February 1992

LIDS-TH-2092

Research Supported By:

Air Force Office of Scientific Research
Grant AFOSR-F49620-92-J-0002

Office of Naval Research
Grant ONR N00014-91-J-1004

National Science Foundation
Grant NSF 9015281-MIP

Parallel Estimation on One and Two Dimensional Systems

Darrin Taylor

February 1992 LIDS-TH-2092

PARALLEL ESTIMATION ON ONE AND TWO DIMENSIONAL SYSTEMS

by

D. Taylor

This report is based on the unaltered thesis of Darrin Taylor submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the Massachusetts Institute of Technology in February 1992. This research was carried out at the M.I.T. Laboratory for Information and Decision Systems and was supported in part by the Air Force Office of Scientific Research under grant AFOSR-F49620-92-J-0002, Office of Naval Research under grant ONR N00014-91-J-1004 and by the National Science Foundation under grant NSF 9015281-MIP.

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology
Cambridge, MA 02139

Parallel Estimation on One and Two Dimensional Systems

by

Darrin Taylor

B.S. Massachusetts Institute of Technology (1984)
S.M., Massachusetts Institute of Technology (1987)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1992

Author .....
Department of Electrical Engineering and Computer Science
February 4, 1992

Certified by .....
Alan S. Willsky
Professor, Electrical Engineering
Thesis Supervisor

Accepted by .....
Campbell L.Searle
Chairman, Departmental Committee on Graduate Students

Parallel Estimation on One and Two Dimensional Systems

by

Darrin Taylor

Submitted to the Department of Electrical Engineering and Computer Science
on February 4, 1992, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Parallel smoothing algorithms are developed for one and two dimensional systems. The two-dimensional algorithms are completely general and readily extend to the case of higher dimensions. These algorithms take advantage of the reciprocal nature (Markovianity) of the processes, and Maximum Likelihood (ML) estimation philosophy to partition a discrete two-dimensional field into subregions where local statistically independent filtering is performed resulting in locally optimal ML estimates of local boundaries. Filtering is performed in a radial direction from the center of the subregion out towards the boundary of the subregion. Subsequently an interprocessor data exchange step follows where data with precise statistical interpretations are exchanged. Finally each subregion can be updated independently and in parallel, while processing radially inward from the boundary of the subregion towards the center of each subregion. This algorithm has a tree structure typical of divide and conquer algorithms and is well suited for implementation on a hypercube architecture. In addition this recursive implementation has the same structure as the Rauch-Tung-Striebel Algorithm. Several smoothing algorithms are developed in one dimension based upon the two-dimensional algorithms and more traditional algorithms suitable only for one-dimensional smoothing are developed. Both the one and two-dimensional algorithms have the property that the information communicated between processors is the optimal estimate of the state of the process based on an appropriate set of data. As a result a degree of fault tolerance in the smoothing algorithm exists. If a processor is unable to produce the desired information due to failure, the information computed by other processors has a statistical meaning sufficient to allow us the option of computing the optimal estimate based solely on the available information. Furthermore, it is a simple task to alter the algorithm to compute the estimate of the system based on a predetermined subset of the data. This is called Limited Support Estimation (LSE), and the parallel nature of the algorithms allow LSE schemes to be implemented efficiently, speeding up our algorithm in cases in which some of the data contributes only marginally to the accuracy of the smoothed estimate of a given state. A trade-off between accuracy and time is established, and optimal partitioning of the data is discussed. To aid in the analysis of the smoothing algorithms, general recursive estimation algorithms and square root recursive algorithms are developed to

produce ML estimates which account for the cases of both perfect measurements and states which are not completely estimable. Common recursive smoothing algorithms such as the Mayne-Fraser and Rauch-Tung-Striebel algorithm are developed for the Separable Two Point Boundary Value descriptor Systems (STPBVDS). STPBVDS are general acausal one-dimensional systems which are used to construct models of two-dimensional systems with a radial time coordinate, and a state whose dimension changes with time.

Thesis Supervisor: Alan S. Willsky
Title: Professor, Electrical Engineering

Acknowledgments

I like to thank Professor Alan S. Willsky for his patience, insight, and guidance through all of this. Few can match is enthusiasm and insight.

Mrs. Clark how you knew I would come to this point is beyond me. Back in 1980 I must have surely called you crazy, or thought so but here I am. Your confidence has remained with me all along. Janie... thanks. I would not have started this if not for you. I could not finish if not for you. It's so hard to say good bye to yesterday. Thanks also to Lynda Jordan who told me that the difference between me and her was just a matter of time.

Honorable mention: Rosalinda, Mark and Master Luke for being the closest thing to family for me.

Cynthia... Great minds think alike!

Pnina and Adam for having been the closest thing to family for me.

Gloria thanks for being there every day.

Jeffrey H Lang thanks for your support and help.

Linda Nathan, thanks for your friendship and helping me to pursue some of my higher interests. Norman Fortenberry Erastus

Jenneine

Janet Roberts Magwood

Captain Kenny G thanks for help to make MIT livable and providing the other point of view. GrandDad.....

Darryl D Williams thanks for letting me steal you name! Robert

Mom & Dad.

Others: Susan and Marjorie, thanks for putting up with me. Thanks Julie Maria Sierra for harassing me for 11 years. Thanks to Jennifer Nichole Sykes for harassing me for the last year and a half. Vivian Kim Angela Perkins

Other others: Karl Wyatt Derek Washington Gallon, Isaac Kwame Sah Baidoo, Ray E. Samuel, Jackie

Contents

List of Figures

Chapter 1

Introduction

With the advent of parallel processing environments comes a corresponding change in the way algorithms and computations are designed and performed. The Kalman filter which is both the culmination and the basis of a large part of our knowledge of statistical signal processing, was formulated as a recursive algorithm in the age of single processor environments. With parallel processing environments there is much discussion that recursive algorithms related to the Kalman filter should be abandoned in favor of iterative techniques for solving linear systems of equations. With advanced iterative techniques such as multigrid, what was discussion has become cries for the use of iterative techniques for solving linear systems of equations. Iterative algorithms tend to parallelize easily, are applicable to a wider choice of linear equations, and achieve comparable accuracy to other methods of solving linear equations.

There is more however that we may demand from our algorithms. Since the class of linear systems of equations under study is not arbitrary but quite specific, we can benefit from our knowledge of dynamic systems and stochastic processes to arrive at algorithms which reflect this knowledge. This is one of the major advantages of the Kalman filter. The Kalman filter is important today not only because it is a recursive implementation, but more importantly because it is formulated as a sequence of Bayesian estimation problems, each problem producing an estimate with a clear and precise statistical interpretation. Each stage of the computation is understood thoroughly. If the computation ends prematurely for any reason, the truncated results

have meaning and implications towards the final desired result. With the deeper understanding gained through knowledge of stochastic processes, we will see that the issue is not only to find clever ways of partitioning a problem into subproblems to be computed in parallel but also how to link small well understood problems together in a manner which represents the solution to larger and larger problems of increasing complexity. We therefore emphasize the importance that statistically meaningful information be generated at all points in the course of our computations.

There are many benefits of this approach. Consider a distributed processing and decision-making environment. Clearly optimal decisions are best made using globally optimal (smoothed) data. This global information may not be available in a timely fashion, and therefore it may be necessary to to use local data to make at least preliminary decisions. It is best, as a result, that the local information be processed in a manner that makes it directly useful for such decision-making. The benefits of generating statistically meaningful quantities is also important in the parallel computing environment. As the number of processors increase, the probability of a processor failure in any interval of time increases. While in some military applications we may be able to perform all calculations in triplicate and vote out discrepant computations, this may be too costly in general. We may be forced to carry out our computations without the data available from a failed processor. This lends support to the notion of performing limited support estimation, i.e. optimal estimation based on some subset of the data. In addition global information is often not necessary to produce accurate estimates of the state. Limited support estimation could therefore be a way to produce accurate smoothed estimates while reducing the computational burden. Still the estimates produced have precise statistical meaning, as do their corresponding error covariances which provide us with the information needed to asses their quality. We will show that there is little algorithmic difference between full support and limited support estimation, with the algorithms which are based upon producing estimates recursively, and in parallel. This now produces a trade-off between computational complexity of the algorithm and precise statistical accuracy of the estimates and therefore it is possible to compute the optimal number of processors, and par-

titioning of the data given a certain desired level of accuracy. Another reason for which we are interested in producing precise estimates, is that we would like to be able to model the effects of the process outside of a local region as being manifested solely at the boundary of the local region. Statistically it should be the case that in a parallel algorithm, in each local region all of the external processing can be reduced to a single measurement of the boundary of a local process. If this is understood then it could become an important ingredient in producing efficient parallel algorithms for random processes of arbitrary dimension.

Although a multitude of algorithms exist in one-dimension [13] ,[14], [15], [16], [29], [11], [12], they in general do not generalize to higher dimensional algorithms. Algorithms which are applicable to higher dimensions allow us to learn what are the important similarities between one dimensional systems and their higher dimensional counterparts. This knowledge should then be able to find applications more generally in the field of digital signal processing and other related disciplines. The application to higher dimensions necessitates the use of parallel processing environments. In two-dimensions the boundary of a process grows at least as the square root of the size of the region. The boundary provides a notion of the state of the system, and in addition indicates its complexity. As a result, to manage the complexity of the system, it makes sense to partition the region into subregions and have individual processors work on 'smaller' problems. What remains is to determine exactly what each processor computes and how the processors communicate to each other.

The parallel estimation algorithms described here have common characteristics. First, the data is partitioned among the processors. Local calculations are performed by processors on their own sets of data. Local information is then exchanged between processors and this is followed by a parallel post-processing step in which each processor updates the estimates on its subinterval to produce the final globally optimal estimate over the entire data interval. While a variety of approaches have been developed for various optimal estimation problems [13] ,[14], [15], [16], [29], [11], [12], only two of these[14], [15] employ a similar data partitioning structure for parallel filtering and smoothing for causal systems. In [14] a square root algorithm is used for parallel

filtering on the subintervals assuming perfect knowledge of the state at one endpoint. This is followed by an interprocessor information exchange and computation step. This step which is based on a change of initial condition formula in order to correct for imperfect endpoint knowledge, is similar in structure to the Mayne-Fraser two filter smoother in order to obtain optimal smoothed estimates at the boundaries of the data intervals and to allow subsequent parallel computation of smoothed estimates within each subinterval. A somewhat more efficient algorithm, with a similar structure, is described in /citeTewfik. This procedure deals symmetrically with the two endpoints of each subinterval by initially processing data outward toward and in the final step inward from the boundary points (essentially using in each interval a joint model for $x(k)$ and $x(-k)$, with the time index $k = 0$ corresponding to the center of the interval). The interprocessor exchange step makes use of the so-called partition theorem [19], resulting again in a two-filter sweep from processor to processor in both directions to produce optimal estimates at all boundary points.

One issue which needs to be addressed is that of modeling. The basic class of systems on which we focus for the most part in this thesis are Separable Two Point Boundary Value Descriptor Systems(STPBVDS's). These systems are naturally acausal in that the dynamics are descriptor, and the boundary conditions are independently specified in part, at each end of the interval over which the system is defined. As we will see, this class of systems is in fact rather large, as we can in fact use such systems to model the case where the boundary conditions at each end are coupled. In addition, if the dimension of the state is allowed to vary, multi-dimensional problems are also accommodated. In particular, an advantage of using STPBVDS's is that they are able to be solved without the aid of shooting algorithms which most solutions to two point boundary value descriptor systems (TPBVDS's) without separable boundary conditions require[24],[2], [20]. As we will see, general non-separable TPBVDS, and multidimensional systems, can be converted to 1-D STPBVDS's by defining radial recursions. In particular, for multi-dimensional processes the state is defined along the perimeter of a square of a given radius. This notion of defining the state as a function of radius applies in a consistent fashion to the one-dimensional

systems. As a result filtering a (rectangular) subregion for a two-dimensional system or an interval for a one-dimensional system involves filtering outward towards the boundary from the center and back from the boundary towards the center.

STPBVDS have boundary conditions specified at each end of the interval which are independent both algebraically and statistically. However with the exception of causal systems, the boundary condition is not specified, completely at either end. In general only partial information is known. There are times in fact where this partial information is perfect. In these cases a filtering algorithm would need to be initialized with an ill-defined covariance matrix because part of the state is known perfectly, and part is completely unknown.

In well-posed STPBVDS, if the state is estimable given all of the available data, it may not be estimable given only causal or anticausal data. However in a parallel processing algorithm, it may not be desirable to propagate a priori information to each subregion in order to avoid preprocessing. Since the elimination of the a priori information for a local subinterval processor implies that the state may in fact be (locally) non-estimable Bayesian estimation is largely abandoned here in favor of Maximum Likelihood (ML) estimation.

ML estimation is used at each stage of our processing to compute estimates based on the locally available information. Here as in [23], we essentially adopt the perspective that a priori statistics, dynamical relationships, and actual observations all play the same role, namely as noisy constraints. The use of this formalism has several important implications, perhaps most notably in the simplification and greatly enhanced flexibility it provides us in the interprocessor exchange step. However, let us first comment on some of the implications for the local processing step.

Recursive ML estimation requires the confrontation of the problem of estimation in the face of degeneracy, where the linear equations yielding the ML estimate need not have a unique solution (so that at least some part of $x(t)$ is unconstrained by available information) but may yield perfect estimates of other parts of $x(t)$. The framework for generalized estimation in the static case is developed in [18] (see also [17]). In [23] the results of [18] are used to develop recursive filtering procedures for TPBVDS

in the case when all variables are estimable, (so that P is well defined). What we describe in Chapter 4 are algorithms for optimal STPBVDS smoothing in the general case. In particular we describe generalizations of the well known Mayne-Fraser and Rauch-Tung-Striebel algorithms and in fact provide a completely symmetric version of the first of these in which each of the two filters is initialized with the independent boundary information available to it. These algorithms in addition to being of interest in their own right, also provide us with the initial local and final local processing steps for our data partitioned parallel processing procedures. Three new algorithms for the middle interprocessor data exchange step are also described for the one dimensional case. In the multidimensional case, a new algorithm is presented which is highly parallel, and takes advantage of the reciprocal nature of the process, in order to efficiently compute globally smoothed estimates of regions of finite dimension. As in [14], [15], we can view the output of the first step as producing 'measurements' of $x(k)$ at the boundaries. However in the Bayesian approaches of [14], [15], the errors in these 'measurements' are correlated since each local processor makes use of common prior information. This leads to the comparatively involved two filter procedure in [14], [15], for exchanging and fusing endpoint information among processors. In contrast, by adopting the ML formalism we guarantee that the result of out first local processing step produces independent 'measurements' of boundary points. This leads to an algorithms, similar in structure but far simpler than the approach in [14], or [15].

Adaptations for these algorithms are provided in the case of limited support estimation, and the trade-off between computation and accuracy is discussed. Square root techniques are discussed in the context of recursive estimation yielding algorithms which yield better numerical accuracy.

In Chapter 2, a brief comparison is made highlighting differences between Maximum Likelihood Estimation and Bayesian Estimation. The machinery needed to perform estimation when the noise which corrupts the observations have ill-defined covariances and when the parameters are not fully estimable is established, in both a standard ML context and a square root context.

In Chapter 3, separable two point boundary value descriptor systems are discussed. The condition of separability is examined. STPBVDS's are diagonalized, and their Markovianity is established.

In Chapter 4, separable two point boundary value descriptor systems are discussed. Analytical machinery is developed which allows the construction of general recursive ML algorithms which are presented analogous to the Mayne-Fraser and Rauch-Tung-Striebel algorithms in addition to STPBVDS square root algorithms which are also presented and are analogous to standard algorithms.

In Chapter 5, parallel algorithms are presented which are designed to operate on a linear array of processors. Past work is discussed and three new algorithms are presented. These algorithms exploit Markovianity and the ML philosophy to generate algorithms which are conceptually simpler than other parallel processing algorithms. Furthermore, a tree topology is possible, which allows extension to processes of higher dimension.

In Chapter 6 the algorithms of chapter 5 are examined in the context of limited support estimation. Trade-offs between complexity and accuracy are presented. Optimal numbers of processors are discussed to obtain specified accuracy goals.

In Chapter 7 the multi-dimensional estimation problem is discussed and parallel algorithms are developed. Examples are presented and suboptimal methods are discussed.

Chapter 2

Estimation

2.1 Maximum Likelihood Estimation

As mentioned in the introduction the development of recursive algorithms in this thesis involves successive ML estimations to recursively compute estimates of the state based on some set of data. The formulation of the Kalman Filter follows a Bayesian philosophy. Here we choose an ML estimation philosophy because it allows us to deal with a more general class of problems. As developed in [23] Bayesian least squares estimation can be converted to ML problems by viewing prior statistics and noisy dynamics as additional measurements. In the linear estimation problem the likelihood function $\mathcal{P}_{y|x}(y_o|x_o)$ is derived from the observation

$$y = Hx + v \tag{2.1}$$

$$\mathcal{P}_{y|x}(y_o|x_o) = \mathcal{P}_v(y_o - Hx_o) \tag{2.2}$$

$$\mathcal{P}_v = N(0, R) \tag{2.3}$$

$$y \in \Re^p \tag{2.4}$$

$$x \in \Re^n \tag{2.5}$$

The maximization of the likelihood function is well-defined when R has full rank and H has full column rank. The solution obtained is equivalent to minimizing

$v^T R^{-1} v$ subject to the constraint that $y = Hx + v$. The noises are therefore being estimated to be the smallest disturbance consistent with the observation equation and the statistics. With the solution $\hat{v}$, $\hat{x}$ can be solved from

$$Hx = y - \hat{v} \tag{2.6}$$

If H does not have full column rank then $\hat{x}$ cannot be determined uniquely. The projection of x in the nullspace of H is unconstrained. In a minimum norm maximum likelihood estimation problem, we choose this projection of x to be equal to zero. Regardless of how this projection of x is set, any $\hat{x}$ solving $H\hat{x} = y - \hat{v}$ is an ML estimate.

Two concerns remain. The first is the event that the observation noise has a singular covariance. There are several ways of dealing with this. One is to separate the noiseless part from the remainder of the observation.

Let the matrices S_p and S_s satisfy the following

$$S_p v \equiv 0 \tag{2.7}$$

$$S_s R S_s^T > 0 \tag{2.8}$$

$$\det \begin{bmatrix} S_p \\ S_s \end{bmatrix} \neq 0 \tag{2.9}$$

then, the problem can be reformulated as solving for x given the observations

$$S_p H x = S_p y \tag{2.10}$$

$$S_s H x = S_s y + S_s v \tag{2.11}$$

which is equivalent to minimizing $v^T S_s^T (S_s R S_s^T)^{-1} S_s v$ subject to the constraints (2.10), and (2.11). This is done explicitly in the square root algorithm in Section 2.2.

An equivalent way to deal with the singularity of the observation noise is to model

the noise as being of reduced dimension with identity covariance

$$y = Hx + Lv \tag{2.12}$$

and the solution can be obtained by minimizing $v^T v$ subject to the constraint $y - Hx - Lv = 0$. This constrained optimization problem can be computed using the method of Lagrange multipliers where

$$\hat{x} = \arg\min v^T v + \lambda^T (y - Hx - Lv) \tag{2.13}$$

where $LL^T = R$ and λ is the Lagrange multiplier. The solution to this is given by solving

$$\begin{bmatrix} LL^T & H \\ H^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ x \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{2.14}$$

Both methods yield equivalent results.

Moving further we consider the case where the observation has the form

$$y = Hx + Lv \tag{2.15}$$

yet v has only an information matrix associated with it $E[vv^T]^{-1} = S$. Note that if S is singular, part of v is therefore completely unknown while a part of Lv is known perfectly if L has a left nullspace. The ML problem is equivalent to minimizing $v^T S v + \lambda^T (y - Hx - Lv)$ which is equivalent to solving

$$\begin{bmatrix} -S & L^T & 0 \\ L & 0 & H \\ 0 & H^T & 0 \end{bmatrix} \begin{bmatrix} v \\ \lambda \\ x \end{bmatrix} = \begin{bmatrix} 0 \\ y \\ 0 \end{bmatrix} \tag{2.16}$$

By placing S in a diagonalized form it can be shown to be equivalent to (2.14). Let

$$S = \begin{bmatrix} 0 & 0 \\ 0 & \tilde{S} \end{bmatrix} \tag{2.17}$$

where $\tilde{S}$ is invertible, and

$$v = \begin{bmatrix} \overline{v} \\ \tilde{v} \end{bmatrix} \tag{2.18}$$

then solving for the ML estimate of x is equivalent to minimizing $\tilde{v}^T \tilde{S} \tilde{v} + \lambda^T (y - Hx - \overline{L}\overline{v} - \tilde{L}\tilde{v})$. Since $\overline{v}$ is completely unknown this can be looked at as an attempt to estimate $\overline{v}$ also. The estimate of x is obtained through solving

$$\begin{bmatrix} \tilde{L}\tilde{S}^{-1}\tilde{L}^T & H & \overline{L} \\ H^T & 0 & 0 \\ \overline{L}^T & 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ x \\ \overline{v} \end{bmatrix} = \begin{bmatrix} y \\ 0 \\ 0 \end{bmatrix} \tag{2.19}$$

Given observations of the form

$$y = Hx + \overline{L}\overline{v} + \tilde{L}\tilde{v} \tag{2.20}$$

where $\overline{v}$ is unknown we may choose to eliminate the effect of $\overline{v}$ on our computations since it provides no information about x. Premultiplying by $\overline{L}^{\perp}$ where $\overline{L}^{\perp}\overline{L} = 0$ and $\begin{bmatrix} \overline{L}^T \\ \overline{L}^{\perp} \end{bmatrix}$ has full rank yields

$$\overline{L}^{\perp} y = \overline{L}^{\perp} H x + \overline{L}^{\perp} \tilde{L} \tilde{v} \tag{2.21}$$

This returns our observations to the form of equation (2.1). We will see that in the recursive estimation context that $\overline{L}$ is presented in the form of a projection matrix which makes $\overline{L}^{\perp}$ particularly easy to compute.

Finally the last situation to consider is in the event the matrices in (2.14), (2.16), and (2.19) are not invertible. This will occur under two conditions. The first is the case where we have redundant perfect information, and the second is when the state is not completely estimable. The standard ML problem which we wish to solve has the form

$$\begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ x \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{2.22}$$

If we assume that the pseudo-inverse of the matrix in (2.22) has the form

$$\begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix}^{\dagger} = \begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \tag{2.23}$$

Then we wish only that γ be specified adequately to generate the appropriate ML estimate and if we wish to use the the form for the covariance provided by Nikoukhah [23] we require

$$\delta = -\gamma R \gamma^T \tag{2.24}$$

Nikoukhah [22], [23] considers the case where x is fully estimable, where H has full column rank, but R may be singular. In this case, the symmetric indefinite matrix in (2.19) is invertible except under the condition where there are redundant perfect measurements. Redundant perfect measurements may be deleted without affecting the estimate or its covariance. Nikoukhah shows that by allowing the pseudoinverse to satisfy $AXA = A$ that γ, though not unique due to the non-unique ways in which redundant perfect information may be used, will always provide the ML estimate of x, and δ which is given by (2.24) is unique, and yields the covariance of the estimate.

If the state x is not estimable, the matrix in (2.19) is not invertible and the condition $AXA = A$ is not sufficient to determine a parameter γ which will produce a unique estimate, nor will δ be interpretable as the error covariance of the estimate. Since the part of x in the nullspace of H is not estimable, any specification of this value is allowed. To specify a unique estimate, we set this part of the estimate to zero yielding a minimum norm ML solution. Given a γ which yields a minimum norm ML solution, we desire $\delta = -\gamma R \gamma^T$, allowing us to use the same form for the error covariance as used in Nikoukhah [23]. To this end we require the Moore-Penrose pseudo-inverse, which is uniquely given by the following four conditions.

$$\textit{Condition 1}: \quad AXA = A \tag{2.25}$$

$$\textit{Condition 2}: \quad XAX = X \tag{2.26}$$

$$\textit{Condition 3}: \quad AX = (AX)^T \tag{2.27}$$

$$\textit{Condition } 4: \quad XA = (XA)^T \tag{2.28}$$

We will indicate the Moore-Penrose pseudo-inverse with the # symbol. Pseudo-inverses which have only a subset of these conditions are referred to by the number corresponding to the properties which they possess. For our purposes, Condition 1, guarantees a solution to our ML problem, which is unique if x is estimable. Condition 4 guarantees that γ is adequately specified to yield the minimum norm ML estimate. To properly specify δ both Condition 2, and Condition 3 are required. Another consequence of using the Moore-Penrose pseudo-inverse is that in (2.23), $\beta = \gamma^T$.

If we consider further the following product

$$\begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix}^{\#} \begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix} = \begin{bmatrix} P_\lambda & 0 \\ 0 & P_x \end{bmatrix} \tag{2.29}$$

we find that the use of the Moore-Penrose pseudo-inverse results in symmetric projection matrices. The projection matrix P_x projects onto the estimable subspace for the vector x, while $I - P_x$ which we will denote by $\overline{P}_x$ projects onto the nullspace of H. What results is that the ML estimate which is given by

$$\hat{x}_{ML}[y] = \left\{ \begin{bmatrix} 0 \\ I \end{bmatrix}^T \begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix}^{\#} \right\} \begin{bmatrix} y \\ 0 \end{bmatrix} = \gamma y \tag{2.30}$$

satisfies

$$\hat{x}_{ML} = P_x x + \tilde{x}_{ML} \tag{2.31}$$

where P_x is given by

$$P_x = \left\{ \begin{bmatrix} 0 \\ I \end{bmatrix}^T \begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix}^{\#} \right\} \begin{bmatrix} H \\ 0 \end{bmatrix} = \gamma H \tag{2.32}$$

and the error covariance for $P_x x$ given by

$$\Sigma_x = -\left\{ \begin{bmatrix} 0 \\ I \end{bmatrix}^T \begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix}^{\#} \right\} \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{2.33}$$

satisfies

$$\Sigma_x = P_x \Sigma_x P_x = Cov(\hat{x}_{ML} - P_x x) \tag{2.34}$$

The ML estimate obtained by using the Moore-Penrose pseudo-inverse is the minimum norm ML solution [18]. Before continuing further we will consider an example to show explicitly the computations involved with general ML estimation. Suppose observations of a random vector x are given by

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{2.35}$$

where the following hold

$$E[v_1 v_1^T] = 0 \tag{2.36}$$

$$E[v_2 v_2^T] > 0 \tag{2.37}$$

and v_3 is a completely unknown parameter. In constructing the ML estimate of x based on the measurement y there are a few observations to make. The first consideration is that the observation noise v_3 is completely unknown. Since there is no a priori information about v_3, all observations which are corrupted by v_3 cannot be used, because there is no way to infer anything about x if there is no information about the observation noise. Premultiplying equation (2.35) by $\overline{L}^{\perp}$ which is given by

$$\overline{L}^{\perp} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.38}$$

as in (2.21) yields

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{2.39}$$

The solution is given by

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{\#} \begin{bmatrix} y_1 \\ y_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.40}$$

which after the evaluation of the pseudo-inverse yields

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \\ \hat{x}_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.41}$$

The estimate as in (2.31) is given by

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 - y_1 \\ 0 \\ 0 \end{bmatrix} \tag{2.42}$$

where the square matrix in (2.42) is the projection matrix P_x. The covariance is given

by

$$\Sigma_x = Cov\left[\left[\begin{array}{c} y_1 \\ y_2 - y_1 \\ 0 \\ 0 \end{array}\right] - \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right]\left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \end{array}\right]\right] = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \tag{2.43}$$

Note that $P_x \Sigma_x P_x = \Sigma_x$.

The estimation machinery developed so far is adequate for the measurement update step of the filters discussed in Chapter 4. However the prediction step forces us to consider the following linear transformation

$$z = Ax \tag{2.44}$$

where x is not completely estimable. Equation (2.44) can be rewritten in the following form

$$z = AP_x x + A\overline{P}_x x \tag{2.45}$$

where $P_x x$ is the part of x which is estimable and $\overline{P}_x x$ is the part of x which is not. The minimum norm ML solution $\hat{z}_{ML}$ cannot be a function of $A\overline{P}_x x$. We therefore seek the largest rank symmetric projection matrix P_z whose nullspace contains $A\overline{P}_x$. As a result, $P_z A = P_z A P_x$. Such a matrix is determined by

$$\overline{P}_z = (A\overline{P}_x A^T)^\dagger (A\overline{P}_x A^T) \tag{2.46}$$

where a (1,4) pseudo inverse is indicated. As a result,

$$P_z z = P_z A P_x x \tag{2.47}$$

and

$$\hat{z} = P_z A \hat{x} \tag{2.48}$$

Equivalently the part of equation (2.44) which provides no information is elimi-

nated. Specifically the equation

$$\overline{P}_z z = \overline{P}_z A x \tag{2.49}$$

is not used in the estimation of z. The error covariance is given by

$$\Sigma_z = P_z A \Sigma_x A^T P_z \tag{2.50}$$

As an example, suppose that

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & -1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \tag{2.51}$$

where the x_i are those in equation (2.35). From (2.46) the projection matrix P_z is given by

$$\begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.52}$$

It is a simple task to verify that

$$\begin{bmatrix} \hat{z}_1 \\ \hat{z}_2 \\ \hat{z}_3 \end{bmatrix} = \begin{bmatrix} y_2 \\ y_2 \\ y_1 \end{bmatrix} \tag{2.53}$$

while the error covariance computed via (2.50) is given by

$$\Sigma_z = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{2.54}$$

Before continuing with an examination of square-root estimation techniques, con-

sider the computation involved in the ML solution. Let the system we wish to solve be $AX = B$ where X, and B are $n \times g$ matrices because we need to recover the error covariance, and in some cases the associated projection matrix. If A is invertible then X can be solved in $\mathcal{I}(n, g)$ floating point operations (flops) using gaussian elimination where $\mathcal{I}(n, g)$ is given by

$$\mathcal{I}(n, g) \equiv 2n^3/3 + 2n^2 g \tag{2.55}$$

The estimates for specific computations were obtained from Golub and Van Loan[31]. The function $\mathcal{I}$ is one of a series of polynomial function we will use to define the number of flops required to perform different estimation procedures.

If the inverse of the matrix does not exist, then a (1,4) pseudo-inverse for the matrix in (2.22) which computes the minimum norm solution can be easily computed with the aid of the QR factorization. Specifically, the QR factorization with pivoting can be used to generate a lower triangular matrix L given by

$$L = \Pi A U^T \tag{2.56}$$

where Π is a permutation matrix, and U is an orthogonal projection matrix. The system $AX = B$ can be rewritten as

$$L\{UX\} = \Pi B \tag{2.57}$$

or

$$\begin{bmatrix} L_{11} & 0 \\ L_{21} & 0 \end{bmatrix} \begin{bmatrix} U_1 X \\ U_2 X \end{bmatrix} = \begin{bmatrix} \Pi_1 B \\ \Pi_2 B \end{bmatrix} \tag{2.58}$$

The pseudo-inverse of L is easily computed, and is given by

$$\begin{bmatrix} L_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \tag{2.59}$$

The solution to the problem is therefore given by

$$U_1 X = L_{11}^{-1} \Pi_1 B \tag{2.60}$$

$$U_2 X = 0 \tag{2.61}$$

If the rank of A is equal to r then the the number of computations required to compute the QR factorization is given by $4[n^2 r - r^2 n + r^3/3]$ flops. Solving (2.60) for UX can be performed in gr^2 flops. To recover X from UX requires an additional $2gr(2n - r)$ flops. The total flop count is given by

$$\mathcal{E}(n, g, r) \equiv 4(n^2 r - r^2 n + r^3/3 + grn) - gr^2 \tag{2.62}$$

To compute the Moore-Penrose pseudo-inverse, a complete orthogonal decomposition [31] is required. A complete orthogonal factorization results in the factorization $T = QAU^T$ where Q and U are orthogonal matrices, and T has the following form

$$T = \begin{bmatrix} T_{11} & 0 \\ 0 & 0 \end{bmatrix} \tag{2.63}$$

where T_{11} is invertible. In addition if QR factorizations are used to compute this complete orthogonal factorization, T_{11} is triangular. The system $AX = B$ can be rewritten as

$$T\{UX\} = QB \tag{2.64}$$

or

$$\begin{bmatrix} T_{11} & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} U_1 X \\ U_2 X \end{bmatrix} = \begin{bmatrix} Q_1 B \\ Q_2 B \end{bmatrix} \tag{2.65}$$

The pseudo-inverse of T is easily computed, and is given by

$$\begin{bmatrix} T_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \tag{2.66}$$

The solution to the problem is therefore given by

$$U_1 X = T_{11}^{-1} Q_1 B \tag{2.67}$$

$$U_2 X = 0 \tag{2.68}$$

The factorization requires $8[n^2 r - r^2 n + r^3/3]$ flops. Solving (2.65) for UX requires gr^2 flops, and recovering X from UX and the construction of QB each requires $2gr(2n-r)$ flops. The total flop count is given by

$$\mathcal{M}(n,g,r) \equiv 8(n^2 r - r^2 n + r^3/3) + gr^2 + 4gr(2n - r) \tag{2.69}$$

As a result when these counts are applied to the ML problem in (2.22), with the original definition of n as the dimension of x the invertible case yields $\mathcal{I}(n+p, n+1)$ flops where we have substituted n+1 for g because the covariance and the estimate can be obtained from

$$\begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix} \begin{bmatrix} \lambda & -\beta \\ x & \Sigma_x \end{bmatrix} = \begin{bmatrix} y & 0 \\ 0 & -I \end{bmatrix} \tag{2.70}$$

Furthermore, we may divide the computation into 'on-line' and 'off-line' computation. Off-line computations involve computing the covariance and projection matrices, and on-line computations involve computing the estimate. When the state is estimable, the off-line computation is given by

$$\begin{bmatrix} \gamma^T \\ -\Sigma_x \end{bmatrix}^T = \begin{bmatrix} 0 \\ I \end{bmatrix}^T \begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix}^{-1} \tag{2.71}$$

when this inverse exists, and requires $\mathcal{I}(n+p, n)$ flops. The on-line computation is given by

$$\hat{x}_{ML}[y] = \gamma y \tag{2.72}$$

and requires $2pn$ flops.

If the matrix is not invertible but the state is still estimable then solving (2.70)

involves $\mathcal{E}(n+p, n+1, r)$ flops where r is the rank of the matrix. The off-line computation has the form

$$\begin{bmatrix} \gamma^T \\ -\Sigma_x \end{bmatrix}^T = \begin{bmatrix} 0 \\ I \end{bmatrix}^T \begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix}^{\#} \tag{2.73}$$

and requires $\mathcal{E}(n+p, n, r)$ flops and the on-line computation given again by (2.72) requires $2pn$ flops.

If the state is not estimable then the appropriate projection matrix must also be computed. It can be computed from the following

$$\begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix} \begin{bmatrix} \lambda & 0 & \beta \\ x & P_x & \Sigma_x \end{bmatrix} = \begin{bmatrix} y & H & 0 \\ 0 & 0 & -I \end{bmatrix} \tag{2.74}$$

The ML solution therefore requires $\mathcal{M}(n+p, n+1, r)$ flops. The off-line computation is given by (2.73) and

$$P_x = \gamma H \tag{2.75}$$

which together total to $\mathcal{M}(n+p, n, r) + 2n^2p$ flops. The on-line computation is $2pn$. Often it is difficult to know in advance what the rank of the matrix will be. As a result substituting $r = n+p$ is a useful overestimate of the work required to perform the necessary computations. As a result we will assume that (2.70) requires $\mathcal{E}(n+p, n+1, n+p)$ flops and (2.74) requires $\mathcal{M}(n+p, n+1, n+p)$ flops.

In solving the estimation problem associated with equation (2.44), the computations are quite explicit. Let $A \in \Re^{m \times n}$ where m is the dimension of z and n is the dimension of x. Equation (2.46) requires two matrix-matrix multiplies and a (1,4) pseudo-inverse computation. Equation (2.50) require three matrix multiplies where the quantity $P_z A$ will be saved to compute equation (2.48) which requires only a matrix-vector multiply.

$\mathcal{U}(n, r) \equiv 6mn^2 + 4m^2n + \mathcal{E}(m, m, r)$ flops are needed to compute the projection and covariance. The on-line computation is given by $2mn$ flops. If z is estimable then equation (2.48) requires one matrix vector multiply and (2.50) requires 2

matrix matrix multiplies yielding $\mathcal{V}(n) \equiv 2mn^2 + 2mn^2$ flops for off-line computation and $2mn$ on-line.

2.2 Square-Root ML Estimation

In Section 2.1, several issues were handled automatically for us. The case where perfect measurement information is present did not have to be addressed as a special case except when there are perfect redundant measurements. In that case the matrix in (2.22) is not invertible. Since we want to be able to include the case where the state may not in fact be estimable, the lack of invertibility due to redundant perfect measurements is no longer a serious consideration because the lack of invertibility due to non-estimability requires us to use a much stronger pseudoinverse.

There is extensive literature on the square root information filter. Here we consider its adaptation to both the case of non-estimable states and perfect measurements, which imply ill-defined covariances and information matrices. The square-root ML estimation algorithm introduced in this section makes heavy use of the QR factorization to partition the state into three parts, the part of the state which can be estimated perfectly, the part of the state which has we an invertible covariance, and the part of the state which cannot be estimated. It is based on the algorithm provided by Bierman[6] and it requires a total of two QR factorizations for our general ML estimation problem. Another square-root ML algorithm is provided in [33] and it offers the advantage that the case of singular covariances is easily handled; however the algorithm as presented in [33] requires two QR factorizations for the case where the the state is estimable, and an additional QR factorization would have to be included to account for the case where the state is not completely estimable.

Instead of the implicit way in which Section 2.1 handles perfect information, we choose a 'reduced order' estimation algorithm[32] in which we deal explicitly with perfect measurements separately and explicitly eliminate the non-estimable portion of the state which is in the nullspace of the observation matrix H. To illustrate the ideas behind this approach, let us return to the simple example in Section 2.1. In equation (2.39) note explicitly that estimation of x_3 and x_4 is not possible since the measurement y_3 was removed, and the remaining measurements are not influenced

by x_3, or x_4. The problem can be reduced again to the form

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{2.76}$$

The problem has now been reduced to the case where the observation noise covariance is well-defined, which implies that all of the observations y are useful, and to the case where the observation matrix, H has only the origin for its null space implying that the states to be determined are indeed estimable. Since $v_1 = 0$, clearly $\hat{x}_{1ml} = y_1$. Information from y_2 cannot improve upon this estimate, which was made in the absence of noise. The remainder of the problem is one of estimating x_2 given the observation $y_2 - x_1$. This is a well-defined ML problem yielding $\hat{x}_{2ml} = y_2 - \hat{x}_{1ml}$.

The part of the state which can be computed from perfect data has zero covariance. The part of the state which has not been estimated have an infinite error covariance, and the remaining part of the state has a positive definite error covariance. This example was particularly easy because the coordinate system made it easy to identify these three components of x. In a square-root implementation, the coordinate system of y and x are rotated to this preferred coordinate system via orthogonal transformations derived from the QR factorization to efficiently, and robustly compute the ML estimate.

Techniques for computing the QR factorization are given in [31]. Here we exploit the QR algorithm to describe ML estimation under a variety of conditions which allows to implement in Section 4.10 a causal Kalman filter for so-called separable systems. The same techniques will carry over to the Mayne-Fraser, and Rauch-Tung Striebel algorithms.

The following example treats the ML estimation problem as needed in preparation for the square root filtering algorithms in Chapter 4. A standard treatment of square root algorithms is provided by [6]. The first example to consider is the standard ML estimation problem which is to estimate a random vector where the observation noise has a well-defined, and full rank, covariance. In addition, the observation matrix, H in equation (2.77), has only the origin in its right null space. We will call this the

standard ML estimation problem. As in (2.1), let the observations be given by

$$y = Hx + v \tag{2.77}$$

where

$$E[vv^T] = R^{1/2}R^{T/2} \tag{2.78}$$

By normalizing the noise, the original problem can be written in the following manner.

$$R^{-1/2}y = R^{-1/2}Hx + R^{-1/2}v \tag{2.79}$$

where

$$E[R^{-1/2}vv^TR^{-T/2}] = I \tag{2.80}$$

Any orthogonal matrix Q which premultiplies the normalized noise $R^{-1/2}v$ will not change the variance of the observation noise in (2.79). By applying the QR algorithm we can find a matrix Q such that the product $QR^{-1/2}H$ is upper triangular. Unless the matrix is square, the lower portion of this matrix will be all zeroes.

$$QR^{-1/2}y = QR^{-1/2}Hx + QR^{-1/2}v \tag{2.81}$$

By making obvious substitutions, equation (2.81) can be written as

$$d = Ax + w \tag{2.82}$$

where again the matrix A is upper triangular. This can be rewritten as

$$\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} = \begin{bmatrix} A_1 \\ 0 \end{bmatrix} x + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \tag{2.83}$$

Since the noise is normalized, the two noises w_1 and w_2 are independent. Since the

matrix A_1 is of full rank, the ML estimate is thus obtained by solving

$$d_1 = A_1 \hat{x}_{ML} \tag{2.84}$$

Since A_1 is upper triangular and square, $\hat{x}_{ML}$ can be determined easily without recourse to further processing of the matrix A_1. In addition since the noise w_1 has the identity for its covariance A_1^{-1} can be identified as the inverse of the square root of the estimation error covariance for x. By premultiplying equation (2.84) by A_1^T it can be seen that the solution given by this method is

$$\hat{x}_{ML} = (H^T R^{-1} H)^{-1} H^T R^{-1} y \tag{2.85}$$

which is the standard ML solution.

A different approach to square-root ML estimation is presented in [33]. Let the observation in (2.77) be written as

$$y = Hx + Lv \tag{2.86}$$

where v has identity covariance so that $L = R^{\frac{1}{2}}$. Then by applying the QR factorization directly to H yields

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} H_1 \\ 0 \end{bmatrix} x + \begin{bmatrix} L_1 \\ L_2 \end{bmatrix} v \tag{2.87}$$

An orthogonal matrix Z can be computed to multiply v such that (2.87) has the form

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} H_1 \\ 0 \end{bmatrix} x + \begin{bmatrix} 0 & L_{11} & L_{12} \\ 0 & 0 & L_{22} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{2.88}$$

where

$$Zv = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{2.89}$$

Here we see that v_1 is orthogonal to the estimate of x, v_2 will contribute to the estimation error, and v_3 is directly observed. The estimate is given by

$$\hat{x} = H_1^{-1}(y_1 - L_{12}L_{22}^{-L}y_2) \tag{2.90}$$

Note that L could be singular in this formulation, and that the square-root of the error covariance is given by $H_1^{-1}L_{11}$. In this algorithm, normalizing the noise is avoided, but two QR factorizations are required. To adapt this algorithm to the issues which will be presented, in this section, three QR factorizations would be required.

The standard ML estimation algorithm, formulated in equations (2.79)- (2.83), is too restrictive for the estimation needs in this thesis. The first concern is that the covariance of v is not necessarily normalizable. This may happen if there are observations which are known perfectly. Furthermore in the problems we will encounter v is a parameter which corrupts our observation, and parts of v may be unknown as in equation (2.20). Furthermore, if x is not fully estimable then A_1 is not invertible, and pseudoinverses need to be considered again. The following algorithm accounts for all of the potential problems in the preceding discussion while maintaining the same essential properties of the algorithm described above.

Returning to the observation equation (2.77), for purposes of recursive estimation, we interpret v to be a vector which is partly unknown, and a part of it is identically zero. In Section 4.10 v will include the estimation error from the state estimated at an earlier time step. We will assume that the value of the ML estimate of v is zero. In contrast to the previous example, H is no longer restricted to have only the origin in its right nullspace . The availability of an orthogonal matrix $\mathcal{S}$ is assumed, which

can be partitioned as follows

$$S^T = \begin{bmatrix} S_p^T & S_s^T & S_u^T \end{bmatrix} \tag{2.91}$$

where $S_p v$ is identically zero, and the covariance of $S_s v$ is positive definite and is equal to R_0. The vector $S_u v$ is completely unknown. In the recursive estimation problem S is computed recursively and falls out directly from the estimation process. The portion of the observation vector which corresponds to the unknown vector $S_u v$ will not count as an observation since it provides no information about x, and it will not be used in the estimation procedure. This is equivalent to the step in (2.21) where $L_o^{\perp}$ is used to eliminate the 'observations' which contain no information. In the following treatment, all invertible matrices will be given a zero subscript.

If (2.77) is premultiplied by $L_o^{\perp} = [S_p^T \; S_s^T]^T$ yielding

$$\begin{bmatrix} S_p \\ S_s \end{bmatrix} y = \begin{bmatrix} S_p \\ S_s \end{bmatrix} Hx + \begin{bmatrix} 0 \\ S_s \end{bmatrix} v \tag{2.92}$$

then the QR factorization can be used to obtain the parts of x which can be determined perfectly, the part which can be determined with full rank covariance, and the part which is completely unspecified by (2.92). This is performed by finding the orthogonal transformation U which lower triangularizes the matrix $[S_p \; S_s]^T HU$ in

$$\begin{bmatrix} S_p \\ S_s \end{bmatrix} y = \begin{bmatrix} S_p H U_p^T & 0 & 0 \\ S_s H U_p^T & S_s H U_s^T & 0 \end{bmatrix} \begin{bmatrix} U_p x \\ U_s x \\ U_u x \end{bmatrix} + \begin{bmatrix} 0 \\ S_s \end{bmatrix} v \tag{2.93}$$

The matrix U has been partitioned as follows

$$U^T = \begin{bmatrix} U_p^T & U_s^T & U_u^T \end{bmatrix} \tag{2.94}$$

where $U_p x$ is known perfectly, and the error covariance of $U_s x$ is invertible and is equal to Σ_0. The vector $U_u x$ is completely unspecified. If there are no redundant

perfect measurements, U_p can be determined directly from (2.93) because $S_p H U_p^T$ is lower triangular. If there are redundant perfect measurements, then only a subset of $S_p y$ is needed to specify $U_p x$. More generally, any left inverse of $S_p H U_p^T$ may be used to compute $U_p x$. After $U_p x$ has been computed, $U_s x$ can then be determined by normalizing the noise in (2.93)

$$R_0^{-1/2} S_s (y - H U_p^T \{U_p \hat{x}_{ML}\}) = R_0^{-1/2} S_s (H U_s^T \{U_s x\} + v) \tag{2.95}$$

The random vector $R_0^{-1/2} S_s v$ has the identity for its covariance. The QR factorization can be used to solve this 'standard' problem by computing the orthogonal matrix J to upper triangularize the matrix $J R_0^{-1/2} S_s H U_s^T$ in

$$J R_0^{-1/2} S_s (y - H U_p^T \{U_p \hat{x}_{ML}\}) = J R_0^{-1/2} S_s (H U_s^T \{U_s x\} + v) \tag{2.96}$$

where

$$J R_0^{-1/2} S_s H U_s^T = \begin{bmatrix} \Sigma_{x,o}^{-1/2} \\ 0 \end{bmatrix} \tag{2.97}$$

contains the square root of the inverse error covariance of $U_s \hat{x}_{ML}$.

Earlier we stated that equations including S_u provided no information about x. As a result in equation (2.93) we did not include S_u in the premultiplication because in the equation

$$S_u y = \begin{bmatrix} S_u H U_p^T & S_u H U_s^T & S_u H U_u^T \end{bmatrix} \begin{bmatrix} U_p x \\ U_s x \\ U_u x \end{bmatrix} + S_u v \tag{2.98}$$

we find that the term $S_u v$ is completely unknown and occurs only in this equation. No algebraic constraints are placed upon x by this equation. This equation is only useful if we desired to estimate $S_u v$ in addition to x.

In summary we have used two QR factorizations, to recover the estimate of the

estimable part of x and its projection P_x from equation (2.93) which is given by

$$P_x = \begin{bmatrix} U_p^T & U_s^T \end{bmatrix} \begin{bmatrix} U_p \\ U_s \end{bmatrix} \tag{2.99}$$

In addition, the covariance of the estimable part of the estimate is given by

$$Cov(P_x x - \hat{x}_{ML}) = \Sigma_x^{\#} = U_s^T \Sigma_{x,o}^{-1} U_s \tag{2.100}$$

The amount of computation nedded to compute $\hat{x}_{ML}$ is a function which depends on the rank of S_s, and the rank of U_s because these ranks determine the amount of work involved in performing the QR factorization which determines J. The QR factorization which constructs U involves at most $2n^2(p - n/3)$ flops, and the factorization which constructs J requires at most $2n^2(p-n/3)$ flops. The normalization of the noise requires at most p^2n flops and solving for x requires n^2 flops. The premultiplication by S required at most $2p^2n$ flops. The total flop count is $4pn^2 + 3p^2n - 4n^3/3$

Again for the purposes of considering the update step of the filter we wish to examine (2.44) which is given again as follows:

$$z = Ax \tag{2.101}$$

where $A \in R^{n \times m}$. Given the coordinate transformation used to estimate x given by U partitioned similarly to $\mathcal{S}$ in (2.91), and the covariance of $U_s x$ given by $\Sigma_{x,o}$, the QR factorization is used to find the orthogonal matrix Q to upper triangularize the product

$$G = Q \begin{bmatrix} AU_u^T & AU_s^T & AU_p^T \end{bmatrix} \tag{2.102}$$

As a result Q can be partitioned as

$$Q = \begin{bmatrix} Q_u^T & Q_s^T & Q_p^T \end{bmatrix}^T \tag{2.103}$$

It then follows that

$$Q_p z = Q_p A U_p^T \{U_p x\} \tag{2.104}$$

$$Q_s z = Q_s A U_p^T \{U_p x\} + Q_s A U_s^T \{U_s x\} \tag{2.105}$$

$$Q_u z = Q_u A U_p^T \{U_p x\} + Q_u A U_s^T \{U_s x\} + Q_u A U_u^T \{U_u x\} \tag{2.106}$$

The vector $Q_p z$ can be determined perfectly, and the vector $Q_s z$ can be estimated with a full rank covariance. The vector $Q_s z$ cannot be estimated at all since $Q_u A U_u^T$ has full column rank and $U_u x$ is completely unknown. Since we are computing square roots of covariances and inverse square roots a further computation is necessary to compute the square root of the covariance of $Q_s z$. By noting that $Q_s z$ is given by

$$Q_s z = Q_s A U_p^T \{U_p x\} + Q_s A U_s^T \Sigma_{x,o}^{1/2} \{\Sigma_{x,o}^{-1/2} U_s \hat{x}\} + Q_s A U_s^T \Sigma_{x,o}^{1/2} \{\Sigma_{x,o}^{-1/2} U_s \tilde{x}\} \tag{2.107}$$

The error covariance of the estimate is computed via the QR factorization to find the orthogonal matrix T which lower triangularizes the matrix

$$\begin{bmatrix} \Sigma_{z,o}^{1/2} & 0 \end{bmatrix} = Q_s A U_s^T \Sigma_{x,o}^{1/2} T \tag{2.108}$$

In summary, we have obtained the perfectly known parts of the estimate, $Q_p \hat{z}_{ML}$. In addition the remainder of the estimable portion of the estimate is given by $Q_s \hat{z}_{ML}$, and the square root of the inverse of its covariance by $\Sigma_{z,o}^{1/2}$ in (2.108). The matrix $[Q_p^T \; Q_s^T]$ representing parts of coordinate rotations used to perform estimation, is the square root of P_z, specifically

$$P_z = \begin{bmatrix} Q_p^T & Q_s^T \end{bmatrix} \begin{bmatrix} Q_p \\ Q_s \end{bmatrix} \tag{2.109}$$

The remainder of the vector is completely non-estimable. Finally, the minimum norm, minimum variance, estimate of Q_u is zero. Note that since we are performing norm preserving, i.e. orthogonal transformations, setting $Q_u \hat{z}$ to zero is the correct

operation to perform as the resulting estimate for z in its original coordinate system will indeed have minimum norm. The product in (2.102) requires $2nm^2$ flops, and the computation of Q and $Q\{AU\}$ requires at most $4n^3/3 + 2(m-n)n^2$ flops. To obtain $Q\hat{z}$ requires at most nm and the computation of Σ_x requires at most $4mn^2 - 2n^2(m+n) + 4n^3/3 + nm^2$ yielding a total of $-4n^3/3 + 4mn^2 + 3nm^2$ flops.

The matrices J and T which are instrumental in determining the covariances, along with U and Q show that there are two QR decompositions associated with each estimation step. In the presence of perfect information and states that are not completely estimable the QR techniques will be of reduced dimension. In Section 4.10 the QR techniques will be incorporated into the forward maximum likelihood filter (FMLF).

Chapter 3

Two Point Boundary Value Descriptor Systems

In order to perform smoothing on multidimensional systems, a model is needed which is more general than the standard causal systems. The systems on which we focus on in this thesis are called Two Point Boundary Value Descriptor Systems (TPBVDS's) and have the following form.

$$E_{k+1}x(k+1) = A_k x(k) + B_k u(k) \qquad K_0 \leq k \leq K_1 - 1 \tag{3.1}$$

$$E_{K_0}x(K_0) = A_{K_1}x(K_1) + B_{K_1}u(K_1) \tag{3.2}$$

$$y(k) = C_k x(k) + r(k) \qquad K_0 < k < K_1 \tag{3.3}$$

where

$$x(k) \in R^{n_k} \tag{3.4}$$

$$B_k \in R^{q_k \times m_k} \tag{3.5}$$

$$u(k) \sim N(0, I) \tag{3.6}$$

$$r(k) \sim N(0, R_k) \tag{3.7}$$

where E_k, A_k, and C_k are compatibly defined but not necessarily full rank matrices and the noises $u(k)$, and $r(k)$ are independent sequences. In this chapter it is assumed

that our TPBVDS is well-posed. Allowing E_k and A_k to be rank deficient implies that $x(k)$ does not have a recursive definition. These systems are interesting for several reasons. They arise naturally in the case of discretization of partial differential equations (pde's) in two dimensions and boundary value ordinary differential equations in one dimension.

General TPBVDS's create problems for estimation because the boundary condition links the state at each end of the interval. For example, as developed in [22] the Hamiltonian is a TPBVDS which relates the smoothed estimate of the state $x(k)$ with the smoothed estimates at neighboring points with the help of auxiliary variables $\lambda(k)$ which are appended to the state which represent the complementary process[36],[3], or which represents Lagrange multipliers used in solving the optimization problem required to solve for the optimal estimates. The dynamics of the Hamiltonian associated with a STPBVDS can be decoupled into two filters which propagate in opposite directions. This decoupling is called diagonalization. For general TPBVDS's however, the associated Hamiltonian cannot be 'diagonalized' into two independent filters, as the boundary conditions of the two filters cannot be decoupled. Shooting methods [20] are then required to solve the equations of the coupled filters. The result is that the filters which are involved in the shooting method of solving the Hamiltonian cannot be directly interpreted as providing estimates of the state based on any particular set of data.[22],[24]

3.1 Separable Two Point Boundary Value Descriptor Systems

The class of Separable Two Point Boundary Value Descriptor Systems (STPBVDS's) are equivalent to the full set of TPBVDS's yet avoids the aforementioned problems. STPBVDS's have a form given by

$$E_{k+1}x(k+1) = A_k x(k) + B_k u(k) \qquad K_0 \leq k \leq K_1 - 1 \tag{3.8}$$

$$E_{K_0}x(K_0) = B_{K_0-1}u(K_0 - 1) \tag{3.9}$$

$$0 = A_{K_1} x(K_1) + B_{K_1} u(K_1) \tag{3.10}$$

$$y(k) = C_k x(k) + r(k) \qquad K_0 < k < K_1 \tag{3.11}$$

Here the boundary conditions are independent both algebraically, and statistically. STPBVDS's are of interest for a variety of reasons. One is that, as we will see, any TPBVDS can be expressed as a STPBVDS. Also, the Hamiltonian associated with the estimation problem is an STPBVDS. When this Hamiltonian is diagonalized, the resulting forward and backward propagating filters have independent boundary conditions, and shooting is no longer required. Furthermore the separable form allows the solution to the smoothing problem to be easily formulated in an ML framework without the benefit of the Hamiltonian. Another reason which will also be demonstrated is that the filters associated with the smoothers of STPBVDS produce estimates of the state. Finally all STPBVDS are Markov.

If (3.9), and (3.10) are combined into one equation one concludes that STPBVDS's are a subclass of TPBVDS's. Continuing, we will show that STPBVDS's are TPBVDS's whose boundary condition can be split into two independent boundary conditions. One will be specified at $x(K_0)$, while the other will be specified at $x(K_1)$. In the following we assume that $-K_0 = K_1 = K$. Given the TPBVDS in equation (3.1), (3.2), (3.3), and (3.8), the system is separable [1], [22] if

$$E_{-K}^T \{B_K B_K^T\}^{-1} A_K = 0 \tag{3.12}$$

To see what is implied by the condition of separability, consider the equations

$$E_{-K} x(-K) = A_K x(K) + B_K u(K) \tag{3.13}$$

$$u(k) \sim N(0, I) \tag{3.14}$$

and (3.12). Premultiplication of equation (3.13) by $E_{-K}^T \{B_K B_K^T\}^{-1}$, and $A_K^T \{B_K B_K^T\}^{-1}$ yields

$$E_{-K}^T \{B_K B_K^T\}^{-1} E_{-K} x(-K) = E_{-K}^T \{B_K B_K^T\}^{-1} B_K u(K) \tag{3.15}$$

$$A_K^T\{B_K B_K^T\}^{-1} A_K x(K) = -A_K^T\{B_K B_K^T\}^{-1} B_K u(K) \tag{3.16}$$

where, thanks to (3.12) the noises on the right hand sides of equations (3.15) and (3.16) are independent. If we define a projection matrix P_e given by

$$P_e = E_{-K}(E_{-K}^T\{B_K B_K^T\}^{-1} E_{-K})^\dagger E_{-K}^T\{B_K B_K^T\}^{-1} \tag{3.17}$$

where a generalized inverse satisfying $A^\dagger A A^\dagger = A^\dagger$ is indicated [18], then equations (3.15), and (3.16) may be given by

$$E_{-K}x(-K) = P_e B_K u(K) \tag{3.18}$$

$$-A_K x(K) = (I - P_e)B_K u(K) \tag{3.19}$$

where the additional equalities

$$P_e\{B_K B_K\}^{-1}(I - P_e)^T = 0 \tag{3.20}$$

$$P_e E_{-K} = E_{-K} \tag{3.21}$$

$$P_e A_K = 0 \tag{3.22}$$

$$(I - P_e)E_{-K} = 0 \tag{3.23}$$

$$(I - P_e)A_K = A_K \tag{3.24}$$

are satisfied. The existence of any matrix P_e satisfying (3.18) through (3.24), allows separability to be defined in terms of $B_K B_K^T$ instead of $\{B_K B_K^T\}^{-1}$, and relieves us of the restriction that $B_K B_K^T$ be invertible. Furthermore we can define $B_{-1}u(-1)$ by $B_{-1}u(-1) = P_e B_K u(K)$ and the TPBVDS can be identified with the STPBVDS model in equations (3.8), (3.9) and (3.10). Separability is the ability to describe the boundary condition as separate independent boundary conditions, one at $k = 0$, and at $k = K$. In addition, the measurements of the boundary must also be independent observations of the initial and final state.

Earlier it was stated that TPBVDS can be represented as STPBVDS. This fact was demonstrated by Adams [1] for the case where $E_k = I$. Separability was con-

nected to the generation of a causally equivalent model to the acausal TPBVDS. In this section, we will generate a STPBVDS from a TPBVDS without constructing a causal equivalent. Before doing so, we note that Nikoukhah showed that TPBVDS's can be represented by STPBVDS's of double the dimension but over half the time interval[22]. Specifically, if we set $-K_0 = K_1 = K$ in (3.8), then Nikoukhah's separable representation for the TPBVDS given by (3.1) is written as

$$\begin{bmatrix} E_{k+1} & 0 \\ 0 & A_{-k-1} \end{bmatrix} \begin{bmatrix} x(k+1) \\ x(-k-1) \end{bmatrix} = \begin{bmatrix} A_k & 0 \\ 0 & E_{-k} \end{bmatrix} \begin{bmatrix} x(k) \\ x(-k) \end{bmatrix} + \begin{bmatrix} B_k & 0 \\ 0 & -B_{-k-1} \end{bmatrix} \begin{bmatrix} u(k) \\ u(-k-1) \end{bmatrix} \quad 0 \le k \le K_1 \tag{3.25}$$

with the boundary condition (3.2) included in

$$\begin{bmatrix} I & -I \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(+0) \\ x(-0) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ E_K & A_{-K} \end{bmatrix} \begin{bmatrix} x(K) \\ x(-K) \end{bmatrix} = \begin{bmatrix} 0 \\ B_K u(K) \end{bmatrix} \tag{3.26}$$

This boundary condition is separable and algorithms which can operate on separable systems can in general be applied to non-separable systems by using this representation.

In the event it is undesirable or inconvenient to augment $x(k)$ with $x(-k)$, it is still possible to construct an STPBVDS by essentially augmenting $x(k)$ with $x(K_0)$ or $x(K_1)$. Given any well-posed system of the form (3.1), we can refer to the projection matrix (3.17) to facilitate the construction of a separable system. Let $K_0 = 0$, and $K_1 = K$. The boundary condition has been decoupled into two equations

$$E_{-K}x(-K) + P_e A_K x(K) = P_e B_K u(K) \tag{3.27}$$

$$(I - P_e)A_K x(K) = (I - P_e)B_K u(K) \tag{3.28}$$

where P_e is given by equation (3.17) and

$$P_e P_e = P_e \tag{3.29}$$

The system can be made separable by augmenting $x(k)$ with the constant vector $\xi(k) = x(K)$, and imposing the following boundary condition.

$$\begin{bmatrix} E_{-K} & P_e A_K \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(-K) \\ \xi(-K) \end{bmatrix} = \begin{bmatrix} P_e B_K u(K) \\ 0 \end{bmatrix} \tag{3.30}$$

$$\begin{bmatrix} (I - P_e)A_K & 0 \\ -I & I \end{bmatrix} \begin{bmatrix} x(K) \\ \xi(K) \end{bmatrix} = \begin{bmatrix} (I - P_e)B_K u(K) \\ 0 \end{bmatrix} \tag{3.31}$$

It turns out however that all of $x(K)$ need not be appended to $x(t)$. A 'reduced order' separable system can be constructed by noting that often the rank of the matrix $P_e A_K$ in equation (3.27) will have some value r which is less than n, the dimension of $x(K)$. When the system is separable, the value of r is equal to zero. Regardless of the rank of $P_e A_K$, the boundary conditions can be decomposed in the following manner.

$$P_e A_K = \theta_L \theta_R \tag{3.32}$$

where θ_L is an $n \times r$ matrix and θ_R is an $r \times n$ matrix. The boundary condition can thus be modified in the following manner.

$$\begin{bmatrix} E_{-K} & \theta_L \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(-K) \\ \zeta(-K) \end{bmatrix} = \begin{bmatrix} P_e B_K u(K) \\ 0 \end{bmatrix} \tag{3.33}$$

$$\begin{bmatrix} (I - P_e)A_K & 0 \\ -\theta_R & I \end{bmatrix} \begin{bmatrix} x(K) \\ \zeta(K) \end{bmatrix} = \begin{bmatrix} (I - P_e)B_K u(K) \\ 0 \end{bmatrix} \tag{3.34}$$

The system which accompanies this boundary condition is given by

$$\begin{bmatrix} E_{k+1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k+1) \\ \zeta(k+1) \end{bmatrix} = \begin{bmatrix} A_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ \zeta(k) \end{bmatrix} + \begin{bmatrix} B_k & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u(k) \\ 0 \end{bmatrix}$$

(3.35)

A similar separable decomposition was done by Adams in the context of developing Markovian models for noncausal systems. This differs in that a separable model was obtained without significantly altering the representation of the system. In addition $x(0)$ was appended in Adams derivation while $x(K)$ is appended to $x(k)$ in this derivation.

