A Multiscale Approach to State Estimation with Applications in Process Operability Analysis and Model Predictive Control

by

Matthew Simon Dyer

BSc(Hons), University of Cape Town (1993)
S.M., Massachusetts Institute of Technology (1995)

Submitted to the Department of Chemical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

Signature of Author . . . .
Department of Chemical Engineering
22 May 2000

Certified by . . . .
George Stephanopoulos
Arthur D. Little Professor of Chemical Engineering
Thesis Supervisor

Accepted by . . . .
Robert E. Cohen
St. Laurent Professor of Chemical Engineering
Chairperson, Department Committee on Graduate Students

A Multiscale Approach to State Estimation with Applications in Process Operability Analysis and Model Predictive Control

by

Matthew Simon Dyer

Submitted to the Department of Chemical Engineering
on 22 May 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This thesis explores the application of multiscale ideas to the areas of state estimation and control. The work represents a significant departure from the traditional representations in the time and frequency domains, and provides a novel framework that leads to fast, efficient, and modular estimation algorithms.

Multiscale methods were rediscovered through wavelet theory in the mid-eighties, as a tool for the geophysics community. Like Fourier theory, it provides a more instructive representation of data than time series alone, by decomposition into a different set of orthonormal basis functions. Multiscale models and data sets exist on multiscale trees of nodes. Each node represents a place holder corresponding to a time point in a time series. The nodes of a tree form a structure which may contain measurements, states, inputs, outputs, and uncertainties. Each level of the tree represents the set of data at a given level of resolution. This dual localization in time and frequency has benefits in the storage of information, since irrelevant data and pure noise can be identified and discarded. It also preserves time and frequency information in a way that Fourier theory cannot. Grouping and condensing of important information follows naturally, which facilitates the making of decisions at a level of detail relevant to the question being asked.

Multiscale systems theory is a general approach for multiscale model construction on a tree. This thesis derives the multiscale models corresponding to the Haar transform, which produces a modified hat transform for input data. Autoregressive models, commonly used in time series analysis, give rise to multiscale models on the tree. These allow us to construct numerical algorithms that are efficient and parallelizable, and scale logarithmically with the number of data points, rather than the linear performance typical for similar time-series algorithms. This multiscale systems theory generalizes easily to other wavelet bases. Multiscale models of the underlying physics and the measurement model can be combined to construct a cost function which estimates the underlying physical states from a set of measurements. The resulting set of normal equations is sparse and contains a specialized structure, leading to a highly efficient solution strategy.

A modified multiscale state estimation algorithm incorporates prior estimates, consistent with the Kalman filter, with which it is linked. A constrained multiscale state estimator incorporates constraints in the states, and in linear combinations of the states. All incarnations of the multiscale state estimator provide a framework for the optimal fusion of multiple sets of measurements, including those taken at different levels of resolution. This is particularly useful in estimation and control problems where measurement data and control strategies occur at multiple rates. The arbitrary size of the state allows for the use of higher order underlying physical models, without modification of the estimation algorithm. Finally, the algorithm accommodates an arbitrary specification of the uncertainty estimates at any combination of time points or level of resolution.

The structure of the solution algorithm is sufficiently flexible to use the same intermediate variables for all of these modifications, leading to considerable reusability, both of code, and of prior calculations. Thus, the multiscale state estimation algorithm is modular and parallelizable.

An uncertainty analysis of the algorithm represents state estimation error in terms of the underlying model and measurement uncertainties. Depending on the size of the problem, different techniques should be used to construct the probability distribution functions of the error estimates. This thesis demonstrates direct integration, propagation of the moments of the measurement and model errors, polynomial chaos expansions, and an approximation using Gaussian quadrature and Monte Carlo simulation.

A sample of smaller case studies shows the range of uses of the algorithm. Three larger case studies demonstrate the multiscale state estimator in realistic chemical engineering examples. The terephthalic acid plant case study successfully incorporates a non-linear model of the first continuously stirred tank reactor into the multiscale state estimator. The paper-rolling case study compares the multiscale state estimator to the Karhunen-Loeve transform as a means of state estimation. Finally, the heavy oil fractionator of the Shell Control Problem demonstrates the multiscale state estimator in a control setting.

Thesis Supervisor: George Stephanopoulos
Title: Arthur D. Little Professor of Chemical Engineering

Acknowledgements

This thesis has been a long and difficult part of my life, and a substantial cast of people has entered, exited and played a variety of roles. The completion of this thesis is a tribute to them. I'd like to thank my advisor, Professor George Stephanopoulos for providing me with a topic to explore that was challenging, risky, and had the potential to change the way people view the field of estimation and control. I leave with respect for your willingness to follow uncertain paths in search of dramatic changes, in the true spirit of academic research.

My committee members, Professor Paul Barton and Professor Greg McRae deserve thanks for providing a different perspective to the project, for guiding me through the considerable literature on optimisation and error analysis, without which this thesis would be shorter, but incomplete. Professors Blankschtein and Deen had faith in my project, and I thank them for their time and support.

The other professors in the department who have advised and guided me through the difficult patches of this experience know who they are - thanks.

Janet, Elaine, Carol and Gabrielle, and the rest of the support staff in building 66 have smiled when I've needed a smile, joked, and generally provided a warm and loving home from home.

The members of LISPE-BAMEL have kept me entertained, frustrated and intellectually stimulated over the past six years - and I thank Alex for getting me started on my tour of wavelets, Andreas, Ajay, Enrique and Manuel for their mature influence, Christine for making 66-363 a functional wrokplace (sic), Kiko for much shared bitterness, and support, Shahin for his advice, warmth and sense of humour, and my new boys (and woman), Bill, Jatin, Ahmed, Joanne, Daehee and Ameya, for scintillating conversation and for bringing youth back to the office. Jerry has been through most of the major steps with me - thanks for your friendship, and in particular for kicking my arse through the last few months, and a huge thanks to Orhan for making me work, laugh, despair, for challenging my thoughts constructively, and for being a wonderfully person to know and work with.

MIT is not all work, definitely not in my case, and many people have provided memories of my life outside the office. I thank my Tangmates, Bernard, Hung-Chou and Naved, for teaching me about MIT undergrad life, the Asian red ears, and when not to enter a room. I thank my Willow Ave housemates for many happy house dinners, parties, peasant lunches, and rarely, all being in the same room together - Roger, for his exuberance, enthusiasm, and for introducing me to my box, Pat, for his unique perspective, Simon, for dragging me out when I needed it, and for dragging the machine out when noone needed it, Derek, for sharing the third floor, and for caring enough to listen, Stefan, for his taste in furniture, Paul, for professional advice and friendship, Matt for his fondness for fine food, and young Phil, well, for being Phil. And to Market Street, where John has been a fabulous source of entertainment, “Oh, man!”.

The residents of West Newton St have complemented the straight male dominance of Willow Ave, and in more ways than with red shoes. I thank Vanessa for providing me with someone to buy “*that* book” for, Karen for hiking, roadtrips to Canada, and for just being around, and Chris, for embarking on Precision, and for doing what he wanted to do. Yoshi, Peter and the rest of thepack for Cornwalls (including Geo’s butt), Yoshifest, and assorted tastings at the World Trade Center. The residents of Columbia St, Marky Mark, Dave and Becs have been a house next door to visit, sometimes very late, and really know how to throw a New Years Party.

My later years have been tremendously influenced by my involvement with Nightline, where I have made a number of close friendships, and been moved and supported by many wonderful people, and had a safe haven to escape from many of my self-imposed disasters. I thank my co-coordinators, A and B for dealing with a number of tough issues, and for making the work pleasant, and Peter and Kim for keeping an eye on me. And to everyone else, keep up the good work, even when it is tough to do so, and I thank you all for many happy nights, cups of tea, retreats, movie nights, buddies, traumas and laughs.

Bridge has brought me much pleasure, and I thank CoCo for organising many relaxing Sundays, LoLo and JoJo, for joining in, and the other ChrisP bridge people for keeping me sane. The MIT-DL bridge club has provided a home for my competitive bridge endeavours, and Jason has been patient at dealing with my constant desire to add things to our system, and repeated silly bids of 7H. And my social bridge friends have provided many an evening of relaxation and red wine.

My friends in the department have made lunch a happy part of my day for years - from the days when it was mainly the Cohen group, Kane, Randy, John and Diane, to the recent Ying group dominance, the humour of Peter and the non-linear effect I have with Indranil, it has been fun and you will all be missed. Joydeep, Arpan, and Fred, and later Leonel's posse have been great hiking companions. Stevie must be thanked for introducing me to the pleasure of a good haircut and that golf course image.

The MIT South African community has provided a fun, supportive memory of home every month, and I thank Fred and Antonia, in particular, for taking care of me when I arrived. The Boston gay community has provided a different sort of family and a number of close friendships, in particular, those formed at the MIT coffeehouse and Dignity, and the patrons of ManRay. I thank Buzz for helping me through some tough times, and John and Alex for much fun.

This thesis has been tough to write, despite the accompaniment of the MIT-Logarhythms, who have kept me going through many a lonely night. David Collins and Josh Taylor have been the main people to keep me going, listen to me pine and moan, humour me, and generally be there when I needed someone to talk to. Thanks for being more help than I think you realise you were.

Finally, I'd like to thank my friends from Cape Town and Joburg, especially Dave, Jules and Derek, who have kept an eye on me from afar, and even visited me, and pushed and cajoled, and kept this experience in perspective. And most of all, my family, without whom I would certainly not have come this far, and would never have finished this thesis. Thanks very much - I love you all.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The multiscale formulation provides an attractive alternative to traditional time and frequency frameworks, for solving problems in process operations analysis and model predictive control. The framework originated as an outgrowth of developments in the wavelet decomposition, which has been used for the analysis of discrete systems. The framework represents process data on trees, which may be dyadic, or of higher order, where the nodes of the tree represent the process at multiple levels of resolution, and could contain states, measurements, input variables, and uncertainties in any of these. The dual localisation provided by this framework is attractive for many estimation and control problems, since it allows an explicit representation of knowledge acquired, measurements taken, and, decisions made at different levels of resolution and over specific finite windows of time. In addition, the algorithms that make use of the multiscale framework are considerably more efficient than those in the time or frequency domains, due to the parallelisability of the algorithms, which usually scale by $O(log(n))$, rather than $O(n)$, for n data points.

The multiscale state estimation algorithm discussed in this thesis solves a multiscale optimisation which formulates the state estimation problem. The multiscale optimisation problem arises naturally from similar optimisation problems in the time domain. The multiscale state estimation algorithm is performed by dividing the optimisation problem into parallelisable steps up each leg of the tree, which are solved to produce the optimal estimate of all states on a multiscale tree for a given cost function. The multiscale state equations are derived directly from time-domain state equations and a suitable averaging scheme, and this new set of consistent equations is used to constrain the optimisation problem. Alternative constraints on the measurement and modelling uncertainties may be imposed by physical considerations. The multiscale state estimation algorithm allows the operator to pose questions at relevant levels of resolution, rather than those specified by the data. It also allows for the optimal fusion of sets of data taken over different time scales. Further, the estimation algorithm can be used to identify a richer feedback error structure for use in a model predictive control scheme.

1.2 Motivation for multiscale research

In many chemical engineering applications, it is of interest to recognise, model and analyse phenomena occurring at different scales, and to utilise this information in process simulation, design, operations and control. This is particularly important for processes where phenomena at widely different time and length scales affect the behaviour of a process or structure, and need to be explicitly dealt with. Conventional Fourier techniques fail because of their inherent inability to deal with spatial localisation.

Wavelet transforms decompose signals and processes, both in time (or space), and scale, and offer a potential framework for such analysis, [6], [7], [8], [9], and [14]. The theory of multiscale representations leads naturally to models of signals on trees, such as those illustrated in Chapter 2. The collection of nodes at each scale represents some process, image or physical structure at a given resolution. The emphasis of this thesis will be on the interaction of the descriptions of a process at a given resolution, with a description of the same process at a different resolution.

The general problems tackled in this thesis are the identification of physical models from measurements at different resolutions, the reconciliation of physical models with sets of measurements, the fusion of different, possibly conflicting sets of measurements of the same process. The synthesis of internally consistent or compatible models of processes at different resolutions will begin the thesis, and concrete methods will be developed to obtain these models using the framework of the tree.

The specific problems that we tackle are state estimation with and without constraints and fusion of sensor data from a process where the sensors sample at multiple rates. An important application is the state estimation at various scales for model predictive control. In conventional model predictive control, a subset of the recorded input-output data set is used to construct a single model at the physical level, or at the resolution of a single sampling interval. This research advances that by identifying multiple internally compatible models at different resolutions to the sampling rate, which may be of different order to the finest level model. These models should produce internally compatible responses in the predicted output values. These problems are discussed in Chapter 6.

The first problem to be explored is the relationship between physical models at various scales, actual measurements at various scales, or equivalently, numerical approximations of physical behaviour at various scales, and an optimal reconciliation between these for incomplete data sets. The motivation for using this approach is the existence of multiscale features in many chemical processes, and the success of similar algorithms for statistical processes. We show that the multiscale domain is an efficient, versatile and intuitive framework for dealing with multiscale issues in chemical engineering.

1.3 Philosophical motivations for multiscale aspects in chemical engineering

There are three distinct ways in which multiscale features can enter a chemical engineering problem.

Firstly, the phenomenon under investigation may possess features and physically significant effects at multiple scales. Chemical kinetics can involve very different time scales in a single reaction mixture. Spectroscopic images that contain spatially localised features can be thought of as a superposition of fine features on a more coarsely varying background. Any process which exhibits self-similarity falls into this category.

Secondly, the data may be collected either at different sampling rates, or at different spatial resolutions, regardless of whether the physical process has multiscale features or not. Examples of this type of chemical engineering problem are numerous. A wide range of analytical measurements are made with different equipment involving resolutions varying by orders of magnitude, such as the range of resolutions from electron- to light microscopy. Furthermore, even if a single sensor type is used, it can often be operated in zoom modes, or at different sampling rates, which the scientist or engineer may wish to fuse. Any process in which multiple sampling rates exist and the information needs to be fused are prime candidates for multiscale analysis.

Thirdly, the algorithm for the data processing can utilise a multiscale structure to increase computational speed, whether the physical phenomenon or the data have multiscale features or not. multiscale algorithms often allow the user to specify the level of computational complexity that is desirable by specifying the desired level of accuracy. A feature of a multiscale algorithm, such as a multigrid relaxation scheme, is that it utilises an existing coarse solution to solve the problem at finer resolutions. This approach typically leads to computational efficiency and flexibility.

This thesis will focus primarily on the second and third aspects. The case studies all have to potential for multiple data sets, and for multirate data to be used, while the algorithms produced are clearly multiscale in nature.

