation $\tilde{u}^k$ will be

$$\tilde{u}_k = M_k \; u_k + I_{k-1}^k (L^{k-1})^{-1} \; I_k^{k-1} f_k. \tag{5.11}$$

M_k will be the Multigrid operator on grid G_k we will concentrate on, for it is its spectral radius that determines whether the iteration converges or not. By M_k^c we will mean c multiples of the MG operator applied on the k grids. The following recursion defines this operator, which begins at grid level 2 and proceed up to $k = K - 1$ where we have K grid levels:

$$\begin{aligned} M_k^{k-1} &= S_k^{j_2}(I_k - I_{k-1}^k (L^{k-1})^{-1} \; I_k^{k-1} L^k) S_k^{j_1} \\ A_k^{k+1} &= S_{k+1}^{j_2} \; I_k^{k+1} : G_k \to G_{k+1} \\ A_{k+1}^k &= (L^k)^{-1} \; I_{k+1}^k (L^{k+1})^{-1} \; S_{k+1}^{j_1} : G_{k+1} \to G_k \end{aligned} \tag{5.12}$$

Thus we can write,

$$M_{k+1} = M_{k+1}^k + A_k^{k+1} \; M_k^c \; A_{k+1}^k. \tag{5.13}$$

Now if $\|M_{k+1}^k\|, \|A_k^{k+1}\|$ and $\|A_{k+1}^k\|$ for $k \leq K-1$ are known, then one can obtain an estimate of $\|M_K\|$, where $\|\cdot\|$ represents any reasonable operator norm; we refer to [17] for more detailed results on stability and convergence.

For the case of interest here, i.e. the discretization of the Zakai equation (5.2), following well known methodology for MG application we first identify a simpler, albeit characteristic problem. This is the problem with no input, i.e. when the right hand side of (5.2) becomes $V^k(\epsilon)$. In other words if one understands how MG is applied to the discretized Fokker-Planck equation

$$(I - \Delta A(\epsilon))\, V^{l+1}(\epsilon) = V^l(\epsilon), \tag{5.14}$$

then complete understanding of the application of MG to the Zakai equation is straightforward.

It follows [16] that the spectral radius of the associated MG operator (5.12) is determined entirely by the matrix $I - \Delta A(\epsilon)$, along with the choice of relaxation scheme. Also note that because $I - \Delta A(\epsilon)$ is not time dependent all *program parameters are precomputable.* In particular for the Zakai equation, they do not depend on the sample path $y(\cdot)$.

A highly recommended relaxation method in MG applications is the so called *successive overrelaxation method* (SOR). To define it suppose one wants to solve

$$Ax = b$$

with $a_{ii} \neq 0$. Then define B to be the $n \times n$ matrix

$$b_{ij} = \begin{cases} -a_{ij}/a_{ii}, & i \neq j \\ 0, & i = j \end{cases}$$

and define the vector c in R^n to have components, $c_i = b_i/a_{ii}$. Then let us consider the $L - U$ decomposition of B, $B = L + U$. Choose a real number ω, and define the iteration

$$x_{n+1} = \omega(Lx_{n+1} + Ux_n + c) + (1 - w)x_n. \tag{5.15}$$

This is the SOR method. If $\omega = 1$, the SOR method reduces to the Gauss-Seidel method, with $\omega > 1$ implying overcorrecting, and $\omega < 1$ implying undercorrecting.

Recall that the matrix $I - \Delta A(\epsilon)$, for the discretized Zakai equation, is strongly diagonally dominant. Furthermore this matrix is also an L-matrix, i.e. it has positive diagonal elements and non positive off diagonal elements. Finally this matrix is *consistently ordered* [16]. This is a consequence of the natural ordering on a rectangular grid. One can measure the properties of smoothing operators with a variety of measures [16]. So one can describe "optimal" smoothing operation. We thus have [16]:

Theorem 4: Because of the properties of $I - \Delta A(\epsilon)$, the MG operator converges and the optimal relaxation scheme for the Zakai equation is the SOR method. There is an optimal choice for ω in (5.15) with respect to convergence as well.

6. Architectures for Implementing MG in Real-Time.

In this section we analyze the complexity of the MG schemes described in section 5, in particular with respect to real time implementation. We shall see that the result is a multilayer processor network. Here the processors and the interconnections are more complicated than the ones used in the systolic architecture of [7]. So fabrication is a much harder problem.