Note in either case, that the formation of a separable system involves the generation of a boundary condition with singular noise. Returning to the model given by (3.8) if A_K does not have full column rank then it is not possible to estimate $x(K)$ based only on boundary information. In the case where the state of the STPBVDS is given by $[x(k), x(-k)]$, then the state at $k = 0$, is obviously degenerate in that the two components are known to be equal. As a result a well-defined covariance cannot be specified of the state based on boundary information. Filtering therefore presents a problem because of the problem of initializing a Kalman filter with the proper covariance information.

Moving forward we consider two more properties of STPBVDS, namely the diagonalizability property and the Markov property.

3.2 Diagonalizability of STPBVDS's

Consider a well posed STPBVDS All of the equations can be written together in the form

$$\mathcal{A}X_K = \mathcal{B}_K U_K \tag{3.36}$$

$$Y = \mathcal{C}_K X_K + V_K \tag{3.37}$$

where the vectors X_K, U_K, Y_K, and V_K are given by

$$X_K^T = [x^T(0), x^T(1), \cdots, x^T(K-1), x^T(K)]^T \tag{3.38}$$

$$U_K^T = [u^T(-1), u^T(0), u^T(1), \cdots, u^T(K-1), u^T(K)]^T \tag{3.39}$$

$$Y_K^T = [y^T(0), y^T(1), \cdots, y^T(K-1), y_K^T]^T \tag{3.40}$$

$$V_K^T = [v^T(0), v^T(1), \cdots, v^T(K-1), v^T(K)]^T \tag{3.41}$$

We will define $\overline{X}_k$ to satisfy the following

$$\begin{bmatrix} X_k \\ \overline{X}_k \end{bmatrix} = X_K \tag{3.42}$$

Similarly, we can defined overlined quantities of all the vectors in (3.38). In addition,

$$E[U_K U_K^T] = \text{diag}[I, \cdots, I] = I \tag{3.43}$$

$$E[V_K V_K^T] = \text{diag}[R_0, \cdots, R_K] = \mathcal{R}_K \tag{3.44}$$

and the matrices $\mathcal{A}$, $\mathcal{B}_K$, and $\mathcal{C}_K$ are given by

$$\mathcal{A} = \begin{bmatrix} E_0 & & & & & \\ -A_0 & E_1 & & & & \\ & -A_1 & E_2 & & & \\ & & \ddots & \ddots & & \\ & & & -A_{K-1} & E_K & \\ & & & & -A_K & \end{bmatrix} \tag{3.45}$$

$$\mathcal{B}_K = \text{diag}[B_{-1}, B_0, B_1, \cdots, B_{K-1}, B_K] \tag{3.46}$$

$$\mathcal{C}_K = \text{diag}[C_0, C_1, , \cdots, C_{K-1}, C_K] \tag{3.47}$$

For the purpose of defining recursive filters in Chapter 4 we will define in addition the matrices $\mathcal{A}_k$ and $\overline{\mathcal{A}}_k$ by

$$\mathcal{A}_k = \begin{bmatrix} E_0 & & & & \\ -A_0 & E_1 & & & \\ & -A_1 & E_2 & & \\ & & \ddots & \ddots & \\ & & & -A_{k-1} & E_k \end{bmatrix} \tag{3.48}$$

and

$$\overline{\mathcal{A}}_k = \begin{bmatrix} -A_k & E_{k+1} & & & \\ & -A_{k+1} & E_{k+2} & & \\ & & \ddots & \ddots & \\ & & & -A_{K-1} & E_K \\ & & & & -A_K \end{bmatrix} \tag{3.49}$$

The matrices $\overline{B}_k$, and $\overline{C}_k$ satisfy the following

$$diag(B_k, \overline{B}_k) = B_K \tag{3.50}$$

$$diag(C_k, \overline{C}_k) = C_K \tag{3.51}$$

This STPBVDS system is well-posed if $\mathcal{A}$ is invertible. However if the dimension of $x(k)$ varies with time then the partitions indicated in (3.45) will be rectangular with different but compatible sizes. Consider the inverse of $\mathcal{A}$ given by S, where S is partitioned in the same fashion as $\mathcal{A}^T$. Although the elements of $\mathcal{A}$ are provided explicitly in (3.45) we will denote the elements of $\mathcal{A}$ by $A_{i,j}$, and the elements of S by $S_{i,j}$. Since

$$S\mathcal{A} = I \tag{3.52}$$

and

$$\mathcal{A}S = I \tag{3.53}$$

the elements of these matrices satisfy

$$\sum_i S_{a,i} A_{i,b} = I\delta_{a,b} \tag{3.54}$$

and also

$$\sum_a A_{j,a} S_{a,i} = I\delta_{j,i} \tag{3.55}$$

where the identity is defined to have compatible dimension. From (3.45) it is seen

that the A matrix has a block bidiagonal structure.

$$A_{i,j} = 0 \quad j+1 \neq i \neq j \tag{3.56}$$

where the indices of A are in the set $\{[0, N+1], [0, N]\}$ As a result the above summations (3.55), and (3.56) can be written in the following fashion.

$$S_{a,i}A_{i,i} + S_{a,i+1}A_{i+1,i} = I\delta_{a,i} \qquad 0 \leq i, a \leq N \tag{3.57}$$

$$A_{i,i}S_{i,j} + A_{i,i-1}S_{i-1,j} = I\delta_{i,j} \qquad 0 < i, j < N+1 \tag{3.58}$$

when $i = 0$ or $N+1$ then (3.58) is replaced by

$$A_{0,0}S_{0,j} = I\delta_{0,j} \tag{3.59}$$

and

$$A_{N+1,N}S_{N,j} = I\delta_{N+1,j} \tag{3.60}$$

respectively. The matrices given by $S_{a,a}A_{a,a}$, $S_{a,a+1}A_{a+1,a}$, $A_{a,a}S_{a,a}$, and $A_{a,a-1}S_{a-1,a}$ are important because they are projection matrices and have interesting properties. By applying equations (3.57), and (3.58) we find the following to be true for $c < a+1$

$$\begin{aligned} S_{a,a}A_{a,a}S_{a,c} &= S_{a,c} - S_{a,a+1}A_{a+1,a}S_{a,c} \\ &= S_{a,c} - S_{a,a+2}A_{a+2,a+1}S_{a+1,c} \\ &= S_{a,c} - S_{a,N+1}A_{N+1,N}S_{N,c} \\ &= S_{a,c} \end{aligned} \tag{3.61}$$

and for $c \geq a+1$

$$S_{a,a}A_{a,a}S_{a,c} = S_{a,c} - S_{a,a+1}A_{a+1,a}S_{a,c}$$

$$
\begin{aligned}
&= S_{a,c} - S_{a,c}A_{c,c-1}S_{c-1,c} \\
&= S_{a,c} - S_{a,c} + S_{a,c}A_{c,c}S_{c,c} \\
&= -S_{a,c+1}A_{c+1,c}S_{c,c} \\
&= -S_{a,N+1}A_{N+1,N}S_{N,c} \\
&= 0 \qquad (3.62)
\end{aligned}
$$

as a result by considering the case where $c = a$ it follows that $S_{a,a}A_{a,a}$, and $A_{a,a}S_{a,a}$ are projection matrices.

Consider the system given by the STPBVDS

$$A_{i,i}x(i) = -A_{i,i-1}x(i-1) + \mu(i) \qquad (3.63)$$

with boundary conditions given by

$$A_{0,0}x(0) = \mu(0) \qquad (3.64)$$

and

$$A_{N+1,N}x(N) = \mu(N+1) \qquad (3.65)$$

The solution to this equation is given directly in terms of the elements of S.

$$x(i) = \sum_j S_{i,j}\mu(j) \qquad (3.66)$$

Consider premultiplying $x(i)$ by the product $S_{i,i}A_{i,i}$

$$S_{i,i}A_{i,i}x(i) = \sum_j S_{i,i}A_{i,i}S_{i,j}\mu(j) \qquad (3.67)$$

From (3.61), and (3.62), the projection matrix $S_{i,i}A_{i,i}$ kills future dynamics in the system (3.63).

$$S_{i,i}A_{i,i}x(i) = \sum_{j<i+1} S_{i,i}A_{i,i}S_{i,j}\mu(j) \qquad (3.68)$$

Similarly from (3.57) it follows that $I - S_{i,i}A_{i,i} = S_{i,i+1}A_{i+1,i}$ is a projection matrix

which kills the influence of past dynamics.

$$S_{i,i+1}A_{i+1,i}x(i) = \sum_{j>i} S_{i,i+1}A_{i+1,i}S_{i,j}\mu(j) \tag{3.69}$$

Since from (3.57) the set of projection matrices at each point in time are complete, it therefore follows that if recursive equations can be written for the states when projected upon the space of past dynamics then the system will be successfully decoupled into forward and past dynamics. Consider the product $A_{a,a-1}S_{a-1,a}A_{a,a}$, this can be shown to be equivalent to the product $A_{i,i}S_{i,i+1}A_{i+1,i}$. Also the product $A_{i,i}S_{i,i}A_{i,i-1}$, can be shown to be equal to $A_{i,i-1}S_{i-1,i-1}A_{i-1,i-1}$. As a result the system given by (3.63) can be decoupled into a forward and backward propagating subsystem.

$$A_{i,i}S_{i,i}(A_{i,i}x(i)) = A_{i,i}S_{i,i}(A_{i,i-1}x(i-1) + \mu(i)) \tag{3.70}$$

$$= A_{i,i}(S_{i,i}A_{i,i})x(i) = A_{i,i-1}(S_{i-1,i-1}A_{i-1,i-1})x(i-1) + A_{i,i}S_{i,i}\mu(i) \tag{3.71}$$

We need to establish that (3.71) is causal. In (3.70), which is equivalent to (3.71), the premultiplication by $A_{i,i}S_{i,i}$ allows at most rank$(A_{i,i}S_{i,i})$ degrees of freedom on both sides of (3.70) and (3.71). Since rank$(A_{i,i}S_{i,i}A_{i,i})$ is equal to both rank$(A_{i,i}S_{i,i})$ and rank$(S_{i,i}A_{i,i})$, the rank$(S_{i,i}A_{i,i})$ degrees of freedom in $S_{i,i}A_{i,i}x(i)$ are constrained only by the right hand side of (3.70) and (3.71). Since information is not lost by premultiplying the boundary condition in (3.64) by $S_{0,0}$ due to (3.59). This system is therefore a causal recursion for $S_{i,i}A_{i,i}x(i)$. Similarly,

$$A_{i,i-1}S_{i-1,i}[A_{i,i}x(i)] = A_{i,i-1}S_{i-1,i}[A_{i,i-1}x(i-1) + \mu(i)] \tag{3.72}$$

$$= A_{i,i}[S_{i,i+1}A_{i+1,i}]x(i) \tag{3.73}$$

$$= A_{i,i-1}[S_{i-1,i}A_{i,i-1}]x(i-1) + A_{i,i-1}S_{i-1,i}\mu(i) \tag{3.74}$$

which in turns denotes a system which propagates backwards because the rank of $A_{i,i-1}[S_{i-1,i}A_{i,i-1}]$ is equal to the rank of $S_{i-1,i}A_{i,i-1}$.

At this point it helps to translate these results into the notation for STPBVDS.

The system is given by

$$E_{k+1}x(k+1) = A_k x(k) + B_k u(k) \tag{3.75}$$

where $E_k = A_{k,k}$, $A_k = A_{k+1,k}$, and $\mu_k = B_{k-1}u(k-1)$. Define the matrices $P^f_{k,k} = S_{i,i}A_{i,i}$, $P^b_{k,k} = S_{i,i+1}A_{i+1,i}$, $P^f_{k+1,k} = A_{i,i}S_{i,i}$, $P^b_{k+1,k} = A_{i,i-1}S_{i-1,i}$. Recall that since S exists, the system is well-posed. The forward propagated state is given by $P^f_{k,k}x(k)$, and the backward propagated state is given by $P^b_{k,k}x(k)$. In fact the E_k and A_k matrices have been successfully broken into two parts. The matrix E_{k+1} has been broken into $P^f_{k+1,k}E_{k+1}P^f_{k+1,k+1}$, and $P^b_{k+1,k}E_{k+1}P^b_{k+1,k+1}$. The matrix A_k has been broken into $P^f_{k+1,k}A_kP^f_{k,k}$, and $P^b_{k+1,k}A_kP^b_{k,k}$.

Next our aim is to find an invertible coordinate transformation M_k which will allow (3.75) to be partitioned into forward and backward subsystems (with decomposed boundary conditions) of reduced dimension. Following from the experience of diagonalizing the Hamiltonian equations for causal systems[22],[1] we expect that an additional matrix N_k will be needed to premultiply (3.75) to aid in this diagonalizing decomposition. Let the columns of N_k be the right eigenvectors of the projection matrix $P^f_{k+1,k}$ (which therefore are also the right eigenvectors of $P^b_{k+1,k} = I - P^f_{k+1,k}$. That is,

$$N_k = \begin{bmatrix} N_{1,k} & N_{2,k} \end{bmatrix} \tag{3.76}$$

where for each column, η, in $N_{1,k}$, $P^f_{k+1,k}\eta = \eta$, and for each column, η, in $N_{2,k}$, $P^f_{k+1,k}\eta = 0$. Then the projection matrix $P^f_{k+1,k}$ can be written as

$$P^f_{k+1,k} = \begin{bmatrix} N_{1,k} & 0 \end{bmatrix} \begin{bmatrix} N_{1,k} & N_{2,k} \end{bmatrix}^{-1} \tag{3.77}$$

and $\overline{P}^f_{k+1,k}$, which in the notatiion following (3.75) also equals $P^b_{k+1,k}$, by

$$P^b_{k+1,k} = \begin{bmatrix} 0 & N_{2,k} \end{bmatrix} \begin{bmatrix} N_{1,k} & N_{2,k} \end{bmatrix}^{-1} \tag{3.78}$$

Similarly, let the rows of M_k be the left eigenvectors of the projection matrix $P^f_{k,k}$

(which therefore are also the left eigenvectors of $P^b_{k,k} = I - P^f_{k,k} = \overline{P}^f_{k,k}$). That is,

$$M_k = \begin{bmatrix} M_{1,k} \\ M_{2,k} \end{bmatrix} \tag{3.79}$$

where for each column, m, in $M_{1,k}$, $P^f_{k,k}m = m$, and for each column, m, in $M_{2,k}$, $P^f_{k,k}m = 0$. Thus,

$$P^f_{k,k} = \begin{bmatrix} M_{1,k} \\ M_{2,k} \end{bmatrix}^{-1} \begin{bmatrix} M_{1,k} \\ 0 \end{bmatrix} \tag{3.80}$$

and $\overline{P}^f_{k,k} = P^b_{k,k}$ is given by

$$P^b_{k,k} = \begin{bmatrix} M_{1,k} \\ M_{2,k} \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ M_{2,k} \end{bmatrix} \tag{3.81}$$

These matrices are invertible since they comprise the eigenvectors of a projection matrix and its complement.

As a result, the equation given by

$$N^{-1}_{k+1}E_{k+1}M^{-1}_{k+1}\{M_{k+1}x(k+1)\} = N^{-1}_{k+1}A_kM^{-1}_k\{M_kx(k)\} + N^{-1}_{k+1}B_ku(k) \tag{3.82}$$

has the form

$$\begin{bmatrix} L^f_{k+1} & 0 \\ 0 & L^b_{k+1} \end{bmatrix} \begin{bmatrix} x^f(k+1) \\ x^b(k+1) \end{bmatrix} = \begin{bmatrix} J^f_k & 0 \\ 0 & J^b_k \end{bmatrix} \begin{bmatrix} x^f(k) \\ x^b(k) \end{bmatrix} + \begin{bmatrix} B^f_k \\ B^b_k \end{bmatrix} u(k) \tag{3.83}$$

where the sequences L^a_{k+1} and J^b_k are square and invertible matrices. Finally the STPBVDS can be written as

$$\begin{bmatrix} I^f_{k+1} & 0 \\ 0 & J^{b,-1}_k L^b_{k+1} \end{bmatrix} \begin{bmatrix} x^f(k+1) \\ x^b(k+1) \end{bmatrix} = \begin{bmatrix} L^{a,-1}_{k+1}J^b_k & 0 \\ 0 & I^b_k \end{bmatrix} \begin{bmatrix} x^f(k) \\ x^b(k) \end{bmatrix} + \begin{bmatrix} L^{a,-1}_{k+1}B^f_k \\ J^{b,-1}_k B^b_k \end{bmatrix} u(k) \tag{3.84}$$

where the super and subscripts on the identity matrices are to indicate that they may

vary in size. In addition $x^f(k) = M_{1,k}x(k)$, and $x^b(k) = M_{2,k}x(k)$. The next step is to show that all STPBVDS are Markov.

3.3 Markovianity of STPBVDS's

To demonstrate the Markovianity we need to demonstrate that given $x(k)$, the states $x(k+\tau)$, and $x(k-s)$ are independent for any $s, \tau > 0$. We will use the diagonalized representation of the system to prove this assertion. Clearly the forward and backward subsystems are separately Markov. We represent the STPBVDS by

$$\begin{bmatrix} I & 0 \\ 0 & E^b_{k+1} \end{bmatrix} \begin{bmatrix} x^f(k+1) \\ x^b(k+1) \end{bmatrix} = \begin{bmatrix} A^f_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x^f(k) \\ x^b(k) \end{bmatrix} + \begin{bmatrix} b^f_k \\ b^b_k \end{bmatrix} u(k) \tag{3.85}$$

where obvious substitutions from (3.84) has been made. By defining forward and backward state transition matrices as

$$\phi^f(L,K) = A^f_{L-1} \cdots A^f_{K+1} A^f_K \tag{3.86}$$

$$\phi^b(K,L) = E^b_{K+1} \cdots E^b_{L-1} E^b_L \tag{3.87}$$

where $K \leq L$ we may then write the variation of constants formula for the forward and backward subsystems.

$$x^f(k) = \phi^f(k,j)x^f(j) + \sum_{\kappa=j}^{k-1} \phi^f(k,\kappa+1) b^f_\kappa u(\kappa) \tag{3.88}$$

$$x^b(j) = \phi^b(j,k)x^b(k) + \sum_{\kappa=j}^{k-1} \phi^b(j,\kappa) b^b_\kappa u(\kappa) \tag{3.89}$$

or equivalently

$$x^f(k) = \phi^f(k,j)x^f(j) + x^f(k;j) \tag{3.90}$$

$$x^b(j) = \phi^b(j,k)x^b(k) + x^b(j;k) \tag{3.91}$$

where obvious substitutions have been made. Restricting the process to a subinterval involves restricting the causal and anticausal systems to the same subinterval. The net result is that restrictions of STPBVDS's are themselves STPBVDS's. To prove Markovianity we need to demonstrate that

$$E[(x(0) - E[x(0)|x(t)])(x(K) - E[x(K)|x(t)])^T] = 0 \tag{3.92}$$

This statement is equivalent to

$$E[x(0)x^T(K)] = E[E[x(0)|x(t)]E[x^T(K)|x(t)]] \tag{3.93}$$

which is also equivalent to

$$E[x(0)x^T(K)] = E[x(0)x^T(t)]E[xT(t)x^T(t)]^{-1}E[x(t)x^T(K)] \tag{3.94}$$

and finally in terms of the diagonalized system,

$$E\left[\begin{bmatrix} x^f(0) \\ x^b(0) \end{bmatrix}\begin{bmatrix} x^{f,T}(K) & x^{b,T}(K) \end{bmatrix}\right] = E\left[\begin{bmatrix} x^f(0) \\ x^b(0) \end{bmatrix}\begin{bmatrix} x^{f,T}(k) & x^{b,T}(k) \end{bmatrix}\right]$$
$$\times E\left[\begin{bmatrix} x^f(k) \\ x^b(k) \end{bmatrix}\begin{bmatrix} x^{f,T}(k) & x^{b,T}(k) \end{bmatrix}\right]^{-1} E\left[\begin{bmatrix} x^f(k) \\ x^b(k) \end{bmatrix}\begin{bmatrix} x^{f,T}(K) & x^{b,T}(K) \end{bmatrix}\right] \tag{3.95}$$

This equality can be easily established by considering the following substitutions

$$x^f(K) = \phi^f(K,k)x^f(k) + x^f(K;k) \tag{3.96}$$
$$x^b(0) = \phi^b(0,k)x^b(k) + x^b(0;k) \tag{3.97}$$
$$x^f(0) = b^f_{-1}u(-1) \tag{3.98}$$
$$x^b(K) = b^b_K u(K) \tag{3.99}$$

where $x^f(k)$ and $x^b(k)$ are independent random vectors.

It is a simple matter to demonstrate that the first expectation in equation (3.95)

is given by

$$E\left[\begin{bmatrix} x^f(0) \\ x^b(0) \end{bmatrix} \begin{bmatrix} x^{f,T}(K) & x^{b,T}(K) \end{bmatrix}\right] \tag{3.100}$$
$$= \begin{bmatrix} E[x^f(0)x^{f,T}(k)]\phi^{f,T}(K,k) & 0 \\ \phi^b(0,k)E[x^b(k)x^{f,T}(K;k) + E[x^b(0;k)x^{f,T}(k)]\phi^{f,T}(K,k) & \phi^b(0,k)E[x^b(k)x^{b,T}(K)] \end{bmatrix}$$

The second expectation in equation (3.95) is given by

$$E\left[\begin{bmatrix} x^f(0) \\ x^b(0) \end{bmatrix} \begin{bmatrix} x^{f,T}(k) & x^{b,T}(k) \end{bmatrix}\right] = \begin{bmatrix} E[x^f(0)x^{f,T}(k)] & 0 \\ E[x^b(0;k)x^{f,T}(k)] & \phi^b(0,k)\Sigma_k^b \end{bmatrix} \tag{3.101}$$

The third expectation in equation (3.95) is given by

$$E\left[\begin{bmatrix} x^f(k) \\ x^b(k) \end{bmatrix} \begin{bmatrix} x^{f,T}(k) & x^{b,T}(k) \end{bmatrix}\right] = \begin{bmatrix} \Sigma_k^f & 0 \\ 0 & \Sigma_k^b \end{bmatrix} \tag{3.102}$$

Finally, the fourth expectation in equation (3.95) is given by

$$E\left[\begin{bmatrix} x^f(k) \\ x^b(k) \end{bmatrix} \begin{bmatrix} x^{f,T}(K) & x^{b,T}(K) \end{bmatrix}\right] = \begin{bmatrix} \Sigma_k^f\phi^{f,T}(K,k) & 0 \\ E[x^b(k)x^{f,T}(K;k)] & E[x^b(k)x^{b,T}(K)] \end{bmatrix} \tag{3.103}$$

From these relations (3.95) is confirmed and the system is Markov.

Chapter 4

Estimation of Separable Two Point Boundary Value Descriptor Systems

4.1 Machinery for Recursive Estimation

In this section machinery is presented which is important to perform recursive estimation for STPBVDS's. Lemma 4.1 which is a generalization of a result in [23] to the case where x need not be estimable, provides one way around the use of the Hamiltonian while insuring that what we compute at each point in time are optimal ML estimates. Lemma 4.2 shows that the estimate can be decomposed into two independent estimates, which by virtue of Lemma 4.1, each estimate can be implemented recursively. Lemma 4.3 allows for the updating of state parameters when additional indirect measurements are available.

4.1.1 ML Recursive Estimation Lemma

One method to derive smoothing equations for STPBVDS's is to solve the entire problem as one large optimization problem. The result is the Hamiltonian formulation for the optimal smoother[24]. The Hamiltonian itself is a STPBVDS which relates the

smoothed estimate of the state with the smoothed estimate of its neighbors. Useful algorithms are obtained when the dynamics are diagonalized and triangularized, yielding the Mayne-Fraser and Rauch-Tung-Striebel algorithms respectively. However it is not clear that the state of the filters in the Mayne-Fraser and the Rauch-Tung-Striebel algorithms are themselves ML estimates. In addition, modifying the Hamiltonian for the case of ill-defined covariances adds additional complication to the algebraic process of diagonalization, and triangularization. A detailed understanding of the smoothing problem is enhanced by examining the estimation problem at each point in time, instead of solving the entire problem at once and recovering recursive computations from the final solution. Lemma 4.1 provides one way around the use of the Hamiltonian while insuring that what we compute at each point in time is an optimal ML estimate. In addition, Lemma 4.1 allows us the freedom to incorporate measurements in any order, in addition to incorporating them in a sequential fashion, leading us in Section 5.5 to a new parallel recursive estimation algorithms.

Lemma 4.1

The optimal estimate of the vector z (or x) based on the measurements

$$y = Hx + v \tag{4.1}$$

and

$$w = Jx + Kz + u \tag{4.2}$$

is equal to the estimate of the vector z (or x) based on the measurements

$$\hat{x}[y] = P_x x + \tilde{x}[y] \tag{4.3}$$

and (4.2), where $\hat{x}[y]$ is the estimate of x based on (4.1) alone, P_x is the symmetric projection matrix which projects onto the estimable subspace of x, $\tilde{x}[y]$ is the estimation error associated with $\hat{x}[y]$, and u and v are zero mean, independent random

vectors.

Proof of Lemma 4.1

We will prove this using the same machinery and notation used in the Section 2.2 in the context of square-root ML estimation. The main task is to show that the two sets of observations are equivalent by noting that the information which is thrown away in the estimation x based on y, would be eliminated anyway in estimating z (or x) based on all of the data.

Here, the estimate of x based on y will be constructed while maintaining all of the information which is ordinarily lost in the estimation process.

Premultiply (4.1) by $\mathcal{S}$ in (2.91).

$$\begin{bmatrix} S_p \\ S_s \\ S_u \end{bmatrix} y = \begin{bmatrix} S_p \\ S_s \\ S_u \end{bmatrix} Hx + \begin{bmatrix} 0 \\ S_s \\ S_u \end{bmatrix} v \tag{4.4}$$

Use the matrix U given in (2.94) to separate the parts of x which can be estimated perfectly, and are completely unknown from the remainder.

$$\begin{bmatrix} S_p \\ S_s \\ S_u \end{bmatrix} y = \begin{bmatrix} S_p H U_p^T & 0 & 0 \\ S_s H U_p^T & S_s H U_s^T & 0 \\ S_u H U_p^T & S_u H U_s^T & S_u H U_u^T \end{bmatrix} \begin{bmatrix} U_p x \\ U_s x \\ U_u x \end{bmatrix} + \begin{bmatrix} 0 \\ S_s \\ S_u \end{bmatrix} v \tag{4.5}$$

In this next step we shall estimate the perfect part of x and normalize the part of the observation noise which has an invertible covariance.

$$\begin{bmatrix} \{S_p H U_p^T\}^{-L} S_p \\ R_o^{-1/2} S_s \\ S_u \end{bmatrix} y = \begin{bmatrix} I & 0 & 0 \\ R_o^{-1/2} S_s H U_p^T & R_o^{-1/2} S_s H U_s^T & 0 \\ S_u H U_p^T & S_u H U_s^T & S_u H U_u^T \end{bmatrix} \begin{bmatrix} U_p x \\ U_s x \\ U_u x \end{bmatrix} + \begin{bmatrix} 0 \\ R_o^{-1/2} S_s \\ S_u \end{bmatrix} v \tag{4.6}$$

Since the perfect portion of x has been identified, what remains is to find the matrix

J used in (2.96) which solves the standard ML problem.

$$\begin{bmatrix} \{S_p H U_p^T\}^{-L} S_p \\ J R_o^{-1/2} S_s \\ S_u \end{bmatrix} y \tag{4.7}$$
$$= \begin{bmatrix} I & 0 & 0 \\ J R_o^{-1/2} S_s H U_p^T & J R_o^{-1/2} S_s H U_s^T & 0 \\ S_u H U_p^T & S_u H U_s^T & S_u H U_u^T \end{bmatrix} \begin{bmatrix} U_p x \\ U_s x \\ U_u x \end{bmatrix} + \begin{bmatrix} 0 \\ J R_o^{-1/2} S_s \\ S_u \end{bmatrix} v$$

The term $J R_o^{-1/2} S_s v$ has unity covariance and next we examine the decomposition performed in (2.83). Toward this end, J is partitioned appropriately as $J^T = [J_1^T \ J_2^T]$ Then (4.7) can be written as

$$\begin{bmatrix} \{S_p H U_p^T\}^{-L} S_p \\ J_1 R_o^{-1/2} S_s \\ J_2 R_o^{-1/2} S_s \\ S_u \end{bmatrix} y \tag{4.8}$$
$$= \begin{bmatrix} I & 0 & 0 \\ J_1 R_o^{-1/2} S_s H U_p^T & J_1 R_o^{-1/2} S_s H U_s^T & 0 \\ J_2 R_o^{-1/2} S_s H U_p^T & 0 & 0 \\ S_u H U_p^T & S_u H U_s^T & S_u H U_u^T \end{bmatrix} \begin{bmatrix} U_p x \\ U_s x \\ U_u x \end{bmatrix} + \begin{bmatrix} 0 \\ J_1 R_o^{-1/2} S_s \\ J_2 R_o^{-1/2} S_s \\ S_u \end{bmatrix} v$$

The term $J_1 R_o^{-1/2} S_s H U_s^T$ is invertible and is the inverse square root of the covariance of $U_s x$. This measurement can be written as two measurements. The first measurement is equivalent to (4.3), the second are those parts which are discarded in the estimation x based on y

$$\begin{bmatrix} \{S_p H U_p^T\}^{-L} S_p \\ \{J_1 R_o^{-1/2} S_s H U_s^T\}^{-1} J_1 R_o^{-1/2} S_s \end{bmatrix} y = \tag{4.9}$$
$$\begin{bmatrix} I & 0 \\ J_1 R_o^{-1/2} S_s H U_p^T & I \end{bmatrix} \begin{bmatrix} U_p x \\ U_s x \end{bmatrix} + \begin{bmatrix} 0 \\ \{J_1 R_o^{-1/2} S_s H U_s^T\}^{-1} J_1 R_o^{-1/2} S_s \end{bmatrix} v$$

and

$$\begin{bmatrix} J_2 R_o^{-1/2} S_s & -J_2 R_o^{-1/2} S_s H U_p^T \\ S_u & -S_u H U_p^T \end{bmatrix} \begin{bmatrix} y \\ U_p x \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ S_u H U_s^T & S_u H U_u^T \end{bmatrix} \begin{bmatrix} U_s x \\ U_u x \end{bmatrix} + \begin{bmatrix} J_2 R_o^{-1/2} S_s \\ S_u \end{bmatrix} v$$

From (4.9) the estimate of x based on y can be directly recovered and is given by

$$\begin{bmatrix} \{S_p H U_p^T\}^{-L} S_p \\ \{J_1 R_o^{-1/2} S_s H U_s^T\}^{-1} J_1 R_o^{-1/2} S_s - \{S_p H U_p^T\}^{-L} S_p J_1 R_o^{-1/2} S_s H U_p^T \end{bmatrix} y \quad (4.10)$$
$$= \begin{bmatrix} U_p \hat{x}_{ML} \\ U_s \hat{x}_{ML} \end{bmatrix} = \begin{bmatrix} U_p x \\ U_s x \end{bmatrix} + \begin{bmatrix} 0 \\ \{J_1 R_o^{-1/2} S_s H U_s^T\}^{-1} J_1 R_o^{-1/2} S_s \end{bmatrix} v$$

At this point (4.1) and (4.2) has been shown to be equal to (4.2), (4.3) and (4.10). Equation (4.10) can be discarded from the estimation process for two reasons. A direct observation is obtained for the noise term $J_2 R_o^{-1/2} S_s v$ which however is independent of u and thus contributes nothing to (4.2). Finally since $S_u v$ is completely unknown, and is not observed in (4.2), this equation places no constraints on x. The equality between the two sets of measurements is therefore established.

4.1.2 Independent measurement ML Lemma

The Mayne-Fraser two filter algorithm computes smoothed estimates by combining pairs of independent estimates. Lemma 4.2 demonstrates that the two independent measurements of a specific vector can be obtained from two general measurements, both of which involve the vector of interest.

Definition 4.1

The vectors $\tilde{a}$ and $\tilde{b}$, which are comprised of both random variables and unknown parameters, will be called independent if they satisfy the following.

- There is an invertible matrix $[M_a^T \quad N_a^T]$ such that $M_a \tilde{a}$ is a Gaussian random variable with a well defined covariance, and $N_a \tilde{a}$ is completely unknown.
- There is an invertible matrix $[M_b^T \quad N_b^T]$ such that $M_b \tilde{b}$ is a Gaussian random

variable with a well defined covariance, and $N_b\tilde{b}$ is completely unknown.

- $E[(M_a\tilde{a} - E[M_a\tilde{a}])(M_b\tilde{b} - E[M_b\tilde{b}])^T] = 0$
- There is no redundancy in $N_a\tilde{a}$, and $N_b\tilde{b}$, i.e. together they represent rank(N_a) + rank(N_b) degrees of freedom.

Definition 4.2

For any matrix G, $G^\perp$ is defined to be any matrix which satisfies the following.

$$G^\perp G = 0 \tag{4.11}$$

$$\begin{bmatrix} G^\perp \\ G^T \end{bmatrix} \text{ has full column rank} \tag{4.12}$$

Lemma 4.2

Let $\tilde{a}$ and $\tilde{b}$ be independent vectors. The optimal estimate of x given the independent observations

$$a = G_a z + J_a x + \tilde{a} \tag{4.13}$$

$$b = G_b z + J_b x + \tilde{b} \tag{4.14}$$

is equal to the optimal estimate of x given the independent observations

$$G_a^\perp a = G_a^\perp J_a x + G_a^\perp \tilde{a} \tag{4.15}$$

$$G_b^\perp b = G_b^\perp J_b x + G_b^\perp \tilde{b} \tag{4.16}$$

Furthermore, the observations given by (4.13), and (4.14) are equivalent to

$$\hat{x}[a] = P_a x + \tilde{x}[a] \tag{4.17}$$

$$\hat{x}[b] = P_b x + \tilde{x}[b] \tag{4.18}$$

where the indices indicate which observation the estimates where obtained from.

Proof of Lemma 4.2

Multiply (4.13) by $[G_a^{\perp T} \;\; G_a]^T$ and (4.14) by $[G_b^{\perp T} \;\; G_b]^T]$.

$$\begin{bmatrix} G_a^{\perp} \\ G_a^T \end{bmatrix} a = \begin{bmatrix} G_a^{\perp} \\ G_a^T \end{bmatrix} J_a x + \begin{bmatrix} G_a^{\perp} \tilde{a} \\ G_a^T \tilde{a} + G_a^T G_a z \end{bmatrix} \tag{4.19}$$

$$\begin{bmatrix} G_b^{\perp} \\ G_b^T \end{bmatrix} b = \begin{bmatrix} G_b^{\perp} \\ G_b^T \end{bmatrix} J_b x + \begin{bmatrix} G_b^{\perp} \tilde{b} \\ G_b^T \tilde{b} + G_b^T G_b z \end{bmatrix} \tag{4.20}$$

z is an unknown vector. As a result the bottom halves of (4.19), and (4.20) do not provide algebraic or statistical constraints for x. Upon removing these equations (4.15), and (4.16) result. Equations (4.17), and (4.18) follow directly from the application of Lemma 4.1 to (4.15), and (4.16).

4.1.3 Updating ML Lemma

Lemma 4.3 is important to the derivation of the Rauch-Tung-Striebel algorithm and the parallel algorithm developed in Section 4.3.

Lemma 4.3

Let $\tilde{b}$ be independent from both $\tilde{\alpha}$ and $\tilde{\beta}$. Then the set of measurements given by

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} L_{aa} & L_{ab} \\ 0 & L_{bb} \end{bmatrix} \begin{bmatrix} x_a \\ x_b \end{bmatrix} + \begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix} \tag{4.21}$$

and

$$b = H x_b + \tilde{b} \tag{4.22}$$

where the observation error is given by

$$\begin{bmatrix} P_\alpha & 0 \\ 0 & P_\beta \end{bmatrix} \begin{bmatrix} \tilde{\alpha} \\ \tilde{\beta} \end{bmatrix} \sim N \left[0; \begin{bmatrix} \Sigma_{\alpha,\alpha} & \Sigma_{\alpha,\beta} \\ \Sigma_{\beta,\alpha} & \Sigma_{\beta,\beta} \end{bmatrix} \right] \tag{4.23}$$

$$P_b \tilde{b} \sim N[0; \Sigma_{bb}] \tag{4.24}$$

is equivalent to (4.21) and

$$B = x_b + \tilde{B} \tag{4.25}$$

where (4.25) is the estimate of x_b based on (4.22) and the second half of (4.21) given by

$$\beta = L_{bb} x_b + \tilde{\beta} \tag{4.26}$$

Here,

$$B = \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}^T \begin{bmatrix} \Sigma_{\beta,\beta} & 0 & L_b \\ 0 & P_{b,b} & H \\ L_b^T & H^T & 0 \end{bmatrix}^{\#} \begin{bmatrix} \beta \\ b \\ 0 \end{bmatrix} \tag{4.27}$$

and

$$\tilde{B} \sim N(0, Q) \tag{4.28}$$

where Q is given by

$$Q = - \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}^T \begin{bmatrix} \Sigma_{\beta,\beta} & 0 & L_b \\ 0 & \Sigma_{b,b} & H \\ L_b^T & H^T & 0 \end{bmatrix}^{\#} \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix} \tag{4.29}$$

In addition, the measurement given by (4.21) and (4.25) yields an equivalent measurement for x_a given by

$$L_{aa}\hat{x}_a = -\{L_{ab} - \Sigma_{\alpha,\beta}\Sigma^{\#}_{\beta,\beta}L_b\}B + \alpha - \Sigma_{\alpha,\beta}\Sigma^{\#}_{\beta,\beta}\beta \tag{4.30}$$

and its error covariance is given by

$$Cov(L_{aa}\hat{x}_a - L_{aa}x_a) = \Sigma_{\alpha,\alpha} - \Sigma_{\alpha,\beta}\Sigma^{\#}_{\beta,\beta}\Sigma_{\beta,\alpha} + \{L_{ab} - \Sigma_{\alpha,\beta}\Sigma^{\#}_{\beta,\beta}L_b\}Q\{L_{ab} - \Sigma_{\alpha,\beta}\Sigma^{\#}_{\beta,\beta}L_b\}^T \tag{4.31}$$

Proof of Lemma 4.3

As in Lemma 4.1, our aim is two show equivalence between the two sets by keeping track of the information used by each set of observations and evaluating whether the

difference in the observations are necessary to perform the desired estimation.

The observations in (4.21) can be divided by an invertible transformation into two independent measurements given by:

$$\alpha - \Sigma_{\alpha,\beta}\Sigma^{\#}_{\beta,\beta}\beta = L_{aa}x_a + \{L_{ab} - \Sigma_{\alpha,\beta}\Sigma^{\#}_{\beta,\beta}L_b\}x_b + \{\tilde{\alpha} - \Sigma_{\alpha,\beta}\Sigma^{\#}_{\beta,\beta}\tilde{\beta}\} \tag{4.32}$$

and

$$\beta = L_{bb}x_b + \tilde{\beta} \tag{4.33}$$

The measurements given by (4.21), and (4.22), are thus equivalent to (4.32), (4.33) and (4.22). Equating (4.21) with (4.2), and equating (4.22), and (4.33) with (4.1) yields via Lemma 4.1 that the above observations are equivalent to (4.32) and (4.25). Clearly no new information is recovered by including and additional (4.33). Recombining (4.32) equation with (4.33) demonstrates that the two sets of measurements are equivalent. Equations (4.30), and (4.31) follow directly from equations (4.32), and (4.33).

Lemma 4.3 and its implications for recursive estimation are explored in more detail in Section 4.3, in which we derive the Rauch-Tung-Striebel smoothing algorithm for STPBVDS's.

4.2 Consequences of Lemmas 4.1 and 4.2

The first proposition applies Lemma 4.1 to STPBVDS's. Using the notation for STPBVDS's in Chapter 3, let $\hat{x}[\kappa|Y_\kappa]$ be the ML estimate of $x(\kappa)$ based on the observations Y_κ defined in equation (3.40) for $k = \kappa$. We may conclude that

Proposition 4.1

The estimate $\hat{x}[k|Y_k]$, is a sufficient statistic to carry along for the causal recursive estimation problem for STPBVDS's.

Proof of Proposition 4.1

From Lemma 4.1 the ML estimate of $x(k+1)$ based on

$$\mathcal{A}_{k+1}X_{k+1} = \mathcal{B}_{k+1}U_{k+1} \tag{4.34}$$

$$Y_{k+1} = \mathcal{C}_{k+1}X_{k+1} + V_{k+1} \tag{4.35}$$

is equivalent to the measurement

$$\begin{bmatrix} 0 & \cdots & 0 & -A_k \end{bmatrix} X_k + E_{k+1}x(k+1) = B_{k+1}u_{k+1} \tag{4.36}$$

$$y_{k+1} = C_{k+1}x(k+1) + v(k+1) \tag{4.37}$$

and the optimal estimate of X_k based on

$$\mathcal{A}_k X_k = \mathcal{B}_k U_k \tag{4.38}$$

$$Y_k = \mathcal{C}_k X_k + \tilde{Y}_k \tag{4.39}$$

Due to the bidiagonal structure of $\mathcal{A}_k$ the matrix in (4.36) which premultiplies X_k acts only on $x(k)$. From Lemma 4.2, all that is required from (4.38) is an equivalent measurement of only $x(k)$ since the remainder of X_k is not needed. As a result, the ML estimate of $x(k)$ based on Y_k, $\hat{x}[k|Y_k]$, is a sufficient statistic to carry along for the recursive estimation problem for STPBVDS's.

Proposition 4.2

The estimation problem for STPBVDS's can be reduced for each k to the estimation of $x(k)$ based on two independent measurements.

Proof of Proposition 4.2

For each k, the matrix $\mathcal{A}$ defined in (3.36) can be partitioned into two parts yielding the following partitioning of (3.36), and (3.37).

$$\begin{bmatrix} 0 \\ Y_k \end{bmatrix} = \begin{bmatrix} \mathcal{A}_k \\ C_k \end{bmatrix} \begin{bmatrix} X_{k-1} \\ x(k) \end{bmatrix} + \begin{bmatrix} -B_k U_k \\ V_k \end{bmatrix} \tag{4.40}$$

$$\begin{bmatrix} \overline{\mathcal{A}}_k \\ 0 \ \vdots \ \overline{C}_k \end{bmatrix} \begin{bmatrix} x(k) \\ \cdots \\ \overline{X}_k \end{bmatrix} = \begin{bmatrix} \overline{B}_k \overline{U}_k \\ \overline{V}_k \end{bmatrix} \tag{4.41}$$

The point here is that (4.40) and (4.41) can be thought of as two observations of the vector $x(k)$, capturing all of the available information. We know from Lemma 4.2 that equivalent measurements of $x(k)$ can be constructed for (4.40) and (4.41). From Lemma 4.1 these measurements can be reduced to independent estimates of $x(k)$. In the section on the Mayne-Fraser smoother we will show how these estimates can be constructed recursively.

4.3 Maximum Likelihood Filtering

In this section we present a general ML filtering algorithm for STPBVDS. We assume that the system matrices E_k, A_k, etc. may be time varying and that the process noise $u(k)$ is independent of the observation noise $v(k)$. The algorithm uses two intermediary vectors $z^f(k)$, and $z^b(k)$ which represent the forward and backward predicted estimates of the state $x(k)$. This is needed particularly in the case in which the E_k and the A_k matrices are rank deficient.

Consider a general STPBVDS:

$$E_{k+1}x(k+1) = A_k x(k) + B_k u(k) \quad 0 \le k \le K-1 \tag{4.42}$$

$$E_0 x(0) = B_{-1} u(-1) \tag{4.43}$$

$$0 = A_K x(K) + B_K u(K) \tag{4.44}$$

$$y(t) = C_k x(k) + v(k) \qquad 0 \le k \le K \tag{4.45}$$

$$\mathrm{Cov}(u(k)) = I \tag{4.46}$$

$$\mathrm{Cov}(v(k)) = R_k \tag{4.47}$$

Viewing (4.42) through (4.47) as providing a set of noisy constraints, we can apply the ML estimation results of Subsection 2.1, Lemma 4.1, Proposition 4.1, and

Proposition 4.2 to obtain recursive estimation algorithms. From Proposition 4.2, the algorithm involves constructing two independent measurements of the random variables $x(k)$ for each k. From Proposition 4.1, these independent measurements are computed from two ML filters one operating on only causal data and the other operating on only anticausal data. Each filter computes recursively ML estimates which propagate ML estimates, a projection matrix which keeps track of the estimable subspace, and the covariance of the estimable portion of $x(k)$.

In presenting these algorithms it is useful to define two auxiliary (forward and backward prediction) variables.

$$z^f(k) = E_k x(k) \tag{4.48}$$

$$z^b(k) = A_k x(k) \tag{4.49}$$

Let $\hat{x}^f_{ML}[l|k]$ denote the ML estimate of $x(l)$ based on (4.42) for $0 \leq \kappa \leq k-1$, and (4.47) for $0 \leq \kappa \leq k$, and $\hat{z}^f_{ML}[l|k]$ denote the ML estimate of $z(l)$ based on (4.42) for $0 \leq \kappa \leq k$, and (4.47) for $0 \leq \kappa \leq k$. Then the observations required for the measurement update step are

$$\begin{bmatrix} \hat{z}^f_{ML}[k|k-1] \\ y(k) \end{bmatrix} = \begin{bmatrix} P_{z^f[k|k-1]} E_k \\ C_k \end{bmatrix} x(k) + \begin{bmatrix} \tilde{z}^f_{ML}[k|k-1] \\ r(k) \end{bmatrix} \tag{4.50}$$

We then obtain the following measurement update equation for the forward ML filter (FMLF) equations:

$$\begin{bmatrix} \hat{x}^{f,T}_{ML}[k|k] \\ \Sigma^f_x[k|k] \\ P^f_{x[k|k]} \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}^T \begin{bmatrix} \Sigma^f_z[k|k-1] & 0 & P_{z^f[k|k-1]} E_k \\ 0 & R_k & C_k \\ E_k^T P_{z^f[k|k-1]} & C_k^T & 0 \end{bmatrix}^{\#} \times \begin{bmatrix} \hat{z}^f_{ML}[k|k-1] & 0 & P_{z^f[k|k-1]} E_k \\ y(k) & 0 & C_k \\ 0 & -I & 0 \end{bmatrix} \tag{4.51}$$

Often it is the case that the covariance computations can be carried out in advance

of the acquisition of data. In that event (4.51) can be written as two equations given by

$$\begin{bmatrix} F^{f,T}(k) \\ G^{f,T}(k) \\ -\Sigma_x^f[k|k] \end{bmatrix} = \begin{bmatrix} \Sigma_z^f[k|k-1] & 0 & P_{z^f[k|k-1]}E_k \\ 0 & R_k & C_k \\ E_k^T P_{z^f[k|k-1]} & C_k^T & 0 \end{bmatrix}^{\#} \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix} \tag{4.52}$$

$$\hat{x}_{ML}^f[k|k] = F_k \hat{z}_{ML}^f[k|k-1] + G_k y(k) \tag{4.53}$$

$$P_{x[k|k]}^f = F_k P_{z^f[k|k-1]} E_k + G_k C_k \tag{4.54}$$

The constraints which are needed to determine the prediction step are given by

$$\begin{aligned} z^f(k+1) &= A_k x(k) + B_k u(k) \\ \hat{x}_{ML}^f[k|k] &= P_{x[k|k]}^f x(k) + \tilde{x}_{ML}^f[k|k] \end{aligned} \tag{4.55}$$

The prediction step is given by

$$\hat{z}_{ML}^f[k+1|k] = P_{z^f[k+1|k]} A_k \hat{x}_{ML}^f[k|k] \tag{4.56}$$

$$\Sigma_{z^f}[k+1|k] = P_{z^f[k+1|k]}(A_k \Sigma_{x^f}[k|k] A_k^T + B_k B_k^T) P_{z^f[k+1|k]} \tag{4.57}$$

$$\overline{P}_{z^f[k+1|k]} = (A_k \overline{P}_{x^f[k|k]} A_k^T)^{\#} (A_k \overline{P}_{x^f[k|k]} A_k^T) \tag{4.58}$$

where $P_{x^f[k|k]}$ indicates the symmetric projection matrix which defines the estimable part of $x(k)$ based on past data through time k, and $P_{z^f[k|k-1]}$ is the symmetric projection matrix which defines the estimable part of $z^f(k)$. Note again that part of this equation which corresponds to the portion of $z^f(k)$ which is inestimable, $\overline{P}_{z^f[k|k-1]} z^f(k)$ is not used in the forward estimation step. Equations (4.51) through (4.58) represent the generalization of [23] to allow for the possibility that $x(k)$ and/or $z^f(k)$ are not completely estimable. Also $\Sigma_x^f[k|k]$ can be thought of as the error covariance in the estimate of $\hat{x}_{ML}^f[k|k]$ in the sense of (2.31). $\Sigma_{x^f[k|k]}$ represents the corresponding error covariance for the estimable part of $x(k)$. The matrix $\Sigma_z^f[k+1|k]$

has a similar interpretation for $\hat{z}^f_{ML}[k+1|k]$. We are assured via the Lemma 4.1 that the ML estimates which are obtained at each step are precisely ML estimates based on all of the causal data.

Similarly we can define the backward ML filter (BMLF) where $\hat{x}^b_{ML}[l|k]$ denotes the ML estimate of $x(l)$ based on (4.42), for $k \leq \kappa \leq K$ and (4.47) for $k \leq \kappa \leq K$, and $\hat{z}^b_{ML}[l|k]$ denotes the ML estimate of $z^b(l)$ based on (4.42), for $k-1 \leq \kappa \leq K$ and (4.47) for $k \leq \kappa \leq K$. The observations required for the measurement step of the BMLF are

$$\begin{bmatrix} \hat{z}^b_{ML}[k|k+1] \\ y(k) \end{bmatrix} = \begin{bmatrix} P_{z^b[k|k+1]}A_k \\ C_k \end{bmatrix} x(k) + \begin{bmatrix} \tilde{z}^b_{ML}[k|k+1] \\ r(k) \end{bmatrix} \tag{4.59}$$

The BMLF is then given by

$$\begin{bmatrix} \hat{x}^{b,T}_{ML}[k|k] \\ \Sigma^b_x[k|k] \\ P^b_x[k|k] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}^T \begin{bmatrix} \Sigma^b_z[k|k+1] & 0 & P_{z^b[k|k+1]}A_k \\ 0 & R_k & C_k \\ A^T_k P_{z^b[k|k+1]} & C^T_k & 0 \end{bmatrix}^{\#} \times \tag{4.60}$$

$$\begin{bmatrix} \hat{z}^b_{ML}[k|k+1] & 0 & P_{z^b[k|k+1]}A_k \\ y(k) & 0 & C_k \\ 0 & -I & 0 \end{bmatrix}$$

Again the covariance computation can be separated from the equation governing the propagation of the estimates in the following fashion.

$$\begin{bmatrix} F^{b,T}_k \\ G^{b,T}_k \\ \Sigma^b_x[k|k] \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}^T \begin{bmatrix} \Sigma^b_z[k|k+1] & 0 & P_{z^b[k|k+1]}A_k \\ 0 & R_k & C_k \\ A^T_k P_{z^b[k|k+1]} & C^T_k & 0 \end{bmatrix}^{\#} \tag{4.61}$$

$$\hat{x}^{b,T}_{ML}[k|k] = F^b_k \hat{z}^b_{ML}[k|k+1] + G^b_k y(k) \tag{4.62}$$

$$\Sigma^b_x[k|k] = F^b_k P_{z^b[k|k+1]}A_k + G^b_k C_k \tag{4.63}$$

The measurements for the prediction step are given by

$$\begin{aligned} P_{z^b[k|k+1]}z^b(k) &= P_{z^b[k|k+1]}E_{k+1}x(k+1) - P_{z^b[k|k+1]}B_k u(k) \\ \hat{x}^b_{ML}[k|k] &= P^b_x[k|k]x(k) + \tilde{x}^b_{ML}[k|k] \end{aligned} \tag{4.64}$$

The prediction equations are therefore given by

$$\hat{z}^b_{ML}[k-1|k] = P_{z^b[k-1|k]}E_k\hat{x}^b_{ML}[k|k] \tag{4.65}$$

$$\Sigma_{z^b}[k-1|k] = P_{z^b[k-1|k]}(E_k\Sigma^b_x[k|k]E_k^T + B_{k-1}B_{k-1}^T)P_{z^b[k-1|k]} \tag{4.66}$$

$$\overline{P}_{z^b[k-1|k]} = (E_k\overline{P}_{x[k|k]}E_k^T)^{\#}(E_k\overline{P}_{x[k|k]}E_k^T) \tag{4.67}$$

The projection matrices which appear in these equations increase the complexity of the estimation equations. It is therefore important to know whether or not the propagation of the projection matrices need be carried out for the entire data length. From the standpoint of determining the estimable, parts of the vectors, it does not matter if the observation noises which determine the estimable subspace have full rank covariances, or are zero. The value of the data is irrelevant also. It can be set equal to zero, and the relationship between the estimable and non-estimable subspaces would not change. As a result estimability can be studied by considering the causal system given by

$$\begin{bmatrix} E_{k+1} \\ C_{k+1} \end{bmatrix} x(k+1) = \begin{bmatrix} A_k \\ 0 \end{bmatrix} x(t) \tag{4.68}$$

The state $x(k+1)$ is fully specified given $x(t)$ if the matrix in the first half of (4.68) has full column rank. This therefore is one assumption we place on the matrices E_k, and C_k. Furthermore there is a sequence of matrices L_k such that premultiplying (4.68) by L_{k+1} will yield

$$\begin{bmatrix} I \\ 0 \end{bmatrix} x(k+1) = \begin{bmatrix} \mathcal{A}_k \\ \mathcal{H}_k \end{bmatrix} x(t) \tag{4.69}$$

which is a causal system. If $\mathcal{A}_k$, $\mathcal{H}_k$ is uniformly completely observable, we know that

$x(t)$ is estimable in a finite number of steps. If there however are unobservable modes which are stable, $x(t)$ is still not estimable because knowledge that the mode decays is not sufficient to estimate it when the amplitude of the mode was completely unknown initially. We therefore require all modes which are not uniformly completely estimable to be uniformly nilpotent. As a result, though a mode may be completely unknown, it will be forced to zero with certainty in a finite period of time which is uniform over the entire interval. These modes are therefore estimable in a finite period of time. We call a system which satisfies these conditions, estimable. Estimability, is therefore a condition which is stronger than detectability, yet weaker than observability.

4.3.1 Computational Complexity of the Filtering Equations

The computation required for the FMLF and the BMLF are identical. As shown in Chapter 2, the amount of computation varies with how much of the state is estimable, and the existence of certain matrix inverses. The amount of required for one time step in the ML filter is denoted by $\mathcal{K}(n, m, p; description)$ where n is the dimension of the state, m is the dimension of the driving noise , p is the dimension of the observations, and 'description' represents any other relevant information. We may then define the following polynomial functions.

$\mathcal{K}(n, m, p; \text{off} - \text{line}, \text{measurement} - \text{update}, \text{non} - \text{estimable})$ represents the off-line flop count for the measurement update equations for the case where the state is not completely estimable. It represents the computation required for (4.52), (4.54) and constructing the product $P_{zf[k+1|k]}E_{k+1}$

$$\begin{aligned} &\mathcal{K}(n, m, p; \text{off} - \text{line}, \text{measurement} - \text{update}, \text{non} - \text{estimable}) \\ &= \mathcal{M}(2n + p, n, 2n + p) + 4n^3 + 2pn^2 \\ &= \tfrac{8}{3}(2n + p)^3 + 5n(2n + p)^2 + 4n^3 + 2pn^2 \end{aligned} \tag{4.70}$$

$\mathcal{K}(n, m, p; \text{off} - \text{line}, \text{prediction} - \text{update}, \text{non} - \text{estimable})$ represents the off-line flop count for the prediction update equations for the case where the state is not completely estimable. These equations are entirely off-line. If we plan to propagate

predicted estimates then by combining (4.53), and (4.56) together we must compute the quantities $P_{z^f[k+1|k]}A_kF_k$ and $P_{z^f[k+1|k]}A_kG_k$. If filtered estimates are being propagated then $F_{k+1}P_{z^f[k+1|k]}A_k$ need be computed. Computing predicted estimates requires more flops, and will be used as the basis for these computations.

$$\begin{aligned}
&\mathcal{K}(n,m,p;\text{off}-\text{line},\text{prediction}-\text{update},\text{non}-\text{estimable}) \\
&= \mathcal{E}(n,n,n) + 14n^3 + 4pn^2 + 4mn^2 \\
&= 18\tfrac{1}{3}n^3 + 4pn^2 + 4mn^2
\end{aligned} \tag{4.71}$$

$\mathcal{K}(n,m,p;\text{off}-\text{line},\text{measurement}-\text{update},\text{estimable})$ represents the off-line flop count for the measurement update equations for the case where the state is completely estimable.

$$\begin{aligned}
&\mathcal{K}(n,m,p;\text{off}-\text{line},\text{measurement}-\text{update},\text{estimable}) \\
&= \mathcal{E}(2n+p,n,2n+p) \\
&\tfrac{4}{3}(2n+p)^3 + 3n(2n+p)^2
\end{aligned} \tag{4.72}$$

$\mathcal{K}(n,m,p;\text{off}-\text{line},\text{prediction}-\text{update},\text{estimable})$ represents the off-line flop count for the prediction update equations for the case where the state is completely estimable.

$$\begin{aligned}
&\mathcal{K}(n,m,p;\text{off}-\text{line},\text{prediction}-\text{update},\text{estimable}) \\
&= 6n^3 + 2mn^2 + 2pn^2
\end{aligned} \tag{4.73}$$

$\mathcal{K}(n,m,p;\text{off}-\text{line},\text{measurement}-\text{update},\text{invertible})$ represents the off-line flop count for the measurement update equations for the case where the state is completely estimable and there are no perfect observations.

$$\begin{aligned}
&\mathcal{K}(n,m,p;\text{off}-\text{line},\text{measurement}-\text{update},\text{invertible}) \\
&= \mathcal{I}(2n+p,n) \\
&= \tfrac{2}{3}(2n+p)^3 + 2n(2n+p)^2
\end{aligned} \tag{4.74}$$

$\mathcal{K}(n,m,p;\text{off}-\text{line},\text{prediction}-\text{update},\text{invertible})$ represents the off-line flop count

for the prediction-update equations for the case where the state is completely estimable.

$$\begin{aligned}\mathcal{K}(n,m,p;\text{off}-\text{line},\text{prediction}-\text{update},\text{invertible}) \\ = 6n^3+2mn^2+2pn^2\end{aligned} \tag{4.75}$$

$\mathcal{K}(n,m,p;\text{off}-\text{line},\text{non}-\text{estimable})$ represents the total off-line flop count for the filter equations for the case where the state is not completely estimable.

$$\begin{aligned}&\mathcal{K}(n,m,p;\text{off}-\text{line},\text{non}-\text{estimable}) \\ &= 18.33n^3+4pn^2+4mn^2+2.67(2n+p)^3+5n(2n+p)^2+4n^3+2pn^2 \\ &= 63.67n^3+56n^2p+21np^2+2.67p^3+4n^2m\end{aligned} \tag{4.76}$$

$\mathcal{K}(n,m,p;\text{off}-\text{line},\text{estimable})$ represents the total off-line flop count for the filter equations for the case where the state is completely estimable.

$$\begin{aligned}&\mathcal{K}(n,m,p;\text{off}-\text{line},\text{estimable}) \\ &= \tfrac{4}{3}(2n+p)^3+3n(2n+p)^2+6n^3+2mn^2+2pn^2\end{aligned} \tag{4.77}$$

$\mathcal{K}(n,m,p;\text{off}-\text{line},\text{invertible})$ represents the total off-line flop count for the filter equations for the case where all required inverses are invertible.

$$\begin{aligned}&\mathcal{K}(n,m,p;\text{off}-\text{line},\text{invertible}) \\ &= \tfrac{2}{3}(2n+p)^3+2n(2n+p)^2+6n^3+2mn^2+2pn^2\end{aligned} \tag{4.78}$$

$\mathcal{K}(n,m,p;\text{on}-\text{line})$ represents the total on-line flop count for the filter equations. This count is valid whether the state is estimable or not. The on-line computations are given by combining (4.53), and (4.56) into the following

$$\hat{z}[k+1|k] = P_{z^f[k+1|k]}A_kF_k\hat{z}[k|k-1] + P_{z^f[k+1|k]}A_kG_ky(k) \tag{4.79}$$

The on-line flop count is given by

$$\begin{aligned} &\mathcal{K}(n,m,p;\text{on}-\text{line}) \\ &= 2n(n+p) \end{aligned} \tag{4.80}$$

4.3.2 The prediction equation

In a sense, $z^f(k)$, and $z^b(k)$ are non-essential variables, introduced in our estimation process, since, instead of performing one step predictions of these quantities, we could just as well perform one step forward and backward predictions for all of $x(k)$. Specifically, as opposed to the development in [22] in which $z^f(k)$, and $z^b(k)$ were introduced because of a desire to compute estimates for estimable quantities only, the whole point of our development is to relax this estimability condition. We can certainly apply this idea here avoiding the introduction of $z^f(k)$, and $z^b(k)$. We now describe this approach and in the process make clear why this is not a desirable alternative, as it is computationally more involved, and it does not yield in a direct way the estimable projection matrix for these one step predicted estimates which are explicitly needed in the subsequent processing steps.

