Estimation Problems in Sense and Respond Systems

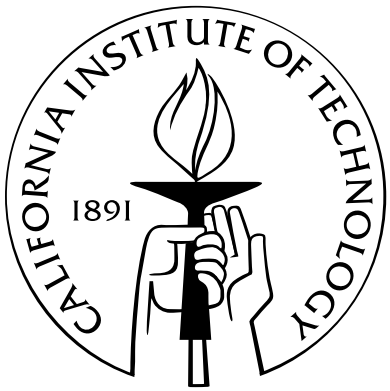
Thesis by

Agostino Capponi

In Partial Fulfillment of the Requirements

for the Degree of

Master of Science



California Institute of Technology

Pasadena, California

2006

(Submitted May 25, 2006)

© 2006
Agostino Capponi
All Rights Reserved

To my cousin Francesco.

Acknowledgements

I would like to express my gratitude to my advisor, Prof. Mani Chandy, for believing in me and giving me the wonderful opportunity to work with him. I would like to thank him for his constant support and guidance during my first two years at the California Institute of Technology. It has been a great pleasure to work under his supervision. I would also like to thank Prof. Ibrahim Fatkullin for useful discussions, suggestions and sharing some of his ideas. I consider myself very fortunate to work in the highly stimulating intellectual atmosphere of the Department of Computer Science at the California Institute of Technology. I want to thank Dr. Daniel Zimmerman for carefully reviewing the draft of my thesis and providing insightful comments. I am also grateful to other members of the Infospheres Lab including Andrey Khorlin and Lu Tian for creating a pleasant working environment that helped me during my first two years of studies at Caltech. I want to thank my parents for sacrificing some of their pleasure in order to pay for my studies and bring me up in the best traditions of a creative environment. I want to thank Concetta Pilotto for her love and support. Last, but not least, I would like to thank the SISL grants that supported my research during these first two years. This work was funded in part by the National Science Foundation under ITR grant, CCR-0312778 and also by the Social and Information Sciences Laboratory at the California Institute of Technology.

Abstract

In this thesis we study problems arising in the design of sense and respond systems and present analytical solutions to them as well as results from experiments dealing with real systems. Sense and respond systems employ sensors and other sources of data to sense what is happening in their environments, process the obtained information, and respond appropriately. A goal of the processing stage is to reconstruct the best possible estimate of the state of the environment using messages received from sensors. Due to the large number of messages that need to be processed, it is desirable to have algorithms that can incrementally process the received measurements and recover the state. The state estimation process becomes more problematic if measurements obtained from the sensors are noisy or they are sent at unpredictable times. First, we study models of state estimation and present algorithms that can incrementally compute accurate linear state estimates of the surrounding environment. Second, we define a framework called *predicate signaling* that allows us to make tradeoffs between message generation rates and the quality of the state estimate through specification of suitable predicates. We show how predicate signaling generalizes commonly used signaling schemes and present a detailed analysis based on stochastic processes to evaluate schemes based on predicate signaling.

Contents

List of Figures

List of Tables

Introduction to Sense and Respond Systems

A *sense and respond system* (1) estimates the history of global states of the environment from information in streams of events and other data, (2) detects critical conditions—threats or opportunities—by analyzing this history, and (3) responds in a timely fashion to these conditions. The term sense and respond system was introduced by Haeckel and Rolan [13, 14]. The essential characteristic of a sense and respond system is that it can be defined by a set of *when-then* rules of the following form: *when* a condition holds *then* respond in the prescribed way. The designs of *when-then* rules vary widely depending on the application being considered. For example, in the context of homeland security the *when* clause may be the detection of an intruder in a monitored area and the *then* clause may be the fast engagement of the threat.

An important problem in sense and respond systems is minimizing the amount of time needed for evaluating the *when* clause and executing the *then* clause. For example, an intrusion detection application, may fuse information coming from different sensors in order to predict the presence of an intruder in the area. If the time needed to reach a decision becomes too high, the execution of the *then* clause may be useless since the intruder may have already caused serious damage. Similarly, the execution of the *then* clause should be efficient. Consider, for example, path planning for an autonomous vehicle. The optimal path is computed initially using a given set of data. If the *when* clause detects a change in this data set, it might be possible to recompute the optimal path given the new change in the data set very rapidly without having to recompute everything from scratch. This example suggests the importance of incremental algorithms for processing *when-then* rules.

It is impossible to define a general accurate evaluation measure for sense and respond systems. This is because there are many different ways of specifying and evaluating clauses. Sense and respond systems offer a theoretical framework for a wide range of applications. However, it is possible to identify some basic evaluation criteria which are common to most of the applications.

(1) The cost of a *false positive*, *i.e.*, the cost of executing a response to a condition that did not arise. In terms of *when-then* rules, this is the cost of executing the *then* clause while the *when* clause is false. An example is the cost of attacking an innocuous object erroneously determined

to be an intruder.

(2) The costs of a *false negative*, *i.e.*, the cost of not executing a response to a condition that did arise. In terms of *when-then* rules, this is the cost of not executing the *then* clause while the *when* clause is true. An example is the cost of not engaging a dangerous intruder erroneously determined to be innocuous.

(3) The *detection delay*, *i.e.*, the time between the instant the condition in the *when* clause becomes true and the instant the system detects that it is true.

(4) The *response delay*, *i.e.*, the time between the detection of the condition in the *when* clause and the execution of the response action defined in the *then* clause.

Depending on the application, it may be necessary to include other evaluation criteria; for example, it may be appropriate to measure the effectiveness of a response action. Suppose that, in an intrusion detection application, the objective of the response is to engage an intruder. We want to define parameters measuring how successful the engagement mission has been. Did the mission completely neutralize the intruder or did it only partially accomplish the task? Defining these measures require careful consideration and strongly depends on the domain of the sense and respond application.

The design of sense and respond systems is simplified by separating two problems: the detection of the critical condition (corresponding to the evaluation of the *when* clause) and the execution of the response (corresponding to the execution of the *then* clause). This thesis addresses both of these problems. The remainder of the thesis is organized as follows. Part I studies the problem of detecting changes in time-varying statistical models (*when* clause) and rapidly executing responses to the detected changes (*then* clause). We consider linear statistical models and use the total least square method as an estimation procedure. Part II focuses on the problem of reducing the amount of communication needed to signal that a condition defined in a *when* clause has begun to hold. Such problems are important for applications such as sensor networks, where many sensors with limited communication capabilities are deployed to detect anomalous conditions. If communication is not properly controlled such sensors may run out of battery power.

Part I

Stream Processing Algorithm for Change Detection and Adaptation

Chapter 1

Introduction

System behaviors vary over time. For example, an information network varies from heavily loaded to lightly loaded conditions; patterns of incidence of disease change at the onset of pandemics; file access patterns change from proper usage to improper usage that may signify insider threat. The models that represent behavior need to be updated frequently to reflect such changes; in the limit, models need to be updated with each new event. Usually the model used is unknown and needs to be estimated on the basis of received measurements. This chapter discusses algorithms that fuse information in multiple event streams with the objective of estimating the most recent model describing system behavior.

Algorithms that adapt to changes in behavior may depend on a appropriate amount of history: those that give too much weight to the distant past will not adapt to changes in behavior rapidly; those that don't consider enough past information may conclude incorrectly, from noisy data, that behavior has changed when it has not. Efficient algorithms are incremental; the computational time required to incorporate each new event should be small and ideally independent of the length of the history. This part illustrates how incremental algorithms can be efficiently derived for the case when the model is linear and the estimation procedure used is the total least square method.

1.1 Overview

A sense and respond system learns about its environment from information in event streams and other data sources. The environment is represented by a model, and algorithms continuously estimate models as new events arrive on event streams. The learned model is used to determine the best response to critical conditions.

In some problem areas, critical conditions are signaled by changes in behavior of a system. Information assurance systems monitor applications, such as email, and usage of information assets, such as files, to generate alerts when changes in behavior that may signal misuse are detected. Financial applications detect potential non-compliance with regulations by monitoring changes in

patterns of income and expenditure. Pharmaceutical companies monitor changes in patterns of side effects reported by customers to detect potential problems with medicines.

These applications develop and continuously update models of system behavior. As system behavior changes, model parameters change too; significant changes in parameters indicate probable changes in behavior. The systems of interest consist of groups of entities. For instance, in the pharmaceutical example, the system consists of all customers who have bought a product, and the events in the system are activities by customers such as logging of complaints or indications of satisfaction. The system generates a stream of events: the sequence of events generated by all the customers collectively. Successive events may be generated by different entities; for example, a complaint registered by one customer may be followed by complaints from many other customers before an event is generated by the first customer again.

1.2 Model of Behavior

A signal is represented by a point in a multidimensional space where the dimensions are attributes of behavior. The dimensions in a pharmaceutical example dealing with blood sugar monitors include the age of the product, the strength of the battery, the type of erroneous reading, length of experience with this type of product, and so on. Our algorithm is fed a stream of signals (sometimes called event information) and thus is continuously fed new points in this space. A model is a surface in this space, and a metric of the fitness of the model is the average mean-square distance of points from the surface.

The system may change its behavior, and this change may be gradual or abrupt. In the blood sugar monitor example, a change may be caused by the introduction of a defective component in some batches of the product. The signals that are generated after the change reflect the changed behavior. The algorithm updates its estimate of the model parameters each time it receives a signal with the goal of maintaining an accurate model at all times. Fig. 1.1 illustrates a changing behavior in 3–dimensional space. Points are generated on the one surface before the change and on the other surface after the change.

1.3 Incremental Algorithms for Stream Processing

A stream processing algorithm takes a sequence of events as its inputs and scans this sequence only once in order of event occurrence [4, 19]. The computational complexity measures are the space used by the algorithm and the time required to process each new event. An *incremental* algorithm is one for which the time required to fuse a single event with the history of events is small compared with the length of the history; we seek algorithms in which the time is independent of the length

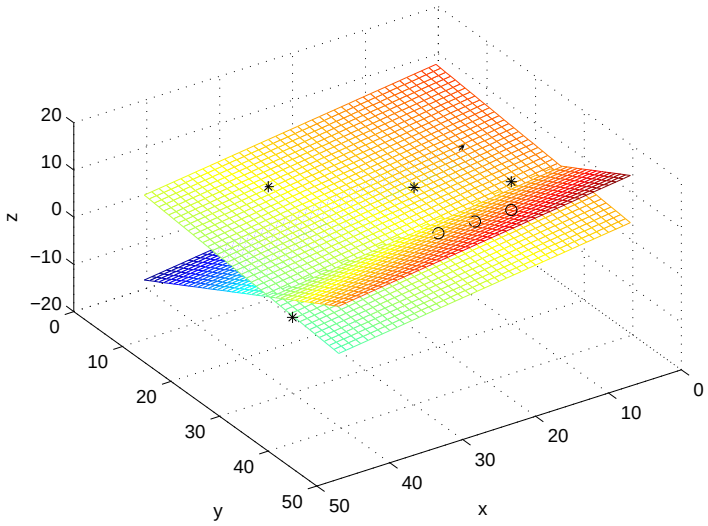


Figure 1.1: Two three-dimensional surfaces corresponding to two different models of behavior. One surface models the the pre-change behavior and the other surface models the post-change behavior. Each generated point falls on one of the two surfaces: points marked with o belong to one surface and points marked with * to the other surface.

of the history or is a low-degree polylog or polynomial. For example, consider the computation of a moving-point average over a window. When the window moves by one value, the computation of the new moving–point average can be done in time independent of the length of the window: merely add the leading-edge value and subtract the trailing-edge value. We present algorithms for adapting to behavioral change that come close to being incremental.

1.4 Related Work

A related problem, though different in spirit, is the change-point detection problem. Here, the goal is to detect changes in the model underlying the data assuming that there exists a time when the model changes. The following is a brief description of the problem. For a more complete description of the problem, the reader is referred to Gustafsson [12] and Lorden [20].

Let f_0 and f_1 be two different probability density functions and $\{X_i\}$ be a sequence of random variables defined as follows:

$$\begin{cases} X_1, X_2, \ldots, X_{\nu-1} & \sim f_0 \\ X_\nu, X_{\mu+1}, \ldots, X_n & \sim f_1, \end{cases} \tag{1.1}$$

where ν is unknown and denotes the time when the change in distribution occurs. The probability densities f_0 and f_1 may be known or unknown. The goal is to detect that a change has taken place at some time N as close as possible to ν subject to a constraint on the rate of false positives. Let

us assume that one observation is received at any discrete time stamp and no more observations are received after a change is detected. Let N be a random variable denoting the number of observations received; equivalently, N denotes the time at which the test decides to stop taking observations, since we observe X one at a time. Since the decision to stop at time N is based only on the first N observations, N is called a *stopping time*. Let us denote by $E_t[N]$ the expected number of observations received assuming that a change takes place at time t. Then the change point detection problem may be formulated as:

$$\begin{aligned} \text{minimize} \quad & E_\nu[N - (\nu - 1) | N \geq \nu] \\ \text{subject to} \quad & E_\infty[N] \geq c \end{aligned}$$

Here, $E_\infty[N]$ represents the expected number of observations until a false positive occurs (since a change never occurs). Therefore, $\frac{1}{E_\infty[N]}$ represents the false positive and the first constraint restricts the false positive rate to be less than $\frac{1}{c}$. The objective function describes the expected delay in signaling a change after its occurrence. The problem above has received much attention in the literature. When both the pre-change distribution f_0 and the post-change distribution f_1 are completely specified, the problem is well understood and has been solved under a variety of criteria. The most popular procedures used are Page's CUSUM test [94] and the procedures of Shiryarev [26] and Roberts [24]. The first asymptotic theory of change point detection was provided by Lorden [20].

In practical problems the assumption that both pre-change and post-change distributions are known is too restrictive. It is likely that both pre-change and post-change distributions involve some unknown parameters for which only partial information can be given. Ideally, we would like to minimize false alarm rates and detection delays for all problems. There is no attractive definition in the literature of optimal procedures for detecting changes for the most general version of the problem. Yajun Mei [22] gives an interesting definition of optimality for the case of unknown pre-change and post-change distribution and proposes procedures that are then proved to be asymptotically optimal, thereby generalizing Lorden's asymptotic theory.

In the context of our work, both the pre-change and post-change model are unknown and the objective is to estimate the post-change model as accurately as possible. The algorithm that we propose is aimed at smoothing occurring changes as quickly as possible. More than one change can potentially occur and, for any change, both the pre-change and the post-change model are assumed to be unknown. Our goal is to use the received stream of data and capture changes in the underlying model as efficiently as possible. The process of using streams of data and adapting to the surrounding environment is called in the literature *adaptive stream processing*. Much attention has recently been given to this topic [4, 5, 6, 18].