1.3.1 Sensor fusion

State estimation at the time scale is fairly well developed for complete sets of measurements, collected at a single, regular sampling frequency. Often multiple sensors will exist that take data at different sampling frequency or spatial resolution. These independent measurements may cover the entire space, or only part of the space. They may be from a complete data set, or a partially damaged or corrupted data set. Multiple data points need to be combined in a way that produces the most realistic description of the observed process.

The types of problems that motivate the use of multiscale concepts are the following. Two or more sets of measurements are provided that represent measurements of the same quantity at different time scales, or spatial resolutions. The coarser set of measurements is an average, in some sense, of the finer sets. Chemical engineering examples of these would be microscopic measurements of surfaces with different magnifications, and thus with more information about specific regions. Composition analysis of different sample sizes taken from the same general area will provide similar averaging effect.

Two sets of measurements of the same quantity are provided where the coarser is a decimated version of the first. An example is a probed system where fewer samples are taken between one sampling and the next. A digital thermometer with zero response time would be such an probe, if we obtain the measurements by changing the sampling rate between experiments. This example becomes less trivial if we imagine an accurate, but slow thermometer working in collaboration with a cheap, but fast one.

1.3.2 Model predictive control (MPC)

Model predictive control is an optimal control based method which utilises an explicit model of the plant and set of operational and physical constraints to select the best control inputs. These inputs are selected to minimise an appropriate cost function, where future values of the system variables are predicted, based on a good estimate of the present states. Frequency analysis techniques used to design controllers, which provide guarantees for robustness and stability, fail because of the hard bounds on control inputs. The global information content of the frequency domain techniques cannot predict localised phenomena such as bound violations. multiscale analysis, localised both in time and scale (frequency), provides an advanced, integrated environment of both time and frequency domains and their corresponding techniques. In this thesis, we demonstrate how multiscale state estimation can be integrated into the multiscale formulation of the MPC problem. This results in an environment where measurements at different scales (rates) are fused consistently and efficiently, and optimal control problems are defined with flexible design considerations. Parallelisable algorithms are formulated to solve the integrated estimation and control problem, reducing the complexity drastically, thus giving an opportunity to solve larger problems faster and with more flexibility. The main multiscale model predictive control algorithm is discussed in the thesis of Orhan Karsligil.

1.4 Historical perspective

Previous work in the multiscale area has concentrated on modelling noise processes with a known statistical structure. The LIDS research group at MIT produced a series of theses, [4], [10], [11], [13], which developed and applied a multiscale state estimator for noise processes such as Brownian motion. The argument was that the wavelet decomposition of the signals and images at various resolutions was self-similar, and thus that state estimation could make use of this by constructing models that recognise the self similarity. Certain aspects of their work were very appealing, and have provided the motivation for the work in this thesis. Like us, they begin with a signal split into its wavelet scale components.

A dynamic system, with scale as the independent parameter, relates the values of the nodes at different scales. These dynamic systems in scale are used to derive a Kalman filtering type algorithm that incorporates measurements taken at different scales, and produces a maximum-likelihood estimate of the process at all scales. A benefit of this algorithm is its ability to handle sparse data, or missing data, and to reconcile conflicting data sets at different scales in a statistically optimal way. My thesis aims to extend this idea to deal with systems found in chemical engineering, where the states transform in a deterministic way, coupled with an uncertainty component that is statistical in nature. This contrasts with the purely statistical approach found in the LIDS work.

Ken Chou's doctoral thesis [4] and a series of papers in conjunction with Basseville et al [1],[2], [3] provide the framework of the analysis and modelling of statistical phenomena at multiple scales and the efficient estimation of signals given noisy or incomplete data sets. The algorithm they propose is suited to data sets recorded at different resolutions. This efficient optimal estimation algorithm is a multiscale adaptation of the Rauch-Tung-Striebel smoothing algorithm, and the Kalman filtering algorithm. While the Kalman filter was developed for processes that proceed in time, this multiscale filtering algorithm proceeds using scale as the time-like variable. This requires a dynamic system proceeds in scale, or up and down a tree, rather than across it. The algorithm presented in [4] uses complete sets of measurements at a given scale as new information, in the same way that a single new reading arrives for incorporation into a temporal Kalman filter.

The standard Rauch-Tung-Striebel Algorithm, is designed for a time series, and involves a forward Kalman filtering sweep followed by a backward sweep to compute smoothed estimates (Rauch et al 1965). The forward sweep incorporates new information as it is received, and produces state estimates based on a subset of the information, while the backward smoothing algorithm produces the best state estimates that satisfy the *a priori* specified model based on all of the data.

To initiate their tree-based RTS algorithm, the user specifies the tree-dynamic system to be used for the analysis - the A and B matrices at each point on the tree, and the statistics of the uncertainty variables, or the prior information. The algorithm begins with an upward sweep, where the measurements are incorporated from the finest level to the coarsest level in a process similar to Kalman filtering, but proceeding in scale rather than time. The filtered state estimates are used in a fine to coarse smoothing step to estimate a maximum likelihood of the state at each node, as well as the error covariance.

After the fine-to-coarse, or upwards sweep, we have an optimally smoothed estimate of the root node (or node at the top of the tree). Chou [4], [5], [5] demonstrates how it is possible to use this estimate to produce optimal smoothed estimates of all remaining nodes.

Despite the difference of having scale as the independent variable, there are strong similarities between this and the classical Kalman filter [12]. In both cases, the user inputs a set of measurements of some process which may not represent a complete set at any scale. Where measurements are missing, or unobtainable, the algorithm uses prior information to predict a best estimate of the unmeasured states. These measurements may represent a 1- or 2-dimensional spatial set of measurements sampled at various resolutions or a time series of measurements of some physical process.

The appealing features of this algorithm are that it is extremely efficient and highly parallelisable. The completely parallelised algorithm runs in $O(M)$ where M is the depth of the tree, and thus $O(log_2 N)$. This makes it ideal for massive data sets. The algorithm incorporates data at multiple resolutions and provides a maximum likelihood estimate of the states at all resolutions, thus it is suitable for researchers with data collected using different spectroscopic techniques that provide data at multiple resolutions.

The LIDS research has been successfully applied to a variety of problems where there is no *a priori* structure to the data being analysed. Applications have been found in the estimation of the height of the ocean based on the fusion of measurements taken from satellite data, passing ships, aeroplanes and oil rigs. The ocean is expected to be relatively flat, and thus any observed heights can be modelled as noise. Further applications have included the analysis of radar data to detect movement, essentially for use in military applications, where an enemy is to be identified and shot. Again, nothing is expected, thus the goal is to distinguish a real object in the viewfinder from the surrounding noise.

We were motivated by this research, and decided to adapt it to more structured models, and in particular those that retain their temporal component, as opposed to the static structure. First order models describe a substantial body of chemical problems, yet these did not fit into the statistical description presented by the LIDS group. Further, the error structures observed by the LIDS group did not seem to be appropriate for noise processes, and exogenous inputs in chemical processes. Our goal was to construct a state estimation algorithm that was tree-based, fast, parallelisable, and had the ability to incorporate measurements at many levels. This required the development of a new set of tree-based first order models, and then an adaptation of this set of models to a multiscale state estimator.

1.5 Outline of the thesis

The structure of the thesis is as follows. Chapter 2 contains the theoretical basis for the work, the development of wavelet theory and dynamic systems on a tree. These concepts show the evolution of a noise process on a multiscale tree when it is subjected to the Haar wavelet transform. The Modified Hat Transform is introduced as a necessary by product of our first order processes and the Haar process. The unconstrained multiscale state estimator is derived from first principles, followed by the constrained state estimator, and the chapter concludes with a discussion of the relationship between the time-based Kalman filter and the multiscale state estimator.

Chapter 3 explores the state estimate error statistics on the tree. An algorithm is constructed to derive the state error estimates in terms of a model for the measurement and modelling error processes. This enables a statistical description of the state estimate to be produced in terms of the moments of the measurement and modelling error processes.

Chapter 4 presents simple numerical case studies to demonstrate the features of the multi-scale state estimation algorithms, such as the use in higher order processes, sensor fusion, and the comparison to the Kalman filter.

Chapter 5 presents the paper rolling case study, where the estimation of paper thickness is obtained using the Karhunen-Loeve transform and a wavelet description of the coefficients in time. Chapter 7 presents the terephthalic acid case study, and techniques for dealing with non-linearities in the dynamic system. Chapter 6 presents the link between the multiscale state estimator and the multiscale model predictive control algorithm.

Chapter 8 presents conclusions of the work, and some directions for future research in this area.

Bibliography

[1] Basseville M., Benveniste A., Willsky A. "Multiscale Autoregressive Processes, Part 1: Schur-Levinson Parametrizations," *IEEE Transactions on Signal Processing*, *40*(8):1915–1934 (1992).

[2] Basseville M., Benveniste A., Willsky A. "Multiscale Autoregressive Processes, Part 2: Lattice Structures for Whitening and Modeling," *IEEE Transactions on Signal Processing*, *40*(8):1935–1953 (1992).

[3] Basseville M. Benveniste A., Chou K.C., Golden S.A. Nikoukhah R. Willsky A. "Modeling and Estimation of Multiresolution Stochastic Processes," *IEEE Transactions on Information Theory*, *38*(2):766–784 (1992).

[4] Chou, K.C. *A stochastic approach to multiscale signal processing*. PhD dissertation, Massachusetts Institute of Technology, 1991.

[5] Chou K.C., Willsky A.S., Nikoukhah R. "Multiscale Systems, Kalman Filters, and Ricatti Equations," *IEEE Transactions on Automatic Control*, *39*(3):479–492 (1994).

[6] C.K., Chui. *An Introduction to Wavelets*. Academic Press, 1992.

[7] C.K., Chui, editor. *Wavelets: A Tutorial in Theory and Applications*. Academic Press, 1992.

[8] Daubechies, I. "The wavelet transform, time-freqency localization and signal analysis," *IEEE Trans. Inf. Theory*, *36*:961 (1990).

[9] Daubechies, I. *Ten lectures on wavelets*. SIAM, 1992.

[10] Fieguth, P.W. *Application of multiscale estimation to large scale multidimensional imaging and remote sensing problems*. PhD dissertation, Massachusetts Institute of Technology, 1995.

[11] Irving, W. *Multiscale stochastic realization and model identification with applications to large-scale estimation problems*. PhD dissertation, Massachusetts Institute of Technology, 1995.

[12] Kalman, R. and R. Bucy. "New Results in Linear Filtering and Prediction Theory," *Journal of Basic Engineering:Trans of ASME*, 95–108 (1961).

[13] Luettgen, M. *Image Processing with multiscale stochastic models*. PhD dissertation, Massachusetts Institute of Technology, 1993.

[14] Mallat, S. G. "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans on PAMI*, *11*:674 (1989).

Chapter 2

Theoretical Aspects

2.1 Introductory mathematics

2.1.1 Autoregressive processes

Process models often utilise autoregressive, moving average or state space equations,[15]. The nth order autoregressive system with first-order input dynamics, abbreviated $AR(p)$ has the form

$$x[n+1] = a_0x[n] + a_1x[n-1] + \ldots + a_px[n-p] + b_0w[n] + u[n] \tag{2.1}$$

where $x[n]$ represents some state at time n, which is assumed to be evolving in time and could be any physical quantity, the nth order derivative of a physical property, or a vector made up of combinations of these. The a's and b's are constants, dependent on degree of discretisation and physical parameters. $w[n]$ is a noise process at time n, which, unless we specify otherwise, will be assumed to be white Gaussian noise with unit variance. This noise process collects unknown disturbances and uncertainties - the parts of the problem over which the user has no control. $u[n]$ is a user specified signal, and is usually referred to as a control variable. The model shown is a restriction to first-order input dynamics - higher order input dynamics would require the presence of terms $u[n-1], ..., u[n-k]$.

The moving average process, $MA(p)$ has the form

$$x[n+1] = a_0 x[n] + b_0 w[n] + b_1 w[n-1] + \ldots + b_{p-1} w[n-p+1] + u[n] \tag{2.2}$$

The autoregressive and moving average models are general enough to represent a variety of real chemical processes, and form the basis of a substantial number of models used for the control of chemical processes. The identification of states and system models is based on the manipulation of set of measurements in various ways. In all cases there is an implied measurement model of the form

$$y[n] = C[n] + v[n] \tag{2.3}$$

where C is the measurement matrix and $v[n]$ is a measurement noise process. Exact state measurements have $C = 1$ and $v[n] = 0$ for all times. For these cases only, $x[n]$ can be used to denote the measurement as well as the state.

While these processes have been described as processes varying in time, they could as easily describe a spatially varying process. For example, the solution of the steady state, one-dimensional heat equation can be represented as an $AR(2)$ process. These autoregressive processes form the basis for the dynamic systems of state used in the estimators developed in this chapter.

2.1.2 Trees and notation

The concept of a tree in multiscale systems theory is parallel to that of the time line in standard systems theory and dynamic systems. The tree is a collection of nodes that index information at a given resolution and within a given time period, in the same way that the time line is used to index information in time.

The multiscale system theory of Benveniste et al. [3] is defined on homogenous trees of order q. These are infinite, acyclic, undirected, connected graphs such that each node of the graph is connected by branches to exactly $q+1$ nodes. The nodes are indexed by τ, which denotes the position on the tree. Nodes can be used to index many types of variables, such as images and functions, $f(\tau)$, states, $x(\tau)$, measurements, $y(\tau)$, noise terms, $w(\tau)$, all of which

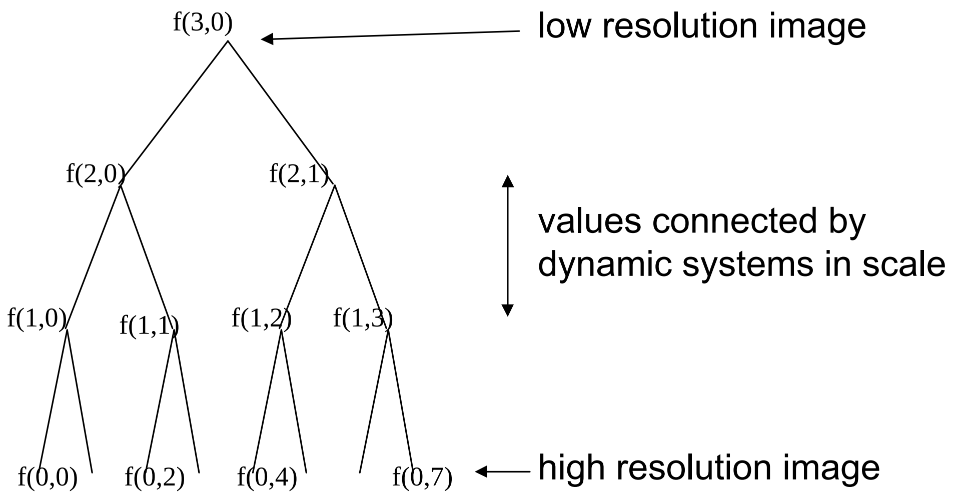

Figure 2-1: A Multiscale Tree of Order 2, Containing Process f

may be scalar or vector. The index τ can be replaced by a pair of numbers to indicate scale and position on the tree.