The observation for the prediction step is given by

$$\begin{bmatrix} \hat{x}^f_{ML}[k|k] \\ 0 \end{bmatrix} = \begin{bmatrix} P^f_x[k|k] & 0 \\ A_k & -E_{k+1} \end{bmatrix} \begin{bmatrix} x(k) \\ x(k+1) \end{bmatrix} + \begin{bmatrix} \tilde{x}^f_{ML}[k|k-1] \\ Bu(k) \end{bmatrix} \tag{4.81}$$

After the elimination of the variable $x(k)$, (4.81) can be written more compactly as

$$P^f_z[k+1|k]A_k\hat{x}^f_{ML}[k|k] = P^f_z[k+1|k]E_{k+1}x(k+1) - P^f_z[k+1|k](A_k\tilde{x}^f_{ML}[k|k] + Bu(k)) \tag{4.82}$$

We then obtain the following prediction step (FMLF) equations:

$$\begin{bmatrix} \hat{x}^{f,T}_{ML}[k+1|k] \\ \Sigma^f_x[k+1|k] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ I \end{bmatrix}^T \begin{bmatrix} \Sigma^f_x[k|k] & 0 & P^f_x[k|k] & 0 \\ 0 & BB^T & A_k & -E_{k+1} \\ P^f_x[k|k] & A^T_k & 0 & 0 \\ 0 & -E^T(k+1) & 0 & 0 \end{bmatrix}^\# \begin{bmatrix} \hat{x}^f_{ML}[k|k] & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -I \end{bmatrix}$$

(4.83)

What remains is to compute the associated projection matrix $P^f[k+1|k]$. While we are able to compute the estimate and covariance, we should note that (4.81) should be viewed as a joint ML estimation problem for $x(k)$, as well as for $x(k+1)$. Thus applying the results of Chapter 2 to obtain the projection matrices, we obtain a projection matrix for the composite vector $[x^T(k), x^T(k+1)]$ The size of this projection matrix is $2n \times 2n$, rather than $n \times n$ and additional work is required to extract the proper projection matrix. In addition the pseudo-inverse of a larger matrix for $x(k+1)$ is required in (4.83) meaning that the introduction of the variables $z^f(k)$, and $z^b(k)$ is the proper step to reduce unnecessary computation.

One reason for the additional complexity in the approach described in this section is that if E_{k+1} is singular then the second equation in (4.81) does provide additional information about $x(k)$. By using $z^f(k+1)$, we explicitly eliminate $x(k)$ and focus attention only on the part of the dynamics related to the prediction of $x(k+1)$. A consequence of this is that in the context of smoothing we have to incorporate the part of the dynamics which we have discarded. We will see this explicitly in the Rauch-Tung-Striebel Algorithm described in Section 4.5.

4.4 The Mayne-Fraser Smoother

From Lemma 4.2 and Proposition 4.2 two independent measurements of the state $x(k)$ can be constructed. One is constructed from causal data and the other from anti-causal data. Furthermore, the structure of the measurements in (4.40) and (4.41) form block bidiagonal systems of equations, and thanks to Lemma 4.1 and Proposition 4.1 this structure can be exploited for the generation of efficient recursive estimation algorithms as discussed in Section 4.2. We have the choice of including the observation $y(k)$ with either causal or anti causal data. As a result when computing the smoothed estimate of the state, we have the choice of combining forward predicted with backwards filtered estimates, or forward filtered estimates with

backward predicted estimates. Other variation are possible. We could for example propagate forward and backward filtered estimates and combine the estimates $x^f(k)$ and $x^b(k+1)$ and use the intervening dynamic constraint to compute the smoothed estimate, similarly we can propagate forward and backward predicted estimates and include the local observation to generate the smoothed estimate of the state.

The observations required to compute the smoothed estimate from forward filtered and backward predicted estimates are given by

$$\begin{bmatrix} \hat{x}^f_{ML}[k|k] \\ \hat{z}^b_{ML}[k|k+1] \end{bmatrix} = \begin{bmatrix} P^f_{x[k|k]} \\ P_{z^b[k|k+1]}A_k \end{bmatrix} x(k) + \begin{bmatrix} \tilde{x}^f_{ML}[k|k] \\ \tilde{z}^b_{ML}[k|k+1] \end{bmatrix} \tag{4.84}$$

and the smoothed estimate is computed via

$$\hat{x}^s_{ML}(k) = L^f(k)\tilde{x}^f_{ML}[k|k] + L^b(k)\tilde{z}^b_{ML}[k|k+1] \tag{4.85}$$

$$P^s_{x(k)} = L^f(k)P^f_{x[k|k]} + L^b(k)P_{z^b[k|k+1]}A_k \tag{4.86}$$

where $L^f(k)$, $L^b(k)$, and the error covariance is given by

$$\begin{bmatrix} L^{f,T}(k) \\ L^{b,T}(k) \\ \Sigma^s_x(k) \end{bmatrix} = \begin{bmatrix} \Sigma^f_x[k|k] & 0 & P^f_x(k) \\ 0 & \Sigma_{z^b}[k|k+1] & P_{z^b}(k)A_k \\ P^f_x(k) & A^T_k P_{z^b}(k) & 0 \end{bmatrix}^{\#} \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix} \tag{4.87}$$

Similar equations can be written which involve combining forward predicted with backwards filtered estimates. The FMLF and the BMLF together with (4.85), (4.86), and (4.87) form a generalization of the Mayne-Fraser two filter formulas for optimal smoothing on STPBVDS's in the case where $x(k)$ may not be estimable, while portions of it are specified perfectly. Specifically if A=I and only final conditions are specified (making the system well-posed), the FMLF and the BMLF and (4.85)- (4.87) reduce to the usual Mayne-Fraser equations when applied to anti-causal systems. As a result, the generalization to STPBVDS deals in a symmetric way with information available at the two ends of the interval.

4.4.1 Computational Complexity of Mayne-Fraser Equations

Here we discuss the additional amount of computation required to compute the smoothed estimates in equations (4.85)- (4.87). The computation required for the forward and backward filters are discussed in Sections 4.3 and 4.3.1. To describe the amount of computation involved in combining forward and backward filtered estimates, we will define the following polynomials.

$\mathcal{F}(n; \text{off} - \text{line}, \text{non} - \text{estimable})$ will represent the amount of off-line computation associated with computing the covariance and projection for the smoothed estimate in (4.86), and (4.87), if the smoothed estimate is infact not completely estimable.

$$\begin{aligned} &\mathcal{F}(n; \text{off} - \text{line}, \text{non} - \text{estimable}) \\ &= \mathcal{M}(3n, n, 3n) + 4n^3 \\ &= 121n^3 \end{aligned} \tag{4.88}$$

$\mathcal{F}(n; \text{off} - \text{line}, \text{estimable})$ will represent the amount of off-line computation associated with computing the covariance and projection for the smoothed estimate, if the smoothed estimate is completely estimable.

$$\begin{aligned} &\mathcal{F}(n; \text{off} - \text{line}, \text{estimable}) \\ &= \mathcal{E}(3n, n, 3n) \\ &= 63n^3 \end{aligned} \tag{4.89}$$

$\mathcal{F}(n; \text{off} - \text{line}, \text{invertible})$ will represent the amount of off-line computation associated with computing the covariance and projection for the smoothed estimate, if the smoothed estimate is completely estimable and if the forward and backwards filtered estimates contain no redundant perfect information.

$$\begin{aligned} &\mathcal{F}(n; \text{off} - \text{line}, \text{invertible}) \\ &= \mathcal{I}(3n, n) \\ &= 36n^3 \end{aligned} \tag{4.90}$$

$\mathcal{F}(n; on - line)$ represents the flop count for constructing the smoothed estimates

from the forward and backward estimates in (4.85).

$$\mathcal{F}(n; on-line) = 4n^2 \tag{4.91}$$

4.5 Rauch-Tung-Striebel Algorithm

It is possible to generalize the Rauch-Tung-Striebel algorithm to STPBVDS's in the ML framework. Most derivations of the Rauch-Tung-Striebel algorithm arise from an algebraic point of view. This is difficult to do in the general ML frame work, and therefore we will use a statistical argument. This algorithm involves a forward sweep to compute $\hat{x}^f_{ML}[k|k]$ for each k producing the smoothed estimate $\hat{x}_s(K) = \hat{x}^f_{ML}[K|K]$ at one endpoint, which initiates a reverse sweep to compute $\hat{x}_s(k) = \hat{x}^f_{ML}[k|K]$ over the entire interval. The key to this backward sweep is again to interpret it as the computation of ML estimates based on an appropriate set of observations. In particular suppose that we have computed $\hat{x}_s(k+1)$ and its corresponding error covariance $\Sigma^s_x(k+1) = \Sigma^f_x[k+1|K]$, where $\Sigma_x[k+1|0,K]$ is interpreted as in (2.31) if $x(k)$ is not estimable. Then the computation of $\hat{x}_s(k)$ and $\Sigma^s_x(k+1)$ can be obtained by solving the following ML estimation problem which captures all of the relevant information relating $x(k)$ to $x(k+1)$ and the available estimates of each of these:

$$\begin{bmatrix} \hat{x}^f[k|k] \\ \hat{z}^f[k+1|k] \\ 0 \\ \hat{x}_s(k+1) \end{bmatrix} = \begin{bmatrix} P^f_{x[k|k]} & 0 \\ 0 & P_{z^f[k+1|k]}E_{k+1} \\ \overline{P}_{z^f[k+1|k]}A_k & \overline{P}_{z^f[k+1|k]}E_{k+1} \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ x(k+1) \end{bmatrix} + \begin{bmatrix} \tilde{x}^f[k|k] \\ \tilde{z}^f[k+1|k] \\ \overline{P}_{z^f[k+1|k]}Bu(k) \\ \tilde{x}_s(k+1) \end{bmatrix} \tag{4.92}$$

The first and second measurements in (4.92) are the the filtered and predicted estimates which result from forward filtering. As a result their observation noises are correlated. The third observation in equation (4.92) is a dynamic constraint which lies in the null space of P_{zf}. It therefore was not used in the filtering process as it corresponds to the part of the dynamics (4.50)? not used in the one step prediction of $x(k+1)$. For the smoothing problem, this dynamic constraint must be included

so that all information is accounted for.

Consider the Mayne-Fraser two filter smoother. We know that the smoothed estimate of $x(k)$ and $x(k+1)$ can be constructed from the following measurements.

$$\hat{x}^f[k|k] = P^f_{x[k|k]}x(k) + \tilde{x}^f[k|k] \tag{4.93}$$

$$\hat{x}^b[k|k] = P^b_{x[k|k]}x(k) + \tilde{b}^f[k|k] \tag{4.94}$$

$$0 = E_{k+1}x(k+1) - A_k x(k) - B_k u(k) \tag{4.95}$$

This is verified by noting that (4.93), and (4.95) are the measurements required to construct the predicted estimate of $x(k+1)$ which is independent of the backward filtered estimate (4.94). A similar argument can be used to construct the independent measurements required for the smoothed estimate of $x(k)$. Lemma 4.1 allows for the incorporation of measurements in any order. First we replace (4.93), (4.94), and (4.95) by the following set of measurements

$$\hat{x}^f[k|k] = P^f_{x[k|k]}x(k) + \tilde{x}^f[k|k] \tag{4.96}$$

$$0 = P^f_{z[k+1|k]}[E_{k+1}x(k+1) - A_k x(k) - B_k u(k)] \tag{4.97}$$

$$0 = \overline{P}^f_{z[k+1|k]}[E_{k+1}x(k+1) - A_k x(k) - B_k u(k)] \tag{4.98}$$

$$\hat{x}^b[k+1|k+1] = P^b_{x[k+1|k+1]}x(k+1) + \tilde{x}^b[k+1|k+1] \tag{4.99}$$

where we have taken equation (4.95) and premultiplied it by $P_{z^f[k+1|k]}$, and $\overline{P}_{z^f[k+1|k]}$ in (4.97), and (4.98) respectively. Since Lemma 4.1 allows us to combine these equations in any order, we replace (4.96)- (4.99) by an equivalent set of measurements.

$$\hat{x}^f[k|k] = P^f_{x[k|k]}x(k) + \tilde{x}^f[k|k] \tag{4.100}$$

$$\hat{z}^f[k+1|k] = P^f_{z[k+1|k]}E_{k+1}x(k+1) + \tilde{z}^f[k+1|k] \tag{4.101}$$

$$0 = \overline{P}^f_{z[k+1|k]}[E_{k+1}x(k+1) - A_k x(k) - B_k u(k)] \tag{4.102}$$

$$\hat{x}^b[k+1|k+1] = P^b_{x[k+1|k+1]}x(k+1) + \tilde{x}^b[k+1|k+1] \tag{4.103}$$

where the measurement in (4.101) is constructed from the measurements in (4.96),

and (4.97). Note that from the development of the Mayne Fraser algorithm, we know that (4.101), and (4.103) can be combined to recover the smoothed estimate of $x(k+1)$. Thus equations (4.100)-(4.103) are equivalent to the set of measurements indicated by (4.92). We will return to this shortly, but let us now continue with the transformation of (4.100)-(4.103). Specifically the set of measurements in (4.100)-(4.103), being equivalent to (4.93)-(4.94), are sufficient to construct the smoothed estimate of $x(k)$. The errors in the observations in (4.100)-(4.103) given by $\tilde{x}^{f,T}[k|k]$, $\tilde{z}^{f,T}[k+1|k]$, $\overline{P}^T_{z^f[k+1|k]}B_k u(k)$, and $\tilde{x}^{b,T}[k+1|k+1]$, have a joint error covariance given by

$$\begin{bmatrix} \Sigma_x[k|0,k] & \Sigma_x[k|0,k]A_k^T P^f_{z[k+1|k]} & 0 & 0 \\ P^f_{z[k+1|k]}A_k\Sigma_x[k|0,k] & \Sigma^f_z[k+1|0,k] & P^f_{z[k+1|k]}B_k B_k^T C\overline{P}^f_{z[k+1|k]} & 0 \\ 0 & \overline{P}^f_{z[k+1|k]}B_k B_k^T P^f_{z[k+1|k]} & \overline{P}^f_{z[k+1|k]}B_k B_k^T \overline{P}^f_{z[k+1|k]} & 0 \\ 0 & 0 & 0 & \Sigma_x[k+1|k+1,K] \end{bmatrix} \tag{4.104}$$

Clearly we can write (4.100)-(4.103) as one measurement in the form $y = Hx + v$ and compute the smoothed estimate $\hat{x}^s(k)$ given all of the measurements. However we can reduce the amount of computation if we note that (4.100)-(4.103) can be written as two measurements given by the smoothed estimate of $x(k+1)$ and a measurement with measurement noise independent of $\tilde{x}^s(k+1)$. Specifically, we note from the covariance in (4.104) that $\tilde{x}^b[k+1|k+1]$ is independent of all other measurements. Since $\hat{x}^b[k+1|k+1]$ and $\hat{z}^f[k+1|k]$ can be combined to yield $\hat{x}^s(k+1)$, we thus seek to find from the remaining three measurements (4.100)-(4.102) that part whose measurement noise is orthogonal to $\hat{z}^f[k+1|k]$. The joint covariance of $\tilde{x}^{f,T}[k|k]$, $\overline{P}^T_{z^f[k+1|k]}B_k u(k)$, and $\tilde{z}^{f,T}[k+1|k]$, is given by

$$\begin{bmatrix} \Sigma_x[k|0,k] & 0 & \Sigma_x[k|0,k]A_k^T P^f_{z[k+1|k]} \\ 0 & \overline{P}^f_{z[k+1|k]}B_k B_k^T \overline{P}^f_{z[k+1|k]} & \overline{P}^f_{z[k+1|k]}B_k B_k^T P^f_{z[k+1|k]} \\ P^f_{z[k+1|k]}A_k\Sigma_x[k|0,k] & P^f_{z[k+1|k]}B_k B_k^T \overline{P}^f_{z[k+1|k]} & \Sigma^f_z[k+1|0,k] \end{bmatrix} \tag{4.105}$$

If we define an invertible matrix T_k by

$$T_k = \begin{bmatrix} I & 0 & -\Sigma_x[k|0,k]A_k^T\Sigma_z^{\#}[k+1|k] \\ 0 & I & -\overline{P}^f_{z[k+1|k]}B_kB_k^T\Sigma_z^{\#}[k+1|k] \\ 0 & 0 & I \end{bmatrix} \tag{4.106}$$

then, if (4.105) is premultiplied by T_k, and post-multiplied by T_k^T, we obtain

$$\begin{bmatrix} \Sigma_x[k|k][I - A_k^T\Sigma^{\#}_{z^f}[k+1|k]A_k\Sigma_x[k|k]] & -\Sigma_x[k|k]A_k^T\Sigma^{\#}_{z^f}[k+1|k]B_kB_k^T\overline{P}_{z^f[k+1|k]} & 0 \\ -\overline{P}_{z^f[k+1|k]}B_kB_k^T\Sigma^{\#}_{z^f}[k+1|k]A_k\Sigma_x[k|k]A_k^T & \overline{P}_{z^f[k+1|k]}B_kB_k^T[I - \Sigma^{\#}_{z^f}[k+1|k]B_kB_k^T]\overline{P}_{z^f[k+1|k]} & 0 \\ 0 & 0 & \Sigma^f_z[k+1|k] \end{bmatrix} \tag{4.107}$$

As a result the matrix T_k can be used to obtain two independent measurements from (4.100)- (4.98). The first is given by

$$\begin{bmatrix} \hat{x}^f_{ML}[k|k] - \{\Sigma_x[k|k]A_k^T\Sigma^{\#}_{z^f}[k+1|k]\}\hat{z}^f_{ML}[k+1|k] \\ \{\overline{P}_{z^f[k+1|k]}B_kB_k^T\Sigma^{\#}_{z^f}[k+1|k]\}\hat{z}^f_{ML}[k+1|k] \end{bmatrix} = \tag{4.108}$$

$$\begin{bmatrix} P^f_{x[k|k]} & -\Sigma_x[k|k]A_k^T\Sigma^{\#}_{z^f}[k+1|k]E_{k+1} \\ -\overline{P}_{z^f[k+1|k]}A_k & \overline{P}_{z^f[k+1|k]}(I - B_kB_k^T\Sigma^{\#}_{z^f[k+1|k]})E_{k+1} \end{bmatrix} \begin{bmatrix} x(k) \\ x(k+1) \end{bmatrix} + \nu(k)$$

The second is given by

$$\hat{z}^f[k+1|k] \;=\; P_{z^f[k+1|k]}E_{k+1}x(k+1) + \tilde{z}^f_{ML}[k+1|k] \tag{4.109}$$

The estimate $\hat{z}^f[k+1|k]$ may be combined with $\hat{x}^b[k+1|k+1]$ to obtain the smoothed estimate $\hat{x}^s(k+1)$ whose error $\tilde{x}^s(k+1)$ is independent of $\nu(k)$. The error covariance for $\nu(k)$ is given by

$$\begin{bmatrix} \Sigma_x[k|k][I - A_k^T\Sigma^{\#}_{z^f}[k+1|k]A_k\Sigma_x[k|k]] & -\Sigma_x[k|k]A_k^T\Sigma^{\#}_{z^f}[k+1|k]B_kB_k^T\overline{P}_{z^f[k+1|k]} \\ -\overline{P}_{z^f[k+1|k]}B_kB_k^T\Sigma^{\#}_{z^f}[k+1|k]A_k\Sigma_x[k|k]A_k^T & \overline{P}_{z^f[k+1|k]}B_kB_k^T[I - \Sigma^{\#}_{z^f}[k+1|k]B_kB_k^T]\overline{P}_{z^f[k+1|k]} \end{bmatrix} \tag{4.110}$$

which is the upper left most block of (4.107). The computation of the smoothed

estimate $x(k)$ is specified from the following measurements.

$$P_k^{rts}\Theta_k\hat{x}[k|k] - P_k^{rts}\Xi_k\hat{x}^s(k+1) = P_k^{rts}\Lambda_k x(k) + P_k^{rts}\Xi_k\tilde{x}^s(k+1) + P_k^{rts}\nu(k) \tag{4.111}$$

where Θ_k, Ξ_k, and Λ_k are given by

$$\Theta_k = \begin{bmatrix} I - \Sigma_x[k|k]A_k^T\Sigma_{zf}^{\#}[k+1|k]A_k \\ \overline{P}_{zf[k+1|k]}B_kB_k^T\Sigma_{zf}^{\#}[k+1|k]A_k \end{bmatrix} \tag{4.112}$$

$$\Lambda_k = \begin{bmatrix} P_{x[k|k]}^f \\ -\overline{P}_{zf[k+1|k]}A_k \end{bmatrix} \tag{4.113}$$

$$\Xi_k = \begin{bmatrix} -\Sigma_x[k|k]A_k^T\Sigma_{zf}^{\#}[k+1|k]E_{k+1} \\ \overline{P}_{zf[k+1|k]}(I - B_kB_k^T\Sigma_{zf[k+1|k]}^{\#})E_{k+1} \end{bmatrix} \tag{4.114}$$

$$\overline{P}_k^{rts} = [\Xi_k\overline{P}_{k+1}^s\Xi_k^T]^{\#}[\Xi_k\overline{P}_{k+1}^s\Xi_k^T] \tag{4.115}$$

The equation for the estimate is given by

$$\hat{x}_{ML}^s(k) = L_{rts}(k)[\Theta_k\hat{x}[k|k] - \Xi_k\hat{x}^s(k+1)] \tag{4.116}$$

where $L_{rts}(k)$ and the error covariance for the smoothed estimate, $\Sigma_x^s(k)$, are given by

$$\begin{bmatrix} L_{rts}^T(k) \\ \Sigma_x^s(k) \end{bmatrix} = \begin{bmatrix} P_k^{rts}[\Xi_k\Sigma_x^s(k+1)\Xi_k^T + Cov(\nu(\tilde{k}))]P_k^{rts} & P_k^{rts}\Lambda_k \\ \Lambda_k^T P_k^{rts} & 0 \end{bmatrix}^{\#} \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{4.117}$$

$$P_k^s = L_{rts}(k)P_k^{rts}\Lambda_k \tag{4.118}$$

4.5.1 Computational Complexity of the Backward Sweep

Here we discuss the computational complexity of the backward sweep of the Rauch-Tung-Striebel algorithm. The equations as presented are complex for two major reasons. One reason is that the the smoothed state may not be estimable. The other reason is that the state may not be estimable from causal data alone. We will provide

flop counts for the general backward sweep, for the case where the state is estimable based on all of the data, and the case where the state is estimable based on causal data alone. To describe these flop counts we introduce the following functions.

$\mathcal{T}(n; \text{off} - \text{line}, \text{non} - \text{estimable})$ represents the off-line computation required to compute the smoothed estimate assuming that the state is not estimable.

$$\begin{aligned} &\mathcal{T}(n; \text{off} - \text{line}, \text{non} - \text{estimable}) \\ &= 82n^3 + 4n^2m + \mathcal{M}(n, 2n, n) + \mathcal{M}(3n, n, 2n) + \mathcal{E}(2n, 2n, n) \\ &= 223.33n^3 + 4n^2m \end{aligned} \tag{4.119}$$

$\mathcal{T}(n; \text{off} - \text{line}, \text{non} - \text{causally} - \text{estimable})$ represents the off-line computation required to compute the smoothed estimate assuming that the state is estimable, but is not estimable based on causal data. Computational savings comes from not having to compute the projection matrix in (4.115), and forming products with this projection matrix and the quantities in (4.112)- (4.114). In addition (4.118) will not have to be computed.

$$\begin{aligned} &\mathcal{T}(n; \text{off} - \text{line}, \text{non} - \text{causally} - \text{estimable}) \\ &= 42n^3 + 4n^2m + \mathcal{M}(n, 2n, n) + \mathcal{E}(3n, n, 2n) \\ &= 101.33n^3 + 4n^2m \end{aligned} \tag{4.120}$$

$\mathcal{T}(n; \text{off} - \text{line}, \text{causally} - \text{estimable})$ represents the off-line computation required to compute the smoothed estimate assuming that the state is estimable during the filtering step. Computational savings comes from the fact that the quantities in (4.112)-(4.114) are of reduced dimension because $\overline{P}_{zf[k+1|k]} = 0$. In addition the pseudo-inverse in (4.117) will not have to be computed. Since the state is estimable, $\Lambda_k = I$. As a result $L_{rts} = I$. The smoothed covariance is given by

$$Cov(\tilde{x}^s(k)) = \Xi_k Cov(\tilde{x}^s(k+1))\Xi_k^T + Cov(\nu(k)) \tag{4.121}$$

$$\begin{aligned}
&\mathcal{T}(n; \text{off} - \text{line}, \text{causally} - \text{estimable}) \\
&= 12n^3 + \mathcal{M}(n,n,n) \\
&= 19\tfrac{2}{3}n^3
\end{aligned} \tag{4.122}$$

$\mathcal{T}(n; \text{off} - \text{line}, \text{invertible})$ represents the off-line computation required to compute the smoothed estimate assuming that the state is estimable during the filtering step and has a full rank error covariance. Computational savings comes from the fact that the quantities in (4.112)- (4.114) are of reduced dimension because $\overline{P}_{zf[k+1|k]} = 0$. In addition the pseudo-inverse (4.117) and (4.106) will not have to be computed. Since the state is estimable, $\Lambda_k = I$. As a result $L_{rts} = I$. The smoothed covariance is given by

$$Cov(\tilde{x}^s(k) = \Xi_k Cov(\tilde{x}^s(k+1))\Xi^T + Cov(\tilde{\nu(k)}) \tag{4.123}$$

$$\begin{aligned}
&\mathcal{T}(n; \text{off} - \text{line}, \text{invertible}) \\
&= 12n^3 + \mathcal{I}(n,n) \\
&= 14\tfrac{2}{3}n^3
\end{aligned} \tag{4.124}$$

$\mathcal{T}(n; \text{on} - \text{line})$ represents the on-line computation required to compute the smoothed estimate in (4.116) regardless of the estimability of the step.

$$\mathcal{T}(n; \text{on} - \text{line}) = 4n^2 \tag{4.125}$$

4.5.2 Inward-Outward filtering of TPBVDS's

The general problem with the smoothing of TPBVDS's is that the boundary conditions cannot be matched to implement filters whose individual computations can be interpreted as producing estimates of the state. In Section 3.1 it was shown that any TPBVDS can be described by a STPBVDS where the state is constructed by augmenting $x(k)$ with $x(-k)$. This suggests the notion of filtering outward to the boundaries of the system, and inward towards the center. Given that the system is separable, no new equations need be written since they follow immediately by sub-

stituting the model (3.25) for (3.8) and applying the filtering algorithm in Section 4.2.

4.6 Square Root Smoothing Algorithm

4.6.1 Introduction

The new smoothing algorithms presented in this thesis are based on the ML estimation philosophy. Specifically, all of the algorithms presented consist of the solution of a sequence of ML estimation problems. In addition these algorithms can all be cast in the square-root framework. To illustrate this, a square root algorithm for the FMLF algorithm. Square-root algorithms for the Mayne-Fraser and the Rauch-Tung-Striebel can be similarly derived. The square-root FMLF algorithm essentially computes a reduced order observer [32] for the case in which perfect measurements are available. It therefore shares the flavor of the algorithms [6] and [27]. However, our algorithm does not rely on the invertibility of the A_k matrices as does the algorithm in [6].

4.6.2 Square Root Forward Maximum Likelihood Filter

Here we consider the FMLF separated into measurement update steps and time update steps. The time update step is given by the following equations. The vector $z^f(k+1)$ to be estimated satisfies

$$z^f(k+1) = A_k x(k) + B_k u(k) \tag{4.126}$$

We assume that an estimate of $x(k)$ based on Y_k exists and that we have available an orthogonal matrix U_k with the following partitioning

$$U_k = \begin{bmatrix} U_{p,k} \\ U_{s,k} \\ U_{u,k} \end{bmatrix} \tag{4.127}$$

such that, based on Y_k, $U_{p,k}x(k)$ is known perfectly, the ML estimate based on Y_k exists for $U_{s,k}x(k)$, with an invertible error covariance, and $U_{u,k}x(k)$ is completely unknown. The orthogonal matrix which separates the relevant subspaces of R^n to estimate $z^f[k+1|k]$ is given by solving for the matrix, Q_{k+1} which upper triangularizes the following matrix.

$$Q_{k+1}\left[\begin{array}{ccc} A_kU_{u,k}^T & \{A_kU_{s,k}^T\vdots B_k\} & AU_{p,k}^T \end{array}\right] = \Gamma_k \tag{4.128}$$

The matrix Q_k is partitioned as follows.

$$Q_k = \left[\begin{array}{c} Q_{u,k} \\ Q_{s,k} \\ Q_{p,k} \end{array}\right] \tag{4.129}$$

The following equations then determine $z^f(k+1)$ in the new coordinate system.

$$\left[\begin{array}{c} Q_{p,k+1}z^f(k+1) \\ Q_{s,k+1}z^f(k+1) \end{array}\right] = \left[\begin{array}{cc} Q_{p,k+1}A_kU_{p,k}^T & 0 \\ Q_{s,k+1}A_kU_{p,k}^T\{U_{p,k}x(k)\} & Q_{s,k+1}\left[\begin{array}{cc} A_kU_{s,k}^T & B_k \end{array}\right] \end{array}\right] \left[\begin{array}{c} \{U_{p,k}x(k)\} \\ \{U_{s,k}x(k)\} \\ u(k) \end{array}\right] \tag{4.130}$$

Note that in equation (4.130) the desired result is obtained through direct multiplication, not solving for the solution of simultaneous algebraic equations. The products $Q_{k+1}A_kU_{p,k}$, and $Q_{k+1}A[U_{s,k}\vdots B]$ have already been calculated in the triangularization process in equation (4.128). The predicted estimate of $Q_{p,k+1}z(k+1)$ is given by

$$Q_{p,k+1}\hat{z}^f[k+1|k] = Q_{p,k+1}A_kU_{p,k}^T\{U_{p,k}\hat{x}[k|k]\} \tag{4.131}$$

The predicted estimate of the vector $Q_{s,k+1}z^f(k+1)$ is obtained by noting that it satisfies

$$Q_{s,k+1}z^f(k+1) = \quad Q_{s,k+1}A_kU_{p,k}^T\{U_{p,k}x(k)\} + \tag{4.132}$$

$$Q_{s,k+1}\left[\begin{array}{cc} A_k U_{s,k}^T \Sigma_{x,o}^{f,1/2}[k|k] & B_k \end{array}\right] S_k \times S_k^T \left[\begin{array}{c} \{\Sigma_{x,o}^{f,-1/2}[k|k] U_{s,k}\hat{x}(k)\} \\ \hat{u}(k) \end{array}\right] +$$

$$Q_{s,k+1}\left[\begin{array}{cc} A_k U_{s,k}^T \Sigma_{x,o}^{f,1/2}[k|0,k] & B_k \end{array}\right] S_k \times S_k^T \left[\begin{array}{c} \{\Sigma_{x,o}^{f,-1/2}[k|k] U_{s,k}\tilde{x}(k)\} \\ \tilde{u}(k) \end{array}\right]$$

where the orthogonal matrix S_k^T which premultiplies the noise term is chosen to lower triangularize the matrix $Q_{s,k+1}[A_k U_{s,k}^T \Sigma_{x,o}^{f,1/2}[k|k] : B_k] S_k$. The square root of the error covariance is the square and lower triangular portion of the matrix

$$Q_{s,k+1}[A_k U_{s,k}^T \Sigma_{x,o}^{f,1/2}[k|k] : B_k] S_k = [\Sigma_{zf,o}^{1/2}[k+1|k] : 0] \tag{4.133}$$

The estimate of $Q_{s,k+1} z^f(k+1)$, which is given by $\hat{z}^f[k+1|k]$, is obtained by removing the noise terms from (4.132) and setting $\hat{u}(k) = 0$.

The next step is to compute the measurement update step in the FMLF. The 'measurements' required to produce the ML estimate $x(k)$ given data up to and including time k is given by the following

$$\hat{z}^f[k+1|k] = E_{k+1} x(k+1) + \tilde{z}^f[k+1|k] \tag{4.134}$$

$$y(k+1) = C_{k+1} x(k+1) + v(k+1) \tag{4.135}$$

This problem is in the form $y = Hx + v$ used in the discussion in Chapter 2. Assume that there may be perfect information in the observation vector $y(k)$. Let L_k be an invertible matrix which is partitioned as follows.

$$L_k = \left[\begin{array}{c} L_{p,k} \\ L_{s,k} \end{array}\right] \tag{4.136}$$

We will assume that we are given L_k with the data, and that $L_{p,k} y(k)$ is known perfectly, and $L_{s,k} y(k)$ has an invertible covariance $R_{0,k}$. If we combine equation (4.134) and (4.135) into a larger single vector equation for $[\hat{z}^{f,T}[k+1|k],\ y^T(k+1)]$, we then

premultiply it by the following matrix

$$\mathcal{Q}_{k+1} = \begin{bmatrix} \mathcal{Q}_{p,k+1} \\ \mathcal{Q}_{s,k+1} \\ \mathcal{Q}_{u,k+1} \end{bmatrix} = \begin{bmatrix} Q_{p,k+1} & 0 \\ 0 & L_{p,k+1} \\ Q_{s,k+1} & 0 \\ 0 & L_{s,k+1} \\ Q_{u,k+1} & 0 \end{bmatrix} \tag{4.137}$$

Following the development of square-root ML estimation in Section 2.2 the matrix U_{k+1}, which is constructed to lower triangularize the product

$$\mathcal{Q}_{k+1} \begin{bmatrix} E_{k+1} \\ C_{k+1} \end{bmatrix} U_{k+1} \tag{4.138}$$

where U_{k+1} is partitioned as in (4.127). The part of the state $x(k+1)$ which can be determined perfectly is obtained by considering the following 'measurement'.

$$\begin{bmatrix} Q_{p,k+1} z^f(k+1) \\ L_{p,k+1} y(k+1) \end{bmatrix} = \begin{bmatrix} Q_{p,k+1} E_{k+1} U_{p,k+1}^T \\ L_{p,k+1} C_{k+1} U_{p,k+1}^T \end{bmatrix} \{U_{p,k+1} x(k+1)\} \tag{4.139}$$

where $U_{p,k+1}x(k+1)$ is the part of $x(k+1)$ which can be estimated perfectly. Equation (4.139) can be uniquely solved to determine the vector $\{U_{p,k+1}x(k+1)\}$ which is given by

$$\begin{bmatrix} Q_{p,k+1} E_{k+1} U_{p,k+1}^T \\ L_{p,k+1} C_{k+1} U_{p,k+1}^T \end{bmatrix}^{-L} \begin{bmatrix} Q_{p,k+1} z^f(k+1) \\ L_{p,k+1} y(k+1) \end{bmatrix} = \{U_{p,k+1} x(k+1)\} \tag{4.140}$$

The part of the state with an invertible covariance is obtained as the solution of the

ML estimation problem where the observation noise has been normalized.

$$\begin{bmatrix} \Sigma_{z^f,o}^{-1/2}[k+1|k]Q_{s,k+1}z^f(k+1) \\ R_{o,k+1}^{-1/2}L_{s,k+1}y(k+1) \end{bmatrix} = \begin{bmatrix} \Sigma_{z^f,o}^{-1/2}[k+1|k]Q_{s,k+1}E_{k+1} \\ R_{o,k+1}^{-1/2}L_{s,k+1}C_{k+1} \end{bmatrix} x(k+1) + \begin{bmatrix} \Sigma_{z^f,o}^{-1/2}[k+1|k]Q_{s,k+1}\tilde{z}^f(k+1) \\ R_{o,k+1}^{-1/2}L_{s,k+1}r(k+1) \end{bmatrix} \tag{4.141}$$

As in (2.96), the part of the estimate which has been determined perfectly is removed resulting in a 'measurement' for the part of x for which we can assign a full rank covariance. What results is the following 'measurement' for a 'standard' square-root problem.

$$J_{k+1}\left[\begin{bmatrix} \Sigma_{z^f,o}^{-1/2}[k+1|0,k]Q_{s,k+1}\hat{z}^f(k+1) \\ R_{o,k+1}^{-1/2}L_{s,k+1}y(k+1) \end{bmatrix} - \begin{bmatrix} \Sigma_{z^f,o}^{-1/2}[k+1|0,k]Q_{s,k+1}E_{k+1}U_{p,k+1}^T \\ R_{o,k+1}^{-1/2}L_{s,k+1}C_{k+1}U_{p,k+1}^T \end{bmatrix} U_{p,k+1}x(k+1)\right]$$
$$= J_{k+1}\left[\begin{bmatrix} \Sigma_{z^f,o}^{-1/2}[k+1|0,k]Q_{s,k+1}E_{k+1}U_{s,k+1}^T \\ R_{o,k+1}^{-1/2}L_{s,k+1}C_{k+1}U_{s,k+1}^T \end{bmatrix} U_{s,k+1}x(k+1) + \begin{bmatrix} \Sigma_{z^f,o}^{-1/2}[k+1|0,k]Q_{s,k+1}\tilde{z}^f(k+1) \\ R_{o,k+1}^{-1/2}L_{s,k+1}r(k+1) \end{bmatrix}\right] \tag{4.142}$$

where the orthogonal matrix J_{k+1} is determined via the QR factorization to upper triangularize the the following matrix

$$J_{k+1}\begin{bmatrix} \Sigma_{z^f,o}^{-1/2}[k+1|0,k]Q_{s,k}E_{k+1}U_{s,k+1} \\ R_{o,k+1}^{-1/2}L_{s,k+1}C_{k+1}U_{s,k+1} \end{bmatrix} = \begin{bmatrix} \Sigma_{x,o}^{f,-1/2}[k+1|k+1] \\ 0 \end{bmatrix} \tag{4.143}$$

The matrix U_k represents a coordinate transformation which can be used to return the estimate of the state to the coordinate system in which the state was originally defined. In addition the covariance matrix $\Sigma_{x,o}[k+1|k+1]$ can be placed in the original coordinate system to yield $\Sigma_x^f[k+1|k+1]$.

Chapter 5

Parallel Smoothing for One-Dimensional Systems

Our aim in this chapter is to find parallel algorithms for smoothing TPBVDS's given by

$$\begin{aligned} E_{k+1}x(k+1) &= A_k x(k) + B_k u(k) \\ E_0 x(0) &= A_K x(K) + B_K u(K) \end{aligned} \tag{5.1}$$

based on observations given by

$$y(k) = C_k x(k) + v(k) \tag{5.2}$$

where $u(k)$, and $v(k)$ are independent white noise sequences where

$$\begin{aligned} u(k) &\sim N(0; I) \\ v(k) &\sim N(0; R_k) \end{aligned} \tag{5.3}$$

In this chapter we consider algorithms for one-dimensional processes where the data is spatially partitioned in order to limit the amount of data a processor would have to access, and to limit the complexity of the local processing. The data is partitioned along the time axis into intervals, where one processor is assigned to one interval of data. A given processor operates on local data, and also communicates with other processors operating on other intervals of data. The problem is divided

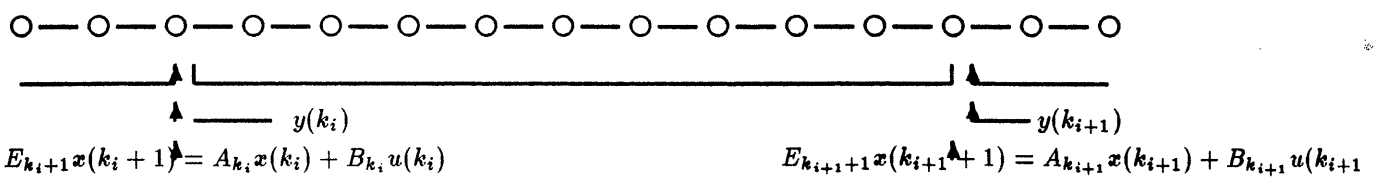


Figure 5-1: partitioning of the data among processors

evenly among L processors. We assume further that the i^{th} processor has access to $y(k_{i-1}+1)$ through $y(k_i)$ and the dynamic constraints which link the states $x(k_{i-1})$ through $x(k_i)$. See Figure 5-1. The value of k_i in these algorithms are given by $i\frac{K}{L}$, where $\frac{K}{L}$ is an odd integer.

The parallel processing algorithms which use this data partitioning typically have the following steps. During the first step, each processor operates in parallel on local data, and computes certain sufficient statistics which contain information needed by other processors to compute smoothed or filtered estimates. In the second step, there is an interprocessor exchange step, where sufficient statistics are exchanged between processors, processed and exchanged again, until each processor has information necessary to compute globally filtered or smoothed estimates without further communication. In the final step, each processor computes globally smoothed or filtered estimates in parallel based on local data, and the sufficient statistics obtained by other processors.

In the first local processing step, many methods of processing the data are possible. For example, in the algorithm of Morf et al.[14], which produces filtered estimates for causal systems, the local processing takes the form of two filters. The first filter is a Kalman filter where the state is locally initialized with zero initial condition with a covariance which is also set to zero. The second filter is a backward information filter over the same subinterval which computes the equivalent of ML estimates based on local, but 'future' data.

Another example of local filtering is provided by Tewfik, et al[15]. in which a model which is radially causal is filtered locally from the center of each subinterval to the boundary of the respective subinterval. A system is called radially causal, when

the system with the state taken as $[x(t),\ x(-t)]$ is causal in t.

The algorithms of Morf et al., and Tewfik et al. compute estimates locally which are correlated with the estimates which are computed by processors operating over other regions. Furthermore a correction step is needed to deal with local assumptions made about the boundaries of the individual subregions in both algorithms, and the repetition of a priori information in each subregion. In these algorithms this is performed during the interprocessor exchange step.

Borrowing from the literature on solving partial differential equations (pde's), the domain decomposition approach [9] offers new insight into the smoothing problem. Domain decomposition algorithms are iterative procedures for solving pde's in which iterations are performed to match boundary conditions of the local solutions with those of neighboring subregions. These reduced problems, focusing on boundary matching, have less structure than the local subproblems. The algorithm iterates between solving the local subproblems and matching the boundary conditions until it converges to a solution. Similarly, locally smoothed estimates can be computed in each subregion instead of locally filtered estimates. The problem at the boundary can be solved to provide the local subregions with the boundary information needed to update locally smoothed estimates to produce globally smoothed estimates. A smoothing domain decomposition algorithm is provided in Section 5.3.

Another novel example of local processing is one we refer to as the oblique projection approach. Each processor produces forward and backward filtered estimates, based on the assumption that boundary measurements will be available from neighboring processors. As a result local processors do not compute optimal estimates based on their own data in an effort to make the incorporation of boundary information easier.

Finally, we consider for the initial processing step the use of using filters which compute ML estimates based on local data. The result is that all of the estimates produced in each subregion are statistically independent. No assumptions are made about the nature of the boundary of each subregion, other than that which is supported by local data, and furthermore, prior information is treated locally where it is

defined, and in the same manner as all other 'noisy constraints'.

Continuing with the second step in these parallel processing algorithms we consider the interprocessor exchange step. The purpose of this step is to provide all of the information required to compute globally filtered or smoothed estimates of the boundaries of each subregion. The typical way of achieving this is to perform a forward and, if smoothed estimates are to be obtained, backward filtering operation on the boundaries of the subregions using boundary statistics and estimates computed locally in each subregion. The details of the filtering process is largely determined by the first local processing step. Such a forward-backward filtering process implies that there is a total ordering to the computations used in the interprocessor communication step. This total ordering is not necessary and as an example, an algorithm is provided in Section 5.6 where the interprocessor communication is mapped on the binary tree structure.

The final local processing step must propagate filtered or smoothed estimates in parallel throughout each subinterval. Again, given the operations executed in the first processing step, few options are available for the final processing step. One method to accomplish this is to refilter or resmooth the original data given the global estimates of the boundary of each subinterval. Another method is to apply change of initial condition (CIC) equations to locally filtered quantities, and apply the Mayne-Fraser algorithm to recover smoothed estimates. In the case where the filtering step is radial, the backward sweep of the Rauch-Tung-Striebel algorithm can be used to update filters estimates to smoothed estimates.

As an aid to analyzing parallel smoothing algorithms, there are minor issues which we want to address. We are interested in obtaining smoothed estimates, however many parallel estimation algorithms primarily address obtaining filtered estimates. We view these algorithms as part of a parallel implementation of the Mayne-Fraser two filter algorithm. A second minor issue concerns the availability of a clear statistical interpretation for parallel algorithms. For example, in some cases the derivation of an algorithm begins with the specification of the solution to the smoothing problem e.g. with the Hamiltonian TPBVDS and proceeds with parallelizing the computations.

In contrast, in our approaches we parallelize the smoothing problem, so that each processing step has a clear statistical interpretation.

The method of dividing the data discussed in this chapter is not the only one which is suited for extension for two dimensional systems. Another method is to distribute to each processor, every L^{th} datum as adopted by Hashemipour and Laub[13]. This 'round-robin' approach requires a central processor to merge the local estimates to produce globally filtered or smoothed estimates. This method of distributing the data among processors has been demonstrated in the literature to be viable for 2-D iterative algorithms for solving pde's [9],[30]. Specifically the iterative algorithms which use red-black ordering are examples of the use of multidimensional round-robin techniques. Using the analogy of a checker board, data in red regions are updated simultaneously, then data in black regions are updated. The process then repeats on the red data. While one can imagine assigning all of the red data to one processor and all of the black data to another processor, greater efficiencies can be obtained by assigning one red and one black region to one processor. From the point of view of understanding the operations on the underlying process, difficulty arises because the reciprocal nature of the process is lost under sampling. This contrasts with the 1-D case where all samplings of 1-D reciprocal processes, are themselves reciprocal. Indeed it is essentially this fact that causes the two-dimensional algorithms which use red black ordering to be iterative rather than recursive. This fact is exploited heavily in the algorithm which is presented in section 5.6 at the end of this chapter.

The analysis of complexity of the algorithms in this thesis basically amounts to an accounting of the computation, storage, and communication requirements needed to implement the algorithms in this thesis. For any given computation there are a variety of ways in which it may be computed, with various trade-offs involving, accuracy, and operation count. These considerations vary with the specific processor on which the computation is to be implemented. The approach which we take here is to look at issues specific to the algorithm, and to choose a 'reasonable' means of computation. We assume that the amount of time which a processor spends in computation is proportional to the number of additions and multiplications required

to perform the computation. This may not necessarily be the case but without specific data on the processor and the algorithm by which matrix multiplies, for example, are implemented this is also a reasonable assumption. To aid in understanding the complexity computations we refer the reader to the computations on complexity for the FMLF, Mayne-Fraser smoother, the Rauch-Tung-Striebel smoother and the ML estimation problem in Sections 4.3, 4.4, 4.5, and 2.1 respectively.

The comparison of off-line computations for these algorithms varies substantially if there are fundamental differences in the way that estimates are computed. For example, the quantities

$$\Sigma_x - \Sigma_x H^T (H\Sigma_x H^T + R)^{-1} H\Sigma_x \tag{5.4}$$

$$(\Sigma_x^{-1} + H^T R^{-1} H)^{-1} \tag{5.5}$$

and

$$\begin{bmatrix} 0 \\ 0 \\ -I \end{bmatrix}^T \begin{bmatrix} \Sigma_x & 0 & P_x \\ 0 & R & H \\ P_x & H^T & 0 \end{bmatrix}^{\#} \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix} \tag{5.6}$$

are increasingly complex ways to compute the same quantity, however some expressions are applicable when others are not. For example in (5.4) the matrix R may be positive semidefinite. Only $(H\Sigma_x H^T + R)$ need be invertible. In (5.5), R must always be invertible. In (5.6) few restrictions are placed upon the matrices Σ_x, R and H, but as a result we must compute pseudo-inverses. Because the manner in which the quantities are computed interferes with our ability to compare one algorithm to another we will assume that locally, all algorithms compute quantities using FMLF unless it is clearly unnecessary. It should be noted however that the computations listed here are all $\mathcal{O}(n^3)$ computations.

The on-line computations will vary little despite dramatic differences in the off-

line computations. This is simply because the on-line computations simply reflect the basic linear relationships that the estimate at $k+1$ have with the data $y(k)$ and the estimate at k. All of the issues of estimability, pseudo-inverses, and perfect data are all handled off-line.

We would expect the computation involved with any algorithm using processors which are arranged on a linear array to be comparable. The amount of computation associated with the data exchange step rises linearly with the number of processors, while the remainder of the computations we would expect to be inversely proportional to the number of processors. Thus we expect the amount of time required to produce the results to be given in the following form

$$T = \frac{K}{L}\tau_1 f(n,p,m) + L\tau_2 g(n) \tag{5.7}$$

where τ_i are machine dependent parameters, K is the length of the entire interval on which the process is defined, L is the number of processors, and f and g are algorithm dependent functions of the parameters n, p and m which are the dimensions of the state, the observations and the driving noise respectively. This general relationship of the amount of time to perform computations to the number of processors holds for both the off-line and on-line computations. Sometimes it is necessary that the off-line computations be carried out in advance of the on-line computations, but for the algorithms discussed in this chapter, the off-line computations can also be computed in parallel.

The algorithm in Section 5.6 does not operate on a linear array of processors. Instead since the operations map to a binary tree, we will assume a hypercube interconnection between processors. As a result we expect that the form for the computation time to be given by

$$T = \frac{K}{L}\tau_1 f(n,p,m) + \tau_2 g(n)\log_2 L \tag{5.8}$$

In our algorithm discussed in Section 5.6, both the off-line and the on-line computations have a binary tree structure. As a result the same time dependence applies to both off-line and on-line computation.

We may further compute the optimal number of processors by minimizing (5.7) yielding

$$\hat{L} = \sqrt{\frac{K\tau_1 f(n,p,,m)}{\tau_2 g(n)}} \tag{5.9}$$

Similarly the same minimization can be carried out in (5.8) yielding

$$\hat{L} = \frac{K\tau_1 f(n,p,,m)}{\tau_2 g(n)} \tag{5.10}$$

and checking the two integers which bound $\hat{L}$.

The optimal computation time for algorithms described by (5.7) is given by

$$\hat{T} = \sqrt{\tau_2 g(n) \times K\tau_1 f(n,p,m)} \tag{5.11}$$

which increases with the square root of K the number of data points. The optimal computation time for algorithms described by (5.8) is given by

$$\hat{T} = \tau_2 g(n)(1 + \log_2 \frac{K\tau_1 f(n,p,m)}{\tau_2 g(n)}) \tag{5.12}$$

Thus in this case the amount of time it takes algorithms described by (5.8) to compute estimates increases with the logarithm of the number of points K.

5.1 Parallel estimation algorithm which uses the Partition Theorem

Morf, Dobbins, Freidlander, and Kailath [14] presented a square root algorithm for parallel filtering, and smoothing causal systems in discrete time. We will not focus on the square root nature of the algorithm or the details of the computation, but will focus on the essentials of the interprocessor communication step. The filtering algorithm follows. To aid in this discussion we will define the notation for three estimates. The first is the expectation operator $E[x(k)|\alpha,\beta]$ representing the conditional expectation of $x(k)$ given data $y(\tau)$, $\alpha \leq \tau \leq \beta$. The second is the expectation operator given the same data set plus the information that the initial condition $x(\alpha)$ is exactly zero which will be denoted by $E_o[x(k)|\alpha,\beta]$. The third, denoted by $\hat{x}[\alpha|\alpha,\beta]$, represents an ML estimate of $x(\alpha)$ based on data $y(\alpha)$, through $y(\beta)$.

Step 1

On the i^{th} segment, forward predicted estimates $E_o[x(k)|k_i, k-1]$ are computed as is the estimate $\hat{x}[k_i|k_i,k]$. From the partition theorem [37] the estimate $\hat{x}[k_i|k_i,k]$ is shown to be obtainable from forward filtering the innovations which result from computing $E_o[x(k)|k_i,k-1]$.

Step 2

Information is exchanged between neighboring processors to compute the global filtered estimates at each k_i.

The interprocessor exchange step involves combining the globally filtered estimate at k_i, denoted by $E[x(k_i)|k_o,k_i-1]$ which is obtained from the previous processor, and combining it with the local backwards filtered estimate $\hat{x}[(k_i)|k_i,k_{i+1}-1]$ to compute a smoothed estimate $E[x(k_i)|k_0,k_{i+1}-1]$. Finally $E[x(k_i)|k_0,k_{i+1}-1]$ can be combined with the locally filtered estimate $E_o[x(k)|k_i,k-1]$ to arrive at the globally filtered estimate $E[x(k_{i+1})|k_o,k_{i+1}-1]$.

The set of equations which govern this is given by the partition theorem and is

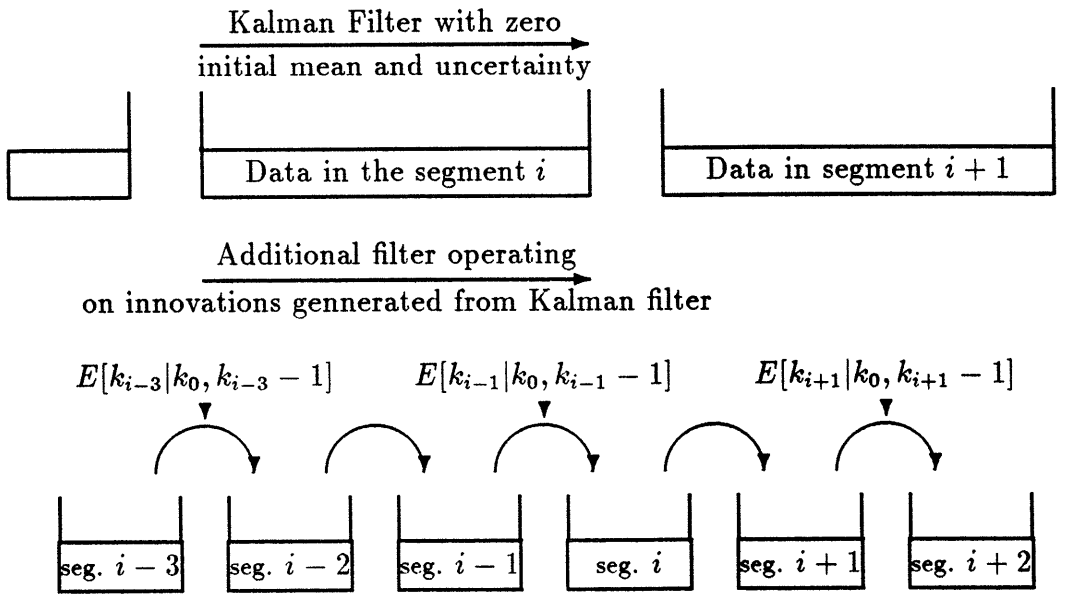


Figure 5-2: Parallel estimation in algorithm by Morf et al.

shown by the following.

$$\begin{aligned} E[x(k_{i+1})|k_0, k_{i+1}-1] &= \Phi(k_{i+1}, k_i)E[x(k_i)|k_0, k_{i+1}-1] \\ &+ E_o[x(k_{i+1})|k_i, k_{i+1}-1] \end{aligned} \tag{5.13}$$

$$\begin{aligned} P(k_{i+1}|[k_0, k_{i+1}-1]) &= \Phi(k_{i+1}, k_i)P(k_i|[k_0, k_{i+1}-1])\Phi^T(k_{i+1}, k_i) \\ &+ P_o(k_{i+1}|k_i, k_{i+1}-1) \end{aligned} \tag{5.14}$$

where $\Phi(k_{i+1}, k_i)$ is the state transition matrix for the system, $P_o(k_{i+1}|k_i, k_{i+1}-1)$ is the covariance associated with the estimate $E_o[x(k_{i+1})|k_i, k_{i+1}-1]$ and $P(k_{i+1}|k_i, k_{i+1}-1)$ is the covariance associated with the estimate $E[x(k_{i+1})|k_i, k_{i+1}-1]$.

The above equations for the interprocessor communication step are implemented as a filter with a structure resembling that of the Kalman filter.

$$\begin{aligned} E[x(k_{i+1})|k_0, k_{i+1}-1] &= \Phi(k_{i+1}, k_i)E[x(k_i)|k_0, k_i-1] + E_o[x(k_i)|k_i, k_{i+1}-1] \\ &+ K_i(E[x(k_i)|k_i, k_{i+1}-1] - E[x(k_i)|k_0, k_i-1]) \end{aligned} \tag{5.15}$$

Here the local boundary estimates can be interpreted as observations. The compu-

tations are a little more complex due to the fact that embedded in the problem are corrections for the fact that the local processing makes assumptions about the value of the process at the boundary. However this encourages the notion that the interprocessor communication step can be formulated as the simple application of a known filtering algorithm on a sampled system using the local estimates as observations.

Step 3

The globally filtered estimates of the interior points $E[x(k)|k_0, k-1]$ are obtained by combining the global filtered estimate of the boundary $E[x(k_i)|k_o, k_i-1]$, with the local fixed point smoothed estimate of the same boundary given by $\hat{x}[k_i|k_i, k-1]$ and the locally filtered estimate of the interior points $E_o[x(k)|k_i, k-1]$ in the following fashion.

We will not go into the complexity of this algorithm because we are not discussing the details of this algorithm but just the essentials of the partitioning of the problem. Local estimation based on the assumption of perfect boundary knowledge aids in partitioning the estimation problem both in this algorithm and in the algorithm discussed in the next section. However we will see that it is possible to partition the problem such that the local estimates have the interpretation of optimal estimates based on local data without additional assumptions concerning perfectly known initial conditions.

5.2 Inward and outward recursions in a parallel smoothing algorithm

Tewfik, Willsky, and Levy [15] describe a distributed smoothing algorithm based on the partition theorem and the notion of inward and outward processes. It is included in this section for three reasons. First it is an interesting algorithm in its own right. Secondly we will adopt inward and outward processing techniques in the new algorithms presented in Section 5.5 and 5.6. Finally one may ask how his algorithm applies to STPBVDS's.

In particular, the algorithm discussed in Tewfik, et al.[15] involves the construction of an outward Markov model. This model is written as a recursion from the center of the interval outward to the boundary. A priori information for these models is specified at the center of the interval. The parallel algorithm discussed in this section starts with this model. However, since all STPBVDS's are Markov, a causal model could be constructed first, then the outward Markov Model can be obtained as described in [15]. Similarly, the outward Markov Model can also be obtained directly.

Although the algorithm in [15] is given in continuous time, we present here a discrete time analog. This formulation also yields additional simplification if the process is stationary. Using a joint model for $x(k)$ and $x(-k)$, local Kalman filtering is performed radially outward from the center of each interval. Information is exchanged between neighboring processors to compute the globally smoothed estimates. Finally the interior points of each segment are updated radially inward and in parallel to obtain the globally smoothed estimate of the state everywhere.

Given a causal system

$$\begin{aligned} x(k+1) &= A_k x(k) + B_k u(k) \\ y(k) &= C_k x(k) + D_k u(k) \end{aligned} \tag{5.16}$$

where the following holds

$$B_k D_k^T = 0 \tag{5.17}$$

The data is segmented along the time axis to be processed independently. In each segment consider the case where the time origin has been shifted to the center of each interval. Tewfik defines the following process $x_{pm}(k)$.

$$x_{pm}(k) = \begin{bmatrix} x(k) \\ \Sigma_{-k}^{-1} x(-k) \end{bmatrix} \tag{5.18}$$

where Σ_k satisfies the Lyapunov equation

$$\Sigma_{k+1} = A_k \Sigma_k A_k^T + B_k B_k^T \tag{5.19}$$

and

$$y_{pm} = \begin{bmatrix} y(k) \\ y(-k) \end{bmatrix} \tag{5.20}$$

The model for the system is described by the following matrices

$$\mathcal{A}_k = \begin{bmatrix} A_k & 0 \\ 0 & A_{-k-1}^T \end{bmatrix} \tag{5.21}$$

$$\mathcal{B}_k = \begin{bmatrix} B_k & 0 \\ 0 & \Sigma_{-k-1}^{-1} A_{-k-1}^{-1} B_{-k} \end{bmatrix} \tag{5.22}$$

$$\mathcal{C}_k = \begin{bmatrix} C_k & 0 \\ 0 & C_{-k-1} \end{bmatrix} \tag{5.23}$$

$$\mathcal{D}_k = \begin{bmatrix} D_k & 0 \\ 0 & -D_{-k-1} \end{bmatrix} \tag{5.24}$$

(5.25)

Step 1:

Locally, Kalman filtering is performed on the following model

$$x_{pm}(k+1) = \mathcal{A}_k x_{pm}(k) + \mathcal{B}_k w(k) \tag{5.26}$$

$$y_{pm}(k) = \mathcal{C}_k x_{pm}(k) + \mathcal{D}_k w(k) \tag{5.27}$$

where

$$E[x_{pm}(0)x_{pm}^T(0)] = \begin{bmatrix} \Sigma(0) & I \\ I & \Sigma^{-1}(0) \end{bmatrix} = \Sigma_{pm}(0) \tag{5.28}$$

In the original time coordinate system the Bayesian estimate of the local boundary is given by $\hat{x}(k_{i-1}|k_{i-1}, k_i)$ and $\hat{x}(k_i|k_{i-1}, k_i)$.

