Construction of Nonlinear Filter Algorithms Using the Saddlepoint Approximation

by

Esosa O. Amayo
Bachelor of Science, Electrical Science and Engineering

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology
July 18, 2006

Author____________________________
Department of Electrical Engineering and Computer Science
July 18, 2006

Certified by____________________________
Emery N. Brown
Thesis Supervisor

Certified by____________________________
John L. Wyatt
Thesis Co-Supervisor

Accepted by____________________________
Arthur C. Smith
Chairman, Department Committee on Graduate Studies

Construction of Nonlinear Filter Algorithms Using the Saddlepoint Approximation

by

Esosa O. Amayo

Submitted to the Department of Electrical Engineering and Computer Science
July 18, 2005

in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

ABSTRACT

In this thesis we propose the use of the saddlepoint method to construct nonlinear filtering algorithms. To our knowledge, while the saddlepoint approximation has been used very successfully in the statistics literature (as an example the saddlepoint method provides a simple, highly accurate approximation to the density of the maximum likelihood estimator of a non-random parameter given a set of measurements), its potential for use in the dynamic setting of the nonlinear filtering problem has yet to be realized. This is probably because the assumptions on the form of the integrand that is typical in the asymptotic analysis literature do not necessarily hold in the filtering context. We show that the assumptions typical in asymptotic analysis (and which are directly applicable in statistical inference since the statistics applications usually involve estimating the density of a function of a sequence of random variables) can be modified in a way that is still relevant in the nonlinear filtering context while still preserving a property of the saddlepoint approximation that has made it very useful in statistical inference, namely, that the shape of the desired density is accurately approximated. As a result, the approximation can be used to calculate estimates of the mean and confidence intervals and also serves as an excellent choice of proposal density for particle filtering. We will show how to construct filtering algorithms based on the saddle point approximation.

Thesis Supervisor: Emery Brown
Thesis Co-Supervisor: John Wyatt

To the memory of my father Professor Airen Amayo

Acknowledgements

I would like to thank my advisor, Professor Emery Brown for being such an inspiring role model as a researcher. I am deeply grateful for his patience with me and constant support through some very difficult personal trials. In some sense, this thesis would not have happened had Professor John Wyatt not been such a clear and inspiring teacher when I took 6.432 with him. This thesis is the result of my interest in estimation problems that was sparked by that class. I am also greatly indebted to him for his friendship, constant support, encouragement and invaluable advice. Words are inadequate to express my gratitude to Professor Tayo Akinwande who has been a source of inspiration, and above all a father to me since I first arrived here as an undergraduate. My graduate student life would have been stillborn had it not been for the financial support from Dean Isaac Colbert that saved me when I could not initially get funding. I would also like to express my thanks to Dr Chris Oriakhi at Hewlett-Packard Research, whose achievements helped motivate and spur me on and whose friendship and advice helped me through a difficult time.

This thesis would not have been possible without the support and understanding of my colleagues at Aware, in particular my manager Dr Michael Lund and the head of our Engineering division, Dr Rick Gross .

My thanks also go out to Anne Hunter, Vera Sayzew and all the good folks at the course 6 Undergraduate Office for their patience (which I have tested on several occasions over the years and found to be truly limitless) and willingness to go above and beyond the call of duty to help students. I cannot thank Cheryl Charles enough for her kindness, motherly concern and support. I would also like to thank Carol Frederick, my first boss at MIT, for all her support and friendship.

To Dr Jack Lloyd, thank you.

What is life without ones family and friends?

To Stanislav Funiak, Yogishwar Maharaj, Mamat Kaimudeen, Emilia Simeonova, Elana Wang, Cang Kim Truong, Buddhika Khottachi, Felecia Caic, Senkodan Thevendran, Eric Escobar Cabrera, Fitih Mustafa Mohammed, Nathaniel Choge, Ming Zhang, Riyaz Bachani, Bill Blackwell, Pia Bhanerjee, Abhinav Kumar, Bosun Adeoti, Paul Njoroge, Kwaku Owusu Abrokwah, Nii Lartey Dodoo and all my friends I say "thanks *Bushtis*" for making my life bearable.

I would like to thank my lab buddies, Chris Long, Patrick Purdon, Uri Eden, Somangshu Mukherji, Mike Prerau, Supratim Saha, Riccardo Barbieri, Murat Okatan, Tyler Gibson, and Julie Scott for always being there to talk about anything even at the busiest of times.

I spent countless hours talking with Ram Srinivasan about my thesis, and about life outside of the thesis and each conversation left me with renewed encouragement and energy.

To my sister, Dr Ruth Uwaifo, I say "thank you" from the bottom of my heart for the constant prayers and support by letters while you were in Nigeria and by phone when you moved over here.

To my brothers Muyiwa Olubuyide, Jitin Asnaani, and Premraj Janardanan, I say "thank you" from the bottom of my heart for being there and for enriching my life. Your friendship has made me a better person.

Last but definitely not least, I want to thank my mother and sister who refused to be deterred by little nuisances like the Atlantic Ocean in making me feel their love even here.

Contents

Chapter 1

Introduction

1.1 The Nonlinear Filtering Problem

Our goal is to sequentially estimate the states of a discrete-time system that has nonlinearities in the state and/or the observation process and whose state and/or observation noise processes are not necessarily Gaussian. We will be working with systems that can be described by the following general state space model

$$\begin{aligned} x_n &\sim q(x_n \mid x_{n-1}) \\ y_n &\sim r(y_n \mid x_n) \end{aligned} \tag{1.1}$$

where y_n is an observation from the system, x_n is the unknown state process, $q(x_n \mid x_{n-1})$ is the conditional distribution of x_n given x_{n-1} and $r(y_n \mid x_n)$ is the conditional distribution of y_n given x_n. The initial state is distributed according to the distribution $p(x_0 \mid Y_0)$. We assume that the states follow a first order Markov process. That is

$$p(x_k \mid x_{k-1}, x_{k-2}, \ldots, x_0) = q(x_k \mid x_{k-1}) \tag{1.2}$$

We also assume that the observations are independent given the states. That is

$$p(y_k \mid x_k, y_l, y_m \ldots, y_s) = r(y_k \mid x_k) \text{ if } k > l > m > \ldots > s \tag{1.3}$$

The nonlinear filtering problem is to evaluate $p(x_k \mid Y_k)$, the distribution of the current state given the observations up to the present time $Y_k = \{y_1, \ldots, y_k\}$ and the initial distribution of the state process. Once we have the filtering density we can calculate a variety of estimates of the state such as the mean (which is the minimum mean squared error estimate of the state), the mode, the median, confidence intervals and so on.

The general state space model considered above includes various important models such as:

(i) The linear state space model with Gaussian or non-Gaussian white noises w_n and ε_n

$$\begin{aligned} x_n &= F x_{n-1} + G w_n \\ y_n &= H x_n + \varepsilon_n \end{aligned} \tag{1.4}$$

(ii) The nonlinear model with additive noise

$$\begin{aligned} x_n &= f(x_{n-1}) + w_n \\ y_n &= h(x_n) + \varepsilon_n \end{aligned} \tag{1.5}$$

(iii) The more general nonlinear model with known input u_n

$$\begin{aligned} x_n &= f(x_{n-1}, u_{n-1}, w_{n-1}) \\ y_n &= h(x_n, u_n, \varepsilon_n) \end{aligned} \tag{1.6}$$

(iv) Discrete Process with probabilistic description of observation process parameterized by the state (for example state space system with point process observations)

$$\begin{aligned} x_n &= F x_{n-1} + G w_n \\ y_n &\sim Dist(x_n) \end{aligned} \tag{1.7}$$

1.1.1 Recursive Formulae for Filtering

Using only the definition of conditional probability, the Chapman-Kolmogorov equation for marginal distributions, Bayes' law, and the Markov assumptions mentioned in the preceding section we obtain recursive formulae for the one step ahead prediction and filtering densities as follows:

One step ahead prediction density:

$$p(x_k \mid Y_{k-1}) = \int q(x_k \mid x_{k-1}) p(x_{k-1} \mid Y_{k-1}) dx_{k-1} \tag{1.8}$$

Filtering density:

$$p(x_k \mid Y_k) = \frac{r(y_k \mid x_k) p(x_k \mid Y_{k-1})}{p(y_k \mid Y_{k-1})} \tag{1.9}$$

where $p(y_k \mid Y_{k-1})$ is calculated from

$$p(y_k \mid Y_{k-1}) = \int r(y_k \mid x_k) p(x_k \mid Y_{k-1}) dx_k \tag{1.10}$$

1.2 Brief Summary of Current Approaches to the Nonlinear Filtering Problem

If the state and observation equations are linear and initial state as well as the system and observation noise processes are Gaussian, the system of equations (1.8)-(1.9) has an exact Gaussian solution and the means and covariances of the one-step and filtering densities are given by the well-known Kalman Filter recursion. However in the general not-necessarily-Gaussian case these densities have to be approximated. The most commonly used practical approaches to this problem are the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF), methods relying on Gaussian approximations to the left hand sides of (1.8) and (1.9), perfect Monte Carlo simulation (when it is possible to sample directly from the true posterior density) and particle filtering (which is used when it is not possible to sample directly from the true posterior distribution). We will outline the salient points of some of these methods in the next chapter.

1.3 Objectives and Contributions of the Thesis

In this thesis we propose the use of the saddlepoint method to construct nonlinear filtering algorithms. To our knowledge, while the saddlepoint approximation has been used very successfully in the statistics literature (as an example the saddlepoint method provides a simple, highly accurate approximation to the density of the maximum likelihood estimator of a non-random parameter given a set of measurements [Barndorff-Nielsen, 1983]), its potential for use in the dynamic setting of the nonlinear filtering problem has yet to be realized. This is probably because the assumptions on the form of the integrand that is typical in the asymptotic analysis literature (for example [De Bruijn, 1981]) do not necessarily hold in the filtering context.

In the sequel we will develop filtering algorithms based on the saddle point approximation. This gives us an approximation whose shape accurately approximates the shape of the target density. As a result, the approximation can be used to calculate estimates of the mean and confidence intervals and also serves as an excellent choice of proposal density for particle filtering.

In Chapter 2 we will discuss the most common approaches to the nonlinear filtering problem. We will then discuss the existing theory for approximating asymptotic integrals using the Laplace method (the analogue of the Saddlepoint method on the real line) and then develop some new results with modified assumptions that can be used to approximate probability densities that can not necessarily expressed as the integral of a function which is the exponential of a well-behaved function scaled by a large constant. We will then show in Chapter 4 how the results obtained in Chapter 3 can be used to construct filtering algorithms.

Chapter 2

Summary of Current Approaches To The Filtering Problem

In this chapter we will first describe the well-known Kalman Filter solution to the linear estimation problem. We will then outline some of the methods currently being used to deal with the more general non-linear estimation problem such as the Extended Kalman Filter (EKF), which is based on the Kalman Filter and Sequential Monte Carlo (Particle Filtering) methods.

2.1 Linear Least-Squares Estimation

In this section we will summarize the Kalman Filter solution for obtaining the linear minimum mean-squared error estimate of a random process x_n from measurements y_n. Our treatment will closely follow that of [Luenberger, 1969, Chapter 4].

The random process x_n is assumed to evolve according to a vector difference equation

$$x_{n+1} = F_n x_n + w_n, \qquad n = 0,1,2,..., \tag{2.1}$$

where x_n is an n-dimensional random vector, F_n is a known $n \times n$ matrix and w_n is an n-dimensional random vector input with mean zero satisfying $E[w(k)w'(l)] = Q_k \delta_{kl}$ (in other words, the random vector input is "white"). The initial random vector is assumed to have mean $\hat{x}_0$ and covariance matrix P_0.

Measurements of the process are assumed to be of the form

$$y_n = H_n x_n + \varepsilon_n, \qquad n = 0,1,2,..., \tag{2.2}$$

where H_n is a known $m \times n$ matrix and ε_n is an m-dimensional random measurement error having mean zero and satisfying $E[\varepsilon(k)\varepsilon'(l)] = R_k \delta_{kl}$ where R_k is positive definite (in other words, not only is the measurement error "white", no component of each measurement error vector is a deterministic affine combination of the other components) .

The above state-space model is actually motivated by current understanding of the physical properties underlying most real-world systems. It is believed that basic randomness at the microscopic scale including electron emissions, molecular gas velocities, and elementary particle fission are basically uncorrelated processes. When their effects are observed at the macroscopic scale with, for example, a voltmeter, we obtain some average of the past microscopic effects [Luenberger, 1969].

The linear least-squares estimation problem is most conveniently tackled by formulating it as an equivalent minimum norm problem in the Hilbert space of random vectors with finite second moments [Luenberger, 1969] [Kailath et al, 2000] and inner product $\langle u,v\rangle = E[uv']$ where u,v are elements of the Hilbert space. The optimal estimate of x_n given the observations up to time n, which we will henceforth refer to as $\hat{x}_{n|n}$, is the projection of x_n onto the space spanned by the random vectors $y_0, y_1, \ldots, y_n$. The recursive Kalman filter solution is obtained as follows.

First, we assume that we have measured $y_0, y_1, \ldots, y_{n-1}$. and that the optimal one-step estimate, $\hat{x}_{n|n-1}$ which is the projection of x_n onto the space spanned by the random vectors $y_0, y_1, \ldots, y_{n-1}$, together with the error covariance matrix $P_{n|n-1} = E[(x_{n|n-1} - x_n)(x_{n|n-1} - x_n)']$, have been computed. The updated estimate can then be shown to be [Luenberger, 1969] [Kailath et al, 2000]

$$\hat{x}_{n|n} = \hat{x}_{n|n-1} + P_{n|n-1}H_n'[H_nP_{n|n-1}H_n' + R_n]^{-1}[y_n - H_n\hat{x}_{n|n-1}] \tag{2.3}$$

with associated error covariance

$$P_{n|n} = P_{n|n-1} + P_{n|n-1}H_n'[H_nP_{n|n-1}H_n' + R_n]^{-1}H_nP_{n|n-1}\,. \tag{2.4}$$

The optimal estimate of x_{n+1} given the measurements $y_0, y_1, \ldots, y_n$ is then given by

$$\hat{x}_{n+1|n} = F_n \hat{x}_{n|n} \tag{2.5}$$

with error covariance matrix

$$P_{n+1|n} = F_n P_{n|n} F_n' + Q_n \tag{2.6}$$

The above Kalman Filter solution to the linear least-squares estimation problem does not assume that the input and measurement error processes are Gaussian and requires only knowledge of the means, variances and covariances. However, when the input and measurement error processes are Gaussian, the Kalman filter solution is also the solution to the general optimal minimum mean-squared error estimation problem. In other words, the Kalman filter recursions provide the mean and covariance matrix of the Gaussian posterior density at each time step [Luenberger, 1969] [Kailath et al, 2000].

2.2 Approximate Nonlinear Filtering Using the Extended Kalman Filter

Since most practical systems are nonlinear, a lot of effort has been put into developing approximate nonlinear estimation algorithms. One of the earliest methods developed is the Extended Kalman Filter (EKF), which is based on a linearization of the state and measurement equations of the nonlinear state-space model. As an illustration of the general idea, consider the model given below.

$$x_n = f_{n-1}(x_{n-1}) + g_{n-1} w_{n-1} \tag{2.7}$$

$$y_n = h_n(x_n) + \varepsilon_n \tag{2.8}$$

Let $\hat{x}_{n|n-1}$ be the estimate of x_n given the measurements $y_0, y_1, \ldots, y_{n-1}$. Replacing $f_{n-1}(x_{n-1})$ and $h_{n-1}(x_{n-1})$ above with the first order terms from their respective Taylor series expansions about $\hat{x}_{n|n-1}$ and then applying the Kalman filter algorithm to the resulting linearized system then results in the Extended Kalman filter algorithm for the state-space model. This algorithm is given below [Kailath et al, 2000]. In the algorithm below R_k, Q_k, $\hat{x}_{n|n}$, $\hat{x}_{n|n-1}$, $P_{n|n-1}$, $P_{n|n}$ are specific to the linearized system and all have the same meanings as in the previous section.

$$\hat{x}_{n|n} = \hat{x}_{n|n-1} + P_{n|n-1}H_n'[H_nP_{n|n-1}H_n' + R_n]^{-1}[y_n - h_n\hat{x}_{n|n-1}] \tag{2.9}$$

$$P_{n|n} = P_{n|n-1} + P_{n|n-1}H_n'[H_nP_{n|n-1}H_n' + R_n]^{-1}H_nP_{n|n-1} \tag{2.10}$$

$$\hat{x}_{n+1|n} = f_n(\hat{x}_{n|n}) \tag{2.11}$$

$$P_{n+1|n} = F_nP_{n|n}F_n' + G_nQ_nG_n' \tag{2.12}$$

Where $F_n = \left.\dfrac{\partial f_n(x)}{\partial x}\right|_{x=\hat{x}_{n|n-1}}$, $H_n = \left.\dfrac{\partial h_n(x)}{\partial x}\right|_{x=\hat{x}_{n|n-1}}$, and $G_n = g_n(\hat{x}_{n|n-1})$.

Unfortunately, there are almost no useful analytical results on the performance of the EKF. A considerable amount of experimentation and adjustment is needed to get a reasonable working filter [Kailath et al, 2000].

2.3 Sequential MonteCarlo (Particle Filtering)

Importance Sampling

In perfect MonteCarlo Simulation the mean of any function of the state process at a given time conditional on the observations up to that time can approximated by a weighted sum of samples drawn from the posterior distribution. The law of large numbers then guarantees that the accuracy of this approximation will increase as the number of samples drawn increases. Unfortunately, it is often impossible to sample from the posterior density. This difficulty can be overcome by drawing samples from a known function that is easy to sample as will be illustrated below.

Let $X_k = \{x_0,...,x_k\}$ and $Y_k = \{y_1,...,y_k\}$ be the states and observations up to the present time and let $s(X_k|Y_k)$ be the proposal density that is easy to sample from. We will show below how samples from the proposal distribution can be used to approximate the mean of a function $g_k(X_k)$. The mean of $g_k(X_k)$ is given below:

$$E[g_k(X_k)] = \int g_k(X_k)p(X_k|Y_k)dX_k \tag{2.13}$$

The mean can be re-expressed in terms of the proposal distribution as follows (see [van der Merwe, 2000] for more details)

$$E[g_k(X_k)] = \frac{\int w_k(X_k)g_k(X_k)s(X_k|Y_k)dX_k}{\int w_k(X_k)s(X_k|Y_k)dX_k} \tag{2.14}$$

where the weight $w_k(X_k)$ is given by

$$w_k(X_k) = \frac{p(Y_k|X_k)p(X_k)}{s(X_k|Y_k)} \tag{2.15}$$

In (2.4) above, the mean of the function has been re-expressed as a ratio of means, each taken with respect to the proposal density. Hence by drawing samples, $X_k^{(i)} = \{x_0^{(i)},...,x_k^{(i)}\}$, from the proposal distribution we can approximate the expectation in (2.13) above by the following

$$\begin{aligned}\bar{g}_k(X_k) &= \frac{\frac{1}{N}\sum_{i=1}^{N} w_k(X_k^{(i)})g_k(X_k^{(i)})}{\frac{1}{N}\sum_{i=1}^{N} w_k(X_k^{(i)})} \\ &= \sum_{i=1}^{N} \tilde{w}_k(X_k^{(i)})g_k(X_k^{(i)})\end{aligned} \tag{2.16}$$

The estimate in (2.16 above is biased as it involves a ratio of estimates. However, if the conditions below hold:

1. $X_k^{(i)}$ is a set of independent and identically distributed samples drawn from the proposal distribution, the support of the proposal density includes the support of the posterior density and the desired mean in (2.13) exists and is finite

2. The weights $w_k(X_k)$ and the variance of $g_k(X_k)$ are bounded

the estimate in (2.16) will converge asymptotically to the true mean [van der Merwe, 2000].

Sequential Importance Sampling

The result in (2.16) does not lend itself to the sequential estimation of the mean as new observations are received. In order to compute a sequential estimate at a given time without having to modify the previously simulated states, proposal distributions of the following form can be used:

$$s(X_k|Y_k) = s(X_{k-1}|Y_{k-1})s(x_k|X_{k-1}, Y_k) \tag{2.17}$$

Under our assumptions in the previous chapter that the states correspond to a Markov process and that the observations are conditionally independent of given the states, we get the following:

$$\begin{aligned} p(X_k) &= p(x_0)\prod_{j=1}^{j=k} p(x_j|x_{j-1}) \\ p(Y_k|X_k) &= \prod_{j=1}^{j=k} p(y_j|x_j) \end{aligned} \tag{2.18}$$

By substituting (2.17) and (2.18) into (2.15) we get the following recursive expression for the importance weights:

$$w_k = w_{k-1} \frac{r(y_k | x_k) q(x_k | x_{k-1})}{s(x_k | X_{k-1}, Y_k)} \tag{2.19}$$

Hence, given an appropriate choice of $s(x_k | X_{k-1}, Y_k)$ the importance weights can be updated sequentially and hence estimate of the mean given in (2.16) can also be updated sequentially as illustrated in the algorithm below:

1. **Initialization:** set $k = 0$ and for $i = 1, \ldots, M$ particles, draw the initial states $x_0^{(i)}$ from $p(x_0 | Y_0)$ and set $w_0^{(i)} = M^{-1}$ for all i.