The Haar wavelet tree illustrated in figure 2-1 is an example of such a tree with $q = 2$. The process f can be thought of as some one-dimensional process, or image, being viewed at various levels of resolution, with the first variable indicating the scale, or level of resolution, and the second the spatial position. The next section will provide details of how these are obtained.

The general framework has an obvious interpretation for the cases $q = 2$ in 1-D problems, for example temperature, pressure and concentration profiles in time, or spatial profiles that vary in a single direction only. Trees with $q = 4$ are used in 2-D problems, and would be used to describe two-dimensional images, or processes that vary both in space and time. Figure 2-2 contains an example of three levels of such a quadtree. The uppermost level contains a single node, and is the coarsest description of the image or process. This can be viewed as

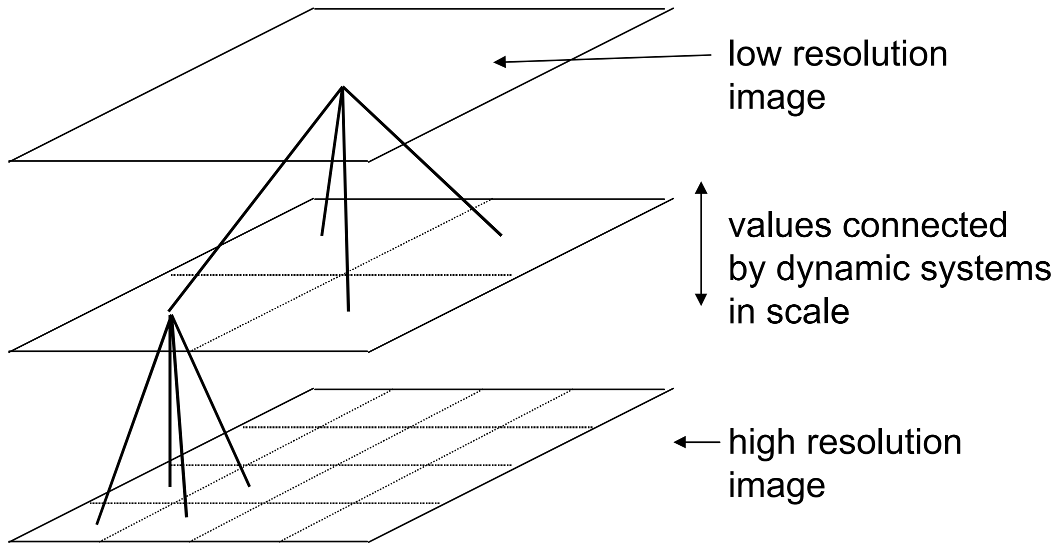

Figure 2-2: A Quadtree (q=4) with a Two Dimensional Process

a global average of the image, or the image perceived from a considerable difference. The middle level contains three more pieces of information, and thus a more detailed version of the information, while the lowest level contains the highest resolution. In principle, these levels extend infinitely far in all directions - to arbitrarily coarse descriptions of the process up the tree, and to arbitrarily fine descriptions down the tree. Note that each node of the quadtree contains four descendant nodes leading to a finer description, and a single ascendant node, leading to a coarser description.

The time line has a simple method of indexing, since there is bijection with the set of integers, one selects an anchor and enumerates outwardly from it. This anchor, or origin, is typically some arbitrary $t = 0$. On the tree, the indexing is more complicated since in addition to the arbitrary choice of origin, there are also multiple paths from a node. An indexing system is illustrated in Figure 2-3 .

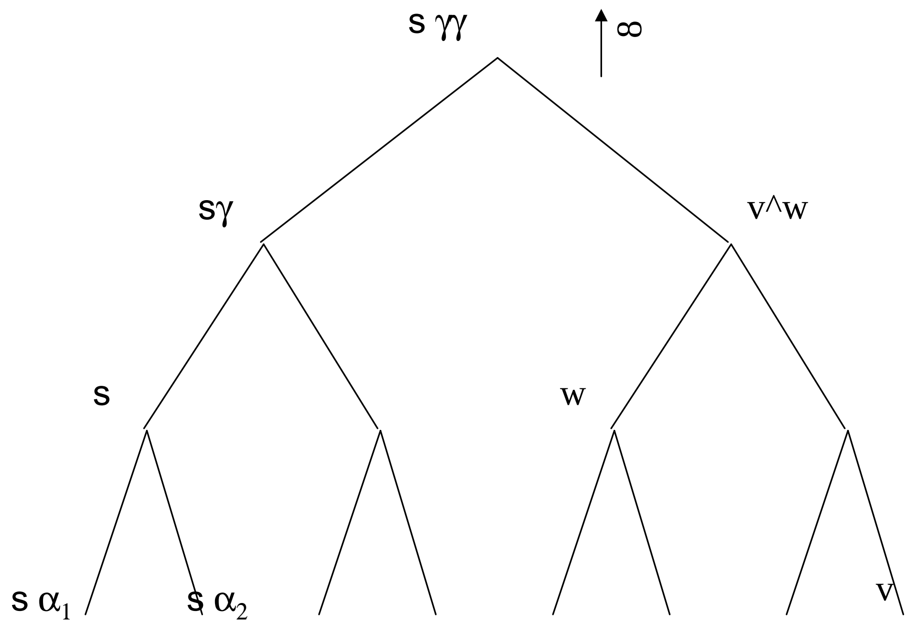

Figure 2-3: An Indexing System for a Multiscale Tree of Order q=2

For two nodes τ_1 and τ_2, there is a distance $d(\tau_1, \tau_2)$ which is the number of branches that must be traversed on the unique path joining τ_1 to τ_2. We can define a point of highest abstraction, referred to as ∞, from which there is a partial ordering of the nodes, τ, in terms of their distance from ∞. This corresponds to the arbitrary selection of a zero on the time line. This partial ordering leads naturally to the equivalence class of *horocycles*, or points that are equidistant from ∞. These horocycles are referred to (equivalently) as scales, levels of resolution, levels, and can be thought of as containing all of the information about a variable at a given level of detail. In the ordered pair technique of indexing, the first is assigned to the level.

Greek letters are used to indicate relationships between the nodes. The single upward branch of any node, or a movement towards ∞, is described by γ, and the node $\tau\gamma$, the node immediately above node τ on the tree, is referred to as the parent node. This is occasionally

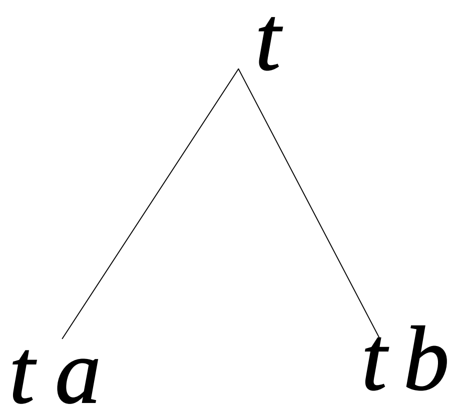

Figure 2-4: Descendant Node Geometry

referred to as the mother node in certain literature. Each descendant node is assigned an argument α_k, enumerated by definition. For $q = 2$, we define $k = 1$ as the left node, $k = 2$ as the right node, and for simplicity replace them with α and β. One can easily define a similar scheme for the quadtree, $q = 4$. For the remainder of the document, $\tau\alpha$ will refer to the descendant node from the left branch of τ while $\tau\beta$ will refer to the descendant node from the right branch of τ as illustrated in Figure 2-4.

All of these index relations can be applied recursively. In Figure 2-3, three nodes s, w and v are shown, with a parent and grandparent node of s illustrated by two applications of γ, as well as the two descendant nodes. The pair w and v are used to indicate the concept of lowest common ascendant, indicated by $w\hat{}v$. This is the unique node that is farthest from ∞, and can be described purely by expressions of the form $w\gamma^{k_w}$ and $v\gamma^{k_v}$, where the k's represent integers.

2.1.3 Wavelet theory

A substantial body of literature now exists on the topic of wavelet decompositions [6], [7], [8], [9], [14], [17], [11], [20], [1]. This treatment will focus only on topics directly pertinent to this research. Further details on the linear algebra aspects of wavelets may be found in Daubechies [8], the filtering aspects are dealt with in Mallat [17], and the vector space aspects are discussed in Strang [20].

The multiscale representation of a continuous function, $f(t) \in L^2(R)$, is a projection of the function onto a vector space spanned by a set of orthogonal functions called wavelets. Mallat [17] makes the following observations about the approximation operators:

1. There is a projection operator, A_{2j}, that approximates the signal $f(t)$ at resolution 2^J. It is a linear projection operator onto the space V_{2j}, which is the space of all possible functions in $L^2(R)$ that can be represented in terms of translations of the scaling function pertinent to that level.

2. Among all possible approximated functions at the resolution 2^J, $A_{2j}f(t)$ is the function which is most similar to $f(t)$.

3. The approximation of a signal at resolution 2^{J+1} contains all of the information required to make the approximation at 2^J. This leads to a set of nested vector spaces of functions of increasing resolution $V_{2j} \subset V_{2j+1}$.

4. Since the approximation operator is similar at all scales, the vector spaces at all scales should be related by appropriate scaling,

$$f(x) \in V_{2j} \Longleftrightarrow f(2x) \in V_{2j+1} \tag{2.4}$$

5. Each vector space, or scale, contains a characteristic length. Translation by a multiple of this characteristic length is invariant under projection from the original signal to the approximation.

6. The limits of these vector spaces are the zero function for the coarsest approximation and $L^2(R)$ for the finest approximation.

Many sets of vector spaces satisfy these conditions, specifically any scaling function, $\phi(x) \in$

$L^2(R)$ that satisfies

$$\phi_{2j}(x) = 2^j \phi(x) \ \ j \in Z \tag{2.5}$$

can be used to create a basis from its integer translations,

$$\sqrt{2^{-j}}\phi_{2j}(x - 2^{-j}n) \ \ j, n \in Z \tag{2.6}$$

which will span the vector space V_{2j}.

Mallat show how to construct wavelets, ψ, from these scaling functions to span the detail spaces, D_{2j}. These are defined to complement the vector spaces by capturing information lost in the transition from on vector space to the next. At all levels

$$\begin{aligned} D_{2j} &\perp V_{2j} \\ V_{2j} \oplus D_{2j} &= V_{2(j+1)} \end{aligned} \tag{2.7}$$

For the purposes of this thesis, we are interested in the representation of the signals at the various resolutions, and thus in the reconstructed versions of the signals that live in the spaces V_{2j} and D_{2j}. The Haar basis is chosen for its simplicity and computational efficiency. The reconstructions can thus be represented by vectors of sampled data points from the functions f_{2j} in V_{2j} and δf_{2j} in D_{2j}. In this thesis, and indeed in most other work on the topic, there is an abuse of notation that refers to the sampled data points in the two functions as "scaling function coefficients" and "wavelet coefficients", whereas in fact they differ by a factor of $\frac{1}{\sqrt{2}}$ from the coefficients, and have a considerably more complicated relationship if a different wavelet is used. Nevertheless, the notation is now so widespread that it is employed here.

This approach allows an approximation of the function to be represented by a discrete set of values, that collect the information in such a way that temporal and frequency based information are preserved. For the purposes of this discussion, the zeroth level contains regularly sampled data at some frequency, usually determined by physical considerations.

$$\{f^{(0)}(t)\} = \{f_{0,0}(= f^{(0)}(t_0)), f_{0,1}(= f^{(0)}(t_1)), \dots, f_{0,2k}, f_{0,2k+1}\} \tag{2.8}$$

is a sequence of discrete-time values of $f(t)$ for a given sampling interval.

The simplest example of a scaling- and wavelet function is the Haar function, [2], and corresponding wavelet [20]. These are used to generate the coarser representations of the function $f(t)$, by convolving $f^{(0)}(t)$ with $h = \frac{1}{\sqrt{2}}\left[\begin{array}{cc} 1 & 1 \end{array}\right]$(the Haar scaling filter) to obtain a coarser average, and by convolving with $g = \frac{1}{\sqrt{2}}\left[\begin{array}{cc} 1 & -1 \end{array}\right]$(the Haar wavelet filter) to obtain the information lost in the averaging scheme. This approach reduces to the Haar wavelet decomposition of $f(t)$. It produces the average and difference of a function at a given level from the average of the function at the scale below. The explicit form is given for the zeroth level.

$$f_{1,k} = \frac{1}{\sqrt{2}}(f_{0,2k} + f_{0,2k+1}) \tag{2.9}$$

$$\delta f_{1,k} = \frac{1}{\sqrt{2}}(f_{0,2k} - f_{0,2k+1}) \tag{2.10}$$

Subsequent filtering of $f_{1,k}$, leads to coarser depictions of $f(t)$. The complete Haar decomposition of $\{f(0)(t)\}$ leads to a set of coefficients, which can be indexed to correspond to the nodes of a binary tree, as shown in Figure 2-1.

With the concept of a signal on a tree clearly defined, and the means of splitting it into its frequency bands using wavelet decomposition, it is time to explore more structured signals, and the additional constraints that are imposed when one uses a model to describe the signal f. First it is necessary to introduce what is meant by a model on a tree.

2.2 Motivation for the multiscale state estimator

2.2.1 Direct use of the Kalman filter cost function on the tree

A classical model prediction control algorithm involves a system identification step which requires the solution of an optimisation problem such as the following one.

Problem 1

$$\min_{\{\hat{x}_k\}} x^{eT}_{0|-1} P^{-1}_{0|-1} x^{e}_{0|-1} + \sum_{k=0..n} \hat{v}^T_k R^{-1}_k \hat{v}_k + \hat{w}^T_k Q^{-1}_k \hat{w}_k \tag{2.11}$$

subject to

$$\begin{aligned}\hat{x}(k+1) &= A\hat{x}(k) + Bu(k) + \hat{w}(k) \\ y(k) &= C\hat{x}(k) + \hat{v}(k) \\ \underline{g} &\leq G\hat{x}(k) \leq \bar{g} \\ \underline{h} &\leq \hat{w}(k) \leq \bar{h}\end{aligned}$$

$\hat{x}(k)$ is the resulting estimate of the state and $\hat{w}(k)$ the associated modelling error or disturbance. The hat notation implies that they are both obtained from the solution to an estimation problem, as opposed to being true physical values, such as the measurement, y, and the process input, u.

Using this state estimation, one is able to lump all model uncertainty into a single parameter, $d = \hat{x}_{k|k} - \hat{x}_{k|k-1}$, for use in prediction of future state variables. This newly identified variable will be used to construct a control policy using MPC, which will be discussed in detail in a later chapter.

A richer model structure than this can be constructed by allowing a more elaborate model uncertainty structure, specifically one based on the richer state dynamics structure of the multiscale domain. If state identification takes place at multiple scales, we can obtain estimates of the errors in the models we construct at multiple resolutions, which can be used to produce a control policy based on a richer summary of our historical data.