Step 2:

A two filter algorithm is used to smooth the data at the endpoints of the segments. Thus the smoothed estimate at an endpoint is given by

$$\begin{aligned} \Sigma_s^{-1}(k_i|k_o, k_N)\hat{x}(k_i|k_o, k_N) &= \Sigma^{-1}(k_i|k_o, k_i)\hat{x}(k_i|k_o, k_i) \\ &+ \Sigma^{-1}(k_i|k_i, k_N)\hat{x}(k_i|k_i, k_N) \end{aligned} \tag{5.29}$$

$$\begin{aligned} \Sigma_s^{-1}(k_i|k_o, k_N) &= \Sigma_s^{-1}(k_i|k_o, k_i) \\ &+ \Sigma_s^{-1}(k_i|k_i, k_N) - \Sigma_{k_i}^{-1} \end{aligned} \tag{5.30}$$

where $\Sigma_{k_i}^{-1}$ is the a priori covariance of $x(k_i)$. The forward filtered estimates for the endpoints are given by

$$\begin{aligned} \Sigma_s^{-1}(k_{i-1}|k_o, k_i)\hat{x}(k_{i-1}|k_o, k_i) &= \Sigma^{-1}(k_{i-1}|k_o, k_{i-1})\hat{x}(k_{i-1}|k_o, k_{i-1}) \\ &+ \Sigma^{-1}(k_{i-1}|k_{i-1}, k_i)\hat{x}(k_{i-1}|k_{i-1}, k_i) \end{aligned} \tag{5.31}$$

$$\begin{aligned} \Sigma_s^{-1}(k_{i-1}|k_o, k_i) &= \Sigma^{-1}(k_{i-1}|k_o, k_{i-1}) \\ &+ \Sigma^{-1}(k_{i-1}|k_{i-1}, k_i) - \Sigma_{k_{i-1}}^{-1} \end{aligned} \tag{5.32}$$

and

$$\begin{aligned}\hat{x}(k_i|k_o,k_i) &= \hat{x}(k_i|k_{i-1},k_i) \qquad (5.33)\\ &+ \Phi(0,\frac{K}{L})[\hat{x}(k_{i-1}|k_o,k_i)-\hat{x}(k_{i-1}|k_{i-1},k_i)]\\ \Sigma(k_i|k_o,k_i) &= \Sigma(k_i|k_{i-1},k_i)\\ &+ \Phi(0,\frac{K}{L})[\Sigma(k_{i-1}|k_o,k_i)-\Sigma(k_{i-1}|k_{i-1},k_i)]\Phi^T(0,\frac{K}{L}) \qquad (5.34)\end{aligned}$$

Before continuing to Step 3, equations (5.31) and (5.33) need further discussion. Assume locally at the start of a subinterval that the initial condition was known exactly. We may then write

$$\begin{aligned}\hat{x}(k_i|k_o,k_i) &= \hat{x}^o(k_i|k_{i-1},k_i) \qquad (5.35)\\ &+ \Phi(0,\frac{K}{L})\hat{x}(k_{i-1}|k_o,k_i)\\ \Sigma(k_i|k_o,k_i) &= \Sigma^o(k_i|k_{i-1},k_i)\\ &+ \Phi(0,\frac{K}{L})\Sigma(k_{i-1}|k_o,k_i)\Phi^T(0,\frac{K}{L}) \qquad (5.36)\end{aligned}$$

We still need to compute $\hat{x}^o(k_i|k_{i-1},k_i)$, the estimate based on the 'assumption' that $x(k_{i-1}) \equiv 0$. Since $x^o(k) = x(k) - \Phi(0,k)x(0)$ we find that

$$\begin{aligned}\hat{x}^o(k_i|k_{i-1},k_i) &= \hat{x}(k_i|k_{i-1},k_i) - \Phi(0,\frac{K}{L})\hat{x}(k_{i-1}|k_{i-1},k_i) \qquad (5.37)\\ \Sigma^o(k_i|k_{i-1},k_i) &= \Sigma(k_i|k_{i-1},k_i) - \Phi(0,\frac{K}{L})\Sigma(k_i|k_{i-1},k_i)\Phi^T(0,\frac{K}{L}) \qquad (5.38)\end{aligned}$$

Equation (5.35), and (5.37) can be combined to produce (5.33). This is an application of the partition theorem[37], [19]. Similar equations exist for the backwards filtered estimates. We will see that parallel processing algorithms may have much simpler statistical interpretation of the manner in which data is combined than the partition theorem lends in this algorithm.

Once the smoothed estimate of the state is computed at each of the k_i, the data in each interval is updated via the backward sweep of the Rauch-Tung-Striebel Algorithm [21]. Following the development of the Rauch-Tung-Striebel Algorithm in

Section 4.3, under the assumption of no perfect observations and estimable states, it is given by

$$\hat{x}_{pms}(k) = \hat{x}_{pm}(k) - \Sigma_{pm}[k|k]\mathcal{A}_k^T\Sigma_{pm}^{-1}[k+1|k](\hat{x}_{pms}(k+1) - \mathcal{A}_k^T\hat{x}_{pm}(k)) \qquad (5.39)$$

where $\Sigma_{pm}[k|k]$ is the filtered covariance obtained through Kalman filtering.

This algorithm involves the preprocessing to generate the outward Markov model in equations (5.26), (5.27), and (5.28).

5.2.1 Complexity

Step 1

Off-line

Besides the obvious computation involved in this algorithm, given by the on-line computations and the computation of the matrices F_k, G_k and the covariances, there are two other sources of computational complexity which are incurred in preparation to use this algorithm. The first is that if the process is presented as a normal state space system, then a priori information must be propagated to the center of each subinterval and to the subinterval boundaries. This is accomplished by using the Lyapunov equation. More importantly further computation is involved to construct the outwardly causal model used locally in this algorithm. Both the causal, and outward causal models are used in this algorithm. Locally the outward causal model is used. During the interprocessor exchange step, the standard causal model is used, as evidenced by the use of the state transition matrix. If the original system is specified as a STPBVDS, or as the outward model, then additional computation would be involved to generate the state transition matrix.

The local processing amounts to applying the Rauch-Tung-Striebel algorithm to a system whose state dimension is $2n$ while the dimension of the observations is given by $2p$ and the dimension of the driving noise is given by $2m$. The filter is only run over half the interval since the time axis locally is radial for the local processing.

Since the interprocessor communication requires the computation of matrix inverses, we will assume that that there are no perfect observations, and as a consequence the error covariance of the boundary estimate has full rank. The computation of one processor acting on one region is given by

$$\frac{1}{2}\frac{K}{L}\mathcal{K}(2n, 2p, 2m; \text{off} - \text{line}, \text{invertible}) \tag{5.40}$$

where $\mathcal{K}(n, p, m; \text{off} - \text{line}, \text{invertible})$ is defined in Section 4.3.1 by

$$\frac{2}{3}(2n+p)^3 + 2n(2n+p)^2 + 6n^3 + 2mn^2 + 2pn^2 \tag{5.41}$$

On-line

The on-line computation of one processor acting on one region is given by

$$\frac{1}{2}\frac{K}{L}\mathcal{K}(2n, 2p, 2m; \text{on} - \text{line}) \tag{5.42}$$

where $\frac{K}{2L}$ represents the radius of the subintervals and $\mathcal{K}(n, p, m; \text{on} - \text{line}, \text{invertible})$ is defined in Section 4.3.1 by

$$\mathcal{K}(n, m, p; \text{on} - \text{line}) = 2n(n+p) \tag{5.43}$$

Step 2

Off-line

The off-line computation required to implement the interprocessor communication can be implemented with

$$\begin{aligned} &(L-1)(4\mathcal{I}(n,n) + 8n^3) \\ &= (L-1)10.67n^3 \end{aligned} \tag{5.44}$$

flops.

On-line

The on line processing can be carried out in

$$(L-1)8n^2 \tag{5.45}$$

flops.

Step 3

Off-line

Here the backward sweep of the Rauch-Tung-Striebel algorithm is implemented. Thus the computation of one processor acting on one region is given by

$$\tfrac{1}{2}\tfrac{K}{L}\mathcal{T}(2n; \text{off} - \text{line}, \text{invertible}) \tag{5.46}$$

where $\mathcal{T}(n; \text{off} - \text{line}, \text{invertible})$ is defined in Section 4.5.1 by

$$14\frac{2}{3}n^3 \tag{5.47}$$

On-line

The on-line computation of one processor acting on one region is given by

$$\tfrac{1}{2}\tfrac{K}{L}\mathcal{T}(2n; \text{on} - \text{line}) \tag{5.48}$$

where $\mathcal{T}(n; \text{on} - \text{line})$ is defined in Section 4.5.1 by

$$4n^2 \tag{5.49}$$

In summary, the total amount of off-line time required to compute the smoothed estimate of the state in this parallel smoothing algorithm is given by

$$\begin{aligned} T &= \tfrac{1}{2}\tfrac{K}{L}\tau_1[\mathcal{T}(2n; \text{off} - \text{line}, \text{invertible}) + \mathcal{K}(2n, 2p, 2m; \text{off} - \text{line}, \text{invertible})] + (L-1)10.67\tau_2 n^3 \\ &= \tfrac{K}{2L}\tau_1[165.33n^3 + 5.33(2n+p)^3 + 16n(2n+p)^2 + 16mn^2 + 16pn^2] + (L-1)10.67\tau_2 n^3 \end{aligned} \tag{5.50}$$

where the appropriate f and g functions in (5.7) for this case can be immediately identified. The amount of on-line time required to compute the smoothed estimate

of the state in this parallel smoothing algorithm is given by

$$\begin{aligned} T &= \tfrac{1}{2}\tfrac{K}{L}\tau_1[\mathcal{T}(2n;\text{on}-\text{line}) + \mathcal{K}(2n;\text{on}-\text{line})] + (L-1)\tau_2 10.67n^3 \\ &= \tfrac{K}{L}\tau_1[12n^2 + 4np] + (L-1)\tau_2 8n^2 \end{aligned} \tag{5.51}$$

Again the appropriate f and g functions can be readily identified.

5.3 A Recursive Domain Decomposition Algorithm

In the first step of this algorithm we produce locally ML smoothed estimates. The second step solves the ML smoothing problem at the boundaries, and propagates the boundary information to each of the processors. The third step updates the locally smoothed estimates to produce globally smoothed estimates. This algorithm is essentially a parallelization of the Mayne-Fraser two filter smoother where, to aid in the interprocessor exchange step, forward and backward fixed point smoothers are implemented instead of the usual forward and backward filters. Since ML estimates are used in each subinterval, the computed estimates in each subinterval are independent, allowing simple estimation equations to be used during the second step to produce globally smoothed estimates of the boundaries of the subintervals.

In order to minimize the number of projection matrices required to describe the algorithm in this section, we will assume that the state of the system is estimable given local data although the state may not be estimable based on local causal or anti causal data alone.

Step 1:

Locally each processor implements the 2n dimensional fixed point smoother for (5.1) given by computing the FMLF for the following augmented system

$$\begin{bmatrix} E_{k+1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k+1) \\ \xi(k+1) \end{bmatrix} = \begin{bmatrix} A_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ \xi(k) \end{bmatrix} + \begin{bmatrix} B_k \\ 0 \end{bmatrix} u(k) \quad (5.52)$$

$$\begin{bmatrix} I & -I \end{bmatrix} \begin{bmatrix} x(k_{i-1}) \\ \xi(k_{i-1}) \end{bmatrix} = 0 \quad (5.53)$$

$$y(k) = \begin{bmatrix} C_k & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \xi(k) \end{bmatrix} + v(k) \quad (5.54)$$

$$k_{i-1}+1 \leq k \leq k_i \quad (5.55)$$

The fixed point smoother produces at each point the following estimate given local

data

$$\begin{bmatrix} \hat{x}[k|k_{i-1}+1,k] \\ \hat{x}[k_{i-1}|k_{i-1}+1,k] \end{bmatrix} = P_{x(k),x(k_{i-1})|k_{i-1}+1,k} \begin{bmatrix} x(k) \\ x(k_{i-1}) \end{bmatrix} + \begin{bmatrix} \tilde{x}[k|k_{i-1}+1,k] \\ \tilde{x}[k_{i-1}|k_{i-1}+1,k] \end{bmatrix} \tag{5.56}$$

where $P_{x(k),x(k_{i-1})|k_{i-1}+1,k}$ is the projection matrix generated by the FMLF indicating the estimable subspace and the error statistics are given by

$$P_{x(k),x(k_{i-1})|k_{i-1}+1,k} \begin{bmatrix} \tilde{x}[k|k_{i-1},k] \\ \tilde{x}[k_{i-1}|k_{i-1},k] \end{bmatrix} \sim N \left[0; \begin{bmatrix} \Sigma_{xx}[k|k_{i-1}+1,k] & \Sigma_{x\xi}[k|k_{i-1}+1,k] \\ \Sigma_{\xi x}[k_{i-1}|k_{i-1}+1,k] & \Sigma_{\xi\xi}[k_{i-1}|k_{i-1}+1,k] \end{bmatrix} \right] \tag{5.57}$$

In particular for $k = k_i$, a measurement of the boundary given local data is obtained. Since $x(k)$ is estimable given local data, the projection matrix is equal to the identity at the boundary of the local subinterval. Similarly a backward fixed point smoother can be constructed by computing the BMLF for the following augmented system

$$\begin{bmatrix} E_{k+1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k+1) \\ \zeta(k+1) \end{bmatrix} = \begin{bmatrix} A_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ \zeta(k) \end{bmatrix} + \begin{bmatrix} B_k \\ 0 \end{bmatrix} u(k) \tag{5.58}$$

$$\begin{bmatrix} I & -I \end{bmatrix} \begin{bmatrix} x(k_i) \\ \zeta(k_i) \end{bmatrix} = 0 \tag{5.59}$$

$$y(k) = \begin{bmatrix} C_k & 0 \end{bmatrix} \begin{bmatrix} x(k) \\ \zeta(k) \end{bmatrix} + v(k) \tag{5.60}$$

$$k_{i-1}+1 \leq k \leq k_i \tag{5.61}$$

From the backward fixed point smoother one obtains

$$\begin{bmatrix} \hat{z}^b[k|k+1,k_i] \\ \hat{x}[k_i|k+1,k_i] \end{bmatrix} = P_{z^b(k),x(k_i)|k+1,k_i} \begin{bmatrix} A_k & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ x(k_i) \end{bmatrix} + \begin{bmatrix} \tilde{z}^b[k|k+1,k_i] \\ \tilde{x}[k_i|k+1,k_i] \end{bmatrix} \tag{5.62}$$

where the error statistics are given by

$$P_{z^b(k),x(k_i)|k+1,k_i}\begin{bmatrix}\tilde{z}[k|k+1,k_i]\\ \tilde{x}[k_i|k+1,k_i]\end{bmatrix}\sim N\left[0;\begin{bmatrix}\Sigma_{zz}[k|k+1,k_i] & \Sigma_{z\zeta}[k|k+1,k_{i-1}]\\ \Sigma_{\zeta z}[k_i|k+1,k_{i-1}] & \Sigma_{\zeta\zeta}[k_{i-1}|k+1,k_{i-1}]\end{bmatrix}\right] \tag{5.63}$$

Since we are implementing a parallel Mayne-Fraser algorithm, there are two independent measurements of $x(k)$ at each point in time. One resulting from the forward filter, and one from the backward filter. One possibility is that locally smoothed estimates can be computed by incorporating all of the local information first. Considering the local 'measurements' of the local states provided by the local forward and backward prediction filters, and local 'measurements' of the states which bound the subregion, the ML estimate based on these 'measurements'

$$\begin{aligned}&\begin{bmatrix}P_{x(k_{i-1}),x(k)|k_{i-1},k} & \vdots & 0\\ \cdots\cdots & & \cdots\cdots\\ 0 & \vdots & P_{z^b(k),x(k_i)|k+1,k_i}\end{bmatrix}\begin{bmatrix}\hat{x}[k_{i-1}|k_{i-1},k]\\ \hat{x}[k|k_{i-1},k]\\ \cdots\cdots\\ \hat{z}^b[k|k+1,k_i]\\ \hat{x}[k_i|k+1,k_i]\end{bmatrix}\\ &=\begin{bmatrix}I & 0 & 0\\ 0 & I & 0\\ \cdots\cdots & & \\ 0 & E_k & 0\\ 0 & 0 & I\end{bmatrix}\begin{bmatrix}x(k_{i-1})\\ x(k)\\ x(k_i)\end{bmatrix}+\begin{bmatrix}\tilde{x}[k_{i-1}|k_{i-1},k-1]\\ \tilde{x}[k|k_{i-1},k-1]\\ \cdots\cdots\\ \tilde{z}^b[k|k+1,k_i,]\\ \tilde{x}[k_i|k+1,k_i]\end{bmatrix}\end{aligned} \tag{5.64}$$

results in the locally smoothed ML estimate of the boundaries of the subinterval and the state $x(k)$ given by

$$\begin{bmatrix}\hat{x}[k|k_{i-1},k_i]\\ \hat{x}[k_{i-1}|k_{i-1},k_i]\\ \hat{x}[k_i|k_{i-1},k_i]\end{bmatrix}=\begin{bmatrix}x(k)\\ x(k_{i-1})\\ x(k_i)\end{bmatrix}+\begin{bmatrix}\tilde{x}[k|k_{i-1},k_i]\\ \tilde{x}[k_{i-1}|k_{i-1},k_i]\\ \tilde{x}[k_i|k_{i-1},k_i]\end{bmatrix} \tag{5.65}$$

where thanks to the assumption of the estimability of the state based on local data,

the projection matrix associated with the estimate in (5.65) is equal to the identity. The error covariance for (5.64) is obtained from the forward and backward filters, and the error covariance in (5.65) comes directly from the solution to the ML estimation problem applied to (5.65).

Step 2:

The globally forward filtered estimate is constructed by implementing a filter which operates on the boundaries of the subintervals using local boundary measurements as observations. From Lemma 4.3, the estimate $\hat{x}[k_i|0,k_i]$ can be constructed at each point in time with the addition of the measurement $\hat{x}[k_{i-1}|0,k_{i-1}]$. The estimate which processor $i+1$ requires is obtained from a filter constructed from the following measurements.

$$\begin{bmatrix} \hat{x}[k_i|k_{i-1},k_i-1] \\ \hat{x}[k_{i-1}|k_{i-1},k_i-1] \\ \hat{x}[k_{i-1}|k_0,k_i-1] \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k_i) \\ x(k_{i-1}) \end{bmatrix} + \begin{bmatrix} \tilde{x}[k_i|k_{i-1}+1,k_i] \\ \tilde{x}[k_{i-1}|k_{i-1}+1,k_i] \\ \tilde{x}[k_{i-1}|0,k_{i-1}] \end{bmatrix} \tag{5.66}$$

where represents the predicted estimate of the state given by (4.48). The globally filtered estimates are given by

$$\begin{aligned} \hat{x}[k_{i+1}|k_0,k_{i+1}] = & \begin{bmatrix} 0 & 0 & 0 & 0 & I \end{bmatrix} \\ & \begin{bmatrix} \Sigma_{\tilde{x}\tilde{x}}[k_i,k_i|k_i+1,k_{i+1}] & \Sigma_{\tilde{x}\tilde{x}}[k_i,k_{i+1}|k_i+1,k_{i+1}] & 0 & I & 0 \\ \Sigma_{\tilde{x}\tilde{x}}[k_{i+1},k_i|k_i+1,k_{i+1}] & \Sigma_{\tilde{x}\tilde{x}}[k_{i+1},k_{i+1}|k_i+1,k_{i+1}] & 0 & 0 & I \\ 0 & 0 & \Sigma_{\tilde{x}\tilde{x}}[k_i,k_i|k_0,k_i] & I & 0 \\ I & 0 & I & 0 & 0 \\ 0 & I & 0 & 0 & 0 \end{bmatrix}^{\#} \\ & \times \begin{bmatrix} \hat{x}[k_i|k_i+1,k_{i+1}] & \hat{x}[k_{i+1}|k_i+1,k_{i+1}] & \hat{x}[k_i|k_{i-1}+1,k_i] & 0 & 0 \end{bmatrix}^T \end{aligned} \tag{5.67}$$

Similarly a backward prediction filter can be constructed for computing backward global predicted estimates of the boundaries of the process.

Step 3:

At each point in time a locally and jointly smoothed estimate of $x(k)$ and the local

boundary $[x^T(k_{i-1})x^T(k_i)]^T$ is available along with the associated joint error covariance. This joint estimate was obtained in Step 1 in equation (5.65). Also globally filtered estimates of the local boundaries given by (5.68) are available from the processing in (5.67) in Step 2 and from the associated backward filter. The measurements (5.65) can be combined with the boundary measurement provided by the forward and backward filters in the interprocessor exchange step to obtain the smoothed estimate of the state at each point in the subinterval. Figure 5.3 shows a processor which operates on data form k_{i-1} to k_i recieving boundary measurements from neighboring processors to construct the smoothed estimate of the boundary at k_{i-1} and k_i.

$$\begin{bmatrix} \hat{x}[k_{i-1}|0,k_{i-1}] \\ \hat{x}[k_i|k_i,K] \end{bmatrix} = \begin{bmatrix} x(k_{i-1}) \\ x(k_i) \end{bmatrix} + \begin{bmatrix} \tilde{x}[k_{i-1}|0,k_{i-1}] \\ \tilde{x}[k_i|k_i,K] \end{bmatrix} \tag{5.68}$$

In this algorithm the boundary of the local interval is appended to the state at each point in time. If we were to consider adapting this algorithm to the two dimensional case, we would need again to append the state with the boundary. In 2-D however the boundary states of local subregions have high dimension. The smoothing algorithm would suffer a dramatic increase in computational complexity. Providing locally smoothed estimates is not the direction we wish to take in higher dimensional processors.

This method is comparable to the domain decomposition method[9] where a pde is solved in local regions in an attempt to compress the problem to solving a system of equations at the boundary. In this one-dimensional algorithm provided here, the complexity of the boundary smoothing problem has not increased and updating the interior points is simple. Domain decomposition methods will be discussed further in Chapter 6 in connection to multidimensional state estimation.

5.3.1 Complexity

Step 1

Off-line

The forward filter in this algorithm is a fixed point smoother. The state dimension

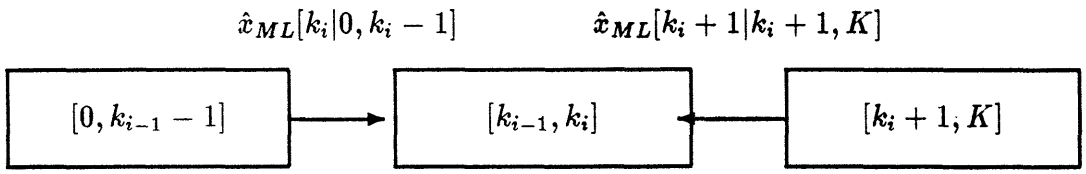


Figure 5-3: Boundary Measurements from Neighboring Processors

is given by $2n$. The dimension of the driving noise and the observations remains unchanged from the original model. The amount of computation which a given processor acting on a single region of data must execute is given by

$$\begin{aligned} &2\tfrac{K}{L}[\mathcal{K}(2n, m, p; \text{off} - \text{line}, \text{non} - \text{estimable}) + \mathcal{M}(7n, 3n, 7n)] \\ &= 2\tfrac{K}{L}[2159n^3 + 224n^2p + 16n^2m + 42np^2 + 2.67p^3] \end{aligned} \tag{5.69}$$

where $\mathcal{K}$ is defined in Section 4.3, and $\mathcal{M}$ is defined in Section 2.1.

On-line

The online computation required is given by

$$\begin{aligned} &\tfrac{K}{L}[2\mathcal{K}(2n, m, p; \text{on} - \text{line}) + 8n^2] \\ &= 2\tfrac{K}{L}[12n^2 + 4np] \end{aligned} \tag{5.70}$$

where the factor of two indicates that the computation needs to be carried out both for the forward filter and the backward filter.

Step 2

Off-line

The off-line computation required for the interprocessor communication is given by

$$(L-1)\mathcal{E}(5n, 2n, 5n) = (L-1)316.67n^3 \tag{5.71}$$

On-line

The on-line computation required for the interprocessor communication is given by

$$(L-1)6n^2 \tag{5.72}$$

Step 3

Off-line

Here local estimates of the state and the boundary are combined with globally filtered boundary measurements. The off-line computations are given by

$$\frac{K}{L}\mathcal{E}(8n, 3n, 8n) = \frac{K}{L}874.67n^3 \tag{5.73}$$

On-line

The amount of on-line computations is given by

$$T = \frac{K}{L}10n^2 \tag{5.74}$$

In summary, the total time required to compute the off-line computations is given by

$$\begin{aligned} T = \; & 2\tfrac{K}{L}[\mathcal{K}(2n, m, p; \text{off} - \text{line}, \text{non} - \text{estimable}) + \mathcal{M}(7n, 3n, 7n) \\ + \; & \tfrac{1}{2}\mathcal{E}(8n, 3n, 8n)] + (L-1)\mathcal{E}(5n, 2n, 5n)] \\ = \; & 2\tfrac{K}{L}[2596.33n^3 + 224n^2p + 16n^2m + 42np^2 + 2.67p^3] \\ + \; & 316.67(L-1)n^3 \end{aligned} \tag{5.75}$$

and the total time required to compute the estimates on-line is given by

$$T = \; 2\tfrac{K}{L}[\mathcal{K}(2n, m, p; \text{on} - \text{line}) + 4n^2 + 5n^2] + (L-1)6n^2 = \; 2\frac{K}{L}[17n^2 + 4np] + (L-1)6n^2 \tag{5.76}$$

5.4 Method of Oblique Projections

Many parallel processing algorithms take advantage of the partition theorem [19], or somehow compute locally optimal estimates and use the information obtained from neighboring processors to create globally optimal estimates. Simplification may result if each processor behaves with the knowledge that it will receive information from neighboring processors. Specifically, our aim is to represent equations (3.36), and (3.37) by the following 'measurements'.

$$Y(i) = \sum_j H(i,j)x(j) + v(i) \tag{5.77}$$

where $i \in \{1, 2, 3, ...L\}$. We then seek the operators $\mathcal{L}(j,i)$ such that

$$\hat{x}(j) = \sum_i \mathcal{L}(j,i)Y(i) \tag{5.78}$$

where $\mathcal{L}$ are oblique projection operators. Properties of these operators are discussed in by Ayalar and Weinert in [4]. The advantage of these operators are that the contributions of neighboring processors can be simply added together. It is desirable however that the operations indicated by $\mathcal{L}(j,i)$ be factorable into two operators of low dimension so that few parameters need be transmitted to neighboring processors. In other words, we desire $\mathcal{L}(j,i)$ to be written as $\mathcal{L}(j,i) = \mathcal{L}_1(j,i)\mathcal{L}_2(j,i)$ where $\mathcal{L}_2(j,i)Y(j)$ is a sufficient statistic of lowest dimension to be passed to neighboring processors, and is easily computed while computing $\mathcal{L}(j,j)Y(j)$. In addition we desire $\mathcal{L}_1(j,i)$ to be an operation which can be carried out in the i^{th} processor during the third step in the parallel processing algorithm. In this formulation, the computation in a given subinterval is suboptimal. This partitioning of the problem is based on the assumption that the proper information will become available from other processors. As a result all of the gain matrices in each subinterval are based on the knowledge of the statistics of the entire process and all of the data instead of having information only about the process and the data in the local interval. As a result after local estimates of the state has been obtained, the processors will not have to communicate

with neighboring processors to obtain covariance information.

The algorithm presented in this section has a structure similar to those discussed in Section 5.1, and 5.2. The data is partitioned into segments. Local filtering is performed upon each segment. The filter in this algorithm is an n dimensional filter instead of a $2n$ dimensional filter as in Section 5.1 and 5.2, which also results in computational savings over the algorithms discussed in those sections both off-line and on-line. In the data exchange step between neighboring processors sufficient statistics are passed to compute the smoothed estimate at the endpoints of each segment. Finally the points within each segment are updated from the information available at the endpoints of each segment. This algorithm is distinctive in that the global equations are implemented locally in all aspects except that the estimate of the boundary of the subinterval is set to zero. Since the local processing requires knowledge of the statistics of the process and data over the entire data interval, covariance information must be determined in advance of processing the data. In the other algorithm discussed in this chapter, and in the change of initial condition equations discussed in Ljung and Kailath[19], global statistical information are not assumed to be available to local subinterval processors.

The algorithm has the property that from the perspective of a particular processor processing data on the i^{th} interval, the computations of neighboring processors provide measurements of the process on the boundary of the i^{th} interval. These measurements are then used to provide measurements of the boundaries of neighboring processor in addition to updating the local estimates to their smoothed values.

The algorithm is as follows. Starting from (5.1) we seek a parallel implementation of the Mayne-Fraser smoother. The Mayne-Fraser algorithm constructs forward predicted estimates $z_f(k)$ and backward filtered estimates $x_b(k)$. The backward filtered estimates can be obtained from the BMLF, or if the system in (5.1) is causal, the backward estimates can be obtained from from backward information filtering [21]. If the system is causal then a backward Markovian model can be constructed for implementing a backward Kalman filter [34]. The filtered estimates are combined via equation (4.85)

The forward ML filter can be written as

$$\hat{x}_f(k+1) = \Phi_f(k+1,k)\hat{x}_f(k) + G_k^f y(k) \tag{5.79}$$

where

$$\Phi_f(k+1,k) = F_k^f P_z(k) A_k \tag{5.80}$$

and where F_k^f, and G_k^f are given by

$$\begin{bmatrix} F_k^{f,T} \\ G_k^{f,T} \\ -\Sigma_x[k+1|k+1] \end{bmatrix}^T = \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}^T \begin{bmatrix} \Sigma_{zf}[k+1|k] & 0 & E_{k+1} \\ 0 & R_{k+1} & C_{k+1} \\ E_{k+1}^T & C_{k+1}^T & 0 \end{bmatrix}^{\#} \tag{5.81}$$

where $P_x[k+1|k]$ is given by

$$P_x[k+1|k] = F_k^f E_{k+1} + G_k^f C_{k+1}^T \tag{5.82}$$

The algorithm is obtained by attempting to implement the above equation for the estimates directly. Specifically the gains in each subinterval should be based on all of the statistics of the process and data prior to the data in the given subinterval. If a parallel algorithm is required to compute F_k^f, and G_k^f, then the parallel algorithm in Section 5.3 can be used for this purpose.

Given that locally, all processors have the gains F_k^f, and G_k^f based on the statistics of the entire process, filtering the data is simple. Locally, each processor computes the zero state response (ZSR) solution to the filtering equations. When boundary information becomes available, the zero input response (ZIR) equations are used to update the output of the ZSR filter to the optimal globally filtered solution. In addition, the ZIR equations are used to compute the boundary condition for the subsequent processor.

Step 1:

Using the F_k^f, and G_k^f matrices, each processor computes the ZSR for the FMLF.

Thus locally each processor will compute

$$x_f^*(k) = \sum_{k_{i-1}}^{k-1} \Phi_f(k, \kappa+1) G_\kappa^f y(\kappa) \tag{5.83}$$

which is the solution to the FMLF equations assuming $x^*(k_{i-1}) = 0$. The state is computed for all k in the interval $[k_{i-1}, k_i]$. The matrix $\Phi_f(k, \kappa)$ is the state transition matrix for the FMLF. All of the information available from processors operating on data before time k_{i-1} is used to construct the estimate of the state at k_{i-1}. It is this estimate that has the interpretation as a boundary measurement.

Similarly with the F_k^b and G_k^b matrices, generated from the backward filter, each processor computes the ZSR for the BMLF. Thus locally each processor will compute

$$z_b^*(k) = \sum_{k}^{k_i} \Phi_b(k, \kappa) P_{z^b}(\kappa) E_\kappa G_\kappa^b y(\kappa) \tag{5.84}$$

which is the solution to the BMLF equations assuming $z^*(k_i+1) = 0$. $\Phi_b(k, k+1) = P_z(k) E_k F_{k+1}^b$ is the state transition matrix for the BMLF. The state is computed for all k in the interval $[k_{i-1}, k_i]$. All of the information available from processors operating on data before time k_{i-1} is used to construct the estimate of the state at k_{i-1}. This estimate also has the interpretation as a boundary measurement.

Step 2:

In the data exchange step, the i^{th} processor can immediately compute the initial condition, $\xi_f(i)$, for the next processor when the initial condition at time k_{i-1} given by $\xi_f(i-1)$ becomes available. This is accomplished using the state transition matrix of the FMLF as follows

$$\xi_f(k_i) \;=\; x_f^*(i) + \Phi_f(k_i, k_{i-1}) \xi_f(i-1) \tag{5.85}$$

where $\xi_f(i)$ represents the initial condition at time k_i. Also for the backward filter, the i^{th} processor can immediately compute the initial condition,$\xi_b(i)$, for the previous processor when the initial condition at time k_i given by $\xi_b(i)$ becomes available. This

is accomplished using the state transition matrix of the BMLF as follows

$$\xi_b(i-1) \;=\; x_b^*(k_{i-1}) + \Phi_b(k_{i-1}, k_i)\xi_b(i) \tag{5.86}$$

where $\xi_b(i)$ represents the initial condition at time k_i.

Step 3:

Similarly the results of the local processing in Step 1, can also be updated to the globally filtered estimates by using the state transition matrix.

$$x_f(k) \;=\; x_f^*(k) + \Phi_f(k, k_{i-1})\xi_f(i-1) \tag{5.87}$$

Similar processing can be done for the backwards estimates. The final result for the smoothed estimates is a weighted sum of the forwards and backwards estimates, given by equations (4.85), (4.86), and (4.87).

Before the estimates can be computed however, all of the local processors must be given sufficient information to compute gains based on the statistics of the entire process. These gains can be precomputed off line and distributed to each processor in advance of the estimation process.

5.4.1 Complexity

Since locally each processor requires knowledge about the global statistics of the process in order to implement the local filters, a certain amount of preprocessing is necessary if we intend off-line and on-line computations to take place concurrently.

Here we will briefly outline one method of computing the gains in parallel. The off-line computations for a fixed point smoother can be run in each subinterval in parallel as described in Section 5.3. Then an FMLF and a BMLF are run on the boundary points to generate the appropriate covariances and projection matrices as in Step 2 of Section 5.3. Given the covariances and projections for globally filtered estimates at the boundaries, the off-line computation proceeds as described in Section 4.3 and 4.4 with a local FMLF and BMLF which compute the required gains. The

state transition matrix for the FMLF, and the BMLF must be computed for the on-line interprocessor communication step.

Step 0

Preprocessing

Preprocessing consists of running the off-line computations for a forward fixed point smoother, and performing the interprocessor communication necessary to insure that globally filtered covariance data of each boundary has been propagated to each subinterval processor. The preprocessing requires

$$\begin{aligned} T = & \frac{K}{L}[\mathcal{K}(2n, m, p; \text{off} - \text{line}, \text{non} - \text{estimable}) + (L-1)\mathcal{M}(5n, 2n, 5n) \\ = & \frac{K}{L}[509.33n^3 + 224n^2p + 16n^2m + 42np^2 + 2.67p^3] + (L-1)583.33n^3] \end{aligned} \tag{5.88}$$

flops where we assume that the state is not causally or anticausally estimable. The remaining off-line computations can take place concurrently with the on-line calculation and will be included below with the on-line calculations.

Step 1

Off-line

After the preprocessing just described, forward and backward ML filters can be locally implemented using globally determined statistics. The off-line computation is given by

$$\begin{aligned} & 2\frac{K}{L}[\mathcal{K}(n, m, p; \text{off} - \text{line}, \text{non} - \text{estimable}) + 4n^3] \\ & = 2\frac{K}{L}[67.67n^3 + 56n^2p + 4N^2m + 21np^2 + 2.67p^3] \end{aligned} \tag{5.89}$$

where the additional $4n^3$ flops are needed to compute the state transition matrix for the ML filters.

On-line

The on-line computation required is given by

$$\frac{K}{L}8n(n+p) \tag{5.90}$$

Step 2

Off-line

There is no off-line component to this algorithm which can be implemented concurrently with this step because all off-line interprocessor communication takes place during the preprocessing step.

On-line

The on-line computation required for the interprocessor communication is given by

$$(L-1)2n^2 \tag{5.91}$$

Step 3

Off-line

The estimates from the forward and backward filters are combined to produce the smoothed estimate. The amount of off-line computation needed for this step is given by

$$\begin{aligned} &\tfrac{K}{L}\mathcal{F}(n,m;\text{off}-\text{line},\text{estimable}) \\ &\quad = \tfrac{K}{L}63n^3 \end{aligned} \tag{5.92}$$

where $\mathcal{F}$ is defined in section 4.3 and it is assumed that the state is estimable given all of the data.

On-line

The amount of on-line computations is given by

$$\tfrac{K}{L}2n^2 \tag{5.93}$$

In summary, the total off-line time required is given by

$$\begin{aligned} T = \;& \tfrac{K}{L}[2\mathcal{K}(n,m,p;\text{off}-\text{line},\text{non}-\text{estimable}) + 8n^3 + \mathcal{F}(n,m;\text{off}-\text{line},\text{estimable})] \\ + \;& \{\tfrac{K}{L}[\mathcal{K}(2n,m,p;\text{off}-\text{line},\text{non}-\text{estimable}) + (L-1)\mathcal{M}(5n,2n,5n)]\} \\ = \;& 2\tfrac{K}{L}[67.67n^3 + 56n^2p + 4N^2m + 21np^2 + 2.67p^3] + \tfrac{K}{L}63n^3 \\ + \;& \{\tfrac{K}{L}[509.33n^3 + 224n^2p + 16n^2m + 42np^2 + 2.67p^3] + (L-1)583.33n^3]\} \end{aligned} \tag{5.94}$$

where the term in braces indicates the preprocessing required. The on-line computation is given by

$$\frac{K}{L}(10n^2 + 8np) + (L+1)4n^2 \tag{5.95}$$

where f and g in (5.7) are readily identifiable.

5.5 Parallel ML Smoothing

In the treatment of parallel processing in this section, the issues of parallel processing are simplified as much as possible. The local processing consists of ML filtering starting from the center of each interval to the boundary. At the boundary, the optimal estimate is obtained given all of the local data and dynamic constraints. These boundary measurements represent 'observations' of the sampled system whose states are the boundaries of the local subintervals. Since the initial local processing computes ML estimates based on local data, these 'observations' are independent. The interprocessor communication step amounts to implementing the Mayne-Fraser two filter smoother on the sampled system consisting only of the boundary points, resulting in smoothed estimates for the boundary of each subinterval. Finally the backward sweep of the Rauch-Tung-Striebel algorithm is used in each subinterval to produce globally smoothed estimates starting from the boundary of the subinterval and proceeding to its center.

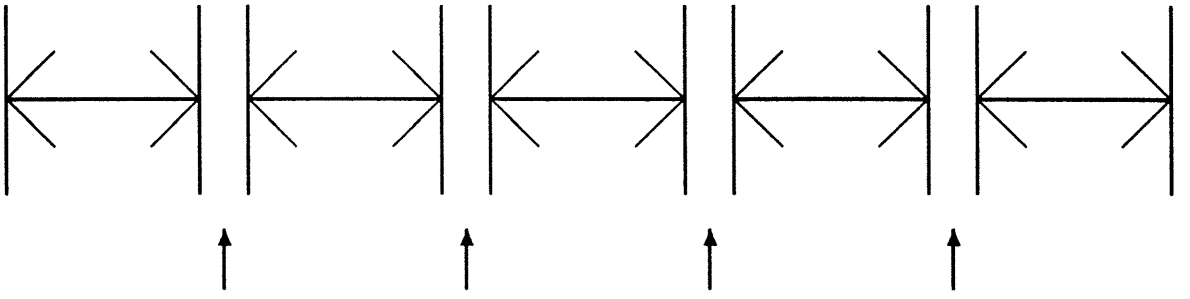
The first aim is to show that local processing can result in an equivalent but sampled process, where the local computations play the role of state observations.

Step 1

Let the local processors filter outward from the center of the interval to the boundary by applying the FMLF equations using the folded model indicated in equations (3.25)-(3.26) which was shown to be equivalent to (5.1). The local processors compute locally filtered estimates of the boundary based on local data and dynamic constraints. From the outward filtering process described in Section 4.4, the result of filtering to the boundaries from the center of each subinterval is

$$\begin{bmatrix} \hat{x}[k_l|k_l, k_{l+1}-1] \\ \hat{x}[k_{l+1}|k_l, k_{l+1}-1] \end{bmatrix} = \begin{bmatrix} P_{k_l k_l|k_l,k_{k+1}-1} & P_{k_l k_{l+1}|k_l,k_{k+1}-1} \\ P_{k_{l+1} k_l|k_l,k_{k+1}-1} & P_{k_{l+1} k_{l+1}|k_l,k_{k+1}-1} \end{bmatrix} \begin{bmatrix} x(k_l) \\ x(k_{l+1}) \end{bmatrix} + \begin{bmatrix} \tilde{x}[k_l|k_l, k_{l+1}-1] \\ \tilde{x}[k_{l+1}|k_l, k_{l+1}-1] \end{bmatrix} \tag{5.96}$$

Sampled System



Dynamic Constraints

Figure 5-4: This represents the system immediately after the local filtering which starts at the center of each subinterval and ends at the boundary of each subinterval has been performed. The the state of the system has dimension 2n, and the local estimates produced by each processor represents local estimates of the state. Linking the states together are the remaining constraints. If neighboring states do not intersect then the constraints are noisy and are a subset of the of the original set of descriptor equations (5.1). Otherwise neighboring states intersect and equality constraints exist (??)

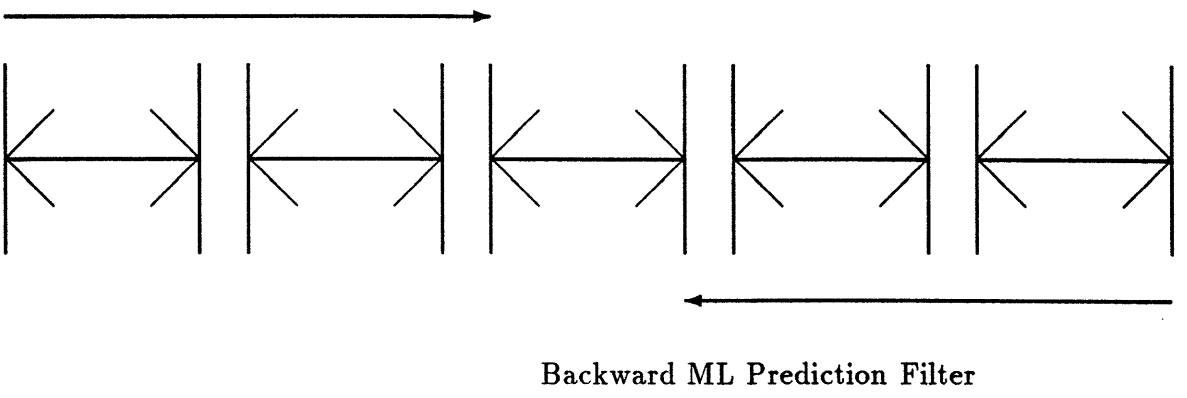


Figure 5-5: The interprocessor communication step consists of constructing two filters which will provide a measurement of the boundary for a given subinterval. The smoothed estimate of this boundary is constructed from the local measurements and the two measurements provided by the forward and backward filters.

where the matrix of $P_{i,j|k,l}$ represents the projection matrix associated with the estimate of the local boundary. If the system is estimable, and if the local subinterval is large enough, then the projection matrix which premultiplies the boundary $[x_{k_l} \;\; x_{k_{l+1}}]$ in (5.96) is the identity. Note that this differs from the algorithm discussed by Tewfik, et al., in that while both locally filter folded processes whose state is given by $[x^T(k) \;\; x^T(-k)]^T$, this model, as shown in Chapter 3, is simply a reordering of the dynamic constraints in (5.1), while the model which appears in Section 5.2 is an outwardly causal model. Our algorithm therefore avoids any cost associated with the construction of this outwardly causal model. The covariance of the estimation error in (5.96) is given by

$$\begin{bmatrix} \tilde{x}[k_l|k_l, k_{l+1}-1] \\ \tilde{x}[k_{l+1}|k_l, k_{l+1}-1] \end{bmatrix} \sim N \left[0; \begin{bmatrix} \Sigma_{\tilde{x}\tilde{x}}[k_l, k_l|k_l, k_{l+1}-1] & \Sigma_{\tilde{x}\tilde{x}}[k_l, k_{l+1}|k_l, k_{l+1}-1] \\ \Sigma_{\tilde{x}\tilde{x}}[k_{l+1}, k_l|k_l, k_{l+1}-1] & \Sigma_{\tilde{x}\tilde{x}}[k_{l+1}, k_{l+1}|k_l, k_{l+1}-1] \end{bmatrix} \right] \tag{5.97}$$

Step 2 The remaining constraints required to produce globally filtered estimates

follow the form

$$\begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} x(k_l) \\ x(k_{l+1}) \end{bmatrix} = \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} x(k_{l-1}) \\ x(k_l) \end{bmatrix} \tag{5.98}$$

The sampled system in (5.98) can be filtered using (5.96) as observations via the Mayne-Fraser algorithm to produce forward and backward filtered estimates and as a result produce smoothed estimates of the states of this sampled system. Figure 5-4 represents the system after the local filtering step, and the location of the remaining dynamic constraints.

By acknowledging that after filtering the system is a sampling of the original system, the issue of transmitting information from one processor to another is just an implementation of the Mayne-Fraser smoothing algorithm. In this case we propagate forward and backward predicted estimates of the 2n dimensional boundary $[x^T(k_l), x^T(k_{l+1})]$ and combine them with the estimate of the boundary which is produced by each subinterval processor. Given only past data (i.e. causal data excluding the measurement of the current boundary), at most an estimate can be constructed for $x(k_l)$ while no information is available about $x(k_{l+1})$. Therefore if we were to interpret (5.98) as a descriptor system whose state is 2n dimensional, and (5.96) as the observations for this system then a 'prediction' filter can at most estimate the n dimensional subvector for $[I, 0][x^T(k_l), x^T(k_{l+1})]^T$. As a result the interprocessor communication need only communicate n dimensional vectors, and $n \times n$ covariances, and projections instead of larger matrices and vectors. Figure 5-5 shows the application of the Mayne Fraser algorithm to the sampled system.

The smoothed estimate is obtained by combining n dimensional forward and backward predicted estimates of the $2n$ dimensional state with the local measurement of

the state. The FMLF for the sampled process is given by

$$\hat{x}[k_{l+1}|k_0, k_{l+1}-1] = \begin{bmatrix} 0 & 0 & 0 & 0 & I \end{bmatrix}$$
$$\begin{bmatrix} \Sigma_{\tilde{x}\tilde{x}}[k_l, k_l|k_l, k_{l+1}-1] & \Sigma_{\tilde{x}\tilde{x}}[k_l, k_{l+1}|k_l, k_{l+1}-1] & 0 & I & 0 \\ \Sigma_{\tilde{x}\tilde{x}}[k_{l+1}, k_l|k_l, k_{l+1}-1] & \Sigma_{\tilde{x}\tilde{x}}[k_{l+1}, k_{l+1}|k_l, k_{l+1}-1] & 0 & 0 & I \\ 0 & 0 & \Sigma_{\tilde{x}\tilde{x}}[k_l, k_l|k_0, k_l-1] & I & 0 \\ I & 0 & I & 0 & 0 \\ 0 & I & 0 & 0 & 0 \end{bmatrix}^{\#}$$
$$\times \begin{bmatrix} \hat{x}[k_l|k_l, k_{l+1}-1] & \hat{x}[k_{l+1}|k_l, k_{l+1}-1] & \hat{x}[k_l|k_{l-1}, k_l-1] & 0 & 0 \end{bmatrix}^T \tag{5.99}$$

where we assume that $x(k_{l+1})$ is estimable based on data over the interval from k_l to $k_{l+1}-1$. If $x(k_{l+1})$ is not estimable then projection matrices must be included in the computation. As an aside we will now include the equivalent computation for the case where $x(k_{l+1})$ is not estimable.

$$\begin{bmatrix} L_l & M_{l+1} & N_l & 0 & \Delta_{\hat{x}}[k_{l+1}|k_0, k_{l+1}-1] \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ I \end{bmatrix}^T \begin{bmatrix} R & H_1 & H_2 \\ H_1^T & 0 & 0 \\ H_2^T & 0 & 0 \end{bmatrix}^{\#} \tag{5.100}$$

where

$$R = \begin{bmatrix} \Sigma_{\tilde{x}\tilde{x}}[k_l, k_l|k_l, k_{l+1}-1] & \Sigma_{\tilde{x}\tilde{x}}[k_l, k_{l+1}|k_l, k_{l+1}-1] & 0 \\ \Sigma_{\tilde{x}\tilde{x}}[k_{l+1}, k_l|k_l, k_{l+1}-1] & \Sigma_{\tilde{x}\tilde{x}}[k_{l+1}, k_{l+1}|k_l, k_{l+1}-1] & 0 \\ 0 & 0 & \Sigma_{\tilde{x}\tilde{x}}[k_l, k_l|k_0, k_l-1] \end{bmatrix} \tag{5.101}$$

$$H_1 = \begin{bmatrix} P_{k_l k_l|k_l, k_{l+1}-1} \\ P_{k_{l+1} k_l|k_l, k_{l+1}-1} \\ P_{k_l|k_0, k_l-1} \end{bmatrix} \tag{5.102}$$

and finally

$$H_2 = \begin{bmatrix} P_{k_l k_{l+1}|k_l, k_{l+1}-1} \\ P_{k_{l+1} k_{l+1}|k_l, k_{l+1}-1} \\ 0 \end{bmatrix} \tag{5.103}$$

The projection matrix for the estimate of $x(k_{l+1})$ is computed from

$$\begin{aligned} P_{k_{l+1}|k_0,k_{l+1}-1} &= (L_l P_{k_l k_{l+1}|k_l,k_{l+1}-1} + M_l P_{k_{l+1}k_{l+1}|k_l,k_{l+1}-1})^{\#} \\ &\times (L_l P_{k_l k_{l+1}|k_l,k_{l+1}-1} + M_l P_{k_{l+1}k_{l+1}|k_l,k_{l+1}-1}) \end{aligned} \tag{5.104}$$

The estimate and covariance are given by

$$\hat{x}[k_{l+1}|k_0,k_{l+1}-1] = P_{k_{l+1}|k_0,k_{l+1}-1}(L_l\hat{x}[k_l|k_l,k_{l+1}-1]+M_l\hat{x}[k_{l+1}|k_l,k_{l+1}-1]+N_l\hat{x}[k_l|k_0,k_l-1]) \tag{5.105}$$

$$\Sigma_{\tilde{x}\tilde{x}}[k_{l+1}k_{l+1}|k_0,k_{l+1}-1] = P_{k_{l+1}|k_0,k_{l+1}-1}\Delta_{\hat{x}}[k_{l+1}k_{l+1}|k_0,k_{l+1}-1]P_{k_{l+1}|k_0,k_{l+1}-1} \tag{5.106}$$

Returning to the assumption that the state is estimable, the equations for the BMLF can be written in a manner similar to (5.99). For the estimable case, they are given by

$$\begin{aligned} \hat{x}[k_l|k_l,k_{L-1}] = &\begin{bmatrix} 0 & 0 & 0 & 0 & I \end{bmatrix} \times \\ &\begin{bmatrix} \Sigma_{\tilde{x}\tilde{x}}[k_l,k_l|k_l,k_{l+1}-1] & \Sigma_{\tilde{x}\tilde{x}}[k_l,k_{l+1}|k_l,k_{l+1}-1] & 0 & I & 0 \\ \Sigma_{\tilde{x}\tilde{x}}[k_{l+1},k_l|k_l,k_{l+1}-1] & \Sigma_{\tilde{x}\tilde{x}}[k_{l+1},k_{l+1}|k_l,k_{l+1}-1] & 0 & 0 & I \\ 0 & 0 & \Sigma_{\tilde{x}\tilde{x}}[k_{l+1},k_{l+1}|k_{l+1},k_{L-1}] & I & 0 \\ I & 0 & I & 0 & 0 \\ 0 & I & 0 & 0 & 0 \end{bmatrix}^{\#} \\ &\begin{bmatrix} \hat{x}[k_l|k_l,k_{l+1}-1] & \hat{x}[k_{l+1}|k_l,k_{l+1}-1] & \hat{x}[k_l|k_{l-1},k_l-1] & 0 & 0 \end{bmatrix}^T \end{aligned} \tag{5.107}$$

The BMLF can also be written for the case where the state is not estimable, which is similar to that provided to the FMLF. We will not provide these equations but refer the reader to equations (5.100) - (5.105) the equations for the FMLF as a model to construct the BMLF for the non estimable case. We will now continue with constructing the smoothed estimate for the the estimable case.

The smoothed estimate of the boundary of subregion denoted by $[\hat{x}_s^T(k_l)\hat{x}_s(k_{l+1})]^T$

can be constructed from

$$\begin{bmatrix} \hat{x}_s(k_l) \\ \hat{x}_s(k_{l+1}) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix}^T$$

$$\begin{bmatrix} \Sigma_{\tilde{x}\tilde{x}}[k_l,k_l|k_l,k_{l+1}] & \Sigma_{\tilde{x}\tilde{x}}[k_l,k_{l+1}|k_l,k_{l+1}] & 0 & 0 & I & 0 \\ \Sigma_{\tilde{x}\tilde{x}}[k_{l+1},k_l|k_l,k_{l+1}] & \Sigma_{\tilde{x}\tilde{x}}[k_{l+1},k_{l+1}|k_l,k_{l+1}] & 0 & 0 & 0 & I \\ 0 & 0 & \Sigma_{\tilde{x}\tilde{x}}[k_l,k_l|k_l,k_{L-1}] & 0 & I & 0 \\ 0 & 0 & 0 & \Sigma_{\tilde{x}\tilde{x}}[k_l,k_l|k_l,k_{L-1}] & 0 & I \\ I & 0 & I & 0 & 0 & 0 \\ 0 & I & 0 & I & 0 & 0 \end{bmatrix}^{\#} \quad (5.108)$$

$$\times \begin{bmatrix} \hat{x}[k_l|k_l,k_{l+1}] & \hat{x}[k_{l+1}|k_l,k_{l+1}-1] & \hat{x}[k_l|k_0,k_l] & \hat{x}[k_{l+1}|k_{l+1},k_{L-1} & 0 & 0 \end{bmatrix}^T$$

when the state is estimable. When the state is not estimable it can be constructed from

$$\begin{bmatrix} \hat{x}_s(k_l) \\ \hat{x}_s(k_{l+1}) \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}^T \begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix}^{\#} \begin{bmatrix} Y \\ 0 \end{bmatrix} \quad (5.109)$$

where

$$R = \begin{bmatrix} \Sigma_{\tilde{x}\tilde{x}}[k_l,k_l|k_l,k_{l+1}] & \Sigma_{\tilde{x}\tilde{x}}[k_l,k_{l+1}|k_l,k_{l+1}] & 0 & 0 \\ \Sigma_{\tilde{x}\tilde{x}}[k_{l+1},k_l|k_l,k_{l+1}] & \Sigma_{\tilde{x}\tilde{x}}[k_{l+1},k_{l+1}|k_l,k_{l+1}] & 0 & 0 \\ 0 & 0 & \Sigma_{\tilde{x}\tilde{x}}[k_l,k_l|k_l,k_{L-1}] & 0 \\ 0 & 0 & 0\Sigma_{\tilde{x}\tilde{x}}[k_l,k_{l+1}|k_l,k_{l+1}] & \end{bmatrix} \quad (5.110)$$

$$H = \begin{bmatrix} P_{k_l,k_l|k_l,k_{l+1}} & P_{k_l,k_{l+1}|k_l,k_{l+1}} \\ P_{k_{l+1},k_l|k_l,k_{k+1}} & P_{k_{l+1},k_{l+1}|k_l,k_{l+1}} \\ P_{k_l,k_l|k_l,k_{L-1}} & 0 \\ 0 & P_{k_{l+1},k_{l+1}|k_l,k_{L-1}} \end{bmatrix} \quad (5.111)$$

and

$$Y = \begin{bmatrix} \hat{x}[k_l|k_l, k_{l+1}] \\ \hat{x}[k_{l+1}|k_l, k_{l+1}-1] \\ \hat{x}[k_l|k_0, k_l] \\ \hat{x}[k_{l+1}|k_{l+1}, k_{L-1}] \end{bmatrix} \tag{5.112}$$

Equivalently the local processing can produce forward and backward filtered estimates while leaving the remaining dynamic constraints for the subsampled process. The local processors can produce

$$\begin{bmatrix} \hat{x}_{ML}[k_l|k_l, k_{l+1}-1] \\ \hat{x}_{ML}[k_{l+1}-1|k_l, k_{l+1}-1] \end{bmatrix} = \begin{bmatrix} x(k_l) \\ x(k_{l+1}) \end{bmatrix} + \begin{bmatrix} \tilde{x}_{ML}[k_l|k_l, k_{l+1}-1] \\ \tilde{x}_{ML}[k_{l+1}-1|k_l, k_{l+1}-1] \end{bmatrix} \tag{5.113}$$

The resulting subsampled process after the local processors have performed their filtering is given by

$$\begin{bmatrix} E_{k_l} & 0 \end{bmatrix} \begin{bmatrix} x(k_l) \\ x(k_{l+1}-1) \end{bmatrix} = \begin{bmatrix} 0 & A_{k_l-1} \end{bmatrix} \begin{bmatrix} x(k_{l-1}) \\ x(k_l-1) \end{bmatrix} + B_{k_l-1}u(k_l-1) \tag{5.114}$$

Since the process reduces to a filtering problem of a subsampled process using locally outward filtered boundaries as the state observations, it is clear that the new problem is self similar to the original problem. The Mayne-Fraser algorithm therefore represents a linear non-parallel solution to the sampled process. We therefore can consider the possibility of computing the smoothed estimate of the sampled process in parallel. The above algorithm can therefore be exploited recursively to obtain an algorithm with logarithmic time. In the next section we consider this concept incorporating the highest degree of parallelism.

Step 3 Since the elements of this sampled system are the boundaries of the local subregions, the local subregions can be smoothed via the backward sweep of the Rauch-Tung-Striebel algorithm for the local models given by (3.25)- (3.26).

An alternate approach to looking at this problem is to examine exactly what information is needed to compute smoothed estimate. We take the STPBVDS and

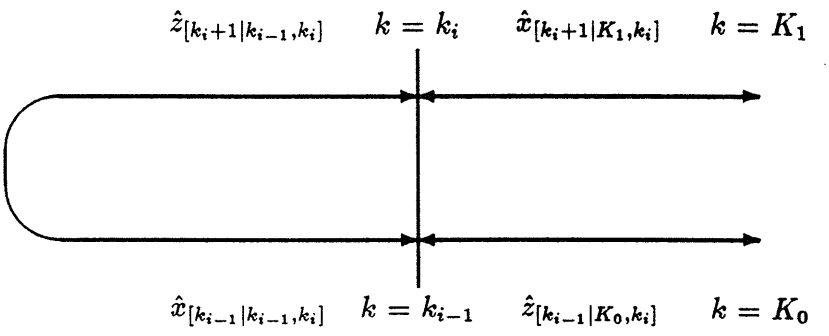


Figure 5-6: Combining estimates at $\kappa = \pm k_o$

fold it over and relabel the states such that the state is now $[x^T(k) \;\; x^T(-k)]^T$. The new system, as in (3.25) is separable and we therefore know that the smoothed estimate is provided by the Mayne-Fraser two filter algorithm discussed in Section 4.4. Figure 5.5 illustrates this example. The boundary condition at $k = \pm K$ is decoupled because the original system is separable. As a result the backward filter for the 2n dimensional system can be decoupled into two subsystems, with one filter for each leg. Ultimately, the estimates provided by these filters can be computed from outward filters themselves propagating outward from the centers of the partitions formed by each leg. What was originally a two partition algorithm results in a three partition algorithm. The smoothed estimate is obtained by combining three estimates. The joint estimate provided by the local processing from the center processor (5.96), and the estimate of the boundary given by

$$\hat{x}_{ML}[k_i|K_1, k_i] \tag{5.115}$$

$$\hat{x}_{ML}[k_i|K_1, k_i] \tag{5.116}$$

which are provided by the processors working on each leg on the right side of Figure 5.3. An advantage of this view of combining data at the boundaries is that by considering the analogy to the Mayne-Fraser filter, the information required to compute the estimates are made clear. The interprocessor exchange step is seen to combine three of the four measurements required to compute the smoothed estimate at the boundaries of the subinterval. The filter which propagates the estimate from

processor to processor is obtained simply by removing either (5.115), or (5.116) from consideration. What results is the forward filter or a backward filter, identical to that obtained when filtering the sampled process in (5.98).

5.5.1 Complexity

The computation involved in this algorithm depends on the fact that the state, though twice what single processor would require, only propagates over half the interval. The dimension of the data is $2p$ and the dimension of the driving noise is $2m$. While the structure of this algorithm is similar to that in Section 5.2, there is no prior processing used to generate a 'preferred' model.

Step 1

Off-line

Since filtering begins from the center and proceeds to the boundaries the amount of computation which a given processor acting on a single region of data must execute is given by

$$\begin{aligned} &\tfrac{1}{2}\tfrac{K}{L}\mathcal{K}(2n, 2m, 2p; \text{off} - \text{line}, \text{non} - \text{estimable}) \\ &= \tfrac{1}{2}\tfrac{K}{L}509.33n^3 + 448n^2p + 32n^2m + 168np^2 + 21.33p^3 \end{aligned} \tag{5.117}$$

On-line

The online computation required is given by

$$\begin{aligned} &\tfrac{K}{2L}\mathcal{K}(2n, 2p; on - line) \\ &= \tfrac{K}{2L}8n^2 + 8np \end{aligned} \tag{5.118}$$

Step 2

Off-line

The off-line computation required for the interprocessor communication is given by

$$(L-1)2[\mathcal{M}(5n, 2n, 5n) + \mathcal{E}(n, n, n) + 2n^3] = (L-1)2[589.67n^3] \tag{5.119}$$

where the factor of two indicates that the computation needs to be carried out both for the forward filter and the backward filter.

On-line

The on-line computation required for the interprocessor communication is given by

$$(L-1)6n^2 \tag{5.120}$$

Step 3

Off-line

The computations here are the backward sweep of the Rauch-Tung-Striebel algorithm. The off-line computations are given by

$$\begin{gathered}\tfrac{1}{2}\tfrac{K}{L}\mathcal{T}(2n,2m;\mathrm{off}-\mathrm{line},\mathrm{non}-\mathrm{causally}-\mathrm{estimable}) \\ \tfrac{1}{2}\tfrac{K}{L}941.33n^3+32n^2m\end{gathered} \tag{5.121}$$

On-line

The amount of on-line computations are given by

$$\begin{gathered}\tfrac{K}{2L}\mathcal{T}(2n;\mathrm{on}-\mathrm{line}) \\ \tfrac{K}{2L}16n^2\end{gathered} \tag{5.122}$$

Here we have assumed that the state is estimable given all of the data but not estimable given only causal or anticausal data. In summary, the off-line time required for computation is given by

$$\begin{aligned}T &= \tfrac{1}{2}\tfrac{K}{L}[\mathcal{K}(2n,2m,2p;\mathrm{off}-\mathrm{line},\mathrm{non}-\mathrm{estimable})+\mathcal{T}(2n,2m;\mathrm{off}-\mathrm{line})] \\ &\quad +(L-1)2[\mathcal{M}(5n,2n,5n)+\mathcal{E}(n,n,n)+2n^3] \\ &= \tfrac{K}{2L}1450.67n^3+448n^2p+64n^2m+168np^2+21.33p^3+(L-1)2[589.67n^2]\end{aligned} \tag{5.123}$$

The total time for the on-line computations are given by

$$\tfrac{K}{2L}[\mathcal{K}(2n,2p;\mathrm{on}-\mathrm{line})+\mathcal{T}(2n,2p;\mathrm{on}-\mathrm{line})]+(L-1)6n^2 \tag{5.124}$$

5.6 Parallel ML Smoothing smoothing with Binary Tree Interconnections

The algorithms discussed in detail in the previous sections basically have the following structure. Processing in the local subregions produce local estimates of the boundaries given local data. The interprocessor communication step basically amounts to implementing a two filter algorithm on a sampled system using the results of the local processing as observations and including in the computation, if necessary, any additional dynamic constraints which were not used in the local processing. Once the information required to obtained smoothed estimates has been propagated to each boundary point, the points interior to each subinterval can be updated to their smoothed values. Once we recognize that the interprocessor communication step is a two filter algorithm, not unlike the Mayne-Fraser algorithm, it is clear that the interprocessor communication step can be implemented in parallel. This approach towards parallelization suggests a different connectivity between the processors. Figure 5-7 is a possible architecture when the filters for the interprocessor communication step are implemented in parallel. We will, however take a slightly different approach to the issue of parallel processing. We will map the smoothing problem directly to the binary tree by analyzing the parallel smoothing problem of a region divided into two subregions. This approach allows a priori information in the form of a two point boundary condition (and in fact certain multi-point boundary conditions) can be incorporated quite easily into the smoothing algorithm. Since Mayne-Fraser algorithms are perfectly suited for STPBVDS's it follows that these algorithms are not limited to STPBVDS's, they can also be applied directly to TPBVDS's. Furthermore, additional measurements of local boundaries can be included and the possibility exists to adapt the algorithm to a wide variety of measurements of subregion boundaries. We will find that the algorithm presented has similarities to the multi-resolution algorithms discussed by Chou[26]. Furthermore, it will be shown that the methodology in this section is completely generalizable to smoothing algorithms for multidimensional processes.