Set $k = 1$

2. **Importance Sampling**: For $i = 1, \ldots, M$, draw $\hat{x}_k^{(i)}$ from $s_k(x_k | X_{k-1}, Y_k)$ using the acceptance/rejection algorithm and set $\hat{x}_{0:k}^{(i)} \triangleq (\hat{x}_{0:k-1}^{(i)}, \hat{x}_k^{(i)})$. Evaluate the importance weights:

$$w_k^{(i)} = w_{k-1}^{(i-1)} \frac{r(y_k | x_k) q(x_k | x_{k-1}))}{s_k(x_k | X_{k-1}, Y_k)}$$

and then normalize them:

$$\tilde{w}_k^{(i)} = w_k^{(i)} [\sum_{j=1}^{M} w_k^{(j)}]^{-1}$$

3. **Update Mean and Variance**: Compute the marginal conditional mean and variance as follows:

$$x_{k|k} = M^{-1} \sum_{i=1}^{M} x_k^{(i)}$$

$$W_{k|k} = M^{-1} \sum_{i=1}^{M} (x_k^{(i)} (x_k^{(i)})' - x_{k|k} x'_{k|k})$$

Chapter 3

Saddle Point Approximation: Theory

We will first discuss how the Laplace method can be used to approximate asymptotic integrals with vanishingly small approximation error. Unfortunately the assumptions on the form of the integrand may not necessarily apply in the nonlinear filtering context. Using a method analogous to the Laplace method for asymptotic integrals, we obtain bounds on the integral and approximation error for the integrals of what we will refer to as exponentially concentrated functions. These results can then be applied in understanding the error performance when the Laplace approximation is used in the nonlinear filtering context.

3.1 The Laplace Method for Approximating Integrals

We will focus our attention on integrals over real intervals where both the integration interval and the integrand may depend on some parameter t, as illustrated in Equation 3.1 below. The Laplace method is usually used to investigate the asymptotic behavior of such integrals as $t \to \infty$.

$$F = \int_{-\infty}^{\infty} g(x,t)dx \qquad (t \to \infty) \tag{3.1}$$

In this section we will first present a heuristic argument for determining the functional form of the Laplace approximation for a very general case. Finally we will formally state and prove the result. The proof will follow the outline provided in [DeBruijn, 1981 Ch.4] while filling in details omitted in that text.

Heuristic Derivation of the Laplace Approximation

The key idea behind the Laplace method is that if there is an interval I such that the integral over the complement of that interval is small compared to the integral over I, we can then try to approximate the integrand by simpler functions throughout I. To make this a little more specific we will consider integrals of the form:

$$F = \int_{-\infty}^{\infty} exp\{th(x)\}dx \ . \tag{3.2}$$

where t is a positive real constant and $h(x)$ is a real continuous function. Furthermore, we will assume that $h(x)$ has a unique global maximum at the point $x = x_0$ and that its first and second derivatives exist in some interval. Since $h(x)$ has a unique global maximum at the point x_0, the value of the integral for large values of t will be dominated by the behavior of $h(x)$ near its maximum. Therefore there will be a small interval I around the maximum such that the integral of $exp\{th(x)\}$ over the complement of I is small compared to the integral over I.

Further assume that the Taylor's series expansion of $h(x)$ about the point x_0 converges in I:

$$h(x) = h(x_0) + h'(x_0)(x - x_0) + \frac{1}{2}h''(x_0)(x - x_0)^2 + \ higher\ order\ terms \tag{3.3}$$

Since $h(x_0)$ is a global maximum, $h'(x_0) = 0$. Therefore $h(x)$ may be approximated by simpler functions as

$$h(x) \approx \hat{h}(x) = h(x_0) + \frac{1}{2}h''(x_0)(x - x_0)^2 \ , \quad for\ x \in I \tag{3.4}$$

Since $\left(\int_{-\infty}^{\infty}-\int_{I}\right)exp\{th(x)\}dx$ is small compared to $\int_{I}exp\{th(x)\}dx$ we can approximate the integral in (3.2) by the integral over I, which in turn can be approximated by the integral of the simpler function over I. This integral is then approximated by the integral over the entire interval (which is easily evaluated) yielding the Laplace approximation. These steps are illustrated below:

$$\int_{-\infty}^{\infty} exp\{th(x)\}dx \approx \int_{I} exp\{th(x)\}dx$$

$$\approx \int_{I} exp\{th(x_0)+\frac{1}{2}th''(x_0)(x-x_0)^2\}dx$$

$$\approx exp\{th(x_0)\}\int_{-\infty}^{\infty} exp\{\frac{1}{2}th''(x_0)(x-x_0)^2\}dx$$

$$=(2\pi)^{1/2}(-th''(x_0))^{-1/2}exp\{th(x_0)\}$$

So the Laplace approximation to the integral in (3.2) is

$$\hat{F}(t)=(2\pi)^{1/2}(-th''(x_0))^{-1/2}exp\{th(x_0)\} \tag{3.5}$$

Rigorous Statement and Proof of the Result

In this section we will establish bounds on the integral in (3.2). Before we state the proof of the main result we will prove a lemma that will be used in the proof.

Lemma 3.1 *Let $h(x)$ be a real and continuous function. If*

i. $h(x)$ has a unique global maximum at $x=x_0$

ii. $h'(x)$ exists in some neighborhood of x_0, $h''(x_0)$ exists and $h''(x_0) < 0$

Then, given positive ε, we can determine $\delta \geq 0$ such that

$$\left| h(x) - h(x_0) - \frac{1}{2}(x-x_0)^2 h''(x_0) \right| \leq \varepsilon (x-x_0)^2 \quad , for \ |x-x_0| \leq \delta$$

Proof Let $\varphi(x) = h(x) - h(x_0) - \frac{1}{2}(x-x_0)^2 h''(x_0)$. Since $h(x)$ is maximal at $x = x_0$ we infer that $h'(x_0) = 0$. Consequently $\varphi(x_0) = \varphi'(x_0) = \varphi''(x_0) = 0$. $\varphi''(x_0) = 0$ implies that $(x-x_0)^{-1}(\varphi'(x) - \varphi'(x_0)) \to 0$ when $x \to x_0$. Since $\varphi'(x_0) = 0$ this means that $\varphi'(x) = o(x - x_0) \ ,(x \to x_0)$. Applying the mean value theorem [Rudin, 1976 Theorem 5.5] to $\varphi(x)$ we see that $\varphi(x) - \varphi(x_0) = (x-x_0)^{-1}\varphi'(x_0 + \theta(x-x_0))$ for some $0 < \theta < 1$. So $\varphi(x) = (x-x_0)o(\theta(x-x_0)) = o((x-x_0)^2) \ ,(x \to x_0)$. Since $\varphi(x) = o((x-x_0)^2)$, for a given positive ε we can find $\delta \geq 0$ such that $|\varphi(x)| \leq \varepsilon (x-x_0)^2$ for $|x - x_0| \leq \delta$ which proves the lemma.

We will now state the main result below.

Theorem 3.2 (Bounds on the Integral) *Let $F(t) = \int_{-\infty}^{\infty} \exp\{th(x)\}dx$, where $h(x)$ is a real and continuous function and $t \geq 1$. If*

i. $h(x)$ has a unique global maximum $H_{\max}$ at $x = x_0$

ii. there exist numbers $A < H_{\max}$ and a such that $h(x) \leq A$ if $|x - x_0| \geq a$

iii. $F(t)$ exists for $t = 1$ and for all sufficiently large values of t

iv. $h'(x)$ exists in some neighborhood of x_0, $h''(x_0)$ exists and $h''(x_0) < 0$

Then for any ε such that $0 < 3\varepsilon < |h''(x_0)|$, there exists a positive real number T such that for $t \geq T$ the following inequality holds:

$$e^{th(x_0)}(2\pi)^{\frac{1}{2}}(-h''(x_0) + 3\varepsilon)^{-\frac{1}{2}} t^{-\frac{1}{2}} < F(t) < e^{th(x_0)}(2\pi)^{\frac{1}{2}}(-h''(x_0) - 3\varepsilon)^{-\frac{1}{2}} t^{-\frac{1}{2}}$$

Since ε is arbitrary, for sufficiently large t we have the following Laplace approximation:

$$F(t) \approx \hat{F}(t) = (2\pi)^{\frac{1}{2}}(-th''(x_0))^{-\frac{1}{2}} exp\{th(x_0)\}$$

Proof Applying Lemma 3.1 we see that for any positive ε we can find $\delta \geq 0$ such that for $|x-x_0| \leq \delta$ the following inequality holds:

$$\left| h(x)-h(x_0)-\frac{1}{2}(x-x_0)^2 h''(x_0) \right| \leq \varepsilon (x-x_0)^2 .$$

Therefore the following inequalities hold:

$$\int_{x_0-\delta}^{x_0+\delta} e^{\frac{1}{2}(x-x_0)^2 t(h''(x_0)-2\varepsilon)} dx < \int_{x_0-\delta}^{x_0+\delta} e^{t(h(x)-h(x_0))} dx < \int_{x_0-\delta}^{x_0+\delta} e^{\frac{1}{2}(x-x_0)^2 t(h''(x_0)+2\varepsilon)} dx \tag{3.6}$$

As the integrals over $(-\infty,\infty)$ of the leftmost and rightmost integrands will be used later in the proof to establish bounds on $F(t)$, we will limit our choice of ε to the range $0 < 3\varepsilon < |h''(x_0)|$. This will ensure that the exponent of the integrand in the rightmost integral in (3.6) is negative.

Let D_1, D_2 and D_3 be the amounts by which the integrals above differ from the corresponding integrals over $(-\infty,\infty)$. That is:

$$D_1 = \left(\int_{-\infty}^{x_0-\delta} + \int_{x_0+\delta}^{\infty} \right) e^{t(h(x)-h(x_0))} dx$$

$$D_2 = \left(\int_{-\infty}^{x_0-\delta} + \int_{x_0+\delta}^{\infty} \right) e^{\frac{1}{2}(x-x_0)^2 t(h''(x_0)+2\varepsilon)} dx$$

$$D_3 = \left(\int_{-\infty}^{x_0-\delta} + \int_{x_0+\delta}^{\infty} \right) e^{\frac{1}{2}(x-x_0)^2 t(h''(x_0)-2\varepsilon)} dx$$

Adding $D_1 + D_2$ to both sides of the inequality on the right and $D_1 + D_3$ to both sides of the inequality on the left in (3.6) above we get the following inequalities for the integrals over $(-\infty, \infty)$:

$$\int_{-\infty}^{+\infty} e^{\frac{1}{2}(x-x_0)^2 t(h''(x_0)-2\varepsilon)} dx + D_1 - D_3 < e^{-th(x_0)} F(t) < \int_{-\infty}^{+\infty} e^{\frac{1}{2}(x-x_0)^2 t(h''(x_0)+2\varepsilon)} dx + D_1 - D_2 \qquad (3.7)$$

Since $\int_{-\infty}^{+\infty} e^{\frac{1}{2}(x-x_0)^2 t(h''(x_0)\pm 2\varepsilon)} dx = (2\pi)^{\frac{1}{2}} (-h''(x_0) \mp 2\varepsilon)^{-\frac{1}{2}} t^{-\frac{1}{2}}$ the inequalities in (3.7) become:

$$(2\pi)^{\frac{1}{2}} (-h''(x_0) + 2\varepsilon)^{-\frac{1}{2}} t^{-\frac{1}{2}} + D_1 - D_3 < e^{-th(x_0)} F(t) \qquad (3.8)$$

$$e^{-th(x_0)} F(t) < (2\pi)^{\frac{1}{2}} (-h''(x_0) - 2\varepsilon)^{-\frac{1}{2}} t^{-\frac{1}{2}} + D_1 - D_2 \qquad (3.9)$$

We will now establish bounds on D_1, D_2 and D_3. It follows from our assumptions that for any $\delta \geq 0$ there exists a positive number $\eta_1(\delta)$ such that $h(x) - h(x_0) \leq -\eta(\delta)$ when $|x - x_0| \geq \delta$. If $\delta \geq a$ then assumption (ii) in the statement of Theorem 3.2 implies that $\eta_1(\delta) = H_{\max} - A$. If $\delta \leq a$, let B be the maximum of the continuous function $h(x)$ in the interval $\delta \leq |x - x_0| \leq a$. $B > A$ then implies that $\eta_1(\delta) = H_{\max} - B$ otherwise $\eta_1(\delta) = H_{\max} - A$.

Rewriting $t[h(x) - h(x_0)]$ as $t[h(x) - h(x_0)] = (t-1)[h(x) - h(x_0)] + [h(x) - h(x_0)]$ and applying the inequality $h(x) - h(x_0) \leq -\eta_1(\delta)$ to the second term on the right hand side we get the following inequality:

$t(h(x) - h(x_0)) \leq -(t-1)\eta_1(\delta) + h(x) - h(x_0)$ for $t > 1$.

The inequality above implies that:

$$D_1 \le \left(\int_{-\infty}^{-\delta} + \int_{\delta}^{\infty} \right) \exp\{-(t-1)\eta_1(\delta) + h(x) - h(x_0)\} dx$$

$$\le \int_{-\infty}^{\infty} \exp\{-(t-1)\eta_1(\delta) + h(x) - h(x_0)\} dx$$

Therefore

$D_1 \le K_1 \exp\{-t\eta_1(\delta)\}$ where $K_1 = \exp\{\eta_1(\delta) - h(x_0)\} \int_{-\infty}^{\infty} \exp\{h(x)\} dx$.

The functions $\frac{1}{2}(x-x_0)^2 t(h''(x_0)+2\varepsilon)$ and $\frac{1}{2}(x-x_0)^2 t(h''(x_0)-2\varepsilon)$ each have a unique global maximum value of 0 at $x = x_0$. Furthermore, since the constants $t(h''(x_0)+2\varepsilon)$ and $t(h''(x_0)-2\varepsilon)$ are both negative, the functions are both monotonically decreasing for $|x-x_0|>0$ and $\exp[\frac{1}{2}(x-x_0)^2 t(h''(x_0) \pm 2\varepsilon)]$ is a Gaussian function and hence its integral exists. As a result we see that these functions satisfy the same assumptions (i)-(iii) which we made on $h(x)$. Hence we can establish the following bounds for D_2 and D_3 using the same argument we used above for D_1:

$D_2 \le K_2 \exp\{-t\eta_2(\delta)\}$ where $K_2 = \exp\{\eta_2(\delta)\} \int_{-\infty}^{\infty} \exp\{\frac{1}{2}(x-x_0)^2 (h''(x_0)+2\varepsilon)\} dx$ and $\eta_2(\delta) = \frac{1}{2}\delta^2(-h''(x_0)-2\varepsilon)$

$D_3 \le K_3 \exp\{-t\eta_3(\delta)\}$ where $K_3 = \exp\{\eta_3(\delta)\} \int_{-\infty}^{\infty} \exp\{\frac{1}{2}(x-x_0)^2 (h''(x_0)-2\varepsilon)\} dx$ and $\eta_2(\delta) = \frac{1}{2}\delta^2(-h''(x_0)+2\varepsilon)$.

Since D_1, D_2 and D_3 are positive, the following inequalities hold:

$D_1 - D_2 < D_1 \le K_1 \exp\{-t\eta_1(\delta)\}$

$D_1 - D_3 > -D_3 \ge -K_3 \exp\{-t\eta_3(\delta)\}$

Plugging the above inequalities into (3.8) and (3.9) we get the following inequalities:

$$(2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}} - K_3\exp\{-t\eta_3(\delta)\} < e^{-th(x_0)}F(t) \tag{3.10}$$

$$e^{-th(x_0)}F(t) < (2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}} + K_1\exp\{-t\eta_1(\delta)\} \tag{3.11}$$

As t increases $K_3\exp\{-t\eta_3(\delta)\}$ will decay much faster than the difference between $(2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}}$ and $(2\pi)^{\frac{1}{2}}(-h''(x_0)+3\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}}$ (which decays as $t^{-\frac{1}{2}}$). Therefore when t is sufficiently large $(2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}} - K_3\exp\{-t\eta_3(\delta)\}$ will be greater than $(2\pi)^{\frac{1}{2}}(-h''(x_0)+3\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}}$. This together with (3.1) implies that:

$$(2\pi)^{\frac{1}{2}}(-h''(x_0)+3\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}} < e^{-th(x_0)}F(t) \tag{3.12}$$

Similarly, as t increases $K_1\exp\{-t\eta_1(\delta)\}$ will decay much faster than the difference between $(2\pi)^{\frac{1}{2}}(-h''(x_0)-3\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}}$ and $(2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}}$ (which decays as $t^{-\frac{1}{2}}$). Therefore when t is sufficiently large $(2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}} + K_1\exp\{-t\eta_1(\delta)\}$ will be less than $(2\pi)^{\frac{1}{2}}(-h''(x_0)-3\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}}$. This together with (3.11) implies that:

$$e^{-th(x_0)}F(t) < (2\pi)^{\frac{1}{2}}(-h''(x_0)-3\varepsilon)^{-\frac{1}{2}}t^{-\frac{1}{2}} \tag{3.13}$$

and completes the proof.

Theorem 3.3 (Direct Evaluation of the Laplace Approximation Error)

$F(t) = \hat{F}(t) + Kt^{-1}$ *for some positive constant* K.

Proof

The Taylor's series expansion of $h(x)$ about the point x_0 is given by:

$$h(x) = h(x_0) - \frac{1}{2\sigma^2}(x-x_0)^2 + \frac{1}{6}H_3(x-x_0)^3 + \frac{1}{24}H_4(x-x_0)^4 + \textit{higher order terms}$$

where, $\sigma^2 = -\,{}^{1}\!/_{h''(x_0)}$, and $H_k = (d/dx)^k h(x)\big|_{x=x_0}$. The integral can then be expressed as

$$\int e^{th(x)}dx = e^{th(x_0)} \int e^{-\frac{1}{2t^{-1}\sigma^2}(x-x_0)^2} \prod_{k\geq 3} e^{\frac{t}{k!}H_k(x-x_0)^k} dx \tag{3.14}$$

Expanding each term of the form $e^{\frac{1}{k!}H_k(x-x_0)^k}$ in its Taylor series expansion about 0 we get the following:

$$\prod_{k\geq 3} e^{\frac{t}{k!}H_k(x-x_0)^k} = \prod_{k\geq 3}\left(1 + \frac{t}{k!}H_k(x-x_0)^k + \frac{t^2}{2\bullet(k!)^2}H_k^2(x-x_0)^{2k} + \frac{t^3}{6\bullet(k!)^3}H_k^3(x-x_0)^{3k} + \cdots\right)$$

$$= 1 + \sum_{k\geq 3} A_k(x-x_0)^k$$

for some constants A_k. Since integrals of the form $\int (x-x_0)^k e^{-\frac{1}{2t^{-1}\sigma^2}(x-x_0)^2} dx$ equal zero for odd numbered integers k, only the even powers in the sum above contribute to the integral in (3.14). Therefore we have the following

$$\int e^{th(x)}dx = e^{th(x_0)}\left[\int e^{-\frac{1}{2t^{-1}\sigma^2}(x-x_0)^2}dx + \sum_{k\geq 2} A_{2k}\int (x-x_0)^{2k} e^{-\frac{1}{2t^{-1}\sigma^2}(x-x_0)^2}dx\right]$$

Using known results on the central moments of Gaussian distributions to evaluate the first few integrals in the sum above yields the following result (see the appendix of [Tierney and Kadane, 1986]):

$$\int e^{th(x)}dx = (2\pi\sigma t^{-1})^{\frac{1}{2}} e^{th(x_0)} \left(1 + B_1 t^{-1} + B_2 t^{-2} + O(t^{-3})\right)$$

where

$$B_1 = \frac{1}{8}\sigma^4 H_4 + \frac{5}{24}\sigma^6 H_3^2$$

$$B_2 = \frac{1}{48}\sigma^6 H_6 + \frac{35}{384}\sigma^8 H_4^2 + \frac{7}{48}\sigma^8 H_3 H_5 + \frac{35}{64}\sigma^{10} H_3^2 H_4 + \frac{385}{1152}\sigma^{12} H_3^4$$

3.2 New Results Based on the Laplace Method for Approximating Integrals

Laplace Approximation for Exponentially Concentrated Functions

In the previous section we presented a heuristic argument to derive the Laplace approximation for integrals of functions of the form $exp\{th(x)\}$ for large positive real values of t (which must be greater than 1) and a properly behaved function $h(x)$. Our eventual goal is to apply this method in approximating the one-step update integral in the BCK equations for the general filtering problem given in (1.8). Unfortunately, the integrands of interest in the general filtering problem cannot necessarily be expressed in the form $exp\{th(x)\}$.

An obvious first step towards our eventual goal would be to examine approximations to integrals of the following form

$$F = \int_{-\infty}^{\infty} f(x)dx = \int_{-\infty}^{\infty} exp\{h(x)\}dx \ .$$

We could apply a similar heuristic argument as the one used at the beginning of the previous section to arrive at the following approximation:

$$\hat{F} = (2\pi)^{1/2}(-h''(x_0))^{-1/2} exp\{h(x_0)\} \tag{3.15}$$

However, the results establishing bounds on the integral and asymptotic error performance of the approximation presented in the previous section will not necessarily hold. In order to use the approximation in (3.15) to construct nonlinear filtering algorithms it is essential to have a good intuitive understanding of how the behavior of the integrand will affect the error performance of the approximation in (3.15).