Chapter 2

Theory

2.1 Types of Models

A popular way of estimating a model that fits a set of points is regression. One of the variables of the model is identified to be a *dependent* variable and the other variables are *independent* variables. A model predicts the value of the dependent variable given the values of all the independent variables. The differences between the values of the dependent variables in the actual set of data points and the values predicted by the model are the errors of the model. There are many possible ways of defining the best fitting model. A good choice that can often be motivated for statistical reasons and also leads to a simple computational problem is the least square (LS) estimate. Formally, let $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ be a matrix where each row represent a point lying in an p-dimensional space and let $\boldsymbol{b}$ be the vector of dependent variables. Then the ordinary *least square* problem seeks to

$$\begin{array}{ll} \text{minimize}_{\hat{\boldsymbol{b}} \in R^p} & \|\boldsymbol{b} - \hat{\boldsymbol{b}}\|_2 \\ \text{subject to} & \hat{\boldsymbol{b}} \in R(\boldsymbol{X}), \end{array} \tag{2.1}$$

where $R(\boldsymbol{X})$ is the affine subspace generated by the column of the matrix $\boldsymbol{X}$. Once a minimizing $\hat{\boldsymbol{b}}$ is found, then any $\boldsymbol{\beta}$ satisfying the relation $\boldsymbol{X}\boldsymbol{\beta} = \hat{\boldsymbol{b}}$ is called a LS estimate and $\Delta\boldsymbol{b} = \boldsymbol{b} - \hat{\boldsymbol{b}}$ is the corresponding LS correction. The quadratic programming problem in 2.1 is solved when $\hat{\boldsymbol{b}}$ is the orthogonal projection of vector $\boldsymbol{b}$ onto $R(\boldsymbol{X})$. Thus, the LS problem amounts to perturbing the vector of dependent variables $\boldsymbol{b}$ by a minimum amount $\Delta\boldsymbol{b}$ so that $\hat{\boldsymbol{b}} = \boldsymbol{b} - \Delta\boldsymbol{b}$ can be predicted by the columns of $\boldsymbol{X}$.

Weighted Data Points. Our objective is to adapt the estimate to changes occurring in the model, thus a reasonable procedure is to associate each data point with a weight. The larger the weight, the more likely it is that the data point is associated with the last model used. The new problem where each row of the matrix $\boldsymbol{X}$ is multiplied by the correspond weight is called a *weighted*

least square problem and allows the weights to determine the contribution of each observation to the final parameter estimates.

However, for our purposes all variables are equivalent. Furthermore, each variable represents an attribute of behavior and may be affected by sampling or measurement errors. The underlying assumption in weighted least square regression is that errors only occur in the vector $\boldsymbol{b}$ and that the matrix $\boldsymbol{X}$ of predictors is exactly known. Clearly, this is an invalid assumption for our problem.

A more suitable approach for our problem is the *total least square* method [16], sometimes also called *orthogonal* regression. This approach defines the error to be the *minimum* distance of a data point from the surface.

The best-fitting hyperplane with respect to a set of points is obtained by solving the minimization problem

$$\begin{aligned} &\min \textstyle\sum_{i=1}^{n} \alpha_i \cdot (\boldsymbol{\beta}^T \cdot \boldsymbol{x}_i)^2 \\ &\text{subject to } \|\boldsymbol{\beta}\| = 1, \end{aligned} \tag{2.2}$$

where n denotes the number of points, $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$ is the column vector of unknowns defining the normal to the estimated hyperplane, $\alpha_i < 1$ is the weighting factor and $\boldsymbol{x}_i$ is a column vector denoting the point received at time i. We define the matrix $\boldsymbol{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_3, \ldots, \boldsymbol{z}_n\}$ consisting of weighted row vectors $\boldsymbol{z}_i$, where $\boldsymbol{z}_i = \alpha_i \boldsymbol{x}_i^T$ is the weighted observation received at time i.

Using matrix-vector notation, we can write the equation 2.2 as

$$\begin{aligned} &\text{minimize } \|\boldsymbol{Z\beta}\|^2 \\ &\text{subject to } \|\boldsymbol{\beta}\| = 1 \end{aligned} \tag{2.3}$$

Using Lagrange multipliers, the problem can be formulated as finding the vector $\boldsymbol{\beta}$ minimizing the expression

$$E = \|Z\boldsymbol{\beta}\|^2 - \lambda(\|\boldsymbol{\beta}\| - 1), \tag{2.4}$$

which is equivalent to minimizing

$$E = \boldsymbol{\beta^T Z^T Z\beta} - \lambda(\boldsymbol{\beta}^T\boldsymbol{\beta} - 1). \tag{2.5}$$

In order to minimize the error E we take the derivatives in matrix form and set them to zero; we are then left with the following eigenvalue problem:

$$\begin{aligned} &\boldsymbol{Z^T Z\beta} = \lambda\boldsymbol{\beta} \\ &\text{subject to } \|\boldsymbol{\beta}\| = 1. \end{aligned} \tag{2.6}$$

Thus, the solution of our problem $\boldsymbol{\beta}$ will be an eigenvector of the matrix $\boldsymbol{Z^T Z}$. It remains to decide which of these eigenvectors minimizes the expression . Since the goal is to minimize expression

2.4, we want λ to be as small as possible. Thus, the solution will be the eigenvector $\boldsymbol{\beta}$ associated with the minimum eigenvalue λ of the matrix $\boldsymbol{Z^T Z}$.

We conclude the chapter with some considerations about the minimum eigenvalue λ. It is a well-known fact that the matrix $\boldsymbol{Z^T Z}$ is a positive semi-definite matrix, thus all its eigenvalues will be real and non-negative. For convenience of notation, assume $\lambda = \lambda_1 \leq \lambda_2 \leq \ldots \lambda_p$, where each λ_i is an eigenvalue of $\boldsymbol{Z^T Z}$. Let us make the further assumption that the columns of $\boldsymbol{Z}$ have zero mean. If this is not the case, we can always center the matrix by subtracting the mean of each column from its elements, do the analysis, and then add the mean at the end. If we now project the data matrix $\boldsymbol{Z}$ onto the eigenvector $\boldsymbol{\beta}$ and call the resulting projection $\boldsymbol{q} = Z\boldsymbol{\beta}$, we can see that the variance estimate of $\boldsymbol{q}$ is

$$q^T q = \boldsymbol{\beta^T Z^T Z \beta} = \boldsymbol{\beta^T} \lambda_1 \boldsymbol{\beta}. \tag{2.7}$$

Since $\boldsymbol{Z^T Z}$ is symmetric, all its eigenvectors will be orthogonal and therefore the variance estimate of $\boldsymbol{q}$ becomes exactly λ_1. Therefore, geometrically speaking, we are looking for the direction $\boldsymbol{\beta}$ in the dimensional space where the variance is minimized.

2.2 Exponential Smoothing and Sliding Window

A key issue is that of determining the weight to be given to old information in estimating models. Too much weight given to old information results in algorithms that do not update models rapidly, but the more information that is used, the better the estimates in the presence of noise. Popular algorithms for dealing with different emphases on newer and older data are *sliding window* and *exponential smoothing*. A sliding window protocol with window size W estimates a model using only the W most recent data points; it treats all W data points in the window with equal weight, and effectively gives zero weight to points outside the window. An exponential smoothing algorithm parameterized by α, $0 \leq \alpha \leq 1$, gives a weight of α^k to a data point k units in the past; thus, an algorithm using a small value of α "forgets faster". We refer to α as the *forgetting factor*.

Incremental stream-processing algorithms can be obtained for both sliding window and exponential smoothing. Next, we show how this can be done for exponential smoothing under the assumption (made throughout this work) that the dimensionality p of the model is negligible with respect to the number of data points considered. Let $\boldsymbol{P}_i = \boldsymbol{Z}_i^T \boldsymbol{Z}_i$, where $\boldsymbol{Z}_i$ is the matrix of dependent variables at step i. It follows that $\boldsymbol{P}_{i+1} = \alpha \boldsymbol{Z}_i + \boldsymbol{x}_{i+1} \boldsymbol{x}_{i+1}^T$, where $\boldsymbol{x}_{i+1}$ is the column vector representing the point received in iteration $i+1$. Since $\boldsymbol{x}_{i+1}^T \boldsymbol{x}_i$ is an outer product of a p-dimensional vector with itself, the resulting p-dimensional matrix can be computed in $O(p^2)$ operations. The last step is to compute the minimum eigenvalue of the square p- dimensional matrix $\boldsymbol{P}_{i+1}$. Using well-known methods from numerical analysis such as the power method, or Cholesky decomposition, it is possible to compute the eigenvector associated with the minimum eigenvalue in $O(p^3)$. Therefore the worst

case complexity to recover the least square estimate for the weighted total least square problem is $O(p^3)$, which shows that the computation can be done incrementally, *i.e.*, independently on the size of the received data set. A similar analysis shows that incremental algorithms can be obtained for the sliding window protocol. For a detailed description of computational techniques for total least square problems the reader is referred to Van Huffel and Vandewalle [16].

An important issue is that of determining the appropriate α to use at each time T. The value of α can range from 1 (in which case all points from 0 to T are weighted equally) to 0 (in which case only the data point that arrived at T is considered). Small values of α adapt to changes rapidly because they give less weight to old data, whereas large values of α are better at smoothing out noise.

One approach is to change the relative weights given to old and new data when a change is estimated. For instance, suppose the algorithm estimates at time 103 that with high probability a change occurred at time 100; the algorithm then reduces the weight given to signals received before 100 and increases the weight given to signals received after 100. A disadvantage of this approach is that, if the algorithm estimates that a behavioral change has taken place when in reality no change has occurred, it discards valuable old data needlessly. The same approach can be used with sliding windows.

2.3 Experimental Setup

At any point in time, the behavior of a system is captured by a model that is represented by a bounded surface. Our algorithm attempts to estimate the true model given a sequence of noisy signals. We call the model and the surface estimated from signals the estimated values as opposed to the "true" values. The true model is changed at some point in time during the experiment and we evaluate whether the estimated model follows the true model accurately.

At each point in time, a signal is generated as follows. First, a point q is generated on the true bounded surface randomly; next, a random error term e is generated randomly using the given error distribution; and finally, a data point $r = q + e \cdot v$ is generated, where v is the unit normal to the true surface at point q.

2.3.1 Algorithm

Input at time T: A sequence of $T-1$ points that arrived in the time interval $[0, T-1]$ and a new point that arrived at time T.

Output at time T: An estimate of the surface—a hyperplane in a linear model— at time T.

Goal: Minimize the deviation between the estimated and true surfaces.

The true model changes over time, and the manner of change is described separately.

2.3.2 Angle Between Planes

One measure of the fit of the estimated model to the true model is the angle between the surfaces. The inner product of the unit normals to the hyperplanes representing the estimated and true models is the cosine of the angle between the hyperplanes, and we use this as a measure of goodness. The cosine is 1 if the hyperplanes are parallel and 0 if they are orthogonal.

2.3.3 Comparison of Distances of Points from True and Estimated Surfaces

Another measure of the fit is represented by the differences in distances of data points from the true and estimated surfaces. Let $D_{k,t}$ be the minimum distance of the data point that arrived at time k from the true surface at time t. Recall that $d_{k,t}$ is the minimum distance of the data point that arrived at time k from the surface estimated at time t. Let E be defined as follows:

$$E = \sum_{k} (D_{k,k} - d_{k,k})^2. \tag{2.8}$$

Now E is a measure of goodness; the smaller the value of E, the better the resulting estimate. E is usually called the *residual sum of squares* error. Notice that E can be nonzero even if the true and estimated hyperplanes are parallel, because this error term is zero if and only if the two hyperplanes are the same.

Chapter 3

Experiments

We restrict ourselves to linear models; thus, the surface is a hyperplane in a multidimensional space. In each of our experiments we assume that we are given the true model; we generate noisy data from the true model, compute an estimated model from the noisy data, and compare the true and estimated models. At an arbitrary point in time, we change the true model. Noisy data is now generated using the new true model (the distribution of noise terms is assumed to remain unchanged even though the true model changes). Since the estimation algorithm has no specific indication that the true model has changed, the algorithm uses data from before the change as well as data from after the change. Therefore, the estimated model may not be close to the new true model during and immediately after the change. We would like the estimated model to become closer to the true model as the time since the last change increases.

The set of experiments is restricted to 2–dimensional surfaces. Noise is assumed to be Gaussian. This is not a necessary assumption; in fact, the algorithm may be applied with any white noise vector. We consider cases where the noise is low ($\sigma^2 = 1$) or high ($50 \leq \sigma^2 \leq 100$), and study the effect of different values of α on the accuracy of the model. We choose values of α as follows. We pick a positive integer w that we call the *window size* (not to be confused with the window in the sliding window algorithm) and a positive number γ that we call the *threshold*. The value of γ is set to 0.5 in the experiments. Given w and γ, we pick α such that the total weight assigned to all the signals w or more time units into the past is exactly (up to a rounding error) γ. For instance, if $w = 4$ and $\gamma = 0.5$ then we know that the first w signals have a total weight of $\frac{1}{2}$, the next w have a total weight of $\frac{1}{4}$, the next w have a weight of $\frac{1}{8}$, and so forth.

3.1 Experiments With Changing Behaviors

We ran many experiments. In each experiment a change occurs after a certain number of time units. Each change is a translation followed by a rotation of the (true) plane. Fig. 3.1 illustrates the 2D case; each line is associated with a behavior change. Here we report on the following experiments:

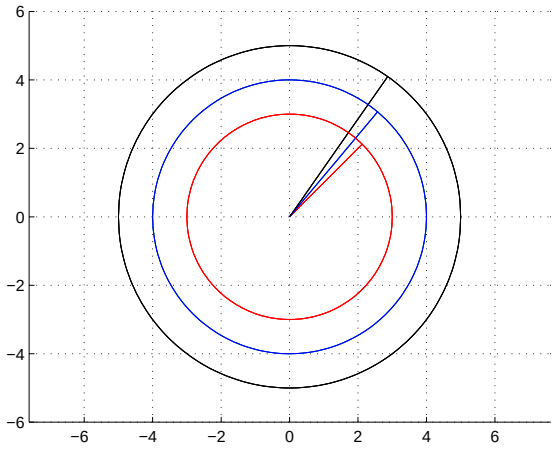


Figure 3.1: The model of behavioral change used in the performed experiments. When the model changes, points are generated using a different line.

- The true model is changed after 500 time units. The translation is 0.75 and the rotation is 10 degrees.
- The true model is changed after 50 time units. The translation is 0.02 and the rotation is 1 degree.
- The true model is changed after 5 time units. The translation is 0.0018 and the rotation is 0.1 degrees.

Each experiment was run for several thousand time units and thus covered many changes of the true model. For ease of visualization, we only show 1500 points in the figures; however, the same pattern occurs for larger numbers of points.

Fig. 3.2 shows the cosine and the angle between the true and estimated hyperplanes at different points in time for the low variance case. Fig. 3.3 shows the relative distance error as a function of time for the low variance case. Figures 3.4 and 3.5 show results for the high variance case. The angle between the true and estimated planes increases sharply at the point of the change and then decreases. The angle at the instant of change is larger for higher values of α; this is not surprising, because higher values of α give greater weight to pre-change data. Also, algorithms with higher values of α take longer after a change to reduce the error.