To achieve this aim, it is necessary to obtain a deeper understanding of the issues involved in modelling at multiple scales, and in estimation and identification based on multiscale models. Although the MPC formulation has motivated much of this research, we will begin with a development of multiscale mathematical concepts that are suitable for more general applications of multiscale modelling.

Ideally, the cost function for Problem 1 would be directly applicable for multiscale problems, by a simple orthogonal transform, such as the Haar transform. Let us consider why this is not the case. Consider a reduced time based problem of the same form as Problem 1, but with

$R = Q = 1$, $P^{-1} = 0$ and no constraints on any of the variables. The problem reduces to:

$$\begin{aligned} & \min_{\{\hat{x}_k\}} \hat{v}^T \hat{v} + \hat{w}^T \hat{w} \\ \hat{v} &= y - C\hat{x} && (2.12) \\ \hat{w} &= \hat{x}_{k+1} - Ax_k - Bu_k && (2.13) \end{aligned}$$

In performing a wavelet transform, such as the Haar transform, there is an orthonormal matrix H, that produces a linear transformation from the time space to a wavelet space:

$$\begin{aligned} H\hat{x}_k &= \hat{x}_\tau && (2.14) \\ \hat{x}_k &= \{\hat{x}_0, \hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{N-1}\} && (2.15) \\ \hat{x}_{k+1} &= \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N\} && (2.16) \\ \hat{x}_\tau &= \{\hat{x}_{top}, \delta\hat{x}_{top}, \delta\hat{x}_{top\alpha}, \ldots, \delta\hat{x}_{bottom}\} && (2.17) \end{aligned}$$

Ideally we would like to produce a solution of the type

$$\begin{aligned} & \min_{\{\hat{x}_k\}} \hat{v}^T H^T H \hat{v} + \hat{w}^T H^T H \hat{w} \\ H\hat{v} &= Hy - HC\hat{x}_k && (2.18) \\ H\hat{w} &= H\hat{x}_{k+1} - HAx_k - HBu_k && (2.19) \end{aligned}$$

Consider the structure of H for the Haar transform, illustrated here for the eight point problem, or a tree of depth 3.

$$H = \begin{bmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & & & & \\ & & & & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & & & & & & \\ & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & & & & \\ & & & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & & \\ & & & & & & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{2.20}$$

The general form for trees of arbitrary depth extends the pattern with a constant top row of $\frac{1}{\sqrt{n}}$ where n is the number of points at the zeroth level of the tree, and a predictable pattern of block differences at each row, with each scale differing by a factor of $\sqrt{2}$. The matrix H follows directly from the Haar decomposition model, 2.9, and is trivially orthonormal. The measurement term produces no problems since all elements in the measurement equation are at the same node - the measurement, y, the state, x, and the error, v. The transformation is straightforward, and since H is orthonormal, the measurement terms in the time based problem sum to the same value as in the tree based problem. The modelling term, which describes the state dynamics, would be identical if we were prepared to use a Haar transform for the modelling error, yet this produces an awkward state description at higher levels. The reason for this is the term $H\hat{x}_{k+1}$ in 2.19. The technique of transforming using Haar fails since there is incompatibility between the trees being used: the transforms $H\hat{x}_k$ and $H\hat{x}_{k+1}$ map onto different multiscale spaces, and we have a non-trivial, and therefore useless, relationship between the state variables that we are trying to find.

We must thus seek an alternate formulation of the problem, or an alternate transform. An alternative is to change the cost function to include a differently weighted set of modelling errors. Our approach is to introduce the concept of dynamic systems on the tree, and demonstrate the extent of their equivalence to dynamic systems at the physical level.

2.3 Dynamic systems and noise processes on trees

2.3.1 multiscale models on trees

The concept of representation of models on binary trees was introduced in [5], [10], [13], [16]. The approach taken by the LIDS group differs significantly from ours, since they were largely concerned with independent noise processes on trees, and their evolution in time. Our approach seeks to convert physical models in time to multiscale models on a tree. This section describes the process for a first-order system.

2.3.2 Equivalence of dynamic systems on the tree and dynamic systems in time

We begin with a zeroth order dynamic system on the time line such as the measurement equation above.

$$\hat{v} = y - C\hat{x} \tag{2.21}$$

The measurement equation transforms trivially into the wavelet domain.

$$H\hat{v} = Hy - HC\hat{x} \tag{2.22}$$

The significance of this will become more clear once the first-order system has been presented.

Beginning with a first-order dynamic system in time,

$$x_{k+1} = Ax_k + Bu_k \tag{2.23}$$

Note that we can restrict our attention to u for the moment - w follows similar dynamics since it appears as a k term in the formulation, and will thus transform in the same way. We wish to apply a Haar wavelet transform to the state values, x_k, of this system, and investigate the dynamic system created at higher levels.

The Haar transform is

$$x_\tau = \frac{1}{\sqrt{2}}\left(x_{\tau\alpha} + x_{\tau\beta}\right) \tag{2.24}$$

Choosing initially $\tau\alpha = 0$, that is a point on the zeroth level, the zeroth order state equation is

$$x_1 = Ax_0 + Bu_0 \tag{2.25}$$

transforming using the Haar decomposition,

$$\begin{aligned} x_{0\gamma} &= \frac{1}{\sqrt{2}}\left(x_0 + x_1\right) \\ &= \frac{1}{\sqrt{2}}\left(x_0 + Ax_0 + Bu_0\right) && (2.26) \\ &= \frac{1}{\sqrt{2}}\left(I + A\right)x_0 + \frac{1}{\sqrt{2}}Bu_0 && (2.27) \end{aligned}$$

For $\tau\alpha = 2$ on the zeroth level,

$$\begin{aligned} x_{2\gamma} &= \frac{1}{\sqrt{2}}\left(x_2 + x_3\right) \\ &= \frac{1}{\sqrt{2}}\left(x_2 + Ax_2 + Bu_2\right) \\ &= \frac{1}{\sqrt{2}}\left(\left(I + A\right)\left(Ax_1 + Bu_1\right) + Bu_2\right) \\ &= \frac{1}{\sqrt{2}}\left(\left(I + A\right)A(Ax_0 + Bu_0) + \left(I + A\right)Bu_1 + Bu_2\right) \\ x_{2\gamma} &= A^2x_{0\gamma} + \frac{1}{\sqrt{2}}\left(ABu_0 + \left(I + A\right)Bu_1 + Bu_2\right) && (2.28) \end{aligned}$$

The geometry of this relationship is general over the entire zeroth level. This representation reveals the existence of a first-order relationship at the first level, with input relations as observed in the above. This observation allows us to define the Modified Hat Transform as follows:

$$B_{0\gamma}u_{0\gamma} = \frac{1}{\sqrt{2}}\left(ABu_0 + \left(I + A\right)Bu_1 + Bu_2\right) \tag{2.29}$$

From a first-order dynamic system with input dynamics at a given level, it is possible to construct, using the Haar decomposition, a first-order dynamic system at the level immediately above, with input dynamics that obey the Modified Hat transform. Note that this transformation is unique - the transformation of the states at the zeroth level to higher levels using the Haar transform naturally produces the Modified Hat Transform in the input variables, thus there is no benefit to seeking a simpler input transform.

It should be noted that it is possible to generate a similar first-order relationship between wavelet coefficients at the higher level using a similar construction to the above as follows.

$$\delta x_{2\gamma} = A^2 \delta x_{0\gamma} + \frac{1}{\sqrt{2}} \left(ABu_0 + (I - A) Bu_1 - Bu_2 \right) \tag{2.30}$$

thus,

$$B_{0\gamma} u^*_{0\gamma} = \frac{1}{\sqrt{2}} \left(ABu_0 + (I - A) Bu_1 - Bu_2 \right) \tag{2.31}$$

A first-order difference equation at one level of the tree gives rise to a first-order difference equation at all other levels on the tree, and further, a first-order difference equation on one level gives rise to a first-order difference equation between two adjacent levels on the tree.

In a similar way, an nth order system at the physical level can be transformed into an nth order system at any higher level using the Haar transform, where the input variable will be defined by a generalised modified hat transform of support $2n + 1$ nodes. Likewise, one can construct nth order difference equations up the tree - difference equations linking connected nodes at $n + 1$ adjacent levels of the tree.

A Haar wavelet transform is applied to the state variables, from which the following two-scale model [12] can be created for any node, τ.

$$\begin{aligned} x_\tau &= \frac{1}{\sqrt{2}} \left(x_{\tau\alpha} + x_{\tau\beta} \right) && (2.32) \\ &= \frac{1}{\sqrt{2}} \left(I + A \right) x_{\tau\alpha} + \frac{1}{\sqrt{2}} B u_{\tau\alpha} && (2.33) \end{aligned}$$

or equivalently, by creating and comparing wavelet and scaling function decompositions,

$$\begin{aligned}
\delta x_\tau &= \frac{1}{\sqrt{2}}\left(x_{\tau\alpha} - x_{\tau\beta}\right) && (2.34)\\
&= \frac{1}{\sqrt{2}}\left(I - A\right) x_{\tau\alpha} - \frac{1}{\sqrt{2}} B u_{\tau\alpha} && (2.35)\\
&= \left(I + A\right)^{-1}\left(I - A\right) x_\tau - \sqrt{2}\left(I + A\right)^{-1} B u_{\tau\alpha} && (2.36)
\end{aligned}$$

Since the dynamic system at higher levels has the same dynamics as the zeroth level dynamic system, similar first-order models can be generated recursively at all higher levels. In particular, for any left-node at level-m, the following are the multiscale counterparts of 2.33 and 2.36.

$$\begin{aligned}
x_{\tau\alpha} &= \sqrt{2}\left(I + A^{(m)}\right)^{-1} x_\tau - \sqrt{2}\left(I + A^{(m)}\right)^{-1} B^{(m)} u_{\tau\alpha} && (2.37)\\
\delta x_\tau &= \left(I + A^{(m)}\right)^{-1}\left(I - A^{(m)}\right) x_\tau - \sqrt{2} B^{(m)}\left(I + A^{(m)}\right)^{-1} u_{\tau\alpha} && (2.38)
\end{aligned}$$

where $A^{(m)}$, or A_τ , is A^{2^m} and m refers to the scale.

Thus, a discrete-time model, given by equation 2.23, describing process behaviour over $2N$ points, transforms to a multiscale model, given by equation 2.37 or 2.38, defined over $2N$ left-nodes, τ, of a binary tree (with N levels, excluding the level of the finest resolution). Each node of the binary tree describes process behaviour over localised sections of time and scale (range of frequencies). Importantly, the information contained in the two representations is identical.

In this section, no distinction has been made between a control input and a modelling uncertainty, because their form in the state equation is identical, the two terms can be treated in the same way, and the modified hat transform is appropriate for both.

2.3.3 Different forms of the first-order system between adjacent levels

There are a number of ways to adapt the cost function to create something more computationally tractable. A first step is to recognise that even though the first-order system at a given level gives rise to a first-order system at other levels, and a separate first-order system between two adjacent levels, there is more than one such first-order system. Clearly some of the relationships below are trivial, but all of the useful ones can be found - a relationship between a state and its left descendant state, between a state and its right descendant state, between a state and its wavelet coefficient, and between the wavelet states and either of the descendant states. As shown above, the existence of a zeroth level dynamic system implies the same dynamic system at higher levels, with predictable parametric modifications, thus the equations below are general.

The following discussion demonstrates that a first-order dynamic system can be constructed by choosing any two from the set of the two adjacent points, their scaling function coefficient and their wavelet coefficient, and relating them to the input on the left branch of the tree.

Given a first-order multiscale dynamic system,

$$x_{\tau\beta} = Ax_{\tau\alpha} + Bu_{\tau\alpha} \tag{2.39}$$

find suitable coefficients to satisfy for any choice of scalars a, b, c, and, d,

$$\begin{aligned} ax_{\tau\alpha} + bx_{\tau\beta} &= \theta\left(cx_{\tau\alpha} + dx_{\tau\beta}\right) + \psi u_{\tau\alpha} \\ ax_{\tau\alpha} + b\left(Ax_{\tau\alpha} + Bu_{\tau\alpha}\right) &= \theta\left(cx_{\tau\alpha} + d\left(Ax_{\tau\alpha} + Bu_{\tau\alpha}\right)\right) + \psi u_{\tau\alpha} \\ \left(a + bA - \theta c - \theta dA\right)x_{\tau\alpha} &= \left(\theta dB + \psi - bB\right)u_{\tau\alpha} \qquad (2.40) \end{aligned}$$

$x_{\tau\alpha}$ and $u_{\tau\alpha}$ are independent, so for complete generality, the two quantities within parentheses must be zero, thus

$$\theta = \left(c + dA\right)^{-1}\left(a + bA\right) \tag{2.41}$$

$$\psi = \left(c + dA\right)^{-1} B\left(bc - ad\right) \tag{2.42}$$

The first-order description thus has many different forms represented by the following equation, where any element of the first brace can be used in combination with any element of the second:

$$\left\{\begin{array}{c} x_{\tau\alpha} \\ x_{\tau\beta} \\ x_{\tau} \\ \delta x_{\tau} \end{array}\right\} = \theta\left\{\begin{array}{c} x_{\tau\alpha} \\ x_{\tau\beta} \\ x_{\tau} \\ \delta x_{\tau} \end{array}\right\} + \psi u_{\tau\alpha} \tag{2.43}$$

The following tables show the explicit values for the 16 possible cases of θ and ψ. The

	c	d	$x_{\tau\alpha}$	$x_{\tau\beta}$	x_τ	δx_τ
a			1	0	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$
b			0	1	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$
$x_{\tau\alpha}$	1	0	1	A	$\frac{1}{\sqrt{2}}(I+A)$	$\frac{1}{\sqrt{2}}(I-A)$
$x_{\tau\beta}$	0	1	A^{-1}	1	$\frac{1}{\sqrt{2}}(I+A)A^{-1}$	$\frac{1}{\sqrt{2}}(I-A)A^{-1}$
x_τ	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$\sqrt{2}(I+A)^{-1}$	$\sqrt{2}(I+A)^{-1}A$	1	$(I+A)^{-1}(I-A)$
δx_τ	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$\sqrt{2}(I-A)^{-1}$	$-\sqrt{2}(I-A)^{-1}A$	$(I-A)^{-1}(I+A)$	1

Table 2.1: Table of Phi values for All First Order Equations

	c	d	$x_{\tau\alpha}$	$x_{\tau\beta}$	x_τ	δx_τ
a			1	0	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$
b			0	1	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$
$x_{\tau\alpha}$	1	0	0	B	$\frac{1}{\sqrt{2}}B$	$-\frac{1}{\sqrt{2}}B$
$x_{\tau\beta}$	0	1	$-A^{-1}B$	0	$-\frac{1}{\sqrt{2}}A^{-1}B$	$-\frac{1}{\sqrt{2}}A^{-1}B$
x_τ	$\frac{1}{\sqrt{2}}$	$\frac{1}{\sqrt{2}}$	$-(I+A)^{-1}B$	$(I+A)^{-1}B$	0	$-\frac{1}{\sqrt{2}}(I+A)^{-1}B$
δx_τ	$\frac{1}{\sqrt{2}}$	$-\frac{1}{\sqrt{2}}$	$(I-A)^{-1}B$	$(I-A)^{-1}B$	$\frac{1}{\sqrt{2}}(I-A)^{-1}B$	0

Table 2.2: Table of Psi values for All First Order Equations

leftside of the equation runs along the top row, the right side down the first column, the values of a, b, c and d are listed, and finally in the bottom right quadrant the parameter values are listed.