By focusing on the integrals of functions which have the following properties (we will refer to such functions as being *exponentially concentrated*)

- existence of a unique global maximum
- the integral of the function over the complement of a neighborhood of the maximum decays exponentially as the radius of the neighborhood increases

bounds on the approximation in (3.15) can be established using a method analogous to that used in the proof of Theorem 3.2. In particular, we show that for exponentially concentrated functions, the following bound on the integral holds (see the proof of Theorem 3.5 in the next section for more details and a more rigorous statement)

$$(2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}} - K_2(t)e^{-t\delta} < e^{-h(x_0)}F < (2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}} + K_1(t)e^{-t\delta}$$

In the above, ε and δ together are a measure of how well $h(x)$ can be approximated by a quadratic in a neighborhood of the global maximum at $x = x_0$. For a given upper limit on the error of the approximation (parameterized by ε), δ is the largest neighborhood over which that upper limit is satisfied. We then take $\hat{F} = (2\pi)^{\frac{1}{2}}(-h''(x_0))^{-\frac{1}{2}}\exp(h(x_0))$, a value that also lies between the upper and lower bounds in the inequality above, to be the Laplace approximation.

If $h(x)$ is exactly quadratic (for example if $f(x)$ is a Gaussian function) then the approximation will be exact. For other cases, we show in Corollary 3.6 that the relative error of the approximation, $\xi = \left|F - \hat{F}\right| / F$, satisfies the following inequality

$$\xi \leq \xi_1 + \xi_2 + \xi_3$$

The quantities $\xi_1, \xi_2,$ and ξ_3 help us to understand the factors determining how well the approximation works. The first quantity ξ_1 is a very small value that decreases as ε decreases and hence depends on how well $h(x)$ can be approximated by a quadratic in a neighborhood of the global maximum. The second quantity ξ_2 is a measure of the proportion of the total area under $f(x)$ that is concentrated about the global maximum. The more of the area under $f(x)$ that is concentrated around the global maximum, the smaller ξ_2 will be. Finally, ξ_3 is a measure of how accurately $h(x)$ can be approximated by a quadratic in a neighborhood of the global maximum. The better $h(x)$ is approximated by a quadratic around x_0 then the smaller ξ_3 will be.

Application to approximating a marginal probability density given the joint density

Given the joint density $p(x,y)$ of two random variables x and y, we can express the probability density $p(y)$ of y at a given point $y = y^*$ by marginalizing the joint density as shown below:

$$p(y^*) = \int_{-\infty}^{\infty} p(x, y^*) dx$$

Lemma 3.7 establishes conditions under which the integrand in the above equation is exponentially concentrated. Under these conditions, we can then apply the result of Theorem 3.5 to obtain the following bound on the probability density at a fixed point (see the proofs of Lemma 3.7 and Corollary 3.8 for more details):

$$\hat{p}_-(y^*) - K_2(t)e^{-t\delta} < e^{-h(x_0)}p(y^*) < \hat{p}_+(y^*) + K_1(t)e^{-t\delta}$$

where $\hat{p}_+(y^*) = (2\pi)^{1/2}(-h''(x_0(y^*)) - 2\varepsilon)^{-1/2}$ and $\hat{p}_-(y^*) = (2\pi)^{1/2}(-h''(x_0(y^*)) + 2\varepsilon)^{-1/2}$. The Laplace approximation in this case is analogolously:

$$\hat{p}(y^*) = (2\pi)^{1/2}(-h''(x_0(y^*))^{-1/2}p(x_0(y^*), y^*) .$$

Furthermore, observations about the error performance, which are analogous to those made for the general case earlier, also hold here.

Application to the nonlinear filtering problem

In chapter 1 we saw that the BCK equations (repeated below) provide a framework for sequentially updating the posterior density.

One step ahead prediction density:

$$p(x_k|Y_{k-1}) = \int q(x_k|x_{k-1})p(x_{k-1}|Y_{k-1})dx_{k-1}$$

Filtering density:

$$p(x_k|Y_k) = \frac{r(y_k|x_k)p(x_k|Y_{k-1})}{p(y_k|Y_{k-1})}$$

Typically, the transition density $q(x_k|x_{k-1})$ and the likelihood $r(y_k|x_k)$ are already known and the numerator of the expression for the filtering density is just a scaling constant.

The one-step prediction density can be approximated using the laplace approximation. The approximation to the filtering density will then be proportional to the product of the Laplace approximation to the one-step density and the data likelihood $r(y_k|x_k)$. We outline below a filtering algorithm based on this approach:

Initial Step

Put $\hat{p}(x_0 | Y_0) = p(x_0)$

Step k

1. Put $h(x_{k-1}) = \log q(x_k | x_{k-1}) + \log \hat{p}(x_{k-1} | Y_{k-1})$

2. Compute the value of x_{k-1} that maximizes $h(x_{k-1})$. Call this $\hat{x}_{k-1}$.

3. Approximate the one-step prediction density as

$$\hat{p}(x_k | Y_{k-1}) = (2\pi)^{1/2}(-h_k''(\hat{x}_{k-1}))^{-1/2} q(x_k | \hat{x}_{k-1}) \hat{p}(\hat{x}_{k-1} | Y_{k-1})$$

4. The approximation to the posterior density is then

$$\hat{p}(x_k | Y_k) \propto r(y_k | x_k) \hat{p}(x_k | Y_{k-1})$$

Step $k = 1$ is the first time the Laplace approximation is used to estimate the one-step prediction density. If the integrand of the integral for the one-step prediction density at this time point, $f(x_0) = q(x^* | x_0) p(x_0)$, satisfies the conditions in Lemma 3.7 the results we have derived so far will apply. As was mentioned earlier the relative error of the approximation to $p(x_1 | Y_0)$ will be determined by $q(x_1 | x_0)$ and the initial state density $p(x_0)$. Specifically, the relative error of the approximation at the point $x_1 = x^*$ will be small if $h(x_0) = \log q(x^* | x_0) + \log p(x_0)$ is well approximated by a quadratic and if a significant proportion of the total area under the function $f(x_0) = q(x^* | x_0) p(x_0)$ is contained in a small neighborhood of its global maximum at $\hat{x}_0$.

At future steps k the error will depend on the effect of the data likelihood at the previous step $r(y_{k-1} | x_{k-1})$ as well as the transition density.

To illustrate these points we consider one of the state-space models mentioned in chapter 1. This model can describe, for example, a state-space system with stochastic point process observations as we will see in the next chapter (it can be shown that for the model discussed in the next chapter the conditions in Lemma 3.7 hold allowing us to apply the results developed in this chapter that give us bounds on the relative error and an intuitive understanding of the factors determining the size of the relative error).

$$x_n = F x_{n-1} + w_n$$
$$y_n \sim Dist(x_n)$$

In the above, the noise process w_n consists of independent, zero-mean Gaussian random variables at each time point with variance σ_w^2 and the initial state is drawn from a zero-mean Gaussian distribution with variance σ_0^2. As a result $q(x_1|x_0)$ for fixed $x_1 = x^*$ and taken as a function of x_0 will itself be Gaussian with mean x^* and variance σ_w^2. The function $f(x_0) = q(x^*|x_0)p(x_0)$, being the product of two Gaussian functions, will also be Gaussian. Therefore, at step $k=1$ the Laplace approximation to the one-step density will be exact.

At the next step the integrand is now the product $f(x_1) = q(x^*|x_1)p(x_1|Y_0)r(y_0|x_0)$. If the variances σ_w^2 and σ_0^2 are both small the supports of $q(x^*|x_1)$ and $p(x_1|Y_0)$ (which are both Gaussian) will each be very narrow. As a result, the support of $f(x_1)$ will be narrow and most of the area under $f(x_1)$ will be concentrated around its global maximum. Consequently the contribution of the ξ_2 term to the relative error of the approximation to the one-step prediction density at the point $x_2 = x^*$ will be small. Since both $q(x^*|x_1)$ and $p(x_1|Y_0)$ are Gaussian functions, it is the data likelihood from the previous step $r(y_0|x_0)$ that will determine how well $h(x_1) = \log f(x_1)$ is approximated by a quadratic. This in turn will determine how big a contribution the ξ_1 and ξ_3 terms will make to the relative error. A similar analysis can be carried out at subsequent steps.

3.3 Proofs of New Results Based on the Laplace Method for Approximating Integrals

Definition 3.4 (Scalar Case) A function $f(x)$ will be said to be exponentially concentrated if the following hold

i. $f(x)$ is a real and continuous positive function which has a unique global maximum at some point $x = x_0$ (in other words $f(x) < f(x_0)$ for all $x \neq x_0$)

ii. There exist positive real numbers t and $K(t)$ such that the integral of $f(x)$ over the complement of any neighborhood of x_0 having radius δ satisfies:

$$\left(\int_{-\infty}^{x_0-\delta} + \int_{x_0+\delta}^{\infty} \right) f(x)dx \le K(t)e^{-t\delta}, \quad \forall \delta > 0.$$

Theorem 3.5: Bounds on the Integral

Let $F = \int_{-\infty}^{\infty} f(x)dx$ and $h(x)$ be the natural logarithm of $f(x)$. Assume $f(x)$ is exponentially concentrated with its unique global maximum at $x = x_0$, $h'(x)$ exists in some neighborhood of x_0, $h''(x_0)$ exists and $h''(x_0) < 0$.

For any ε such that $0 < 3\varepsilon < |h''(x_0)|$, there exist positive real numbers δ, t (which is independent of ε), $K_1(t)$ and $K_2(t)$ such that the following inequality holds:

$$(2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}} - K_2(t)e^{-t\delta} < e^{-h(x_0)}F < (2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}} + K_1(t)e^{-t\delta}.$$

We will take $\hat{F} = (2\pi)^{\frac{1}{2}}(-h''(x_0))^{-\frac{1}{2}}\exp(h(x_0))$ to be the Laplace approximation.

Proof Applying Lemma 3.1 we see that for any positive ε we can find $\delta \geq 0$ such that for $|x - x_0| \leq \delta$ the following inequality holds:

$$\left| h(x) - h(x_0) - \frac{1}{2}(x - x_0)^2 h''(x_0) \right| \leq \varepsilon (x - x_0)^2$$

Therefore the following inequalities hold:

$$\int_{x_0-\delta}^{x_0+\delta} e^{\frac{1}{2}(x-x_0)^2(h''(x_0)-2\varepsilon)} dx < \int_{x_0-\delta}^{x_0+\delta} e^{(h(x)-h(x_0))} dx < \int_{x_0-\delta}^{x_0+\delta} e^{\frac{1}{2}(x-x_0)^2(h''(x_0)+2\varepsilon)} dx \qquad (3.16)$$

As the integrals over $(-\infty, \infty)$ of the leftmost and rightmost integrands will be used later in the proof to establish bounds on F, we will limit our choice of ε to the range $0 < 3\varepsilon < |h''(x_0)|$. This will ensure that the exponent of the integrand in the rightmost integral in (3.16) is negative.

Let D_1, D_2 and D_3 be the amounts by which the integrals above differ from the corresponding integrals over $(-\infty, \infty)$. That is:

$$D_1 = \left(\int_{-\infty}^{x_0-\delta} + \int_{x_0+\delta}^{\infty} \right) e^{(h(x)-h(x_0))} dx$$

$$D_2 = \left(\int_{-\infty}^{x_0-\delta} + \int_{x_0+\delta}^{\infty} \right) e^{\frac{1}{2}(x-x_0)^2(h''(x_0)+2\varepsilon)} dx$$

$$D_3 = \left(\int_{-\infty}^{x_0-\delta} + \int_{x_0+\delta}^{\infty} \right) e^{\frac{1}{2}(x-x_0)^2(h''(x_0)-2\varepsilon)} dx$$

Adding $D_1 + D_2$, $D_1 + D_3$ to both sides of the inequality on the right and left respectively in (3.16) above we get the following inequalities for the integrals over $(-\infty, \infty)$:

$$\int_{-\infty}^{+\infty} e^{\frac{1}{2}(x-x_0)^2(h''(x_0)-2\varepsilon)}dx + \mathrm{D}_1 - \mathrm{D}_3 < e^{-h(x_0)}F < \int_{-\infty}^{+\infty} e^{\frac{1}{2}(x-x_0)^2(h''(x_0)+2\varepsilon)}dx + \mathrm{D}_1 - \mathrm{D}_2 \tag{3.17}$$

Since $\int_{-\infty}^{+\infty} e^{\frac{1}{2}(x-x_0)^2(h''(x_0)\pm 2\varepsilon)}dx = (2\pi)^{\frac{1}{2}}(-h''(x_0)\mp 2\varepsilon)^{-\frac{1}{2}}$ the inequalities in (3.16) become:

$$(2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}} + \mathrm{D}_1 - \mathrm{D}_3 < e^{-h(x_0)}F \tag{3.18}$$

$$e^{-h(x_0)}F < (2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}} + \mathrm{D}_1 - \mathrm{D}_2 \tag{3.19}$$

We will now establish bounds on D_1, D_2 and D_3. From our assumptions there exist positive real numbers t and $K_1(t)$ such that the following inequality holds:

$$\mathrm{D}_1 \le K_1(t)e^{-t\delta}$$

Let v_1 and v_2 be zero-mean Gaussian random variables with variance $(-h''(x_0)-2\varepsilon)^{-1}$ and $(-h''(x_0)+2\varepsilon)^{-1}$ respectively. We can therefore re-express D_2 and D_3 as follows:

$$\mathrm{D}_2 = (2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}}\Pr(|v_1|>\delta)$$

$$\mathrm{D}_3 = (2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}}\Pr(|v_2|>\delta)$$

For any positive t an application of Chernoff's bound to $\Pr(|v_1|>\delta)$ and $\Pr(|v_2|>\delta)$ (see for example [Ross, 1996] or [Laha and Rohatgi, 1979]) will then give us the following bounds on D_2 and D_3:

$$\mathrm{D}_2 \le (2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}}M_1(t)e^{-t\delta}$$

$$D_3 \leq (2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}} M_2(t)e^{-t\delta}$$

In the above inequalities $M_1(t)$ and $M_2(t)$ are the moment generating functions of the random variables $|v_1|$ and $|v_2|$ respectively.

Now since D_1, D_2 and D_3 are positive, the following inequalities hold:

$$D_1 - D_2 < D_1 \leq K_1(t)e^{-t\delta}$$

$$D_1 - D_3 > -D_3 \geq -(2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}} M_2(t)e^{-t\delta} = -K_2(t)e^{-t\delta}.$$

Plugging the above inequalities into (3.18) and (3.19) we get the following inequalities:

$$(2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}} - K_2(t)e^{-t\delta} < e^{-h(x_0)}F \tag{3.20}$$

$$e^{-h(x_0)}F < (2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}} + K_1(t)e^{-t\delta} \tag{3.21}$$

thus completing the proof.

Corollary 3.6 (Relative Error of the Approximation)

Assume that the assumptions in Theorem 3.5 hold. Let $\xi = \left|F-\hat{F}\right|/F$ be the relative error of the Laplace approximation. *Then the relative error satisfies:*

$$\xi \leq \xi_1 + \xi_2 + \xi_3.$$

In the above ξ_1 *is a small quantity that decreases as* ε *decreases.* ξ_2 *is a measure of how much of the area under* $f(x)$ *is concentrated around the global maximum. The more* *of the area under $f(x)$ that is concentrated around the global maximum, the smaller ξ_2 will be. Finally, ξ_3 is a measure of how accurately $h(x)$ can be approximated by a quadratic in a neighborhood of the global maximum. If $h(x)$ is well approximated by a quadratic then ξ_3 will be very small.*

Proof Clearly $\hat{F}(t)$ also satisfies the following inequalities :

$$(2\pi)^{\frac{1}{2}}(-h''(x_0)+2\varepsilon)^{-\frac{1}{2}} - K_2(t)e^{-t\delta} < e^{-h(x_0)}\hat{F}$$

$$e^{-h(x_0)}\hat{F} < (2\pi)^{\frac{1}{2}}(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}} + K_1(t)e^{-t\delta}$$

In equations (3.20) and (3.21) of the proof of Theorem 3.5 we see that F has the same upper and lower bounds. The fact that F and $\hat{F}$ share the same upper and lower bound implies that the absolute approximation error is less than the gap between the upper and lower bound. That is

$$e^{-h(x_0)}\left|F-\hat{F}\right| < (2\pi)^{\frac{1}{2}}\left[(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}} - (-h''(x_0)+2\varepsilon)^{-\frac{1}{2}}\right] + \left[(K_1(t)+K_2(t))e^{-t\delta}\right]$$

Dividing both sides of the above inequality by F and multiplying by $e^{h(x_0)}$ gives us the following

$$\xi \le \xi_1 + \xi_2 + \xi_3$$

where

$$\xi_1 = (2\pi)^{\frac{1}{2}} F^{-1} e^{h(x_0)}\left[(-h''(x_0)-2\varepsilon)^{-\frac{1}{2}} - (-h''(x_0)+2\varepsilon)^{-\frac{1}{2}}\right]$$

$$\xi_2 = F^{-1}e^{h(x_0)}K_1(t)e^{-t\delta}$$

$\xi_3 = F^{-1} e^{h(x_0)} K_2(t) e^{-t\delta}$

In the course of the proof of theorem 3.5 we saw that D_1, the integral of $f(x)$ (normalized by $e^{h(x_0)}$) in a complement of a neighborhood (having radius δ) of the global maximum at $x = x_0$, is less than $K_1(t)e^{-t\delta}$. So ξ_2 is an upper bound to the proportion of the area under $f(x)$ that lies outside this neighborhood (having radius δ) of the global maximum. The more of the area under $f(x)$ that is concentrated around the global maximum, the smaller ξ_2 will be for any value of δ.

We also saw that $e^{h(x_0)} K_2(t) e^{-t\delta}$ is an upper bound to D_3 where D_3 is given by:

$$D_3 = \left(\int_{-\infty}^{x_0-\delta} + \int_{x_0+\delta}^{\infty} \right) e^{\frac{1}{2}(x-x_0)^2 (h''(x_0)-2\varepsilon)} dx.$$

Now ε and δ together are a measure of how well $h(x)$ can be approximated by a quadratic in a neighborhood of the global maximum. For a given upper limit on the error of the approximation (parameterized by ε), δ is the largest neighborhood over which that upper limit is satisfied. Hence, if $h(x)$ is well approximated by a quadratic around the global maximum any choice of ε will result in a δ value that is not too small when compared to the width of the support of $h(x)$ and so the integral above will be fairly small. Hence ξ_3 will be small when $h(x)$ is well approximated by a quadratic in a neighborhood of the global maximum.

Lemma 3.7 *Let* x *and* y *be random variables with probability densities* $q(x)$ *and* $p(y)$. *Let* $p(y|x)$ *be the conditional density of* y *given* x *and* $p(x,y)$ *be the joint density of* x *and* y. *Let* $h(x) = \log p(x,y)$ *for fixed* $y = y^*$.

If the following conditions hold

i. *$h(x)$ has a unique global maximum $h(x_0)=h_{max}$ at $x=x_0$, $h'(x)$ exists in some neighborhood of x_0, $h''(x_0)$ exists and $h''(x_0)<0$*

ii. *$p(y^*|x)$ is bounded for all values of x*

iii. *The moment generating function of the random variable $|\mathrm{x}|$ exists*

then the function $x \mapsto p(x,y^*)$ is exponentially concentrated.

Proof For any $\delta>0$ the following holds:

$$\int_{|x-x_0|>\delta} p(x,y^*)dx = \Pr(|\mathrm{x}-x_0|>\delta) \int_{|x-x_0|>\delta} p(y^*|x)\frac{p(x)}{\Pr(|\mathrm{x}-x_0|>\delta)}dx$$

$$= \Pr(|\mathrm{x}-x_0|>\delta) \int_{|x-x_0|>\delta} p(y^*|x)p(x||x-x_0|>\delta)dx$$

$$= \Pr(|\mathrm{x}-x_0|>\delta)E_{\mathrm{x}||\mathrm{x}-x_0|>\delta}[p(y^*|\mathrm{x})]$$

In usual cases, $p(y|x)$ will be bounded as a function of x irrespective of the value of y. This implies that:

$$\int_{|x-x_0|>\delta} p(x,y^*)dx < K_2 \Pr(|\mathrm{x}-x_0|>\delta)$$

where $K_2=\max_x[p(y^*|x)]$. Using Chernoff's bound we have the following inequality for any $t>0$:

$$\Pr(|\mathrm{x}-x_0|>\delta) \le M(t)e^{-t\delta}.$$

In the above inequality $M(t)$ is the moment generating function of the random variable $|x-x_0|$. The fact that the moment generating function of $|x|$ exists implies that the moment generating function of $|x-x_0|$ exists as well and the proof is complete.

Corollary 3.8: Bound on probability density at a fixed point *Assume that the conditions in Lemma 3.7 hold and let:*

$$\hat{p}_+(y^*) = (2\pi)^{1/2}(-h''(x_0(y^*)) - 2\varepsilon)^{-1/2}$$

$$\hat{p}_-(y^*) = (2\pi)^{1/2}(-h''(x_0(y^*)) + 2\varepsilon)^{-1/2}$$

Then For any ε such that $0 < 3\varepsilon < |h''(x_0)|$, there exist positive real numbers δ, t (which is independent of ε), $K_1(t)$ and $K_2(t)$ such that the following inequality holds:

$$\hat{p}_-(y^*) - K_2(t)e^{-t\delta} < e^{-h(x_0)}p(y^*) < \hat{p}_+(y^*) + K_1(t)e^{-t\delta}.$$

Proof We can express $p(y^*)$ as

$$p(y^*) = \int_{-\infty}^{\infty} p(x, y^*)dx.$$

Therefore, Corollary 3.8 follows immediately from Lemma 3.7 and Theorem 3.5.