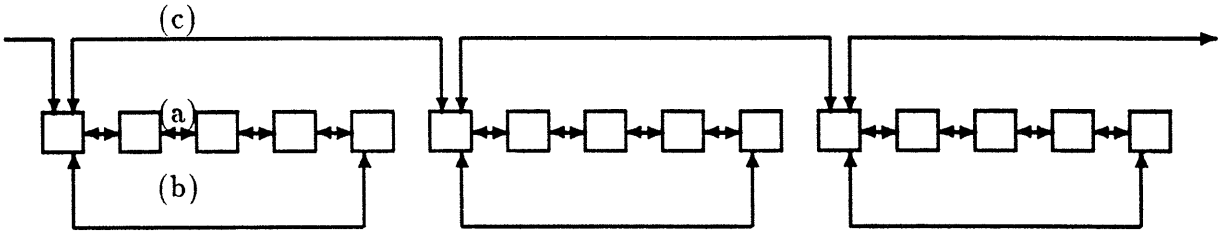


Figure 5-7: Each processor in the parallel algorithm in Section 5.5 produced boundary measurements for its region which are communicated to neighboring processors. Each block represents a processor operating on a specified region and local boundary measurements are communicated to neighboring processors using communication links such as the one indicated by (a). In order to gain greater efficiencies local regions can be grouped in clusters where one processor is responsible for obtaining the boundary measurement for the entire region covered by a cluster of processors. This processor can obtain this information from communication links (a), and (b). Finally all processors which are responsible for obtaining the boundary information from a cluster of processors can then communicate among themselves to produce the optimal boundary estimate for each cluster. Then each cluster in parallel can work to disseminate the optimal cluster boundary information to produce optimal estimates of local boundaries. Finally the optimal boundary information of each local boundary can be used by each processor in parallel to update interior points of each subregion.

We will demonstrate that the parallel smoothing algorithm can be written precisely as the Rauch-Tung-Striebel algorithm when the proper definition of state and notion of time is provided. Since the backward sweep of the Rauch-Tung-Striebel algorithm amounts to the construction of an anticausal (Markov) model for the smoothed process, it can be shown as a direct result that reciprocal processes can be modeled by processes which live on trees. This further suggests new algorithms which take advantage of this structure. We will see in Chapter 6 that multidimensional processes also have tree structured models associated with them.

The systems which we are examining are reciprocal, that is given a closed contour, the interior and exterior are independent. On a line, we define a closed contour to be a boundary consisting of two points. We define the 'state' of the system to be the value of the process on a closed contour. Thus for the points s_1 and s_2 we will define the state to be $[x^T(s_1), x^T(s_2)]$, not unlike the definition of state in Chapter 3 where STPBVDS's are constructed from TPBVDS's. There, the definition of state is is defined on the closed contour given by $\{s, -s\}$, here, we consider a

much larger set of pairs $\{s_1, s_2\}$. Finally, we define an order operation $\preceq$ on the set of contours. Specifically, the contour $\{s_1, s_2\}$ is *interior* to the contour $\{s_3, s_4\}$, i.e., $\{s_1, s_2\} \preceq \{s_3, s_4\}$ if $s_3 \leq s_1 \leq s_4$ and $s_3 \leq s_2 \leq s_4$. We use this definition later and in Chapter 6 when discussing the Markov nature of these algorithms. Before describing a general algorithm for smoothing TPBVDS's we provide two examples of smoothing the following acausal model for a vector process.

$$\begin{aligned} \Gamma_e(t+1)x(t+1) &= \Gamma_a(t)x(t) + u(t) \qquad 0 \leq t \leq T-1 \\ y(t) &= x(t) + v(t) \qquad 0 \leq t \leq T \\ \Gamma_e(0)x(0) &= \Gamma_a(T)x(T) + u(T) \end{aligned} \tag{5.125}$$

where $u(t)$ and $v(t)$ are independent white noise sequences with the following statistics

$$\begin{aligned} u(t) &\sim N(0, q(t)) \\ v(t) &\sim N(0, r(t)) \\ \det q(t) &\neq 0 \qquad \forall\, t \\ \det r(t) &\neq 0 \qquad \forall\, t \end{aligned} \tag{5.126}$$

The two examples which follow differ with respect to the set of noisy constraints which are used in Step 1, the initial processing step. In contrast to all other algorithms discussed in this chapter, we do not consider forward and backward filters operating on the boundaries of the subregions. There are only two subregions in both examples and the calculations involved in updating boundaries can be performed in one computation.

Example 5.1

Here we consider the case where $T = 3$. The data is divided between Processor # 0 and Processor # 1.

Step 1:

In the local processing step, Processor #0 takes the measurements given by

$$\begin{bmatrix} y(0) \\ 0 \\ y(1) \end{bmatrix} = \begin{bmatrix} I & 0 \\ \Gamma_a(0) & -\Gamma_e(1) \\ 0 & I \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \end{bmatrix} + \begin{bmatrix} v(0) \\ u(0) \\ v(1) \end{bmatrix} \tag{5.127}$$

to compute the estimate

$$\begin{bmatrix} \hat{x}[0|[0,1]] \\ \hat{x}[1|[0,1]] \end{bmatrix} = \begin{bmatrix} x(0) \\ x(1) \end{bmatrix} + \begin{bmatrix} \tilde{x}[0|[0,1]] \\ \tilde{x}[1|[0,1]] \end{bmatrix} \tag{5.128}$$

where the notation $[s,t]$ indicate that the estimates are based on all of the dynamic constraints linking $x(s)$ to $x(t)$ and all of the observations $y(s)$ through $y(t)$. The error covariance is given by

$$\text{Cov} \begin{bmatrix} \tilde{x}[0|[0,1]] \\ \tilde{x}[1|[0,1]] \end{bmatrix} = \begin{bmatrix} \Sigma_{00|[0,1]} & \Sigma_{01|[0,1]} \\ \Sigma_{10|[0,1]} & \Sigma_{11|[0,1]} \end{bmatrix} \tag{5.129}$$

As an aid in understanding the structure of this algorithm we will use Figure 5-8 to represent the merging of the estimates of $x(0)$ and $x(1)$ (which are provided in this case by $y(0)$ and $y(1)$), and the dynamic constraint which links them. We include $u(0)$ in the figure to make clear which dynamic constraint is merged with which observations. In the case of Figure (5-8) the observations $y(0)$ and $y(1)$ are combined with the dynamic constraint indicated by $u(0)$. While these computations are being performed, Processor #1 performs an analogous computation for $x(2)$ and $x(3)$. Processor #1 takes the measurements given by

$$\begin{bmatrix} y(2) \\ 0 \\ y(3) \end{bmatrix} = \begin{bmatrix} I & 0 \\ \Gamma_a(2) & -\Gamma_e(3) \\ 0 & I \end{bmatrix} \begin{bmatrix} x(2) \\ x(3) \end{bmatrix} + \begin{bmatrix} v(2) \\ u(2) \\ v(3) \end{bmatrix} \tag{5.130}$$

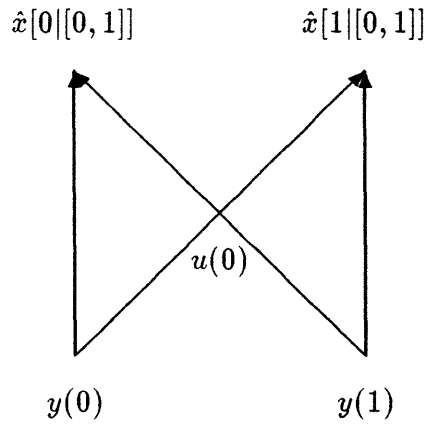


Figure 5-8: Icon representing the combining of two neighboring estimates using the dynamic constraint which links them

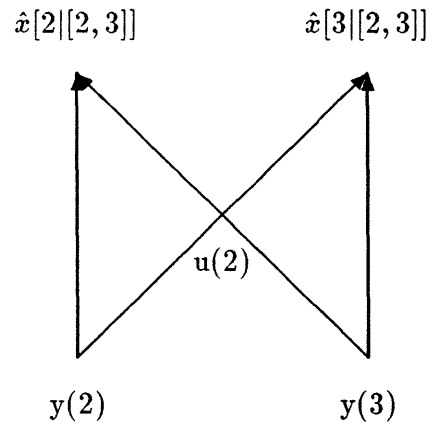


Figure 5-9: Computation of Processor #1

to compute the estimate

$$\begin{bmatrix} \hat{x}[2|[2,3]] \\ \hat{x}[3|[2,3]] \end{bmatrix} = \begin{bmatrix} x(2) \\ x(3) \end{bmatrix} + \begin{bmatrix} \tilde{x}[2|[2,3]] \\ \tilde{x}[3|[2,3]] \end{bmatrix} \tag{5.131}$$

where the error covariance is given by

$$\text{Cov} \begin{bmatrix} \tilde{x}[2|[2,3]] \\ \tilde{x}[3|[2,3]] \end{bmatrix} = \begin{bmatrix} \Sigma_{2|[2,3]} & \Sigma_{23|[2,3]} \\ \Sigma_{32|[2,3]} & \Sigma_{33|[2,3]} \end{bmatrix} \tag{5.132}$$

We will use Figure 5-9 to represent this estimation procedure.

Step 2:

The interprocessor communication step has several parts. For this example, Step 2

of the parallel algorithm has three parts which we will now outline.

Part 1

The first part Step 2 consists of using the remaining dynamic constraint, $0 = \Gamma_a(1)x(1) - \Gamma_e(2)x(2) = u(1)$, to produce a locally smoothed estimate of the boundary which encloses the two regions. This boundary is equal to $[x^T(0),\ x^T(3)]$. The measurements needed to produce the boundary estimate for $[x^T(0),\ x^T(3)]$ is given by

$$\begin{bmatrix} \hat{x}[0|[0,1]] \\ \hat{x}[1|[0,1]] \\ 0 \\ \hat{x}[2|[2,3]] \\ \hat{x}[3|[2,3]] \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & \Gamma_a(1) & -\Gamma_e(2) & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} + \begin{bmatrix} \tilde{x}[0|[0,1]] \\ \tilde{x}[1|[0,1]] \\ u(1) \\ \tilde{x}[2|[2,3]] \\ \tilde{x}[3|[2,3]] \end{bmatrix} \tag{5.133}$$

resulting in the estimate

$$\begin{bmatrix} \hat{x}[0|[0,3]] \\ \hat{x}[1|[0,3]] \\ \hat{x}[2|[0,3]] \\ \hat{x}[3|[0,3]] \end{bmatrix} = \begin{bmatrix} x(0) \\ x(1) \\ x(2) \\ x(3) \end{bmatrix} + \begin{bmatrix} \tilde{x}[0|[0,3]] \\ \tilde{x}[1|[0,3]] \\ \tilde{x}[2|[0,3]] \\ \tilde{x}[3|[0,3]] \end{bmatrix} \tag{5.134}$$

The error covariance for this estimate is given by

$$\mathrm{Cov} \begin{bmatrix} \tilde{x}[0|[0,3]] \\ \tilde{x}[1|[0,3]] \\ \tilde{x}[2|[0,3]] \\ \tilde{x}[3|[0,3]] \end{bmatrix} = \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{01|[0,3]} & \Sigma_{02|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{10|[0,3]} & \Sigma_{11|[0,3]} & \Sigma_{12|[0,3]} & \Sigma_{13|[0,3]} \\ \Sigma_{20|[0,3]} & \Sigma_{21|[0,3]} & \Sigma_{22|[0,2]} & \Sigma_{23|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{31|[0,3]} & \Sigma_{32|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix} \tag{5.135}$$

This estimation step can be interpreted as combining the estimates of the process at two neighboring points, $x(1)$ and $x(2)$ and using the information contained in the estimates at these points to update the boundary enclosing the two regions. Figure 5-10 is a graph which we will use to represent this operation. We call this an estimation module because it is the basic building block for larger and larger

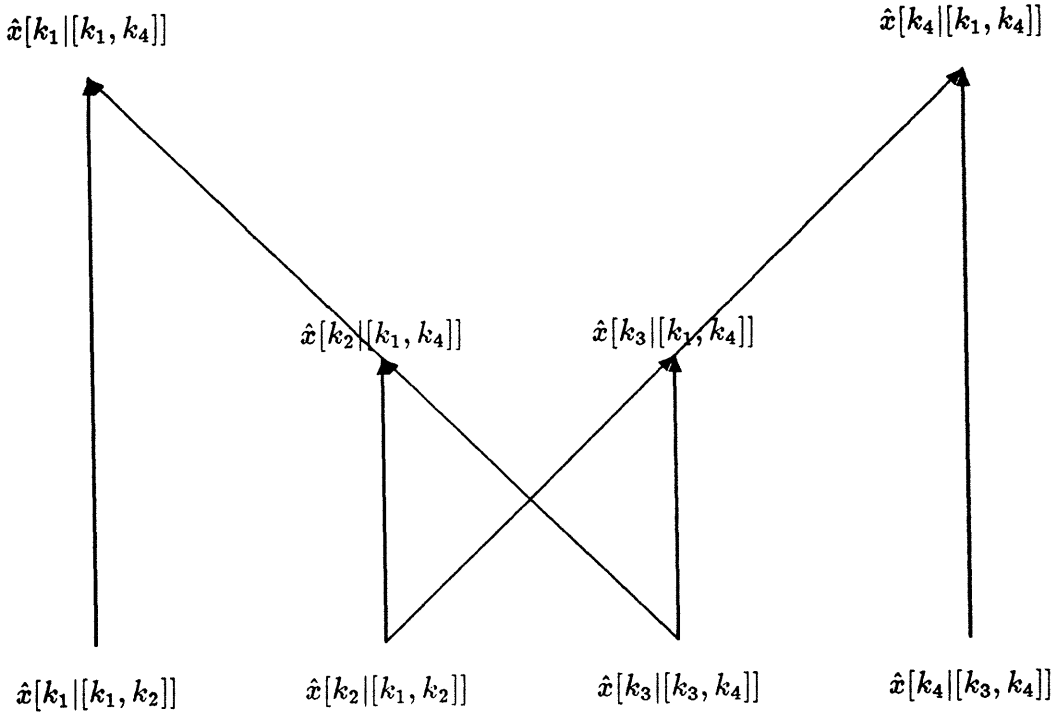


Figure 5-10: Using estimates of $x(k_2)$ and $x(k_3)$ to update $x(k_1)$ and $x(k_4)$ where $k_3 = k_2 + 1$. Specifically, the estimates $\hat{x}[k_2|[k_1,k_2]]$, and $\hat{x}[k_3|[k_3,k_4]]$ are combined to produce the estimate of $x(k_2)$ and $x(k_3)$ given all of the data denoted by $\hat{x}[k_2|[k_1,k_4]]$ and $\hat{x}[k_3|[k_1,k_4]]$ respectively. The correlation between the estimation errors $\tilde{x}[k_1|[k_1,k_2]]$, and $\tilde{x}[k_2|[k_1,k_4]]$ is then used to update $\hat{x}[k_1|[k_1,k_4]]$ to the smoothed value given by $\hat{x}[k_1|[k_1,k_4]]$. This represents the basic building block for the algorithm in this section.

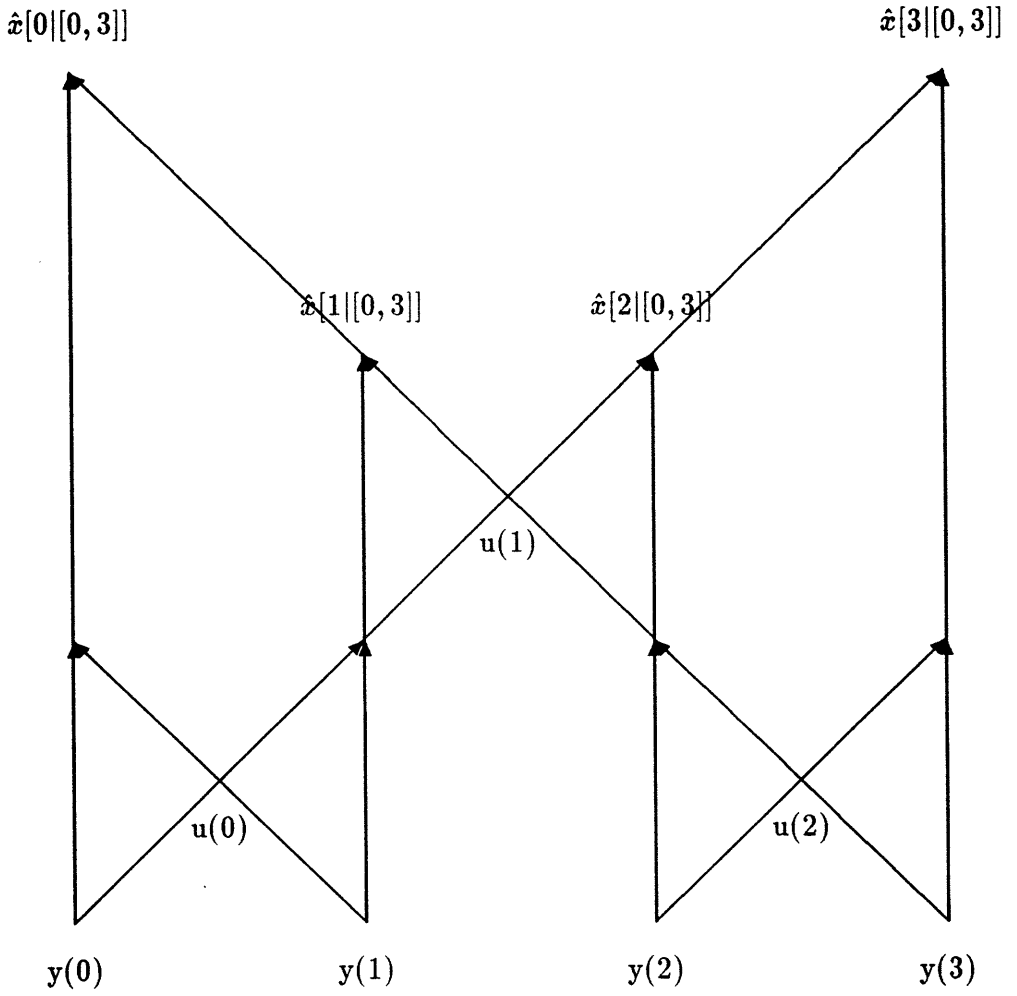


Figure 5-11: Arriving at the smoothed estimate for the boundary of the entire region

estimation algorithms.

Part 2

The estimates of the boundary states are then combined with the two point boundary condition to produce the smoothed estimates of the boundary. The measurements required to produce this smoothed estimate is given by

$$\begin{bmatrix} \hat{x}[0|[0,3]] \\ \hat{x}[3|[0,3]] \\ 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \\ -\Gamma_e(0) & \Gamma_a(3) \end{bmatrix} \begin{bmatrix} x(0) \\ x(3) \end{bmatrix} + \begin{bmatrix} \tilde{x}[0|[0,3]] \\ \tilde{x}[3|[0,3]] \\ u(3) \end{bmatrix} \tag{5.136}$$

The smoothed estimate which result from (5.136) are given by

$$\begin{bmatrix} \hat{x}_s(0) \\ \hat{x}_s(3) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(3) \end{bmatrix} + \begin{bmatrix} \tilde{x}_s(0) \\ \tilde{x}_s(3) \end{bmatrix} \tag{5.137}$$

The smoothed error covariance for the boundary is given by

$$\text{Cov} \begin{bmatrix} \tilde{x}_s(0) \\ \tilde{x}_s(3) \end{bmatrix} = \begin{bmatrix} \Sigma_{00} & \Sigma_{03} \\ \Sigma_{03} & \Sigma_{33} \end{bmatrix} \tag{5.138}$$

Part 3

The final part in Step 2 is to propagate smoothed estimates to the boundaries of the individual subregions, that is, to propagate estimates from the state $[x^T(0), x^T(3)]$ to the the states $[x^T(0), x^T(1)]$ and $[x^T(2), x^T(3)]$. The smoothed estimate of the boundaries of the subregions are provided by

$$\begin{bmatrix} \hat{x}_s(0) \\ \hat{x}_s(1) \end{bmatrix} = \begin{bmatrix} \hat{x}[0|[0,3]] \\ \hat{x}[1|[0,3]] \end{bmatrix} + \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{10|[0,3]} & \Sigma_{13|[0,3]} \end{bmatrix} \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix}^{-1} \times \left(\begin{bmatrix} \hat{x}_s(0) \\ \hat{x}_s(3) \end{bmatrix} - \begin{bmatrix} \hat{x}[0|[0,3]] \\ \hat{x}[3|[0,3]] \end{bmatrix} \right) \tag{5.139}$$

and

$$\begin{bmatrix} \hat{x}_s(2) \\ \hat{x}_s(3) \end{bmatrix} = \begin{bmatrix} \hat{x}[2|[0,3]] \\ \hat{x}[3|[0,3]] \end{bmatrix} + \begin{bmatrix} \Sigma_{20|[0,3]} & \Sigma_{23|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix} \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix}^{-1} \times \left(\begin{bmatrix} \hat{x}_s(0) \\ \hat{x}_s(3) \end{bmatrix} - \begin{bmatrix} \hat{x}[0|[0,3]] \\ \hat{x}[3|[0,3]] \end{bmatrix} \right) \tag{5.140}$$

The smoothed error covariances for each boundary are given by

$$\begin{bmatrix} \Sigma_{00} & \Sigma_{01} \\ \Sigma_{10} & \Sigma_{11} \end{bmatrix} = \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{01|[0,3]} \\ \Sigma_{10|[0,3]} & \Sigma_{11|[0,3]} \end{bmatrix} - \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{10|[0,3]} & \Sigma_{13|[0,3]} \end{bmatrix} \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix}^{-1} \times \left(\begin{bmatrix} \Sigma_{00} & \Sigma_{03} \\ \Sigma_{30} & \Sigma_{33} \end{bmatrix} - \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix} \right) \times \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix}^{-1} \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{10|[0,3]} \\ \Sigma_{03|[0,3]} & \Sigma_{31|[0,3]} \end{bmatrix} \tag{5.141}$$

and

$$\begin{bmatrix} \Sigma_{22} & \Sigma_{23} \\ \Sigma_{32} & \Sigma_{33} \end{bmatrix} = \begin{bmatrix} \Sigma_{22|[0,3]} & \Sigma_{23|[0,3]} \\ \Sigma_{32|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix} - \begin{bmatrix} \Sigma_{20|[0,3]} & \Sigma_{23|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix} \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix}^{-1}$$
$$\times \left(\begin{bmatrix} \Sigma_{00} & \Sigma_{03} \\ \Sigma_{30} & \Sigma_{33} \end{bmatrix} - \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix} \right)$$
$$\times \begin{bmatrix} \Sigma_{00|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{30|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix}^{-1} \begin{bmatrix} \Sigma_{02|[0,3]} & \Sigma_{03|[0,3]} \\ \Sigma_{32|[0,3]} & \Sigma_{33|[0,3]} \end{bmatrix} \tag{5.142}$$

respectively.

Figure 5-12 is used to represent the transference of smoothed estimates from the boundary surrounding the two regions to smoothed estimates of the sub-boundaries.

Step 3

When the algorithm is applied to this simple problem there is no Step 3 because each subregion consists only of its boundary; there are no interior states to update. For general TPBVDS's, the algorithm can always be implemented where the smallest subregions consists of non overlapping pairs of points. States however may not be fully estimable and more complicated equations are needed to deal with this case. We will deal with this case later in this section. If the system is uniformly estimable, we may perform outward filtering in local subregions until we can construct a boundary which has a well defined covariance. Step 2 in this algorithm can be executed until the boundaries of the local subregions have smoothed estimates then finally the Rauch-Tung-Striebel algorithm can be used to smooth interior states in each subregion in parallel.

In Part 2 of Step 2, boundary conditions are incorporated. The result of including the boundary condition is a smoothed estimate of the boundary. This represents the smoothed information which this module may expect from the remainder of a larger algorithm in which this module is embedded. In addition, we can include additional boundary measurements of subintervals throughout the algorithm without altering

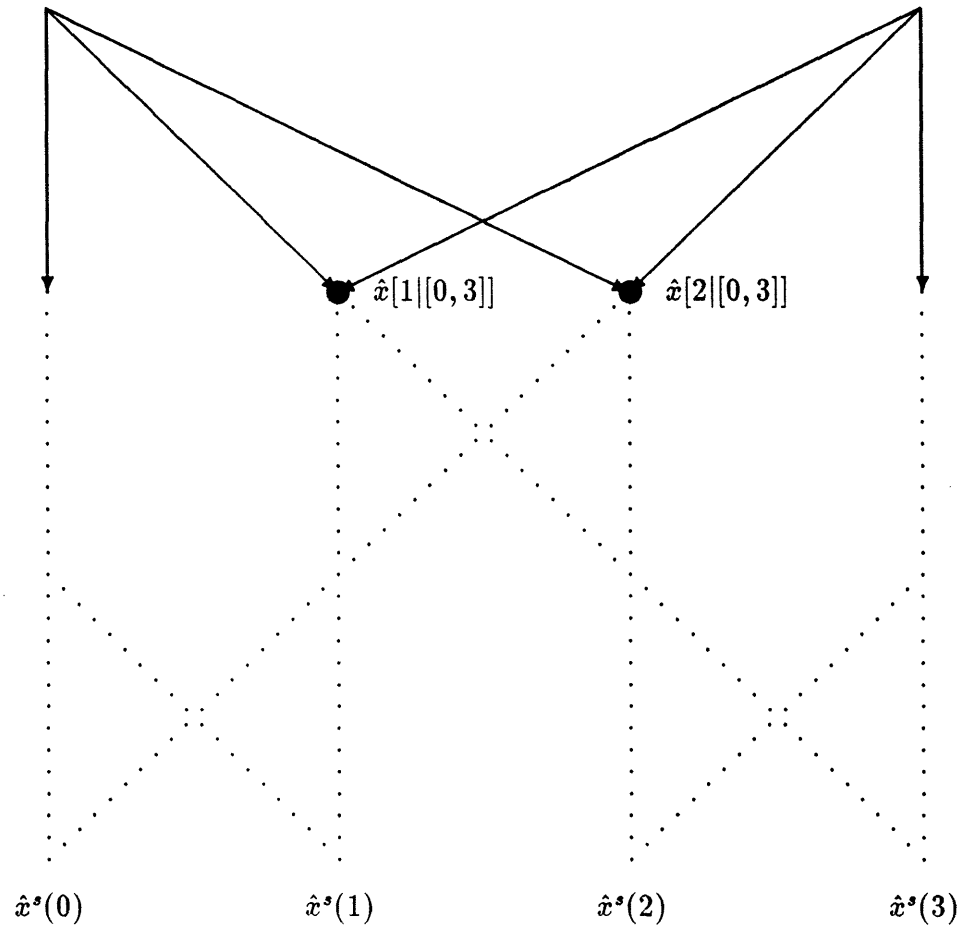


Figure 5-12: Additional boundary information in the form of the smoothed estimates $\hat{x}^s(0)$ and $\hat{x}^s(3)$ are combined with two estimates $\hat{x}[1|[0,3]]$ and $\hat{x}[2|[0,3]]$, which are represented by the large dots, to produce the smoothed estimates $\hat{x}^s(1)$ and $\hat{x}^s(2)$. The dotted lined indicate the flow of data which was used to construct the estimates $\hat{x}[1|[0,3]]$ and $\hat{x}[2|[0,3]]$. The additional boundary information can be obtained by viewing the figure to be a part of an algorithm operating on a larger set of data as in Figure 5-13

the basic structure of this smoothing algorithm. For example when the algorithm is applied to 16 points as shown in Figure 5-14 additional measurements such as

$$\begin{aligned} y_{03} &= C^0_{0,3}x(0) + C^3_{0,3}x(3) + v_{0,3} \\ y_{8,15} &= C^8_{8,15}x(8) + C^{15}_{8,15}x(15) + v_{8,15} \end{aligned} \tag{5.143}$$

can be included which cannot be neatly included in the standard Kalman filtering framework. It also suggests that the choice of subintervals can be chosen to accommodate the available measurements.

We are able to combine the estimation modules which represent this example to construct estimation algorithms which exist over larger and larger data sets. Figure 5-13 shows a graphical representation of the algorithm for $T = 7$ and Figure 5-14 shows a graphical representation of the algorithm for $T = 15$. In this example, dynamic constraints are incorporated at selected points throughout the algorithm to link neighboring boundaries. In the next example, all of the dynamic constraints are used in Step 1 while the the remaining steps include the information that neighboring boundaries intersect. □

Example 5.2

Here we consider the case where $T = 2$. In addition we will assume that $\Gamma_e(t) = I$. These systems are called two point boundary value systems and are discussed in Adams [1]. The data is divided between two processors.

Step 1:

In the local processing step, Processor #0 takes the measurements given by

$$\begin{bmatrix} y(0) \\ 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ \Gamma_a(0) & -I \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \end{bmatrix} + \begin{bmatrix} v(0) \\ u(0) \end{bmatrix} \tag{5.144}$$

to compute the estimate

$$\begin{bmatrix} \hat{x}[0|[0,1)] \\ \hat{x}[1|[0,1)] \end{bmatrix} = \begin{bmatrix} x(0) \\ x(1) \end{bmatrix} + \begin{bmatrix} \tilde{x}[0|[0,1)] \\ \tilde{x}[1|[0,1)] \end{bmatrix} \tag{5.145}$$

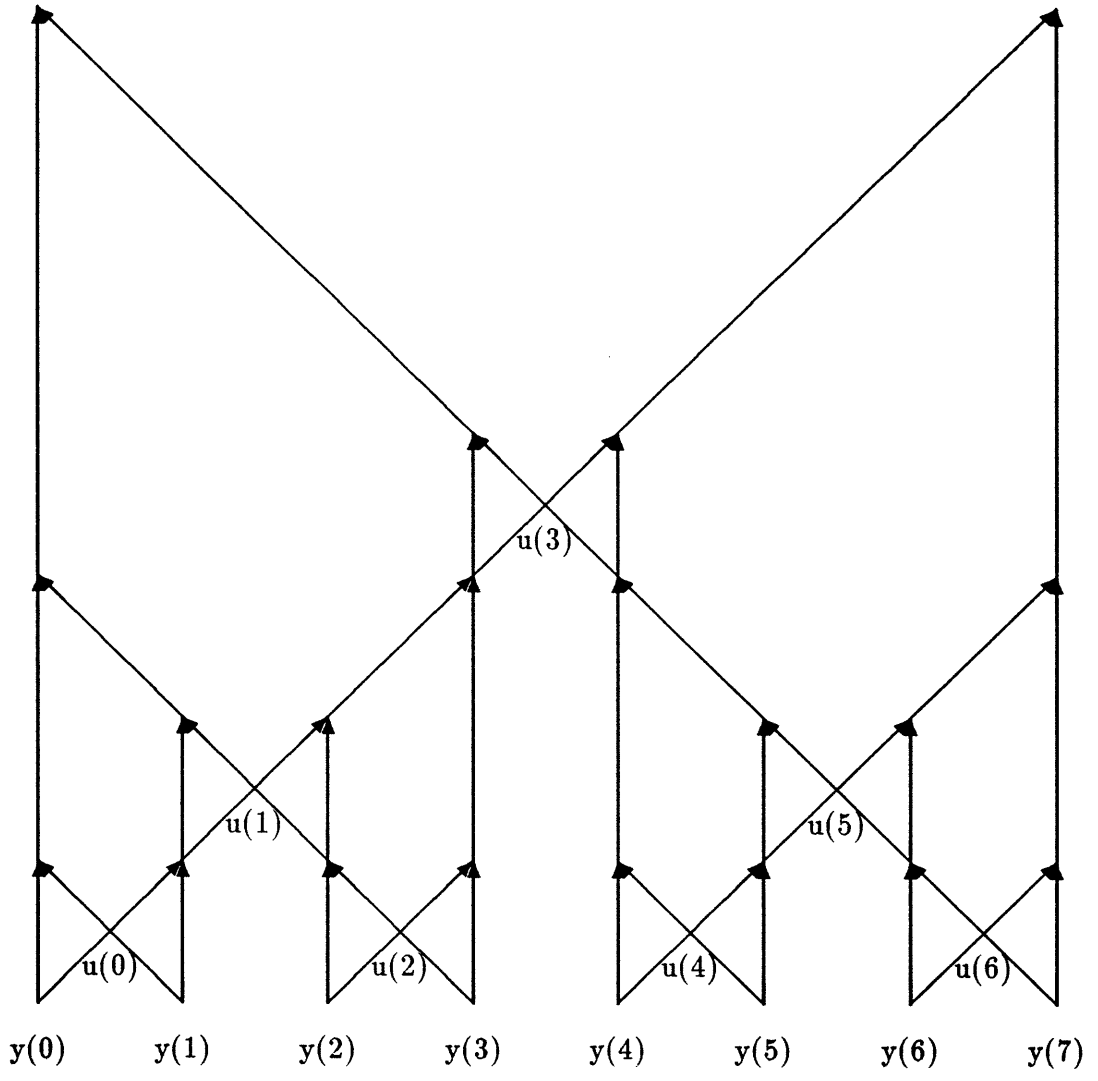


Figure 5-13: $T = 7$

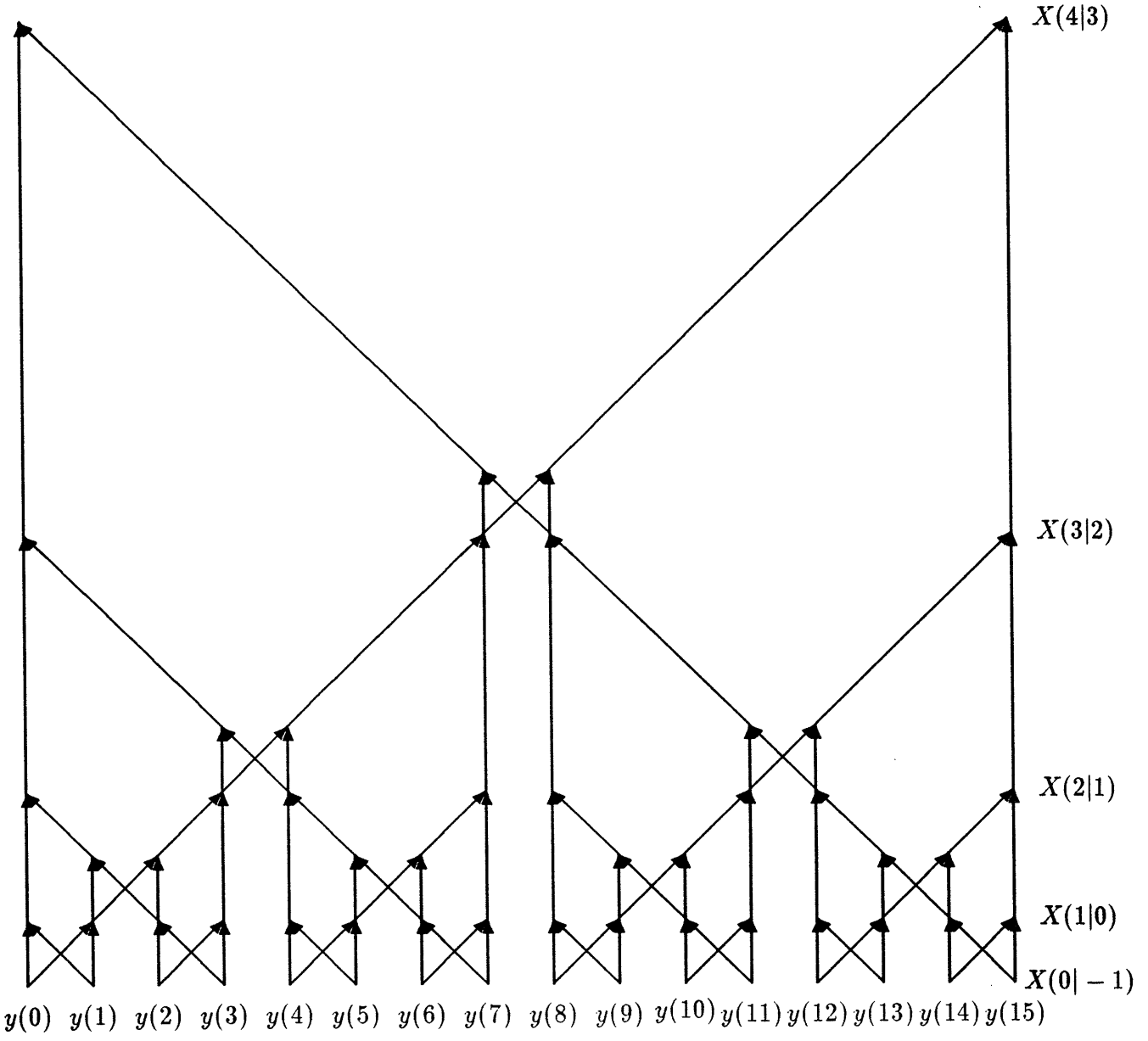


Figure 5-14: Parallel algorithm for 16 data points.

where the notation $[s,t)$ indicate that the estimates are based on all of the dynamic constraints linking $x(s)$ to $x(t)$ and all of the observations $y(s)$ through $y(t-1)$. The error covariance is given by

$$\text{Cov}\begin{bmatrix} \tilde{x}[0|[0,1)] \\ \tilde{x}[1|[0,1)] \end{bmatrix} = \begin{bmatrix} \Sigma_{00|[0,1)} & \Sigma_{01|[0,1)} \\ \Sigma_{10|[0,1)} & \Sigma_{11|[0,1)} \end{bmatrix} \tag{5.146}$$

While these computations are being performed, Processor #1 performs an analogous computation for $x(1)$ and $x(2)$. Processor #1 takes the measurements given by

$$\begin{bmatrix} y(1) \\ 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ \Gamma_a(1) & -I \end{bmatrix} \begin{bmatrix} x(1) \\ x(2) \end{bmatrix} + \begin{bmatrix} v(2) \\ u(2) \end{bmatrix} \tag{5.147}$$

to compute the estimate

$$\begin{bmatrix} \hat{x}[1|[1,2)] \\ \hat{x}[2|[1,2)] \end{bmatrix} = \begin{bmatrix} x(1) \\ x(2) \end{bmatrix} + \begin{bmatrix} \tilde{x}[1|[1,2)] \\ \tilde{x}[2|[1,2)] \end{bmatrix} \tag{5.148}$$

where the error covariance is given by

$$\text{Cov}\begin{bmatrix} \tilde{x}[1|[1,2)] \\ \tilde{x}[2|[1,2)] \end{bmatrix} = \begin{bmatrix} \Sigma_{11|[1,2)} & \Sigma_{12|[1,2)} \\ \Sigma_{21|[1,2)} & \Sigma_{22|[1,2)} \end{bmatrix} \tag{5.149}$$

Step 2:

Here, Step 2 of the parallel algorithm has three parts which we will now outline.

Part 1

The first part of Step 2 consists of providing the information that neighboring boundaries intersect. Specifically, the estimates in each region represent independent measurement of $x(1)$. With the combined information of the intersection of the two boundaries, a locally smoothed estimate of the boundary which encloses the two regions is obtained. This boundary is equal to $[x^T(0),\ x^T(2)]$. The measurements

needed to produce the boundary estimate for $[x^T(0),\ x^T(2)]$ are given by

$$\begin{bmatrix} \hat{x}[0|[0,1)] \\ \hat{x}[1|[0,1)] \\ \hat{x}[1|[1,2)] \\ \hat{x}[2|[1,2)] \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x(2) \end{bmatrix} + \begin{bmatrix} \tilde{x}[0|[0,1)] \\ \tilde{x}[1|[0,1)] \\ \tilde{x}[2|[1,2)] \end{bmatrix} \tag{5.150}$$

resulting in the estimate

$$\begin{bmatrix} \hat{x}[0|[0,2)] \\ \hat{x}[1|[0,2)] \\ \hat{x}[2|[0,2)] \end{bmatrix} = \begin{bmatrix} x(0) \\ x(1) \\ x(2) \end{bmatrix} + \begin{bmatrix} \tilde{x}[0|[0,2)] \\ \tilde{x}[1|[0,2)] \\ \tilde{x}[2|[0,2)] \end{bmatrix} \tag{5.151}$$

The error covariance of this estimate is given by

$$\mathrm{Cov} \begin{bmatrix} \tilde{x}[0|[0,2)] \\ \tilde{x}[1|[0,2)] \\ \tilde{x}[2|[0,2)] \end{bmatrix} = \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{01|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{10|[0,2)} & \Sigma_{11|[0,2)} & \Sigma_{12|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{21|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix} \tag{5.152}$$

This estimation step can be interpreted as combining two independent estimates of $x(1)$, and using the resulting information contained in the estimates of $x(1)$ to update the boundary enclosing the two regions on which the two measurements of $x(1)$ were based. We call this an estimation module because it, as is the corresponding computation in Example 5.1, a basic building block for larger and larger estimation algorithms.

Part 2

The boundary condition is then combined with the boundary estimate to produce the smoothed estimates of the boundary. The measurements required to produce this

smoothed estimate are given by

$$\begin{bmatrix} \hat{x}[0|[0,2)] \\ \hat{x}[2|[0,2)] \\ 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \\ -\Gamma_e(0) & \Gamma_a(2) \end{bmatrix} \begin{bmatrix} x(0) \\ x(2) \end{bmatrix} + \begin{bmatrix} \tilde{x}[0|[0,2)] \\ \tilde{x}[2|[0,2)] \\ u(2) \end{bmatrix} \tag{5.153}$$

The smoothed estimates which result from (5.153) are given by

$$\begin{bmatrix} \hat{x}_s(0) \\ \hat{x}_s(2) \end{bmatrix} = \begin{bmatrix} x(0) \\ x(2) \end{bmatrix} + \begin{bmatrix} \tilde{x}_s(0) \\ \tilde{x}_s(2) \end{bmatrix} \tag{5.154}$$

The smoothed error covariance for the boundary is given by

$$\text{Cov} \begin{bmatrix} \tilde{x}_s(0) \\ \tilde{x}_s(2) \end{bmatrix} = \begin{bmatrix} \Sigma_{00} & \Sigma_{02} \\ \Sigma_{02} & \Sigma_{22} \end{bmatrix} \tag{5.155}$$

Part 3

The final part in Step 2 is to propagate smoothed estimates to the boundaries of the individual subregions, that is, to propagate estimates from the state $[x^T(0), x^T(2)]$ to the states $[x^T(0), x^T(1)]$ and $[x^T(1), x^T(2)]$. The smoothed estimates of the boundaries of the subregions are provided by

$$\begin{aligned} \begin{bmatrix} \hat{x}_s(0) \\ \hat{x}_s(1) \end{bmatrix} &= \begin{bmatrix} \hat{x}[0|[0,2)] \\ \hat{x}[1|[0,2)] \end{bmatrix} + \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{10|[0,2)} & \Sigma_{12|[0,2)} \end{bmatrix} \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix}^{-1} \\ &\times \left(\begin{bmatrix} \hat{x}_s(0) \\ \hat{x}_s(2) \end{bmatrix} - \begin{bmatrix} \hat{x}[0|[0,2)] \\ \hat{x}[2|[0,2)] \end{bmatrix} \right) \end{aligned} \tag{5.156}$$

and

$$\begin{bmatrix} \hat{x}_s(1) \\ \hat{x}_s(2) \end{bmatrix} = \begin{bmatrix} \hat{x}[1|[0,2)] \\ \hat{x}[2|[0,2)] \end{bmatrix} + \begin{bmatrix} \Sigma_{10|[0,2)} & \Sigma_{12|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix} \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix}^{-1} \times \left(\begin{bmatrix} \hat{x}_s(0) \\ \hat{x}_s(2) \end{bmatrix} - \begin{bmatrix} \hat{x}[0|[0,2)] \\ \hat{x}[2|[0,2)] \end{bmatrix} \right) \tag{5.157}$$

The smoothed error covariances for the two subregion boundaries are given by

$$\begin{bmatrix} \Sigma_{00} & \Sigma_{01} \\ \Sigma_{10} & \Sigma_{11} \end{bmatrix} = \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{01|[0,2)} \\ \Sigma_{10|[0,2)} & \Sigma_{11|[0,2)} \end{bmatrix} - \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{10|[0,2)} & \Sigma_{12|[0,2)} \end{bmatrix} \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix}^{-1} \times \left(\begin{bmatrix} \Sigma_{00} & \Sigma_{02} \\ \Sigma_{20} & \Sigma_{22} \end{bmatrix} - \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix} \right) \times \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix}^{-1} \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{01|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{21|[0,2)} \end{bmatrix} \tag{5.158}$$

and

$$\begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} = \begin{bmatrix} \Sigma_{11|[0,2)} & \Sigma_{12|[0,2)} \\ \Sigma_{21|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix} - \begin{bmatrix} \Sigma_{10|[0,2)} & \Sigma_{12|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix} \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix}^{-1} \times \left(\begin{bmatrix} \Sigma_{00} & \Sigma_{02} \\ \Sigma_{20} & \Sigma_{22} \end{bmatrix} - \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix} \right) \times \begin{bmatrix} \Sigma_{00|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{20|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix}^{-1} \begin{bmatrix} \Sigma_{01|[0,2)} & \Sigma_{02|[0,2)} \\ \Sigma_{21|[0,2)} & \Sigma_{22|[0,2)} \end{bmatrix} \tag{5.159}$$

Step 3

Here as in Example 5.1, the local subregions and their boundaries are the same. There are no interior points to update. □

Returning to the issue of smoothing the TPBVDS in equation (5.1), we now continue with our general algorithm for smoothing one dimensional systems. We also return to the notation used to describe estimates and their data dependencies

established in Section ??4.3. The data is partitioned along the time axis into $L = 2^M$ subintervals and one processor is assigned to each subinterval. Since Examples 1 and 2 show the structure of the algorithm when all inverses exist we will now carry out the computations when the state is not estimable.

Step 1

Local filtering is performed in each of the subintervals starting from the center outward toward the boundary to produce local estimates of the boundary of each subinterval. This preprocessing step is therefore identical to the preprocessing step of the algorithm in Section ??5.5. The result of the computation in this step is given by

$$\begin{bmatrix} \hat{x}[\kappa|\kappa, k+\delta] \\ \hat{x}[k+\delta|\kappa, k+\delta] \end{bmatrix} = \begin{bmatrix} x(\kappa) \\ x(k+\delta) \end{bmatrix} + \begin{bmatrix} \tilde{x}[\kappa|\kappa, k+\delta] \\ \tilde{x}[k+\delta|\kappa, k+\delta] \end{bmatrix} \tag{5.160}$$

The error covariance of this estimate is given by

$$\text{Cov} \begin{bmatrix} \tilde{x}[\kappa|\kappa, k+\delta] \\ \tilde{x}[k+\delta|\kappa, k+\delta] \end{bmatrix} = \begin{bmatrix} \Sigma_{\kappa,\kappa|\kappa,k+\delta} & \Sigma_{\kappa,k+\delta|\kappa,k+\delta} \\ \Sigma_{k+\delta,\kappa|\kappa,k+\delta} & \Sigma_{k+\delta,k+\delta|\kappa,k+\delta} \end{bmatrix} \tag{5.161}$$

where the process at the boundary of the subinterval is given by $[x^T(\kappa), x^T(k+\delta)]^T$. The neighboring boundary we will denote by $[x^T(k), x^T(s)]^T$, and the binary variable δ determines whether or not the neighboring boundaries intersect. If $\delta = 1$ the boundaries do not intersect. If $\delta = 0$ the boundaries do intersect.

Step 2

At this point the remainder of the algorithm represents a hierarchical assembling of Step 2 in Example 5.1 or 5.2. The states to be combined for the measurement update step are shown in Figure 5-15. The measurement update step is given by considering

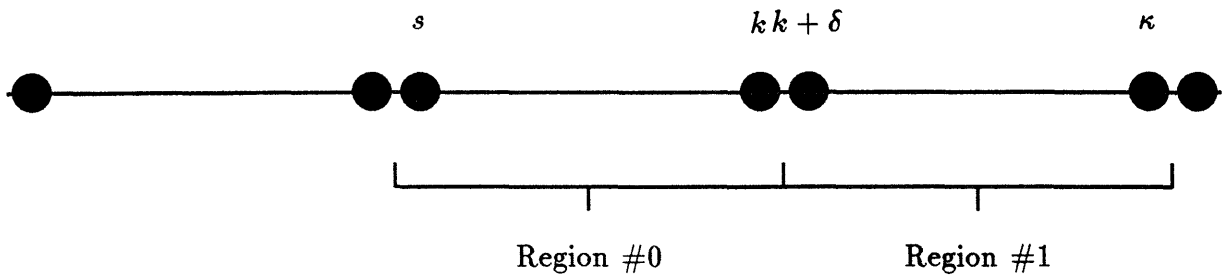


Figure 5-15: The dots represent the boundary of neighboring regions which will be merged together in the measurement update step. Region #0 and Region #1 will be merged together in into a region bounded by the points s and κ.

the following measurement of neighboring boundaries

$$\begin{bmatrix} \hat{x}[\kappa|k+\delta,\kappa] \\ \hat{x}[k+\delta|k+\delta,\kappa] \\ 0 \\ \hat{x}[k|s,k] \\ \hat{x}[s|s,k] \end{bmatrix} = \begin{bmatrix} P_{\kappa,\kappa|\kappa,k+\delta} & P_{\kappa,k+\delta|\kappa,k+\delta} & 0 & 0 \\ P_{k+\delta,\kappa|\kappa,k+\delta} & P_{k+\delta,k+\delta|\kappa,k+\delta} & 0 & 0 \\ 0 & \{E_{k+\delta}\}^{\delta} & -\{A_k\}^{\delta} & 0 \\ 0 & 0 & P_{k,k|k,s} & P_{k,s|k,s} \\ 0 & 0 & P_{s,k|k,s} & P_{s,s|k,s} \end{bmatrix}^{\#} \times \begin{bmatrix} x(\kappa) \\ x(k+\delta) \\ x(k) \\ x(s) \end{bmatrix} + \begin{bmatrix} \tilde{x}[\kappa|k+\delta,\kappa] \\ \tilde{x}[k+\delta|k+\delta,\kappa] \\ \delta B_k u(k) \\ \tilde{x}[k|s,k] \\ \tilde{x}[s|s,k] \end{bmatrix} \tag{5.162}$$

The matrix $P_{i,j|k,l}$ represents a submatrix of the projection matrix which is associated with the estimate of $[x^T(k), x^T(l)]$. The result of the estimation problem is the following estimate

$$\begin{bmatrix} \hat{x}[\kappa|s,\kappa] \\ \hat{x}[k+\delta|s,\kappa] \\ \hat{x}[k|s,\kappa] \\ \hat{x}[s|s,\kappa] \end{bmatrix} = P_{x(\kappa),x(k+\delta),x(k),x(k)|s,\kappa} \begin{bmatrix} x(\kappa) \\ x(k+\delta) \\ x(k) \\ x(s) \end{bmatrix} + \begin{bmatrix} \tilde{x}[\kappa|s,\kappa] \\ \tilde{x}[k+\delta|s,\kappa] \\ \tilde{x}[k|s,\kappa] \\ \tilde{x}[s|s,\kappa] \end{bmatrix} \tag{5.163}$$

where $P_{x(\kappa),x(k+\delta),x(k),x(s)|s,\kappa}$ is the associated projection matrix for the estimate. The

error covariance of the estimate is given by

$$\mathrm{Cov}\begin{bmatrix} \tilde{x}[\kappa|s,\kappa] \\ \tilde{x}[k+\delta|s,\kappa] \\ \tilde{x}[k|s,\kappa] \\ \tilde{x}[s|s,\kappa] \end{bmatrix} = \Sigma_{x(\kappa),x(k+\delta),x(k),x(s)|s,\kappa} \tag{5.164}$$

To continue with further steps of the interprocessor communication step the boundary which encloses the two regions is kept. The system is therefore sampled. Since projection matrices are involved, the proper projection matrix for the sampling must be computed. Let $P_{x(\kappa),x(s)|s,\kappa}$ denote the desired projection matrix which satisfies

$$\begin{aligned} \overline{P}_{x(\kappa),x(s)|s,\kappa} = & \left\{ \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \overline{P}_{x(\kappa),x(k+\delta),x(k),x(k)|s,\kappa} \begin{bmatrix} I & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & I \end{bmatrix} \right\}^{\#} \\ & \times \left\{ \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \overline{P}_{x(\kappa),x(k+\delta),x(k),x(k)|s,\kappa} \begin{bmatrix} I & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & I \end{bmatrix} \right\} \end{aligned} \tag{5.165}$$

then the joint estimate for $x(\kappa)$ and $x(k)$ is given by

$$\begin{bmatrix} \hat{x}[\kappa|s,\kappa] \\ \hat{x}[k|s,\kappa] \end{bmatrix} = P_{x(\kappa),x(k)|s,\kappa} \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \begin{bmatrix} \hat{x}[\kappa|s,\kappa] \\ \hat{x}[k+\delta|s,\kappa] \\ \hat{x}[k|s,\kappa] \\ \hat{x}[k|s,\kappa] \end{bmatrix} \tag{5.166}$$

Since sampling does not introduce new noise to the process, the covariance is

obtained from

$$\Sigma_{x(\kappa),x(k)|s,\kappa} = P_{x(\kappa),x(k)|s,\kappa} \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \Sigma_{x(\kappa),x(k+\delta),x(k),x(s)|s,\kappa} \begin{bmatrix} I & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & I \end{bmatrix} P_{x(\kappa),x(k)|s,\kappa} \tag{5.167}$$

Once smoothed estimates have been obtained at the boundary of the entire process, smoothed estimates may be propagated from a boundary of a subregion to the boundaries of the two next smaller subregions. The following measurements are combined: the smoothed estimates at $\{\kappa, s\}$ and the 'filtered' estimates to $\{\kappa, k+\delta\}$ and $\{k, s\}$

$$\begin{bmatrix} \hat{x}[\kappa|s,\kappa] \\ \hat{x}[k+\delta|s,\kappa] \\ \hat{x}[k|s,\kappa] \\ \hat{x}[s|s,\kappa] \\ \cdots\cdots\cdots \\ \hat{x}[k|0,K] \\ \hat{x}[s|0,K] \end{bmatrix} = \begin{bmatrix} P_{x(\kappa),x(k+\delta),x(k),x(s)|s,\kappa} \\ \cdots\cdots\cdots\cdots \\ 0 \;\vdots\; P_{x(\kappa),x(s)|0,K} \end{bmatrix} \begin{bmatrix} x(\kappa) \\ x(k) \\ \cdots\cdots\cdots \\ x(k) \\ x(s) \end{bmatrix} + \begin{bmatrix} \tilde{x}[\kappa|s,\kappa] \\ \tilde{x}[k+\delta|s,\kappa] \\ \tilde{x}[k|s,\kappa] \\ \tilde{x}[s|s,\kappa] \\ \cdots\cdots\cdots \\ \tilde{x}[k|0,K] \\ \tilde{x}[s|0,K] \end{bmatrix} \tag{5.168}$$

The resulting estimate is the jointly smoothed estimate of the two neighboring boundaries.

$$\begin{bmatrix} \hat{x}[\kappa|0,K] \\ \hat{x}[k+\delta|0,K] \\ \hat{x}[k|0,K] \\ \hat{x}[s|0,K] \end{bmatrix} = P^{s}_{x(\kappa),x(k+\delta),x(k),x(s)|s,\kappa} \begin{bmatrix} x(\kappa) \\ x(k) \\ \cdots\cdots\cdots \\ x(k) \\ x(s) \end{bmatrix} + \begin{bmatrix} \tilde{x}[\kappa|0,K] \\ \tilde{x}[k+\delta|0,K] \\ \tilde{x}[k|0,K] \\ \tilde{x}[s|0,K] \end{bmatrix} \tag{5.169}$$

The estimates of the two boundaries can be decoupled by premultiplication by the

appropriate projection matrices. If we let

$$\overline{P}^{s}_{x(\kappa),x(k+\delta)|0,K} = \left\{ \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \end{bmatrix} \overline{P}^{s}_{x(\kappa),x(k+\delta),x(k),x(k)|0,K} \begin{bmatrix} I & 0 \\ 0 & I \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \right\}^{\#} \times \left\{ \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \end{bmatrix} \overline{P}^{s}_{x(\kappa),x(k+\delta),x(k),x(k)|0,K} \begin{bmatrix} I & 0 \\ 0 & I \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \right\} \tag{5.170}$$

and

$$\overline{P}^{s}_{x(k),x(s)|0,K} = \left\{ \begin{bmatrix} 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \overline{P}^{s}_{x(\kappa),x(k+\delta),x(k),x(k)|0,K} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ I & 0 \\ 0 & I \end{bmatrix} \right\}^{\#} \times \left\{ \begin{bmatrix} 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \overline{P}^{s}_{x(\kappa),x(k+\delta),x(k),x(k)|0,K} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ I & 0 \\ 0 & I \end{bmatrix} \right\} \tag{5.171}$$

then premultiplying (5.169) by

$$\begin{bmatrix} P^{s}_{x(k),x(s)|0,K} & 0 \\ 0 & P^{s}_{x(k),x(s)|0,K} \end{bmatrix}$$

Thus to compute the smoothed estimate of the process at the boundary surrounding two neighboring subregions, we require measurements of the 'filtered' estimates of the process at the boundary of each neighboring subregion, and the smoothed estimate of the boundary which immediately encloses the two subregions.

This process can be continued recursively until a smoothed estimate is obtained

at the boundary of each subinterval marking the end of the interprocessor exchange step.

Step 3

The backward sweep step of the Rauch-Tung-Striebel is used to propagate smoothed estimates from the boundaries inward to the center of each subregion.

The sampling of the system is a key feature of this algorithm. Here we will show that the interprocessor communication can be formulated precisely as the Rauch-Tung-Striebel algorithm operating on an equivalent 'causal' system. This establishes the interprocessor communication step as the simple application of known algorithms on sampled processes. It also lends a different perspective to the notion of 'filtered' and 'predicted' estimates in a parallel processing context. To accomplish this it helps to reinterpret the role that different noisy constraints play. In the following example, the observations $y(k)$ are interpreted as 'a priori' information, the dynamics plays the role of of 'observations', and the sampling of the state plays the role of the 'dynamics' of the system.

Consider again, the model used in Example 5.1. Here we will stack all of the elements of the vector $x(k)$ into one large vector $X(0)$ in order to describe all of the operations which can be carried out in parallel. We will define the 'model' describing the system as simply sampling the process, by dropping the pairs of points required to define the boundaries of larger and larger subregions as indicated in Example 5.1. The model can be written as

$$X(m+1) = A(m)X(m) \tag{5.172}$$

Thus the sampling operation $A(m)$ imposes a recursive definition of the remaining states $X(m)$, given $X(0)$. Typically we would expect a 'causal' model to have a priori statistics defined for $X(0)$. Let the observations $y(k) = x(k)+v(k)$ define the 'a priori' statistics for $X(0)$. We will therefore initialize the state $X(0)$ with the observations $y(s)$ appropriately stacked into one large vector $\hat{X}(0|-1)$. The covariance of this initial estimate is formed from the covariances associated with the individual error

covariances for the observations $y(s)$.

$$\begin{aligned} \hat{X}(0|-1) &= X(0) + \tilde{X}(0|-1) \\ \tilde{X}(0|-1) &\sim N(0; \Sigma(0|-1)) \\ P(0|-1) &= \text{diag}[r(0), r(1), r(2), \cdots, x(2^m)] \end{aligned} \tag{5.173}$$

Finally, the dynamics of the system of the form of $0 = -e(t+1)x(t+1)+a(t)x(t)+u(t)$ can be 'selectively' included in the observations $0 = C(m)X(m) + V(m)$ in order to insure that as many of the computations can be performed in parallel as possible. If $u(t)$ has nonzero mean or if the process has inputs, then we can write $Y(m) = C(m)X(m) + V(m)$ where $Y(m)$ would represent these nonzero means, and inputs to our system. Once the model is written as

$$\begin{aligned} X(m+1) &= A(m)X(m) \\ X(0) &\sim N(\hat{X}(0|-1); \Sigma(0|-1)) \\ 0 &= C(m)X(m) + V(m) \end{aligned} \tag{5.174}$$

then the smoothing algorithm follows directly. Specifically, the Rauch-Tung-Striebel algorithm is given by the filtering equations

$$\begin{aligned} \hat{X}(m|m) &= (I - \Sigma(m|m-1)C^T(m) \\ &\times (C(m)\Sigma(m|m-1)C^T(m) + R(m))^{-1}C(m))\hat{X}(m|m-1) \end{aligned} \tag{5.175}$$

$$\begin{aligned} \Sigma(m|m) &= \Sigma(m|m-1) - \Sigma(m|m-1)C^T(m) \\ &\times (C(m)\Sigma(m|m-1)C^T(m))^{-1}C(m)\Sigma(m|m-1) \end{aligned} \tag{5.176}$$

$$\hat{X}(m+1|m) = A(m)\hat{X}(m|m) \tag{5.177}$$

$$\Sigma(m+1|m) = A(m)\Sigma(m|m)A^T(m) \tag{5.178}$$

and the backward sweep given by

$$\hat{X}(m|M) = \hat{X}(m|m) - \Sigma(m|m)A^T(m)\Sigma^{-1}(m+1|m)(\hat{X}(m+1|M) - \hat{X}(m+1|m)) \tag{5.179}$$

$$\begin{aligned}\Sigma(m|M) &= \Sigma(m|m) - \Sigma(m|m)\mathcal{A}_m^T\Sigma^{-1}(m+1|m) \\ &\times \ (\Sigma(m+1|M) - \Sigma(m+1|m))\Sigma^{-1}(m+1|m)\mathcal{A}_m\Sigma(m|m) \end{aligned} \tag{5.180}$$

Because of the specific model we chose, all inverses exist. The computations which correspond to Step 1, which is the preprocessing step, occur in the filtering step for $m = 0$. $A(0) = I$ because the purpose of Step 1 is to compute measurements of local boundaries based on interior data. Sampling only occurs when boundaries which consist of two points are being combined to enclose larger and larger regions. Here, interior data is simply the dynamic constraints which link each element of the boundary. In this operation half of all of the dynamic constraints are used. We may then combine the dynamics $0 = x(s+1) - a(s)x(s) - b(s)u(s)$ for s even into one 'observation' and, under the assumption that the noise is zero mean, it can be written as

$$\begin{aligned} Y(0) = 0 &= C(0)X(0) + V(0) \\ V(0) &\sim N(0; R(0)) \end{aligned} \tag{5.181}$$

$$\begin{aligned} R(0) &= \text{diag}\{q(0), q(2), q(4), \cdots, q(2^m - 1)\} \\ C(0) &= \text{diag}\{[-a(0) \ \ I], [-a(2) \ \ I], [-a(4) \ \ I], \cdots, [-a(2^m - 2) \ \ I]\} \end{aligned} \tag{5.182}$$

The resulting filtered estimate after this measurement update is performed is given by

$$\hat{X}(0|0) = (I - \Sigma(0|-1)C^T(0)(C(0)\Sigma(0|-1)C^T(0) + R(0))^{-1}C(0))\hat{X}(0|-1) \tag{5.183}$$

This processing can be carried out in parallel, since $C(0)$, $\Sigma(0|-1)$, and $R(0)$ are all block diagonal. We are now prepared for further processing. Also since $A(0) = I$ the 'predicted' estimate and the 'filtered' estimate are the same. Furthermore the 'dynamics' are noiseless, resulting in the 'predicted' error covariance $\Sigma(1|0)$ being equal to the 'filtered' error covariance $\Sigma(0|0)$.