The utility of this formulation is that it allows us flexibility in constructing the cost function. We can choose to define our model uncertainty in terms of the wavelet formulation, the scaling function formulation, or some other way related to the two. The importance of this choice becomes clear if one tries to include both definitions of model uncertainty in the cost functions, and at some point one has to decide which it is to be.

Note though, that the modelling error on the right leg of the tree, $u_{\tau\beta}$, has no definition in terms of a first-order dynamic system on the tree, and is defined in terms of the modified hat transform between one higher and two adjacent nodes. This raises the question of whether minimising a set of modelling errors on the tree is equivalent to minimising modelling errors at the time line. Instead of producing an equally weighted sum of the squares as in the time line formulation, the modified hat transform gives rise to an unequally weighted sum of squares, where the weights depend on the magnitude of A, and the number of levels. To zeroth order, the ratio of the magnitude of the largest and smallest weight is $1/N$ where N is the number of points at the zeroth level. This can be reduced to $1/log_2 N$ by weighting the modelling error at each level by a factor of $2^{level/2}$, where *level* is an integer representing the number of levels above the zeroth level. This is intuitively comforting, since this factor enhances the importance in the cost function, of the coarse layers that include the most averaging, and where measurements should be more reliable.

2.3.4 Correlation structure of the Haar transform

The independent white noise process on the tree produces simple solutions due to its simple correlation structure.

Claim 2 *A zero mean, normally distributed, white noise process of known covariance at the zeroth level*

$$w = [w_0, w_1, \dots, w_N] \tag{2.44}$$

$$E[w] = 0 \tag{2.45}$$

$$E\left[ww^T\right] = IR_0 \tag{2.46}$$

can be converted using the Haar transform to

$$\delta w = [w_{top}, \delta w_{top}, \dots, \delta w_{bottom}] \tag{2.47}$$

$$E[\delta w] = 0 \tag{2.48}$$

$$E\left[\delta w \delta w^T\right] = IR_0 \tag{2.49}$$

Proof. *By level to level construction.*
At the first level,

$$\begin{aligned} E[w_\tau] &= E\left[\frac{1}{\sqrt{2}}(w_{\tau\alpha} + w_{\tau\beta})\right] \\ &= \frac{1}{\sqrt{2}}\left(E[w_{\tau\alpha}] + E[w_{\tau\beta}]\right) \\ &= 0 \end{aligned}$$

from 2.45.

Similarly

$$E\left[\delta w_\tau\right] = 0$$

Further

$$\begin{aligned} E\left[w_\tau w_\tau^T\right] &= E\left[\frac{1}{\sqrt{2}}\left(w_{\tau\alpha}+w_{\tau\beta}\right)\frac{1}{\sqrt{2}}\left(w_{\tau\alpha}+w_{\tau\beta}\right)^T\right] \\ &= \frac{1}{2}\left(E\left[w_{\tau\alpha}w_{\tau\alpha}^T\right]+E\left[w_{\tau\alpha}w_{\tau\beta}^T\right]+E\left[w_{\tau\beta}w_{\tau\alpha}^T\right]+E\left[w_{\tau\beta}w_{\tau\beta}^T\right]\right) \\ &= \frac{1}{2}\left(IR_0+0+0+IR_0\right) \\ &= IR_0 \end{aligned} \tag{2.50}$$

Even further

$$\begin{aligned} E\left[\delta w_\tau \delta w_\tau^T\right] &= E\left[\frac{1}{\sqrt{2}}\left(w_{\tau\alpha}-w_{\tau\beta}\right)\frac{1}{\sqrt{2}}\left(w_{\tau\alpha}-w_{\tau\beta}\right)^T\right] \\ &= \frac{1}{2}\left(E\left[w_{\tau\alpha}w_{\tau\alpha}^T\right]-E\left[w_{\tau\alpha}w_{\tau\beta}^T\right]-E\left[w_{\tau\beta}w_{\tau\alpha}^T\right]+E\left[w_{\tau\beta}w_{\tau\beta}^T\right]\right) \\ &= \frac{1}{2}\left(IR_0-0-0+IR_0\right) \\ &= IR_0 \end{aligned} \tag{2.51}$$

And even further

$$\begin{aligned} E\left[w_\tau \delta w_\tau^T\right] &= E\left[\frac{1}{\sqrt{2}}\left(w_{\tau\alpha}+w_{\tau\beta}\right)\frac{1}{\sqrt{2}}\left(w_{\tau\alpha}-w_{\tau\beta}\right)^T\right] \\ &= \frac{1}{2}\left(E\left[w_{\tau\alpha}w_{\tau\alpha}^T\right]-E\left[w_{\tau\alpha}w_{\tau\beta}^T\right]+E\left[w_{\tau\beta}w_{\tau\alpha}^T\right]-E\left[w_{\tau\beta}w_{\tau\beta}^T\right]\right) \\ &= \frac{1}{2}\left(IR_0-0+0-IR_0\right) \\ &= 0 \end{aligned} \tag{2.52}$$

At this point, all levels are indistinguishable, since $\{w_\tau\}$ at the first level has the same statistics as the zeroth level, and thus the level to level transformations between higher levels will produce similar results.

It remains to show that

$$\begin{aligned} E\left[\delta w_{\tau\alpha}\delta w_{\tau}^{T}\right] &= E\left[\frac{1}{\sqrt{2}}\delta w_{\tau\alpha}\left(w_{\tau\alpha}+w_{\tau\beta}\right)^{T}\right] \\ &= \frac{1}{\sqrt{2}}\left(E\left[\delta w_{\tau\alpha}w_{\tau\alpha}^{T}\right]+E\left[\delta w_{\tau\alpha}w_{\tau\beta}^{T}\right]\right) \\ &= \frac{1}{\sqrt{2}}\left(0+0\right) \\ &= 0 \end{aligned} \tag{2.53}$$

A similar argument completes the proof for all levels. ■

2.3.5 The correlation structure of the modified hat transform

Consider instead the modified hat transform and the expected values due to the new transform. A constant state dynamic system has been assumed, construction of the general case is trivial from the proof below, but offers few simplifications.

Claim 3 *Suppose at the zeroth level,*

$$\begin{aligned} E\left[w\right] &= 0 \\ E\left[ww^{T}\right] &= IR_0 \end{aligned}$$

(by standard abuse of notation, this may refer to a block diagonal matrix with matrix R_0 repeated down the diagonal, and similarly for the remainder of the matrices)

At all higher levels, k.

$$\begin{aligned} E\left[w_{\tau}\right] &= 0 \\ E\left[w_{\tau}w_{\tau}^{T}\right] &= IR_k \\ E\left[w_{\tau+1}w_{\tau}^{T}\right] &= IU_k \\ E\left[w_{\tau}w_{\tau+1}^{T}\right] &= IL_k \\ E\left[w_{\tau}w_{\tau+p}^{T}\right] &= 0 \quad \forall\left|p\right|\geq 2 \end{aligned}$$

where R_k, L_k, U_k are linked by the recursion described below.

Proof. For $k = 0$ this is true by construction with $R_k = R_0$, $U_k = 0$, and, $L_k = 0$. Thus assume it is true for any level.

For τ any higher level, let us begin with the first-order statistics.

$$\begin{aligned}
w_\tau &= \frac{1}{\sqrt{2}}\left(a_{\tau\alpha} w_{\tau\alpha} + (1 + a_{\tau\alpha})\, w_{\tau\beta} + w_{\tau\beta+1}\right) \\
E\left[w_\tau\right] &= \frac{1}{\sqrt{2}}\left(a_{\tau\alpha} E\left[w_{\tau\alpha}\right] + (1 + a_{\tau\alpha})\, E\left[w_{\tau\beta}\right] + E\left[w_{\tau\beta+1}\right]\right) \\
&= 0 \qquad (2.54)
\end{aligned}$$

A similar argument holds for the first-order statistics at higher levels. The second order statistics are more complicated. Again assume it is true at level k, and demonstrate that it is true at level $k+1$.

$$\begin{aligned}
w_\tau &= \frac{1}{\sqrt{2}}\left(a_{\tau\alpha} w_{\tau\alpha} + (1 + a_{\tau\alpha})\, w_{\tau\beta} + w_{\tau\beta+1}\right) \\
R_{k+1} &= E\left[w_\tau w_\tau^T\right] \\
E\left[w_\tau w_\tau^T\right] &= \frac{1}{2}\left(a_{\tau\alpha} E\left[w_{\tau\alpha} w_{\tau\alpha}^T\right] a_{\tau\alpha}^T\right) \\
&+\frac{1}{2}\left((1 + a_{\tau\alpha})\, E\left[w_{\tau\alpha} w_{\tau\alpha}^T\right] (1 + a_{\tau\alpha})^T + E\left[w_{\tau\alpha} w_{\tau\alpha}^T\right]\right) \\
&+\frac{1}{2}\left(a_{\tau\alpha} E\left[w_{\tau\alpha} w_{\tau\alpha+1}^T\right] (1 + a_{\tau\alpha})^T + (1 + a_{\tau\alpha})\, E\left[w_{\tau\alpha} w_{\tau\alpha+1}^T\right]\right) \\
&+\frac{1}{2}\left((1 + a_{\tau\alpha})\, E\left[w_{\tau\alpha+1} w_{\tau\alpha}^T\right] a_{\tau\alpha}^T + E\left[w_{\tau\alpha+1} w_{\tau\alpha}^T\right] (1 + a_{\tau\alpha})^T\right) \\
&+\frac{1}{2} a_{\tau\alpha} E\left[w_{\tau\alpha} w_{\tau\alpha+2}^T\right] + \frac{1}{2} E\left[w_{\tau\alpha+2} w_{\tau\alpha}^T\right] a_{\tau\alpha}^T \qquad (2.55) \\
&= \frac{1}{2}\left(a_{\tau\alpha} R_k a_{\tau\alpha}^T + (1 + a_{\tau\alpha})\, R_k\, (1 + a_{\tau\alpha})^T + R_k\right) \\
&+\frac{1}{2}\left(a_{\tau\alpha} L_k\, (1 + a_{\tau\alpha})^T + (1 + a_{\tau\alpha})\, L_k\right) \\
&+\frac{1}{2}\left((1 + a_{\tau\alpha})\, U_k a_{\tau\alpha}^T + U_k\, (1 + a_{\tau\alpha})^T\right) \qquad (2.56)
\end{aligned}$$

For the off diagonal terms,

$$\begin{aligned}
w_\tau &= \frac{1}{\sqrt{2}}\left(a_{\tau\alpha}w_{\tau\alpha} + \left(1+a_{\tau\alpha}\right)w_{\tau\beta} + w_{\tau\beta+1}\right)\\
L_{k+1} &= E\left[w_\tau w_{\tau+1}^T\right]\\
E\left[w_\tau w_{\tau+1}^T\right] &= \frac{1}{2}\left(a_{\tau\alpha}E\left[w_{\tau\alpha}w_{\tau\alpha+2}^T\right]a_{\tau\alpha}^T + \left(1+a_{\tau\alpha}\right)E\left[w_{\tau\alpha}w_{\tau\alpha+2}^T\right]\left(1+a_{\tau\alpha}\right)^T\right)\\
&\quad+\frac{1}{2}\left(E\left[w_{\tau\alpha}w_{\tau\alpha+2}^T\right] + a_{\tau\alpha}E\left[w_{\tau\alpha}w_{\tau\alpha+3}^T\right]\left(1+a_{\tau\alpha}\right)^T + \left(1+a_{\tau\alpha}\right)E\left[w_{\tau\alpha}w_{\tau\alpha+3}^T\right]\right)\\
&\quad+\frac{1}{2}\left(\left(1+a_{\tau\alpha}\right)E\left[w_{\tau\alpha+1}w_{\tau\alpha+2}^T\right]a_{\tau\alpha}^T + E\left[w_{\tau\alpha+1}w_{\tau\alpha+2}^T\right]\left(1+a_{\tau\alpha}\right)^T\right)\\
&\quad+\frac{1}{2}a_{\tau\alpha}E\left[w_{\tau\alpha}w_{\tau\alpha+4}^T\right] + \frac{1}{2}E\left[w_{\tau\alpha+2}w_{\tau\alpha+2}^T\right]a_{\tau\alpha}^T \qquad (2.57)\\
&= \frac{1}{2}\left(\left(1+a_{\tau\alpha}\right)L_k a_{\tau\alpha}^T + L_k\left(1+a_{\tau\alpha}\right)^T\right) + \frac{1}{2}R_k a_{\tau\alpha}^T \qquad (2.58)
\end{aligned}$$

Similarly for the upper diagonal terms.

$$\begin{aligned}
U_{k+1} &= E\left[w_{\tau+1}w_\tau^T\right]\\
&= \frac{1}{2}\left(a_{\tau\alpha}U_k\left(1+a_{\tau\alpha}\right)^T + \left(1+a_{\tau\alpha}\right)U_k\right) + \frac{1}{2}a_{\tau\alpha}R_k \qquad (2.59)
\end{aligned}$$

And for completion

$$\begin{aligned}
E\left[w_\tau w_{\tau+2}^T\right] &= \frac{1}{2}\left(a_{\tau\alpha}E\left[w_{\tau\alpha}w_{\tau\alpha+4}^T\right]a_{\tau\alpha}^T\right)\\
&\quad+\frac{1}{2}\left(\left(1+a_{\tau\alpha}\right)E\left[w_{\tau\alpha}w_{\tau\alpha+4}^T\right]\left(1+a_{\tau\alpha}\right)^T + E\left[w_{\tau\alpha}w_{\tau\alpha+4}^T\right]\right)\\
&\quad+\frac{1}{2}\left(a_{\tau\alpha}E\left[w_{\tau\alpha}w_{\tau\alpha+5}^T\right]\left(1+a_{\tau\alpha}\right)^T + \left(1+a_{\tau\alpha}\right)E\left[w_{\tau\alpha}w_{\tau\alpha+5}^T\right]\right)\\
&\quad+\frac{1}{2}\left(\left(1+a_{\tau\alpha}\right)E\left[w_{\tau\alpha+1}w_{\tau\alpha+4}^T\right]a_{\tau\alpha}^T + E\left[w_{\tau\alpha+1}w_{\tau\alpha+4}^T\right]\left(1+a_{\tau\alpha}\right)^T\right)\\
&\quad+\frac{1}{2}a_{\tau\alpha}E\left[w_{\tau\alpha}w_{\tau\alpha+6}^T\right] + \frac{1}{2}E\left[w_{\tau\alpha+2}w_{\tau\alpha+4}^T\right]a_{\tau\alpha}^T \qquad (2.60)\\
&= 0 \text{ by the induction assumption.} \qquad (2.61)
\end{aligned}$$