Chapter 4

Construction of a Saddlepoint Filter for a Stochastic Point Process Model

In this chapter we will illustrate how to construct a nonlinear filter for a Point Process model using the Laplace approximation. First we describe the state-space model and its pertinent features and then construct a saddlepoint filter algorithm for estimating the posterior density for this state-space model. We will then present results obtained using our algorithm and compare these results against those obtained using an existing filter.

4.1 Description of the Point Process Model

The Observation Process

Let $(0,T]$ be an observation interval during which the spiking activity of J independent point processes is recorded. For the ith point process the observations consist of the set of spike times $0 < u_{i1} < u_{i2} <,\cdots.< u_{iK_i} \leq T$. For any time t in the observation interval let $N^i_{0:t}(t)$ be the sample path of the ith point process. It is defined as the event $N^i_{0:t} = \{0 < u_{i1} < u_{i2} <,\cdots.< u_{ik} \leq t \cap N_i(t) = k\}$ where $N_i(t)$ is the number of events in $(0,t]$ and $k \leq K_i$. The sample path is a right continuous function that jumps by 1 at the event times and is constant otherwise [Snyder & Miller, 1991]. This function tracks the location and number of spikes in $(0,t]$ and therefore contains all the information in the sequence of event times. We use $\mathbf{N}_{0:t} = \{N^i_{0:t},\cdots,N^J_{0:t}\}$ to represent the ensemble activity in $(0,t]$.

Each point process can be characterized by its conditional intensity function, $\lambda^i(t|x(t), N^i_{0:t})$, where $x(t)$ is the latent state process modulating the activity of each of the point processes [Snyder & Miller, 1991]. The conditional intensity function defines the instantaneous firing rate of the point process in terms of the instantaneous probability of spiking as

$$\lambda^i(t|x(t), N^i_{0:t}) = \lim_{\Delta t \to 0} \frac{\Pr(N_i(t+\Delta t) - N_i(t+\Delta t)|x(t), N^i_{0:t})}{\Delta t} \tag{4.1}$$

Consequently, the probability of a single spike in a small interval $[t, t+\Delta t)$ can be approximated as $\lambda^i(t|x(t), N^i_{0:t})\Delta t$.

In order to facilitate the implementation of the algorithm recursively we will switch to a discrete-time framework. To achieve this we partition the observation interval into smaller intervals $(t_{k-1}, t_k]$ of length Δt. As a result, the spiking information observed for the ith point process in this framework is $n^i_k = N_i(k\Delta t) - N_i((k-)\Delta t)$ for $k = 1, 2, \cdots$. If Δt is sufficiently small the probability of more than one spike occurring in the Δt-spaced intervals will be negligible. So the new observation process, n^i_k, takes on the value 0 if there is no spike in $(t_{k-1}, t_k]$ or 1 if there is a spike. Let $n^i_{1:k} = \{n^i_1, \cdots, n^i_k\}$ represent the spike observations up to time t_k for the ith point process and $\mathbf{n}_{1:k} = \{n^1_{1:k}, \cdots, n^J_{1:k}\}$.

The Latent State Process

The latent state process $x(t)$ modulating the point processes will be taken to be a continuous time first order autoregressive (CT AR (1)) process

$$\frac{dx(t)}{dt} + \alpha_0 x(t) = \eta(t) \tag{4.2}$$

where $\eta(t)$ is a white noise process with variance per unit time σ_η^2. In order to obtain an expression for the relationship between the states at discrete time points we re-express (4.2) in the following form

$$dx(t) = -\alpha_0 x(t) + dW(t).$$

where $W(t)$ is a Wiener process. It is easy to see that the following holds

$$d(e^{\alpha_0 t} x(t)) = e^{\alpha_0 t} dW(t).$$

Now integrating the above equation over the interval $(t_{k-1}, t_k]$ of length Δt will give us

$$x(t_k) = e^{-\alpha_0 \Delta t} x(t_{k-1}) + \int_{t_{k-1}}^{t_k} e^{-\alpha_0 \Delta t} dW(t).$$

Putting $x_k = x(t_k)$, $F = e^{-\alpha_0 \Delta t}$ and $\varepsilon_k = \int_{t_{k-1}}^{t_k} e^{-\alpha_0 \Delta t} dW(t)$ we obtain the equivalent discrete-time first order autoregressive representation of the CT AR (1) process:

$$x_k = F x_{k-1} + \varepsilon_k \tag{4.3}$$

where ε_k is the Gaussian process with $E[\varepsilon_k] = 0$, $Var[\varepsilon_k] = \frac{\sigma_\eta^2}{2\alpha}(1 - F^2)$. Furthermore, ε_k and ε_j are independent when $k \neq j$. Choosing an initial state which is Gaussian with mean, $E[x_0] = 0$, and Variance, $Var[x_0] = \sigma_x^2 = \frac{\sigma_\eta^2}{2\alpha}$, will make the state process covariance stationary.

The likelihood model for the point process observations

Now that we have described the state and observation processes, the final essential component is the likelihood model for the point process observations, $p(\mathbf{n}_k | x_k, \mathbf{N}_{1:k-1})$, where $\mathbf{n}_k = \{n_k^1, \cdots, n_k^J\}$ This model defines the probability of observing spikes in the interval $(t_{k-1}, t_k]$, based on the current state of the system and the past spiking activity. This likelihood is well approximated by

$$p(\mathbf{n}_k | x_k, \mathbf{N}_{1:k-1}) = \exp\left[\sum_{i=1}^{J} n_k^i \log(\lambda_k^i \Delta t) - \lambda_k^i \Delta t\right] \tag{4.4}$$

[Brown, Barbieri, Eden, & Frank, 2003] where $\lambda_k^i = \lambda^i(t_{k-1} | x(t_{k-1}), \mathbf{N}_{0:t_{k-1}})\Delta t$

4.2 Saddlepoint Filter Algorithm for the State-Space Model

The BCK update equations for the system

Bayes' rule can be used to obtain the posterior density at time t_k as a function of the likelihood at time t_k and the one-step prediction density of the state at time t_k given the observations up to time t_{k-1}

$$p(x_k | \mathbf{n}_{1:k}, \mathbf{N}_{1:k}) = \frac{p(\mathbf{n}_k | x_k, \mathbf{N}_{1:k}) p(x_k | \mathbf{n}_{1:k-1}, \mathbf{N}_{1:k-1})}{p(\mathbf{n}_k | \mathbf{n}_{1:k-1}, \mathbf{N}_{k-1})} \tag{4.5}$$

The first term in the numerator of (4.5) is the likelihood and the second term is the one-step prediction density. The one-step prediction density is defined by the Chapman-Kolmogorov equation in terms of the state transition density and the posterior density at time t_{k-1} as shown below.

$$p(x_k | \mathbf{n}_{1:k-1}, \mathbf{N}_{1:k-1}) = \int p(x_k | x_{k-1}) p(x_{k-1} | \mathbf{n}_{1:k-1}, \mathbf{N}_{1:k-1}) dx_{k-1} \tag{4.6}$$

Constructing a saddlepoint filter algorithm for the state-space model

The strategy is to apply the Laplace approximation to the integral for the one-step prediction density in (4.6) at each time-step. This leads to the following algorithm:

Let $\{x^{(i)}\}$ be the common set of grid points at which we evaluate the posterior density at each time step k. The grid points are spaced at intervals of Δx in the range $[x_{min}, x_{max}]$. In other words, $x^{(i)} = x_{min} + i\Delta x,$ for $i = 0,1,\ldots,N_{grid}$ and where $N_{grid} = (x_{max} - x_{min})/\Delta x + 1$.

The outer limits x_{min} and x_{max} have to be chosen by the user in such a way that the value of the posterior density outside this range is negligible at each time step. This can be achieved by implementing the algorithm with an initial choice of x_{min} and x_{max} and then incrementally decreasing and increasing x_{min} and x_{max} respectively until the estimates of the posterior mean and posterior variance obtained using one set of outer limits do not differ significantly from the estimates produced using the previous set of outer limits.

Step $\mathbf{k = 0}$

For $i = 0,1,\ldots,N_{grid}$*:*

Put $p(x^{(i)} | \mathbf{n}_{1:0}, \mathbf{N}_{1:0}) = p_{x_0}(x^{(i)}) = N(x^{(i)}; \overline{x}_{0|0}, \sigma^2_{0|0})$ where $\overline{x}_{0|0} = 0$ and $\sigma^2_{0|0} = \sigma^2_x$ and $N(x^{(i)}; \overline{x}_{0|0}, \sigma^2_{0|0})$ is the value at $x^{(i)}$ of the Gaussian probability density parameterized by mean and variance $\overline{x}_{0|0}, \sigma^2_{0|0}$ respectively.

Step $\mathbf{k = 1}$

For $i = 0,1,\ldots,N_{grid}$*:*

The value of x_0 that maximizes $h(x_0) = \log p(x^{(i)} | x_0) + \log p(x_0 | \mathbf{n}_{1:0}, \mathbf{N}_{1:0})$ is

$$\hat{x}_0 = \left(\frac{\overline{x}_{0|0}}{\sigma_{0|0}^2} + \frac{x^{(i)}}{\sigma_\varepsilon^2} \right) \left(\frac{1}{\sigma_{0|0}^2} + \frac{F}{\sigma_\varepsilon^2} \right)^{-1}$$

and the second derivative of $h(x_0)$ evaluated at the maximum is:

$$h''(\hat{x}_0) = -\left(\frac{1}{\sigma_{0|0}^2} + \frac{F}{\sigma_\varepsilon^2} \right)$$

The one-step prediction density at $x^{(i)}$ is then approximated by:

$$\hat{p}(x^{(i)} \left| \mathbf{n}_{1:0}, \mathbf{N}_{1:0} \right.) = \left[2\pi \Big/ \left(\frac{1}{\sigma_{0|0}^2} + \frac{F}{\sigma_\varepsilon^2} \right) \right]^{\frac{1}{2}} p(x^{(i)} \left| \hat{x}_0 \right.) p(\hat{x}_0 \left| \mathbf{n}_{1:0}, \mathbf{N}_{1:0} \right.)$$

Observe that $\hat{x}_0$ may not fall on one of the grid points used to store values of the density. In such a case a linear interpolation using the nearest grid points will suffice.

The posterior density at $x^{(i)}$ is then given by:

$$\hat{p}(x^{(i)} \left| \mathbf{n}_{1:1}, \mathbf{N}_{1:1} \right.) \propto p(\mathbf{n}_1 \left| x^{(i)}, \mathbf{N}_{1:1} \right.) \hat{p}(x^{(i)} \left| \mathbf{n}_{1:0}, \mathbf{N}_{1:0} \right.) \tag{4.7}$$

After the posterior density is evaluated at all the grid points, numerically compute the mean, $\overline{x}_{1|1}$, and the variance, $\sigma_{1|1}^2$, of the Laplace approximation to the posterior density

Step $\mathbf{k} = 2,3,....$

For $i = 0,1,\ldots,N_{grid}$*:*

The value of x_{k-1} that maximizes $h(x_{k-1}) = \log p(x^{(i)} \left| x_{k-1} \right.) + \log p(x_{k-1} \left| \mathbf{n}_{1:k-1}, \mathbf{N}_{1:k-1} \right.)$ is obtained by approximating $p(x_{k-1} \left| \mathbf{n}_{1:k-1}, \mathbf{N}_{1:k-1} \right.)$ by $N(x_{k-1}; \overline{x}_{k-1|k-1}, \sigma_{k-1|k-1}^2)$. The result obtained is

$$\hat{x}_{k-1} = \left(\frac{\overline{x}_{k-1|k-1}}{\sigma^2_{k-1|k-1}} + \frac{x^{(i)}}{\sigma^2_\varepsilon} \right) \left(\frac{1}{\sigma^2_{k-1|k-1}} + \frac{F}{\sigma^2_\varepsilon} \right)^{-1}$$

and the second derivative of $h(x_{k-1})$ evaluated at the maximum is approximately:

$$h''(\hat{x}_{k-1}) = -\left(\frac{1}{\sigma^2_{k-1|k-1}} + \frac{F}{\sigma^2_\varepsilon} \right)$$

This approximation to the value of x_{k-1} that maximizes $h(x_{k-1})$ works well for this particular model because the true $h(x_{k-1})$ is well approximated by a quadratic in some neighborhood of the maximum.

The one-step prediction density at $x^{(i)}$ is then approximated by:

$$\hat{p}(x^{(i)} | \mathbf{n}_{1:k-1}, \mathbf{N}_{1:k-1}) = \left[2\pi \Big/ \left(\frac{1}{\sigma^2_{k-1|k-1}} + \frac{F}{\sigma^2_\varepsilon} \right) \right]^{\frac{1}{2}} p(x^{(i)} | \hat{x}_{k-1}) \hat{p}(\hat{x}_{k-1} | \mathbf{n}_{1:k-1}, \mathbf{N}_{1:k-1})$$

As pointed out in step $\mathbf{k} = 1$, $\hat{x}_{k-1}$ may not fall on one of the grid points used to store values of the density. In such a case a linear interpolation using the nearest grid points will suffice.

The posterior density is then given by:

$$\hat{p}(x^{(i)} | \mathbf{n}_{1:k}, \mathbf{N}_{1:k}) \propto p(\mathbf{n}_k | x^{(i)}, \mathbf{N}_{1:k}) \hat{p}(x^{(i)} | \mathbf{n}_{1:k-1}, \mathbf{N}_{1:k-1}) \quad (4.8)$$

After the posterior density is evaluated at all the grid points, numerically compute the mean, $\overline{x}_{k|k}$, and the variance, $\sigma^2_{k|k}$, of the Laplace approximation to the posterior density

4.3 Applications

To illustrate the algorithm, we choose a simple form of the conditional intensity function. That is, we take the conditional intensity function for each point process as

$$\lambda^i(k\Delta t \,|\, x(k\Delta t), N_{1:k}^i) = \exp(\mu_i + \beta_i x(k\Delta t)) \tag{4.9}$$

Example: Multiple Independent Point Processes Driven by Common Latent State Process

One realization of the model given by Equations (4.2) and (4.9) was simulated. The time interval for the simulation was 1000 milliseconds and the latent state process model parameters were $F = 0.99$ and $\sigma_\varepsilon^2 = 10^{-3}$ while the variance of the initial state was chosen as $\sigma_0^2 = \sigma_\varepsilon^2 / (1 - F^2) = 0.0503$ in order to make the latent state process covariance-stationary. The log of the background firing rate was chosen as $\mu = -4.9$ for all 10 observation processes, while the gain coefficients β_i were chosen randomly on the interval $[0.9\ 1.1]$. All model parameters are summarized in Table 1. The simulated latent process and the first five point processes are illustrated in Figure 1.

The ten point process realizations were each simulated using an algorithm based on the time-rescaling theorem described in Brown et al. (2001), while the state equations were updated every millisecond. The posterior mean, variance and density at various time points were estimated using the Saddlepoint algorithm described in the previous section with $\Delta x = 0.001, x_{min} = -2.0$ and $x_{max} = 3.0$. The estimation was carried out using different numbers of observation processes. Specifically, one, two, four, eight and 10 independent observation processes were used. The results obtained were compared against those obtained by another algorithm, the Stochastic State Point Process Filter (SSPPF). The SSPPF algorithm is obtained by approximating the posterior density in Equation (4.5) by a Gaussian density. This then leads to a recursive algorithm for the posterior mean and variance in terms of observed and previously estimated quantities (see [Eden et al, 2004] for more details). This filter was a good choice as it has been shown to be superior to point process analogues of the extended kalman filter, the recursive least squares filter and the Steepest Descent filter when estimating a latent state process from point process observations (see [Eden et al, 2004] and [Eden, 2005]).

Parameter	Value	Parameter	Value	Parameter	Value
x_{min}	-2.000	μ	-4.900	β_6	1.0524
x_{max}	3.000	β_1	1.0900	β_7	0.9913
Δx	0.0010	β_2	0.9462	β_8	0.9037
F	0.9900	β_2	1.0214	β_9	1.0643
σ_ε^2	0.0010	β_4	0.9972	β_{10}	0.9889
σ_0^2	0.0503	β_5	1.0783		

Table 1: Model and Simulation Parameters

For this model, the SSPPF performed poorly in estimating the posterior density. While the mean of its guassian approximation to the density was located very close to the true mean, the error in approximating the shape of the posterior density was large. In fact the support of the SSPPF estimate was much narrower than the support of the true density. This is because even though the true density for this model is symmetric about the mean, it is very non-gaussian and has significant moments of third order and higher. As a result, any confidence interval estimates by the SSPPF for this model will be highly inaccurate. The Saddlepoint algorithm did not share this failing. In fact, the posterior density estimated by the Saddlepoint algorithm very accurately approximated the true shape of the posterior density at each time point. Figures 2, 3 and 4 illustrate these observations

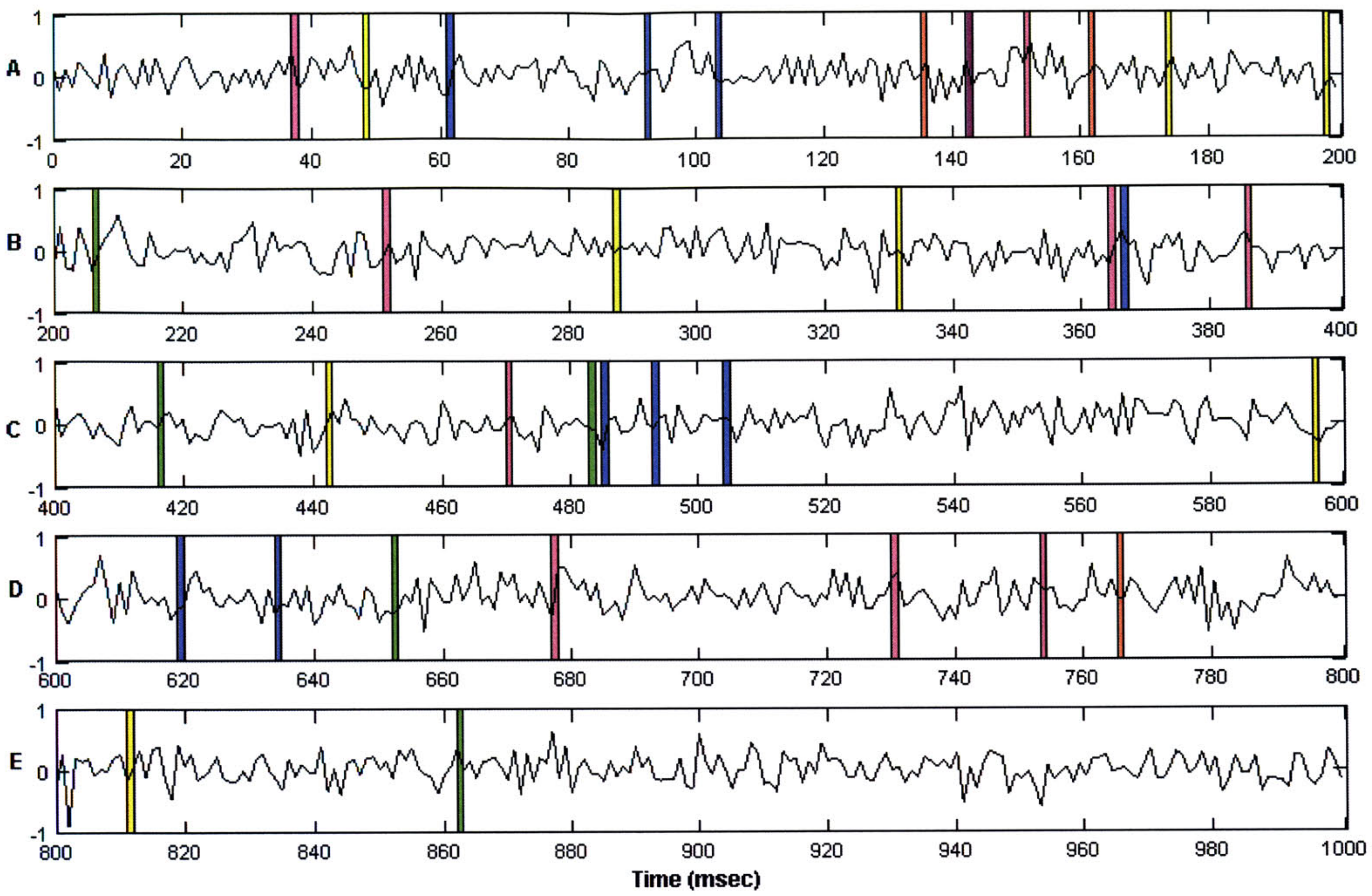


Figure 1: Realization of Latent State Process and First 5 Point Processes

Shaded areas indicate that a spike occurred in that 1 millisecond interval: Blue – Spike from point process 1; Red – Spike from point process 2; Green – Spike from point process 3; Magenta – Spike from point process 4; Yellow – Spike from point process 5.

A) Latent Process and Spiking activity in 1st 200 millisecond time period

B) Latent Process and Spiking activity in 2nd 200 millisecond time period

C) Latent Process and Spiking activity in 3rd 200 millisecond time period

D) Latent Process and Spiking activity in 4th 200 millisecond time period

E) Latent Process and Spiking activity in 5th 200 millisecond time period

at times 107 and 500 milliseconds. The posterior means estimated by both the SSPPF and Saddlepoint approximations were very accurate. However, the error in the Saddlepoint approximation was lower. The SSPPF also approximated the posterior variance well at non-spike times. However, it failed to track jumps in the posterior variance in periods when spikes occurred and shortly after spikes occurred. The saddlepoint filter on the other hand, accurately tracked jumps in the posterior variance during periods of spiking activity.