Higher values of α are less susceptible to noise. This is not apparent from the figures in the low-variance case, but is readily apparent in the high-variance case. Indeed, the algorithm with low α cannot distinguish between a change to the true model and noise in the case of high noise variance. This suggests, as expected, that only high values of α should be used in the case of high noise whether the true model is stationary or not. As discussed earlier, one approach is to adapt the

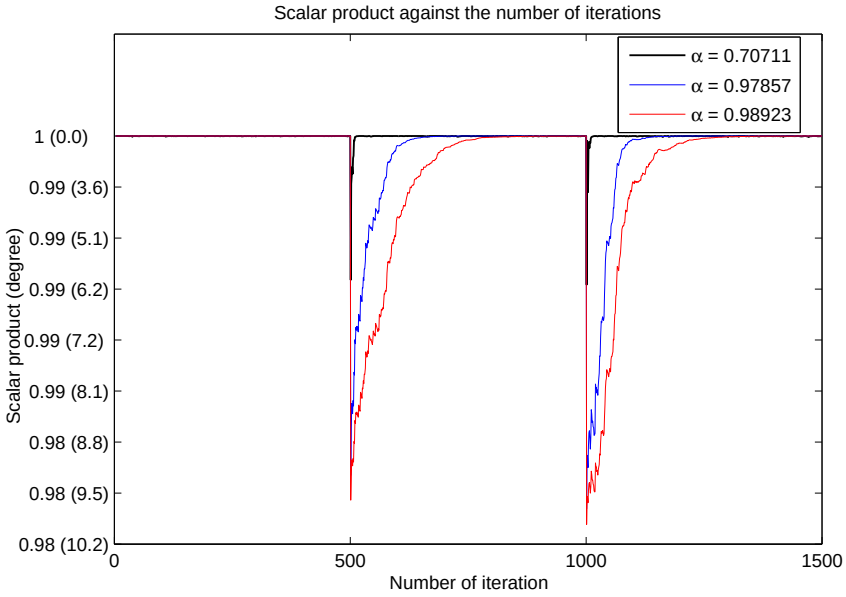


Figure 3.2: The scalar product between the vector of estimated and actual coefficients with threshold 0.5, noise variance 1 and a change occurring after 500 iterations.

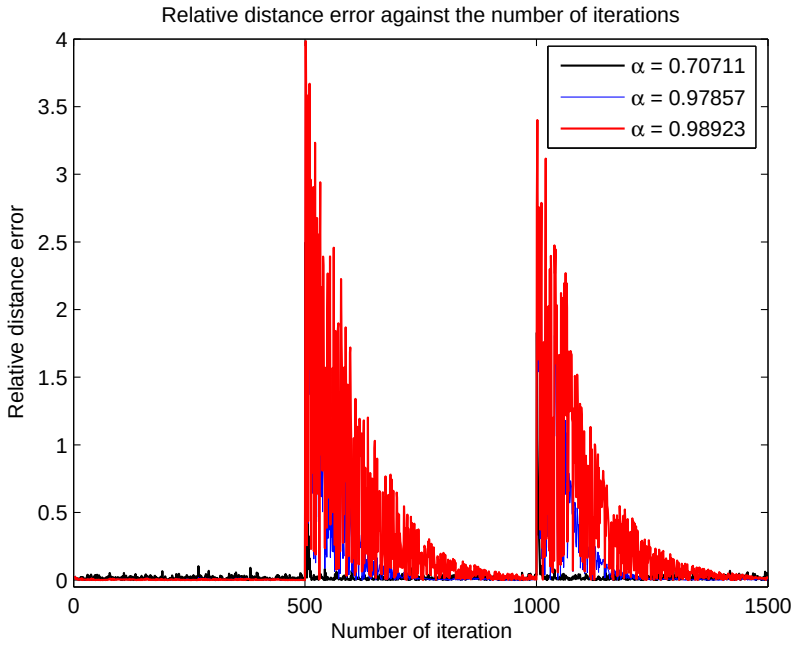


Figure 3.3: The relative distance error between the estimated and actual plane with threshold 0.5, noise variance 1 and change occurring after 500 iterations.

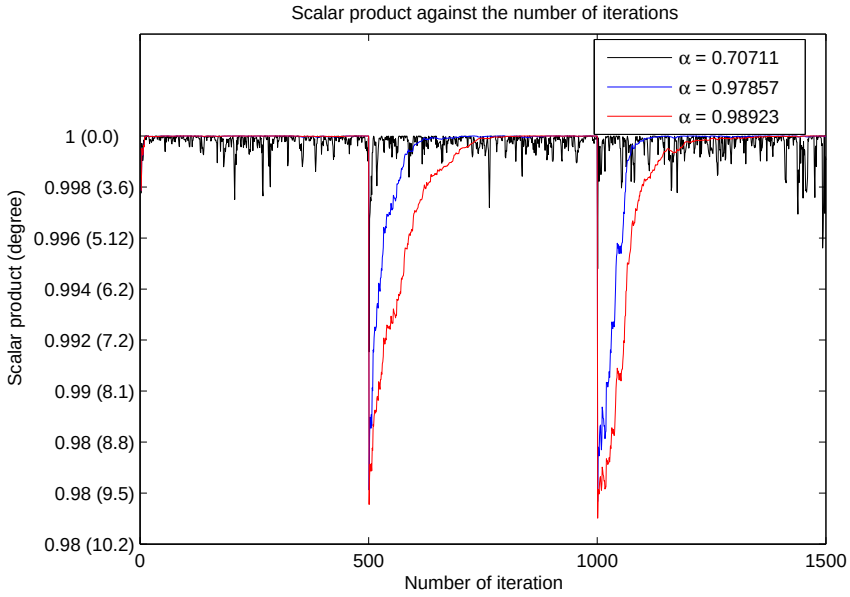


Figure 3.4: The scalar product between the vector of estimated and actual coefficients with threshold 0.5, noise variance 100 and change occurring after 500 iterations.

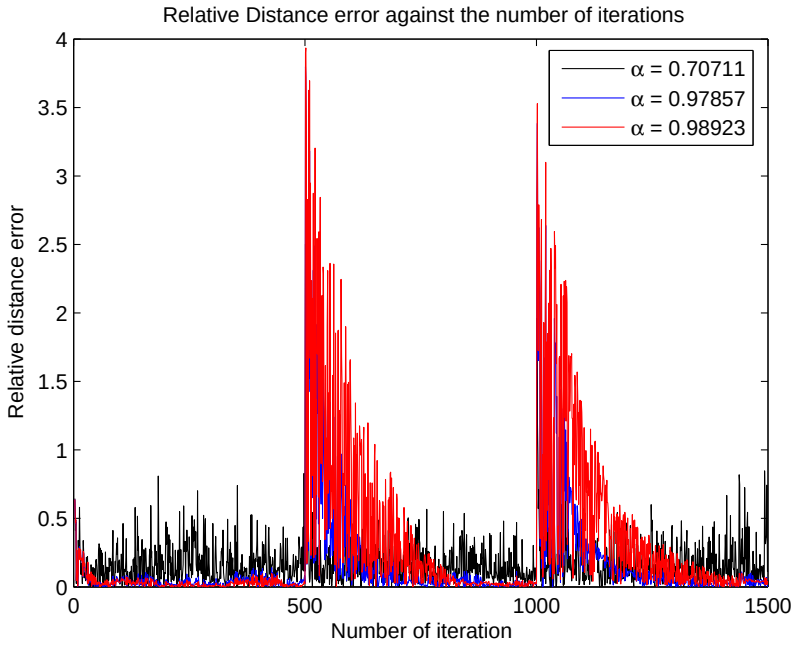


Figure 3.5: The relative distance error between the estimated and actual plane with threshold 0.5, noise variance 100 and change occurring after 500 iterations.

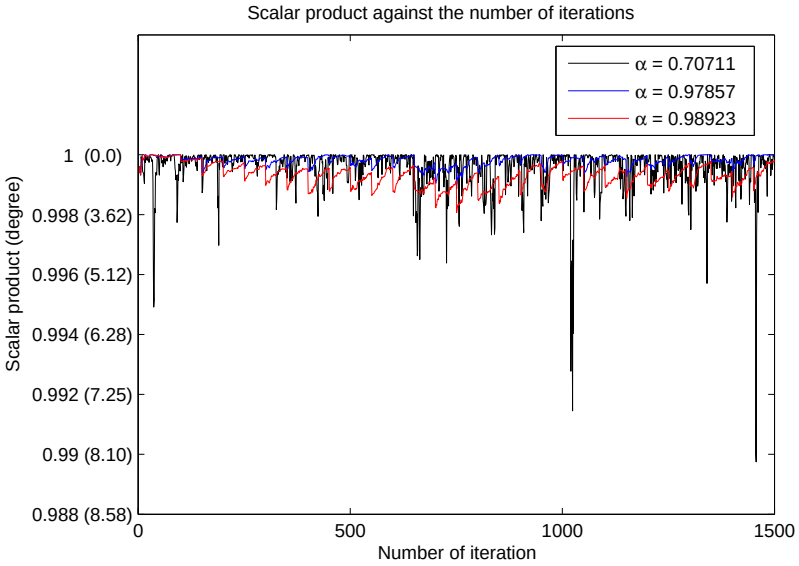


Figure 3.6: The scalar product between the vector of estimated and actual coefficients with threshold 0.5, noise variance 100 and change occurring after 50 iterations.

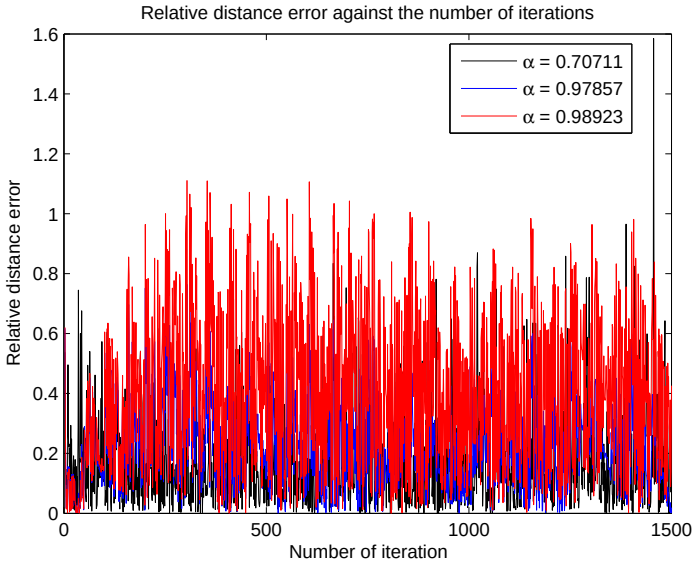


Figure 3.7: The relative distance error between the estimated and actual plane with threshold 0.5, noise variance 100 and a change occurring every 50 iterations.

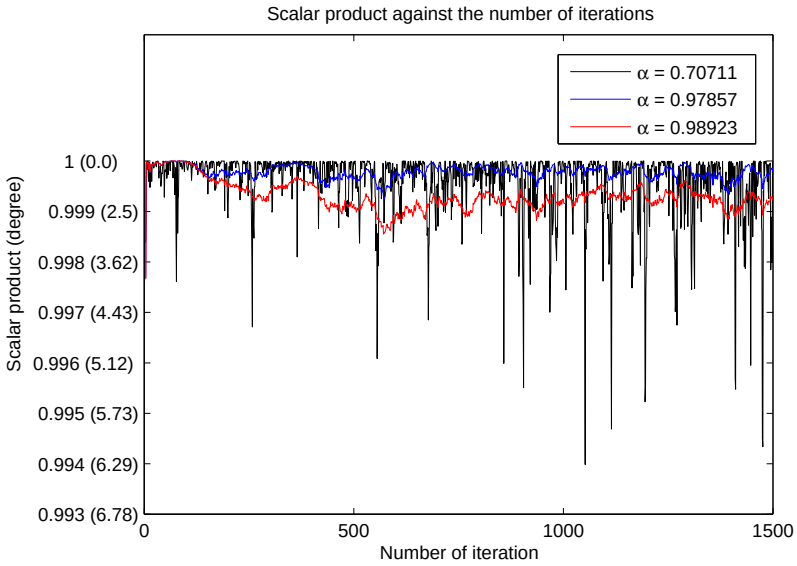


Figure 3.8: The scalar product between the vector of estimated and actual coefficients with threshold 0.5, noise variance 100 and a change occurring every 5 iterations.

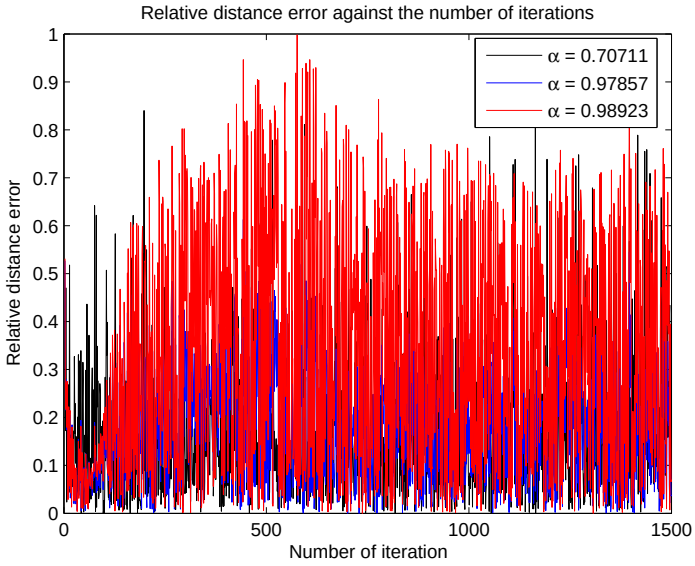


Figure 3.9: The relative distance error between the estimated and actual plane with threshold 0.5, noise variance 100 and a change occurring every 5 iterations.

relative weights given to old and new data when the algorithm estimates that a change has occurred.

When changes occur frequently as in Figures 3.6–3.9, large values of α are not appropriate since they give large weight to data generated according to different past models. Large α values give high accuracy, but only when the amount of data generated according to the same model is high; when this is not the case, smaller values of α give better performance.

The experimental results are explained quite simply by considering the function $\sum_{i=1}^{c} \alpha^{k-i} - \sum_{i=c+1}^{k} \alpha^{k-i}$, where c denotes the time at which the true model changes. The first term is the total weight assigned to pre-change signals and the second to post-change signals. Immediately after the change, the pre-change weights are larger because there are fewer post-change signals. Likewise, the higher the value of α the greater the weight given to pre-change signals.

3.2 Adaptive Algorithms

Adaptive algorithms change the relative weights assigned to older and newer data when a change is detected. The figures show that when a change is abrupt, the change can be detected readily and adaptive algorithms work well. If each change is small but changes occur frequently, the algorithm may come close to accurate predictions for large α values. This can be seen clearly in Figures 3.6–3.9.

An alternative strategy is to compare the model at time T with the models at previous times t, $T - M \leq t \leq T$, where M is a constant window size. So far we have only discussed the case where $M = 1$, which is sufficient for substantial changes. If the algorithm detects a change between any model at a previous time t and the current model, then the algorithm adapts the weights, giving greater weight to signals after time t and less to signals before time t. These experiments are left for future investigation.

Chapter 4

Conclusions

In this part we have described an algorithm for estimating the parameters of a time-varying linear model. The algorithm combines the method of total least squares with exponential forgetting to compute the best estimate of the current model. We have shown that all the computation can be done incrementally using a very small amount of memory. For the tested models we have discussed the recovery of parameters as functions of the frequency of behavioral change of the model.

Part II

Predicate Signaling

Chapter 5

Introduction

In this part, we study mechanisms for reducing communication in distributed sense and respond systems that obtain information from sensors and other sources of data and use it to detect specific conditions. Sensors can generate messages periodically, when anomalies are detected, or when queried by other nodes. We propose a strategy called *predicate signaling* that generalizes these schemes by generating messages when specified predicates, which can deal with both time and anomalies, hold. We show how power consumption, message generation rates and estimation errors can be controlled by choosing predicates appropriately. We compare predicate signaling with other schemes. We derive formulas based on stochastic differential equations to estimate performance measures in predicate signaling. We analyze measurement data, and compare simulations based on measured data with results predicted by our theory. Predicate signaling is a general framework that can be particulary suitable to applications requiring low communication rates for reasons such as energy constraints, secrecy, reduced electromagnetic interference and limited bandwidth.