A similar argument holds for all other distances from the reference point, which completes the proof with the following recursion

$$\begin{aligned}
R_{k+1} &= \frac{1}{2}\left(a_{\tau\alpha}R_k a_{\tau\alpha}^T + (1+a_{\tau\alpha})\,R_k\,(1+a_{\tau\alpha})^T + R_k\right) && (2.62)\\
&+\frac{1}{2}\left(a_{\tau\alpha}L_k\,(1+a_{\tau\alpha})^T + (1+a_{\tau\alpha})\,L_k\right) && (2.63)\\
&+\frac{1}{2}\left((1+a_{\tau\alpha})\,U_k a_{\tau\alpha}^T + U_k\,(1+a_{\tau\alpha})^T\right) && (2.64)\\
L_{k+1} &= \frac{1}{2}\left((1+a_{\tau\alpha})\,L_k a_{\tau\alpha}^T + L_k\,(1+a_{\tau\alpha})^T\right) + \frac{1}{2}R_k a_{\tau\alpha}^T && (2.65)\\
U_{k+1} &= \frac{1}{2}\left(a_{\tau\alpha}U_k\,(1+a_{\tau\alpha})^T + (1+a_{\tau\alpha})\,U_k\right) + \frac{1}{2}a_{\tau\alpha}R_k && (2.66)
\end{aligned}$$

Further, it follows that

$$L_k = U_k^T$$

Note that for symmetric A, $L_{k+1} = U_{k+1}$ ■

2.3.6 Perfect reconstruction using the modified hat transform

While the modified hat transform does not produce an orthonormal transform, the transform can be performed in $O(N)$ steps where N is the number of points. It is not parallelisable, since at any level, all points must be solved in sequence, and rely on the solution of one of the adjacent points. It does, however, produce perfect reconstruction, given the right-most boundary condition, for which we generally assume zero - the expected value of all future values of model uncertainty - in the absence of conflicting information.

2.4 The multiscale state estimation algorithm

The goal of the state estimation algorithm is to produce an optimal state estimate from a set of measurements. The measurements, y, are of some process described by a dynamic system $\frac{dx}{dt} = f(x, u, w)$, where x are the states, u are some known inputs, and w are used to describe any unknown inputs, and unknown dynamics in the process. The measurements y are governed by a measurement equation $y = g(x, v)$, where v represents the uncertainty in the measurement process. This information is used to generate a set of optimal state estimates $\hat{x}$.

The purpose of the multiscale state estimator is to examine the states, state dynamics, the measurements, and the uncertainties in all of these, and to project them onto a multiscale tree so that meaningful state estimates are produced, either in a more efficient way than existing time based algorithms, or in a way that increases the utility of the information.

We begin with the derivation of the multiscale state estimation algorithm for the first-order autoregressive process with model uncertainty, and driven by an input.

$$x(t+1) = Ax(t) + Bu(t) + w(t) \tag{2.67}$$

This is the first-order process on the time line that will be used to generate measurements.

2.4.1 Multiscale processes on trees

Processes of this form can be projected onto multiscale trees to produce a number of equivalent functional forms on the tree as described in Section 2.3.3. For its simplicity and efficiency of solution, choose the functional form relating the wavelet coefficients and the scaling function at the same node.

$$(I + A_{\tau\alpha})\, \delta x_\tau = (I - A_{\tau\alpha})\, x_\tau + \sqrt{2} B u_{\tau\alpha} + \sqrt{2} w_{\tau\alpha} \tag{2.68}$$

2.4.2 The basis set of states on the tree

The states to be estimated are the set of Haar wavelet coefficients of the states at the zeroth level, $\delta\hat{x}_\tau$, and the top scaling function coefficient, $\hat{x}_{top}$. It is convenient to use the other scaling coefficients throughout the algorithm, since they provide a convenient shorthand in the recursion. With the exception of the top node, the scaling function coefficients are not part of the basis set, since they are a linear combination of the wavelet coefficients and the top scaling function coefficient. The dimension of the basis in the time- and tree-based problems is consistent.

2.4.3 Inputs on the tree

The $u_{\tau\alpha}$ are the input values produced through the modified hat transform of the zeroth level inputs, u_t, consistent with the selection of the dynamic system on the tree. The modelling uncertainty variables, w_τ, will also be governed by the modified Hat transform. Since the inputs, u_t, are known *a priori*, their transform can be computed as part of the setup of the optimisation problem.

The dynamic system matrix A_τ changes from level to level, using the recursion from scale to scale

$$A_\tau = A_{\tau\alpha} \cdot A_{\tau\beta} \tag{2.69}$$

2.4.4 Measurements on the tree

The measurement model on the time line is assumed to take the form

$$y(t) = Cx(t) + v(t) \tag{2.70}$$

for some known measurement matrix, C. The measurements, y_t, are also propagated onto the tree, along with the states, to produce a tree-based representation of the variables in time and scale.

$$y_\tau = Cx_\tau + v_\tau \tag{2.71}$$

with measurement uncertainty variable, v_τ. The measurements and states are transformed using the Haar wavelet decomposition, and from the linearity of the Haar transform operator, it follows that the measurement uncertainties will be governed by the Haar wavelet decomposition.

2.4.5 Properties of the various uncertainties

The cost function to optimise state estimation will be the minimum of a selected norm of the various uncertainties. A simple algorithm results from the two-norm, of the measurement error, weighted by its covariance. The covariance requires computation, described in Section 2.3.4.

Since one selects a covariance from experience, or past data, the recursion on the second order statistics can be performed off line, or as part of the start up computations.

The off-diagonal elements of the covariance matrix are needed to generate the covariance matrix recursively, at each level, but they are not used in the cost function itself, since we are concerned about minimising the sum of the squares of the modelling uncertainties, and not the value of the cross terms. The choice of modelling error representation is where the cost functions for the tree- and time-based algorithms differ. The two representations with their different cost functions will necessarily give different optimal solutions when fed data containing any uncertainty. Neither solution is necessarily better, the two solutions simply reflect the somewhat arbitrary nature of the choice of error modelling. While the time based estimation algorithm weights the uncertainties equally in time, the tree based algorithm attempts to weight them evenly over time and scale, and this will be dictated by the choice of averaging scheme.

2.4.6 The cost function

Equipped with the necessary dynamic system and second order statistics, the cost function can be constructed. The cost function, Φ, has the following general form:

$$\begin{aligned}\Phi \;=\; & \sum_{\theta\epsilon M}((y^{\theta}_{top} - C\hat{x}_{top})^T R^{\theta-1}_{top}(y^{\theta}_{top} - C\hat{x}_{top}) + \sum_{\tau\epsilon T}(\delta y^{\theta}_{\tau} - C\delta\hat{x}_{\tau})^T R^{\theta-1}_{\tau}(\delta y^{\theta}_{\tau} - C\delta\hat{x}_{\tau})) + \\ & + \sum_{\tau\epsilon T}((1+A_{\tau\alpha})\delta\hat{x}_{\tau} - (1-A_{\tau\alpha})\hat{x}_{\tau} + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})^T(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_{\tau}(\sqrt{2}B_{\tau\alpha})^{-1}\ldots \\ & \ldots((1+A_{\tau\alpha})\delta\hat{x}_{\tau} - (1-A_{\tau\alpha})\hat{x}_{\tau} + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha}) \qquad (2.72)\end{aligned}$$

Features of the cost function

The number of measurement sets is arbitrary. On a first reading, it is recommended that the reader ignore the superscript θ, and consider the case of a single set of regularly sampled measurements, y. There will be no summation over θ, but the remainder of the algorithm will be identical.

The more general cost function is constructed to receive multiple sets of measurements. An example of this might be a regular set of concentration measurements inferred from an online pH measurement, coupled with a less regular set of concentration measurements from

GC analysis. Each set of measurements is indexed by a θ, and should be equipped with an estimate of its covariance, R^{θ}. The superset M contains the indices for the sets, in our example, $M = \{pH, GC\}$. The sets maybe taken using different measurement techniques - as in our example, at different sampling rates. Each set is subject to the Haar transform with y_{top} representing the wavelet coefficients of the measurement sets. Any number of measurement sets can be incorporated into the algorithm. The optimal sensor fusion problem is solved by using one of these indices for each measurement set, as will be shown in the case studies section.

The first two summation terms in the cost function represent the inclusion of the various sets of measurements. The first term contains the scaling function at the top node for each set of measurements, while the second is the contribution from the wavelet coefficients. Each measurement error is weighted by a term R^{-1} which is typically chosen to be the covariance of the measurement error. Further uses are possible - if one wished to suppress contributions to measurement error within a certain frequency band, one could adjust the R^{-1} values for those nodes accordingly. Sets of measurements with missing or spurious information are easily taken care of with an infinite uncertainty on the measurement.

The number of state variables and inputs is arbitrary. The algorithm is designed for any number of state variables and has been written in vector notation. The MATLAB coded version of the algorithm given in the appendix allows for states of any specified size. The specification is done early in the program along with the choice of dynamic system matrix and input vectors. Since B is specified independently, and u is either specified by the user, or obtained from measurements of the input variables, the algorithm allows complete specification of input vectors of arbitrary size. The only requirement is matrices of consistent dimension.

The first-order dynamic system is arbitrary. The solution algorithm is written to accept any matrix A as the state transition matrix, and all higher level versions of A_{τ} are computed as part of the initial calculations of the algorithm. The A_{τ} is never directly inverted, eliminating the restriction that it should be non-singular, however $a_{\tau}^{\dagger}$, a subsidiary variable derived from A_{τ}, is inverted frequently during the algorithm, thus it is possible to construct pathological dynamic systems that will cause the algorithm to fail. This has yet to be a problem with any of the case studies performed to date.

Polynomial dynamic systems of arbitrary order can be used. For any function $x(t) = a_1x(t-1)+\cdots+a_nx(t-n)+bu(t-1)+w(t-1)$, one can use the algorithm in its current form, by state augmentation. This dramatically increases the range of dynamic systems suitable for study.

Since B is specified independently, and u is either specified by the user, or obtained from measurements of the input variables, the algorithm allows complete specification of input vectors of arbitrary size.

Independent weighting of the measurements, modelling errors or variables at different levels is possible. This would be useful if one wished to produce a state estimate that suppressed high frequency fluctuations in the state, or if one wished to suppress the impact of measurements taken at some resolution.

2.4.7 Derivation of recursive system for multiscale optimisation on the tree for measurements at multiple scales

The derivation presented here corresponds to a fairly complicated version of the problem. A simpler first reading can be obtained by ignoring the summation over θ, the multiple sets of measurements, and the inputs, $u_{\tau\alpha}$, so that we view the undriven case. The derivation begins by posing a cost function. This leads to the necessary and sufficient conditions for optimality, which are a set of linear equations in the basis variables. One equation corresponds to each node above the zeroth level on the tree, with one extra equation for the scaling function state at the top of the tree.

These equations are then manipulated to produce a recursive solution strategy. The elimination is performed from the bottom of the tree upwards, level by level. In the presentation below, this is done explicitly for the first three levels to demonstrate the origin of the subsidiary variables d^*_τ, d'_τ, and $d^\dagger_\tau$, and the corresponding intermediates for a, b, and c.

Although the scaling function coefficients are not part of the basis, it is convenient to carry them as subsidiary variables, and use a cost function where the dynamic system on the tree relates the scaling function and wavelet coefficients at the same node. This provides a more compact algorithm by reducing the number of derivative terms, and hence the computational complexity.

Define L_m to be the collection of nodes, τ, at level m of the tree, with $m = 0$ the lowest, zeroth, or physical level of the tree, where the original dynamic system is defined. Let the collection of all nodes, τ on the tree be T. α refers to a down-shift to the left, and β refers to a down-shift to the right.

The multiscale state estimation problem with measurements at multiple scales has a cost function of the following form.

$$\min_{\{\hat{x}_{top},\delta\hat{x}_\tau\}} \sum_{\theta \epsilon M}((y^\theta_{top} - C\hat{x}_{top})^T R^{\theta-1}_{top}(y^\theta_{top} - C\hat{x}_{top}) + \sum_{\tau\epsilon T}(\delta y^\theta_\tau - C\delta\hat{x}_\tau)^T R^{\theta-1}_\tau(\delta y^\theta_\tau - C\delta\hat{x}_\tau)) +$$
$$+\sum_{\tau\epsilon T}((1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})^T(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_\tau(\sqrt{2}B_{\tau\alpha})^{-1}\ldots$$
$$\ldots((1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha}) \tag{2.73}$$

Subject to the Haar wavelet decomposition:

$$\delta\hat{x}_\tau = \frac{1}{\sqrt{2}}(\hat{x}_{\tau\alpha} - \hat{x}_{\tau\beta}) \tag{2.74}$$
$$\hat{x}_\tau = \frac{1}{\sqrt{2}}(\hat{x}_{\tau\alpha} + \hat{x}_{\tau\beta}) \tag{2.75}$$

The derivative conditions

For the unconstrained problem, the necessary and sufficient conditions for optimality are the normal equations - the derivatives of the cost function with respect to each basis variable should be zero. This should be true for the set of $\hat{x}_{top}, \delta\hat{x}_\tau$.

$$\frac{d\Phi}{d\hat{x}_{top}} = 0 \tag{2.76}$$
$$\frac{d\Phi}{d\delta\hat{x}_\tau} = 0 \; \forall\tau\epsilon T \tag{2.77}$$

For $\hat{x}_{top}$, this equation reduces to:

$$\frac{d\Phi}{d\hat{x}_{top}} = -2\sum_{\theta\epsilon M} C^T R^{\theta-1}_{top}(y^\theta_{top} - C\hat{x}_{top}) + \sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\hat{x}_{top}} \tag{2.78}$$
$$= 0 \tag{2.79}$$

For $\delta\hat{x}_\tau$:

$$\begin{aligned}\frac{d\Phi}{d\delta\hat{x}_\tau} &= -2\sum_{\theta\epsilon M} C^T R_\tau^{\theta-1}(\delta y_\tau^\theta - C\delta\hat{x}_\tau) + \sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} \qquad (2.80)\\ &\quad +2(1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q_\tau^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}((1+A_{\tau\alpha})\delta\hat{x}_\tau...\\ &\quad -(1-A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})\\ &= 0 \qquad (2.81)\end{aligned}$$

$\sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau}$ refers to contributions from scaling function terms to the cost function. Since scaling functions are composite functions of the wavelet coefficients in the basis, their contribution to the above equations must be computed using the chain rule.