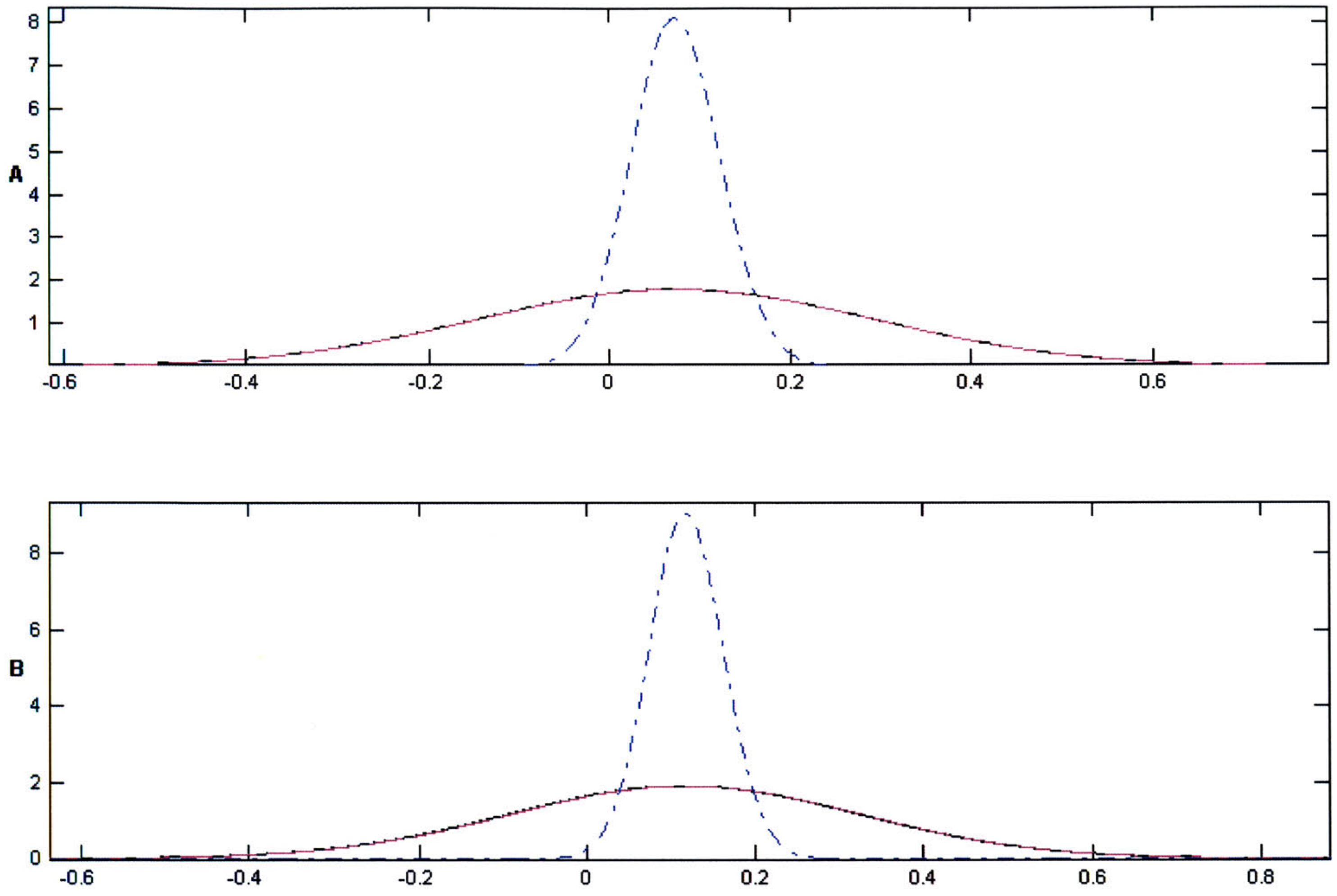


Figure 2: Comparison of SSPPF and Saddlepoint posterior density at 500 milliseconds

Black line - True posterior density; Magenta dashed line– Saddlepoint posterior density; Blue dashed line – SSPPF posterior density

A) 1 point process used in the estimation

B) 8 point processes used in the estimation

To illustrate these observations, the squared error in the SSPPF and Saddlepoint approximations to the posterior mean and variance were plotted at all the timepoints. The saddlepoint error in approximating the mean was about half that of the SSPPF approximation at all time points. The saddlepoint error in approximating the variance was also smaller than that of the SSPPF. The SSPPF variance approximation error increased by a factor of about 10 at periods in which spikes occurred since its estimates did not track well the jumps in the variance at spike time, while the saddlepoint error was consistently low at spike times. These results are presented in Figures 9 to 12.

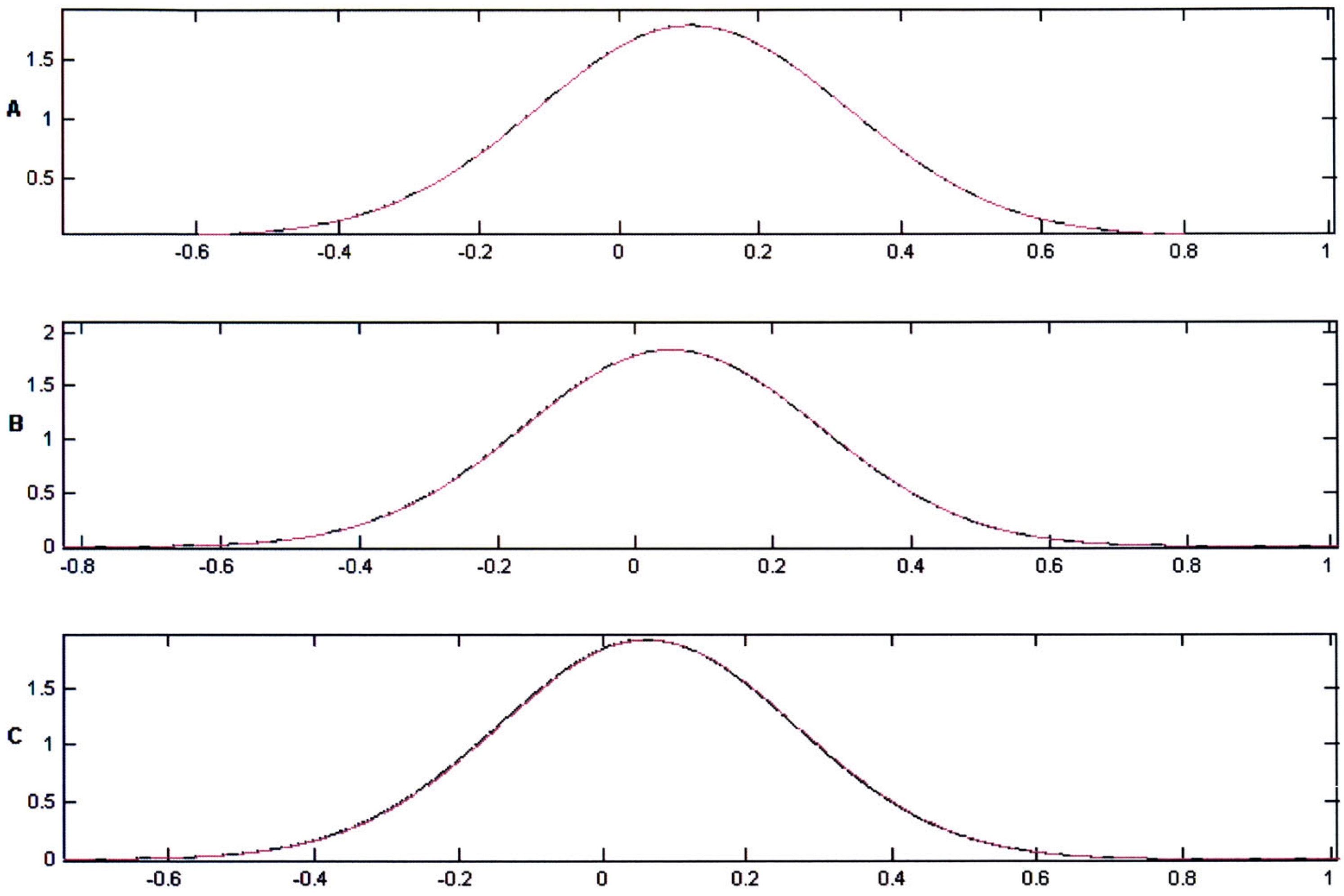


Figure 3: Saddlepoint and Exact posterior density at 107 milliseconds

Black line - True posterior density; Magenta dashed line – Saddlepoint posterior density

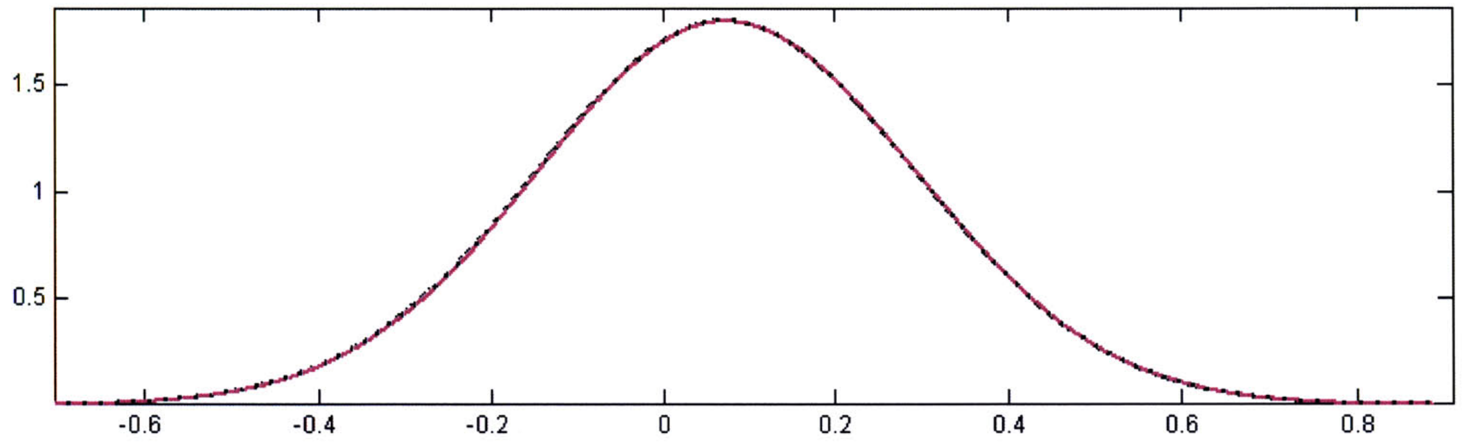
A) 1 point process used in the estimation

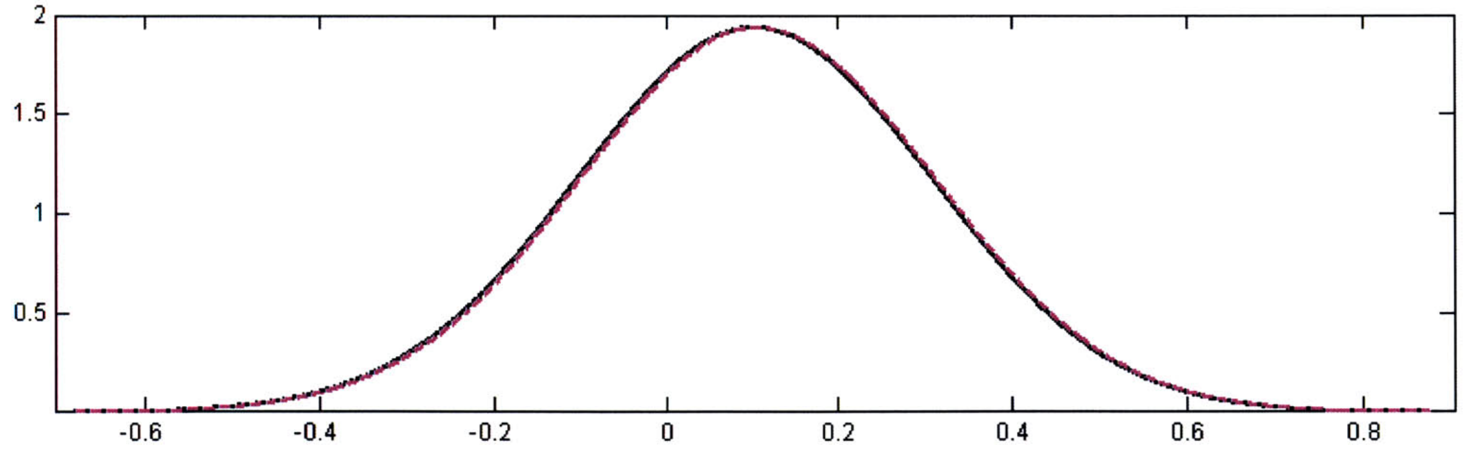
B) 4 point processes used in the estimation

C) 10 point processes used in the estimation

The results above match what we would expect from the theory developed in Chapter 3. Since the integrand in Equation (4.6) is exponentially concentrated the results in Chapter

3 on the error bounds of the Laplace approximation apply for this model. As explained in the previous chapter Laplace approximation error will be zero at the first time step of the algorithm for this model. This error will increase but that increase is controlled by how much of a spreading effect the state transition density has on the posterior density in the integrand of Equation (4.6). The narrower the transition density, the slower will be the growth of the approximation error. The rate of increase will also depend on the effect of the data likelihood on how well the natural logarithm of the integrand in (4.6) is





approximated by a quadratic. At the first step of the algorithm, the natural logarithm of the integrand will be exactly quadratic. For the intensity function in (4.9), the increase in the deviation of the natural logarithm of the integrand from the quadratic will be very slow as the algorithm progresses through subsequent steps. This, together with the fact that the state transition density itself has most of its area concentrated about the mean, ensures that the error will be consistently small as time increases. This is borne out by the results we obtained.

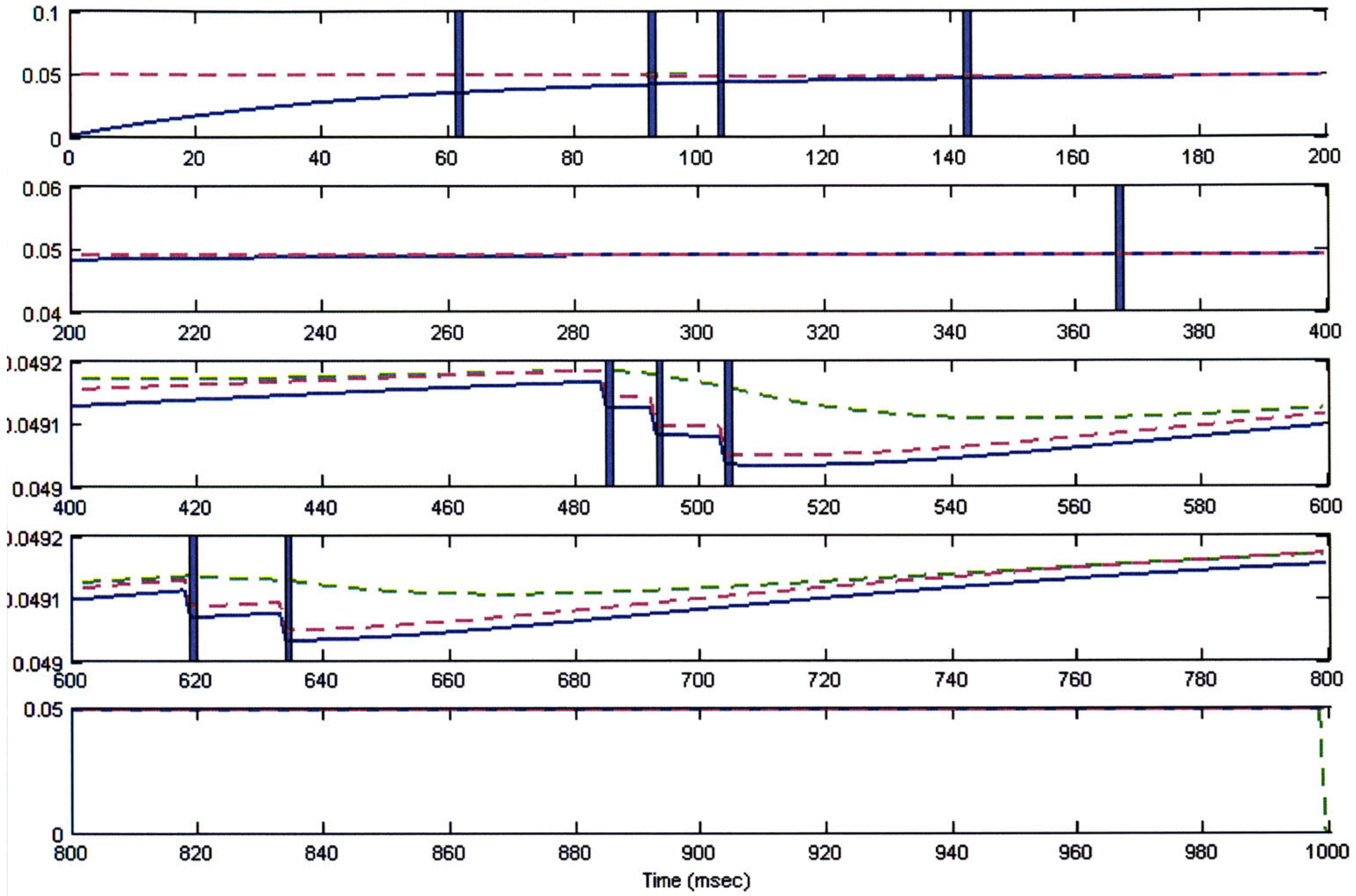


Figure 5: Posterior Variance. Estimating with 1 Point Process

Blue line - True posterior Variance; Magenta dashed line - Saddlepoint Posterior Variance; Green dashed line - SSPPF variance. Shaded areas indicate that a spike occurred in that 1 millisecond interval: Shaded Blue – Spike from point process 1

A) 1st 200 millisecond time period

B) 2nd 200 millisecond time period

C) 3rd 200 millisecond time period

D) 4th 200 millisecond time period

E) 5th 200 millisecond time period

Saddlepoint variance tracks true posterior variance very well even during periods of spiking activity. SSPPF variance fails to track jumps in true posterior variance during periods of spiking activity.

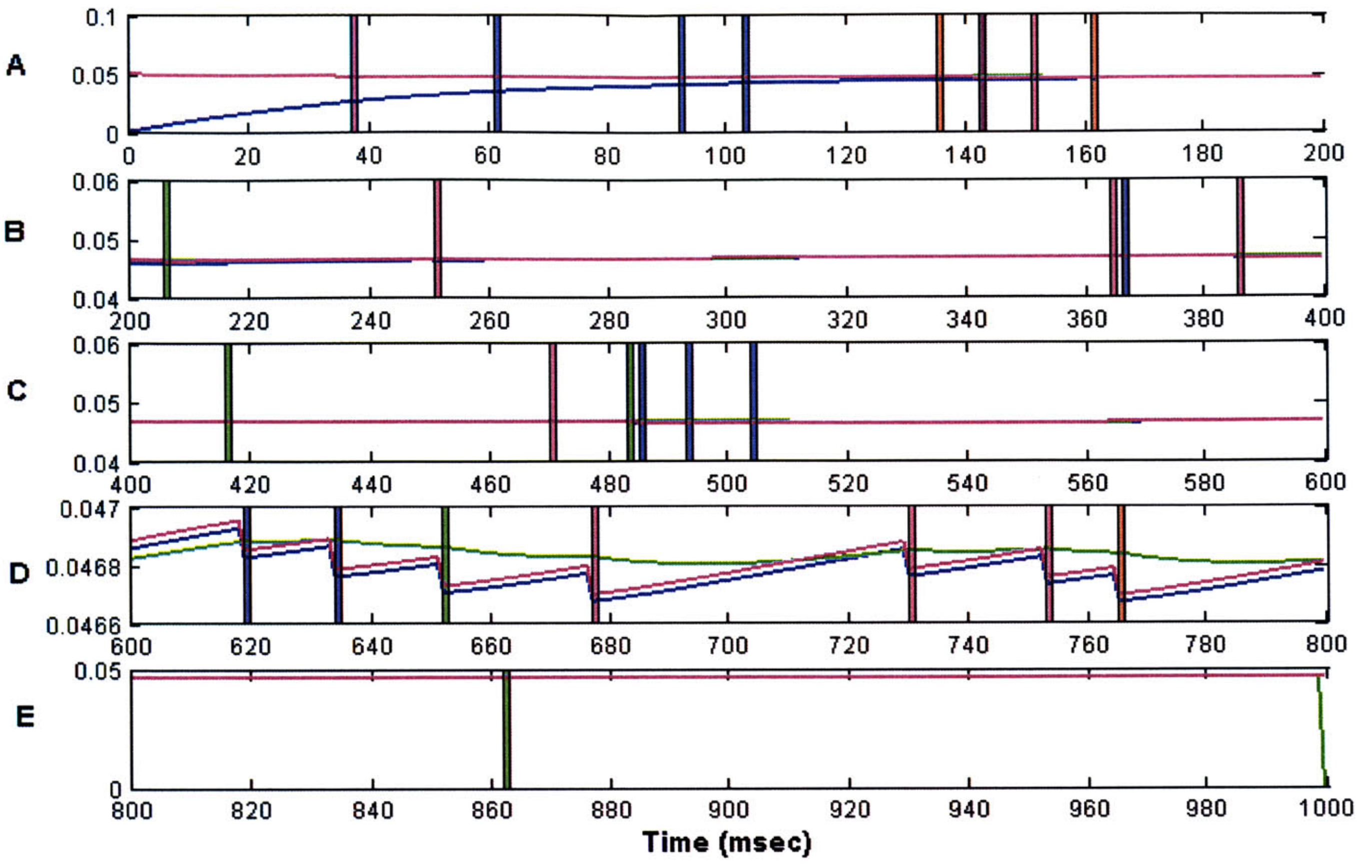


Figure 6: Posterior Variance. Estimating with 4 Point Processes

Blue line - True posterior Variance; Magenta line - Saddlepoint Posterior Variance; Green line - SSPPF variance. Shaded areas indicate that a spike occurred in that 1 millisecond interval: Shaded Blue – Spike from point process 1; Red – Spike from point process 2; Green – Spike from point process 3; Magenta – Spike from point process 4

A) 1st 200 millisecond time period

B) 2nd 200 millisecond time period

C) 3rd 200 millisecond time period

D) 4th 200 millisecond time period

E) 5th 200 millisecond time period

Saddlepoint variance tracks true posterior variance very well even during periods of spiking activity. SSPPF variance fails to track jumps in true posterior variance during periods of spiking activity.