The remainder of the algorithm corresponds to Step 2 in Example 5.1. Half of the remaining dynamic constraints will be included at each step, and the state will be sampled reducing the dimension of the state by 2 when m increases by 1.

With each sampling, the elements of $X(m)$ are given by $x([j-1]2^m)$, and $x(j2^m-1)$ for $1 \leq j \leq 2^{M-m}$. The matrix $A(m)$ is defined to be the matrix which will perform this sampling. $A(m)$ is given by

$$\begin{aligned} A(m) &= \text{diag}\{Z, Z, \cdots, Z\} \\ Z &= \begin{bmatrix} I & 0 & 0 & 0 \\ 0 & 0 & 0 & I \end{bmatrix} \\ A(m) &\in R^{2^{M-m} \times 2^{M-m+1}} \end{aligned} \tag{5.184}$$

The 'dynamics' for the system given by (5.172) are not driven by noise. The 'prediction' update step of the filter is given by

$$\begin{aligned} \hat{X}(m+1|m) &= A(m)\hat{X}(m|m) \\ P(m+1|m) &= A(m)P(m|m)A^T(m) \end{aligned} \tag{5.185}$$

Thus a 'predicted' estimate is formed from sampling the 'filtered' estimates. This amounts to premultiplying $X(m)$ by $A(m)$ which removes two neighboring elements in the center of a larger region while keeping the elements which bound this larger region.

The measurement update step is accomplished by incorporating dynamic constraints which link pairs of 'states' together. The constraints of the form $0 = x(s+1) - a(s)x(s) - b(s)u(s)$ which are used in the measurement update of the state $X(m)$ are those which correspond to $s = j2^m - 1$ for j odd. These constraints can be incorporated into a single measurement of the form $0 = C(m)X(m) + V(m)$.

$$\begin{aligned} 0 &= C(m)X(m) + V(m) \\ V(m) &\sim N(0; R(m)) \end{aligned} \tag{5.186}$$

$$\begin{aligned} R(m) &= \text{diag}\{q(2^m - 1), q(3 \times 2^m - 1), q(5 \times 2^m - 1), \cdots, q(2^M - 2^m - 1) \\ C(m) &= \text{diag}\{[-a(2^m - 1) \;\; I], [-a(3 \times 2^m - 1) \;\; I], [-a(5 \times 2^m - 1) \;\; I], \cdots, [-a(2^M - 2^m - 1) \;\; I]\} \end{aligned} \tag{5.187}$$

The system has been represented by a causal system as in (5.174), and as a result

all of the matrices in the Rauch-Tung-Striebel algorithm have been defined. Since $C(m)$, $A(m)$, $R(m)$, and $\Sigma(0|-1)$ are all block diagonal matrices, all computations including the matrix inverses in (5.175), and (5.176) can be carried out in parallel in the filtering step. At each level all computations are self similar, though fewer in number. On the other hand, the smoothed estimates of the state $X(m)$ are not block diagonal and we would expect that the backward sweep computations could not be carried out in parallel. However in the backward sweep equations of the Rauch-Tung-Striebel algorithm (5.179) and (5.180), $\Sigma(m|m)$, A_m^T, $\Sigma^{-1}(m+1|m)$, are all diagonal. As a result only the diagonal elements of $\Sigma(m+1|M)$ are used in the computation of the diagonal elements of $\Sigma(m|M)$. Therefore the computation of the diagonal blocks of the smoothed covariance can be carried out in parallel.

Using the ML techniques we know that this algorithm can be carried out without 'a priori' information about $X(0)$. In other words this algorithm can be carried out without the data $y(t)$. The resulting smoothed system is nothing other than the process, less observations. With the smoothed estimate of $X(0)$ (in the absence of observations $y(t)$), the process at all other samplings are well defined because $X(m+1) = A(m)X(m)$ is a noiseless process. The process is clearly Markov, and as a result a backward Markov process can be defined. Equivalently, given a 'state' of a reciprocal system to be the process defined at the boundary $\{0, 2^M - 1\}$ then any state on a boundary which is interior to $\{0, 2^M - 1\}$, as defined by the order relation $\preceq$ can be written as a linear function of the state at $\{0, 2^M - 1\}$ plus independent white noise. Specifically we consider the two 'states' at $\{0, 2^{M-1} - 1\}$ and $\{2^{M-1}, 2^M - 1\}$. Consequently if we consider the 'state' at the top of a binary tree to be the process at the boundary $\{0, 2^M - 1\}$ then the 'state' at the descendant nodes are the process at $\{0, 2^{M-1} - 1\}$, and $\{2^{M-1}, 2^M - 1\}$ which are the boundaries of the next smaller subregions. What results is a model not unlike that used by Chou in [26]. As a result the basic principles of smoothing algorithms used in [26] can be applied here.

Consider the processors operating on the regions indicated in Figure 5-16. Each Processor has a binary label associated with it. (Note that the digits in the binary labels when taken backwards count from 0 to 15.) If each processor communicates

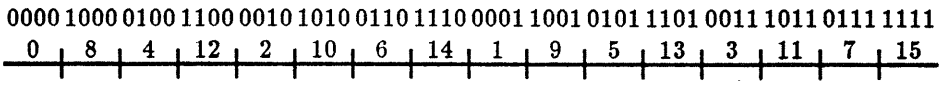


Figure 5-16: Regions are assigned to processors which are arranged with Hypercube interconnections where processors whose label differ by one bit communicate directly with each other. With the above labelling scheme the result of computations which involve a pair of processors is stored with the processor whose label is obtained by removing the most significant bit. For example, Processor 4 and Processor 12 after removing the most significant bit from their labels can determine that the result of computation based on data available to both processors will be left with Processor 4.

with other processors whose binary label differ by one bit, then the interconnection is that of a hypercube. During the interprocessor communication step, processors first communicate in pairs. Processor 0 communicates with Processor 8, Processor 4 communicates with Processor 12, etc.. The location of the new boundary estimates for the region which the processor operated is obtained by removing the highest order bit. Therefore Processor 0, and Processor 8 will place the results of the boundary computations in Processor 0, Processor 5, and Processor 13 will place the results of their computations in Processor 5, etc.. Next the processors which communicate next are Processors 0, 4, 2, 6, 1, 5, 3, and 7. These processors will then communicate in pairs. Processor 0 communicates with Processor 4, Processor 2 communicates with Processor 6, etc.. Again the location of the new boundary estimates for the region which the processor operated is obtained by removing the highest order bit. Therefore Processor 0, and Processor 4 will place the results of the boundary computations in Processor 0, Processor 1, and Processor 5 will place the results of their computations in Processor 1, etc.. Eventually Processor 0 and Processor 1 will remain. The results will be placed in Processor 0, where a priori information will be included. The smoothed estimates will be distributed amongst the processors in the reverse pattern while computing the necessary smoothed estimates of the boundaries of smaller and smaller subregions until each processor has smoothed estimates of the boundary of the region for which it was originally assigned.

5.6.1 Complexity

The computation involved in this algorithm, as does that for the other new algorithms in this chapter, varies with whether or not the state is estimable. Here, we will provide three figures. The first will assume that the states at each step are not estimable. The second will assume that the states are estimable but pseudo-inverses may be needed during the computations. Finally we will consider the case where all inverses exist. For all of these cases, we will assume that the final smoothed estimates are estimable given all of the data.

Non-estimable states

Step 1

Off-line

In Step 1 radial filtering is performed. The amount of computation is given by

$$\frac{K}{2L}\mathcal{K}(2n, 2p, 2m; \text{off} - \text{line}, \text{non} - \text{estimable}) \tag{5.188}$$

$$= \frac{K}{2L}[509.33n^3 + 448n^2p + 32n^2m + 168np^2 + 21.33p^3] \tag{5.189}$$

On-line

The on-line computation in Step 1 is given by

$$\frac{K}{2L}\mathcal{K}(2n, 2p, 2m; \text{on} - \text{line}) \tag{5.190}$$

$$= \frac{K}{2L}[8n^2 + 8np] \tag{5.191}$$

Step 2

'up' Off-line

These computation counts are for one step up the tree.

$$(\mathcal{M}(9n, 4n, 9n) + 160n^3 + \mathcal{E}(2n, 2n, 2n) + 32n^3)\log L \tag{5.192}$$

$$= 4090.67n^3 \log L \tag{5.193}$$

On-line

The on-line computation in Step 1 while merging regions together is given by

$$(32n^2)\log L \tag{5.194}$$

'down' Off-line

These computation counts are for one step 'down' the tree.

$$(\mathcal{M}(10n, 4n, 10n) + 192n^3 + \mathcal{E}(2n, 2n, 2n) + 32n^3)\log L \tag{5.195}$$

$$= (4925.33n^3)\log L \tag{5.196}$$

On-line

The remaining on-line computation in Step 1 is given by

$$(48n^2)\log L \tag{5.197}$$

Step 3

Off-line

In Step 3 then backward sweep of the Rauch-Tung Striebel algorithm is performed. The amount of computation is given by

$$\frac{K}{2L}\mathcal{T}(2n, 2m; \text{off} - \text{line}, \text{non} - \text{causally} - \text{estimable}) \tag{5.198}$$

$$= \frac{K}{2L}[941.33n^3 + 32n^2m] \tag{5.199}$$

On-line

The on-line computation in Step 3 is given by

$$\frac{K}{2L}\mathcal{T}(2n; \text{on} - \text{line}) = \frac{K}{2L}16n^2 \tag{5.200}$$

Estimable states

If the interprocessor communication step deals with well-defined covariances the

amount of computations are given by

Step 1

Off-line

In Step 1 radial filtering is performed. The amount of computation is given by

$$\frac{K}{2L}\mathcal{K}(2n, 2p, 2m; \text{off} - \text{line}, \text{non} - \text{estimable}) \tag{5.201}$$

$$= \frac{K}{2L}[509.33n^3 + 448n^2p + 32n^2m + 168np^2 + 21.33p^3] \tag{5.202}$$

On-line

The on-line computation in Step 1 is given by

$$\frac{K}{2L}\mathcal{K}(2n, 2p, 2m; \text{on} - \text{line}) \tag{5.203}$$

$$= \frac{K}{2L}[8n^2 + 8np] \tag{5.204}$$

Step 2

'up' Off-line

These computation counts are for one step up the tree.

$$\mathcal{E}(9n, 4n, 9n) \log L = 1944n^3 \log L \tag{5.205}$$

On-line

The on-line computation in Step 1 while merging regions together is given by

$$32n^2 \log L \tag{5.206}$$

'down' Off-line

These computation counts are for one step 'down' the tree.

$$\mathcal{E}(10n, 4n, 10n) \log L = 2533.33n^3 \log L \tag{5.207}$$

On-line

The remaining on-line computation in Step 1 is given by

$$48n^2 \log L \tag{5.208}$$

Step 3

Off-line

The amount of computation is given by

$$\frac{K}{2L}\mathcal{T}(2n, 2m; \text{off} - \text{line}, \text{non} - \text{causally} - \text{estimable}) \tag{5.209}$$

$$= \frac{K}{2L}[941.33n^3 + 32n^2m] \tag{5.210}$$

On-line

The on-line computation in Step 3 is given by

$$\frac{K}{2L}\mathcal{T}(2n, 2m; \text{on} - \text{line}) = \frac{K}{2L}16n^2 \tag{5.211}$$

Invertible Covariances

Finally if full rank covariances are available for the interprocessor exchange step, the pseudoinverses are not necessary. The operation count is then given by

Step 1

Off-line

In Step 1 radial filtering is performed. The amount of computation is given by

$$\frac{K}{2L}\mathcal{K}(2n, 2p, 2m; \text{off} - \text{line}, \text{non} - \text{estimable}) \tag{5.212}$$

$$= \frac{K}{2L}[509.33n^3 + 448n^2p + 32n^2m + 168np^2 + 21.33p^3] \tag{5.213}$$

On-line

The on-line computation in Step 1 is given by

$$\frac{K}{2L}\mathcal{K}(2n, 2p, 2m; \text{on} - \text{line}) \tag{5.214}$$

$$= \frac{K}{2L}[8n^2 + 8np] \tag{5.215}$$

Step 2

'up' Off-line

These computation counts are for one step up the tree.

$$\mathcal{I}(9n, 4n) \log L = 1134n^3 \log L \tag{5.216}$$

On-line

The on-line computation in Step 1 while merging regions together is given by

$$32n^2 \log L \tag{5.217}$$

'down' Off-line

These computation counts are for one step 'down' the tree.

$$(\mathcal{I}(10n, 4n)) \log L = 1466.67n^3 \log L \tag{5.218}$$

On-line

The remaining on-line computation in Step 1 is given by

$$48n^2 \log L \tag{5.219}$$

Step 3

Off-line

The amount of computation is given by

$$\frac{K}{2L}\mathcal{T}(2n, 2m; \text{off} - \text{line}, \text{non} - \text{causally} - \text{estimable}) \tag{5.220}$$

$$= \frac{K}{2L}[941.33n^3 + 32n^2m] \tag{5.221}$$

On-line

The on-line computation in Step 1 is given by

$$\frac{K}{2L}\mathcal{T}(2n, 2m; \text{on} - \text{line}) = \frac{K}{2L}16n^2 \tag{5.222}$$

In summary, the total off-line computation time for the case where we assume the state is not estimable locally, is given by

$$T = \frac{K}{2L}[1450.66n^3 + 448n^2p + 64n^2m + 168np^2 + 21.33p^3] + 9016n^3 \log L \tag{5.223}$$

The total on-line computation time for the case where we assume the state is not estimable locally, is given by

$$T = \frac{K}{2L}[24n^2 + 8np] + (96n^2)\log L \tag{5.224}$$

The total off-line computation time for the case where we assume the state is estimable given local data, but pseudo-inverses are needed in the computation, is given by

$$T = \frac{K}{2L}[1450.66n^3 + 448n^2p + 64n^2m + 168np^2 + 21.33p^3] + 4477.33n^3 \log L \tag{5.225}$$

The total on-line computation time for the case where we assume the state is estimable is given by

$$T = \frac{K}{2L}[24n^2 + 8np] + (80n^2)\log L \tag{5.226}$$

The total off-line computation time for the case where we assume that all covariances are invertible, is given by

$$T = \frac{K}{2L}[1450.66n^3 + 448n^2p + 64n^2m + 168np^2 + 21.33p^3] + 2600.67n^3 \log L \tag{5.227}$$

The total on-line computation time for the case where we assume the state is estimable

is given by

$$T = \frac{K}{2L}[24n^2 + 8np] + (80n^2)\log L \tag{5.228}$$

In each of the computation times given, the form follows that outlined in Equation 5.8. The functions f, and g are easily identifiable from these equations. The only difference between these times are in the off-line computations.

Chapter 6

Parallel ML Smoothing for Two-Dimensional Systems

6.1 Introduction

Those algorithms discussed in detail in Sections 5.2 through 5.5, which operate on a linear array of processors, have a structure which does not easily extend to the problem of smoothing in two dimensions. The algorithm in Section 5.6 however, which operates on processors with hypercube interconnections, does extend to algorithms which are applicable in two dimensions in a straightforward fashion. Local processing is performed in parallel in each of the local subregions to produce local estimates of the boundaries given local data. These estimates are obtained by filtering outward from the center of the subregion to the boundary of the subinterval. The interprocessor communication step involves combining the boundaries of two neighboring subregions to produce estimates of both boundaries based on all of the data and dynamics within each subregion and any additional dynamic constraints which may link the two regions, since these may not have been used in the local processing step. The estimate of the process at the boundary which encloses the two subregions, which is based on all enclosed data, is then combined with neighboring boundaries whose estimates have been constructed in the same manner. The process of merging boundaries is carried out recursively until the boundary of the entire process remains whose estimate is

based on all available data. As in Section 5.5, an algorithm analogous to the Rauch-Tung-Striebel is used to propagate smoothed estimates recursively to progressively smaller subregions, until smoothed boundary estimates are obtained for all of the local subregions. Finally smoothed estimates are propagated from the boundary toward the center of each of the local subregions. The computations for the smoothing problem map, as do the computations for the algorithm in Section 5.5, directly to the binary tree. The smoothing algorithm for two-dimensional processes will be analyzed by examining the parallel smoothing problem of a region divided into four subregions, in a manner similar to the two subregion analysis which takes place in Section 5.5. The nature of the smoothing algorithm suggests new approaches to modeling two-dimensional processes, not unrelated to the multi-resolution approaches discussed by Chou[26]. Furthermore the methodology in this section is completely generalizable to smoothing algorithms on higher dimensional processes.

6.2 Local Processing for a Two-Dimensional Region

The first and last steps of the ML parallel algorithms discussed in Section 5.4, Section 5.5, and in this chapter, are precisely the filtering and backward sweep steps, respectively, of the Rauch-Tung-Striebel algorithm. In Section 4.3, the Rauch-Tung-Striebel algorithm for STPBVDS's is presented. In this section we model two-dimensional systems with STPBVDS's. with this model, the Step 1 amounts to the FMLF algorithm outlined in Section 4.1, and Step 3 amounts to the backward sweep of the Rauch-Tung-Striebel algorithm as outlined in Section 4.3. In this section we take a more traditional model for two-dimensional systems and construct STPBVDS's by properly reordering the dynamic constraints.

In our view of processing multidimensional systems, we impose a radial time coordinate. We continue with our notion of processing outward from the center of the region towards the boundary and inward from the boundary toward the center of the region. In Chapter 3, we show that all TPBVDS's can be described as STPBVDS's

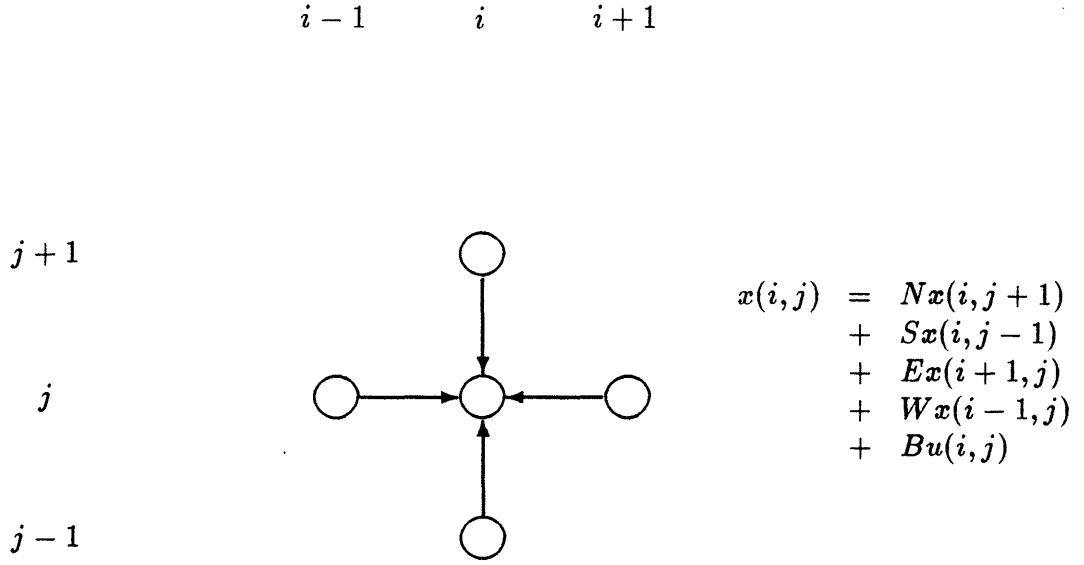


Figure 6-1: Nearest Neighbor Dependency

and as a result have a Markov description. Similarly in this section, we show how a two-dimensional system described by a nearest neighbor model[2](NNM), can be written as a STPBVDS. We could incorporate other two-dimensional models such as Roesser's model [35], Marchesini and Fornasini's model [8], and Jain's scalar NNM[10] into the STPBVDS model quite simply. Smoothing algorithms for NNM's are discussed in [1], and [2]. Since the actual model which we filter is a STPBVDS, our choice of the underlying model or original two-dimensional representation is made to be specific and not out of necessity.

Nearest neighbor models (NNM's) are given by the following

$$x(i,j) = Nx(i,j+1) + Sx(i,j-1) + Ex(i+1,j) + Wx(i-1,j) + Bu(i,j) \tag{6.1}$$

where the the matrices N, S, E, and W represent the dependencies on the neighbors immediately to the north, south, east, and west. Figure 6-1 shows the nearest neighbor dependency for this description of 2-D systems. Details as to the structure of the boundary condition for NNM's is discussed in [1]. We assume in this example that the noise sequence $u(i,j)$ is a white noise sequence. Let ρ_j be a radial coordinate

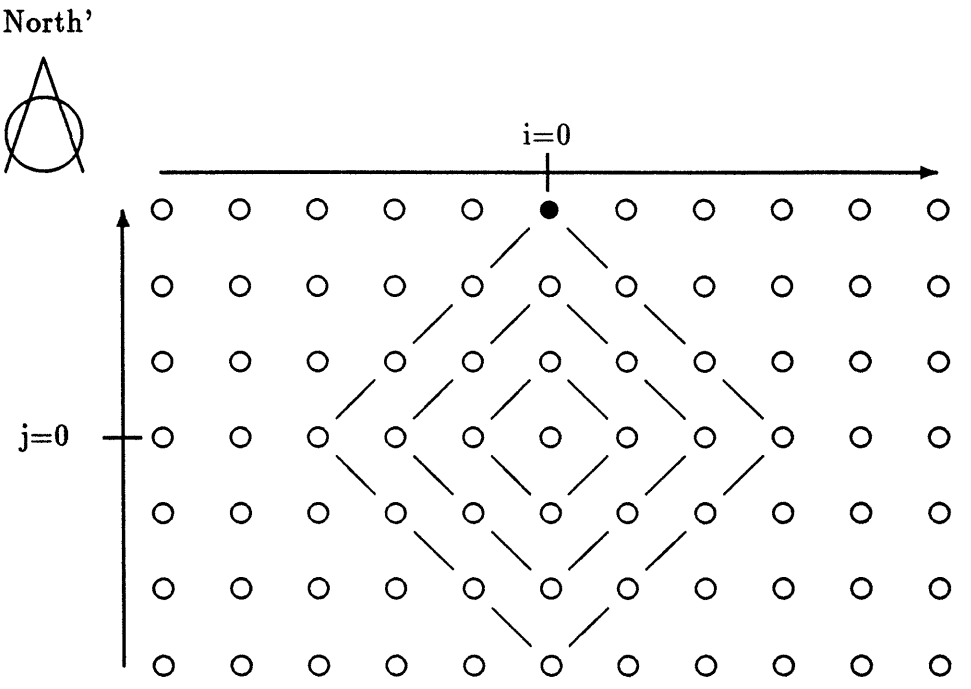


Figure 6-2: Diamond ordering of the elements in $\chi_\mu(\rho)$. $\rho \in \{0,1,2,3\}$, $\mu = 1$

with the following definition

$$\begin{aligned} \rho_\mu &= ||i,j||_\mu \\ \mu &\in \{1,\infty\} \end{aligned} \tag{6.2}$$

The l_1 norm of (i,j) is equal to $|i| + |j|$, and the l_∞ norm is given by $\max\{|i|,|j|\}$. Each of the $x(i,j)$ which lie along a contour of constant ρ_μ can be grouped into single vectors $\chi_\mu(\rho)$. The subscript μ will be dropped when the value of μ is either obvious or irrelevant to the discussion. The number of elements in each vector varies with ρ. The elements which form $\chi_\mu(\rho)$ are shown in Figure 6-2, and Figure 6-3 for the different values of μ. A dynamic model for $\chi_\mu(\rho)$ can be constructed which has the following structure.

$$\begin{aligned} D_{\rho,\mu}\chi_\mu(\rho) &= F_{\rho+1,\mu}\chi_\mu(\rho+1) + G_{\rho-1,\mu}\chi_\mu(\rho-1) + H_{\rho,\mu}w_\mu(\rho) \\ D_{0,\mu}\chi_\mu(0) &= F_{1,\mu}\chi_\mu(1) + H_{0,\mu}w_\mu(0) \\ D_{R,\mu}\chi_\mu(R) &= G_{R-1,\mu}\chi_\mu(R-1) + H_{R,\mu}w_\mu(R) \end{aligned} \tag{6.3}$$

This is a second order model and is not a STPBVDS. One is obtained by combining the vectors $\chi_\mu(\rho)$ in pairs, and writing a descriptor system which propagates the state

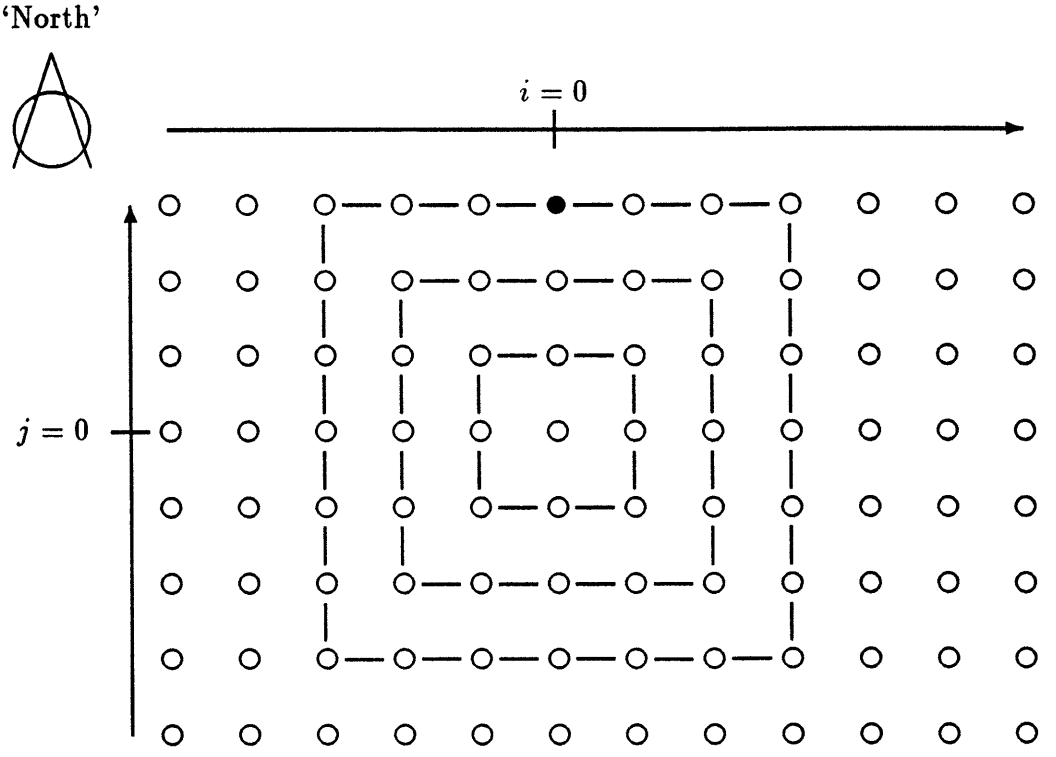


Figure 6-3: Square ordering of the elements of $\chi_\mu(\rho)$. $\rho \in \{0, 1, 2, 3\}$, $\mu = \infty$

$X_\mu(\rho)$ defined as

$$X_\mu(\rho) = \begin{bmatrix} \chi_\mu(\rho) \\ \chi_\mu(\rho - 1) \end{bmatrix} \tag{6.4}$$

We specifically call $X_\mu(\rho)$ a state, and not $x(i, j)$ nor $\chi_\mu(\rho)$ because $X_\mu(\rho)$ is sufficient to give the system a Markov representation. The following STPBVDS's describe the two-dimensional systems.

$$\begin{aligned} & \begin{bmatrix} -F_{\rho+2,\mu} & D_{\rho+1,\mu} \\ 0 & F_{\rho+1} \end{bmatrix} \begin{bmatrix} \chi_\mu(\rho+2) \\ \chi_\mu(\rho+1) \end{bmatrix} \\ = & \begin{bmatrix} G_{\rho,\mu} & 0 \\ D_{\rho,\mu} & -G_{\rho-1,\mu} \end{bmatrix} \begin{bmatrix} \chi_\mu(\rho) \\ \chi_\mu(\rho-1) \end{bmatrix} + \begin{bmatrix} H_{\rho+1,\mu} & 0 \\ 0 & -H_{\rho,\mu} \end{bmatrix} \begin{bmatrix} w(\rho+1) \\ w(\rho) \end{bmatrix} \end{aligned} \tag{6.5}$$

and

$$\begin{bmatrix} -F_{\rho+1,\mu} & D_{\rho,\mu} \\ 0 & I \end{bmatrix} \begin{bmatrix} \chi_\mu(\rho+1) \\ \chi_\mu(\rho) \end{bmatrix} = \begin{bmatrix} 0 & G_{\rho,\mu} \\ -I & 0 \end{bmatrix} \begin{bmatrix} \chi_\mu(\rho) \\ \chi_\mu(\rho-1) \end{bmatrix} + \begin{bmatrix} H_{\rho,\mu} \\ 0 \end{bmatrix} w_\mu(\rho) \tag{6.6}$$

which by making obvious substitutions equations (6.5) has the form

$$E_{\rho+2,\mu} X_\mu(\rho+2) = A_{\rho,\mu} X_\mu(\rho) + B_{\rho,\mu} U_\mu(\rho) \tag{6.7}$$

and equation (6.6) has the form

$$E'_{\rho+1,\mu} X_\mu(\rho+1) = A'_{\rho,\mu} X_\mu(\rho) + B'_{\rho,\mu} w_\mu(\rho) \tag{6.8}$$

The boundary conditions for both models can be written as

$$\begin{aligned} E_{0,\mu} X_\mu(0) &= B_{-1,\mu} U(-1,\mu) \\ A_{R,\mu} X_\mu(R) &= -B_{R,\mu} U_\mu(R) \end{aligned} \tag{6.9}$$

Since the NNM can be fully described by a STPBVDS where the state dimension varies with size, the Mayne-Fraser, and the Rauch-Tung-Striebel algorithms, developed in Chapter 4 for STPBVDS's can be applied to two-dimensional systems with no loss of generality. With this model and the Rauch-Tung-Striebel algorithm, the local processing for each of the local subregions in our parallel processing algorithm has been defined. Specifically, in our parallel algorithm, filtering will begin in parallel from the center of each interval to the boundary, where the local representation of the process in each subregion will be a STPBVDS whose initial state represents the $x(i,j)$ located at the center of the local region. Local estimates of the boundaries are computed based on local data and dynamic constraints. After the interprocessor communication step has been completed, the last step of the parallel smoothing algorithm is the application of the backward sweep of the Rauch-Tung-Striebel algorithm to update the interior states in the local subregions to their smoothed values.

6.2.1 Computation of the system parameter for the 2-D STPBVDS

Before continuing with an explanation of the interprocessor communication step, we will provide the matrices $F_{p,\mu}$, $G_{p,\mu}$, $D_{p,\mu}$, and $H_{p,\mu}$. We will define the vector $\chi_\mu(\rho)$ to begin with the northern most element such that (in local coordinates) $i = 0$. In Figures 6-2 and 6-3, this element is denoted by the shaded dot. The remaining elements of $\chi_\mu(\rho)$ represent the elements of the state proceeding in a counter-clockwise direction. To aid us in defining these matrices we will define a $*$ operator. The action of this operator is defined by performing the following operations on the set of elements $\{N, S, E, W, I\}$.

$$\{N^*, S^*, E^*, W^*, I^*\} = \{S^T, N^T, W^T, E^T, I\} \tag{6.10}$$

Then when we consider any matrix formed from $\{N, S, E, W, I\}$, such as $F_{p,u}$, we use the notation $F^*_{p,u}$ to denote the matrix formed in exactly the same way from $\{N^*, S^*, E^*, W^*, I^*\}$.

We also define a scalar $n_{\rho,\mu}$ which is the dimension of the vector $\chi_\mu(\rho)$. First we define the system matrices when $\mu = 1$.

$$n_{\rho,1} = 4\rho + \delta(\rho) \tag{6.11}$$

$$\begin{aligned} D_{\rho,1} &= I \\ Trace[D_{\rho,1}] &= n_{1,\rho} \end{aligned} \tag{6.12}$$

$$G^T_{\rho,1} = F^*_{\rho+1,1} \tag{6.13}$$

$$F_{\rho,1} \in R^{n_{1,\rho-1} \times n_{1,\rho}} \tag{6.14}$$

Let $F_{\rho,1}(j,k)$ indicate the (j,k) element of $F_{\rho,1}$. Then

$$\begin{aligned}
&F_{\rho,1}(j,k) = W \quad j-1 = i && k-1 = i+1 \\
&F_{\rho,1}(j,k) = S \quad j-1 = \rho-1+i && k-1 = \rho+1+i \\
&F_{\rho,1}(j,k) = E \quad j-1 = (2(\rho-1)+i) mod\ n_{\rho-1,1} && k-1 = (2\rho+1+i) mod\ n_{\rho,1} \\
&F_{\rho,1}(j,k) = N \quad j-1 = (3(\rho-1)+i) mod\ n_{\rho-1,1} && k-1 = (3\rho+1+i) mod\ n_{\rho,1} \\
&0 \le i \le 2\rho-2
\end{aligned} \tag{6.15}$$

$$\begin{aligned}
&H_{\rho,1} = \mathrm{diag}(B,B,\cdots,B) \\
&H_{\rho,1} \in \Re^{n_{\rho,1} \times n_{\rho,1}}
\end{aligned} \tag{6.16}$$

$$\begin{aligned}
&H_{\rho,1} = \mathrm{diag}(B,B,\cdots,B) \\
&H_{\rho,1} \in \Re^{N_{\rho,1} \times N_{\rho,1}}
\end{aligned} \tag{6.17}$$

For the case where $\mu = \infty$, we have the following

$$n_{\infty,\rho} = 8\rho + \delta(\rho) \tag{6.18}$$

We define the matrix Z_ρ to be equal to zero except for the following entries

$$\begin{aligned}
&Z_\rho(j,j+1) = S && j = \rho+i \\
&Z_\rho(j,j+1) = E && j = 3\rho+i \\
&Z_\rho(j,j+1) = N && j = 5\rho+i \\
&Z_\rho(1+[(j-1) mod\ n_{\infty,\rho}], 2+[(j-1) mod\ n_{\infty,\rho}]) = W && j = 7\rho+i \\
&1 \le i \le 2\rho
\end{aligned} \tag{6.19}$$

$$D_{\rho,\infty} = I - Z_\rho - Z_\rho^{*T} \tag{6.20}$$

$$G_{\rho,\infty}^T = F_{\rho+1,\infty}^* \tag{6.21}$$

$$F_{\rho,\infty} \in R^{n_{\infty,\rho-1} \times n_{\infty,\rho}} \tag{6.22}$$

$$
\begin{aligned}
&F_{\rho,1}(j,k) = W \quad j = \rho + i && k = \rho + i + 2 \\
&F_{\rho,1}(j,k) = S \quad j = 3\rho - 2 + i && k = 3\rho + 2 \\
&F_{\rho,1}(j,k) = E \quad j = 5\rho - 4 + i && k = 5\rho + 2 + i \\
&F_{\rho,1}(j,k) = N \quad j - 1 = (7\rho - 7 + i) mod\ n_{\rho-1,\infty} && k - 1 = (7\rho + 1 + i) mod\ n_{\rho,\infty} \\
&0 \leq i \leq 2\rho - 2
\end{aligned}
\tag{6.23}
$$

$$
\begin{aligned}
H_{\rho,\infty} &= \mathrm{diag}(B, B, \cdots, B) \\
H_{\rho,\infty} &\in \Re^{n_{\rho,1} \times n_{\rho,1}}
\end{aligned}
\tag{6.24}
$$

6.2.2 Interprocessor Communication for Two-dimensional smoothing

The interprocessor communication step in this section bears strong resemblance to the Rauch-Tung-Striebel algorithm. The issue of applying the Rauch-Tung-Striebel algorithm to the parallel processing step, requires a natural notion of a time direction which is consistent with the types of two-dimensional systems which we are examining. The systems which we are examining are reciprocal, that is, given a closed contour, the interior and exterior are independent. This closed contour we have defined, in the last subsection, as a 'state' of the system.

We will define an order relation $\preceq$ on the set of closed contours on which states are defined. For these models closed contours are 'thick'. For example, all (i,j) such that $\rho - 1 \leq ||(i,j)||_\mu \leq \rho$, comprise a closed contour for each ρ. Assume that our coordinate system is shifted such that $(i,j) = (0,0)$ falls in the center of the region whose boundary corresponds to the state $X_\mu(\tau)$. Any closed contour X such that for all $x(i,j) \in X$, $||(i,j)||_\mu \leq \tau$ satisfies $X \preceq X_\mu(\tau)$. We have therefore imposed a partial ordering of the states. In particular, Figure 6-4 shows that the state $X \preceq X_\mu(\tau)$. In Figure 7.2, X and $X_\mu(\tau)$ are not ordered with respect to each other.

Continuing in more detail, we next describe the filtering process. We will assume that $\mu = \infty$ unless otherwise stated. We start with a square region and partition it into 2^{2M} subregions by partitioning each side of the square region into 2^M subregions, as shown in Figure 6-6 for $M = 3$, and the processors assigned to each subregion

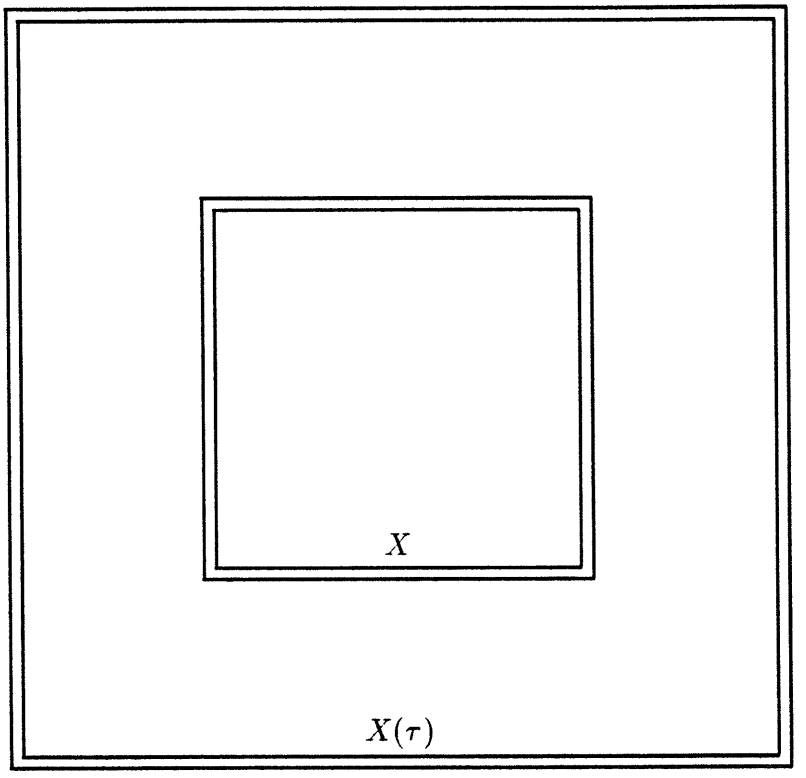


Figure 6-4: Two ordered 'states'

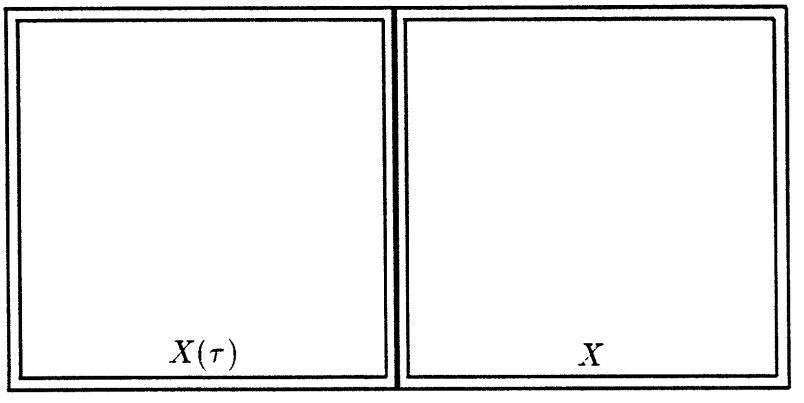


Figure 6-5: Two non-ordered 'states'

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)	(3,7)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)	(4,7)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)	(5,7)
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)	(6,7)
(7,0)	(7,1)	(7,2)	(7,3)	(7,4)	(7,5)	(7,6)	(7,7)

Figure 6-6: A 2-D region is partitioned into squares. Each region is assigned a cartesian coordinate.

filter outward from the center to produce local estimates of the boundaries of each subinterval. We will give each processor and the region to which it is assigned an integer cartesian coordinate to assist in describing this algorithm. In order to describe larger rectangles composed of individual square regions, we will denote a rectangle whose corners are given by (s_1, t_1), (s_1, t_2), (s_2, t_1), and (s_2, t_2), by $(\{s_1, s_2\}, \{t_1, t_2\})$ as shown in Figure 6-7.

Step 1

The model on which this preprocessing step is performed is provided in the previous subsection. Filtering is performed in each subregion from the center to the boundary of each subinterval using the FMLF filtering equations (4.52), (4.53), and (4.54)), applied to the model described in Section 6.2.1.

Step 2

We will analyze the computations involved for a square region divided into four subregions. Again there are several parts to this step.

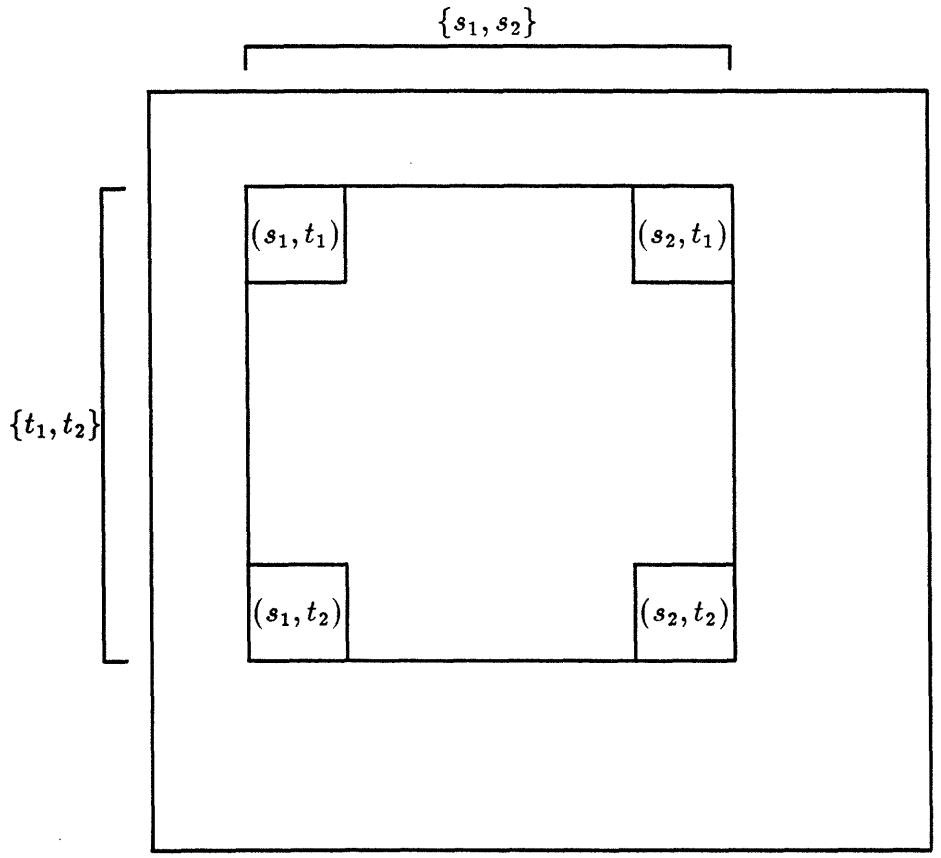


Figure 6-7: The notation $(\{s_1, s_2\}, \{t_1, t_2\})$ is used to refer to rectangles constructed from the union of individual square regions. Similarly the boundary for this region is denoted by $X(\{s_1, s_2\}, \{t_1, t_2\})$

Part 1

Part 1 involves computing the optimal estimates of the boundaries of two neighboring regions from local measurements of these boundaries and any dynamic constraints which may link them (which we may interpret as an observation). After the local filtering step, the boundaries of the local subregions satisfy a nearest neighbor model which in essence is a a sampling of the states of the original system. We will denote the boundary of a given subregion by $X(s,t)$, where the label s,t, consisting of a pair of integers, represents the location of the entire region on a square grid as shown in Figure 6-6. $X(s,t)$ for a specific value of s, and t therefore represents a collection of $x(i,j)$. The nearest neighbor relationship among the boundary vectors can be written as

$$\begin{aligned} A_o X(s,t) &= A_n X(s,t+1) \\ &+ A_s X(s,t-1) \\ &+ A_e X(s+1,t) \\ &+ A_w X(s-1,t) \\ &+ B_o U_o(s,t) \end{aligned} \tag{6.25}$$

with 'observations' which are computed locally given by

$$\hat{X}(s,t) = X(s,t) + \tilde{X}(s,t) \tag{6.26}$$

The A_j's are written only to show the form of the NNM relationship. These matrices will not be written explicitly as were the $F_{\rho,\mu}$, and $G_{\rho,\mu}$ matrices in Section 6.2.1. We will however explore the structure of these equations further. Specifically by examining the structure and the local nature of the dynamic constraints, the relation

between $X(s,t)$ and its neighbors can also be written in the following form.

$$\begin{aligned}
H_n X(s,t) &= \mathcal{N} X(s,t+1) + B_n U_n(s,t) \\
H_s X(s,t) &= \mathcal{S} X(s,t-1) + B_s U_s(s,t) \\
H_e X(s,t) &= \mathcal{E} X(s+1,t) + B_e U_e(s,t) \\
H_w X(s,t) &= \mathcal{W} X(s-1,t) + B_w U_w(s,t) \\
H_{ne} X(s,t) &= \mathcal{N}_e X(s,t+1) + \mathcal{E}_n X(s+1,t) + B_{ne} U_{ne}(s,t) \\
H_{se} X(s,t) &= \mathcal{S}_e X(s,t-1) + \mathcal{E}_s X(s+1,t) + B_{se} U_{se}(s,t) \\
H_{nw} X(s,t) &= \mathcal{N}_w X(s,t+1) + \mathcal{W}_n X(s-1,t) + B_{nw} U_{nw}(s,t) \\
H_{sw} X(s,t) &= \mathcal{S}_w X(s,t-1) + \mathcal{W}_s X(s-1,t) + B_{sw} U_{sw}(s,t)
\end{aligned} \tag{6.27}$$

Let us now take some time to examine the construction of these matrices. This is basically a bookkeeping problem which requires specifying an explicit ordering of the elements in vectors of the form $X(s,t)$. To do this we refer to the construction of $X(\rho)$, i.e., the vector of values that are a distance ρ, and $\rho - 1$ away from the origin (at the center of the region). Let us use θ to denote the variable indexing the elements of this vector, (i.e. the components [1] of $X(\rho)$ are $X_\theta(\rho)$), where by counting we find that

$$1 \leq \theta \leq 16\rho - 8 \tag{6.28}$$

The way in which we order the elements x(i,j) which comprise $X(\rho)$ is given by a function $\Theta(i,j,\rho)$,where since our boundaries are thick (since our model is second order), for each (i,j), $\Theta(i,j,\rho)$ is defined for two values of ρ (i.e. $x(i,j)$ appears in $X(\rho)$ for two values of ρ.

Furthermore Θ is invertible for a fixed ρ. Specifically given $\Theta(i,j,\rho)$ we can uniquely define the integer functions $i(\theta,\rho)$ and $j(\theta,\rho)$. Figure 6-8 shows the values of *theta* for $\rho = 3$. The matrices H_n, H_s, H_e, H_w, etc. are matrices composed of $n \times n$ blocks. These blocks will be referenced directly when defining these matrices.

[1] Note that each 'component' here is a value of $x(i,j)$, which in itself, can be a vector.

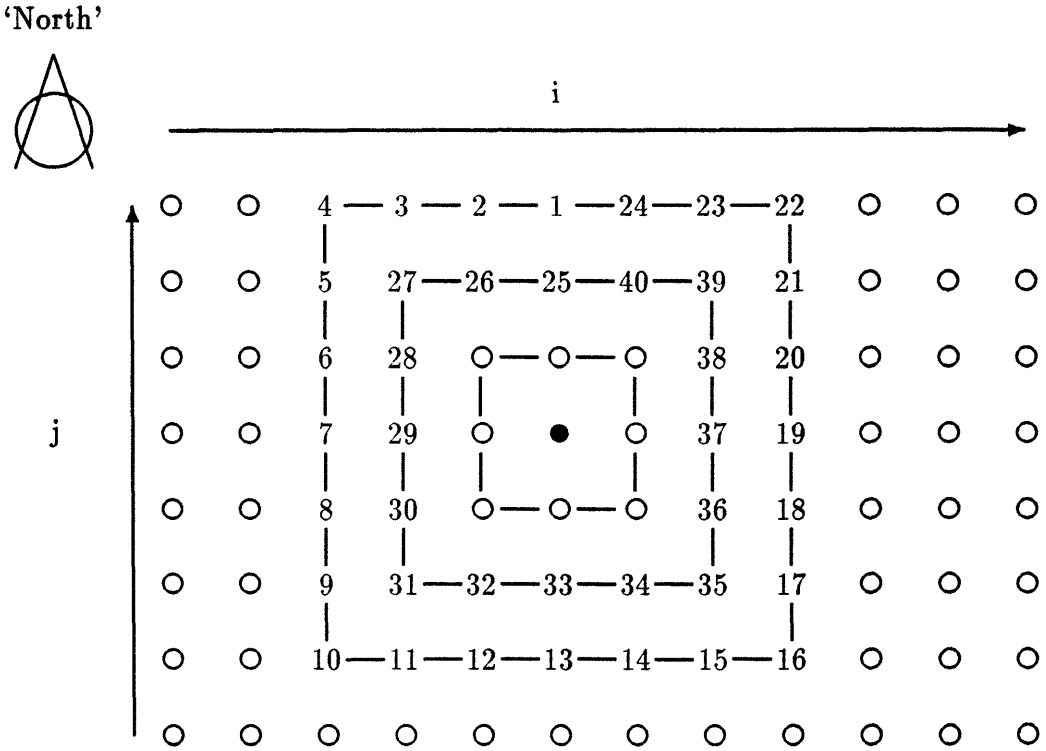


Figure 6-8: The integer function $\Theta(i,j,3)$. The shaded dot is the origin, $(i,j)=(0,0)$. This function orders the elements $x(i,j)$ in the vector $X(\rho)$.

$$H_s \in \Re^{(2\rho-1)n\times(16p-8)n} \tag{6.29}$$

$$\theta \in \{3\rho+2,\cdots,5\rho\} \tag{6.30}$$

$$H_s(\theta-3\rho-1,\theta)=I \tag{6.31}$$

$$H_s(\theta-3\rho-1,\Theta(i(\theta,\rho),1+j(\theta,\rho),\rho))=-N \tag{6.32}$$

$$H_s(\theta-3\rho-1,\Theta(1+i(\theta,\rho),j(\theta,\rho),\rho))=-E \tag{6.33}$$

$$H_s(\theta-3\rho-1,\Theta(-1+i(\theta,\rho),j(\theta,\rho),\rho))=-W \tag{6.34}$$

$$H_e \in \Re^{(2\rho-1)n\times(16p-8)n} \tag{6.35}$$

$$\theta \in \{5\rho+2,\cdots,7\rho\} \tag{6.36}$$

$$H_e(\theta-5\rho-1,\theta)=I \tag{6.37}$$

$$H_e(\theta-5\rho-1,\Theta(i(\theta,\rho),1+j(\theta,\rho),\rho))=-N \tag{6.38}$$

$$H_e(\theta - 5\rho - 1, \Theta(i(\theta,\rho), -1 + j(\theta,\rho), \rho)) = -S \tag{6.39}$$

$$H_e(\theta - 5\rho - 1, \Theta(-1 + i(\theta,\rho), j(\theta,\rho), \rho)) = -W \tag{6.40}$$

$$H_w \in \Re^{(2\rho-1)n \times (16p-8)n} \tag{6.41}$$

$$\theta \in \{\rho + 2, \cdots, 3\rho\} \tag{6.42}$$

$$H_w(\theta - \rho - 1, \theta) = I \tag{6.43}$$

$$H_w(\theta - \rho - 1, \Theta(i(\theta,\rho), 1 + j(\theta,\rho), \rho)) = -N \tag{6.44}$$

$$H_w(\theta - \rho - 1, \Theta(1 + i(\theta,\rho), j(\theta,\rho), \rho)) = -E \tag{6.45}$$

$$H_w(\theta - \rho - 1, \Theta(i(\theta,\rho), -1 + j(\theta,\rho), \rho)) = -S \tag{6.46}$$

$$H_n \in \Re^{(2\rho-1)n \times (16p-8)n} \tag{6.47}$$

$$\theta \in \{7\rho + 2, \cdots, 8\rho, 1, \cdots, \rho\} \tag{6.48}$$

$$H_n(\theta - 7\rho - 1, \theta) = I \quad \theta \in \{7\rho + 2, \cdots, 8\rho\} \tag{6.49}$$

$$H_n(\theta - 1 + \rho, \theta) = I \quad \theta \in \{1, \cdots, \rho\} \tag{6.50}$$

$$H_n(\theta - 7\rho - 1, \Theta(i(\theta,\rho), -1 + j(\theta,\rho), \rho)) = -S \quad \theta \in \{7\rho + 2, \cdots, 8\rho\} \tag{6.51}$$

$$H_n(\theta + \rho - 1, \Theta(i(\theta,\rho), -1 + j(\theta,\rho), \rho)) = -S \quad \theta \in \{1, \cdots, \rho\} \tag{6.52}$$

$$H_n(\theta - 7\rho - 1, \Theta(1 + i(\theta,\rho), j(\theta,\rho), \rho)) = -E \quad \theta \in \{7\rho + 2, \cdots, 8\rho\} \tag{6.53}$$

$$H_n(\theta + \rho - 1, \Theta(1 + i(\theta,\rho), j(\theta,\rho), \rho)) = -E \quad \theta \in \{1, \cdots, \rho\} \tag{6.54}$$

$$H_n(\theta - 7\rho - 1, \Theta(-1 + i(\theta,\rho), j(\theta,\rho), \rho)) = -W \quad \theta \in \{7\rho + 2, \cdots, 8\rho\} \tag{6.55}$$

$$H_n(\theta + \rho - 1, \Theta(-1 + i(\theta,\rho), j(\theta,\rho), \rho)) = -W \quad \theta \in \{1, \cdots, \rho\} \tag{6.56}$$

$$H_{ne} \in \Re^{n \times (16p-8)n} \tag{6.57}$$

$$H_{ne}(7\rho + 1) = I \tag{6.58}$$

$$H_{ne}(7\rho) = -S \tag{6.59}$$

$$H_{ne}(7\rho + 2) = -W \tag{6.60}$$

$$H_{se} \in \Re^{n \times (16p-8)n} \tag{6.61}$$

$$H_{se}(5\rho + 1) = I \tag{6.62}$$

$$H_{se}(5\rho) = -W \tag{6.63}$$

$$H_{se}(5\rho + 2) = -N \tag{6.64}$$

$$H_{nw} \in \Re^{n \times (16p-8)n} \tag{6.65}$$

$$H_{nw}(\rho + 1) = I \tag{6.66}$$

$$H_{nw}(\rho) = -E \tag{6.67}$$

$$H_{nw}(\rho + 2) = -S \tag{6.68}$$

$$H_{sw} \in \Re^{n \times (16p-8)n} \tag{6.69}$$

$$H_{sw}(3\rho + 1) = I \tag{6.70}$$

$$H_{sw}(3\rho) = -N \tag{6.71}$$

$$H_{sw}(3\rho + 2) = -E \tag{6.72}$$

$$\mathcal{S} \in \Re^{(2\rho-1)n \times (16p-8)n} \tag{6.73}$$

$$\theta \in \{3\rho + 2, \cdots, 5\rho\} \tag{6.74}$$

$$\mathcal{S}(\theta - 3\rho - 1, \Theta(i(\theta, \rho), 2\rho + j(\theta, \rho), \rho)) = -S \tag{6.75}$$

$$\mathcal{E} \in \Re^{(2\rho-1)n \times (16p-8)n} \tag{6.76}$$

$$\theta \in \{5\rho + 2, \cdots, 7\rho\} \tag{6.77}$$

$$\mathcal{E}(\theta - 5\rho - 1, \Theta(-2\rho + i(\theta, \rho), j(\theta, \rho), \rho)) = -E \tag{6.78}$$

$$\mathcal{W} \in \Re^{(2\rho-1)n \times (16p-8)n} \tag{6.79}$$

$$\theta \in \{\rho + 2, \cdots, 3\rho\} \tag{6.80}$$

$$\mathcal{W}(\theta-\rho-1,\Theta(2\rho+i(\theta,\rho),j(\theta,\rho),\rho))=-W \tag{6.81}$$

$$\mathcal{N}\in\Re^{(2\rho-1)n\times(16p-8)n} \tag{6.82}$$

$$\theta\in\{7\rho+2,\cdots,8\rho,1,\cdots,\rho\} \tag{6.83}$$

$$\mathcal{N}(\theta-7\rho-1,\Theta(i(\theta,\rho),-2\rho+j(\theta,\rho),\rho))=-N \quad \theta\in\{7\rho+2,\cdots,8\rho\} \tag{6.84}$$

$$\mathcal{N}(\theta+\rho-1,\Theta(i(\theta,\rho),-2\rho+j(\theta,\rho),\rho))=-N \qquad \theta\in\{1,\cdots,\rho\} \tag{6.85}$$

$$\mathcal{N}_e\in\Re^{n\times(16p-8)n} \tag{6.86}$$

$$\mathcal{N}_e=N \tag{6.87}$$

$$\mathcal{E}_n\in\Re^{n\times(16p-8)n} \tag{6.88}$$

$$\mathcal{E}_n=E \tag{6.89}$$

$$\mathcal{S}_e\in\Re^{n\times(16p-8)n} \tag{6.90}$$

$$\mathcal{S}_e=S \tag{6.91}$$

$$\mathcal{E}_s\in\Re^{n\times(16p-8)n} \tag{6.92}$$

$$\mathcal{E}_s=E \tag{6.93}$$

$$\mathcal{N}_w\in\Re^{n\times(16p-8)n} \tag{6.94}$$

$$\mathcal{N}_w=N \tag{6.95}$$

$$\mathcal{W}_n\in\Re^{n\times(16p-8)n} \tag{6.96}$$

$$\mathcal{W}_n=W \tag{6.97}$$

$$\mathcal{S}_w \in \Re^{n\times(16p-8)n} \tag{6.98}$$

$$\mathcal{S}_w = S \tag{6.99}$$

$$\mathcal{W}_s \in \Re^{n\times(16p-8)n} \tag{6.100}$$

$$\mathcal{W}_s = W \tag{6.101}$$

$$B_n \in \Re(2\rho+1)n \times (2\rho+1)n \tag{6.102}$$

$$B_n = B_s = B_e = B_w = \mathrm{diag}(B,\cdots,B) \tag{6.103}$$

$$B_{ne} = B_{se} = B_{nw} = B_s w = B \tag{6.104}$$

where N, S, E, W, and B is defined in Section 6.2. Now we will return to the discusion of merging two boundaries.