5.1 Problem Overview

Distributed systems that respond to conditions in the environment can be specified by a set of rules of the form "*when* a predicate P begins to hold *then* execute action e", where P is a predicate on the history of global states of the system [3]. The system initiates action e when the value of predicate P changes from false to true (the system may have another action e' that is executed when $\neg P$ changes value from false to true). The specification also includes quality of service and accuracy requirements; these aspects are discussed later. Examples of systems that respond to conditions include those that intercept intruders, warn when a tsunami is likely to hit, or respond to situations that require control-law changes in multi-agent control systems.

Errors: Each node has access only to its own local state. A node *estimates* the global state by fusing local state information sent to it by other nodes. Nodes can exactly compute only certain types of predicates on global states [7]; estimates of other predicates may be incorrect. Consider a

node w responsible for initiating action e when the value of predicate P changes from false to true. A *false positive* occurs if node w's estimator for P has value true while P has value false. Let d be the delay from the point in time at which P begins to hold to the point in time at which w initiates an action. During the delay interval, the estimated value for P is false while P is true; thus, a *false negative* occurs for the interval. Design specifications (discussed later) include maximum acceptable rates of error.

Next, we discuss different schemes by which nodes communicate with each other and then propose a common framework —predicate signaling— that unifies these schemes and allows for systematic analysis of tradeoffs among them.

5.2 Common Signaling Schemes

We first describe the three most popular signaling schemes used in the literature. These are in order, *time-driven, anomaly-based* and *query-based* signaling. Then we propose a unifying scheme called *predicate signaling* and compare it with the other three schemes.

5.2.1 Time-Driven Signaling

Consider a node w of a distributed system that executes an action e when its estimate of a predicate P becomes true. Typically P is a function of the local states of other nodes. In time-driven signaling, nodes periodically send local state information that node w needs to estimate P. The optimal period is determined by trading off the cost of erroneous estimates against the costs of more frequent messages and computation. While time-driven signaling is appropriate for applications such as data gathering, it is inappropriate if systems are required to respond only to rare events, such as tsunamis, in which case P rarely becomes true. For example, if a system turns on backup power generators when a brownout is imminent, and if this situation arises only when temperatures exceed 100 degrees, there is little value in sending temperature measurements every minute while the temperature is below 95 degrees.

5.2.2 Anomaly-Based Signaling

A goal for a communication strategy is: *nodes should communicate only when they have to.* Anomaly-based signaling helps to achieve this goal by adopting the following scheme. Predictive models that track measurements exist for many applications including weather, power grids, stock markets, intruder behavior, and virus propagation. The model that predicts a parameter such as amount of rainfall most accurately may be different under different global conditions, such as when a hurricane is approaching or when there is high pressure over nearby deserts. Accurate predictive models can be employed to reduce the volume of communication between nodes.

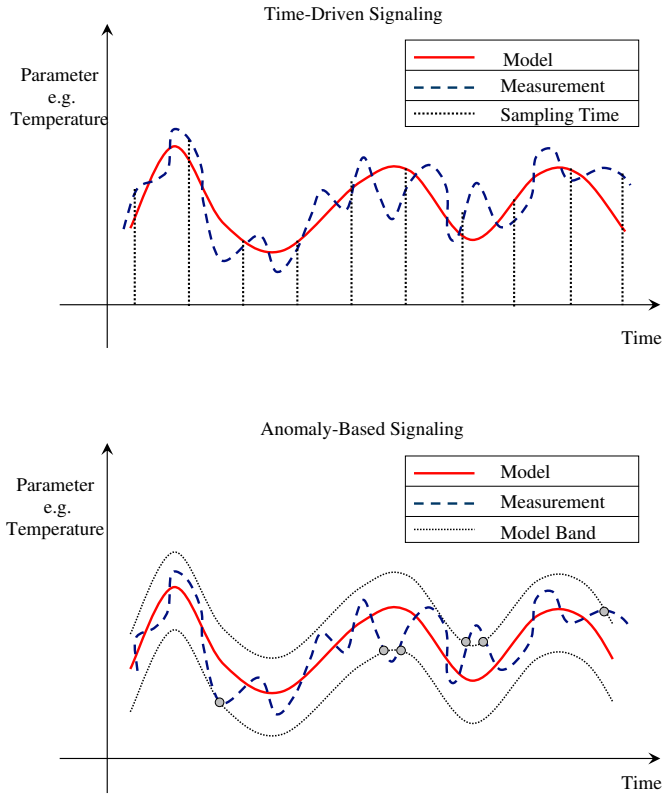


Figure 5.1: Time-Driven and Anomaly-Based signaling schemes

Consider communication from a node v to a node w. Some system parameters are observable by both nodes; for example, both nodes may have local clocks that give the same time (to a sufficient degree of accuracy) and thus time is observable by both nodes. Parameters that are at least partially observable by multiple nodes are not limited to time; for instance multiple nodes may be able to observe signals sent by a satellite. Some parameters that are observable by node v may not be observable by w; for instance, if nodes v and w are far apart, w may not be able to observe the temperature at v. For brevity v's *local parameters* with respect to w are those parameters that v can observe but w cannot. The communication strategy uses models that predict v's local parameters given the parameters observable by both v and w. Associated with communication from node v to node w is a set of such predictive models.

At any given time, node v uses one of the models, say model M, in its set. Node v sends messages to w only when: (1) the values that v measures of its local parameters deviate by more than a specified threshold from the values predicted by the model M that v is currently using; or (2) v changes the model M that it is using. These messages, which we call signals, include the current measured values of v's local parameters and the model that v uses from that point onwards until the next message indicating a model change. Node w estimates v's local parameters from (1) the model M that v is currently employing, (2) the parameters that both v and w can observe, and (3) the messages that w has received.

The rate at which messages are sent decreases with the predictive accuracy of the model. An advantage of anomaly-based signaling is that *the absence of signals conveys information*; namely, the information that measurements match the current model. By contrast, the absence of signals never conveys information in (traditional) models of distributed systems—see theories of process knowledge and learning in traditional distributed systems [15, 8]—because these models do not deal with real time. However, the problem of estimation becomes difficult when measurements are noisy and this issue is discussed later.

An illustration of the two signaling techniques is shown in Figure 5.1. The top plot shows time-driven signaling where values are measured and signals are sent periodically. The bottom plot shows a model that predicts a local parameter, say temperature, of a node v that is observable by another node w, as a function of time. Associated with this model is a tolerance band: a message is sent by v to w only when v's measurement of its temperature crosses this tolerance band. This signal includes the value of v's measurement and the model that v uses from that point onwards. Node w estimates v's temperature given v's current model, the current time (observable by both nodes), and messages received by w from v.

In some cases, an anomalous situation may be represented as a composition of simpler anomalous situations. For example, a power brownout is likely when temperatures are very high **and** when power lines are saturated. In such cases, a predicate P is expressed as a composition, such as

$P = Q \wedge R$, of predicates Q and R. Different nodes estimate the different predicates P, Q and R, and send signals when the estimates of their predicates change; for instance, the node that estimates P does so using the the most recent estimates of Q and R that it receives.

A potential criticism is that accurate models may require a lot of computation, and so tradeoffs between communication and computation should be achieved. However, in many applications such as weather, the model maps time (which is assumed to be the only observable parameter) to the unobservables. Such models can be registered on storage systems that are either centralized or distributed throughout the network. A large spectrum of approaches for constructing sensor storage systems has been developed in recent years, some of which are briefly discussed next. In the simplest approach each sensor can store the data locally (*e.g.*, in flash memory), so that all writes are local and incur no communication overhead. If the size of the model exceeds the sensor storage capability, the remaining part of the model is stored in a distributed fashion on other sensors in the network. The disadvantage is that if the sensor needs a portion of the model stored on another sensor, it needs to send a message to recover the desired information. Such read requests are usually handled by flooding. Research efforts such as Directed Diffusion [17] have attempted to reduce these read costs, however, by using intelligent message routing.

An alternate scheme that is not discussed here is to use another simple model: assume that the last value of the signal received is the best predictor of the future. Initial experiments with weather data suggest that this works well. An advantage of such a model is that no storage or computation is required; a sensor generates a message only when the measured value deviates from the value of the last signal generated.

5.2.3 Query-Based Signaling

Declarative queries have recently assumed the role of a key programming paradigm for networks consisting of a large set of nodes. TinyDB [21] and Cougar [28] are two example query processing systems for multi-hop sensor networks. These systems emphasize in-network processing of declarative queries to reduce data communications and battery usage. TinyDB especially focuses on acquisitional aspects of query processing such as where, when and how often data should be collected from the sensors [25]. Cougar uses sensor update and query occurrence probabilities for view selection and location on top of a carefully constructed aggregation tree. Declarative queries are especially useful because they allow programmers to task an entire network of nodes rather than requiring them to program the individual nodes of the network. However, a major problem arises when querying a sensor network database. In standard databases, which are typically considered to be complete sources of information, the job of the engine is to answer a query correctly based upon available data. In sensor network databases it is often impossible to gather all relevant data needed for a query; this is due to the fact that the physically observable world consists of a set of

continuous phenomena in both time and space. In order to compensate for this a number of data reduction techniques, where the general goal is to trade accuracy for performance in massive data sets, have been proposed. Some of those techniques consist of synopsis data structures [11] that provide a summary of the data set within acceptable levels of accuracy while being much smaller in size. Other techniques involve sampling at discrete points in time. Most recently, model-driven techniques have been proposed for querying sensor networks [9]. Such models provide a framework for optimizing the acquisition of sensor readings; sensors are used to acquire data only when the model is not rich enough to answer the query with acceptable confidence. Performance metrics are resource consumption, delay when answering queries [29] and communication costs. If a sensor is far from the query source, then the query source cannot acquire the reading from it without forwarding the request to other nodes.

In the context of sense and respond systems, a controller generates queries for the sensor network when a predicate P, to which the controller is required to respond, is likely to change value from false to true. Query-based signaling can be integrated into the predicate signaling framework by making the response e (to a predicate becoming true) the generation of a query.

5.2.4 Predicate Signaling

Predicate signaling is a generalization of the signaling schemes, discussed above, that allows designers to make design choices that trade off relative advantages of different signaling schemes within a common framework. Moreover, predicate signaling is a natural framework for sense and respond systems specified by *when-then* rules; *when* a predicate P becomes true *then* execute an action e, where the action for a sensor or fusion node is the generation of a signal. Next, we show how predicate signaling generalizes other schemes.

Periodic signaling with a period D is a special case of predicate signaling in which the predicate is to send messages every D time units:

$$\exists \text{ integer } k :: \ t = k \times D.$$

Anomaly-based signaling is a special case in which the predicate is to send a message when an anomaly occurs. Query-based signaling is incorporated into predicate signaling in the following way: the signal sent when a predicate becomes true is a query to the sensor network to send additional information. The optimal policy for a node v that is responsible for initiating a response when a predicate P becomes true may be to request additional information if v has received no messages for such a long time that its estimate of P is likely to be inaccurate.

Predicate signaling is neither better nor worse than other schemes; it is a unified framework for systematic analysis of tradeoffs.

Problem Setting for Predicate Signaling The *history* of a system at a point T in time specifies the (global) system state at each time t from the instant of system initiation ($t = 0$) to the time $t = T$. The history is a function from time to system states.

We are given the following:

1. A set of pairs (P, e), where P is a predicate on the history of global states of a system and e is an action.

2. For each pair (P, e), the net benefit of initiating action e with delay d after P becomes true.

3. For each pair (P, e), the cost of executing action e while P is false.

4. Constraints on energy, message bandwidth and computing capacity.

The problem is to maximize the net benefit per unit time subject to the given constraints. A definition of net benefit is the difference between the benefits of executing appropriate actions and the costs of executing inappropriate actions. Here, we start investigating only a very small part of the overall problem; we give analytical and simulation results for the dependencies of the estimation error and the message rate on the adopted signaling scheme.

Chapter 6

Design Issues

Next we list some of the relative advantages of different signaling schemes and then discuss how predicate signaling allows designers to make tradeoffs among them within a unified framework.

1. **Reduction in average load**: If predictive models are accurate then communication rates are lower in anomaly-based signaling than in time-driven signaling for the same degree of error. Also, the rate of generation of queries in query-based signaling can be reduced by querying for additional measurements only when local measurements deviate from the current model.

2. **Reduction in energy**: At first, it might be said that anomaly-based signaling conserves energy for the times when it is needed: times when reality doesn't fit the current model used by the estimator. However, it has been observed that in sensor networks, energy is consumed both during active communication and *idle listening*. The sensors in a predicate signaling based architecture spend most of the time in idle listening and only very little time in active communication. The exact cost of idle listening depends on radio hardware and mode of operation. For long-distance radios (0.5km or more), transmission power dominates receiving and listening costs. By contrast, several generations of short-range radios show listening costs of the same order of magnitude as receiving and transmission costs. Therefore, predicate signaling based architectures offer significant advantages in situation where sensors detecting changes in predicates need to send this information over long distances.

3. **Increase in load variance**: The loads on communication and computational resources are more stable in time-driven signaling systems than in anomaly-based signaling systems, which may have long periods of inactivity punctuated by bursts of frenetic activity.

4. **Greater consequence of lost messages**: A single message loss can be catastrophic in anomaly-based signaling. Consider the consequence of the loss of a message from a node v to a node w where the message contains (1) the information that v's estimate of a predicate P has become true and (2) the new model that v starts to use from that point onwards. Node w

will both estimate the value of P incorrectly and use the wrong (old) model in its estimation. There will be no subsequent message from v while v's measurements match v's current model, and node w will continue to use the wrong model until it receives a subsequent message from v. This incorrect use of the wrong model can persist for an arbitrarily long time. By contrast, in periodic signaling, if a single message is lost, the error can be repaired by the message sent in the next period. Since the cost of lost messages is high in anomaly-based signaling, more computationally expensive communication protocols are used than for time-driven signaling; for instance, senders may resend messages repeatedly until an acknowledgment is received.

5. **Computational impact**: In many cases anomaly-based signaling requires less computation, averaged over time, than time-driven signaling.

By suitably combining predicates from time-driven and anomaly-based signaling, we can choose design points in between time-driven and anomaly-based signaling and thus make tradeoffs between them. For instance we can reduce communication and computational requirements while limiting the consequences of lost messages and load variance. Also, by generating queries for additional measurement data only when certain predicates hold, and specifying these predicates appropriately, designers can control message generation rates while limiting errors.

Chapter 7

Filter for Predicate Signaling

Predicates are estimated from measurements, and the measurements may have error; this error is usually called *measurement error*. The dynamic component of the measurement error is called *measurement noise*. Predicate estimation is a problem in filtering: estimating true values from noisy measurements. A huge body of literature exists on filtering with time-driven signaling, beginning with the Kalman filter [27]. There are substantial differences in the algorithms for filtering when signals are generated on a time-driven basis and when signals are generated only when a predicate begins to hold. To illustrate the differences between the algorithms we review, in a few lines, the essential ideas of the Kalman filter and then discuss the new challenges introduced by predicate signaling. In the simple Kalman filter, it is assumed that the *a priori* probability density of the parameter of interest and the noise of the measuring device are normally distributed. Recursively applying Bayes' Law, the normal *a posteriori* probability density can be efficiently derived, and the mean and variance of the *a posteriori* density depends on the means and variances of the *a priori* density and the measurement noise. The derivations of the Kalman filter may be found in Sayed et al. [27]; here we only give the final equations under the assumption of a one-dimensional state space model,

$$\begin{aligned} dX_t^{(1)} &= F(t)X_t^{(1)} + G(t)w_t \\ X_t^{(2)} &= H(t)X_t^{(1)} + v_t, \end{aligned} \tag{7.1}$$

where $X_t^{(1)}$ is the state and $X_t^{(2)}$ is the observed output. The terms w_t and v_t are respectively the process noise and the measurement noise.