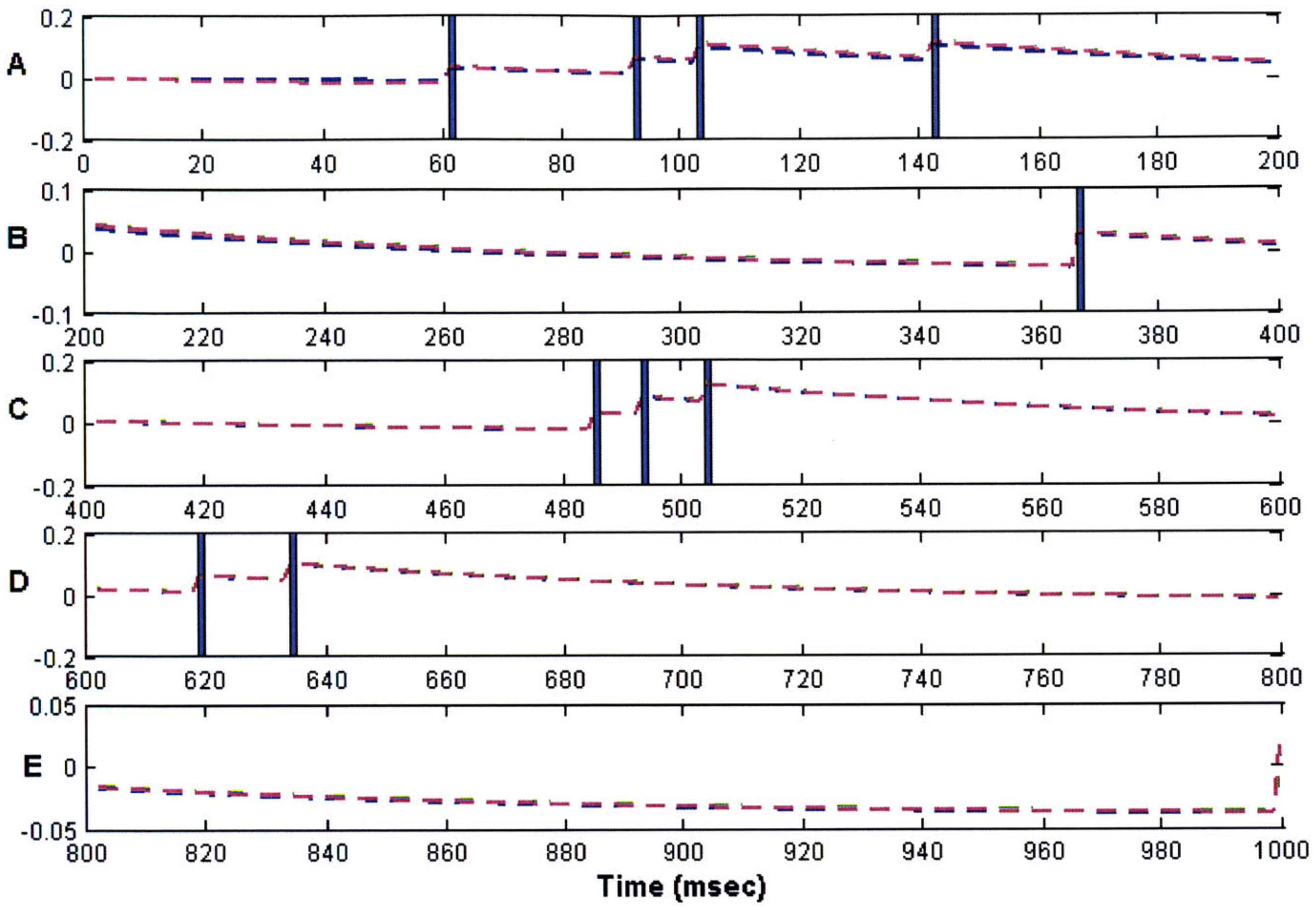


Figure 7: Posterior Mean. Estimating with 1 Point Process

Blue dashed line - True posterior Mean; Magenta dashed line - Saddlepoint Posterior Mean; Green dashed line - SSPPF Mean. Shaded areas indicate that a spike occurred in that 1 millisecond interval: Shaded Blue – Spike from point process 1

A) 1st 200 millisecond time period

B) 2nd 200 millisecond time period

C) 3rd 200 millisecond time period

D) 4th 200 millisecond time period

E) 5th 200 millisecond time period

Both Saddlepoint and SSPPF estimates of the posterior mean track the true posterior mean accurately

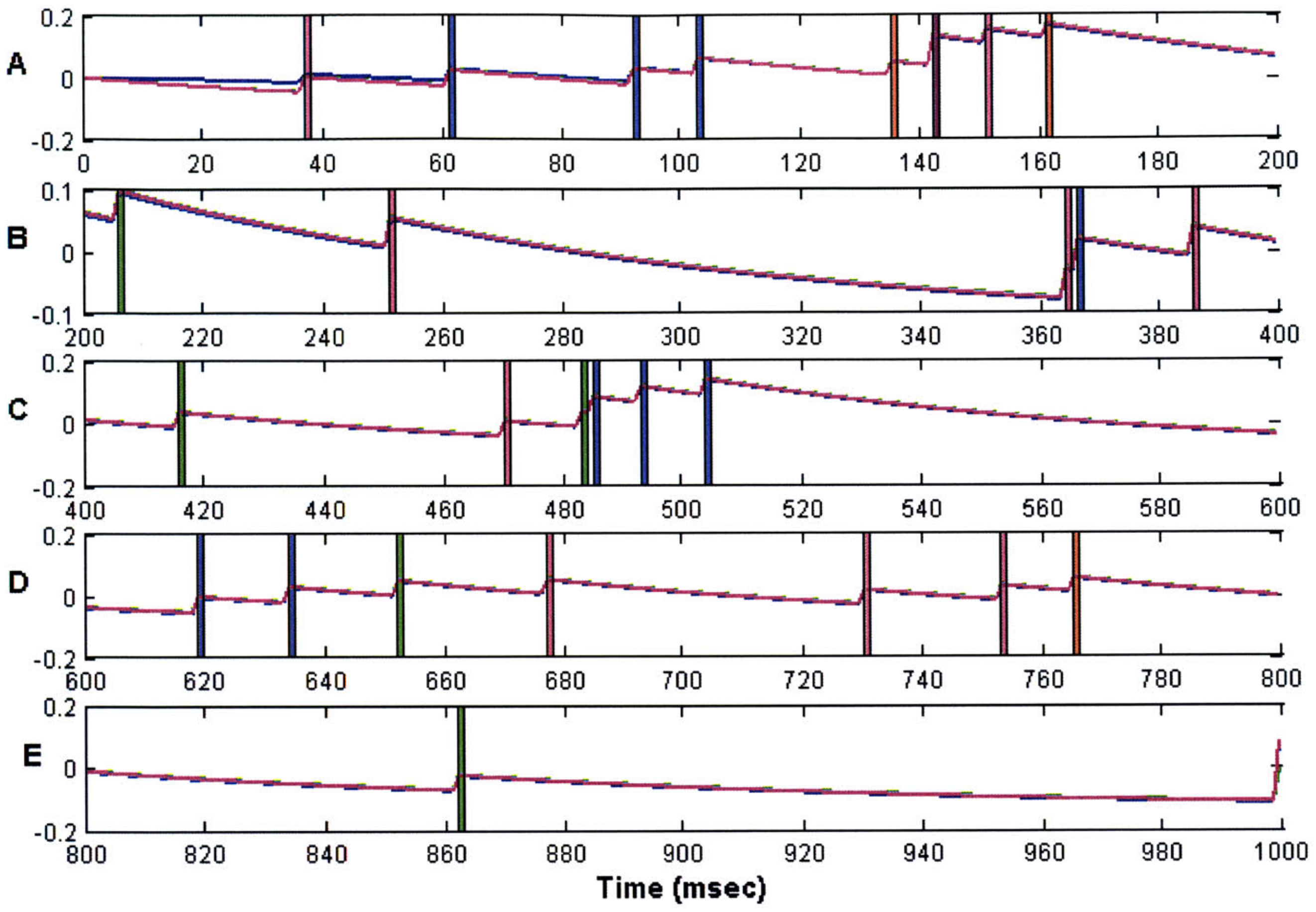


Figure 8: Posterior Mean. Estimating with 4 Point Processes

Blue line - True posterior Mean; Magenta line - Saddlepoint Posterior Mean; Green line - SSPPF Mean. Shaded areas indicate that a spike occurred in that 1 millisecond interval: Shaded Blue – Spike from point process 1; Red – Spike from point process 2; Green – Spike from point process 3; Magenta – Spike from point process 4

A) 1st 200 millisecond time period

B) 2nd 200 millisecond time period

C) 3rd 200 millisecond time period

D) 4th 200 millisecond time period

E) 5th 200 millisecond time period

Both Saddlepoint and SSPPF estimates of the posterior mean track the true posterior mean accurately

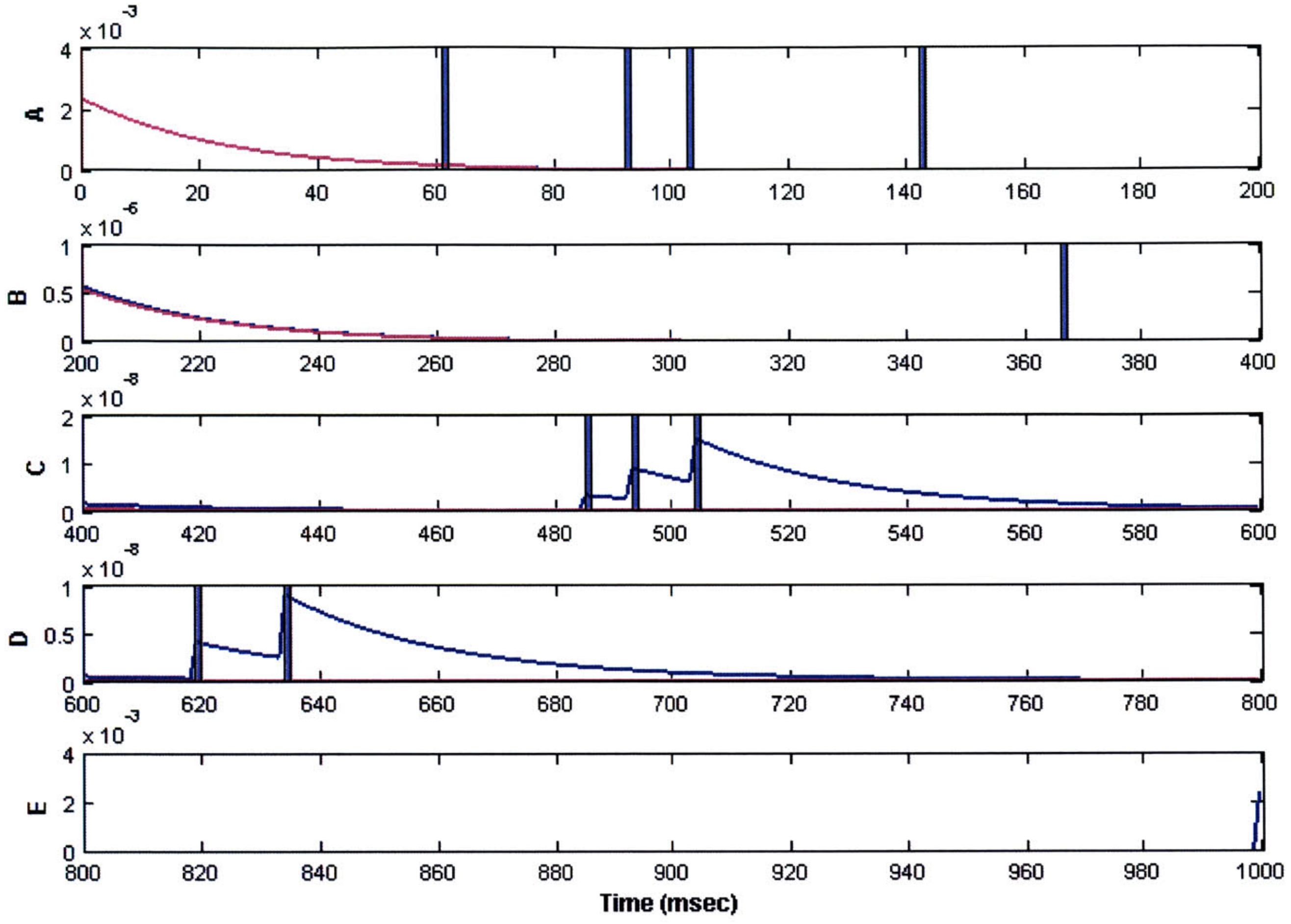


Figure 9: Squared Error in the posterior Variance. Estimating with 1 Point Process

Blue line – SSPPF Error; Magenta line - Saddlepoint error. Shaded areas indicate that a spike occurred in that 1 millisecond interval: Shaded Blue – Spike from point process 1.

A) 1st 200 millisecond time period

B) 2nd 200 millisecond time period

C) 3rd 200 millisecond time period

D) 4th 200 millisecond time period

E) 5th 200 millisecond time period

Saddlepoint error is consistently very small even during periods of spiking activity. SSPPF error increases during periods of spiking activity.

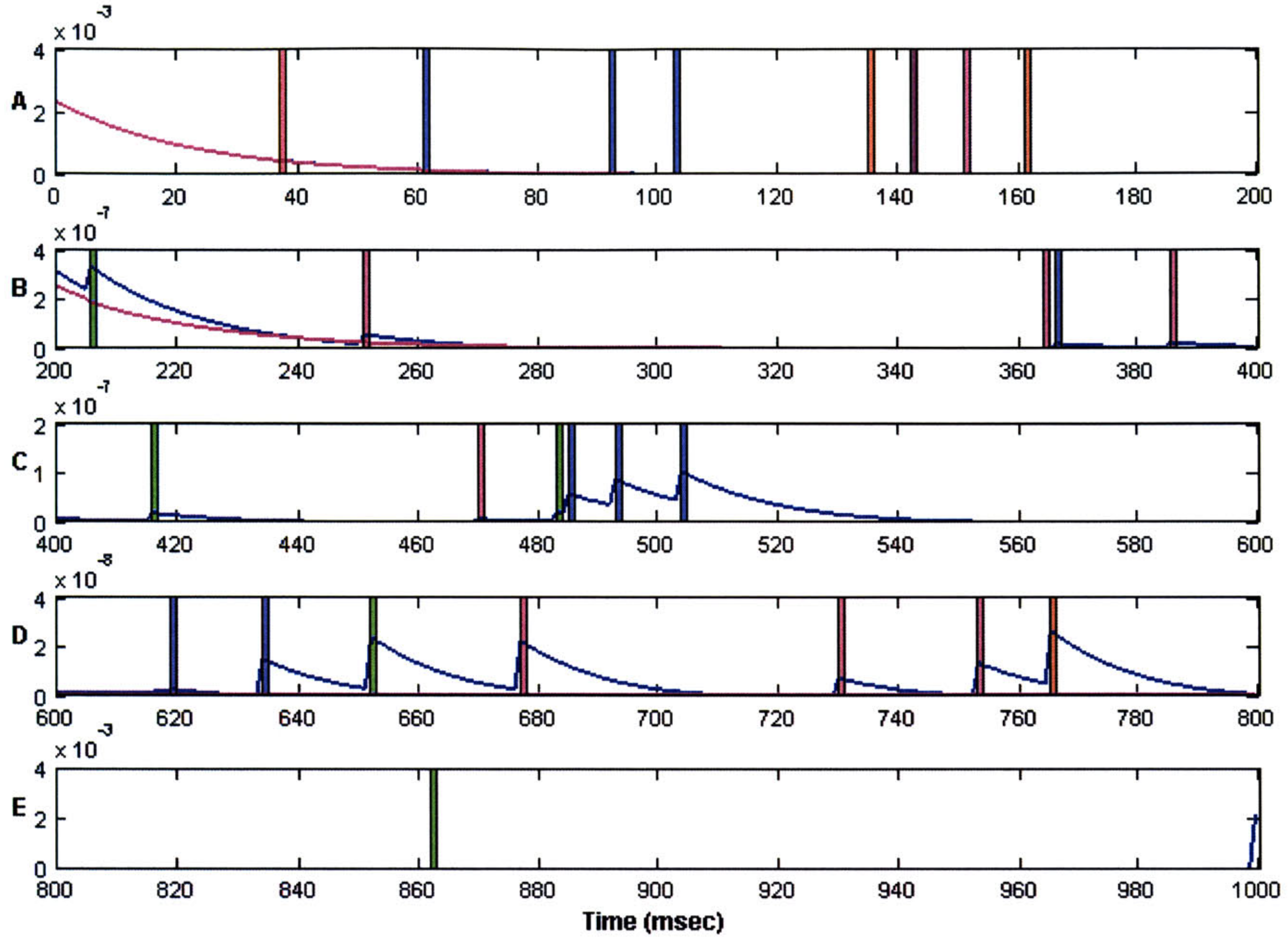


Figure 10: Squared Error in the posterior Variance. Estimating with 4 Point Processes

Blue line – SSPPF Error; Magenta line - Saddlepoint error. Shaded areas indicate that a spike occurred in that 1 millisecond interval: Shaded Blue – Spike from point process 1; Red – Spike from point process 2; Green – Spike from point process 3; Magenta – Spike from point process 4

A) 1st 200 millisecond time period

B) 2nd 200 millisecond time period

C) 3rd 200 millisecond time period

D) 4th 200 millisecond time period

E) 5th 200 millisecond time period

Saddlepoint error is consistently very small even during periods of spiking activity. SSPPF error increases during periods of spiking activity.