The computing network will be a system of grids of identical processing elements. Therefore, we have two kinds of grids, one of points and one of processors, and these will be layered one on top of another. For each $1 \leq k \leq K$, processor grid P_k has $(n_k)^\gamma$ elements, where γ is a positive integer not greater than the problem dimension d. Also, we have $n_K = n$, and $n_i < n_j$, if $i < j$.

Similarly, in keeping with the above notation, there are, for each $1 \leq k \leq K$ a corresponding point grid G_k with $(n_k)^d$ points. (Note that the number of processors per grid is never greater than the number of points). Again we have $n_K = n$ while $n_i < n_j$ if $i < j$. A key assumption, which is quite realistic, is that for each step of the multigrid algorithm on point grid G_k, the processing grid P_k requires $O((n_k)^{d-\gamma})$ time to perform its computations.

To design a parallel machine capable of performing the MG algorithm, we assume our problem is in d dimensions over a rectangular domain using a regular point grid of n^d points. We further have

$$\begin{aligned} n_K &= n \\ n_{k+1} &= a(n_k + 1) - 1, \quad k = 1, 2, \ldots K - 1 \end{aligned} \tag{6.1}$$

for some integer $a \geq 2$. We map grid points in such a way that neighboring grid points reside in the same or neighboring processors.

Smoothing sweeps of at least some type can be accomplished in $O(n^{d-\gamma})$ time with this given connectivity. Let t be the time taken by a single processor to perform the operations at a single gridpoint that, done over the whole grid, constitute a smoothing sweep. Then, setting S as the time needed to perform the smoothing sweep over the whole of grid G_k on processor grid P_k, we have,

$$S = t\ n_k^{d-\gamma}. \tag{6.2}$$

Obviously, it is to our advantage to conduct as few smoothing sweeps as necessary and still assure sufficient accuracy.

Now processor grid P_k is connected to processor P_{k+1}. Processor $i \in P_k$ is connected to processor $a(i + 1) - \mathbf{1} \in P_{k+1}$ where $\mathbf{1} = (1, 1, \ldots, 1)$. These connections allow any intergrid operations, such as interpolation, to be performed in $O(S)$ time. Now define the system of processor grids $\{P_1, P_2, \ldots, P_J\}$ as the machine M_J for $J = 1, 2, \ldots, K$. Then the execution of MG performed by M_k proceeds as follows:

1. First, j smoothing sweeps on grid G_k are done by P_k; all other processor grids idle.
2. The coarse grid equation is formed by P_k and transferred to P_{k-1}.
3. MG is iterated c times on grid G_{k-1} by M_{k-1}. P_k is idle.
4. The solution v^{k-1} is transferred to P_k by iterpolation: $I_{k-1}^k v^{k-1}$
5. The remaining m smoothing sweeps are done by P_k.

Now we let $W(n)$ be the time needed for steps $1, 2, 4, 5$ and find

$$W(n) = (j + m + s)\ t\ n^{d-\gamma}, \tag{6.3}$$

where s is the ratio of the time needed to perform steps 2 and 4 to the time needed for one smoothing sweep. Note that s is independent of n, d and γ.

We discuss now the time complexity of MG. We will denote by $T(n)$ the time complexity of the MG algorithm on a grid of n^d points. It turns out that $T(n)$ solves the recurrence:

$$T(an) = cT(n) + W(an), \tag{6.4}$$

where $W(an)$ denotes the work needed to pre-process and post-process the (an)-grid iterate before and after transfer to the coarser n-grid. In effect the term $W(an)$ includes the smoothing sweeps, the computation of the coarse grid correction equation (i.e., the right-hand side d^{k-1}) and the interpolation back to the fine grid $(I_{k-1}^k\ v^{k-1})$. Then we have

Theorem 5, [18]: Let $T_p(\cdot)$ be a particular solution of (6.4), i.e.,

$$T_p(an) = c\ T_p(n) + W(an).$$

Then the general solution of (6.4) is:

$$T(n) = \alpha\ n^{\log_a c} + T_p(n), \tag{6.5}$$

where α is an arbitrary constant. Using this result we have the general solution to (6.5),

$$T(n) = \begin{cases} \beta(a^p/(a^p - c))n^p & \text{if } c < a^p, \\ \beta n^p \log_a n + O(n^p) & \text{if } c = a^p, \\ O(n^{\log_a c}) & \text{if } c > a^p. \end{cases} \tag{6.6}$$

We see that it would take a single processor $O(n)$ steps to complete the above mentioned tasks on one dimension, while n processors could do the same for a two-dimensional problem in $O(n)$ time.