Let us assume here that $x_0, w_0, w_1, \ldots$ and $v_0, v_1, \ldots$ are jointly Gaussian and independent. Furthermore, assume that the expectation of w_t is $E[w_t] = 0$ and the expectation of v_t is $E[v_t] = 0$. In addition, assume that $E[w_t w_s] = q(t)\delta(t-s)$, $E[v_t v_s] = r(t)\delta(t-s)$ and $E[w_t v_s] = 0$, where $\delta(t_2 - t_1)$ is the Dirac delta function and $q(t)$, $r(t)$ are positive functions. Thus both process noise and measurement noise are uncorrelated in time, and uncorrelated with each other. The goal is to find the estimate of the state vector $X_T^{(1)}$, here denoted by $\hat{X}_T^{(1)}$, which is a linear function of the

measurements $X_t^{(2)}$, $0 \leq t \leq T$ received up to time T and minimizes the mean squared estimation error $E[(X_T^{(1)} - \hat{X}_T^{(1)})^2]$. The initial estimate and covariance matrix are denoted respectively by $\hat{X}_0^{(1)}$ and P_0. The details of the derivation of the Kalman estimator can be found in many books. Here, we only report the differential equations,

$$\dot{P}_t = 2F(t)P(t) + G^2(t)Q(t) - \hat{K}_t^2 R_t$$
$$\dot{\hat{X}}_t^{(1)} = F(t)\hat{X}_t^{(1)} + \hat{K}_t[X_t^{(2)} - H(t)\hat{X}_t^{(1)}],$$

where $\hat{K}_t = P(t)H(t)/R(t)$ is the Kalman gain. The new problem introduced by predicate signaling is *conditioning*. The algorithm estimates the distribution of the true value given that measured values satisfied a specified model ($\neg P$ holds) between signals. Even in the case when $X^{(1)}$ and $X^{(2)}$ are both Gaussian processes, the Kalman Bucy filter cannot be applied if a predicate signaling approach different from time-driven is used for providing the measurements of $X^{(2)}$. This is because the information upon which we are conditioning consists of a set of pairs $\{(t_i, e_i)\}$, where t_i is the time when the predicate changed its truth value for the i^{th} time and e_i is the description of the event that caused the predicate to change at time t_i. Obviously, the process generating this information is no longer Gaussian even if both $X^{(1)}$ and $X^{(2)}$ are continuous Gaussian processes.

This conditioning leads us to a different technique for estimation, and different algorithms.

7.1 Representation of Noise as Stochastic Processes

We begin by studying a model of noise suitable for a filter that can be used with anomaly-based signaling.

Model Noise: We define the noise of a model M, at time t, to be $f(t) - g(t)$: the difference between the true value $f(t)$ of a parameter at time t and the value $g(t)$ predicted for that parameter by model M at t. We also use the term *process noise* for model noise.

Measurement Noise: We define measurement noise at time t to be $m(t) - f(t)$: the difference between the measured value $m(t)$ of a parameter and its true value $f(t)$ at time t.

A signal is generated when the measurement deviates from the current model by more than a given threshold, *i.e.*, when $m(t) - g(t) > \sigma$, where σ is the threshold. Thus a signal is generated when the measured value (model noise plus measurement noise) exceeds the threshold.

Our problem is to estimate the true value $f(t)$ of a parameter at time t given the signals received up to that point in time. This problem is equivalent to estimating the model noise $f(t) - g(t)$ at time t, since the true value $f(t)$ can be obtained by summing the model noise and the model prediction $g(t)$.

Since we assume that measurements are being taken continuously, a reasonable mathematical abstraction for measurement noise and model noise is that they are continuous stochastic processes.

A simple choice for this abstraction is the Ornstein-Uhlenbeck (OU) process [10]. An OU process is the solution of the following stochastic differential equation,

$$dX_t = -\alpha X_t dt + \beta dW_t, \tag{7.2}$$

where W_t is an Wiener process, and α and β are positive constants. An OU process is a Gaussian process that experiences a drift towards the initial value of magnitude proportional to its displacement. Let us denote by X_0 the initial value at time t_0. Then, at any time t, the random variable X_t has the following mean and variance:

$$\begin{array}{ll} X_0 e^{-(t-t_0)\alpha} & \text{(mean)} \\ \frac{\beta}{2\alpha}(1 - e^{-2(t-t_0)\alpha}) & \text{(variance).} \end{array} \tag{7.3}$$

A Wiener process W_t is instead a zero mean Gaussian process with variance t, with the additional property that increments are independent, *i.e.*, for $s < t$, $W_t - W_s$ is independent of W_s. Notice that while the variance of an OU process converges to $\frac{\beta}{2\alpha}$ as $t \to \infty$, the variance of a Wiener process grows linearly with time. We represent model noise by an OU process because we expect to use accurate models, so the variance of process noise does not increase without bound over time. We could represent measurement noise either as an OU process or as a Wiener process; for this analysis we use the simpler Wiener process. This assumption is valid provided predicates are used to ensure that the inter-arrival time between signals is not too large; in fact this can guarantee that the variance does not grow much.

Let $X_t^{(1)}$ be the measurement noise, and let $X_t^{(2)}$ be the model noise. The measurement and model noise processes are described by the following pair of stochastic differential equations,

$$\left\{ \begin{array}{lcll} dX_t^{(1)} & = & a dW_t^{(1)} & \text{(measurement noise)} \\ dX_t^{(2)} & = & -\alpha X_t^{(2)} dt + \beta dW_t^{(2)} & \text{(model noise),} \end{array} \right. \tag{7.4}$$

where $W_t^{(1)}$ and $W_t^{(2)}$ are independent Wiener processes, and a, α and β are parameters. Under the mapping of time $a^2 t \to t$, we can rewrite the system of equation 7.4 as

$$\left\{ \begin{array}{lcll} dX_t^{(1)} & = & dW_t^{(1)} & \text{(measurement noise)} \\ dX_t^{(2)} & = & -\kappa X_t^{(2)} dt + \epsilon dW_t^{(2)} & \text{(model noise),} \end{array} \right. \tag{7.5}$$

where $\kappa = \frac{\alpha}{a^2}$ and $\epsilon = \frac{\beta}{a}$.

The pair of equations 7.5 defines a two-dimensional Ito diffusion process. An Ito diffusion process has the general form

$$d\boldsymbol{X}_t = \boldsymbol{b}(\boldsymbol{X}_t)dt + \hat{\sigma}(\boldsymbol{X}_t)d\boldsymbol{W}_t, \tag{7.6}$$

where $\boldsymbol{X} \in \mathbb{R}^d$, $\boldsymbol{b} : \mathbb{R}^d \to \mathbb{R}^d$, $\sigma : \mathbb{R}^d \to \mathbb{R}^d \times \mathbb{R}^d$, and $\boldsymbol{W}_t$ is an $\mathbb{R}^d$-valued Wiener process.

The generator of an Ito diffusion process [10] is a linear operator of the form

$$\widehat{\mathcal{L}} = \sum_{i=1}^{d} b_i \frac{\partial}{\partial x_i} + \frac{1}{2} \sum_{i=1}^{d} \hat{\sigma}_{ik} \hat{\sigma}_{jk} \frac{\partial^2}{\partial x_i \partial x_j}. \tag{7.7}$$

For our Ito diffusion process defined in equation 7.5 we have

$$d = 2, \quad \boldsymbol{b}(\boldsymbol{X}_t) = \begin{pmatrix} 0 \\ -\kappa X_t^{(2)} \end{pmatrix} \quad \text{and} \quad \hat{\sigma}(\boldsymbol{X}_t) = \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix}.$$

Therefore, applying the formula given by equation 7.7 we obtain that the generator of our Ito diffusion process on $\mathbb{R}^2$ is

$$\widehat{\mathcal{L}} := -\kappa x_2 \frac{\partial}{\partial x_2} + \frac{1}{2} \left(\frac{\partial^2}{\partial x_1 \partial x_1} + \frac{\partial^2}{\partial x_2 \partial x_2} \right). \tag{7.8}$$

7.2 Conditional Probabilities Given Predicates

Initially restrict attention to a simple predicate: the difference between measured and modeled values exceeds a constant. We call this constant σ. Thus, the predicate is

$$|x_t^{(1)} + x_t^{(2)}| = k\sigma \text{ for some } k \in \mathbb{N}.$$

We introduce the following mathematical model. The plane $\mathbb{R}^2$ is split into (overlapping) domains $\Omega_k = \{(x_1, x_2) : |x_1 + x_2 - k\sigma| < \sigma\}$, $k \in \mathbb{Z}$. Observe that the boundary $\partial\Omega_k$ consists of two disconnected components (parallel lines) that belong to Ω_{k-1} and Ω_{k+1}; we denote them by Γ_{k-1} and Γ_{k+1} respectively. An illustrative example including few domains is presented in Figure 7.1. We solve the following problem: given an increasing sequence $0 = t_0 < t_1 < \ldots < t_n$ of hitting times and a sequence n_i ($n_{i+1} = n_i \pm 1$), reconstruct the probability distribution for $\boldsymbol{X}_t$ conditional on the event $\{\tau_i = t_i, k_i = n_i, i = 1 \ldots n, \tau_{n+1} > t\}$. Our approach is to employ the Fokker-Planck equation. Since $\boldsymbol{X}_t \in \Omega_{k_i}$ for $t \in (t_i, t_{i+1})$, the probability density $p_t(x)$ that $\boldsymbol{X}_t$ is found at $\boldsymbol{x}$ satisfies the Fokker-Planck equation

$$\begin{cases} \partial_t p_t(\boldsymbol{x}) &= \widehat{\mathcal{L}}^* p_t(\boldsymbol{x}) \\ p_t(\boldsymbol{x}) &= 0 \qquad\qquad \text{if } \boldsymbol{x} \in \partial\Omega_{K_i} \end{cases} \tag{7.9}$$

Here $\widehat{\mathcal{L}}^* = \frac{1}{2}\left(\frac{\partial^2}{\partial x_1 \partial x_1} + \epsilon^2 \frac{\partial^2}{\partial x_2 \partial x_2} \right) + \kappa \frac{\partial}{\partial x_2} x_2$ is the adjoint of the generator $\widehat{\mathcal{L}}$. This follows immediately from the derivation in Appendix E. The initial condition $p_{t_i}(\boldsymbol{x}) = q_i(\boldsymbol{x}) \delta_{\partial\Omega_{k_{i-1}}}(\boldsymbol{x})$ is the single layer density on $\partial\Omega_{k_{i-1}} \subset \Omega_{k_i}$ corresponding to the distribution of exit locations from the

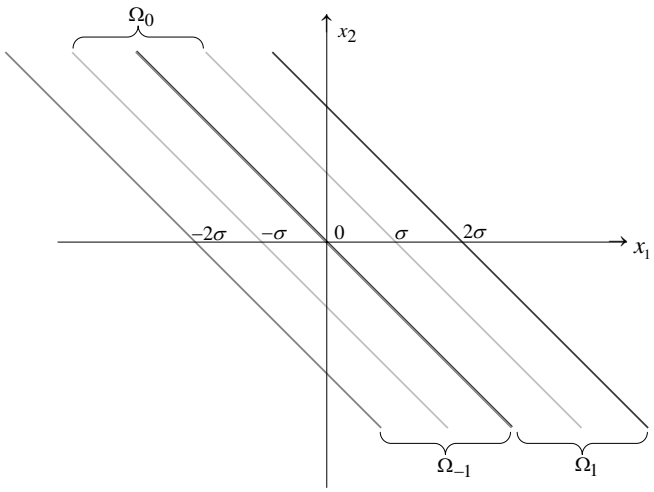


Figure 7.1: Domains Ω_{-1}, Ω_0 and Ω_1

previous domain $\partial\Omega_{k_{i-1}}$. Let λ_n and $\phi_n(x_1, x_2 - k\sigma)$ be the eigenvalues and the eigenfunctions of $\widehat{\mathcal{L}}^*$ in the strip Ω_k with zero boundary conditions (observe that k only appears through translation of x_1). Let us label the eigenvalues as $0 \geq \lambda_0 \geq \lambda_1 \geq ... \geq \lambda_n$. Let φ_n be a set of functions such that, for any pair (i, j), it holds that

$$\int_{\Omega_{k_i}} \phi_i(x_1, x_2 - k\sigma)\varphi_j(x_1, x_2 - k\sigma)d\boldsymbol{x} = \delta_{i,j}.$$

In other words, $(\varphi_n, \phi_n)_{n\in\mathbb{N}}$ form a bi-orthogonal set. Furthermore, assume that $\phi_0(\boldsymbol{x})$ is normalized to have integral one. Using the orthogonality we can write any solution of equation 7.9 as

$$p_t(x_1, x_2) = \sum_{n=0}^{\infty} p_n^{(i)} e^{\lambda_n(t-t_i)}\phi_n(x_1, x_2 - k_i\sigma), \tag{7.10}$$

where the coefficients $p_n^{(i)}$ are related to the initial conditions $q_i(\boldsymbol{x})$ as

$$p_n^{(i)} = \int_{\Omega_{k_i}} p_{t_i}(\boldsymbol{x})\varphi_n(x_1, x_2 - k_i\sigma)d\boldsymbol{x}. \tag{7.11}$$

The equality above follows by expanding p_{t_i} using the definition given in equation 7.10 and then using the fact that $(\varphi_n, \phi_n)_{n\in\mathbb{N}}$ form a bi-orthogonal set. Since $p_{t_i}(\boldsymbol{x}) = q_i(\boldsymbol{x})\delta_{\partial\Omega_{k_{i-1}}}$ we can rewrite the integral as

$$p_n^{(i)} = \int_{\partial\Omega_{k_{i-1}}} q_i(\boldsymbol{x})\varphi_n(x_1, x_2 - k_i\sigma)ds. \tag{7.12}$$

We can finally compute $p_m^{(i+1)}$ using the flux operator $\widehat{\mathcal{F}}$ (see Appendix A) and obtain

$$\begin{cases} p_m^{(i+1)} &= \sum_{n=0}^{\infty} p_n^{(i)} e^{\lambda_n (t_{i+1}-t_i)} T_{m,n}, \\ T_{m,n} &= \int_\Gamma \widehat{\mathcal{F}} \phi_n(x_1, x_2) \boldsymbol{n}(\boldsymbol{x}) \varphi_m(x_1, x_2 \pm \sigma) ds \end{cases} \tag{7.13}$$

where $\Gamma = \{\boldsymbol{x} : x_1 + x_2 = \sigma\}$; the "$\pm$" sign corresponds to $k_{i+1} = k_i \mp 1$. Thus the whole problem is effectively reduced to the computation of the *transfer matrix* $T_{m,n}$ and of the eigenvalues $\lambda_{m,n}$. We next show how to get an analytical expression for the transfer matrix $T_{m,n}$. The flux operator for the generator 7.8 [10] is given by $\hat{\mathcal{F}} = (\partial_{x_1, \epsilon^2 \partial_{x_2}})/2$. Since the normal to any boundary $\partial\Omega_k$ is $n = (1,1)/\sqrt{2}$, we obtain

$$\widehat{\mathcal{F}} u(\boldsymbol{x}) \cdot \boldsymbol{n} = \frac{1}{2\sqrt{2}} [\sigma_{x_1} u(\boldsymbol{x}) + \epsilon^2 \sigma_{x_2} u(\boldsymbol{x})].$$

Expression 7.13 for the transfer matrix may therefore be written as

$$T_{mn} = \frac{1}{2} \int_\Gamma [\partial_{x_1} \phi_n(x_1, x_2) + \epsilon^2 \partial_{x_2} \phi_n(x_1, x_2)] \varphi_m(x_1 \pm \sigma, x_2) d\boldsymbol{x}. \tag{7.14}$$

Chapter 8

Theory and Measurements

In this chapter we evaluate predicate signaling in the setting of habitat monitoring. This problem has received much attention from the sensor network community [1, 2]. We fit our models to measurements of weather data extracted from the historical Integrated Surface Hourly database. The data give parameters such as temperature and pressure at hourly intervals at different locations over 5 years. We use a single *simple predictive model* to illustrate a point: if there are benefits from using even a single predictive model, then there surely are benefits from using a set of more accurate models where the model most appropriate for each point in time is employed. The predicted parameter is the average over the 5 years. For instance, the predicted pressure at 9 AM on December 25 is the average of the measured pressure at 9 AM on December 25 over the 5 years.