Two states to be combined in this step are shown in Figure 6-9. In this formulation the boundaries of the neighboring regions do not intersect, and there are dynamic constraints which link the two regions. The arrows in Figure 6-9 show the dynamic constraints required for the measurement update step. The 'measurement' required to compute the estimate of the boundary which are the shaded circles in Figure 6-10 and which encloses both regions is given by

$$\begin{aligned} H_e X(s,t) &= \mathcal{E}X(s+1,t) + B_e U_e(s,t) \\ H_w X(s+1,t) &= \mathcal{W}X(s,t) + B_w U_w(s+1,t) \\ \hat{X}(s,t) &= X(s,t) + \tilde{X}(s,t) \\ \hat{X}(s+1,t) &= X(s+1,t) + \tilde{X}(s+1,t) \end{aligned} \tag{6.105}$$

All of the observation noises shown in equation (6.105) are independent. The $x(i,j)$ which are estimated in this problem are contributed from the boundary of Region (s,t) and from Region $(s+1,t)$. The elements of this boundary can be separated into those which comprise the boundary which enclose both Region (s,t) and Region $(s+1,t)$ and the remainder which we will call $\{\Upsilon(\{s,s+1\},t)\}$. The elements of $\{\Upsilon(\{s,s+1\},t)\}$

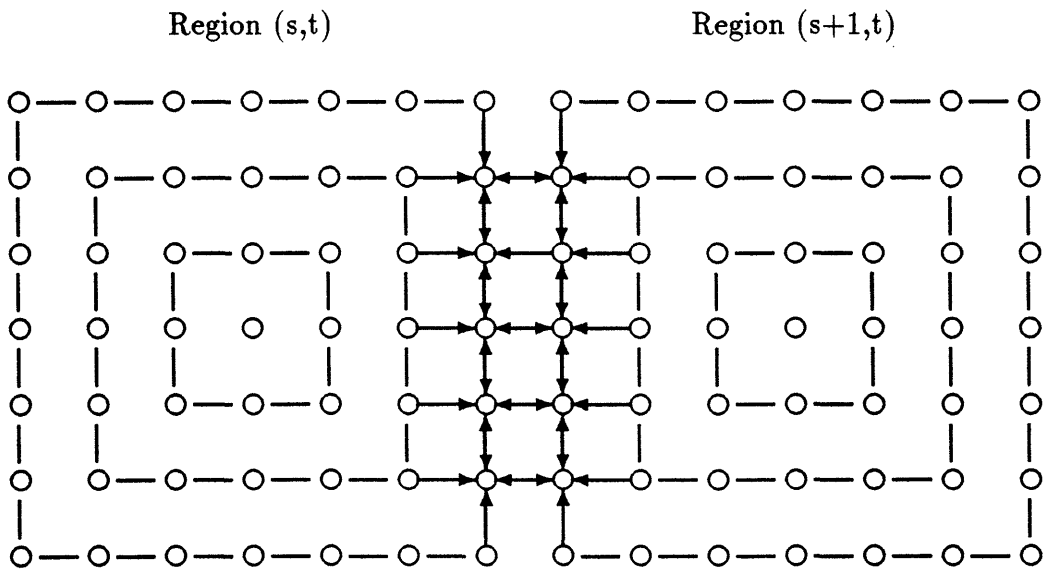


Figure 6-9: Two neighboring regions for the Measurement Update step and the dynamic constraints (which are shown by the arrows) required to merge them.

are enclosed in squares in Figure 6-10. The boundary denoted by $\{X(\{s,s+1\},t)\}$ are the shaded circles in Figure 6-10.

$$\begin{aligned} &\{X(s,t)\} \cup \{X(s+1,t)\} = \{X(\{s,s+1\},t)\} \cup \{\Upsilon(\{s,s+1\},t)\} \\ &\{\Upsilon(\{s,s+1\},t)\} \cap \{X(\{s,s+1\},t)\} = \phi \end{aligned} \tag{6.106}$$

The result of the estimation problem based on the measurements in (6.105) is the following estimate

$$\begin{bmatrix} \hat{X}(s,t|\{s,s+1\},t) \\ \hat{X}(s+1,t|\{s,s+1\},t) \end{bmatrix} = \begin{bmatrix} X(s,t) \\ X(s+1,t) \end{bmatrix} + \begin{bmatrix} \tilde{X}(s,t|\{s,s+1\},t) \\ \tilde{X}(s+1,t|\{s,s+1\},t) \end{bmatrix} \tag{6.107}$$

equivalently we could write

$$\begin{bmatrix} \hat{X}(\{s,s+1\},t|\{s,s+1\},t) \\ \hat{\Upsilon}(\{s,s+1\},t|\{s,s+1\},t) \end{bmatrix} = \begin{bmatrix} X(\{s,s+1\},t) \\ \Upsilon(\{s,s+1\},t) \end{bmatrix} + \begin{bmatrix} \tilde{X}(\{s,s+1\},t|\{s,s+1\},t) \\ \tilde{\Upsilon}(\{s,s+1\},t|\{s,s+1\},t) \end{bmatrix} \tag{6.108}$$

The next step amounts to sampling $[X^T(\{s,s+1\},t)\Upsilon^T(\{s,s+1\},t)]^T$ by keeping the $X^T(\{s,s+1\},t)$ which is the process on the boundary which contains both of the

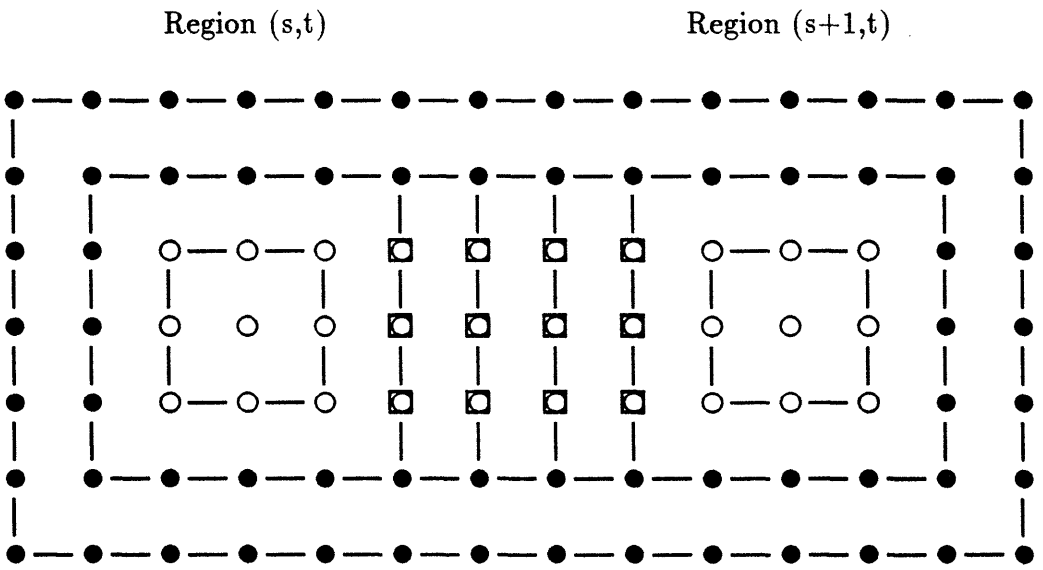


Figure 6-10: Constructing the boundary enclosing two regions from the boundaries of two subregions $X(s,t)$ and $X(s+1,t)$. The black dots represent the boundary which encloses both regions which will be denoted by $X(\{s,s+1\},t)$. The twelve dots which are enclosed by squares are the elements of $\Upsilon(\{ss+1\},t)$. $\Upsilon(\{s,s+1\},t)$ are elements of the union of $X(s,t)$ and $X(s+1,t)$ but are not elements of $X(\{s,s+1\},t)$.

subregions. Sampling is performed in the following

$$\hat{X}(\{s,s+1\},t|\{s,s+1\},t) = \begin{bmatrix} I & 0 \end{bmatrix} \begin{bmatrix} \hat{X}(\{s,s+1\},t|\{s,s+1\},t) \\ \hat{\Upsilon}(\{s,s+1\},t|\{s,s+1\},t) \end{bmatrix} \quad (6.109)$$

Since sampling is a noiseless process, the covariance is given by

$$Cov[\tilde{X}(\{s,s+1\},t|\{s,s+1\},t)] = \begin{bmatrix} I & 0 \end{bmatrix} Cov \begin{bmatrix} \tilde{X}(\{s,s+1\},t|\{s,s+1\},t) \\ \tilde{\Upsilon}(\{s,s+1\},t|\{s,s+1\},t) \end{bmatrix} \begin{bmatrix} I \\ 0 \end{bmatrix} \quad (6.110)$$

Similarly the same computations can be performed for Region (s, t-1) and Region (s+1,t-1) by substituting t-1 for t in Equations (6.106)-(6.110).

Now with an estimate of the boundary surrounding Region (s,t) and Region $(s+1,t)$ (i.e., Region $(\{s,s+1\},t)$), and an estimate of the boundary surrounding Region $(s,t-1)$ and Region $(s+1,t-1)$ (i.e., Region $(\{s,s+1\},t-1)$), they can be combined into an estimate of the boundary for the entire region which comprises Region $(\{s,s+1\},t)$ and Region $(\{s,s+1\},t-1)$ (i.e., Region $(\{s,s+1\},\{t-1,t\})$).

Figure 6-11 shows the dynamic constraints necessary to merge these two regions. These dynamic constraints are represented here by

$$\begin{aligned}
H_n X(s,t-1) &= \mathcal{N} X(s,t) \\
H_s X(s,t) &= \mathcal{S} X(s,t-1) \\
H_n X(s+1,t-1) &= \mathcal{N} X(s+1,t) \\
H_s X(s+1,t) &= \mathcal{S} X(s+1,t-1) \\
H_{ne} X(s,t-1) &= \mathcal{N}_e X(s,t) + \mathcal{E}_n X(s+1,t-1) \\
H_{nw} X(s+1,t-1) &= \mathcal{N}_w X(s+1,t) + \mathcal{W}_n X(s,t-1) \\
H_{se} X(s,t) &= \mathcal{S}_e X(s,t-1) + \mathcal{E}_s X(s+1,t) \\
H_{sw} X(s+1,t) &= \mathcal{S}_w X(s+1,t-1) + \mathcal{W}_n X(s,t)
\end{aligned} \tag{6.111}$$

By examining Figure 6-11 we see that all of the dynamic constraints have a similar structure. The the last four equations in (6.111) appear different only because of the partitioning of the plane into subregions. This suggests a different ordering of the points on the plane than the one used here. We will return to this point later.

The elements of the two boundaries of each subregion can be ordered into two vectors where one represents the boundary of the entire region which is shown as the set of shaded circles in Figure 6-11 and will be denoted by $X(\{s,s+1\},\{t-1,t\})$ and the remaining states will be denoted by $\Upsilon(\{s,s+1\},\{t-1,t\})$ The estimates which have been obtained are denoted by

$$\begin{aligned}
&\begin{bmatrix} \hat{X}(\{s,s+1\},\{t-1,t\}|\{s,s+1\},\{t-1,t\}) \\ \hat{\Upsilon}(\{s,s+1\},\{t-1,t\}|\{s,s+1\},\{t-1,t\}) \end{bmatrix} \\
&= \begin{bmatrix} X(\{s,s+1\},\{t-1,t\}) \\ \Upsilon(\{s,s+1\},\{t-1,t\}) \end{bmatrix} + \begin{bmatrix} \tilde{X}(\{s,s+1\},\{t-1,t\}|\{s,s+1\},\{t-1,t\}) \\ \tilde{\Upsilon}(\{s,s+1\},\{t-1,t\}|\{s,s+1\},\{t-1,t\}) \end{bmatrix}
\end{aligned} \tag{6.112}$$

The next step of the estimation process would incorporate the estimates

$\hat{X}(\{s,s+1\},\{t-1,t\}|\{s,s+1\},\{t-1,t\})$,

$\hat{X}(\{s+2,s+3\},\{t-1,t\}|\{s+2,s+3\},\{t-1,t\})$,

$\hat{X}(\{s,s+1\},\{t-3,t-2\}|\{s,s+1\},\{t-3,t-2\})$,

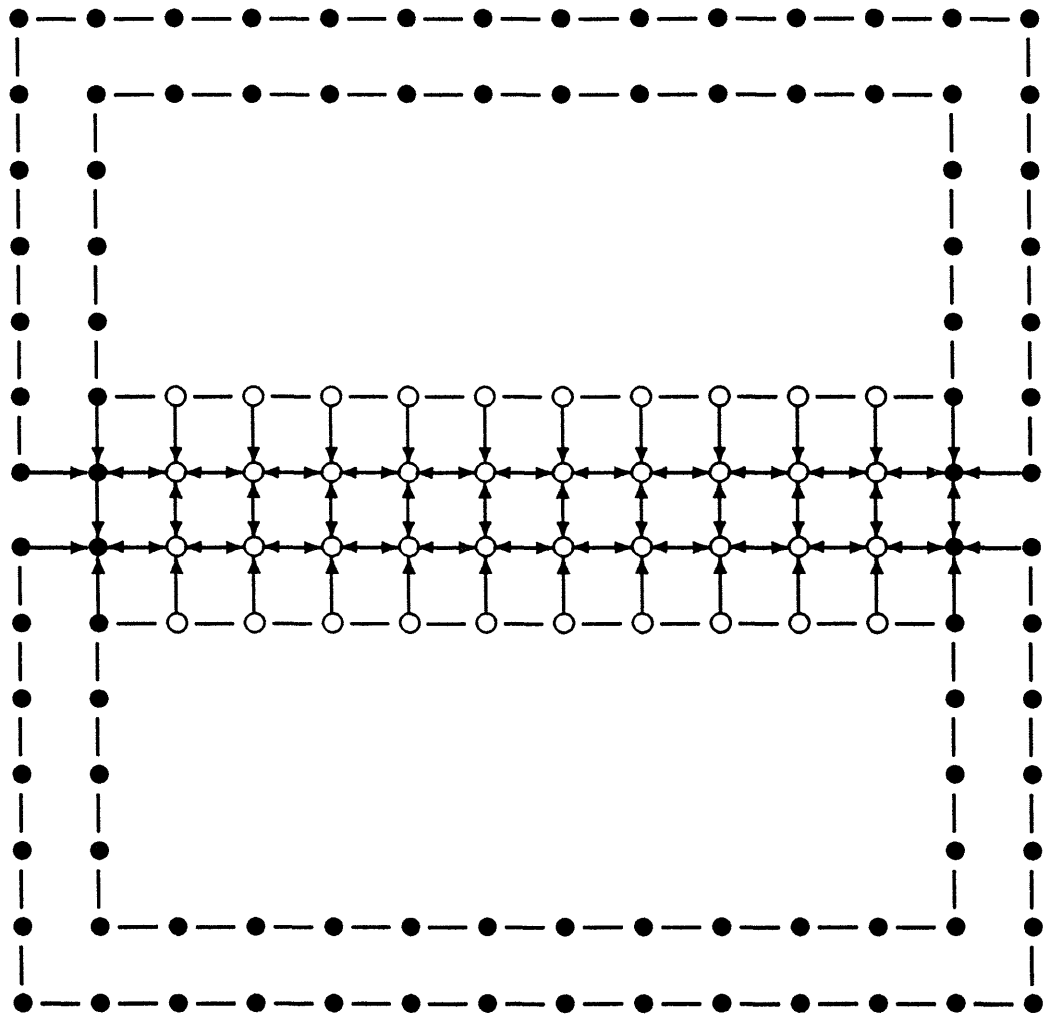


Figure 6-11: The black dots represent the boundary which encloses the two regions which are merged together, $X(\{s, s+1\}, \{t-1, t\})$. The white dots represents the remaining elements of the boundaries of the two original regions which will be denoted by $\Upsilon(\{s, s+1\}, \{t-1, t\})$

and $\hat{X}(\{s+2,s+3\},\{t-3,t-2\}|\{s+1,s+3\},\{t-3,t-2\})$
to construct the estimate of the boundary surrounding region given by
$\hat{X}(\{s,s+3\},\{t-3,t\}|\{s,s+3\},\{t-3,t\})$.

Part 2

Once the smoothed estimate for the boundary of the entire region has been obtained, smoothed estimates are propagated to the boundaries of smaller and smaller subregions until each subregion has a smoothed boundary. Typical measurements for this step are given by the following.

$$\begin{bmatrix} \hat{X}(\{s,s+1\},\{t-1,t\}|\{s,s+1\},\{t-1,t\}) \\ \hat{\Upsilon}(\{s,s+1\},\{t-1,t\}|\{s,s+1\},\{t-1,t\}) \\ \hat{X}(\{s,s+1\},\{t-1,t\}|\{0,S\},\{0,T\}) \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \\ I & 0 \end{bmatrix} \begin{bmatrix} X(\{s,s+1\},\{t-1,t\}) \\ \Upsilon(\{s,s+1\},\{t-1,t\}) \end{bmatrix}$$
$$+ \begin{bmatrix} \tilde{X}(\{s,s+1\},\{t-1,t\}|\{s,s+1\},\{t-1,t\}) \\ \tilde{\Upsilon}(\{s,s+1\},\{t-1,t\}|\{s,s+1\},\{t-1,t\}) \\ \tilde{X}(\{s,s+1\},\{t-1,t\}|\{0,S\},\{0,T\}) \end{bmatrix} \tag{6.113}$$

where S and T represent the maximum values of s, and t respectively. This measurement has the same structure as the measurement given in (4.92) used to derive the Rauch-Tung-Striebel algorithm. From this measurement
$\Upsilon(\{s,s+1\},\{t-1,t\}|\{0,S\},\{0,T\})$ can be estimated and as a result the estimates $\hat{X}(\{s,s+1\},t|\{0,S\},\{0,T\})$, and $\hat{X}(\{s,s+1\},t-1|\{0,S\},\{0,T\})$ have been obtained. Next the following measurements are combined

$$\begin{bmatrix} \hat{X}(\{s,s+1\},t|\{s,s+1\},t) \\ \hat{\Upsilon}(\{s,s+1\},t|\{s,s+1\},t) \\ \hat{X}(\{s,s+1\},t|\{0,S\},\{0,T\}) \end{bmatrix} = \begin{bmatrix} I & 0 \\ 0 & I \\ I & 0 \end{bmatrix} \begin{bmatrix} X(\{s,s+1\},t) \\ \Upsilon(\{s,s+1\},t) \end{bmatrix}$$
$$+ \begin{bmatrix} \tilde{X}(\{s,s+1\},t|\{s,s+1\},t) \\ \tilde{\Upsilon}(\{s,s+1\},t|\{s,s+1\},t) \\ \tilde{X}(\{s,s+1\},t|\{0,S\},\{0,T\}) \end{bmatrix} \tag{6.114}$$

From this measurement $\Upsilon(\{s,s+1\},\{t-1,t\}|\{0,S\},\{0,T\})$ can be estimated

and as a result the estimates $\hat{X}(s,t|\{0,S\},\{0,T\})$, and $\hat{X}(s+1,t|\{0,S\},\{0,T\})$ have been obtained.

This procedure continues until smoothed estimates are obtained for the boundaries of the individual subregions.

Step 3

Once the smoothed estimates of the individual subregions have been obtained, the backward sweep of the Rauch-Tung-Striebel algorithm proceeds in parallel in each subregion.

6.3 Complexity

The computational requirements of the algorithm we have just described, differ from those of the one dimensional algorithms in several respects. First the dimension of the state changes as a function of the ρ coordinate. Specifically in the local filtering step the complexity of the processing in a one dimensional system varied as Kn^3 where K is the length of the interval, and n is the dimension of the state. With the dimension of the state proportional to ρ we can expect the complexity of the off-line computations to rise as the *fourth* power of ρ. Since the on-line computation for a one-dimensional system varies as Kn^2 we may now expect that the complexity of the online computation varies as ρ^3 when summed over the entire region. Secondly in the interprocessor communication step, the size of the state roughly grows by the factor $\sqrt{2}$ at each step. The size of the state grows geometrically. We will see that the state grows so fast that we could ignore all computations except those involving the largest state, i.e the boundary.

Step 1

The key issue in computing the computational complexity for this step are to manage the bookkeeping of the computations in more detail. The important numbers to keep track of are the dimension of the state of the system, the dimension of the driving noises, and the observations. The dimension of the state $X(\rho)$ is given by

$$\begin{aligned} X(\rho) &\in \Re^{16n(\rho-\frac{1}{2})} \quad \rho \geq 2 \\ X(1) &\in \Re^{9n} \end{aligned} \tag{6.115}$$

The dimension of the predicted estimate Z_ρ (defined by equation (4.48) when applied

to the model given in (6.8)) is given by

$$Z(\rho) \in \Re^{16n(\rho-1)} \quad \rho \geq 2 \tag{6.116}$$

The dimension of the driving noise is given by

$$\begin{aligned} U(\rho) &\in \Re^{8\rho m} \quad \rho \geq 1 \\ U(0) &\in \Re^{m} \end{aligned} \tag{6.117}$$

The dimension of the observation is given by

$$\begin{aligned} Y(\rho) &\in \Re^{8\rho p} \quad \rho \geq 2 \\ Y(1) &\in \Re^{9p} \end{aligned} \tag{6.118}$$

The dimension of these vectors dictate the dimension of A_ρ, E_ρ, B_ρ, C_ρ in the STPBVDS model. The dimension of the state satisfies a simple polynomial in ρ agreeing for all ρ except $\rho = 1$. In the following we will approximate the dimension of $X(1)$ by $X(1) \in \Re^{8n}$ instead of $\Re^{9n}$ in order that the polynomial $16n(\rho - \frac{1}{2})$ specify the dimension for all ρ and we may minimize the number of extra computations needed to deal with having to generate separate equations for $\rho = 1$. The dimension of the matrices which describe the STPBVDS satisfy the following

$$A_\rho \in \Re^{16n\rho \times 16n(\rho-\frac{1}{2})} \tag{6.119}$$

$$E_{\rho+1} \in \Re^{16n(\rho+\frac{1}{2})\times 16n\rho} \tag{6.120}$$

$$B_\rho \in \Re^{16n(\rho-\frac{1}{2})\times 8\rho m} \tag{6.121}$$

$$C_\rho \in \Re^{8\rho m \times 16n(\rho-\frac{1}{2})} \tag{6.122}$$

With this, the computations necessary for the filtering step can be computed. The amount of computation was computed in the same manner as the computation in Chapter 5. Here however we need to take into account that all of the dimensions are time varying and therefore formulas for computational complexity are more complex.

Step 1

To determine the amount of computation involved in these computations we returned to the filtering equations in Section 4.2 and apply them to the model given in (6.8) while accounting for the fact that the dimension of the matrices vary with time. The number of operations were computed with the aid of the polynomial functions defined in Section 2.1 First we will compute the amount of operations for *one* time step (i.e., one value of ρ). Then we will compute the amount of operations over the entire region.

Off-line

One time step

If we assume that the state is not completely estimable then the amount of off-line computation required to compute the estimate of $X(\rho)$ from the estimate of $\hat{X}[\rho-1|\rho-1]$ in the filtering step is given by requires

$$\begin{aligned}
& \mathcal{M}(16(2\rho n - \tfrac{3}{2}n + \tfrac{1}{2}\rho p), 16n(\rho - \tfrac{1}{2}), 16(2\rho n - \tfrac{3}{2}n + \tfrac{1}{2}\rho p)) \\
+ \;& (16)^3[\rho^3(16n^3 + n^2p + 2n^2m) + \rho^2(-45n^3 - n^2p - 4n^2m) \\
+ \;& \rho(43.5n^3 + .25n^2p + 2n^2m) - 14.5n^3] \\
+ \;& \mathcal{E}(16n(\rho-1), 16n(\rho-1), 16n(\rho-1)) \\
= \;& (16)^3[\rho^3(70.33n^3 + 27n^2p + 5.25np^2 + .33p^3 + 2n^2m) \\
- \;& \rho^2(146n^3 + 37.5n^2p + 3.625np^2 + 4n^2m) \\
+ \;& \rho(118.75n^3 + 13n^2p + 2n^2m) \\
- \;& 42.125n^3]
\end{aligned} \tag{6.123}$$

If the state is estimable then the amount of off-line computation to compute $\hat{X}[\rho|\rho]$

from $\hat{X}[\rho-1|\rho-1]$ is given by

$$\begin{aligned}
& \mathcal{E}(16(2\rho n - \tfrac{3}{2}n + \tfrac{1}{2}\rho p), 16n(\rho - \tfrac{1}{2}), 16(2\rho n - \tfrac{3}{2}n + \tfrac{1}{2}\rho p)) \\
+ \; & (16)^3[\rho^3(6n^3 + n^m) + \rho^2(-21n^3 - 2n^2m) + \rho(24n^3 + n^2m) - 9n^3 \\
+ \; & \mathcal{E}(16n(\rho-1), 16n(\rho-1), 16n(\rho-1)) \\
= \; & (16)^3[\rho^3(28.66n^3 + 14n^2p + 2.75np^2 + .166p^3 + n^2m) \\
- \; & \rho^2(69n^3 + 19.5n^2p + 1.875np^2 + 2n^2m) \\
+ \; & \rho(46.75n^3 + 6.75n^2p + n^2m) \\
- \; & 16.875n^3
\end{aligned} \tag{6.124}$$

Finally if pseudo-inverses are not required in the computation, then then the amount of off-line computation to compute $\hat{X}[\rho|\rho]$ from $\hat{X}[\rho-1|\rho-1]$ is given by

$$\begin{aligned}
& \mathcal{I}(16(2\rho n - \tfrac{3}{2}n + \tfrac{1}{2}\rho p), 16n(\rho - \tfrac{1}{2})) \\
+ \; & (16)^3[\rho^3(6n^3 + n^m) + \rho^2(-21n^3 - 2n^2m) + \rho(24n^3 + n^2m) - 9n^3 \\
= \; & (16)^3[\rho^3(19.33n^3 + 8n^2p + 1.5np^2 + .083p^3 + n^2m) \\
- \; & \rho^2(49n^3 + 11n^2p + np^2 + 2n^2m) + \rho(43.5n^3 + 3.75n^2p + n^2m) - 13.5n^3]
\end{aligned} \tag{6.125}$$

On-line

One time step

The amount of on-line computation is given by

$$(16)^2[\rho^2(2n^2 + np) + \rho(-4n^2 - .5np) + 1.5n^2] \tag{6.126}$$

The on-line computation does not change if the state is estimable, or if pseudoinverses are unnecessary in the off-line computations.

Off-line computations

Summed over the region

To determine the computation needed to provide filtering from the center to the boundary of a region of radius $\rho = r$, we sum these terms from $\rho = 2$ to $\rho = r$ and

add a separate term for computing the initial estimate $\hat{X}[1|1]$. Using the identities

$$\begin{aligned} \textstyle\sum_{\rho=2}^{\rho=r} \rho^3 &= .25r^4 + .5r^3 + .25r^2 - 1 \\ \textstyle\sum_{\rho=2}^{\rho=r} \rho^2 &= .33r^3 + .5r^2 + .166r - 1 \\ \textstyle\sum_{\rho=2}^{\rho=r} \rho &= .5r^2 + .5r - 1 \end{aligned} \tag{6.127}$$

the operations may be counted by substituting ρ^j with the corresponding sum from 2 to r in equations (6.123)-(6.126). The initial computation required to compute $\hat{X}[1|1]$ is not a function of ρ in addition if we were to compute the optimal number of processors by taking the derivative, this term would not be essential. We will therefore just include only the summation in the following computations. For the case where the state is not estimable, the number of computations is given by

$$\begin{aligned} &\textstyle\sum_{\rho=2}^{\rho=r} \; [\mathcal{M}(16(2\rho n - \frac{3}{2}n + \frac{1}{2}\rho p), 16n(\rho - \frac{1}{2}), 16(2\rho n - \frac{3}{2}n + \frac{1}{2}\rho p)) \\ &\qquad +\mathcal{E}(16n(\rho-1), 16n(\rho-1), 16n(\rho-1)) \\ &\qquad +(16)^3\rho^3(16n^3 + n^2p + 2n^2m) \\ &\qquad +(16)^3\rho^2(-45n^3 - n^2p - 4n^2m) \\ &\qquad +(16)^3\rho(43.5n^3 + .25n^2p + 2n^2m) \\ &\qquad -(16)^3 14.5n^3] \\ = \;& (16)^3[(.25r^4 + .5r^3 + .25r^2 - 1)(70.33n^3 + 27n^2p + 5.25np^2 + .33p^3 + 2n^2m) \\ - \;& (.33r^3 + .5r^2 + .166r - 1)(146n^3 + 37.5n^2p + 3.625np^2 + 4n^2m) \\ + \;& (.5r^2 + .5r - 1)(118.75n^3 + 13n^2p + 2n^2m) \\ - \;& (r-1)42.125n^3] \end{aligned} \tag{6.128}$$

If the state is estimable then the amount of off-line computation needed to compute

filtered estimates in an entire region of radius k is given by

$$\begin{aligned}
&\textstyle\sum_{\rho=2}^{\rho=r} \; [\mathcal{E}(16(2\rho n - \frac{3}{2}n + \frac{1}{2}\rho p), 16n(\rho - \frac{1}{2}), 16(2\rho n - \frac{3}{2}n + \frac{1}{2}\rho p)) \\
&\qquad +(16)^3[\rho^3(6n^3 + n^m) + \rho^2(-21n^3 - 2n^2m) + \rho(24n^3 + n^2m) - 9n^3 \\
&\qquad +\mathcal{E}(16n(\rho - 1), 16n(\rho - 1), 16n(\rho - 1))] \\
= \;& (16)^3[(.25r^4 + .5r^3 + .25r^2 - 1)(28.66n^3 + 14n^2p + 2.75np^2 + .166p^3 + n^2m) \\
- \;& (.33r^3 + .5r^2 + .166r - 1)(+69n^3 + 19.5n^2p + 1.875np^2 + 2n^2m) \\
+ \;& (.5r^2 + .5r - 1)(46.75n^3 + 6.75n^2p + n^2m) \\
- \;& (r - 1)16.875n^3]
\end{aligned} \tag{6.129}$$

Finally if pseudo-inverses are not required in the computation, then the amount of off-line computation required to perform filtering within a radius of r is given by

$$\begin{aligned}
&\textstyle\sum_{\rho=2}^{\rho=r} \; [\mathcal{I}(16(2\rho n - \frac{3}{2}n + \frac{1}{2}\rho p), 16n(\rho - \frac{1}{2})) \\
&\quad + \; (16)^3[\rho^3(6n^3 + n^m) + \rho^2(-21n^3 - 2n^2m) + \rho(24n^3 + n^2m) - 9n^3] \\
= \;& (16)^3[(.25r^4 + .5r^3 + .25r^2 - 1)(19.33n^3 + 8n^2p + 1.5np^2 + .083p^3 + n^2m) \\
- \;& (.33r^3 + .5r^2 + .166r - 1)(+49n^3 + 11n^2p + np^2 + 2n^2m) \\
+ \;& (.5r^2 + .5r - 1)(43.5n^3 + 3.75n^2p + n^2m) \\
- \;& (r - 1)13.5n^3]
\end{aligned} \tag{6.130}$$

On-line

Summed over the region

The amount of on-line computation is given by

$$\begin{aligned}
&(16)^2[(.33r^3 + .5r^2 + .166r - 1)(2n^2 + np) \\
&-(.5r^2 + .5r - 1)(+4n^2 + .5np) + (r - 1)1.5n^2]
\end{aligned} \tag{6.131}$$

Note how the complexity varies with the parameters n and r. The computational complexity of the off-line computations varies with n^3, and since the dimension of the state is proportional to ρ, the complexity is proportional to r^4 where r is the radius of the region being filtered.

Step 3

In step three we compute the number of computation necessary to implement the Rauch-Tung-Striebel equations in Section 4.3. First we will consider the computation for one time step, then we will consider the sum of these operations over the entire region of radius $\rho = r$. If we assume that the state is not causally estimable, we need not compute the projection matrix in (4.115), and form products with this projection matrix and the quantities in (4.112)- (4.114). In addition (4.118) will not have to be computed. The amount of off-line computation required to compute $\hat{X}^s(\rho)$ from $\hat{X}^s(\rho+1)$ in the backward sweep of the Rauch-Tung-Striebel algorithm is given by

$$\begin{aligned} & \mathcal{M}(16\rho n, 32n(2\rho - \tfrac{1}{2}), 16\rho n) \\ + \; & \mathcal{E}(16n(\rho-1), 16n(\rho - \tfrac{1}{2}), 32n\rho) \\ + \; & (16)^3[\rho^3(27n^3 + 3n^2m) \\ + \; & \rho^2(-12.5n^3) + \rho(1.5n^3) \\ + \; & \rho(43.5n^3 + .25n^2p + 2n^2m) - 14.5n^3] \\ = \; & (16)^3[\rho^3(115.66n^3 + 3n^2m) + \rho^2(-65n^3) + \rho(13.5n^3) \end{aligned} \tag{6.132}$$

If the state is causally estimable then the amount of off-line computation to compute $\hat{X}^s(\rho)$ from $\hat{X}^s(\rho+1)$ is reduced because the quantities in (4.112)- (4.114) are of reduced dimension. This follows from $\overline{P}_{zf[k+1|k]} = 0$. In addition the pseudo-inverse in (4.117) will not have to be computed. Since the state is estimable, $\Lambda_k = I$. As a result $L_{rts} = I$. The smoothed covariance is given by

$$Cov(\tilde{x}^s(k)) = \Xi_k Cov(\tilde{x}^s(k+1))\Xi_k^T + Cov(\nu(k)) \tag{6.133}$$

The amount of computation for one time step is given by

$$\begin{aligned} & \mathcal{E}(16\rho n, 16n(\rho - \tfrac{1}{2}), 16\rho n) \\ + \; & (16)^3[\rho^3(12n^3) + \rho^2(-7n^3) \\ & = (16)^3[\rho^3(19.66n^3) + \rho^2(-8.5n^3) \end{aligned} \tag{6.134}$$

Finally if pseudo-inverses are not required in the computation, then then the amount

of off-line computation to compute $\hat{X}^s(\rho)$ from $\hat{X}^s(\rho+1)$ is given by

$$\begin{aligned} & \mathcal{I}(16(\rho n), 16n(\rho - \tfrac{1}{2})) \\ + \quad & (16)^3[\rho^3(12n^3) + \rho^2(-7n^3) \\ = \quad & (16)^3[\rho^3(15.33n^3) + \rho^2(-8n^3) \end{aligned} \tag{6.135}$$

On-line

One time step

The amount of on-line computation is given by

$$(16)^2[\rho^2(4n^2) + \rho(-2n^2)] \tag{6.136}$$

The on-line computation does not change if the state is estimable, or if pseudoinverses are unnecessary in the off-line computations.

Off-line computations

Summed over the entire region

To determine the computation needed to provide filtering for from the center to the boundary of a region of radius $\rho = r$, we sum these terms from $\rho = 1$ to $\rho = r-1$. Using the identities

$$\begin{aligned} \textstyle\sum_{\rho=1}^{\rho=r-1} \rho^3 &= .25r^4 - .5r^3 + .25r^2 \\ \textstyle\sum_{\rho=1}^{\rho=r-1} \rho^2 &= .33r^3 - .5r^2 + .166r \\ \textstyle\sum_{\rho=1}^{\rho=r-1} \rho &= .5r^2 - .5r \end{aligned} \tag{6.137}$$

Note that the limits of summation are different that those for the filtering step. this is because we may estimate $X(1)$ given the smoothed estimate at $X(2)$ while in the filtering step, there is no $X(0)$ defined to aid in estimating $X(1)$. We perform the summation by substituting ρ^j with the corresponding sum from $\rho = 1$ to $\rho = r-1$. For the case where the state is not causally estimable, the number of computations

is given by

$$\begin{aligned}
& \mathcal{M}(16\rho n, 32n(2\rho - \tfrac{1}{2}), 16\rho n) \\
+ \; & \mathcal{E}(16n(\rho - 1), 16n(\rho - \tfrac{1}{2}), 32n\rho) \\
+ \; & (16)^3[\rho^3(27n^3 + 3n^2m) \\
+ \; & \rho^2(-12.5n^3) + \rho(1.5n^3) \\
+ \; & \rho(43.5n^3 + .25n^2p + 2n^2m) - 14.5n^3] \\
= \; & (16)^3[(.25r^4 - .5r^3 + .25r^2)(115.66n^3 + 3n^2m) \\
+ \; & (.33r^3 - .5r^2 + .166r)(-65n^3) \\
+ \; & \rho(13.5n^3)
\end{aligned} \tag{6.138}$$

If the state is causally estimable then the amount of off-line computation to compute $\hat{X}^s(\rho)$ from $\hat{X}^s(\rho + 1)$ is given by

$$\begin{aligned}
& \mathcal{E}(16\rho n, 16n(\rho - \tfrac{1}{2}), 16\rho n) \\
+ \; & (16)^3[\rho^3(12n^3) + \rho^2(-7n^3) \\
+ \; & (16)^3[(.25r^4 - .5r^3 + .25r^2)(19.66n^3) \\
+ \; & (.33r^3 - .5r^2 + .166r)(-8.5n^3)
\end{aligned} \tag{6.139}$$

Finally if pseudo-inverses are not required in the computation, then the amount of off-line computation to compute $\hat{X}^s(\rho)$ from $\hat{X}^s(\rho + 1)$ is given by

$$\begin{aligned}
& \mathcal{I}(16(\rho n), 16n(\rho - \tfrac{1}{2})) + (16)^3[\rho^3(12n^3) + \rho^2(-7n^3) \\
= \; & (16)^3[(.25r^4 - .5r^3 + .25r^2)(15.33n^3) + (.33r^3 - .5r^2 + .166r)(-8n^3)
\end{aligned} \tag{6.140}$$

On-line

The amount of on-line computation is given by

$$(16)^2[(.33r^3 - .5r^2 + .166r)(4n^2) + (.5r^2 - .5r)(-2n^2)] \tag{6.141}$$

Step 2

Part 1

The interprocessor communication requires two types of computations. One is combining two square regions to form a rectangular region. The second is to combine

two rectangular regions to form a square region. We will handle these two cases separately. The computations deal with the measurements depicted in Figures 6-11, and 6-9.

The key issue again is how the dimension of the state varies from step to step. We will use the integer variable t to help us indicate the number of levels of interprocessor communication which take place. At the start of the interprocessor communication, $t = 0$ and it is incremented every two interprocessor communication levels. Thus once four squares are combined into a larger square, t is incremented. In the following the parameter r is fixed, representing the radius of the smallest square region. Earlier it was stated that the dimension of the state increases roughly by a factor of $\sqrt{2}$ at each step. If we define r as the radius of the smallest square subregion then the dimension of a state defined on a square boundary during the interprocessor communication step is given by

$$X(2t) \in \Re^{((16r+8)\times 2^t - 16)n} \tag{6.142}$$

where $2t$ here represents the number of interprocessor communications. Only for even values of $2t$ is the boundary a square. The dimension of the odd indexed states is given by

$$X(2t+1) \in \Re^{((24r+12)\times 2^t - 16)n} \tag{6.143}$$

There are also $n[(4r+2)2^t - 4]$ dynamic constrains used to merge the estimates of two square states together. There are also $n[(8r+4)2^t - 4]$ dynamic constraints involved in combining the estimates of two rectangles into an estimate of the state around a square boundary. Computational economies result if we take advantage of the fact that the parameters N, S, E, W, B, and C , do not vary with position. In these computations however we will assume that independent processors compute different and independent inverse estimation error covariances.

Off-line

One time step

An example of the measurements used to determine the number of computations is given by equation (6.105) for $t = 0$ and is shown in Figure 6-9. The amount of

computation to compute an estimate for the boundary of a rectangular region from two square regions is given by

$$\begin{aligned} & \mathcal{M}(((68r+34)2^t-72)n,((32r+16)\times 2^t-32)n,((68r+34)2^t-80)n)) \\ + \; & \mathcal{E}(((24r+16)2^t-16)n,((24r+16)2^t-16)n,((24r+16)2^t-16)n) \\ + \; & 2((32r+16)2^t-32)^2((36r+18)2^t-40))n^3 \end{aligned} \tag{6.144}$$

Equation (6.111) provides an example of the equations required to combine two rectangles together to construct a square boundary. See also Figure 6-11. The amount of computation required to combine the estimates of the boundaries of two rectangular regions into the estimates of the square boundary which encloses the two subregions is given by

$$\begin{aligned} & \mathcal{M}(((104r+34)2^t-80)n,((48r+32)\times 2^t-32)n,((104r+34)2^t-80)n) \\ + \; & \mathcal{E}(((16r+8)2^{t+1}-16)n,((16r+8)2^{t+1}-16)n,((16r+8)2^{t+1}-16)n) \\ + \; & 2((48r+16)2^t-32)^2((56r+18)2^t-40))n^3 \end{aligned} \tag{6.145}$$

Here for all values greater than $t = 0$ and all $r > 1$ all expressions of the form $(ar+b)2^t+c$ can be approximated by $(ar+b)2^t$. We may therefore approximate (6.144) and (6.145) by

$$\begin{aligned} & \mathcal{M}(((68r+34)2^t-72)n,((32r+16)\times 2^t-32)n,((68r+34)2^t-80)n)) \\ + \; & \mathcal{E}(((24r+16)2^t-16)n,((24r+16)2^t-16)n,((24r+16)2^t-16)n) \\ + \; & 2((32r+16)2^t-32)^2((36r+18)2^t-40))n^3 \\ \approx \; & (16)^3(50.41)2^{3t}n^3(2r+1)^3+(16)^3(.96)2^{3t}n^3(3r+2)^3 \end{aligned} \tag{6.146}$$

and

$$
\begin{aligned}
& \mathcal{M}(((104r+34)2^t-80)n,((48r+32)\times 2^t-32)n,((104r+34)2^t-80)n) \\
+\ & \mathcal{E}(((16r+8)2^{t+1}-16)n,((16r+8)2^{t+1}-16)n,((16r+8)2^{t+1}-16)n) \\
+\ & 2((48r+16)2^t-32)^2((56r+18)2^t-40))n^3 \\
\approx\ & 2^3n^32^{3t}(52r+17)^2(172r+97)
\end{aligned}
\tag{6.147}
$$

respectively.

On-line

One time step The amount of on-line computation required to combine two square boundaries is given by

$$(16)^2 * 6 * 2^{2t}(r^2+r+.25) \tag{6.148}$$

The amount of on-line computation required to combine two square boundaries is given by

$$(16)^2 * 12 * 2^{2t}(r^2+r+.25) \tag{6.149}$$

Off-line

Summing over the entire region Since these equations vary with t which here represents the number of levels of interprocessor communication pairs performed, we need concern ourselves particularly with summing terms which grow exponentially. We therefore need to use the equality

$$\sum_{j=0}^{N} g^j = \frac{g^{N+1}-1}{g-1} \quad g \geq 1 \tag{6.150}$$

From this sum we can tell that the if g is sufficiently large then the total number of computation is approximately g^N. Since the square states alone varies by 2^t, and for one computation, the off-line computation varies as 2^{3t}, and the on-line computation varies as 2^{2t}, in this problem $g = 8$ for the off-line computation and $g = 4$ for online computation. We may conclude that the computation is dominated by the the last computation corresponding to $n = N$, and we could if necessary neglect all prior computations. We will not but it is important to realize that the size of the state

increases so fast that operations on prior states are negligible.

We can conclude that for 2T levels of interprocessor communications, the total number of floating point operations off-line required in computing the necessary gains for combining square boundaries together to form rectangular boundaries is given by

$$(16)^3[(50.41)(2r+1)^3+(.96)(3r+2)^3]n^3[\frac{2^{3(T+1)}-1}{7}] \tag{6.151}$$

and the number of off-line computations involved in combining rectangular regions into square regions is given by

$$2^3n^3(52r+17)^2(172r+97)[\frac{2^{3(T-1)}-1}{7}] \tag{6.152}$$

The total amount of computation involved is given by

$$\{(16)^3[(50.41)(2r+1)^3+(.96)(3r+2)^3]+2^3(52r+17)^2(172r+97)\}n^3[\frac{2^{3(T+1)}-1}{7}] \tag{6.153}$$

For the case that the state is estimable the total amount of computation is approximately given by

$$\begin{aligned}&\mathcal{E}(36n[2^T(r+\tfrac{1}{2})],24n[2^T(r+\tfrac{1}{2})],36n[2^T(r+\tfrac{1}{2})])\\&=3.33(36)^3 2^{3T}n^3(r+\tfrac{1}{2})^3\end{aligned} \tag{6.154}$$

for combining square regions

$$\begin{aligned}&\mathcal{E}(48n[2^T(r+\tfrac{1}{2})],16n[2^{T+1}(r+\tfrac{1}{2})],48n[2^T(r+\tfrac{1}{2})])\\&=5.833(48)^3 2^{3T}n^3(r+\tfrac{1}{2})^3\end{aligned} \tag{6.155}$$

for combining rectangular regions

For the case that pseudo-inverses are not required, the total amount of computation is given by

$$\begin{aligned}&\mathcal{I}(36n[2^T(r+\tfrac{1}{2})],24n[2^T(r+\tfrac{1}{2})])\\&=2(36)^3n^3 2^{3T}(r+\tfrac{1}{2})^3\end{aligned} \tag{6.156}$$

for combining square regions

$$\begin{aligned}&\mathcal{I}(48n[2^T(r+\tfrac{1}{2})], 16n[2^{T+1}(r+\tfrac{1}{2})])\\&= 2.166(48)^3 n^3 2^{3T}(r+\tfrac{1}{2})^3\end{aligned} \tag{6.157}$$

for combining rectangular regions

On-line The total amount of on-line computation required to combine two boundaries is given by

$$(16)^2 * 6 * \tfrac{2^{2(T+1)}-1}{3}(r^2+r+.25) \tag{6.158}$$

We may conclude this section by noting that if K^2 is the number of points in the entire region, and if we use L^2 processors to partition them then the radius of the smallest subregion will be given by $r = \frac{K-L}{2L}$. The number of interprocessor communications will be given by $2T = \log_2 L^2 = 2\log_2 L$. The total computation time for the case where the state is not estimable is given by

$$\begin{aligned}\tau = \;& \{(16)^3[(50.41)(2\tfrac{K-L}{2L}+1)^3 + (.96)(3\tfrac{K-L}{2L}+2)^3] + 2^3(52\tfrac{K-L}{2L}+17)^2(172\tfrac{K-L}{2L}+97)\}n^3[\tfrac{8L^3-1}{7}]\\
+ \;& (16)^3[n^3(15.16\{\tfrac{K-L}{2L}\}^4 - 25.33) + n^2p(6.5\{\tfrac{K-L}{2L}\}^4\\
+ \;& 25.17\{\tfrac{K-L}{2L}\}^3 + 37.63\{\tfrac{K-L}{2L}\}^2 + 12.46\tfrac{K-L}{2L})\\
+ \;& np^2(1.31\{\tfrac{K-L}{2L}\}^3 + 1.41\{\tfrac{K-L}{2L}\}^3 - .5\{\tfrac{K-L}{2L}\}^2 - .61\tfrac{K-L}{2L})]\end{aligned} \tag{6.159}$$

In this term the logarithms and the exponents have canceled leaving an algorithm whose complexity increases in polynomial time. The local processing in this algorithm grows with the size of the region and much more computations are performed at larger radii than for smaller. For a square region of large radius r the ratio of computation performed radially outward, to the amount of computation required if a marching method is used is equal to 128 for the off-line computations and is equal to approximately 42.33 for the on line computations. One thing which is gained by filtering inward and outward is that a general boundary condition is feasible, where as a general boundary condition was not supported in the marching methods used by Adams[1], for example. We will find that certain suboptimal tecniques can bring this ratio to zero asymptotically for the on-line processing. Specifically to compute the

optimal estimate on a region of radius r requires $\mathcal{O}(r^3)$ flops of on-line computation.. We will present suboptimal tequniques which will compute the estimate in a region with $\mathcal{O}(r^2)$ flops, and also $\mathcal{O}(r \log r)$ flops.

6.4 Suboptimal Smoothing

6.4.1 Suboptimal interprocessor communication

The computation in the algorithm as described in this chapter involves combining larger and larger boundaries together. The size of the boundary increase by approximately a factor of $\sqrt{2}$ at each stage in the interprocessor communication. Although the complexity of the operations involved increases dramatically at each step, the number of computations required is greatly reduced. In fact the bottleneck is the fact that the computation involving the larger sub-boundaries including the boundary of the entire region.

As the size of the state grows the complexity of the interprocessor communication grows. The size of the covariance of the state grows as the square of the size of the state. If the state is not estimable, then a projection matrix must be carried along whose size is equal to the size of the covariance of the state.

For the purposes of the discussion in this section we assume that the process at the boundary of the local subregions is estimable based on data in the local subregions, and therefore avoid issues associated with the projection matrix. Consider equation (6.105) given by

$$\begin{bmatrix} 0 \\ 0 \\ \hat{X}(s,t) \\ \hat{X}(s+1,t) \end{bmatrix} = \begin{bmatrix} -H_e & \mathcal{E} \\ \mathcal{W} & -H_w \\ I & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} X(s,t) \\ X(s+1,t) \end{bmatrix} + \begin{bmatrix} B_e U_e(s,t) \\ B_w U_w(s,t) \\ \tilde{X}(s,t) \\ \tilde{X}(s+1,t) \end{bmatrix} \tag{6.160}$$

This equation is of the form

$$y = Hx + v \tag{6.161}$$

where H is given by

$$H = \begin{bmatrix} -H_e & \mathcal{E} \\ \mathcal{W} & -H_w \\ I & 0 \\ 0 & I \end{bmatrix} \tag{6.162}$$

y is given by

$$y = \begin{bmatrix} 0 \\ 0 \\ \hat{X}(s,t) \\ \hat{X}(s+1,t) \end{bmatrix} \tag{6.163}$$

x is given by

$$x = \begin{bmatrix} X(s,t) \\ X(s+1,t) \end{bmatrix} \tag{6.164}$$

and v is given by

$$v = \begin{bmatrix} B_e U_e(s,t) \\ B_w U_w(s,t) \\ \tilde{X}(s,t) \\ \tilde{X}(s+1,t) \end{bmatrix} \tag{6.165}$$

Since the solution in (6.107), and (6.108) is constructed from solving

$$\begin{bmatrix} R & H \\ H^T & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \hat{x} \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix} \tag{6.166}$$

or equivalently since R is invertible from solving

$$H^T R^{-1} H \hat{x} = H^T R^{-1} y \tag{6.167}$$

where $H^T R^{-1} H$ is the inverse estimation covariance of x, it may be desirable to exploit sparsity in $H^T R^{-1} H$ to solve for x, or to exploit sparsity in R or R^{-1} to solve for x.

The approach we will take here is to model the process at the boundary after the

local processing step, as a one dimensional process going around the boundary.

If we partition R as follows

$$R = \begin{bmatrix} B_e B_e^T & 0 & 0 & 0 \\ 0 & B_w B_w^T & 0 & 0 \\ 0 & 0 & \Sigma_{\hat{X}(s,t)} & 0 \\ 0 & 0 & 0 & \Sigma_{\hat{X}(s+1,t)} \end{bmatrix} \tag{6.168}$$

where the diagonal entries of the matrix in (6.168) correspond to the covariances of the elements of v. We are therefore proposing exploiting structure in $\Sigma_{\hat{X}(s,t)}$, and in $\Sigma_{\hat{X}(s+1,t)}$ or their corresponding inverses to compute (6.167).

Specifically we expect that we should either be be able to neglect correlations between elements in the state outside of a neighborhood of some characteristic radius or neglect interactions between elements outside of a neighborhood of some characteristic radius when computing the Gibbs distribution for the equivalent Markov Random Field. The first method roughly amounts to modeling the process as a moving average, and the latter models the system as an auto-regressive system. Outside of this characteristic radius we will assume that the elements of the estimation error covariance, or inverse estimation error covariance are precisely zero. We will choose an example where the largest singular value of the error in approximating the estimation error covariance is small compared to the smallest singular value of the estimation error covariance. If we are using the inverse estimation error covariance to model the process then we wish that the largest singular value of the error in approximating the inverse estimation error covariance is small compared to the smallest singular value of the inverse estimation error covariance. In addition since the number of interprocessor communication steps is small, we can expect to accumulate only a limited amount of modeling errors while gaining a computational advantage in combining the estimates of neighboring boundaries.

In the following example we are considering a NNM of the form (6.1) where

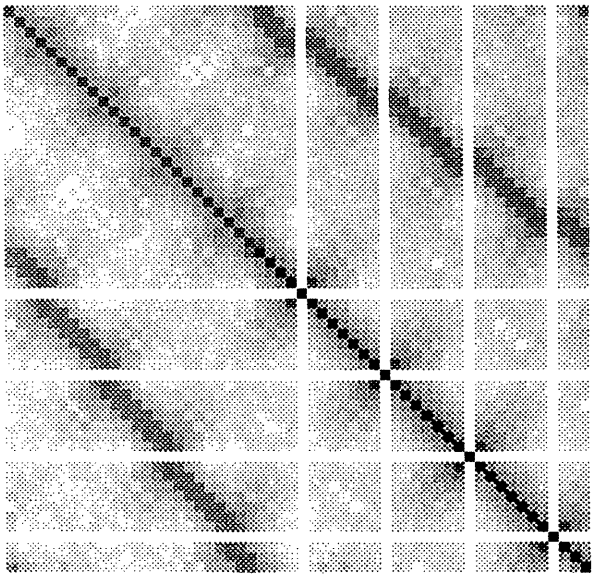


Figure 6-12: grayscale image of 56×56 estimation error covariance of state at $\rho = 4$

$N = S = E = W = B = 1$ with observations given by

$$y(i,j) = x(i,j) + v(i,j) \tag{6.169}$$

where $v(i,j)$ has unit covariance. The system which we are filtering is given by (6.5) where $\mu = \infty$. Here we aim to find for this example when and how to model the process around the boundary of a two dimensional region as a one-dimensional process. In Figures 6-12 and 6-13 we provide the estimation error covariance for $X(\rho)$ for $\rho = 4$ based on data from $\rho = 0$ to $\rho = 4$. The matrix is banded but the bands are relatively large compared to the size of the matrix and suggests few opportunities for meaningful approximation. The inverse estimation error covariance for the same vector is provided in Figures 6-14 and 6-15. This matrix is full and presents few opportunities for meaningful approximation. For $\rho = 11$ we are able to take advantage of sparsity in the estimation error covariance and inverse estimation error covariance matrix. Figure 6-16 and 6-17 show the estimation error covariance for the state $X(11)$ while Figure 6-18 and 6-19 show the inverse estimation error covariance for the same. Both matrices demonstrate a five band structure. In between these bands the elements are small. It is these small elements which we will approximate

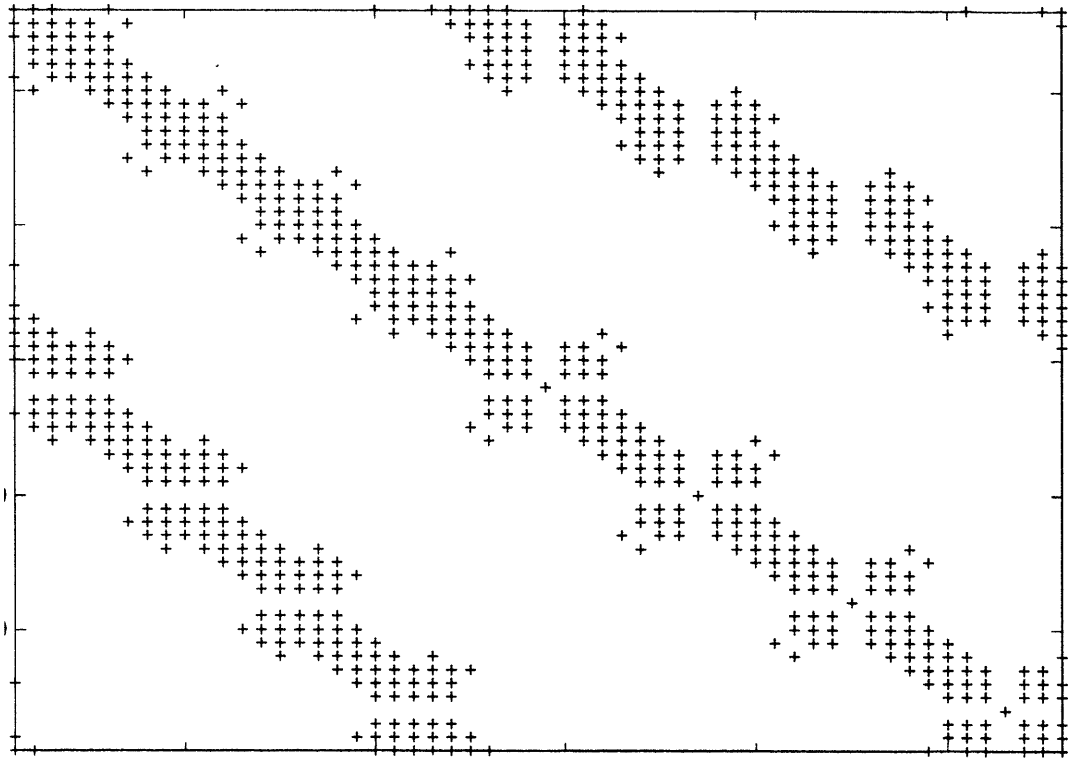


Figure 6-13: 56×56 estimation error covariance of state at $\rho = 4$ showing the entries greater than .01

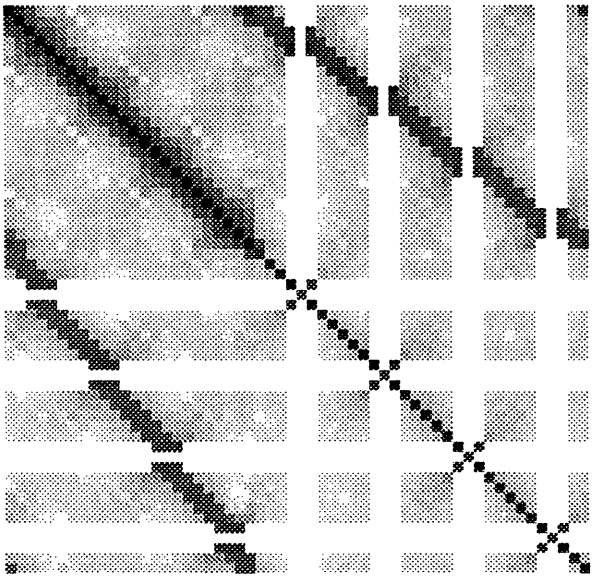


Figure 6-14: grayscale image of 56×56 inverse estimation error covariance of state at $\rho = 4$

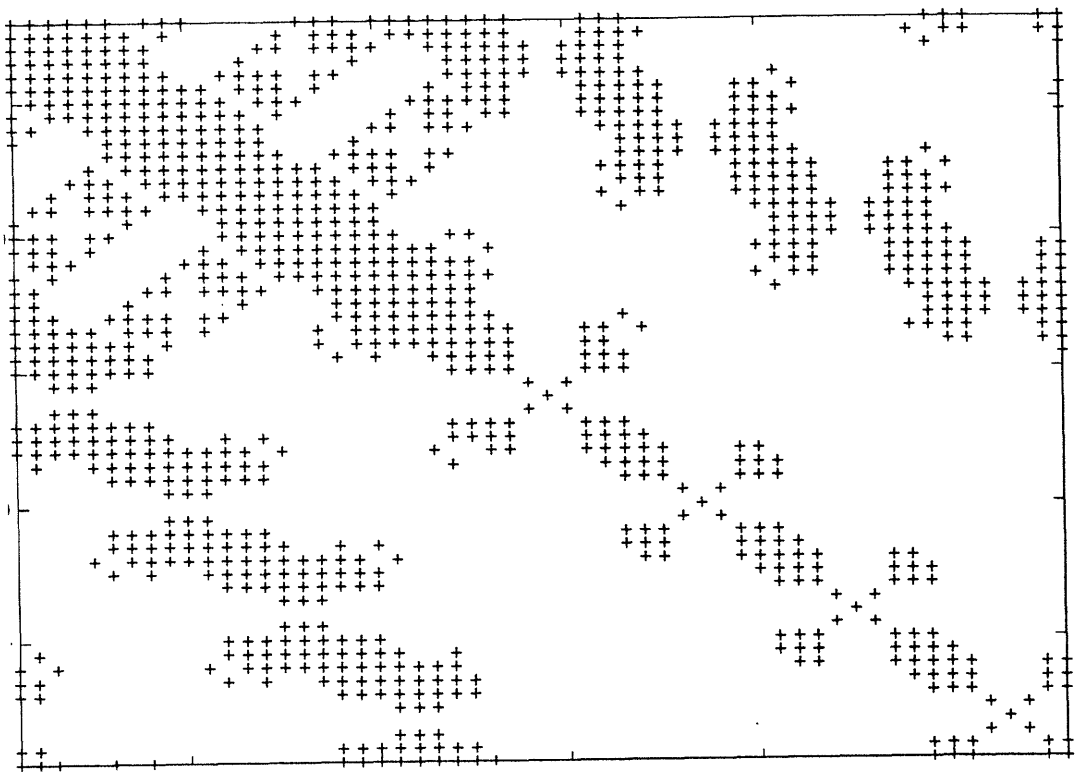


Figure 6-15: 56×56 inverse estimation error covariance of state at $\rho = 4$ showing the entries greater than .01

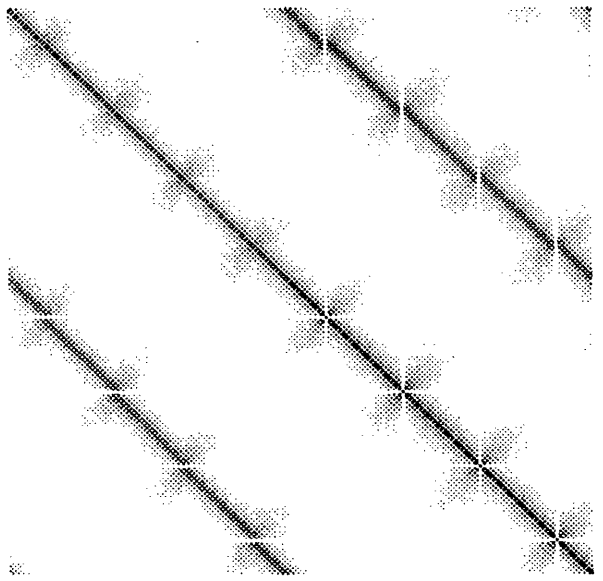


Figure 6-16: grayscale image of 168×168 estimation error covariance of state at $\rho = 11$

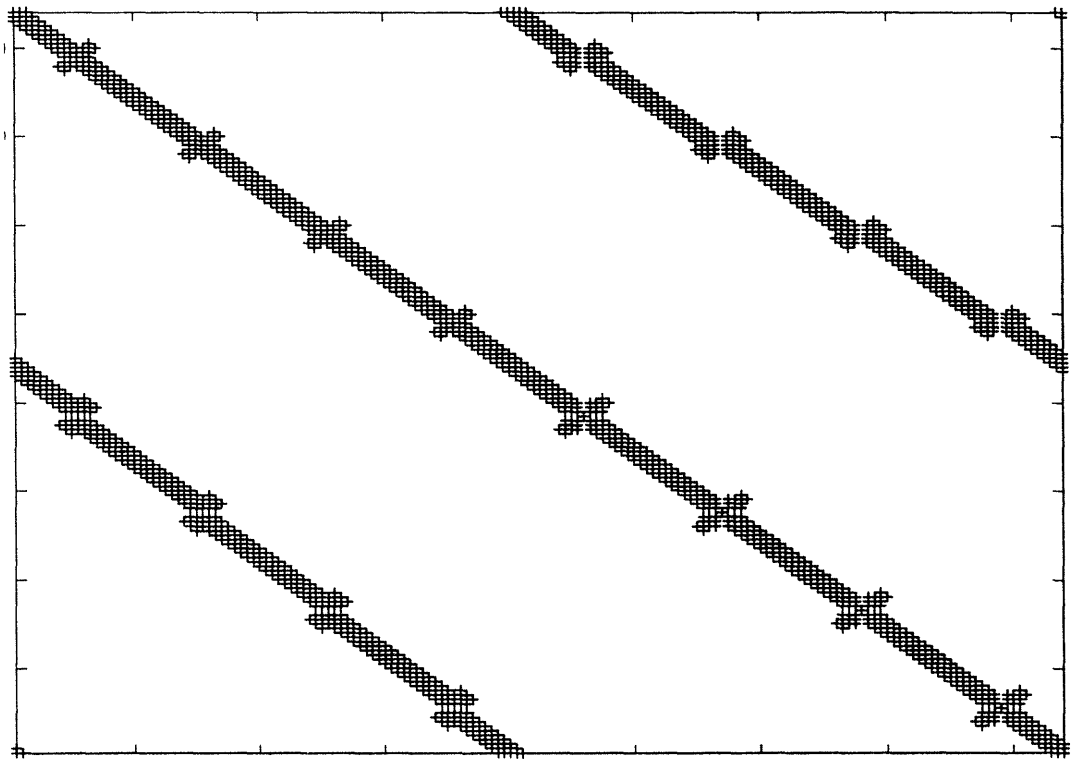


Figure 6-17: 168×168 estimation error covariance of state at $\rho = 11$ showing the entries greater than .01

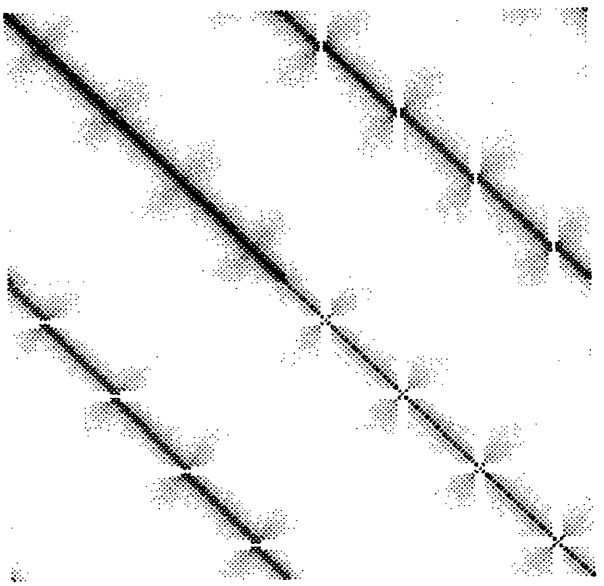


Figure 6-18: grayscale image of 168×168 inverse estimation error covariance of state at $\rho = 11$

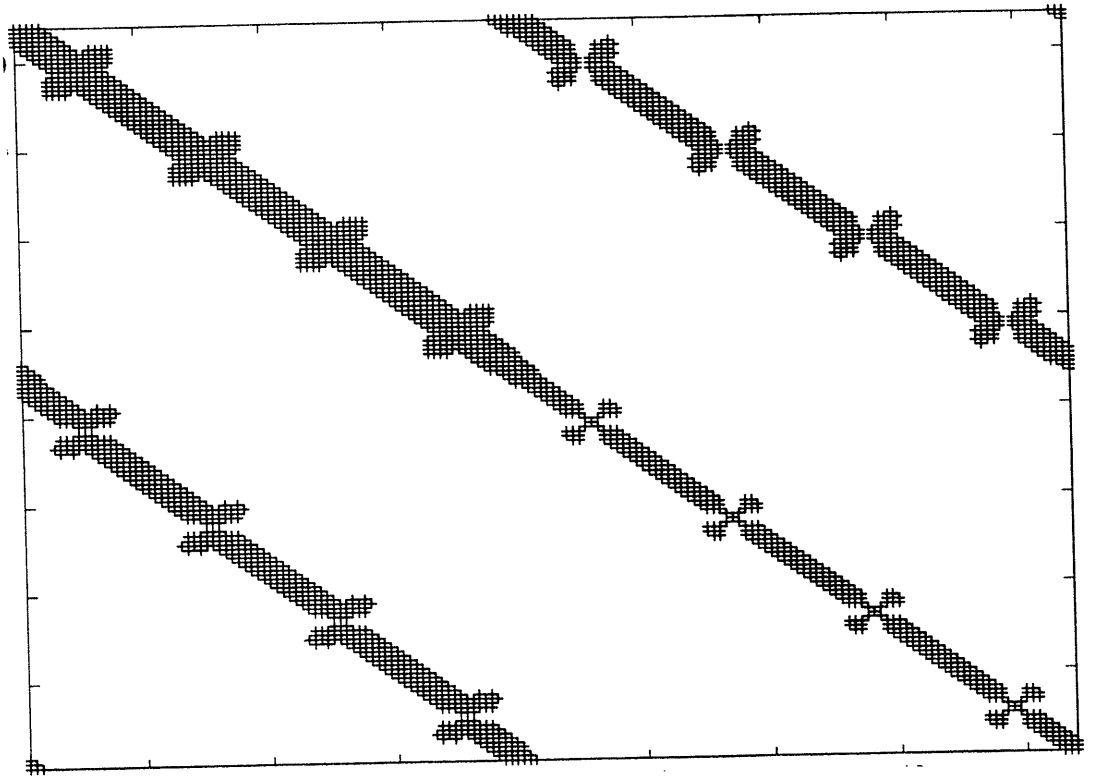


Figure 6-19: 168×168 inverse estimation error covariance of state at $\rho = 11$ showing the entries greater than .01

by zero. The bands in the estimation error covariance are narrower than those of the inverse estimation error covariance and suggest that we may take advantage of a moving average representation of the process. Here we examine approximating the inverse estimation error covariance in anticipation of using further the tools involving STPBVDS developed in earlier chapters.