The scaling function terms contribute terms of the form:

$$\begin{aligned}\frac{\partial\Phi}{\partial\hat{x}_\tau} &= -2(1-A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q_\tau^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}((1+A_{\tau\alpha})\delta\hat{x}_\tau... \qquad (2.82)\\ &\quad -(1-A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})\\ \frac{d\Phi}{d\delta\hat{x}_\tau} &= \frac{\partial\Phi}{\partial\delta\hat{x}_\tau} + \sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} \qquad (2.83)\end{aligned}$$

A recursive application of Haar reconstruction produces the following:

$$\begin{aligned}\frac{\partial\hat{x_\sigma}}{\partial\delta\hat{x}_\tau} &= (\sqrt{2})^{-m(\sigma,\tau)} \text{ if } \sigma\epsilon T\alpha+ \qquad (2.84)\\ \frac{\partial\hat{x_\sigma}}{\partial\delta\hat{x}_\tau} &= -(\sqrt{2})^{-m(\sigma,\tau)} \text{ if } \sigma\epsilon T\beta+ \qquad (2.85)\\ \frac{\partial\hat{x_\sigma}}{\partial\delta\hat{x}_\tau} &= 0 \text{ otherwise} \qquad (2.86)\end{aligned}$$

$T\alpha+$ refers to the collection of $\tau\alpha$ and all nodes that descend from it on the tree, and $m(\sigma,\tau)$ is the number of levels between τ and σ. Also $m(\sigma,\tau) = m(\sigma,\tau\alpha)+1$ if $\sigma\epsilon T\alpha+$ and $m(\sigma,\tau) = m(\sigma,\tau\beta)+1$ if $\sigma\epsilon T\beta+$.

These KKT conditions are a set of n linear equations in the n-dimensional basis $\{\hat{x}_{top}, \delta\hat{x}_\tau\}$, with a special matrix structure. We can use this special structure to solve for the optimal state estimates efficiently. The first step is to perform a type of Gaussian elimination to produce a lower triangular structure, the step of the algorithm that is reminiscent of linear Kalman filtering. This is done successively from level to level, starting at the lowest level of the tree and moving upwards. The back-substitution step of the algorithm proceeds down the tree, and is reminiscent to the Rauch-Tung-Striebel smoothing algorithm.

The following development provides a motivation for the subsidiary variables. We eliminate from 1st to 2nd to 3rd levels until all of the subsidiary variables have been defined, then generate a generic elimination, and thus the efficient, parallelisable algorithm.

The solution at the first level

Consider node τ at the first level. Each node corresponds to specific basis variable, $\delta\hat{x}_\tau$, and thus a corresponding KKT equation. We perform elimination from the first level to the second level above the zeroth level.

$$\begin{aligned}
& \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C\delta\hat{x}_\tau + (1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}((1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau) \\
= & \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1}\delta y_\tau^\theta - (1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} u_{\tau\alpha} \qquad (2.87) \\
& (\sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C + (1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}(1+A_{\tau\alpha}))\delta\hat{x}_\tau \\
& -(1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}(1-A_{\tau\alpha})\hat{x}_\tau \\
= & \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1}\delta y_\tau^\theta - (1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} u_{\tau\alpha} \qquad (2.88)
\end{aligned}$$

Define a_τ, b_τ and c_τ.

$$(\sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C + a_\tau b_\tau)\delta\hat{x}_\tau + a_\tau c_\tau \hat{x}_\tau = d_\tau \qquad (2.89)$$

$$\begin{aligned}
a_\tau &= (1 + A_{\tau\alpha}) && (2.90)\\
b_\tau &= (\sqrt{2}B_{\tau\alpha})^{-T}Q_\tau^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}(1 + A_{\tau\alpha}) && (2.91)\\
c_\tau &= -(\sqrt{2}B_{\tau\alpha})^{-T}Q_\tau^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}(1 - A_{\tau\alpha}) && (2.92)\\
d_\tau &= \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1}\delta y_\tau^\theta - (1 + A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q_\tau^{-1}u_{\tau\alpha} - \frac{1}{\sqrt{2}}(u^*_{\tau\alpha} - u^*_{\tau\beta}) && (2.93)
\end{aligned}$$

At the zeroth level all of the subsidiary variables are zero, as will become clear at higher levels. This definition is included here to avoid duplicate definitions of d_τ

$$u^*_{\tau\alpha} = u^*_{\tau\beta} = 0 \tag{2.94}$$

The goal of the algorithm is to generate equations of the following form.

$$a_\tau^\dagger \delta\hat{x}_\tau + c_\tau^\dagger \hat{x}_\tau = d_\tau^\dagger \tag{2.95}$$

Equivalently,

$$\delta\hat{x}_\tau = a_\tau^{\dagger-1}(d_\tau^\dagger - c_\tau^\dagger \hat{x}_\tau) \tag{2.96}$$

(2.96) expresses $\delta\hat{x}_t$ exclusively in terms of information above it on the tree, since $\hat{x}_\tau$ can be expressed in terms of the top scaling function and higher wavelet coefficients, and completes the Gaussian elimination from the first row to the second row.

The goal of the Gaussian elimination recursion on the remainder of the tree is to create expressions of the form (2.95), by successively incorporating information from lower levels.

The solution at the second level

Consider node τ at the second level. Each node corresponds to specific basis variable, $\delta\hat{x}_\tau$, and thus a corresponding KKT equation. We perform elimination from the second level to the third level above the zeroth level.

$$\begin{aligned}
&\sum_{\theta \epsilon M} C^T R_\tau^{\theta -1} C \delta\hat{x}_\tau + (1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2}B_{\tau\alpha})^{-1}((1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau) \\
&-\frac{1}{\sqrt{2}}(1-A_{\tau\alpha\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_{\tau\alpha}^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}((1+A_{\tau\alpha\alpha})\delta\hat{x}_{\tau\alpha} - (1-A_{\tau\alpha\alpha})\hat{x}_{\tau\alpha}) \\
&+\frac{1}{\sqrt{2}}(1-A_{\tau\beta\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_{\tau\beta}^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}((1+A_{\tau\beta\alpha})\delta\hat{x}_{\tau\beta} - (1-A_{\tau\beta\alpha})\hat{x}_{\tau\beta}) \\
= &\sum_{\theta \epsilon M} C^T R_\tau^{\theta -1} \delta y_\tau^\theta - (1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} u_{\tau\alpha} - \frac{1}{\sqrt{2}}(u^*_{\tau\alpha} - u^*_{\tau\beta}) \qquad (2.97)
\end{aligned}$$

Defining

$$\begin{aligned}
a'_\tau &= (1 - A_{\tau\alpha}) && (2.98) \\
u^*_\tau &= +(1-A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} u_{\tau\alpha} + \frac{1}{\sqrt{2}}(u^*_{\tau\alpha} + u^*_{\tau\beta}) && (2.99)
\end{aligned}$$

$$\begin{aligned}
&\sum_{\theta \epsilon M} C^T R_\tau^{\theta -1} C \delta\hat{x}_\tau + a_\tau b_\tau \delta\hat{x}_\tau + a_\tau c_\tau \hat{x}_\tau \\
&-\frac{1}{\sqrt{2}}(a'_{\tau\alpha} b_{\tau\alpha}\delta\hat{x}_{\tau\alpha} + a'_{\tau\alpha} c_{\tau\alpha}\hat{x}_{\tau\alpha}) + \frac{1}{\sqrt{2}}(a'_{\tau\beta} b_{\tau\beta}\delta\hat{x}_{\tau\beta} + a'_{\tau\beta} c_{\tau\beta}\hat{x}_{\tau\beta}) \\
= \; & d_\tau \qquad (2.100)
\end{aligned}$$

Now, substitute (2.96), from the zeroth level for the state estimates of the descendant nodes, $\delta\hat{x}_{\tau\alpha}$, and $\delta\hat{x}_{\tau\beta}$.

$$\begin{aligned}
&\sum_{\theta \epsilon M} C^T R_\tau^{\theta -1} C \delta\hat{x}_\tau + a_\tau b_\tau \delta\hat{x}_\tau + a_\tau c_\tau \hat{x}_\tau \\
&+\frac{1}{\sqrt{2}}(a'_{\tau\alpha} b_{\tau\alpha} a^{\dagger -1}_{\tau\alpha} c^\dagger_{\tau\alpha} - a'_{\tau\alpha} c_{\tau\alpha})\hat{x}_{\tau\alpha} - \frac{1}{\sqrt{2}}(a'_{\tau\beta} b_{\tau\beta} a^{\dagger -1}_{\tau\beta} c^\dagger_{\tau\beta} - a'_{\tau\beta} c_{\tau\beta})\hat{x}_{\tau\beta} \\
= \; & d_\tau + \frac{1}{\sqrt{2}}(a'_{\tau\alpha} b_{\tau\alpha} a^{\dagger -1}_{\tau\alpha} d^\dagger_{\tau\alpha} - a'_{\tau\beta} b_{\tau\beta} a^{\dagger -1}_{\tau\beta} d^\dagger_{\tau\beta}) \qquad (2.101)
\end{aligned}$$

Apply a wavelet transform to the descendant variables.

$$\begin{aligned}
& \sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} C \delta\hat{x}_\tau + a_\tau b_\tau \delta\hat{x}_\tau + a_\tau c_\tau \hat{x}_\tau \\
& +\frac{1}{2}(a'_{\tau\alpha} b_{\tau\alpha} a_{\tau\alpha}^{\dagger-1} c_{\tau\alpha}^\dagger - a'_{\tau\alpha} c_{\tau\alpha})(\hat{x}_\tau + \delta\hat{x}_t) - \frac{1}{2}(a'_{\tau\beta} b_{\tau\beta} a_{\tau\beta}^{\dagger-1} c_{\tau\beta}^\dagger - a'_{\tau\beta} c_{\tau\beta})(\hat{x}_\tau - \delta\hat{x}_t) \\
= \ & d_\tau^\dagger \qquad (2.102)
\end{aligned}$$

And finally, collect like terms.

$$\begin{aligned}
& (\sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} C + a_\tau b_\tau + \frac{1}{2}(a'_{\tau\alpha} b_{\tau\alpha} a_{\tau\alpha}^{\dagger-1} c_{\tau\alpha}^\dagger - a'_{\tau\alpha} c_{\tau\alpha}) + \frac{1}{2}(a'_{\tau\beta} b_{\tau\beta} a_{\tau\beta}^{\dagger-1} c_{\tau\beta}^\dagger - a'_{\tau\beta} c_{\tau\beta}))\delta\hat{x}_\tau \\
& +(a_\tau c_\tau + \frac{1}{2}(a'_{\tau\alpha} b_{\tau\alpha} a_{\tau\alpha}^{\dagger-1} c_{\tau\alpha}^\dagger - a'_{\tau\alpha} c_{\tau\alpha}) - \frac{1}{2}(a'_{\tau\beta} b_{\tau\beta} a_{\tau\beta}^{\dagger-1} c_{\tau\beta}^\dagger - a'_{\tau\beta} c_{\tau\beta}))\hat{x}_\tau \\
= \ & d_\tau^\dagger \qquad (2.103)
\end{aligned}$$

$$a_\tau^\dagger \delta\hat{x}_\tau + c_\tau^\dagger \hat{x}_\tau = d_\tau^\dagger \qquad (2.104)$$

where

$$\begin{aligned}
a_\tau^\dagger &= (\sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} C + a_\tau b_\tau + \frac{1}{2}(a'_{\tau\alpha} b_{\tau\alpha} a_{\tau\alpha}^{\dagger-1} c_{\tau\alpha}^\dagger - a'_{\tau\alpha} c_{\tau\alpha}) \\
&\quad +\frac{1}{2}(a'_{\tau\beta} b_{\tau\beta} a_{\tau\beta}^{\dagger-1} c_{\tau\beta}^\dagger - a'_{\tau\beta} c_{\tau\beta})) \qquad (2.105) \\
c_\tau^\dagger &= a_\tau c_\tau + \frac{1}{2}(a'_{\tau\alpha} b_{\tau\alpha} a_{\tau\alpha}^{\dagger-1} c_{\tau\alpha}^\dagger - a'_{\tau\alpha} c_{\tau\alpha}) - \frac{1}{2}(a'_{\tau\beta} b_{\tau\beta} a_{\tau\beta}^{\dagger-1} c_{\tau\beta}^\dagger - a'_{\tau\beta} c_{\tau\beta}) \qquad (2.106) \\
d_\tau^\dagger &= \sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} \delta y_\tau^\theta - (1 + A_{\tau\alpha})(\sqrt{2} B_{\tau\alpha})^{-T} Q_\tau^{-1} u_{\tau\alpha} - \frac{1}{\sqrt{2}}(u_{\tau\alpha}^* - u_{\tau\beta}^*) \qquad (2.107) \\
&\quad +\frac{1}{\sqrt{2}}(a'_{\tau\alpha} b_{\tau\alpha} a_{\tau\alpha}^{\dagger-1} d_{\tau\alpha}^\dagger - a'_{\tau\beta} b_{\tau\beta} a_{\tau\beta}^{\dagger-1} d_{\tau\beta}^\dagger) \qquad (2.108)
\end{aligned}$$

Certain groups of terms appear repeatedly in the treatment, due to the scaling function terms that feature at all ascendant nodes. We define further subsidiary variables, c', and d', to simplify the algorithm. From (2.97) and (2.101) above,

$$\begin{aligned}
&(1-A_{\tau\alpha\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q_{\tau\alpha}^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}((1+A_{\tau\alpha\alpha})\delta\hat{x}_{\tau\alpha}-(1-A_{\tau\alpha})\hat{x}_{\tau\alpha}) \\
=\;& a'_{\tau\alpha}b_{\tau\alpha}\delta\hat{x}_{\tau\alpha}+a'_{\tau\alpha}c_{\tau\alpha}\hat{x}_{\tau\alpha} && (2.109)\\
=\;& -\frac{1}{\sqrt{2}}(a'_{\tau\alpha}b_{\tau\alpha}a_{\tau\alpha}^{\dagger-1}c_{\tau\alpha}^{\dagger}-a'_{\tau\alpha}c_{\tau\alpha})\left(\hat{x}_{\tau}+\delta\hat{x}_{\tau}\right)+a'_{\tau\alpha}b_{\tau\alpha}a_{\tau\alpha}^{\dagger-1}d_{\tau\alpha}^{\dagger} && (2.110)
\end{aligned}$$

Define

$$\begin{aligned}
c'_{\tau} &= -a'_{\tau}b_{\tau}a_{\tau}^{\dagger-1}c_{\tau}^{\dagger}+a'_{\tau}c_{\tau} && (2.111)\\
d'_{\tau} &= -a'_{\tau}b_{\tau}a_{\tau}^{\dagger-1}d_{\tau}^{\dagger} && (2.112)
\end{aligned}$$

Then

$$\begin{aligned}
a'_{\tau\alpha}b_{\tau\alpha}\delta\hat{x}_{\tau\alpha}+a'_{\tau\alpha}c_{\tau\alpha}\hat{x}_{\tau\alpha} &= \frac{1}{\sqrt{2}}c'_{\tau\alpha}\left(\hat{x}_{\tau}+\delta\hat{x}_{\tau}\right)+d'_{\tau\alpha} && (2.113)\\
a'_{\tau\beta}b_{\tau\beta}\delta\hat{x}_{\tau\beta}+a'_{\tau\beta}c_{\tau\beta}\hat{x}_{\tau\beta} &= \frac{1}{\sqrt{2}}c'_{\tau\beta}\left(\hat{x}_{\tau}-\delta\hat{x}_{\tau}\right)+d'_{\tau\beta} && (2.114)
\end{aligned}$$

and the first level recursion becomes,

$$\begin{aligned}
a_{\tau}^{\dagger} &= \sum_{\theta\epsilon M}C^{T}R_{\tau}^{\theta-1}C+a_{\tau}b_{\tau}-\frac{1}{2}(c'_{\tau\alpha}+c'_{\tau\beta}) && (2.115)\\
c_{\tau}^{\dagger} &= a_{\tau}c_{\tau}-\frac{1}{2}(c'_{\tau\alpha}-c'_{\tau\beta}) && (2.116)\\
d_{\tau}^{\dagger} &= \sum_{\theta\epsilon M}C^{T}R_{\tau}^{\theta-1}\delta y_{\tau}^{\theta}-(1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q_{\tau}^{-1}u_{\tau\alpha}\\
&\quad -\frac{1}{\sqrt{2}}(u_{\tau\alpha}^{*}-u_{\tau\beta}^{*})-\frac{1}{\sqrt{2}}(d'_{\tau\alpha}-d'_{\tau\beta}) && (2.117)
\end{aligned}$$

At the first level, define

$$d_{\tau}^{*}=\frac{1}{\sqrt{2}}(d'_{\tau\alpha}+d'_{\tau\beta}) \qquad (2.118)$$

There is a clear similarity between this and the Haar transform. Essentially the wavelet portion of d'_{τ} is deposited at each level with the $d_{\tau}^{\dagger}$ while the scaling function portion is projected up the tree.