Next we show that the Ornstein-Uhlenbeck and Wiener processes are satisfactory models for the data, and then present experimental results concerning communication requirements and estimation accuracy.

8.1 Fitting of Data

We estimate the parameters of the pair of stochastic differential equations 7.4 that fits data at a given location and for a given parameter (such as temperature or pressure) as follows. For the purposes of fitting the model to data we assume that there is no measurement noise. We first estimate the parameters κ and ϵ of equation 7.5 that specify the stochastic process for model noise. We do so by using the formula for the variance of the distribution of the difference $h(\tau)$ between model noise values at two times separated by a duration τ:

$$h(\tau) = X^{(2)}_{t+\tau} - X^{(2)}_{t}.$$

This formula was given in equation 7.3.

We then compute the sample correlation function $c(\tau)$ for the parameter from the measured data.

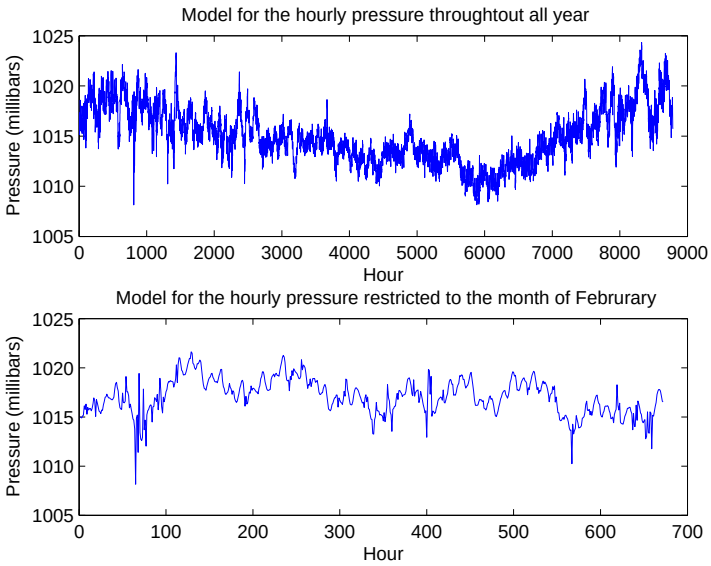


Figure 8.1: Model of the hourly pressure for the location of San Diego North Island

Then we compute the values of κ and ϵ so that the variance of $h(\tau)$ best fits $c(\tau)$.

Due to space limitations we only report results obtained using pressure data collected at a single location (San Diego North Island). All other experiments in this work refer to this data source. For the other locations that we tested, we determined that the quality of the fit was approximately as for this location with parameters $\epsilon \in [0.2, 0.7]$, $\kappa \in [0.01, 0.1]$. Next, consider estimating the parameter a that specifies the degree of measurement noise. In the data, pressure measurements are produced by an instrument that consists of redundant digital pressure transducers. Brownian noise is usually the dominant noise component, though flicker noise and thermal noise also contribute for this transducer. The accuracy of the sensor is ± 0.02 inches of mercury, while the resolution is 0.003 inches of mercury for measurement and 0.005 inches of mercury for reporting. Due to the high accuracy of the sensor, the contribution of measurement noise is small, and we have chosen the parameter a to be 0.2.

Figure 8.2 compares the correlation between measurements T units apart with predictions from a model. Figure 8.3 compares the probability density of the difference between model noise 2 units apart with predictions from a model. The figures suggest that the model fit is satisfactory.

8.2 Message Generation Rates

We expect the rate at which messages are generated to decrease as the value of the threshold increases. We conducted the following experiment to estimate message rates. In the experiment, a message was generated when the model given above (average over the 5 years) deviated from the

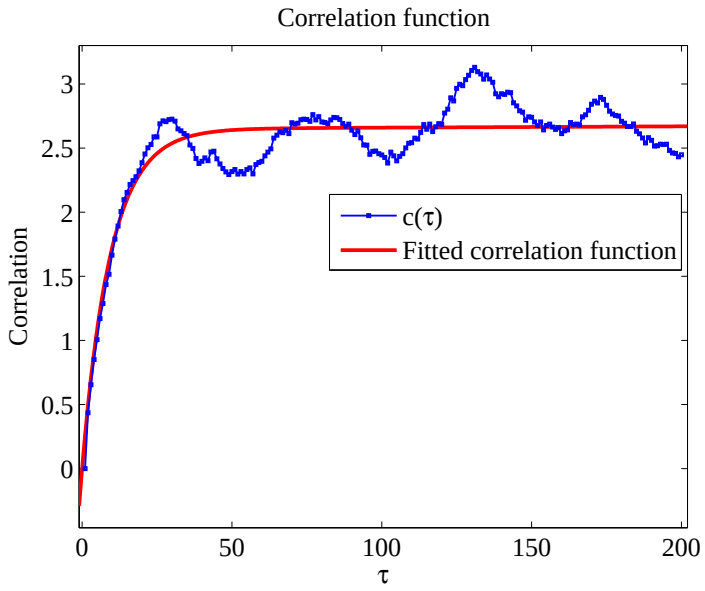


Figure 8.2: Correlation function obtained using hourly sea-level pressure measurements produced by sensors located at San Diego North Island. The fitted parameters are $\epsilon = 0.4972, \kappa = 0.027$

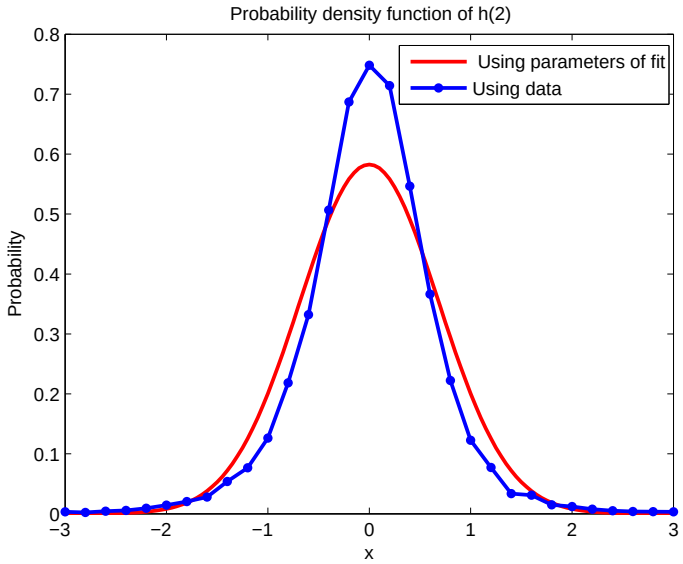


Figure 8.3: Estimated probability density function of the random variable $h(2)$ calculated using sea-level pressure observations reported by sensors located at San Diego North Island. The Gaussian probability density function of $h(2)$ computed using the process parameters $\epsilon = 0.4972$ and $\kappa = 0.027$ has mean 0 and variance 0.4783.

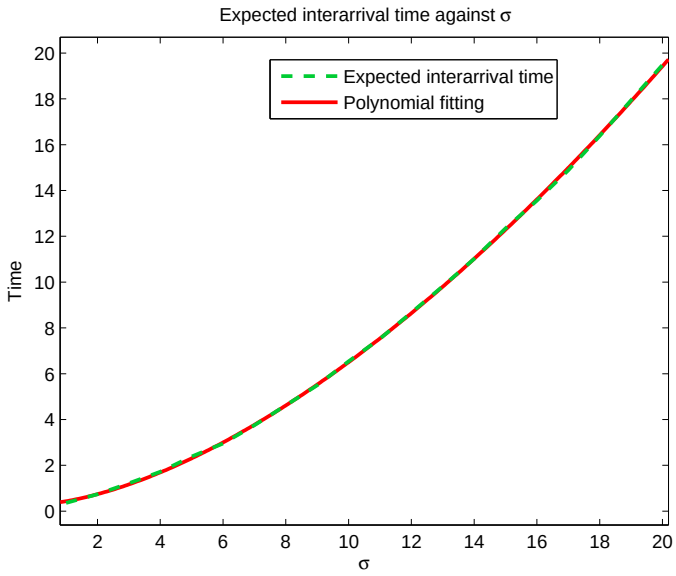


Figure 8.4: Average time between two consecutive messages. The parameters of the polynomial fitting are $a = 0.1485, b = 0.284, \alpha = 1.622$.

measurement by a value greater than the threshold. The model was run over all points —every hour of every day— over the 5 years. The average and maximum times between messages were computed for each value of the threshold. The resulting graphs are shown in Figures 8.4 and 8.5.

As expected, average and maximum message inter-arrival times increase with the threshold. A fitting with the polynomial $ax^{\alpha} + b$ was carried out, and the coefficients of the polynomials are shown.

8.3 Distribution of the Estimator Given Asynchronous Messages

Next, we evaluate how accurately the distribution of the estimator can be computed. At the instant a signal is received giving the value of the parameter, the estimator distribution is determined by the measurement noise. Later, the distribution is influenced by both measurement noise and model noise; the variance for both types of noise is monotonic non-decreasing with time while there are no signals. The estimator distribution converges to the equilibrium distribution as the time after the signal is received becomes large; this distribution is obtained by using only the smallest eigenvalue of the Fokker-Planck operator. The estimator distribution a short time after a signal is received can only be computed using several of the smaller eigenvalues.

We compare the reconstruction of the probability density function obtained using formula 7.10 with the parameters $a = 0.2, \epsilon = 0.4972, \kappa = 0.027$ with the probability density function computed

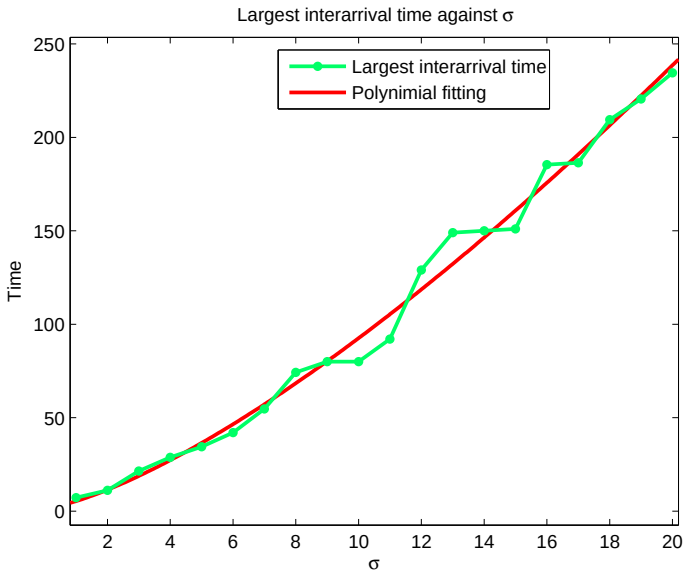


Figure 8.5: Largest time between two consecutive messages. The parameters of the polynomial fitting are $a = 3.782, b = 1.587, \alpha = 1.381$

	λ	p
$(1,0)$	-8.7909	1.2732
$(1,2)$	-10.1409	-0.3183
$(1,4)$	-11.4909	0.0398
$(1,6)$	-12.8409	-0.0033
$(1,8)$	-14.1909	0.00020723

Table 8.1: Values of the five largest eigenvalues and corresponding $p^{(0)}$ obtained using formula 7.10.

by means of numerical simulations. For our experiments we consider the time differences to be respectively 1.5 and 15.0 and fix $t_0 = 0$. Clearly, when $t = 1.5$, a larger number of eigenvalues contribute to the reconstruction of the probability function, while for $t = 15.0$ all eigenvalues except $\lambda_{1,0}$ decay and the probability density function in 7.10 converges to the stationary distribution given by the normalized eigenfunction $\phi_{1,0}$ corresponding to the smallest eigenvalue $\lambda_{1,0}$. Table 8.1 gives the values of the five largest eigenvalues obtained using equation 7.10.

The results are shown in Figures 8.6 and 8.7.

8.4 Decrease in Estimation Confidence With Message Absence

An advantage of predicate signaling is that the absence of messages conveys the information that measurements fit the model. Though the absence of signals conveys information, the presence of

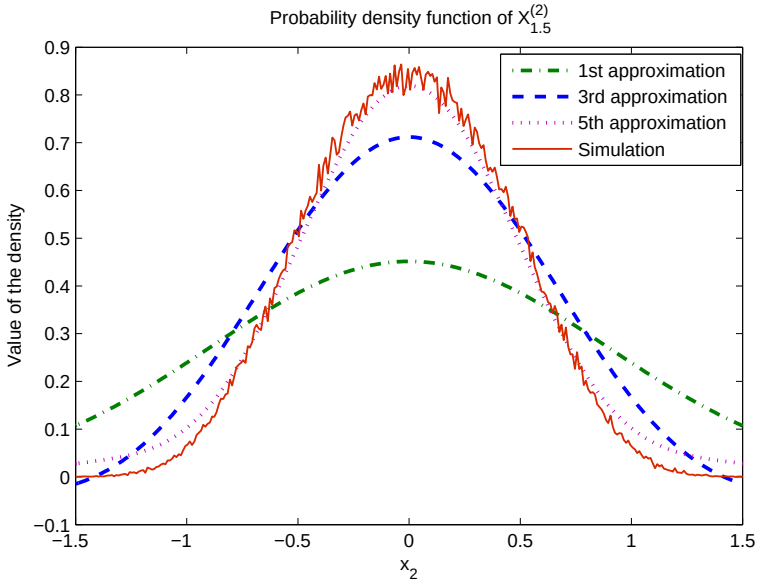


Figure 8.6: Probability density function of $X^{(2)}_{1.5}$.

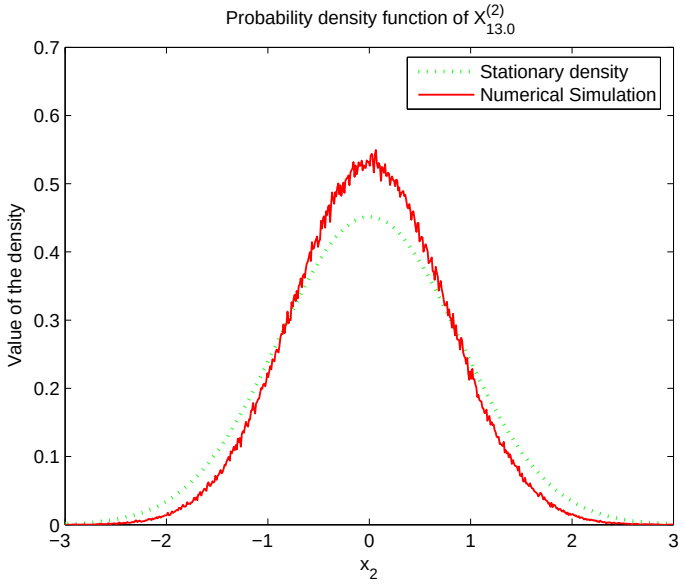


Figure 8.7: Probability density function of $X^{(2)}_{13}$.