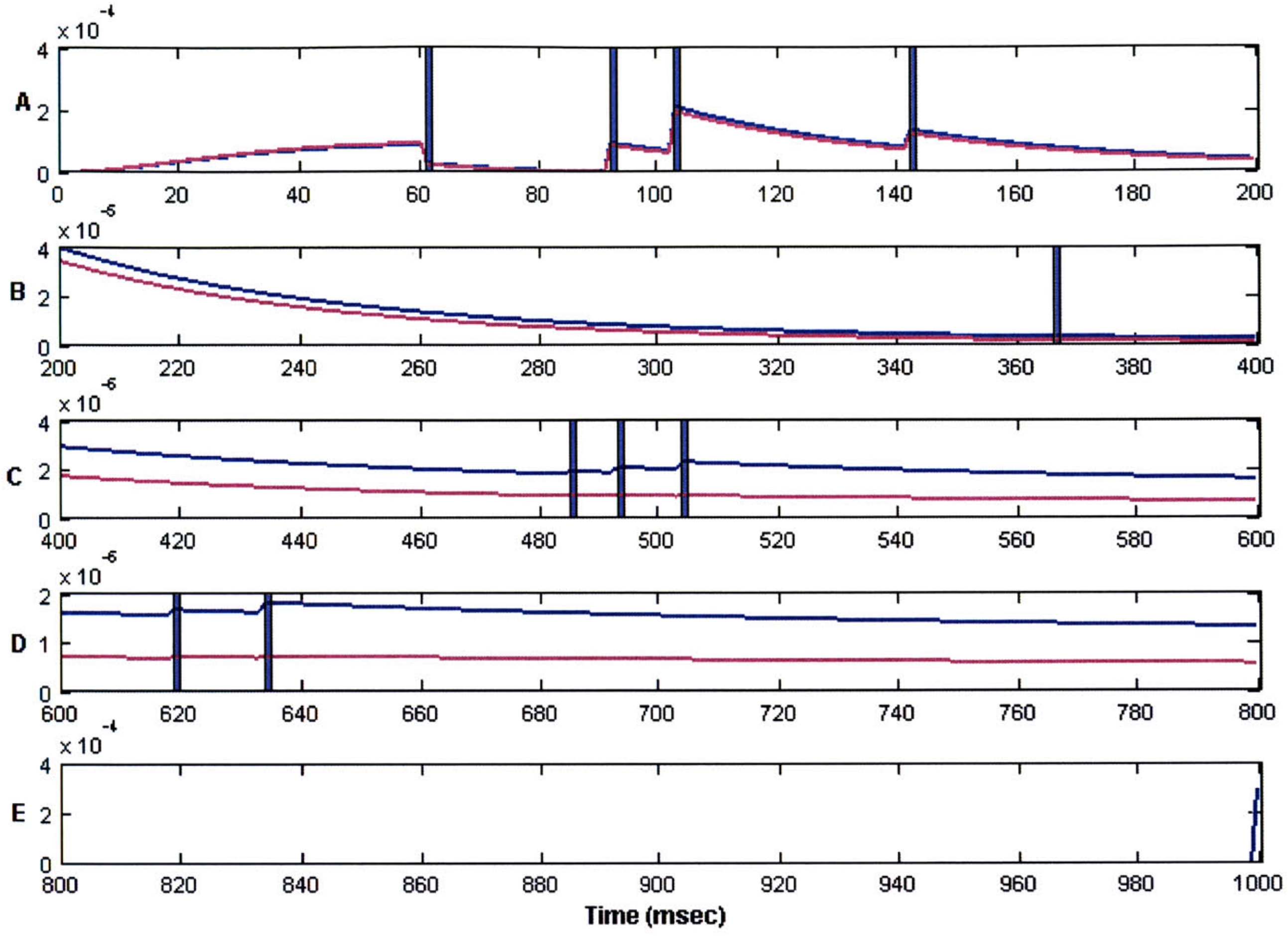


Figure 11: Squared Error in the Posterior Mean. Estimating with 1 Point Process

Blue line – SSPPF error; Magenta line - Saddlepoint error. Shaded areas indicate that a spike occurred in that 1 millisecond interval: Shaded Blue – Spike from point process 1

A) 1st 200 millisecond time period

B) 2nd 200 millisecond time period

C) 3rd 200 millisecond time period

D) 4th 200 millisecond time period

E) 5th 200 millisecond time period

Saddlepoint error about half that of the SSPPF

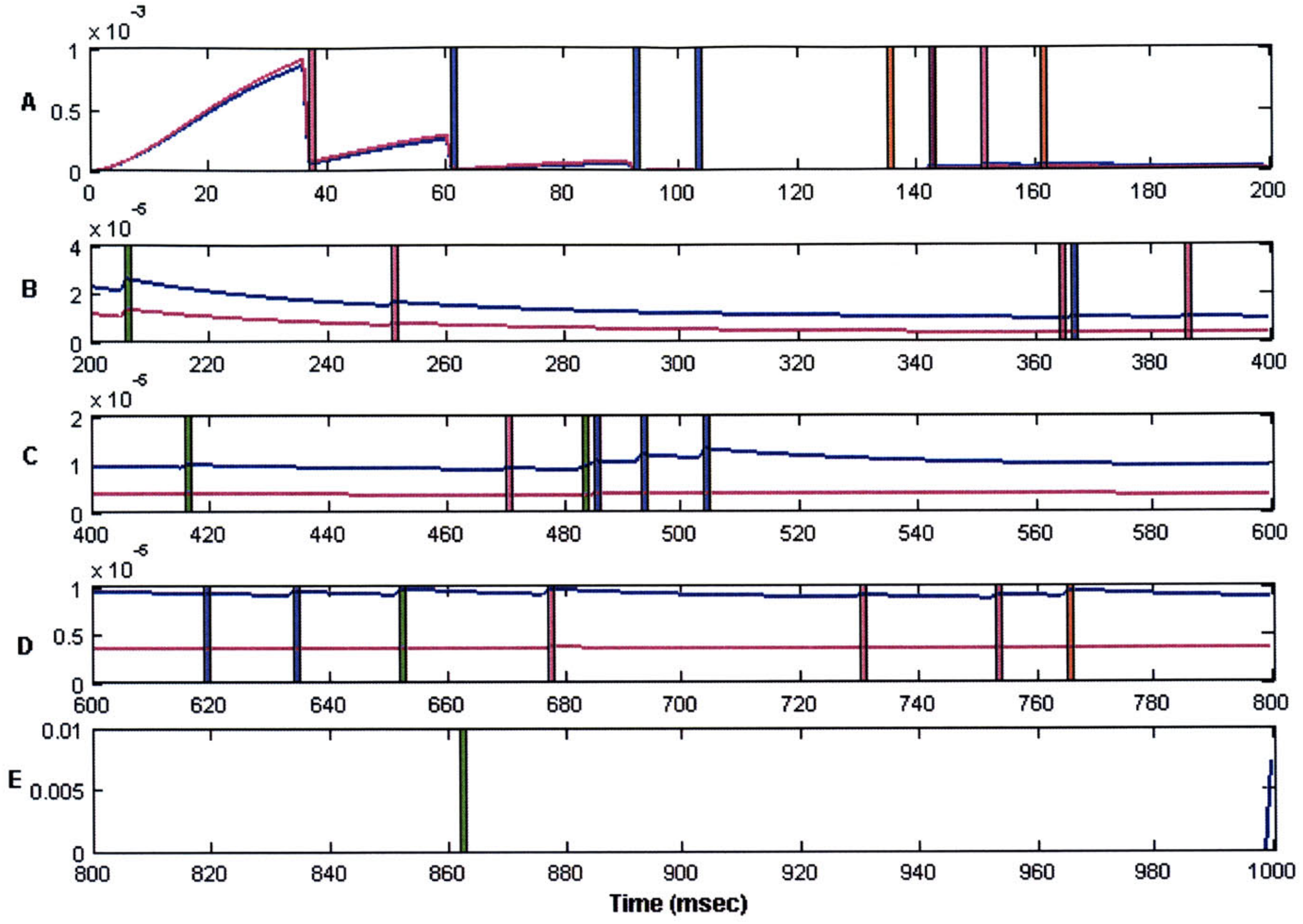


Figure 12: Squared Error in the Posterior Mean. Estimating with 4 Point Processes

Blue line – SSPPF error; Magenta line - Saddlepoint error. Shaded areas indicate that a spike occurred in that 1 millisecond interval: Shaded Blue – Spike from point process 1; Red – Spike from point process 2; Green – Spike from point process 3; Magenta – Spike from point process 4

A) 1st 200 millisecond time period

B) 2nd 200 millisecond time period

C) 3rd 200 millisecond time period

D) 4th 200 millisecond time period

E) 5th 200 millisecond time period

Saddlepoint error about half that of the SSPPF

Chapter 5

Conclusions and Future Directions

This thesis has focused on developing results that will facilitate a better understanding of the limits on the performance of the Laplace approximation when applied in a nonlinear filtering context. Such an understanding will then help guide its application either as a stand-alone nonlinear filtering algorithm or in concert with a Particle filter.

5.1 Summary of Thesis Results

In chapter one we briefly described the nonlinear filtering problem and then reviewed the Bayes-Chapman-Kolmogorov equations for sequentially updating the posterior density. These BCK equations later formed the starting point of our exploration of how the attractive performance properties of the Laplace approximation can be taken advantage of in a nonlinear filtering context. In chapter three we discussed how the Laplace approximation is applied in obtaining estimates of asymptotic integrals. For non-asymptotic integrals, we showed that analogous error bounds could be established for exponentially concentrated functions. These bounds helped us to better understand the factors limiting the relative error of the approximation. We then showed how to construct a nonlinear filtering algorithm using the Laplace approximation for a particular state space model with point process observations. The algorithm based on the Laplace approximation outperformed existing algorithms such as the SSPPF and the EKF in estimating the posterior density and consequently will produce better metrics such as the mean, variance and confidence intervals.

5.1 Ongoing and Future Research

Further investigations into the performance of the Saddlepoint algorithm for point process models with a variety of intensity functions are being carried out. The analysis in this thesis was limited to scalar processes. However, most of these results can be extended to the multivariable case. Further work is planned on exploring state-space models of dimension greater than one. Also of interest is the application of algorithms based on the Laplace approximation in expectation-maximization algorithms for joint parameter and state estimation.

APPENDIX A

MATLAB Code Used to Simulate State Process

```
clear all;
close all;

timeStep = 0.001;    % msec Resolution in time used in simulating Point process
timeStepEst = 1; % msec Resolution in time used in estimating state process
alpha = 0.99;
endTime = 1000; % msec Duration of simulation
noisevar = 0.001;     % Noise Standard Deviation
noiseSD = noisevar^0.5;
initvar = noisevar/(1-alpha^2);
initSD = initvar^0.5;
IC = 0;  % initial mean


time = [0:timeStep:endTime];
n = max(size(time));
state = zeros(1,n);
init = 0;

%  Generate state processes by sampling CT AR(1) process every
%  microsecond

% generate realization of v(t= -timestep).
init = initSD*rand(1);

% generate realization of v(t=0).
state(1) = alpha*init + noiseSD*randn(1);
% generate realizations of v(t = timestep), ...., v(t = endtime)
for k=2:n,
    state(k) = alpha*state(k-1) + noiseSD*randn(1);
end


save /home/omonoba/work/Laplace_Approximation_Research/datafiles/Results_for_model2/signal2.mat;
```

MATLAB Code Used to Simulate Point Process

```
clear all;
close all;


load
/home/omonoba/work/Laplace_Approximation_Research/datafiles/Results_for_model2/signal2.mat;



uch = -4.9;

% beta is randomly selected in the interval [0.9,1.1]
bch = 0.9 + (1.1-0.9).*rand(1,10);

u = ones(10,n);
b = ones(10,n);
spike = zeros(10,n-1);  % The number of spike bins is one less than the  %number of gridpoints
spikePos = zeros(10,n-1);
spikeTime = zeros(10,n-1);


 u= uch.*u;
 for j=1:10,
     b(j,:)= bch(j).*b(j,:);
 end
 for j=1:10,
     [spike(j,:) spikePos(j,:) spikeTime(j,:)] = simspikes(1, alpha, noiseSD, u(j,:), b(j,:)
,state, timeStep, endTime);
 end


save /home/omonoba/work/Laplace_Approximation_Research/datafiles/Results_for_model2/spikes
```

MATLAB Code Used to calculate spiking in 1msec bins

```
clear all;
close all;

% timeStep = point process is simulated using this resolution
% timeStepEst = Estimation is done using this resolution
%               timeStepEst is the size of the spike bins and is
%               chosen such that the average probability of 2 or more spikes in a
%               bin is very small


load
/home/omonoba/work/Laplace_Approximation_Research/datafiles/Results_for_model2/spikes_model2_10
neurons.mat;


numbins = endTime/timeStepEst;
numSimInEst = timeStepEst/timeStep;
spikebins = zeros(10,numbins);

for l=1:10,
    for k=1:numbins,
        spikebins(l,k) = 0;
        for j=1:numSimInEst,
            spikebins(l,k) = spikebins(l,k) + spike(l,(k-1)*numSimInEst + j );
        end
    end
end


save
/home/omonoba/work/Laplace_Approximation_Research/datafiles/Results_for_model2/spikebins_model2
_10neurons.mat;
```

MEX/MaTLAB function for Implementing SSPPF Algorithm

```
/*
    SSPPF FILTER
*/

#include "/home/omonoba/work/Laplace_Approximation_Research/mexheader.h"
#include "mex.h"
#include "sys/time.h"
#include "unistd.h"
#include "stdlib.h"


#define MAXSPIKES 1e6


/*******************************************************************************
  INTERFACE FUNCTION
  */


void mexFunction(
    int           nlhs,           /* number of expected outputs */
    mxArray       *plhs[],        /* array of pointers to output arguments */
    int           nrhs,           /* number of inputs */
    const mxArray *prhs[])          /* array of pointers to input arguments */
{

    double
*spike,*u,*b,*TimeStep,*endtime,*numgrid,*numspikes,*numused,*IC,*initvar,*alpha,*n
oisevar;
    double *spfmean,*spfvar;
    double    *spikeobs,predMean,*predVar,*innovation;
    double      t,dl_dx;
    double      *lambdaest_k,update_k_mean,update_k_var;

    register int         i,j,k,l,nmb,grid,nmbused;
    register int         tIndex,nIndex, sumIndex;
    register int         outputsize;


      /* Check numbers of arguments */
      if ((nrhs != 12) || (nlhs != 2)) {
        mexErrMsgTxt("Usage: [spfmean,spfvar] =
SSPPF(spike,u,b,timestep,endtime,numgrid,numspikes,numused,IC,initvar,alpha,noiseva
r)" );
    }

    spike = mxGetPr(prhs[0]);
    u = mxGetPr(prhs[1]);
    b = mxGetPr(prhs[2]);
    TimeStep=mxGetPr(prhs[3]);
    endtime=mxGetPr(prhs[4]);
    numgrid = mxGetPr(prhs[5]);
    numspikes = mxGetPr(prhs[6]);
    numused = mxGetPr(prhs[7]);
    IC=mxGetPr(prhs[8]);
    initvar = mxGetPr(prhs[9]);
    alpha=mxGetPr(prhs[10]);
    noisevar=mxGetPr(prhs[11]);

    nmb=numspikes[0];
    nmbused = numused[0];
    grid = numgrid[0];

    /* create the output datastructures */
    outputsize = (int) (endtime[0]/TimeStep[0]);
    mexPrintf("os = %d\n",outputsize);
    plhs[0] = mxCreateDoubleMatrix(1, outputsize, mxREAL);
    spfmean = mxGetPr(plhs[0]);
    plhs[1] = mxCreateDoubleMatrix(1, outputsize, mxREAL);
    spfvar = mxGetPr(plhs[1]);

    predVar =(double*) mxCalloc(outputsize, sizeof(double));
    lambdaest_k =(double*) mxCalloc(nmbused, sizeof(double));
    innovation =(double*) mxCalloc(nmbused, sizeof(double));
    spikeobs =(double*) mxCalloc(nmbused, sizeof(double));



 /*************** Initialize the filter  *****************************/

  predVar[0] = alpha[0]*alpha[0]*initvar[0] +noisevar[0];
  predMean = alpha[0]*IC[0];


/**************Algorithm for t > 0 **********************************/

  for( t=0 ; t<endtime[0]-TimeStep[0] ; t+=TimeStep[0] ) {
      tIndex = round(t/TimeStep[0]);
      nIndex = grid*tIndex;

      for(k=0;k<nmbused;k++) {
          spikeobs[k]=0;
          for(sumIndex = nIndex;sumIndex < nIndex+grid ;sumIndex++) {
              spikeobs[k] += spike[k+nmb*sumIndex];
          }
      }


       /******* SSPPF  Algorithm for Updating one-step gaussian approximation
******/
      if (t<endtime[0]-TimeStep[0]) {
          update_k_mean=0;
```

```
            update_k_var=0;
            for (k=0;k<nmbused;k++)
            {
                lambdaest_k[k]=
exp(u[k+grid*tIndex*nmb]+b[k+grid*tIndex*nmb]*predMean);
                innovation[k]=spikeobs[k]-lambdaest_k[k]*TimeStep[0];
                dl_dx=b[k+grid*tIndex*nmb];
                update_k_mean=update_k_mean+dl_dx*innovation[k];
                update_k_var=update_k_var+dl_dx*dl_dx*lambdaest_k[k]*TimeStep[0];
            }


            spfvar[tIndex]=1/(1/predVar[tIndex]+update_k_var);
            spfmean[tIndex]=predMean + spfvar[tIndex]*update_k_mean;

            predVar[tIndex+1] = alpha[0]*alpha[0]*spfvar[tIndex] + noisevar[0];
            predMean = alpha[0]*spfmean[tIndex];
        }

    }


    mxFree(predVar);
    mxFree(lambdaest_k);
    mxFree(spikeobs);
    return;
}
```

MEX/MATLAB Code Used to Implement Saddlepoint Filter

```
/*
    myfilter3ver2.c
    - Filter Based on Laplace Approximation for "non-asymptotic integrals
    - Estimates the posterior density at a given time point
    - Also return estimates of means and variances up to given time point
    - Model: multiple independent point processes modulated by the same latent
    - state process
 */

#include "/home/omonoba/work/Laplace_Approximation_Research/mexheader.h"
#include "mex.h"
#include "sys/time.h"
#include "unistd.h"
#include "stdlib.h"


#define MAXSPIKES 1e6
#define PI 3.1415926


/*******************************************************************************
 * INTERFACE FUNCTION
 */


void mexFunction(
int             nlhs,           /* number of expected outputs */
mxArray         *plhs[],        /* array of pointers to output arguments */
int             nrhs,           /* number of inputs */
const mxArray *prhs[])           /* array of pointers to input arguments */
{

    double *spikeobs,*u,*b,*TimeStep,*stoptime,*initmean,*initSD,*alpha,*noiseSD;
    double
*xmin,*xmax,*xstep,*numUsed,*numTotalNeurons,*mypdf,*post_mean,*post_var;
    double mean_prev,var_prev,argmax_curr,
d2log_curr,max_transpdf,x_low,x_high,index_helper, oldpdf_interpol;
    double      *newpdf, *oldpdf;
    double      p1,p2,a1;
    double      x, xold, t;
    double      *lambda, noisevar;
    double      transpdf,updatepdf, spikepdf;
    double      meansum, varsum,norm;

    register int    k, xIndex, oldIndex,
tIndex,zIndex_low,zIndex_high,numused,numtotal;
    int  pdfsize,outputsize;
```

```
      /* Check numbers of arguments */
    if ((nrhs != 14) || (nlhs != 3)) {
        mexErrMsgTxt("Usage: [mypdf,post_mean,post_var] =
myfilter3ver2(spikeobs,u,b,timestep,stoptime,initmean,initSD,alpha,noiseSD,xmin,xma
x,xstep,numUsed,numTotalNeurons)" );
    }

    spikeobs = mxGetPr(prhs[0]);
    u = mxGetPr(prhs[1]);
    b = mxGetPr(prhs[2]);
    TimeStep=mxGetPr(prhs[3]);
    stoptime=mxGetPr(prhs[4]);
    initmean=mxGetPr(prhs[5]);
    initSD = mxGetPr(prhs[6]);
    alpha=mxGetPr(prhs[7]);
    noiseSD=mxGetPr(prhs[8]);
    xmin=mxGetPr(prhs[9]);
    xmax=mxGetPr(prhs[10]);
    xstep=mxGetPr(prhs[11]);
    numUsed=mxGetPr(prhs[12]);
    numTotalNeurons=mxGetPr(prhs[13]);

    numused = numUsed[0];
    numtotal = numTotalNeurons[0];

    /* create the output datastructures */
    outputsize = (int) (stoptime[0]/TimeStep[0]) + 1;
    pdfsize = (int) (xmax[0]/xstep[0]-xmin[0]/xstep[0])+1;
    plhs[0] = mxCreateDoubleMatrix(1, pdfsize, mxREAL);
    mypdf = mxGetPr(plhs[0]);
    plhs[1] = mxCreateDoubleMatrix(1, outputsize, mxREAL);
    post_mean = mxGetPr(plhs[1]);
    plhs[2] = mxCreateDoubleMatrix(1, outputsize, mxREAL);
    post_var = mxGetPr(plhs[2]);

    /* Helper Variables */
    newpdf = (double *) mxCalloc(pdfsize, sizeof(double));
    oldpdf = (double *) mxCalloc(pdfsize, sizeof(double));
    lambda = (double*) mxCalloc(numused, sizeof(double));


  noisevar = noiseSD[0]*noiseSD[0];

 /*************** Initialize the filter  *****************************/

   /* generate the initial density at t = - timestep */
    for ( x=xmin[0] ; x <=xmax[0] ; x+=xstep[0] ) {
        xIndex = round(x/xstep[0]-xmin[0]/xstep[0]);
        oldpdf[xIndex] = 1/sqrt(2*PI)/initSD[0] * exp( -(x-initmean[0])*(x-
initmean[0])/2/initSD[0]/initSD[0] );
    }
    mean_prev = initmean[0];
    var_prev = initSD[0]*initSD[0];


/**************Algorithm for t >= 0 **********************************/

    /* stoptime must be strictly less than the endtime of the simulation
       of the state space model */
    for( t=0 ; t<=stoptime[0] ; t+=TimeStep[0] ) {
        tIndex = round(t/TimeStep[0]);

        norm = 0;
        meansum = 0;
        varsum = 0;
       /******* Estimate Posterior Density at t  ******/
        for ( x=xmin[0] ; x <= xmax[0] ; x+=xstep[0] ) {
            xIndex = round(x/xstep[0]-xmin[0]/xstep[0]);

            /* Calculate Approx Value for the  arg max of the integrand and */
            /* Form Laplace Approximation to the One-Step Density */
            d2log_curr = ((1/var_prev) + (alpha[0]/noisevar));
            argmax_curr = ((mean_prev/var_prev) + (x/noisevar))/d2log_curr;
            max_transpdf = 1/sqrt(2*PI)/noiseSD[0]*exp( -(x-
alpha[0]*argmax_curr)*(x-alpha[0]*argmax_curr)/2/noisevar );
            /* get previous posterior density at the argmax using linear
interpolation */
            if (argmax_curr < xmin[0])
            {
                 zIndex_low = 0;
                 zIndex_high = 0;
                }
                else if (argmax_curr > xmax[0])
                {
                 zIndex_high = round((xmax[0]- xmin[0])/xstep[0]);
                 zIndex_low = zIndex_high;
                }
                else
                {
                index_helper = (argmax_curr-xmin[0])/xstep[0];
                zIndex_low = (int) (index_helper);
                zIndex_high = zIndex_low+1;
                }

                p1 = oldpdf[zIndex_low];
                p2 = oldpdf[zIndex_high];
                a1  = xmin[0]+zIndex_low*xstep[0];
                oldpdf_interpol = p1 + ((argmax_curr-a1)/xstep[0])*(p2-p1);
                updatepdf = sqrt((2*PI)/d2log_curr)*max_transpdf*oldpdf_interpol;

                /* Update the posterior density, posterior mean and variance estimates
*/
                spikepdf = 1.0;
                for (k=0;k<numused;k++)
                {
                lambda[k] = exp(u[0]+ b[k]*x);
                spikepdf *= exp(-lambda[k]*TimeStep[0]);
                if (spikeobs[k+tIndex*numtotal]) spikepdf *= lambda[k]*TimeStep[0];
                }
                newpdf[xIndex] = updatepdf*spikepdf; /* Posterior Density up to scaling
(numerator of Bayes expression) */
                norm += newpdf[xIndex]*xstep[0];
                meansum += x*newpdf[xIndex]*xstep[0];
            }

            post_mean[tIndex] = meansum/norm;
            mean_prev = post_mean[tIndex];

            for ( x=xmin[0] ; x <=xmax[0] ; x+=xstep[0] ) {
                xIndex = round(x/xstep[0]-xmin[0]/xstep[0]);
                newpdf[xIndex] /= norm;
                oldpdf[xIndex] = newpdf[xIndex];
                mypdf[xIndex] = newpdf[xIndex];
                varsum += (x - post_mean[tIndex])*(x -
post_mean[tIndex])*newpdf[xIndex]*xstep[0];
            }

            post_var[tIndex] = varsum;
            var_prev = post_var[tIndex];
        }


        mxFree(newpdf);
        mxFree(oldpdf);
        mxFree(lambda);
        return;
}
```