We say that the MG algorithm is of *optimal order* if $T(n) = O(n^{d-\gamma})$, a possibility that is sometimes precluded by some choices of c, a, γ and d, which in turn influence $T(n)$. Examination of (6.6) demonstrates the relations between the various parameters. As an

example, in the one-processor case, with $\gamma = 0, d = 2$, we have $g(n) = n^2$. We then have an optimal scheme if $a = 2, c < 4$, for only then is $T(n) = O(n^2)$. But $c \geq 4$ is non-optimal, with $T(n) = O(n^2 \log n)$ for $c = 4$.

In general, we have an optimal scheme if and only if $c < a^d$.

There also exists a natural way to build a VLSI system to implement our algorithms. The $\gamma = 1$ machine can be embedded in two dimensions as a system of communicating rows of processors. The $\gamma = 2$ machine can be embedded in three dimensions as a system of communicating planes, and so on. Realizations in three-space will be possible in a natural way for any value of γ. Consider the case of $d = 2, \gamma = 2$. In this case, we have a set of homogeneous planar systolic arrays layered one on top of the other. If we let $a = 2, K = 3$, and $n_1 = 1, n_2 = 2(1 + 1) - 1 = 3, n_3 = 2(3 + 1) - 1 = 7$, we would have a 7×7 array on top of a 3×3 array which is then on top of a single processor corresponding to n_1. Unfortunately, this design differs from the classical systolic array concept of Kung [19] in that there exists no layout in which wire lengths are all equal. Also, each layer of the system is homogeneous while the entire machine is clearly not.

Now the four parameters c, a, γ, d are to be chosen with any implementation MG, and, of course, they are not unrelated to each other. Extending the earlier notation, we call any one choice of the four a design and denote its corresponding computing time by $T(c, a, \gamma, d)$. We will now begin with an examination of the trade-offs incurred by one choice over another. Following [18], an important issue is *efficiency*, E vs. *speedup* S in a particular design. We define,

$$S(c, a, \gamma, d) = T(c, a, 0, d)/T(c, a, \gamma, d) \tag{6.7}$$
$$E(c, a, \gamma, d) = T(c, a, 0, d)/(P(\gamma)T(c, a, \gamma, d))$$

Note that the speedup S corresponds to the gain in speed going from the one-processor system to that of the multiprocessor. Whereas the efficiency E reflects the trade-off between using more processors vs. time.

We say that a design $T(c, a, \gamma, d)$ is *asymptotically efficient* if E tends to a constant as $n \to +\infty$, and it will be *asymptotically inefficient* if $E \to 0$ asn $n \to +\infty$.

Theorem 6, [18]: Let $\gamma > 0$.

1) If $c < a^{d-\gamma}$ then $E(c,a,\gamma,d) = (a^\gamma - 1)(a^{d-\gamma} - c)/(a^d - c)$.
2) If $c = a^{d-\gamma}$ then $E(c,a,\gamma,d) = (a^\gamma - 1)a^{d-\gamma}/((a^d - c)\log_a n)$.
3) If $c > a^{d-\gamma}$ then

$$E(c,a,\gamma,d) = \begin{cases} O(1/n^{\log_a(c-d+\gamma)}) & \text{if } c < a^d \\ O((\log_a n)/n^\gamma) & \text{if } c = a^d \\ O(1/n^\gamma) & \text{if } c > a^d \end{cases}$$

We have at once that

1) *A design is asymptotically efficient if and only if* $c < a^{d-\gamma}$.

2) *The fully parallel design* $\gamma = d$, *is always asymptotically inefficient.*

3) "Halfway" between asymptotic efficiency and inefficiency is *logarithmic asymptotic efficiency*, with $E = O(\log n)$, as $n \to = \infty$. *A fully parallel design* ($\gamma = d$) *if logarithmically asymptotically efficient iff* $c = 1$.

4) If we start with a non-optimal design in the one processor case, then adding more processors will not make the design asymptotically efficient.