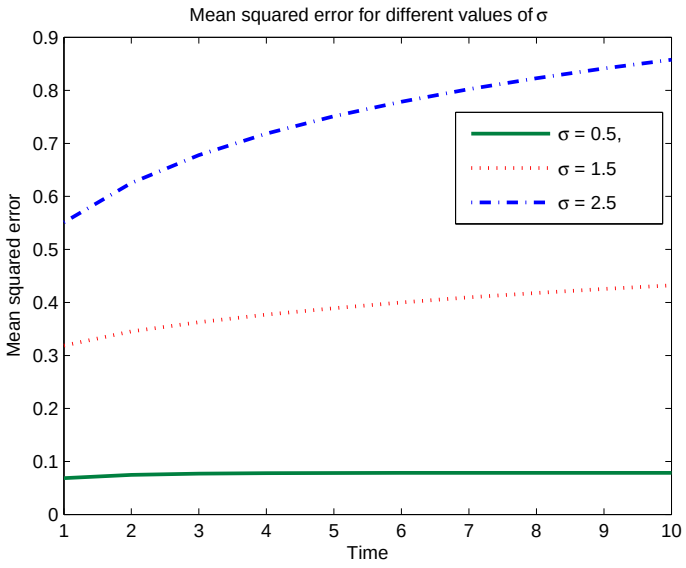


Figure 8.8: Expected loss functions for different values of σ under the assumption that the process is always within the first domain.

signals conveys even more information. The estimate for an unobservable parameter, at any point in time, is a random variable whose variance increases with the time since the last signal was received. A question that arises in predicate signaling is, how rapidly do confidence intervals for estimates grow? If, for instance, the 95% confidence interval gets large very rapidly, then a node fusing information from multiple locations is likely to make erroneous estimates if it hasn't received signals for some time. The rate at which the confidence interval increases is one of the factors that determines the time-driven aspect of predicate signaling. Even if measurements fit the model, signals may need to be generated merely so that confidence intervals of estimates are reasonably small.

Of course, while no signal is generated the measured value falls within the specified threshold. So, one approach to maintaining small confidence intervals is to make the threshold small. But a consequence of small thresholds, as we saw in the previous experiments, is that messages are generated more frequently. Thus, designers have to trade off message frequency against model accuracy. The next set of performance measures deals with this issue.

We have taken the variance computed using the probability density function in equation 7.10 as a measure of the estimation error. The mean squared error is minimized by the expectation of the signal given the measurements and the minimum expected loss is the variance.

Figure 8.8 shows the variance for different values of σ as a function of time, assuming the process to be all the time within the initial domain Ω_0.

Finally, consider a case where a message is lost in a system that uses a protocol in which messages are not resent until an acknowledgment is received. Assume that the next message sent after a

message is lost is indeed delivered to its destination. This second message repairs (at least some of) the damage of the lost message because each message that is delivered tells the receiver the model that the sender is currently using. What are the average and worst-case durations between the sending of two successive messages? The average time is proportional to $\frac{\sigma^2}{1+\epsilon^2}$ as shown in Appendix D. Experiments on the measured data shows that the worst-case time is indeed many hundreds of time steps (see Figure 8.5).

Chapter 9

Conclusions

The graph of average signal inter-arrival time shows the benefits of using model-based or anomaly-based signaling. We expect that more accurate models will give significantly reduced message generation rates. Even for the trivial model, the maximum time between signals is very large. Thus, the loss of a single message can result in the wrong model being used to estimate state for a long time. This suggests that pure anomaly-based signaling is inadequate for even trivial models. Some combination of the common signaling schemes should be used.

The increase in standard deviation (and thus confidence intervals) of estimators with time since the last signal suggests that there are benefits to incorporating query-based signaling: when the confidence interval gets unacceptably large, a query is sent asking for more recent measurements.

The relationships among the performance measures suggest systematic ways for designers to make tradeoffs between message rates and accuracy. Next, we discuss some of the weaknesses of predicate signaling and suggest ways of ameliorating the consequences of these weaknesses.

Sense and respond systems are useful when the rare event (the earthquake, the intruder, the virus) occurs. The model that represents the rare situation may not be as accurate as models that represent typical behavior. Thus, when the rare event occurs, the frequency of messages required to obtain satisfactory accuracy in estimation may be extremely high. This condition holds whether the signaling scheme is time-driven, anomaly-based or query-based; if we cannot model what is going on, we have to measure frequently. A criticism of predicate signaling is that worst-case message rates may be as bad as for time-driven signaling, and thus bandwidth requirements may be just as high.

Predicate signaling does have advantages even in this case. First, energy can be conserved during the long periods during which models accurately represent measurements. The conserved energy can be used when truly required: when actual behavior doesn't fit into expected norms. Second, even though worst-case bandwidths are high, the bandwidth can be used almost all the time for other applications. The bandwidth needs to be reserved for signaling only for the very rare periods during which models don't represent measurement.

Anomaly-based signaling assumes either that models can be computed rapidly or that results are precomputed and placed in flash memories so that the execution of a model can be reduced to a lookup. If the model maps a large number of observable parameters to an unobservable parameter then the cost of table lookup is significant. Initial experiments suggest the adequacy of simple models that make predictions based on the most recent measurements or on recent measurements combined with time.

Appendix A

Fokker-Planck Equations

Consider a diffusion process on $\mathbb{R}^d$ with generator $\hat{\mathcal{L}}$; the latter has a general form

$$\hat{\mathcal{L}}_{\boldsymbol{x}} = \sum_{i=1}^{d} b_i(\boldsymbol{x})\partial_{x_i} + \frac{1}{2}\sum_{i,j=1}^{d} a_{ij}(\boldsymbol{x})\partial^2_{x_i x_j}. \tag{A.1}$$

The probability density $p_t(\boldsymbol{x}|\boldsymbol{y})$ conditioned on the fact that the particle that starts at $\boldsymbol{y}$ at time 0 is located at $\boldsymbol{x}$ at time t satisfies the Fokker-Planck equation

$$\partial_t p_t(\boldsymbol{x}|\boldsymbol{y}) = \hat{\mathcal{L}}^*_{\boldsymbol{x}} p_t(\boldsymbol{x}|\boldsymbol{y}). \tag{A.2}$$

Here $\hat{\mathcal{L}}^*$ is the adjoint of $\hat{\mathcal{L}}$:

$$\hat{\mathcal{L}}^*_{\boldsymbol{x}} = -\sum_{i=1}^{d} \partial_{x_i} b_i(\boldsymbol{x}) + \frac{1}{2}\sum_{i,j=1}^{d} \partial^2_{x_i x_j} a_{ij}(\boldsymbol{x}). \tag{A.3}$$

Let Ω be an open subset of $\mathbb{R}^2$ with smooth boundary $\partial\Omega$. Let us add the initial condition to equation A.2 that the process has not touched the boundary $\partial\Omega$ up to time t. Furthermore, let $\varrho_t(\boldsymbol{x}|\boldsymbol{y})$ be as p_t, except conditional on the no-touching event. Formally,

$$\varrho(\boldsymbol{x}|\boldsymbol{y}) = p_t(\boldsymbol{x}|\boldsymbol{y})\left[\int_\Omega p_t(\boldsymbol{x}|\boldsymbol{y})\right]^{-1}. \tag{A.4}$$

Since p_t satisfies equation A.2 with the boundary condition $p_t(\boldsymbol{x}|\boldsymbol{y}) = 0$ for $x \in \partial\Omega$, we can rewrite it as

$$p_t(\boldsymbol{x}|\boldsymbol{y}) = \sum_{k=0}^{\infty} p_k(\boldsymbol{y}) e^{\lambda_k t}\phi_k(\boldsymbol{x}), \tag{A.5}$$

where λ_k and ϕ_k are respectively the eigenvalues and eigenvectors of $\widehat{\mathcal{L}}^*$ in Ω with zero boundary conditions (we set $0 \geq \lambda_0 \geq \lambda_1 \ldots \geq \lambda_n$). As shown in Appendix B, $\phi_0(\boldsymbol{x})$ is sign-definite; assume it is positive and normalized to have integral over Ω equal to unity. Observe that if ever the

stationary $\varrho(\boldsymbol{x}|\boldsymbol{y})$ exists, then $\phi_0(\boldsymbol{x}) = \lim_{t\to\infty} \varrho_t(\boldsymbol{x}|\boldsymbol{y})$ (for any $\boldsymbol{y} \in \Omega$) and thus the condition of sign-definiteness is equivalent to existence of the stationary conditional (on non-touching) distribution.

Suppose we know that our measured model deviation touched the boundary for the first time at time t; what is the distribution of its location on $\partial\Omega$? Since $\mathcal{L}^* = \nabla_x \cdot \widehat{\mathcal{F}}$, where $\widehat{\mathcal{F}}$ is the flux operator, we can use Gauss's theorem to show that the answer follows from the formula

$$\begin{cases} \int_\Omega \widehat{\mathcal{L}}^* p_t(\boldsymbol{x}) d\boldsymbol{x} & = & \int_{\partial\Omega} \widehat{\mathcal{F}} p_t(\boldsymbol{x}) \cdot d\boldsymbol{s}(\boldsymbol{x}) \\ \widehat{\mathcal{F}} p_t(\boldsymbol{x}) & := & \frac{1}{2}\partial_{\boldsymbol{x}} \cdot [\hat{a}(\boldsymbol{x}) p_t(\boldsymbol{x})] - \boldsymbol{b} \cdot p_t(\boldsymbol{x}) \end{cases} \tag{A.6}$$

On the boundary $\partial\Omega$, $p_t(\boldsymbol{x}) = 0$, thus $\widehat{\mathcal{F}} p_t(\boldsymbol{x}) = \frac{1}{2}\partial_{\boldsymbol{x}} \cdot [\hat{a}(\boldsymbol{x}) p_t(\boldsymbol{x})]$ on $\partial\Omega$. Here $\hat{a}$ is a matrix whose ij–th component is a_{ij}. Thus given a probability density $\varrho(\boldsymbol{x})$, $\widehat{\mathcal{F}}\varrho \cdot d\boldsymbol{s}$ is the probability flow through the element ds. This implies that the single layer density given by $(\widehat{\mathcal{F}}\varrho \cdot \boldsymbol{n})\delta_{\partial\Omega}(\boldsymbol{x})$ (not normalized as written) corresponds to the escape distribution on $\partial\Omega$.

Appendix B

0-order Approximate Operator

We find all eigenvalues and eigenfunctions for the operator

$$\hat{\mathcal{L}}^* = \hat{\mathcal{L}}_1^* + \hat{\mathcal{L}}_2^*, \quad \hat{\mathcal{L}}_1^* := \frac{1}{2}\partial^2_{x_1x_1} + \kappa\partial_{x_1}x_1, \quad \hat{\mathcal{L}}_2^* := \frac{1}{2}\partial^2_{x_2x_2}$$

in the domain $\Omega := \mathbb{R} \times [-L, L]$. In order to find the eigenvalues ν_n and the eigenfunctions $\xi_n(x_1)$ of $\hat{\mathcal{L}}_1^*$, we observe that the greatest (smallest by the absolute value) eigenvalue ν_0 and the corresponding eigenfunction $\xi_n(x_1)$ are given by

$$\nu_0 = 0, \quad \xi_0(x_1) = \sqrt{\frac{\kappa}{\pi}} e^{-\kappa x_1^2}, \tag{B.1}$$

Notice that the eigenfunction ξ_0 is normalized to have total integral unity. Setting $\xi_n(x_1) = \xi_0(x_1)h_N(x_1)$ we obtain equations for $h_n(x_1)$:

$$\partial^2_{x_1x_1} h_n(x_1) - 2\kappa x_1 \partial_{x_1} h_n(x_1) = 2\nu_n h_n(x_1). \tag{B.2}$$

This immediately implies that $h_n(x_1)$ are expressed using Hermite polynomials as

$$h_n(x_1) = H_n(\sqrt{\kappa}x_1), \tag{B.3}$$

thus we obtain

$$\nu_n = -\kappa n, \quad \xi_n(x_1) = \sqrt{\frac{\kappa}{\pi}} e^{-\kappa x_1^2} H_n(\sqrt{\kappa}x_1), \quad n = 0, 1, 2, \ldots \tag{B.4}$$

The operator $\hat{\mathcal{L}}_2^*$ has eigenvalues $\mu_m = -\pi^2 m^2/8L^2$ and eigenvectors

$$\begin{cases} \psi_m(x_2) &= \dfrac{\pi}{4L} \cos \dfrac{\pi m x_2}{2L} \ (m = 1, 3, \ldots) \\ \\ \psi_m(x_2) &= \dfrac{\pi}{4L} \sin \dfrac{\pi m x_2}{2L} \ (m = 2, 4, \ldots). \end{cases} \tag{B.5}$$

Since

$$\hat{\mathcal{L}}^*[\xi_n(x_1)\psi_m(x_2)] = (\hat{\mathcal{L}_1}^* + \hat{\mathcal{L}_2}^*)[\xi_n(x_1)\psi_m(x_2)],$$

and noticing that

$$\hat{\mathcal{L}_1}^*[\xi_n(x_1)\psi_m(x_2)] = \nu_n[\xi_n(x_1)\psi_m(x_2)],\ \hat{\mathcal{L}_2}^*[\xi_n(x_1)\psi_m(x_2)] = \mu_m[\xi_n(x_1)\psi_m(x_2)],$$

we can conclude that

$$\hat{\mathcal{L}}^*[\xi_n(x_1)\psi_m(x_2)] = (\nu_n + \mu_m)\xi(x_1)\psi_m(x_2).$$

Therefore, $\hat{\mathcal{L}}^*$ has the following eigenvalues and eigenvectors:

$$\begin{cases} \lambda_{m,n} = -\dfrac{\pi^2 m^2}{8L^2} - \kappa n \\ \\ \phi_{m,n}(x) = \dfrac{\sqrt{\kappa\pi}}{4L} e^{-\kappa x_1^2} H_n(\sqrt{\kappa}x_1) \cos\dfrac{\pi m x_2}{2L}, \end{cases} \tag{B.6}$$

where $n = 0, 1, 2, \ldots$, $m = 1, 3, 5, \ldots$. If $m = 2, 4, 6, \ldots$, "cos" function is changed to "sin". Observe that $\phi_{1,0}(x)$ is positive with total integral over Ω equal to unity. Since Hermite polynomials are a set of orthogonal polynomials in $\mathbb{R}$ the functions $\varphi_{m,n}$ are defined as

$$\varphi_{m,n} = \frac{4}{\pi 2^n n!} H_n(\sqrt{\kappa}x_1) \cos\frac{\pi m x_2}{2L}; \tag{B.7}$$

as before, "sin" has to be taken if m is even. The normalizing coefficient is chosen to have

$$\int_\Omega \phi_{m,n}(\boldsymbol{x})\varphi_{m',n'}(\boldsymbol{x})d\boldsymbol{x} = \delta_{m,m'}\delta_{n,n'}.$$

Appendix C

Eigenfunctions in the Diagonal Strip

It has been shown in Appendix A that our probability density conditioned on the fact that the domain has not been touched is the solution to the eigenvalue problem formulated in equation A.2. Thus, we need to find the eigenvalues and eigenfunctions of the Fokker-Planck operator

$$\hat{\mathcal{L}}^* = \frac{1}{2}[\partial^2_{x_1x_1}] + \epsilon^2\partial^2_{x_2x_2} + \kappa\partial_{x_2}x_2$$

in a strip $\Omega := \{(x_1, x_2) : |x_1 + x_2| < \sigma\}$ (zero boundary conditions). Introducing

$$y_1 = \frac{x_1 + x_2}{\sqrt{1+\epsilon^2}}, \quad y_2 = \frac{\epsilon x_1 - x_2/\epsilon}{\sqrt{1+\epsilon^2}}, \tag{C.1}$$

we obtain

$$\hat{\mathcal{L}}^* = \frac{1}{2}[\partial^2_{y_1y_1} + \partial^2_{y_2y_2}] + \frac{\kappa}{1+\epsilon^2}[\partial_{y_2}y_2 - \epsilon(y_1\partial_{y_2} + y_2\partial_{y_1}) + \epsilon^2\partial_{y_2}y_2],$$

and the domain becomes $\Omega = \{(y_1, y_2) : |y_1| < \frac{\sigma}{\sqrt{1+\epsilon^2}}\}$. Regular perturbation theory may now be applied to this problem. For the lowest order approximation we may use the results of Appendix B to get

$$\begin{cases} \lambda_{m,n} & \approx & -\dfrac{\pi^2 m^2}{8L^2} - \kappa n \\ \phi_{m,n}(x) & \approx & \dfrac{\sqrt{\kappa\pi}}{4L} e^{-\kappa y_2^2} H_n(\sqrt{\kappa}y_2) \cos\dfrac{\pi m(x_1+x_2)}{2\sigma} \end{cases}$$

("sin" has to be taken if m is even).