MEX/MATLAB Code For Generating Realization of Point Process Using Time-Rescaling Theorem

```
/*
    Function: simspikes.c (MEX FILE)
    Simulate a realization of a Stochastic Point Process with
    intensity function of the form lambda(t) = exp( u + b*x(t) )
    where u and b are constants and
    x(t) is a continuous time AR(1) process.
    A Time interval, delta, is chosen such that the probability of
    2 or more spikes in a time interval of lenght delta is very small.
    The CT AR(1) process is then "sampled" at delta-spaced intervals
    and the samples are used to generate a realization of the
    point process using the time-rescaling theorem

*/

#include "/home/omonoba/work/Laplace_Approximation_Research/mexheader.h"
```

```
#include "mex.h"
#include "sys/time.h"
#include "unistd.h"
#include "stdlib.h"

#define MAXSPIKES 1e6


/*****************************************************************************
  INTERFACE FUNCTION
  */


void mexFunction(
    int           nlhs,           /* number of expected outputs */
    mxArray       *plhs[],        /* array of pointers to output arguments */
    int           nrhs,           /* number of inputs */
    const mxArray *prhs[])         /* array of pointers to input arguments */
{

    double      *spike,*spikePos,*spikeTime;
    double      *numinputs;
    double      *alpha;
    double      *sigSD;
    double      *x;
    double      *u;
    double      *b;
    double      *timestep;
    double      *endtime;
    double      *lambda;
    double      lambdaInt;
    double      lambdaEst;
    double      innovation;
    double      t,t_int;
    mxArray     *rhs1[1],*lhs1[1];
    double      *z,*mu;

    int         j, k;
    int         tIndex,nIndex, oIndex;
    int         outputsize;

      /* Check numbers of arguments */
      if ((nrhs != 8) || (nlhs != 3)) {
 mexErrMsgTxt("Usage: [spike spikePos spikeTime] = simspikes(numinputs, alpha,
sigSD, u, b,x, timestep, endtime)" );
    }

    numinputs = mxGetPr(prhs[0]);
    alpha = mxGetPr(prhs[1]);
    sigSD = mxGetPr(prhs[2]);
    u = mxGetPr(prhs[3]);
    b = mxGetPr(prhs[4]);
    x=mxGetPr(prhs[5]);
    timestep = mxGetPr(prhs[6]);
    endtime = mxGetPr(prhs[7]);


    /* create the output datastructures */
    outputsize = (int) (endtime[0]/timestep[0]);
    mexPrintf("os = %d\n",outputsize);
    plhs[0] = mxCreateDoubleMatrix(numinputs[0], outputsize,  mxREAL);
    spike = mxGetPr(plhs[0]);
    plhs[1] = mxCreateDoubleMatrix(numinputs[0], outputsize,  mxREAL);
    spikePos=mxGetPr(plhs[1]);
    plhs[2] = mxCreateDoubleMatrix(numinputs[0], outputsize,  mxREAL);
    spikeTime=mxGetPr(plhs[2]);


    lambda = (double *) mxCalloc(outputsize*numinputs[0], sizeof(double));

    rhs1[0] = mxCreateDoubleMatrix(1, 1, mxREAL);
    mu =   mxGetPr(rhs1[0]);
    mu[0]=1;


    for ( j=0 ; j<numinputs[0] ; j++ )
    {
        t = 0;
 while (round(t) < round(endtime[0]))
 {
     lambdaInt = 0;
     mexCallMATLAB(1, lhs1, 1, rhs1, "exprnd");
     z = mxGetPr(lhs1[0]);

     while ((lambdaInt < z[0])  && (round(t) < round(endtime[0]-timestep[0])))
     {
     t += timestep[0];
        tIndex = (int) (t/timestep[0]);
        lambda[j+(int) numinputs[0]*tIndex] = exp(u[j+(int)
numinputs[0]*tIndex]+b[j+(int) numinputs[0]*tIndex]*x[tIndex]);
        lambda[j+(int) numinputs[0]*(tIndex-1)] = exp(u[j+(int)
numinputs[0]*(tIndex-1)]+b[j+(int) numinputs[0]*(tIndex-1)]*x[tIndex-1]);
        lambdaInt += 0.5*lambda[j+(int) numinputs[0]*tIndex]*timestep[0];
        lambdaInt += 0.5*lambda[j+(int) numinputs[0]*(tIndex-1)]*timestep[0];
     }

     if (t < endtime[0])
     {
        spike[j + (int) numinputs[0]*(tIndex-1)] = 1;
               spikeTime[j + (int) numinputs[0]*(tIndex-1)]=t-timestep[0];
               spikePos[j+ (int) numinputs[0]*(tIndex-1)]=x[tIndex-1];
     }

        }
   }

    mxFree(lambda);
    return;
}
```

MEX/MaTLAB function for calculating Exact Posterior Density at a given Timepoint

```
/*
    instantexactPDF.c -
    plots exact posterior density at a specified time point
*/

#include "/home/tzvi/Spline/mexheader.h"
#include "mex.h"
#include "sys/time.h"
#include "unistd.h"

#define MAXSPIKES 1e6
#define PI 3.1415926

/******************************************************************************
  INTERFACE FUNCTION
  */

void mexFunction(
    int           nlhs,           /* number of expected outputs */
    mxArray       *plhs[],        /* array of pointers to output arguments */
    int           nrhs,           /* number of inputs */
    const mxArray *prhs[])         /* array of pointers to input arguments */
{
    double      *pdf,*numb,*numbused,*numgrid;
    double      *spike,*spikeobs;
    double      *alpha, *sigSD;
    double      *xmin, *xmax, *spacestep;
    double      *u, *b;
    double      *timestep, *endtime;
    double      *initvar, *initmean;
    double      *newpdf, *oldpdf;
    double      x, xold, t;
    double      *lambda;
    double      CKint;
    double      transmean, transSD, transpdf, spikepdf;
    double      meansum, varsum;
    double      norm;
    int         xIndex, oldIndex, tIndex,nIndex,sumIndex;
    int         outputsize, pdfsize,num,grid,numused,target_time;

    register int k;
    /* Check numbers of arguments */
    if ((nrhs != 15) || (nlhs != 1)) {
 mexErrMsgTxt("Usage: [pdf] = instantexactPDF(alpha, sigSD, u, b, timestep,
endtime, xmin, xmax, spacestep, spike, initmean, initvar,numb,numgrid,numbused)" );
    }

    alpha = mxGetPr(prhs[0]);
    sigSD = mxGetPr(prhs[1]);
    u = mxGetPr(prhs[2]);
    b = mxGetPr(prhs[3]);
    timestep = mxGetPr(prhs[4]);
    endtime = mxGetPr(prhs[5]);
    xmin = mxGetPr(prhs[6]);
    xmax = mxGetPr(prhs[7]);
    spacestep = mxGetPr(prhs[8]);
    spike = mxGetPr(prhs[9]);
    initmean = mxGetPr(prhs[10]);
    initvar = mxGetPr(prhs[11]);
    numb = mxGetPr(prhs[12]);
    numgrid = mxGetPr(prhs[13]);
    numbused = mxGetPr(prhs[14]);

    /* create the output datastructures */
    pdfsize = (int) (xmax[0]/spacestep[0]-xmin[0]/spacestep[0])+1;
    plhs[0] = mxCreateDoubleMatrix(1, pdfsize, mxREAL);
    pdf = mxGetPr(plhs[0]);

    num = numb[0];
    numused = numbused[0];
    grid = numgrid[0];
    newpdf = (double *) mxCalloc(pdfsize, sizeof(double));
    oldpdf = (double *) mxCalloc(pdfsize, sizeof(double));
    lambda = (double *) mxCalloc(numused, sizeof(double));
    spikeobs = (double *) mxCalloc(numused, sizeof(double));

    /* Initialize the pdf to the right value */
    for ( x=xmin[0] ; x <=xmax[0] ; x+=spacestep[0] ) {
        xIndex = round(x/spacestep[0]-xmin[0]/spacestep[0]);
        newpdf[xIndex] = 1/sqrt(2*PI*initvar[0]) * exp( -(x-initmean[0])*(x-
initmean[0])/2/initvar[0]  );
        oldpdf[xIndex] = newpdf[xIndex];
    }

    for ( t=0 ; t<=endtime[0] ; t+=timestep[0] ) {
        tIndex = round(t/timestep[0]);
        nIndex = grid*tIndex;

        for(k=0;k<numused;k++) {
            spikeobs[k]=0;
            for(sumIndex = nIndex;sumIndex < nIndex+grid ;sumIndex++) {
                spikeobs[k] += spike[k+num*sumIndex];
            }
        }

        norm = 0;
        meansum = 0;
        varsum = 0;

        for ( x=xmin[0] ; x <=xmax[0] ; x+=spacestep[0] ) {
            xIndex = round(x/spacestep[0]-xmin[0]/spacestep[0]);
            for(k=0;k<numused;k++){
                lambda[k] = exp(u[k+num*grid*tIndex]+b[k+num*grid*tIndex]*x);
            }
            spikepdf = 1.0;
            for(k=0;k<numused;k++){
                spikepdf *= exp(-lambda[k]*timestep[0]);
                if (spikeobs[k]) spikepdf *= lambda[k]*timestep[0];
            }

            CKint = 0;
            for ( xold=xmin[0] ; xold <=xmax[0] ; xold+=spacestep[0] ) {
                oldIndex = round(xold/spacestep[0]-xmin[0]/spacestep[0]);
                transmean = alpha[0]*xold;
                transSD = sigSD[0];
                transpdf = 1/sqrt(2*PI)/transSD * exp( -(x-alpha[0]*xold)*(x-
alpha[0]*xold)/2/transSD/transSD  );
                CKint += transpdf*oldpdf[oldIndex]*spacestep[0];
            }
            newpdf[xIndex] = spikepdf*CKint;
            norm += newpdf[xIndex]*spacestep[0];
        }
        for ( x=xmin[0] ; x <=xmax[0] ; x+=spacestep[0] ) {
            xIndex = round(x/spacestep[0]-xmin[0]/spacestep[0]);
            newpdf[xIndex] /= norm;
            oldpdf[xIndex] = newpdf[xIndex];
            pdf[xIndex] = newpdf[xIndex];
        }

    }


    mxFree(newpdf);
    mxFree(oldpdf);
    mxFree(lambda);
    mxFree(spikeobs);

    return;
}
```

MEX/MaTLAB function for calculating Exact Mean and Variance For the Point Process model

```
/*
    exactmeanvar.c - Plots Exact Mean and Variance For the Point
                     Process model
 */

#include "/home/omonoba/work/Laplace_Approximation_Research/mexheader.h"
#include "mex.h"
#include "sys/time.h"
#include "unistd.h"

#define MAXSPIKES 1e6
#define PI 3.1415926

/*******************************************************************************
 * INTERFACE FUNCTION
 */

void mexFunction(
int             nlhs,           /* number of expected outputs */
mxArray         *plhs[],        /* array of pointers to output arguments */
int             nrhs,           /* number of inputs */
const mxArray *prhs[])           /* array of pointers to input arguments */
{
    double      *pdfmean, *pdfvar;
    double      *spike;
    double      *numinputs;
    double      *alpha, *sigSD;
    double      *xmin, *xmax, *spacestep;
    double      *u, *b;
    double      *timestep, *endtime;
    double      *initvar, *initmean,*numb,*numbused,*numbgrid;
    double      *newpdf, *oldpdf;
    double      x, xold, t;
    double      *lambda,*spikeobs;
    double      CKint;
    double      transmean, transSD, transpdf, spikepdf;
    double      meansum, varsum;
    double      norm;
    int         xIndex, oldIndex, tIndex,sumIndex,k,grid,num,numused,nIndex;
    int         outputsize, pdfsize;

      /* Check numbers of arguments */
    if ((nrhs != 15) || (nlhs != 2)) {
        mexErrMsgTxt("Usage: [pdfmean pdfvar] = exactmeanvar(alpha, sigSD, u, b,
timestep, endtime, xmin, xmax, spacestep, spike, initmean,
initvar,numb,numbused,numbgrid)" );
    }

    alpha = mxGetPr(prhs[0]);
    sigSD = mxGetPr(prhs[1]);
    u = mxGetPr(prhs[2]);
    b = mxGetPr(prhs[3]);
    timestep = mxGetPr(prhs[4]);
    endtime = mxGetPr(prhs[5]);
    xmin = mxGetPr(prhs[6]);
    xmax = mxGetPr(prhs[7]);
    spacestep = mxGetPr(prhs[8]);
    spike = mxGetPr(prhs[9]);
    initmean = mxGetPr(prhs[10]);
    initvar = mxGetPr(prhs[11]);
    numb = mxGetPr(prhs[12]);
    numbused = mxGetPr(prhs[13]);
    numbgrid = mxGetPr(prhs[14]);

    /* create the output datastructures */
    outputsize = (int) (endtime[0]/timestep[0]);
    pdfsize = (int) (xmax[0]/spacestep[0]-xmin[0]/spacestep[0])+2;
    plhs[0] = mxCreateDoubleMatrix(1, outputsize, mxREAL);
    pdfmean = mxGetPr(plhs[0]);
    plhs[1] = mxCreateDoubleMatrix(1, outputsize, mxREAL);
    pdfvar = mxGetPr(plhs[1]);

    num = numb[0];
    grid = numbgrid[0];
    numused = numbused[0];

    newpdf = (double *) mxCalloc(pdfsize, sizeof(double));
    oldpdf = (double *) mxCalloc(pdfsize, sizeof(double));
    lambda = (double *) mxCalloc(num, sizeof(double));
    spikeobs = (double *) mxCalloc(num, sizeof(double));

    /* Initialize the pdf to the right value */
    for ( x=xmin[0] ; x <=xmax[0] ; x+=spacestep[0] ) {
        xIndex = round(x/spacestep[0]-xmin[0]/spacestep[0]);
        newpdf[xIndex] = 1/sqrt(2*PI*initvar[0]) * exp( -(x-initmean[0])*(x-
initmean[0])/2/initvar[0]  );
        oldpdf[xIndex] = newpdf[xIndex];
    }

    for ( t=0 ; t<endtime[0] ; t+=timestep[0] ) {
        tIndex = round(t/timestep[0]);
        nIndex = grid*tIndex;

        for(k=0;k<numused;k++) {
            spikeobs[k]=0;
            for(sumIndex = nIndex;sumIndex < nIndex+grid ;sumIndex++) {
                spikeobs[k] += spike[k+num*sumIndex];
            }
        }


        norm = 0;
        meansum = 0;
        varsum = 0;

        for ( x=xmin[0] ; x <=xmax[0] ; x+=spacestep[0] ) {
            xIndex = round(x/spacestep[0]-xmin[0]/spacestep[0]);
            for(k=0;k<numused;k++){
                lambda[k] = exp(u[k+num*grid*tIndex]+b[k+num*grid*tIndex]*x);
            }
            spikepdf = 1.0;
            for(k=0;k<numused;k++){
                spikepdf *= exp(-lambda[k]*timestep[0]);
                if (spikeobs[k]) spikepdf *= lambda[k]*timestep[0];
            }
            CKint = 0;
            for ( xold=xmin[0] ; xold <=xmax[0] ; xold+=spacestep[0] ) {
                oldIndex = round(xold/spacestep[0]-xmin[0]/spacestep[0]);
          /*  transmean = exp(-alpha[0]*timestep[0])*xold;
        transSD = sqrt((1-exp(-
2*alpha[0]*timestep[0]))*sigSD[0]*sigSD[0]/2/alpha[0]); */
                transSD = sigSD[0];
                transmean = alpha[0]*xold;
                transpdf = 1/sqrt(2*PI)/transSD * exp( -(x-transmean)*(x-
transmean)/2/transSD/transSD  );
                CKint += transpdf*oldpdf[oldIndex]*spacestep[0];
            }
            newpdf[xIndex] = spikepdf*CKint;
            norm += newpdf[xIndex]*spacestep[0];
            meansum += x*newpdf[xIndex]*spacestep[0];
            if ( (tIndex == 200) || (tIndex == 400) || (tIndex == 600) || (tIndex
== 800) ) mexPrintf("x = %f, E = %f\n",x,spikepdf*CKint);
        }
        pdfmean[tIndex] = meansum/norm;
        if ( (tIndex == 200) || (tIndex == 400) || (tIndex == 600) || (tIndex ==
800) ) mexPrintf("norm = %f, mean = %f\n",norm,pdfmean[tIndex]);
        for ( x=xmin[0] ; x <=xmax[0] ; x+=spacestep[0] ) {
            xIndex = round(x/spacestep[0]-xmin[0]/spacestep[0]);
            newpdf[xIndex] /= norm;
            oldpdf[xIndex] = newpdf[xIndex];
            varsum += (x - pdfmean[tIndex])*(x -
pdfmean[tIndex])*newpdf[xIndex]*spacestep[0];
        }

        pdfvar[tIndex] = varsum;

    }

    mxFree(newpdf);
    mxFree(oldpdf);
    mxFree(lambda);
    mxFree(spikeobs);
    return;
}
```

References

Anderson, B.D.O., & Moore, J.B., (1979). *Optimal Filtering*. New Jersey: Prentice-Hall.

Barndorff-Nielsen, O.E. (1983). On a formula for the distribution of the maximum likelihood estimator. *Biometrika*, 70, 2, 343-365.

Brown, E.N., Barbieri R., Ventura V., Kass, R.E., & Frank L.M. (2001). The time-rescaling theorem and its application to neural spike train data analysis. *Neural Computation*, 14, 325-346.

Brown, E.N., Barbieri R., Eden U.T., & Frank L.M. (2003). Likelihood methods for neural data analysis. In: J. Feng (Ed.), *Computational neuroscience: A comprehensive approach* (pp. 253-286). London: CRC.

Brown, E.N. (2005). Theory of Point Processes for Neural Systems. In: Chow CC, Gutkin B, Hansel D, Meunier C, Dalibard J, (Eds.) *Methods and Models in Neurophysics*. Paris, Elsevier, Chapter 14, pp. 691-726.

Daley, D., & Vere-Jones, D. (1998). *An introduction to the theory of point processes*. New York: Springer-Verlag.

Davison, A.C. (1986). Approximate predictive likelihood. *Biometrika*, 73, 2, 323-332.

De Bruijn, N.G. (1981). *Asymptotic methods in analysis*. New York:Dover.

Eden, U.T., Frank, L.M., Solo, V., & Brown, E.N. (2004). Dynamic analysis of neural encoding by point process adaptive filtering. *Neural Computation*, 16, 971-998.

Eden, U.T. (2005). *Point process filters in the analysis of neural spiking models*. Unpublished Thesis Dissertation.

Julier, S.J., & Uhlmann, J.K. (1994). A general method for approximating nonlinear transformations of probability distributions.

Julier, S.J., Uhlmann, J.K., & Durrant-Whyte, H.F. (1996). A new approach for the nonlinear transformation of means and covariances in linear filters. *IEEE Transactions on Automatic Control.*

Julier, S.J., & Uhlmann, J.K. (1997). A new extenstion of the Kalman filter to nonlinear systems, Proc. of Aerosense: *The 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls, Orlando, Florida*, Vol. Multi Sensor Fusion, Tracking and Resource Management II.

Kailath, T., Sayed, A.H., & Hassibi, B. (2000). *Linear Estimation.* Upper Saddle River, NJ: Prentice Hall.

Laha, R.G. & Rohatgi V.K. (1979). *Probability Theory.*

Luenberger, D.G. (1969). *Optimization by vector space methods.* New York: John Wiley & Sons.

Reid, N. (1988). Saddlepoint methods and statistical inference. *Statistical Science*, 3, 2, 213-238.

Reid, N. (1996). Asymptotic Expansions. *Encyclopedia of Statistical Sciences* .

Ross, S.M. (1996). *Stochastic Processes.* New York: John Wiley & Sons.

Rudin, W. (1976). *Principles of Mathematical Analysis.* New York: McGraw-Hill

Smith, A.C., & Brown, E.N. (2003). Estimating a state-space model from point process observations. *Neural Computation*, 15, 965-991.

Snyder, D., & Miller, M. (1991). *Random point processes in time and space.* New York: Springer-Verlag.

Tanner, M.A. (1993). *Tools for statistical inference*, 2nd ed. New York, NY: Springer-Verlag.

Tierney, L., & Kadane, J.B. (1986). Accurate approximations for posterior moments and marginal densities. *Journal of the American Statistical Association*, 81, 393, 82-86.

Tierney, L., Kass R.E., & Kadane, J.B. (1989). Approximate marginal densities of nonlinear functions. *Biometrika*, 76, 3, 425-433.