To get $T(n) = O(n)$ we have to select $c = 1$.

We also have considered the concurrent iteration schemes of Gannon and Van Rosendale [20].

Thus the fully parallel architecture has a computation time of at most $O(\log n)$ and so it is very competitive with the systolic direct solver. More importantly, this time is largely independent of dimension d, at least for small values of d. Of course, increases in d will result in large increases in circuit layout area, due to an increase in interconnections between grids, and thus a subsequent loss in computing speed.

We can implement the SOR method in a parallel fashion, using the "red-block" or "checker board" method [16]. In higher dimensions we need to utilize multicolor ordering. Employing the intrinsic locality of the SOR we can implement it asynchronously as well.

A detailed analysis of timing performed in [16] demonstrates that if the real-time constraint for the Zakai equation is 1 msec, then we can realistically achieve real-time implementation with the multi-layered networks of this section, only for dimension $d \leq 8$.

References

[1] M. Zakai, "On the Optimal Filtering of Diffusion Processes", *Z. Wahr. Verw. Geb.,* 11, pp. 230-243, 1969.

[2] M. Hazewinkel and J.C. Willems, edts, *Stochastic Systems: The Mathematics of Filtering and Identification*, Proc. of NATO Advanced Study Institute, Les Arcs, France, Dordrecht, The Netherlands: Reidel 1981.

[3] J. Doob, *Stochastic Processes*, Wiley, 1953.

[4] E. Wong and B. Hajek, *Stochastic Processes in Engineering Systems*, Springer-Verlag, 1985.

[5] A. Bensoussan, "Maximum Principle and Dynamic Programming Approaches to the Optimal Control of Partially Observed Diffusions", *Stochastics*, 9, pp. 169-222, 1983.

[6] R.S. Liptser and A.N. Shiryayev, *Statistics of Random Processes I: General Theory*, Springer-Verlag, New York, 1977.

[7] John S. Baras and Anthony LaVigna,"Real-Time Sequential Detection for Diffusion Signals", *Proc. 26th Conf. on Decision and Control*, pp. 1153-1157, 1987.

[8] A. LaVigna, "Real Time Sequential Hypothesis Testing for Diffusion Signals," M.S. Thesis, Univ. of Maryland, 1986.

[9] H.J. Kushner, *Probability Methods for Approximations in Stochastic Control and for Elliptic Equations*, Academic Press, New York, 1977.

[10] J. Schröder, "M-Matrices and Generalizations Using an Operator Theory Approach," *SIAM Review*, 20, pp. 213-244, 1978.

[11] E. Pardoux and D. Talay, "Discretization and Simulation of Stochastic Differential Equations," in *Publication de Mathematiques Appliquees Marseille-Toulon*, Université de Provence, Marseille, 1983.

[12] Y. Yavin, "Numerical Studies in Nonlinear Filtering," in *Lecture Notes in Control and Information Sciences 65*, Springer-Verlag, New York, 1985.

[13] H.T. Kung and C.E. Leiserson, "Algorithms for VLSI Processor Arrays," in *Introduction to VLSI Systems*, C. Mead and L. Conway, pp. 271-292, Addison-Wesley, Reading, Mass., 1980.

[14] S.F. McCormick, Edt, *Multigrid Methods*, Frontiers in Applied Mathematics, SIAM 1987.

[15] W.L. Briggs, *A Multigrid Tutorial*, SIAM, 1987.

[16] K. Holley, "Applications of the Multigrid Algorithm to Solving the Zakai Equation of Nonlinear Filtering With VLSI Implementation", Ph.D. Thesis, University of Maryland, December 1986.

[17] K. Stüben and V. Trottenberg, "Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications", in *Multigrid Methods*, W. Hackbusch and W. Trottenberg (edts), Springer Verlag, 1982.

[18] T. Chen and R. Schreiber, "Parallel Networks for Multi-grid Algorithms: Architecture and Complexity", *SIAM J. Sci. Stat. Comput.*, Vol. 6, No. 3, July 1985.

[19] H.T. Kung, "Systolic Algorithms", in *Large Scale Scientific Computation*, Academic Press, 1984.

[20] D. Gannon and J. Van Rosendale, "Highly Parallel Multigrid Solvers for Elliptic PDE's: An Experimental Analysis", ICASE Report No. 82-36, Nov. 1982.