Appendix D

Expected Interarrival Times

The objective of this appendix is to derive an analytic formula for the expected value of the time between two consecutive exit times (*i.e.*, the time when the boundary of the current domain is touched). This is obtained by using the Dynkin formula [23], which states:

Let $f \in \mathbb{L}^2(\Omega)$. Suppose τ is a stopping time such that $E[\tau] < \infty$. Then

$$E[f(X_\tau^x)] = f(x) + E[\int_0^\tau \hat{\mathcal{L}} f(X_s^x), ds] \tag{D.1}$$

where X_t^x is a process starting at x at time 0.

We want to choose a function f such that

(1) $f(X_\tau^x)$ is easy to compute and does not depend on τ, *e.g.*, f is identically zero on the boundary of the domain.

(2) $\hat{\mathcal{L}} f$ is easy to compute and the result is a simple function of τ, *e.g.*, $\hat{\mathcal{L}} f = 1$, so that the the integral in equation D.1 is just τ.

Applying Dynkin's formula for such a choice of f we would obtain $0 = f(x) + E[\tau]$, thus $E[\tau] = -f(x)$. This gives the expectation of the hitting time τ. Furthermore, it implies that f is the solution of the differential equation $\hat{\mathcal{L}} f = 1$ with Dirichlet (zero) boundary conditions. It can be easily verified that $f(x_1, x_2) = \frac{(x_1+x_2)^2}{1+\epsilon^2} - \frac{\sigma^2}{1+\epsilon^2}$ solves the equation. Under the assumption that the process starts at $(0, 0)$ at time 0, the expected value of τ equals $\frac{\sigma^2}{1+\epsilon^2}$.

Appendix E

Adjoint of a Linear Operator

Consider a linear operator of the form

$$L = \sum_{i=1}^{d}\sum_{j=1}^{d} \alpha_{i,j}(\boldsymbol{x}) \frac{\partial^2}{\partial x_i \partial x_j} + \sum_{i=1}^{d} \beta_i(\boldsymbol{x}) \frac{\partial}{\partial x_i}$$

where $\alpha_{i,j} : \mathbb{R}^d \to \mathbb{R}$, $\beta_i : \mathbb{R}^d \to \mathbb{R}$ are nice enough functions that have continuous second order derivatives with respect to the independent variables $x_1, \ldots, x_d$. For any pair of functions u, v: $\mathbb{R}^d \to \mathbb{R}$ we have

$$\begin{aligned} v\alpha_{i,j}\tfrac{\partial^2 u}{\partial x_i \partial x_j} &= \tfrac{\partial}{\partial x_i}\left(v\alpha_{i,j}\tfrac{\partial u}{\partial x_j}\right) - \tfrac{\partial u}{\partial x_j}\tfrac{\partial}{\partial x_i}(\alpha_{i,j}v) = \\ &= \tfrac{\partial}{\partial x_i}\left(\alpha_{i,j}v\tfrac{\partial u}{\partial x_j}\right) - \tfrac{\partial}{\partial x_j}\left(u\tfrac{\partial}{\partial x_i}(\alpha_{i,j}v\right) + u\tfrac{\partial^2}{\partial x_i \partial x_j}(\alpha_{i,j}v). \end{aligned}$$

The steps above need some justification. The first equality follows because

$$\begin{aligned} \frac{\partial}{\partial x_i}\left(v\alpha_{i,j}\frac{\partial u}{\partial x_j}\right) - \frac{\partial u}{\partial x_j}\frac{\partial}{\partial x_i}(\alpha_{i,j}v) &= \frac{\partial}{\partial x_i}(v\alpha_{i,j})\frac{\partial u}{\partial x_j} + v\alpha_{i,j}\frac{\partial}{\partial x_i}\frac{\partial u}{\partial x_j} - \frac{\partial u}{\partial x_j}\frac{\partial}{\partial x_i}(\alpha_{i,j}v) = \\ &= v\alpha_{i,j}\frac{\partial^2 u}{\partial x_i \partial x_j.} \end{aligned}$$

The second equality follows because

$$\begin{aligned} &\frac{\partial}{\partial x_i}\left(\alpha_{i,j}v\frac{\partial u}{\partial x_j}\right) - \frac{\partial}{\partial x_j}\left(u\frac{\partial}{\partial x_i}(\alpha_{i,j}v)\right) + u\frac{\partial^2}{\partial x_i \partial x_j}(\alpha_{i,j}v) = \\ &= \frac{\partial}{\partial x_i}\left(\alpha_{i,j}v\frac{\partial u}{\partial x_j}\right) - \frac{\partial u}{\partial x_j}\frac{\partial}{\partial x_i}(\alpha_{i,j}v) - u\frac{\partial^2}{\partial x_j x_i}(\alpha_{i,j}v) + u\frac{\partial^2}{\partial x_i \partial x_j}(\alpha_{i,j}v) = \\ &= \frac{\partial}{\partial x_i}\left(\alpha_{i,j}v\frac{\partial u}{\partial x_j}\right) - \frac{\partial u}{\partial x_j}\frac{\partial}{\partial x_i}(\alpha_{i,j}v). \end{aligned}$$

The previous equalities hold for each pair of integers $i, j \in \{1, \ldots, d\}$.

Also, notice that

$$\frac{\partial}{\partial x_i}(\beta_i uv) - u\frac{\partial}{\partial x_i}(\beta_i v) \quad = \quad \frac{\partial}{\partial x_i}(v\beta_i)u + v\beta_i\frac{\partial u}{\partial x_i} - u\frac{\partial}{\partial x_i}(\beta_i v) \quad = \quad v\beta_i\frac{\partial u}{\partial x_i}. \tag{E.1}$$

Now,

$$\begin{aligned}
vLu &= v\Big(\sum_{i=1}^{d}\sum_{j=1}^{d}\alpha_{i,j}\frac{\partial^2 u}{\partial x_i \partial x_j}\Big) + \sum_{i=1}^{d}\beta_i\frac{\partial u}{\partial x_i} = \\
&= \sum_{i=1}^{d}\sum_{j=1}^{d}v\alpha_{i,j}\frac{\partial^2 u}{\partial x_i \partial x_j} + \sum_{i=1}^{d}v\beta_i\frac{\partial u}{\partial x_i} = \\
&= \sum_{i=1}^{d}\sum_{j=1}^{d}u\frac{\partial^2(\alpha_{i,j}v)}{\partial x_i \partial x_j} - u\sum_{i=1}^{d}\frac{\partial}{\partial x_i}(\beta_i v) + \sum_{i=1}^{d}\sum_{j=1}^{d}\frac{\partial}{\partial x_i}\Big(v\alpha_{i,j}\frac{\partial u}{\partial x_j}\Big) + \\
&\quad - \sum_{i=1}^{d}\sum_{j=1}^{d}u\frac{\partial}{\partial x_i}(\alpha_{i,j}v) + \sum_{i=1}^{d}\beta_i uv = \\
&= u\Big[\sum_{i=1}^{d}\sum_{j=1}^{d}\frac{\partial^2(\alpha_{i,j}v)}{\partial x_i \partial x_j} - \sum_{i=1}^{d}\frac{\partial(\beta_i v)}{\partial x_i}\Big] + \\
&\quad + \sum_{i=1}^{d}\frac{\partial}{\partial x_i}\Big(v\sum_{j=1}^{d}\alpha_{i,j}\frac{\partial u}{\partial x_j} - u\sum_{j=1}^{d}\frac{\partial}{\partial x_i}(\alpha_{i,j}v) + \beta_i uv\Big).
\end{aligned} \tag{E.2}$$

Denote by L^* the linear operator

$$L^* = \sum_{i=1}^{d}\sum_{j=1}^{d}\frac{\partial^2(\alpha_{i,j})}{\partial x_i \partial x_j} - \sum_{i=1}^{d}\frac{\partial}{\partial x_i}\beta_i.$$

Also, let $\boldsymbol{P}$ be a vector whose i^{th} component is a scalar function defined by

$$\boldsymbol{P}_i = v\sum_{j=1}^{n}\alpha_{i,j}\frac{\partial u}{\partial x_j} - u\sum_{j=1}^{d}\frac{\partial}{\partial x_i}(\alpha_{i,j}v) + \beta_i uv.$$

We can rewrite equation E.2 as

$$vLu - uL^*v = \nabla\boldsymbol{P}, \tag{E.3}$$

which is also called Lagrange's identity.

Integrating both sides in eq. E.3 we obtain

$$\int_{\mathbb{R}^d} vLu - uL^*v d\boldsymbol{s} = \int_{\mathbb{R}^d}\nabla\boldsymbol{P}d\boldsymbol{s}.$$

Using Gauss's theorem,

$$\int_{\mathbb{R}^d}\nabla\boldsymbol{P}d\boldsymbol{s} = \int_{\partial\mathbb{R}^d}\boldsymbol{P}\boldsymbol{n}d\boldsymbol{s}, \tag{E.4}$$

where $\boldsymbol{n}$ is the outward normal of $\partial\mathbb{R}^d$.

We consider the inner product space $\mathbb{L}^2(\mathbb{R}^d)$ of all square integrable $\mathbb{R}^2$ valued functions with usual inner product (f, g), defined by

$$(f, g) = \int_{\mathbb{R}^2} f(\boldsymbol{x})g(\boldsymbol{x})d\boldsymbol{x}.$$

If the identity $(v, Lu) = (u, L^*v)$ holds, then L^* is called the *adjoint* operator. If u and v are identically zero on the boundary, then the identity holds.

Bibliography

[1] A.Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao. Habitat monitoring: application driver for wireless communications technology. In *Proceedings of the 2001 ACM SIGCOMM Workshop on Data Communications in Latin America and the Caribbean*, pages 20–41. ACM, 2001.

[2] A.Mainwaring, J. Polastre, R. Szewczyk, D. Culler, and J. Anderson. Wireless sensor networks for habitat monitoring. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Application, WSNA02, Atlanta, USA*, pages 88–97. ACM, 2002.

[3] B. Aydemir, K.M. Chandy, E.M. Karpilovsky, and D.M. Zimmerman. Event webs for crisis management. In *2nd IASTED International Conference on Communications, Internet and Information Technology*, 2003.

[4] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Invited paper in Proc. of the 2002 ACM Symp. on Principles of Database Systems (PODS 2002)*. ACM, 2002.

[5] S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S.R. Madden, V. Raman, F. Reiss, and M.A. Shah. TelegraphCQ: Continuous dataflow processing for an uncertain world. *Conference on Innovative Data systems Research*, 2003.

[6] Sirish Chandrasekaran and Michael J. Franklin. Remembrance of streams past: Overload-sensitive management of archived streams. *VLDB*, 2004.

[7] K. Mani Chandy and Leslie Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computing Systems*, 3(1):63–75, February 1985.

[8] K. Mani Chandy and Jayadev Misra. Distributed simulation: A case study in design and verification of distributed programs. *IEEE:Transactions on Software Engineering*, 5(5):440–452, September 1979.

[9] A. Deshpande, C. Guestrin, S.R. Madden, J.M. Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. *Proceedings of the 30th International Conference on Very Large Data Bases*, 2004.

[10] C.W. Gardiner. *Handbook of Stochastic Methods for Physics, Chemistry and the Natural Sciences*, volume 13 of *Springer Series in Synergetics*. Springer, third edition, 2004.

[11] P. Gibbons and Y. Mathias. Synopsis data structures for massive data sets. *External Memory Algorithms, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 50, 1999.

[12] F. Gustafsson. *Adaptive filtering and change detection*. John Wiley and Sons, Boston, 2000.

[13] S. Haeckel. The adaptive enterprise: Creating and leading sense and respond applications. *Harvard Business Review*, 1999.

[14] S. Haeckel and N. Rolan. Managing by wire. *Harvard Business Review*, September 1993.

[15] J.Y. Halpern, R. Fagin, Y. Moses, and M.Y. Vardi. *Reasoning about knowledge*. MIT Press, 1995.

[16] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*. SIAM, 1991.

[17] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Sixth Annual International Conference on Mobile Computing and Networking*, pages 56–67. ACM, 2000.

[18] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004*, pages 180–191. VLDB, 2004.

[19] S. Krishnamurthy, S. Chandrasekaran, O. Cooper, A. Deshpande, M. Franklin, J. Hellerstein, W. Hong, S. Madden, V. Raman, F. Reiss, and M. Shah. TelegraphCQ: An architectural status report. *IEEE Data Engineering Bulletin*, Vol. 26(1), March 2003.

[20] G. Lorden. Procedure for reacting to a change in distribution. *Ann.Math.Statistics*, 42:1897–1908, 1971.

[21] S. Madden, M. Franklin, J. Hellerstein, and W. Hong. TAG: Tiny aggregation service for ad-hoc sensor networks. *OSDI*, 2002.

[22] Y. Mei. Asymptotically optimal methods for sequential change-point detection. Technical Report etd-05292003-133431, California Institute of Technology, 2003.

[23] B. Oksendal. *Stochastic Differential Equations: An Introduction with Applications.* Springer. Springer, sixth edition, 2003.

[24] S. Roberts. A comparison of some control chart procedures. *Technometrics*, 8:411–430, 1966.

[25] J. Hellerstein S. Madden, M. Franklin and W. Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD*. ACM, 2003.

[26] A. Shiryayev. On optimum methods in quickest detection problems. *Theory Prob. and Applic.*, 8:22–46, 1963.

[27] A.H. Sayed T. Kailath and B. Hassibi. *Linear Estimation.* Prentice-Hall, 2000.

[28] Y. Yao and J. Gehrke. Query processing for sensor networks. *CIDR*, 2003.

[29] F. Zhao and L. Guibas. *Wireless sensor networks: an information processing approach.* Morgan Kaufmann, first edition, 2004.