Toward this end, we chose to set all values whose magnitude is less than .01 to 0. To support this approximation we note that for all $\rho \geq 1$ the smallest singular value of the inverse estimation error covariance $\Sigma_{X(\rho)}$ is equal to 1.

$$\underline{\sigma}(\Sigma_{X(\rho)}) = 1 \tag{6.170}$$

Let us denote the difference between the actual inverse estimation error covariance and the approximated inverse estimation error covariance by $\delta\Sigma_{X(\rho)}$ where $\Sigma_{X(\rho)} + \delta\Sigma_{X(\rho)}$ is the approximated inverse estimation error covariance. Figure 6-20 shows that for

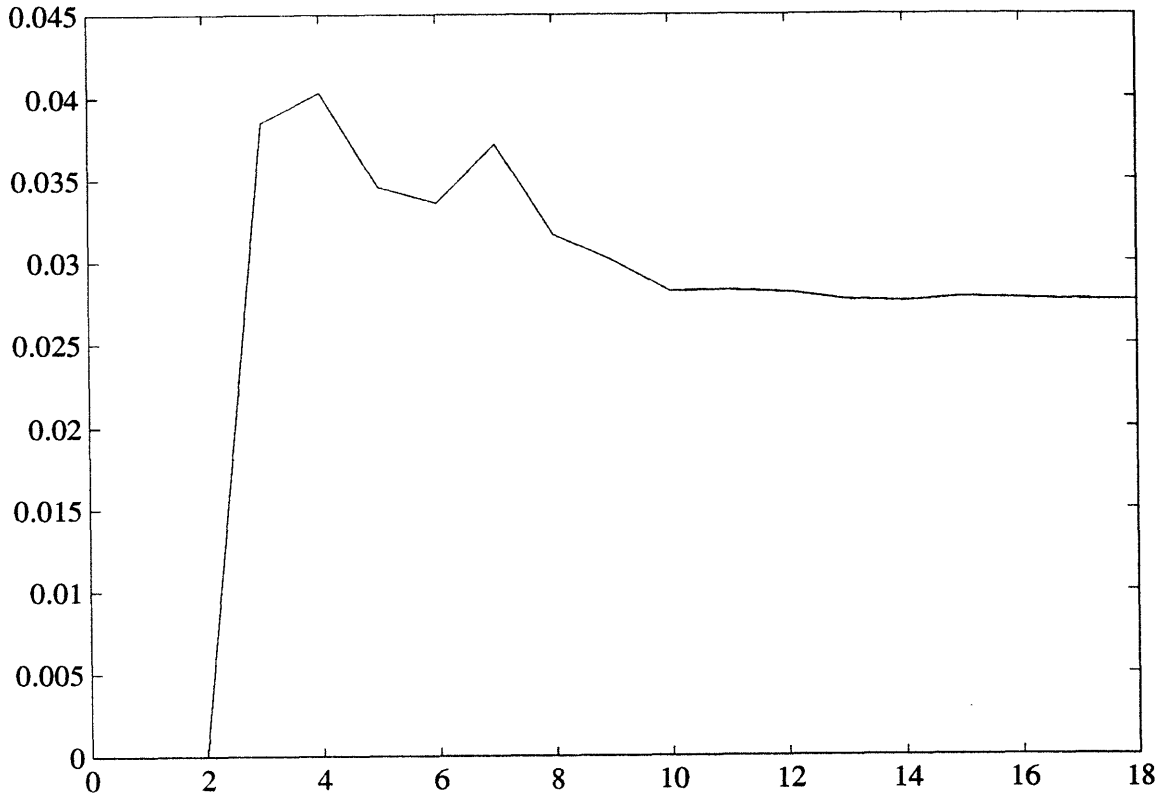


Figure 6-20: Maximum singular value plot for the difference between inverse estimation error covariance for the state at different radii and the approximation to that inverse estimation error covariance based on setting values smaller than .01 to 0. Plot starts at $\rho = 2$.

$\rho \geq 11$ that

$$\overline{\sigma}(\delta\Sigma_{X(\rho)}) \leq .03 \tag{6.171}$$

where

$$\max_{i,j}(\delta\Sigma_{X(\rho)})_{ij} \leq .01 \tag{6.172}$$

With this approximation the computation for the union of two boundaries becomes greatly simplified. We will examine efficient methods to compute the estimate and estimation error covariance $H^T(\Sigma+\delta\Sigma)H$ where $\Sigma = R^{-1}$ is the inverse error covariance of the observations in (6.168) and where $\Sigma + \delta\Sigma = (R + \delta R)^{-1}$ is the approximation to the corresponding inverse estimation error covariance.

We will now outline how to combine the boundaries of two neighboring regions.

First we will reorder the elements of the vector $X(\rho)$ so that large elements of $\Sigma_{X(\rho)}$ are near the main diagonal. This amounts to choosing a different ordering

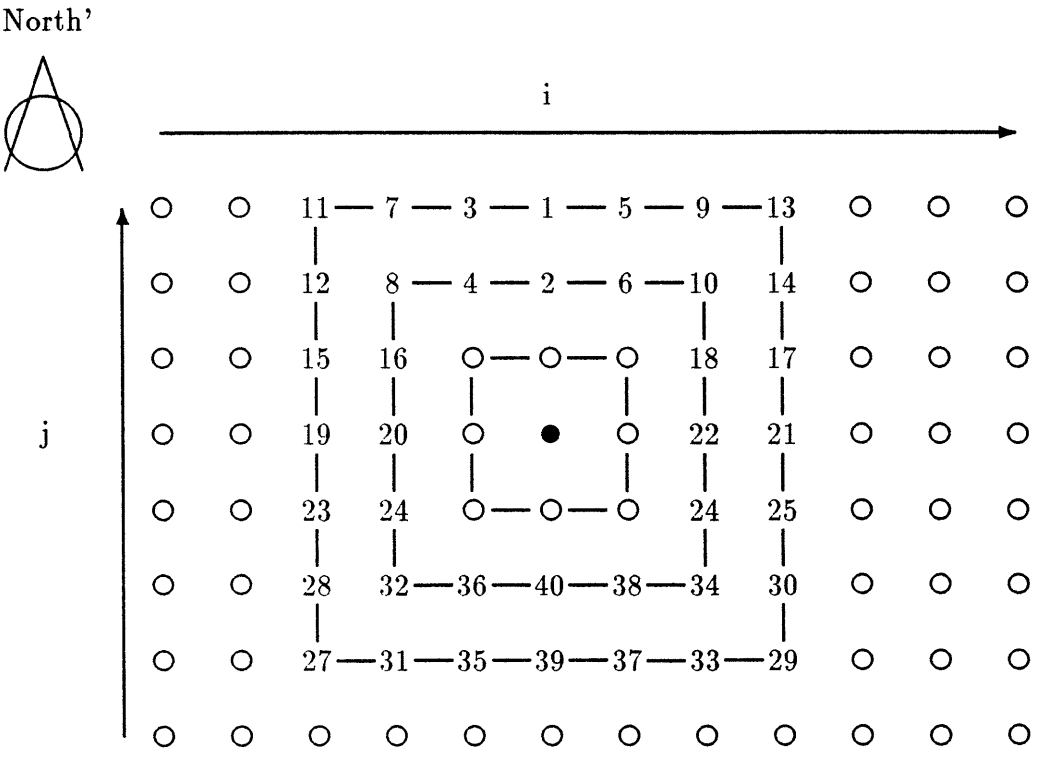


Figure 6-21: The integer function Θ as a function of i and j for $\rho = 3$. Here the elements are ordered so that we may perform our processing moving outward from the center of one leg and eventually terminate moving inward on the opposite leg of the square boundary.

function $\Theta(i, j, \rho)$. An example of the new ordering function is shown in Figure 6-21. Basically the elements are chosen such that we progress both to the left and to the right (outward) from the first element at the top in Figure 6-22 while accounting for elements at ρ, and $\rho - 1$ together. As a result elements which are near each other spatially remain near each other in the vector $X(\rho)$ while elements in the inverse estimation error covariance matrix decay away from the main diagonal. With this ordering only 43 diagonals to the left and to the right of the main diagonal in the inverse estimation error covariance matrices have elements with magnitudes greater than .01. The bandwidth of the matrix remains constant as the size of the inverse error covariance matrix grows with the radius of the region as $16(\rho + \frac{1}{2}) \times 16(\rho + \frac{1}{2})$. In order to model this as a (causal) STPBVDS, the approximation to the inverse estimation error covariance can be partitioned into 44×44 blocks, so that the matrix is block tridiagonal. Before we consider a recursive method of solving

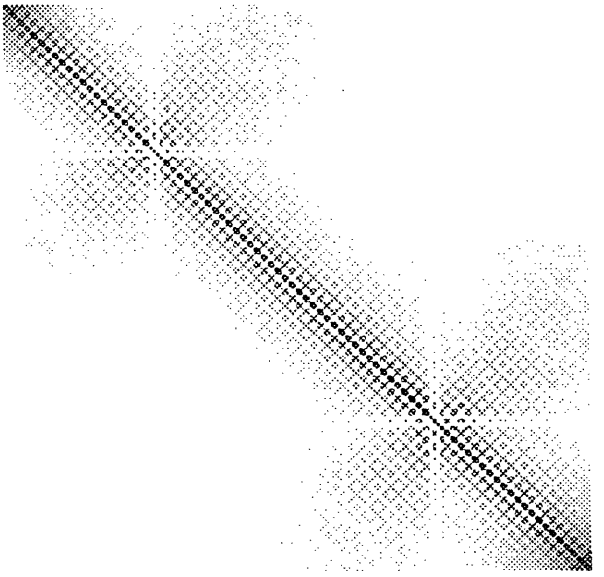


Figure 6-22: The inverse estimation error covariance at $\rho = 11$ after the elements have been reordered. Large elements of the matrix have been moved to locations near the main diagonal.

this suboptimal problem, we will note that there are many algorithms which operate on sparse matrices. If we were to re-order the elements of both boundary estimates we could again arrive at a inverse covariance matrix for the combined estimate which is block tridiagonal with a bandwidth of approximately 176 elements. Cyclic block reduction can be performed on this system[30][31][9], Gauss-Seidel or any of a number of other iterative methods can be used on this system[30],[31],[9]. In particular for this example if we were to perform cyclic block reduction to compute the estimate of the combined region, it would require

$$\mathcal{O}[(t + \log_2(16r - 8))(44)^2] \tag{6.173}$$

flops where the number 44 represents the size of the blocks in the inverse covariance. The scalar r represents the radius of the smallest subregion and $2t$ represents the number of levels of interprocessor communication. The computation time grows with the logarithm of the size of the state, and as the square of the bandwidth of the inverse estimation error covariance. Cyclic block reduction can be performed in parallel and is suitable for implementation on a hypercube, and thus has the advantage of easily

mapping onto the architecture on which the remainder of the algorithm is designed to run.

Gauss Seidel has very good performance with finite bandwidth diagonally dominant positive definite matrices. The amount of computation required to compute one iteration is approximately twice the number of nonzero elements in $H^T(\Sigma + \delta\Sigma)H$ in the expression $x' = [H^T(\Sigma + \delta\Sigma)H]^{-1}[H^T(\Sigma + \delta\Sigma)y$. Since the bandwidth of the system remains constant for larger and larger regions the complexity of the computation increases linearly with the size of the matrices. Furthermore when the matrices are diagonally dominant, the rate of convergence is independent of the size of the matrix. As a result the number of iterations required to compute the estimate is a also independent of the size of the matrix.

Returning to the issue of modeling the process of the boundary by a STPBVDS, we note the two salient features in Figure 6-22 are the bowtie regions which represent the corners of the square region and the fact that the matrix is nearly diagonal. There are only two bowties because in our ordering we reach two corners simultaneously. The bowties exist because at the corners more points fit into a neighborhood of a given size. The diagonal nature of the matrix suggests the opportunity to model the system as a STPBVDS.

The Cholesky factorization can be taken for the approximation to the inverse estimation error covariance yielding a block bidiagonal matrix, whose blocks are the parameters of the system which models the process around the boundary. If we use the variable ψ to count the diagonal blocks of the approximated inverse estimation error covariance matrix, then the system around the boundary has the following representation.

$$E_{\rho,\psi+1}X_\rho(\psi+1) = A_{\rho,\psi}X_\rho(\psi) + B_{\rho,\psi}U_\rho(\psi) \tag{6.174}$$

where $X_\rho(\psi) \in \Re^{44}$ is the state in the ψ^{th} partition of $X(\rho)$ If we account for the fact that the bowtie regions are significantly wider than the remaining regions we could in fact model the process as a STPBVDS with ψ varying dimension yielding greater computational efficiencies. We will not pursue modeling this system with variable

dimensional systems.

Computing the estimate of the common boundary to two neighboring regions is a matter of augmenting the state in one boundary with the state of the other boundary

$$\begin{bmatrix} E_{(s,t),\psi+1} & 0 \\ 0 & E_{(s+1,t),\psi+1} \end{bmatrix} \begin{bmatrix} X_{(s,t)}(\psi+1) \\ X_{(s+1,t)}(\psi+1) \end{bmatrix} = \begin{bmatrix} A_{(s,t),\psi} & 0 \\ 0 & A_{(s+1,t),\psi} \end{bmatrix} \begin{bmatrix} X_{(s,t)}(\psi) \\ X_{(s+1,t)}(\psi) \end{bmatrix} + \begin{bmatrix} B_{(s,t),\psi} & 0 \\ 0 & B_{(s+1,t),\psi} \end{bmatrix} \begin{bmatrix} U_{(s,t)}(\psi) \\ U_{(s+1,t)}(\psi) \end{bmatrix} \tag{6.175}$$

where we have replaced the index ρ with (s,t) since the squares have the same dimensions and we need to distinguish between the boundaries of the two regions. The remaining dynamic constraints which are shown in Figure 6-11 can be written in two forms. Note that the nearest neighbor model is a local model which constrains the $x(i,j)$ a distance of two apart. Since the state $X_{(s,t)}(\psi)$ in our example contains 44 of the $x(i,j)$, all near each other, most of the dynamic constrains will not connect the state for different values of ψ. Therefore the first form which accounts for most of the dynamic constraints can be written in the form

$$0 = H_{(s,t),\psi} X_{(s,t)}(\psi) + H_{(s+1,t),\psi} X_{(s+1,t)}(\psi) + V_{(\{s,s+1\},t)}(\psi) \tag{6.176}$$

In our example this will account for 36 dynamic constraints. The remaining dynamic constraints will link the small number of $x(i,j)$ which lie on the edge of the region corresponding to $X(\psi)$ and $X(\psi+1)$ together. These dynamic constraints can be written in the form

$$0 = \begin{bmatrix} \Lambda_{(s,t),e} & \Lambda_{(s+1,t),e} \end{bmatrix} \begin{bmatrix} X_{(s,t)}(\psi+1) \\ X_{(s+1,t)}(\psi+1) \end{bmatrix} + \begin{bmatrix} \Lambda_{(s,t),a} & \Lambda_{(s+1,t),a} \end{bmatrix} \begin{bmatrix} X_{(s,t)}(\psi) \\ X_{(s+1,t)}(\psi) \end{bmatrix} + W(\psi) \tag{6.177}$$

which accounts for 8 dynamic constraints. The result is that (6.174), (6.175), and (6.176) comprise a STPBVDS which can be filtered and smoothed via common means to compute the required estimates. However to further combine neighboring boundaries we need not compute the entire process, but arrive at a *model* for the

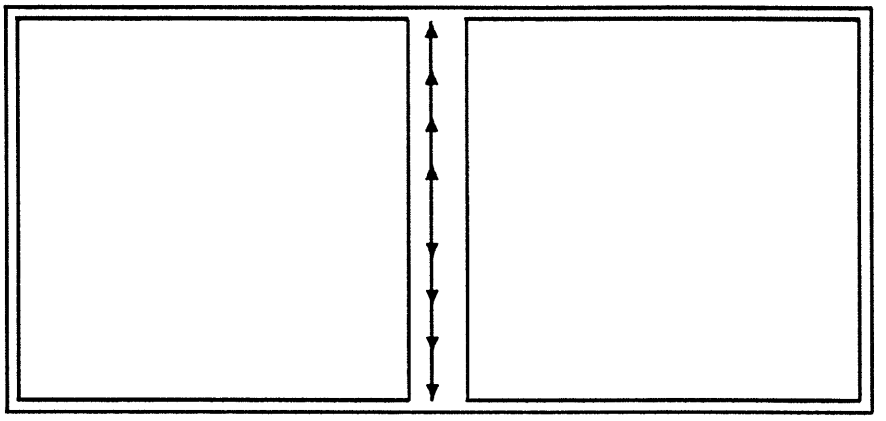


Figure 6-23: Filtering outward along a common boundary

process around the two boundaries which will be used to combine with similar models for neighboring boundaries. We accomplish this by filtering outward only along the boundary common to both regions. We therefore are performing causal filtering from $\psi = 0$ to some ψ_o when the corner is reached. See Figure 6-23. The inverse covariance for the remaining process can be recomputed and the Cholesky factorization computed again to arrive at a new model for the boundary which surrounds the two regions. The process for combining the next two neighboring regions can then proceed after the inverse estimation error covariance is re-approximated.

6.4.2 Suboptimal local filtering

Another opportunity for computational gains through suboptimal estimation arises in the local filtering step. By approximating the inverse covariance as in Section 6.3.1 for each value of ρ we may take advantage of the smoothing and filtering theory for STPBVDS's to perform estimation spiraling outward from the center to produce suboptimal estimates approximating optimal estimates based on data within a specified radius.

Here we will consider the case where we make approximation to the inverse covariance after each measurement update step. From Section 6.1 and 4.2 the measurements

have the form

$$\begin{bmatrix} \hat{X}(\rho) \\ 0 \\ Y(\rho+1) \end{bmatrix} = \begin{bmatrix} 0 & I \\ -E_{\rho+1} & A_\rho \\ C_\rho & 0 \end{bmatrix} \begin{bmatrix} X(\rho+1) \\ X(\rho) \end{bmatrix} + \begin{bmatrix} \tilde{X}(\rho) \\ B_\rho U_\rho \\ V(\rho+1) \end{bmatrix} \tag{6.178}$$

where the inverse estimation error covariance of the estimate at $\rho+1$ is given by

$$\Sigma_{X(\rho+1)} = E_{\rho+1}^T (A_\rho \Sigma_{X(\rho)}^{-1} A_\rho + B_\rho B_\rho^T) E_{\rho+1} + C_\rho^T C_\rho \tag{6.179}$$

The issue here is not to simplify the off-line computations but just the on-line computations. The idea here is to use the approximated inverse estimation error covariance (where all elements have magnitude greater than .01) in place of $\Sigma_{X(\rho)}$ and to approximate the result. The equation may then be written as

$$\hat{\Sigma}_{X(\rho+1)} = \text{Trunc}(E_{\rho+1}^T (A_\rho \hat{\Sigma}_{X(\rho)}^{-1} A_\rho + B_\rho B_\rho^T) E_{\rho+1} + C_\rho^T C_\rho) \tag{6.180}$$

where Trunc returns zero for all values smaller than a certain tolerance (.01).

The next question is how would a filter such as this perform. Figure 6-24 shows a maximum singular value plot for the difference between the optimal inverse estimation error covariance and the inverse of the suboptimal estimation error covariance. Specifically, consider the optimal ML estimate, given by

$$\hat{x} = (H^T \Sigma H)^{-1} H^T \Sigma y \tag{6.181}$$

where

$$\Sigma_x = H^T \Sigma H \tag{6.182}$$

is the inverse estimation error covariance for $\hat{x}$. The suboptimal estimate based on the same observation, is given by.

$$x' = (H^T \Sigma + \delta\Sigma H)^{-1} H^T (\Sigma + \delta\Sigma) y \tag{6.183}$$

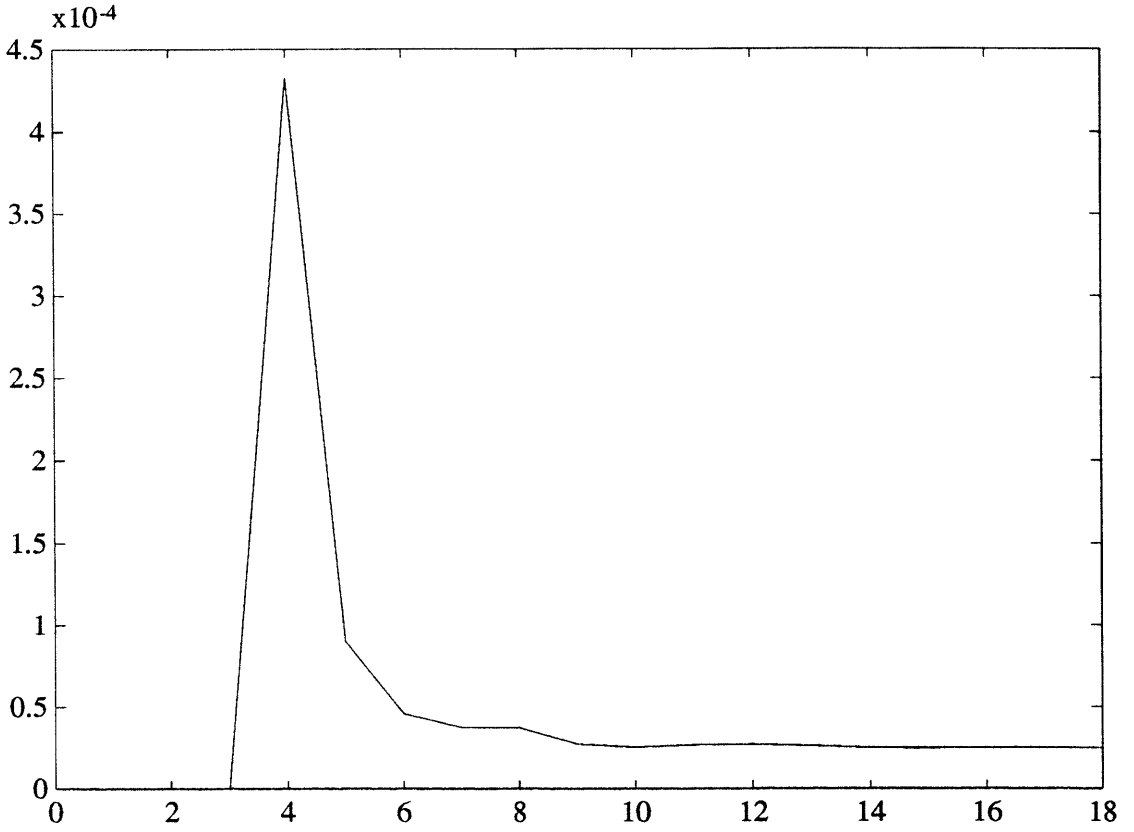


Figure 6-24: Maximum singular value plot for the difference between the optimal inverse estimation error covariance of the optimal filter and the actual inverse estimation error covariance for the suboptimal filter. The minimum singular value for the optimal inverse estimation error covariance is equal to 1.

The inverse estimation error covariance of the suboptimal estimate x' is not given by $H^T(\Sigma + \delta\Sigma)H$ but by

$$\Sigma_{x'} = H^T(\Sigma + \delta\Sigma)H\{H^T(\Sigma + 2\delta\Sigma + \delta\Sigma(\Sigma)^{-1}\delta\Sigma)H\}H^T(\Sigma + \delta\Sigma)H \qquad (6.184)$$

Figure (6-25) shows a maximum singular value plot for the difference between the computed (approximated) inverse estimation error covariance and the optimal inverse estimation error covariance.

With this approximation there are many ways of computing the estimates. Since the inverse covariance is banded we may use Cholesky factorization, or perhaps cyclic block reduction to solve for estimates. Furthermore we could use the results from the Cholesky decomposition to model the process around the boundary as a one dimensional process. The on-line computations turn into a filter operating around

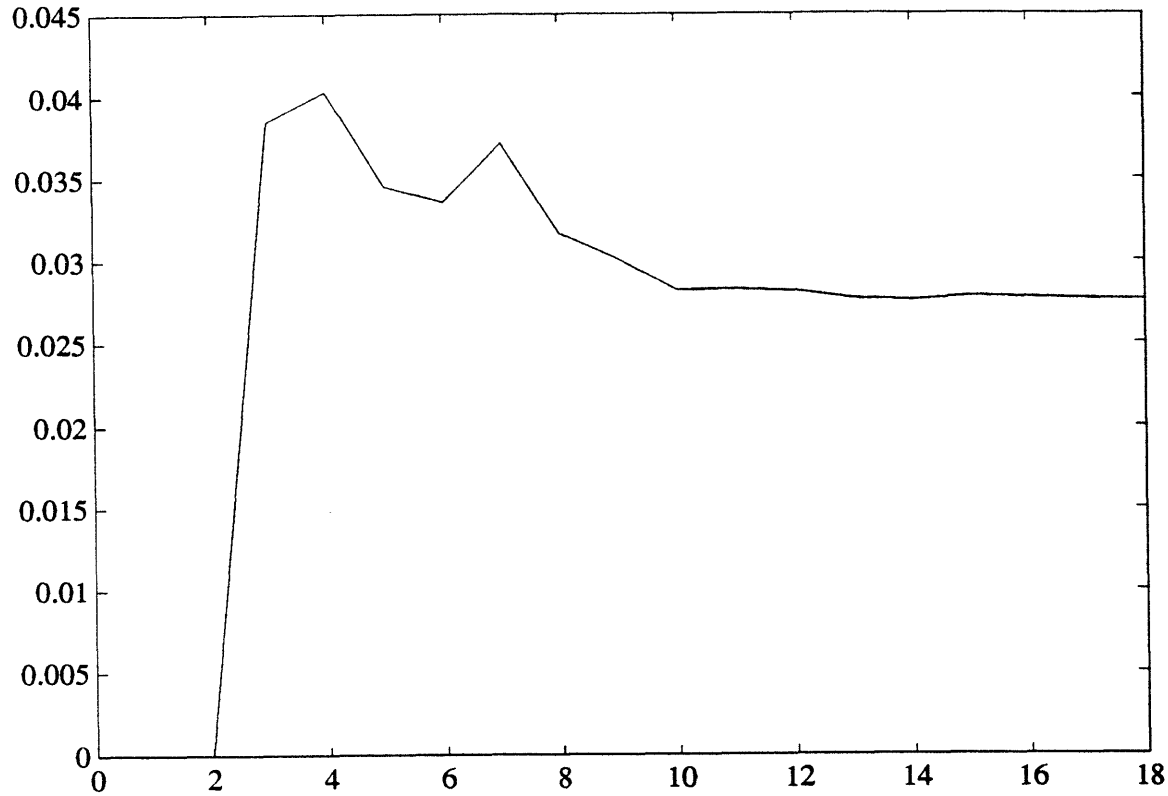


Figure 6-25: Maximum singular value plot of approximated inverse estimation error covariance to the optimal inverse estimation error covariance

the boundary of the region. From Section 6.3.1, the process for $X(\rho)$ is given by

$$E_{\rho,\psi+1}X_\rho(\psi+1) = A_{\rho,\psi}X_\rho(\psi) + B_{\rho,\psi}U_\rho(\psi) \tag{6.185}$$

A small number of the dynamic constraints represented by $E_{\rho+1}$, and A_ρ are given by

$$0 = \begin{bmatrix} L_{\rho,e}(\psi) & L_{\rho+1,e} \end{bmatrix} \begin{bmatrix} X_\rho(\psi+1) \\ X_{\rho+1}(\psi+1) \end{bmatrix} + \begin{bmatrix} L_{\rho,a}(\psi) & L_{\rho+1,a}(\psi) \end{bmatrix} \begin{bmatrix} X_\rho(\psi) \\ X_{\rho+1}(\psi) \end{bmatrix} + W(\psi) \tag{6.186}$$

where the partitioning of $X(\rho+1)$ compared to the partitioning of $X(\rho)$ with respect to the value ψ is shown in Figure 6-26. The remaining dynamic constraints and the observations have the form

$$Y_{\rho+1}(\psi) = H_{\rho+1}(\psi) \begin{bmatrix} X_\rho(\psi) \\ X_{\rho+1}(\psi) \end{bmatrix} + V_{\rho+1}(\psi) \tag{6.187}$$

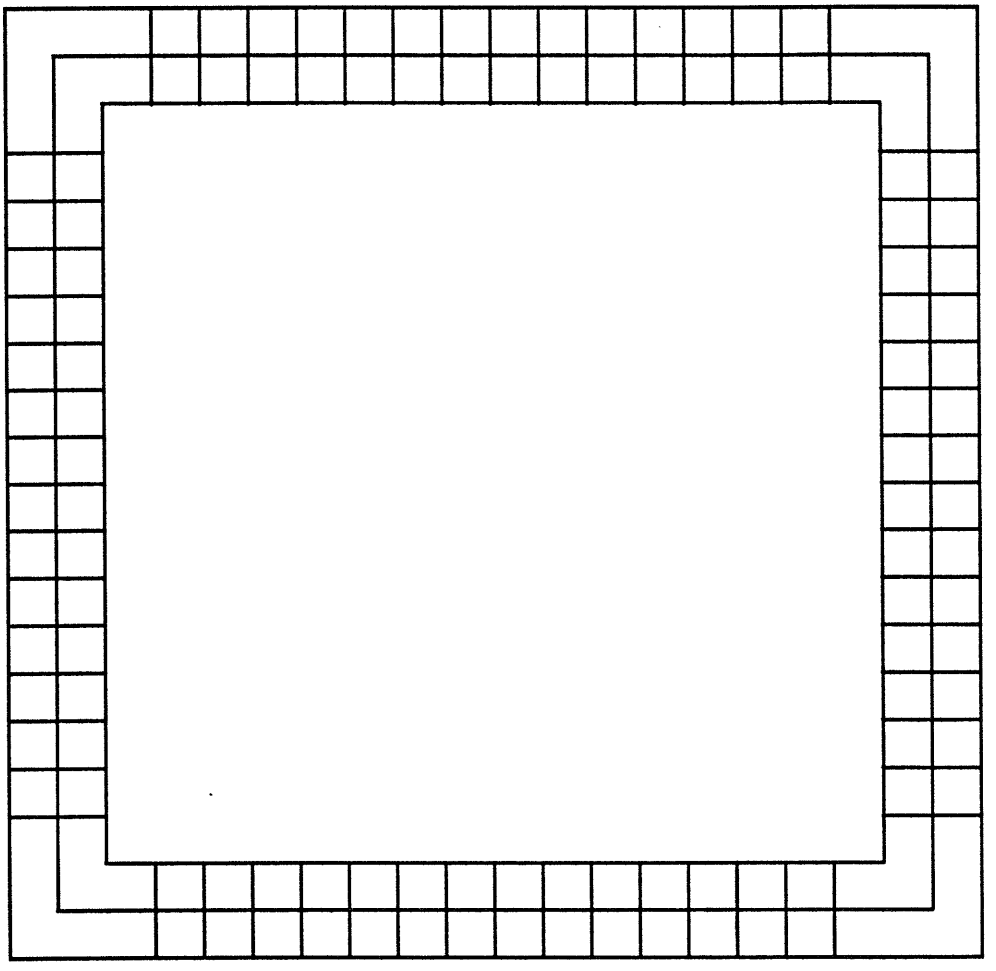


Figure 6-26: Partitioning the state $X(\rho + 1)$ so that it agrees with the partitioning of $X(\rho)$

by augmenting the state $X_\rho(\psi)$ with $X_{\rho+1}(\psi)$ the equations given by (6.185), (6.186), and (6.187) represent a large STPBVDS for which filtering would lead to on-line computational savings.

6.5 Complexity

Here we want to examine the complexity of the suboptimal methods discussed in the previous section. Here we are considering getting complexity estimates of various parts of the algorithm when different methods are applied. Under the assumption that the state is estimable with a well defined inverse error covariance, first we will consider the computation for the interprocessor communication , then we will consider the complexity for the filtering in the local processing step. We note that the complexity going up or down the tree are essentially the same, and as a result it is not necessary to explicitely compute the complexity for propagating smoothed estimates to the individual points in order to understand the functional relationships between the various variables and the complexity of the problem. In addition, the local complexity filtering outward is the same as obtained for the backward sweep of the Rauch-Tung-Striebel algorithm. We will not compute the complexity for this part of the algorithm either.

Interprocessor Communication

Here we will compare three methods for combining the estimates of the boundaries of neighboring square regions together to form estimates of the boundaries of rectangular regions. Here we will consider the complexity for one time step, and later we will sume the computation over all time steps.

Gauss-Seidel

Off-line

The computation involved to set up the Gauss-Seidel Computations is essentially to compute the inverse covariance. This will require approximately

$$\frac{5}{8} * (16)^3 2^{3t} (r + \tfrac{1}{2})^3 \tag{6.188}$$

for combining squares and

$$3 * (16)^3 2^{3t} (r + \tfrac{1}{2})^3 \tag{6.189}$$

for combining rectangles.

On-line

For diagonally dominant matrices, the rate of convergence can be bounded independent of the size of the matrix. As a result, the amount of iterations necessary to compute the estimate is independent of the size of the matrix. This will require approximately

$$C2(16)(88)\{2^t(r + \tfrac{1}{2}) - \tfrac{1}{2}\} \tag{6.190}$$

computations for combining squares together to form rectangles and

$$C2(24)(88)\{2^t(r + \tfrac{1}{2}) - \tfrac{1}{2}\} \tag{6.191}$$

flops for combining rectangles together to form squares. C here represents the number of iterations. Note that the complexity grows geometrically with t because the size of the state grows geometrically with t.

Cyclic Block Reduction

Cyclic block reduction has a structure which fits very well on a a hypercube. We will assume that this algorithm is implemented in parallel.

Off-line

The off-line computation for applying cyclic block reduction is given by

$$\begin{aligned} &\tfrac{5}{8} * (16)^3 2^{3t} (r + \tfrac{1}{2})^3 \\ &+3 * (16)^3 2^{3t} (r + \tfrac{1}{2})^3 \\ &+K(88)^4 \{t + \log_2 \tfrac{(16)(r+\frac{1}{2})}{88}\} \end{aligned} \tag{6.192}$$

where K is a constant associated with the details of the cyclic block reduction. Regardless the term involving K is dwarfed by the term with 2^{3t}.

On-line

The on-line computation for applying cyclic block reduction is given by

$$C(88)^3\{t + \log_2 \frac{(16)(r+\frac{1}{2})}{88}\} \tag{6.193}$$

where C is a constant associated with the on-line details of the cyclic block reduction. The on-line computation is growing linearly where the size of the vectors involved in the computation is growing geometrically.

If we model the process by a STPBVDS we may note that computing the elements of the model via the Cholesky factorization requires.

$$(16)2^t(r+\tfrac{1}{2})((44)^2+3(44)) + 2^t(16r+8)(44)^{-1}\mathcal{I}(218,88)\mathcal{I}(98,98) + 6(88)^2(96) + 4(88)^3] \tag{6.194}$$

flops. The on-line computation increases linearly with the bandwidth and the length of the state and is given approximately by

$$.5(16)^2(88)2^t(r+\tfrac{1}{2}) \tag{6.195}$$

for combining square regions and by

$$.33(24)^2(88)2^t(r+\tfrac{1}{2}) \tag{6.196}$$

for combining rectangular regions

Of these methods Cyclic block reduction has the advantage of being implementable in parallel and easily mapping into the architecture for which the remainder of the algorithm is designed to run. Furthermore the online computations increase logarithmically instead of polynomially with respect to the size of the state.

Filtering

The filtering step also benefits from sub-optimal techniques.

Gauss-Seidel

Off-line

Constructing the joint inverse covariance for $X(\rho)$, and $X(\rho+1)$. This requires approximately

$$10(16)^3\rho^3 \tag{6.197}$$

flops.

On-line

The on-line computation requires approximately

$$C2(88)(16)\rho \tag{6.198}$$

flops .

Cyclic Block Reduction

Off-line

Off-line, cyclic block reduction requires approximately

$$10(16)^3\rho^3 + K\log_2 \tfrac{16(\rho-\frac{1}{2})}{88}(88)^4 \tag{6.199}$$

flops off-line. For large ρ this is dominated by the term ρ^3.

On-line

The on-line computation required grows as

$$C\log_2\{\tfrac{16(\rho-\frac{1}{2})}{88}\}(88)^3 \tag{6.200}$$

Modeling by STPBVDS

Off-line

The amount of computation required for the filtering step is approximately

$$16(\rho-.5)((44)^2+3*44)+36(88)^2(16)\rho \tag{6.201}$$

On-line The amount of computation required for the filtering step is approximately

$$6(88)(16)\rho \tag{6.202}$$

Now that we have estimated the number of computations required for one time step, we will sum the computations over there respective domains.

Interprocessor Communication

Now the figures for one time step will be summed to compute the computational complexity over the proper region. Specifically we will sum the interprocessor communication figures from $t = 0$ to $t = T$

Gauss-Seidel

Off-line

The computation involved to set up the Gauss-Seidel Computations is essentially to compute the inverse covariance. This will require approximately

$$\frac{5}{8} * (16)^3(r + \tfrac{1}{2})^3[\tfrac{2^{3(T-1)}-1}{7}] \tag{6.203}$$

for combining squares and

$$3 * (16)^3(r + \tfrac{1}{2})^3[\tfrac{2^{3(T-1)}-1}{7}] \tag{6.204}$$

for combining rectangles.

On-line This will require approximately

$$C2(16)(88)\{(r + \tfrac{1}{2})\}[\tfrac{2^{2(T-1)}-1}{3}] \tag{6.205}$$

computations over the entire region for combining squares together to form rectangles and

$$C2(24)(88)\{(r + \tfrac{1}{2})[\tfrac{2^{2(T-1)}-1}{3}]\} \tag{6.206}$$

flops for combining rectangles together to form squares.

Cyclic Block Reduction

Off-line

The off-line computation for applying cyclic block reduction is given by

$$\begin{aligned}&\tfrac{5}{8} * (16)^3(r+\tfrac{1}{2})^3[\tfrac{2^{3(T-1)}-1}{7}] \\ &+3 * (16)^3 2^{3t}(r+\tfrac{1}{2})^3 \\ &+K(88)^4\{[.5T^2-.5T]+[T-1]\log_2 \tfrac{(16)(r+\frac{1}{2})}{88}\}\end{aligned} \tag{6.207}$$

As before, for large T, the term involving K is dwarfed by the term with 2^{3T}.

On-line

The on-line computation for applying cyclic block reduction is given by

$$C(88)^3\{[.5T^2-.5T]+[T-1]\log_2 \tfrac{(16)(r+\frac{1}{2})}{88}\} \tag{6.208}$$

The on-line computation is growing quadratically while the size of the vectors involved in the computation is growing geometrically.

Modeling the process by a STPBVDS

Off-line

If we model the process by a STPBVDS we may note that computing the elements of the model via the Cholesky factorization requires.

$$\begin{aligned}&(16)\tfrac{2^{3(T-1)}}{7}(r+\tfrac{1}{2})((44)^2+3(44))+\tfrac{2^{3(T-1)}}{7}(16r+8)(44)^{-1}[\mathcal{I}(218,88)+\mathcal{I}(98,98)] \\ &+6(88)^2(96)+4(88)^3\end{aligned} \tag{6.209}$$

flops. The function $\mathcal{I}$ is define in Section 2.1.

On-line

The on-line computation increases linearly with the bandwidth and the length of the state and is given approximately by

$$.5(16)^2(88)\tfrac{2^{2(T-1)}}{3}(r+\tfrac{1}{2}) \tag{6.210}$$

for combining square regions and by

$$.33(24)^2(88)\frac{2^{2(r-1)}}{3}(r+\tfrac{1}{2}) \tag{6.211}$$

for combining rectangular regions

Of these methods Cyclic block reduction has the advantage of being implementable in parallel and easily mapping into the architecture for which the remainder of the algorithm is designed to run.

Filtering

The filtering step also benefits from sub-optimal techniques.

Gauss-Seidel

Off-line

Constructing the joint inverse covariance for $X(\rho)$, and $X(\rho+1)$. This requires approximately

$$10(16)^3[.25r^4+.5r^3+.25r^2-1] \tag{6.212}$$

flops.

On-line

The on-line computation requires approximately

$$2(88)(16)[.5r^2+.5r-1] \tag{6.213}$$

flops per iteration over the entire sub-region

Cyclic Block Reduction

Off-line

Off-line, cyclic block reduction requires approximately

$$\begin{aligned} &10(16)^3[.25r^4+.5r^3+.25r^2-1] \\ &+K\textstyle\sum_{\rho=2}^{r}\log_2\{\frac{16(\rho-\frac{1}{2})}{88}\}(88)^4 \end{aligned} \tag{6.214}$$

flops off-line. For large r this is dominated by the term r^4.

On-line

The on-line computation required grows as

$$C\sum_{\rho=2}^{r}\log_2\{\tfrac{16(\rho-\frac{1}{2})}{88}\}(88)^3 \leq Cr[\log_2(r-\tfrac{1}{2})] \tag{6.215}$$

Modeling by STPBVDS

Off-line

The amount of computation required for the filtering step is approximately

$$16(.5r^2-r)((44)^2+3*44)+36(88)^2(16)[r^2-r] \tag{6.216}$$

On-line

The amount of computation required for the filtering step is approximately

$$6(16)(88)(r^2-r) \tag{6.217}$$

Cyclic block reduction is clearly the preferred method of performing suboptimal computation with $\mathcal{O}(r\log_2 r)$ growth for local filtering, and T^2 growth in the computations involving the interprocessor communication step.

Chapter 7

Conclusion

Along the road to developing a general multidimensional smoothing algorithm. several things were accomplished.

In Chapter 2 we established the general framework for Maximum Likelihood (ML) estimation to be applied for the remainder of this thesis. This general framework allows for estimation in statistical environments with large dynamic range. Specifically ML estimates are constructed for vectors which are not completely estimable in the presence of perfect observations. In this situation covariance matrices and inverse covariance matrices are not well-defined and as a result we use projection matrices to specify a covariance matrix for the part of the vector which is in fact estimable. The computations involved in finding the ML estimate are in general not well-posed and pseudo-inverses are required. The Moore-Penrose pseudo-inverse guarantees that we compute minimum norm ML estimates. We note that these computations can be formulated in a square root context and noted the complexity of the ML estimation computations for later reference.

In Chapter 3 Two Point Boundary Value Descriptor Systems (TPBVDS's) are discussed. These systems differ from causal systems in two ways. The first is that the boundary conditions at the two endpoints are coupled. Secondly the equations are defined implicitly resulting in no natural direction of recursion. In particular, the presence of possibly singular matrices multiplying both $x(k)$ and $x(k+1)$ make issues of filtering and smoothing more complicated than those associated with causal sys-

tems. We also showed the utility of the class of Separable Two Point Boundary Value Descriptor Systems (STPBVDS's) because these systems are Markov, and are diagonalizable into forward propagating, and backward propagating subsystems. These systems are not restricted to have constant coefficients, nor are they restricted to have a state with constant dimension. Since all TPBVDS can be described by STPBVDS by appending the state $x(-k)$ to $x(k)$, algorithms designed for STPBVDS's are also applicable to TPBVDS's.

In Chapter 4 useful lemmas are developed which encapsulate the basic requirements for recursive estimation and smoothing. A filter is then presented which recursively computes filtered ML estimates for STPBVDS's, generalizing the Kalman filter. The filter does not require that the model be well-posed, and perfect measurements can be handled. This recursive filter in addition to propagating the estimation error covariance and the ML estimate also propagates a projection matrix which keeps track of the estimable subspace. The Mayne-Fraser two filter algorithm and the The Rauch-Tung-Striebel algorithm are adapted for smoothing STPBVDS's. The computational complexities of these algorithms are computed for different statistical environments. A square root version of the forward Maximum Likelihood filter (FMLF) is constructed to demonstrate that all of the algorithms can be easily formulated in the square root context yielding greater numerical accuracy.

In Chapter 5 new parallel smoothing algorithms are developed for one dimensional processes. The algorithms presented all parse the data into segments on each of which one processor is assigned to operate. Local processing is performed first in which each processor produces a sufficient statistic representing the summary of information about the process at the boundary based on local observations. An interprocessor communication step follows where sufficient statistics are passed forward and backward through the processors until all processor have sufficient information to compute the smoothed estimate of the process locally at each of the interior points. Finally the smoothed estimates are computed locally and in parallel. Three algorithms assume a linear array of processor with nearest neighbor communication between them. One algorithm is based on the assumption that the processors are arranged

on a hyper-cube. The complexity of the problem grows with the logarithm of the number of data points.

In Section 5.4 a parallel smoothing algorithm is constructed so that from the perspective of each processor, the computation of the neighbors are known and certain information is expected in order to simplify the computation of the smoothed estimate. Each processor essentially computes the ML estimate of the local process given that the observations of the process outside of its local region is exactly zero. In other words the zero state response of the optimal smoother is computed. The zero input response is used to compute, pass, and update boundary information from processor to processor. Finally the ZIR is used to update interior points in each interval in parallel.

In Section 5.3 a parallel smoothing algorithm is constructed so that locally smoothed estimates are produced in parallel. Forward and backward ML filters which operate on the boundaries bring global forward and backward filtered estimates of local boundaries to each processor. This information is used to update each processor's locally smoothed estimates to produce globally smoothed estimates. It was concluded that this was not the proper way to perform two-dimensional smoothing because the state at each point was augmented with the boundary state. In two dimensional systems, the boundary state is large and would tremendously increase the computational complexity of the algorithm.

In Section 5.5 a parallel smoothing algorithm is presented which performs optimal ML filtering from the center of each local sub-region outward to the boundaries The system is reduced to a sampled STPBVDS made of the dynamic constraints which link the local regions together and where the results of the local computations play the role of observations. The Mayne-Fraser algorithm developed in Chapter 4 is sufficient to smooth the sampled process at the boundaries, and finally the local processor can perform smoothing from their local boundaries back to the center of each sub-region using the Rauch-Tung-Striebel algorithm.

In Section 5.6 it is noted that the interprocessor communication step which is used in the algorithm in Section 5.5 is a non-parallel implementation of the Mayne Fraser

algorithm operating on a sampled system. This interprocessor communication computation can itself be performed in parallel. Carrying this parallelization to repeated levels yields an algorithm where the computations are arranged on a binary tree and the processors and their communication links are arranged on a hyper-cube. This algorithm shows the most promise for adaptation for recursive and parallel multi-dimensional estimation. This algorithm also computes smoothed estimates in in an amount of time proportional to the logarithm of the number of data points, as compared to the polynomial time required for the other algorithms in Chapter 5.

Chapter 6 considers the problem of smoothing two dimensional systems. The region is divided into squares of equal size. The initial local processing involves filtering from the center of each region to the local boundary. Regions are grouped two at a time to combine local boundary estimates into estimates of larger and larger boundaries until the boundary of the entire region is computed. The smoothed estimate at boundary of the entire region is combined with the estimates of the two smaller boundaries from which the larger boundary was constructed. Smoothed estimates are constructed from these smaller boundaries and the process is continued until smoothed estimates are obtained for the boundaries of each individual sub-region. Local smoothing is carried out from the local boundary to the center using the Rauch -Tung-Striebel algorithm. Like the algorithm in Section 5.6 the computation can be arranged on a binary tree and the processors and their communication links can be arranged on a hyper-cube. During the interprocessor communication the size of the state grows geometrically. Although the number of times processors communicate is proportional to the logarithm of the number of processors, the computation on the larger boundaries dominate the complexity. As a result it is necessary to find clever sub-optimal techniques to deal with the boundary computations.

Section 6.4 discusses avenues for sub-optimal computation of the filter boundaries both in the interprocessor communication step and in the local processing steps by exploiting structure, and sparsity in the error covariance and inverse error covariance of the process. Through an example we show that indeed both the error covariance and the inverse error covariance both may exhibit structure allowing us to model the

process around the boundary by a moving average process or by a STPBVDS. Also more traditional methods of exploiting the sparsity such as performing cyclic block reduction on the ML estimation equations or performing Gauss-Seidel iterations allow the amount of computation associated with the processing of the large matrices in the interprocessor communication to be greatly reduced allowing for efficient computation of estimates of large two-dimensional regions.

7.1 Future Work

There are many avenues or research which remain.

While much is known about the system theory regarding STPBVDS's the theory regarding STPBVDS's with time varying coefficients is less plentiful, particularly in the arena of systems whose state dimension varies as a function of time.

Work needs to be done in the finite support estimation problem demonstrating how the statistics and characteristics of the process may lead to reduced computation times and faster algorithms. By eliminating the need to propagate information by all of the processors the interprocessor communication time can be reduced and a tradeoff between smoother performance and computation time will become apparent.

The parallel algorithms are applicable to a variety of statistical environments including the deterministic environment. There is therefore the possibility to construct general algorithms for the parallel implementation of digital filters. The work in this thesis already represents the parallelization of an acausal filter which is the solution to the optimal smoothing problem. In essence, if we know the problem for which an acausal filter is the optimal estimator, all the work has been done to implement this in parallel. Furthermore the possibility exists to construct parallel and acausal state observers and parallel implementation of general digital filters i.e., filters which would not arise in the smoothing context.

The structure of the parallel smoothing algorithms is quite similar to the algorithms for multi-resolution smoothing algorithms described in [26]. This suggests a strong connection between reciprocal processes and tree processes. In fact, the tree

processes as a result are far more general than was originally assumed, and include higher dimensional processes as a result. Higher dimensional processes modeled on trees will have the same issues as discussed in Chapter 6, namely that the size of the state is not constant, and there are several associated computational costs associated with the states whose dimension varies geometrically. There may be a way of incorporating wavelets into the description of reciprocal processes and constructing faster algorithms as a result.

Although Markovianity was used to derive the parallel smoothing algorithms in Sections 5.6 and Chapter 6, we note that it was not necessary by the fact that certain dynamic constraints can be added which do not spoil the the structure of the algorithm. It would be interesting to find just how general a set of dynamic constraints will allow the methods used in Section 5.6 compute smoothed estimates.

There is no reason why the multidimensional smoothing algorithm (particularly when applied to the one dimensional processes in in Section 5.6) cannot compute Bayesian estimates. While using ML estimation reduces the complexity of the algorithm both computationally, and intellectually, prior information can and should be incorporated, to eliminate the need for the projection matrices due to lack of estimability of the state based on local data. This may result in replacing the complexity due to carrying along projection matrices with that involving dealing with correlated estimates. However pseudo-inverses would be eliminated resulting in computational savings.

æ

Bibliography

[1] Adams, Milton Bernard Jr. , *Linear Estimation of Boundary Value Stochastic Processes*, MIT-Laboratory for information and Decision Systems, LIDS-TH-1295

[2] Adams, Milton Bernard, Alan S. Willsky, Bernard C. Levy, *Linear Estimation of Boundary Value Problems-Part II: 1-D Smoothing Problems*, IEEE Transactions on Automatic Control, Vol AC-29, No 9, Sept 1984

[3] Adams, Milton Bernard, Alan S. Willsky, Bernard C. Levy, *Linear Estimation of Boundary Value Problems-Part I: 1-D Smoothing Problems*, IEEE Transactions on Automatic Control, Vol AC-29, No 9, Sept 1984

[4] Kayalar, Selahattin, Howard L.Weinert, *Oblique Projections: Formulas, Algorithms and Error Bounds*, Math. Control Signals and Systems (1989)2: 33-45

[5] Bello, Martin G., Alan S. Willsky, Bernard C. Levy, and David A. Castanon, *Smoothing Error Dynamics and Their Use in the Solution of Smoothing and Mapping Problems*,IEEE Transactions on Information Theory, Vol IT-32, No.4, July 1986

[6] Bierman, Gerald J.,*Factorization Methods for Discrete Sequential Estimation* , Mathematics in Science and Engineering Series, Vol 128, Academic Press, 1977

[7] Levy, Bernard C., David A. Castanon, George C. Verghese, and Alan S. Willsky, *A Scattering Framework for Decentralized Estimation Problems*, Automatica Vol 19, No 4, pp373-384, 1983

[8] Sebek, Michael, Mauro Bisiacco,and Ettore Fornasini Controllability and Reconstructability Conditions for 2D Systems IEEE Transactions on Automatic Control, Vol AC-33, No.5 pp496-499, May 1988

[9] Kuo,Chung-Chieh, *Discretization and Solution of Elliptic PDEs: A Transform Domain Approach*, Center for Intelligent Control Systems, MIT CICS-TH-11 Aug 1987 (domain dec)?

[10] Jain, A. K., and E. Angel *Image Restoration, Modelling, and Reduction of Dimensionality* IEEE Trans Comput. No 5, C-23, pp470-476, May 1974

[11] Speyer J.L., *Computation and Transmission Requirements for a Decentralized Linear-Quadratic-Gaussian Control Problem*, IEEE Transactions on Automatic Control, Vol AC-24, pp266, 1979

[12] Willsky, Alan S., Martin G. Bello, David A. Castanon, Bernard C. Levy and George C. Verghese, *Combining and Updating of Local Estimates and Regional Maps along One-Dimensional Tracks*, IEEE Transactions on Automatic Control, Vol AC-27, No.4, pp799-813, August 1982

[13] Hashemipour, Hamid R., Sumit Roy, and Alan J. Laub, *Decentralized Structures for Parallel Kalman Filtering*, IEEE Transactions on Automatic Control, Vol AC-33, No.1, January 1988

[14] Morf, M., J.R. Dobbins, B. Freidlander, and T. Kailath, *Square Root Algorithms for Parallel Processing in Optimal Estimation*, Automatica, Vol-15, pp. 299-306, 1979

[15] Tewfik, A.H., B.C. Levy, and A.S. Willsky, *A New Distributed Smoothing Algorithm*, MIT Laboratory for Information and Decision Systems, (LIDS-P- 1501), Aug 1988

[16] Tewfik, Ahmed.H., *Parallel Smoothing for Time-Invariant Two-Point Boundary Value Systems* Proceedings of the 27^{th} Conference on Decision and Control

[17] Catlin, Donald E.,*Estimation of random States in General Linear Models*, IEEE Transactions on automatic control, Vol. 36, No.2, February 1991

[18] Campbell, S.L., and C.D. Meyer, Generalized Inverses of Linear Transformations.London: Pitman, 1979

[19] Ljung Lennart, and Thomas Kailath , *A Unified Approach to Smoothing Formulas*, Automatica, Vol. 12, pp147-157, 1976

[20]

[21] Gelb, Arthur ed., Technical Staff of Analytic Sciences Corporation *Applied Optimal Estimation* MIT Press, Cambridge, 1974

[22] Nikoukhah, Ramine *A Deterministic and Stochastic Theory for Two-point Boundary Value Descriptor Systems*, MIT-Laboratory for information and Decision Systems, LIDS-TH-1820

[23] Nikoukhah, R., A.S. Willsky, and B.C. Levy, *Kalman Filtering and Riccati Equations for Descriptor Systems* Proceedings of the 29^{th} IEEE Conference on Decision and Control, Dec 1990

[24] Nikoukhah, Ramine, Milton B. Adams, Adam S. Willsky, and Bernard C. Levy, *Estimation for Boundary Value Descriptor Systems* Circuits Systems Signals Process, Vol 8, No 1, 1989

[25] Verghese,George, C.,T. Kailath, *A Further note on Backward Markovian Models* IEEE IT-25 No. 1, Jan 1979

[26] Chou, K.C., *A Stochastic Modeling Approach Multi-Scale Signal Processing* Ph.D Thesis, MIT, Jun 1991

[27] Fogel, Eli, and Huang, Y.F., *Reduced order state estimator for linear systems with partially noise corrupted measurement*, IEEE Transactions on Automatic Control, Vol. AC-25, No. 5, Oct 1980

[28] Levy, B.C., M.B. Adams, A.S. Willsky, *Solution and Linear Estimation of 2-D Nearest neighbor Models* Proceedings of the IEEE Vol.78, No. 4, April 1990

[29] Levy, Bernard C., David A. Castanon, George C. Verghese, and Alan S. Willsky, *A Scattering Framework for Decentralized Estimation Problems*, Automatica Vol 19, No 4, pp373-384, 1983

[30] Bersekas, Dimitri P., John N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Prentice Hall, 1989

[31] Golub, Gene Howard, Charles F. Van Loan, em Matrix Computations Johns Hopkins University Press, 1989

[32] Faiman, F.W., and L. Luk, *On reducing the order of Kalman Filters for discrete time Stochastic Systems having singular measurement noise*, IEEE Transactions on Automatic Control, Vol AC-30 No 11, Nov 1985

[33] Paige, C.C.,*Computer Solution and Perturbation Analysis of Generalized Least Squares Problems*, Math. Comp. #33 pp 171-184

[34] Wall, Joseph, A.S. Willsky, N.R. Sandell, *On the fixed Interval Smoothing Problem*, Stochastics, Vol 5 ,No. 1,pp1-41, Jan 1981

[35]

[36] Weinert Howard L., and Uday B. Desai, *On Complementary Models and Fixed Interval Smoothing* IEEE TAC-26, No. 4, Aug 1981

[37] Lainiotis, Dimitrius G.,*Joint Detection , Estimation and System Identification* Information and Control 19 pp75-92, 1971