The solution at the third level

Consider node τ at the third level, and perform the elimination.

$$\begin{aligned}
&\sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C\delta\hat{x}_\tau + (1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q_\tau^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}((1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau) \\
&-\frac{1}{\sqrt{2}}(a'_{\tau\alpha}b_{\tau\alpha}\delta\hat{x}_{\tau\alpha} + a'_{\tau\alpha}c_{\tau\alpha}\hat{x}_{\tau\alpha}) + \frac{1}{\sqrt{2}}(a'_{\tau\beta}b_{\tau\beta}\delta\hat{x}_{\tau\beta} + a'_{\tau\beta}c_{\tau\beta}\hat{x}_{\tau\beta}) \\
&-\frac{1}{2}(a'_{\tau\alpha\alpha}b_{\tau\alpha\alpha}\delta\hat{x}_{\tau\alpha\alpha} + a'_{\tau\alpha\alpha}c_{\tau\alpha\alpha}\hat{x}_{\tau\alpha\alpha}) - \frac{1}{2}(a'_{\tau\alpha\beta}b_{\tau\alpha\beta}\delta\hat{x}_{\tau\alpha\beta} + a'_{\tau\alpha\beta}c_{\tau\alpha\beta}\hat{x}_{\tau\alpha\beta}) \\
&+\frac{1}{2}(a'_{\tau\beta\alpha}b_{\tau\beta\alpha}\delta\hat{x}_{\tau\beta\alpha} + a'_{\tau\beta\alpha}c_{\tau\beta\alpha}\hat{x}_{\tau\beta\alpha}) + \frac{1}{2}(a'_{\tau\beta\beta}b_{\tau\beta\beta}\delta\hat{x}_{\tau\beta\beta} + a'_{\tau\beta\beta}c_{\tau\beta\beta}\hat{x}_{\tau\beta\beta}) \\
= \ &\sum_{\theta\epsilon M} C^T R_\tau^{\theta-1}\delta y_\tau^\theta - (1+A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q_\tau^{-1}u_{\tau\alpha} - \frac{1}{\sqrt{2}}(u^*_{\tau\alpha} - u^*_{\tau\beta}) \qquad (2.119)
\end{aligned}$$

Then from (2.110), (2.111) and (2.112), from the two levels below,

$$\begin{aligned}
a'_{\tau\alpha_i}b_{\tau\alpha_i}\delta\hat{x}_{\tau\alpha_i} + a'_{\tau\alpha_i}c_{\tau\alpha_i}\hat{x}_{\tau\alpha_i} &= -(a'_{\tau\alpha_i}b_{\tau\alpha_i}a^{\dagger-1}_{\tau\alpha_i}c^\dagger_{\tau\alpha_i} - a'_{\tau\alpha_i}c_{\tau\alpha_i})\hat{x}_{\tau\alpha_i} \\
&\quad + a'_{\tau\alpha_i}b_{\tau\alpha_i}a^{\dagger-1}_{\tau\alpha_i}d^\dagger_{\tau\alpha_i} \qquad (2.120) \\
&= c'_{\tau\alpha_i}\hat{x}_{\tau\alpha_i} - d'_{\tau\alpha_i} \qquad (2.121)
\end{aligned}$$

Substituting this and into (2.119) gives,

$$\begin{aligned}
&\sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C\delta\hat{x}_\tau + a_\tau b_\tau\delta\hat{x}_\tau + a_\tau c_\tau\hat{x}_\tau \\
&-\frac{1}{\sqrt{2}}(a'_{\tau\alpha}b_{\tau\alpha}\delta\hat{x}_{\tau\alpha} + a'_{\tau\alpha}c_{\tau\alpha}\hat{x}_{\tau\alpha}) + \frac{1}{\sqrt{2}}(a'_{\tau\beta}b_{\tau\beta}\delta\hat{x}_{\tau\beta} + a'_{\tau\beta}c_{\tau\beta}\hat{x}_{\tau\beta}) \\
&-\frac{1}{2}c'_{\tau\alpha\alpha}\hat{x}_{\tau\alpha\alpha} - \frac{1}{2}c'_{\tau\alpha\beta}\hat{x}_{\tau\alpha\beta} + \frac{1}{2}c'_{\tau\beta\alpha}\hat{x}_{\tau\beta\alpha} + \frac{1}{2}c'_{\tau\beta\beta}\hat{x}_{\tau\beta\beta} \\
= \ &d_\tau - \frac{1}{2}d'_{\tau\alpha\alpha} - \frac{1}{2}d'_{\tau\alpha\beta} + \frac{1}{2}d'_{\tau\beta\alpha} + \frac{1}{2}d'_{\tau\beta\beta} \\
= \ &d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) \qquad (2.122)
\end{aligned}$$

And a Haar transform gives,

$$\begin{aligned}
&\sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} C \delta\hat{x}_\tau + a_\tau b_\tau \delta\hat{x}_\tau + a_\tau c_\tau \hat{x}_\tau \\
&-\frac{1}{\sqrt{2}}(a'_{\tau\alpha} b_{\tau\alpha} + \frac{1}{2}(c'_{\tau\alpha\alpha} - c'_{\tau\alpha\beta}))\delta\hat{x}_{\tau\alpha} - \frac{1}{\sqrt{2}}(a'_{\tau\alpha} c_{\tau\alpha} + \frac{1}{2}(c'_{\tau\alpha\alpha} + c'_{\tau\alpha\beta}))\hat{x}_{\tau\alpha} \\
&+\frac{1}{\sqrt{2}}(a'_{\tau\beta} b_{\tau\beta} + \frac{1}{2}(c'_{\tau\beta\alpha} - c'_{\tau\beta\beta}))\delta\hat{x}_{\tau\beta} + \frac{1}{\sqrt{2}}(a'_{\tau\beta} c_{\tau\beta} + \frac{1}{2}(c'_{\tau\beta\alpha} + c'_{\tau\beta\beta}))\hat{x}_{\tau\beta} \\
= \quad & d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) \qquad (2.123)
\end{aligned}$$

Note the symmetry, as before, and define

$$c''_\tau = -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}))a_\tau^{\dagger-1} c_\tau^\dagger + (a'_\tau c_\tau + \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta})) \qquad (2.124)$$

$$d''_\tau = -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}))a_\tau^{\dagger-1} d_\tau^\dagger \qquad (2.125)$$

Then,

$$\begin{aligned}
&\sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} C \delta\hat{x}_\tau + a_\tau b_\tau \delta\hat{x}_\tau + a_\tau c_\tau \hat{x}_\tau \\
&-\frac{1}{\sqrt{2}}(c''_{\tau\alpha}\hat{x}_{\tau\alpha} - c''_{\tau\beta}\hat{x}_{\tau\beta}) \\
= \quad & d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d''_{\tau\alpha} - d''_{\tau\beta}) \qquad (2.126)
\end{aligned}$$

$$\begin{aligned}
&(\sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} C + a_\tau b_\tau - \frac{1}{2}(c''_{\tau\alpha} + c''_{\tau\beta}))\delta\hat{x}_\tau + (a_\tau c_\tau - \frac{1}{2}(c''_{\tau\alpha} - c''_{\tau\beta}))\hat{x}_\tau \\
= \quad & d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d''_{\tau\alpha} - d''_{\tau\beta}) \qquad (2.127)
\end{aligned}$$

Let

$$d^*_\tau = \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} + d^*_{\tau\beta}) + \frac{1}{\sqrt{2}}(d''_{\tau\alpha} + d''_{\tau\beta}) \qquad (2.128)$$

For the third level,

$$a^{\dagger}_{\tau}\delta\hat{x}_{\tau} + c^{\dagger}_{\tau}\hat{x}_{\tau} = d^{\dagger}_{\tau} \tag{2.129}$$

with

$$\begin{aligned} a^{\dagger}_{\tau} &= \sum_{\theta\epsilon M} C^T R^{\theta-1}_{\tau} C + a_{\tau}b_{\tau} - \frac{1}{2}(c''_{\tau\alpha} + c''_{\tau\beta}) && (2.130)\\ c^{\dagger}_{\tau} &= a_{\tau}c_{\tau} - \frac{1}{2}(c''_{\tau\alpha} - c''_{\tau\beta}) && (2.131)\\ d^{\dagger}_{\tau} &= d_{\tau} - \frac{1}{\sqrt{2}}(d^{*}_{\tau\alpha} - d^{*}_{\tau\beta}) - \frac{1}{\sqrt{2}}(d''_{\tau\alpha} - d''_{\tau\beta}) && (2.132) \end{aligned}$$

The different number of primes used in the c', d', c'' and d'' variables is used in the derivation to distinguish more easily between the levels, although this distinction is unnecessary. Thus in this final version of the algorithm, they are reduced to a single prime.

The solution at a general level

We begin with a couple of subsidiary results that will enable us to truncate the terms in the equation. Recall that the scaling function terms contribute to the cost function in the following way:

$$\begin{aligned} 0 &= \frac{1}{2}\frac{d\Phi}{d\delta\hat{x}_{\tau}} \\ &= \frac{1}{2}\frac{\partial\Phi}{\partial\delta\hat{x}_{\tau}} + \frac{1}{2}\sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_{\sigma}}\frac{\partial\hat{x}_{\sigma}}{\partial\delta\hat{x}_{\tau}} \\ &= \sum_{\theta\epsilon M} C^T R^{\theta-1}_{\tau} C\delta\hat{x}_{\tau} + a_{\tau}b_{\tau}\delta\hat{x}_{\tau} + a_{\tau}c_{\tau}\hat{x}_{\tau} - d_{\tau} + \frac{1}{2}\sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_{\sigma}}\frac{\partial\hat{x}_{\sigma}}{\partial\delta\hat{x}_{\tau}} && (2.133) \end{aligned}$$

$$\begin{aligned} \frac{\partial\Phi}{\partial\hat{x}_{\tau}} &= -a'_{\tau}(B_{\tau\alpha})^{-T}Q^{-1}_{\tau}(B_{\tau\alpha})^{-1}((1+A_{\tau\alpha})\delta\hat{x}_{\tau} - (1-A_{\tau\alpha})\hat{x}_{\tau} + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha}) && (2.134)\\ &= -2a'_{\tau}\,(b_{\tau}\delta\hat{x}_{\tau} + c_{\tau}\hat{x}_{\tau}) - 2a'_{\tau}(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_{\tau}u_{\tau\alpha} && (2.135) \end{aligned}$$

and from the Haar reconstruction equation, applied recursively,

$$\begin{aligned}\frac{\partial \hat{x_\sigma}}{\partial \delta \hat{x}_\tau} &= (\sqrt{2})^{-m(\sigma,\tau)} \text{ if } \sigma \epsilon T\alpha + \\ \frac{\partial \hat{x_\sigma}}{\partial \delta \hat{x}_\tau} &= -(\sqrt{2})^{-m(\sigma,\tau)} \text{ if } \sigma \epsilon T\beta+ \end{aligned} \tag{2.136}$$

where $T\alpha+$ refers to the collection of $\tau\alpha$ and all nodes that descend from it on the tree, and $m(\sigma,\tau)$ is the number of levels between τ and σ. Also $m(\sigma,\tau) = m(\sigma,\tau\alpha)+1$ if $\sigma\epsilon T\alpha+$ and $m(\sigma,\tau) = m(\sigma,\tau\beta)+1$ if $\sigma\epsilon T\beta+$.

In the following proofs, I have either abused my notation, or dealt exclusively with the undriven case. This is without loss of generality, since the terms containing u_τ are all included in the u^*_τ terms, and thus are dealt with elsewhere in this treatment.

Claim 4 *The scaling function term splits into a simple combination of its descendant scaling function terms*

Proof.

$$\begin{aligned}\sum_{\sigma\epsilon T} \frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} &= \sum_{\sigma\epsilon T\alpha+} \frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} + \sum_{\sigma\epsilon T\beta+} \frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} \\ &= \sum_{\sigma\epsilon T\alpha+} (\sqrt{2})^{-m(\sigma,\tau)}\frac{\partial\Phi}{\partial\hat{x}_\sigma} - \sum_{\sigma\epsilon T\beta+} (\sqrt{2})^{-m(\sigma,\tau)}\frac{\partial\Phi}{\partial\hat{x}_\sigma} \\ &= \frac{1}{\sqrt{2}}\left(\sum_{\sigma\epsilon T\alpha+} (\sqrt{2})^{-m(\sigma,\tau\alpha)}\frac{\partial\Phi}{\partial\hat{x}_\sigma} - \sum_{\sigma\epsilon T\beta+} (\sqrt{2})^{-m(\sigma,\tau\beta)}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\right) \\ &= \frac{1}{\sqrt{2}}\left(\sum_{\sigma\epsilon T\alpha+} \frac{\partial\Phi}{\partial\hat{x}_\sigma}\left|\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_{\tau\alpha}}\right| - \sum_{\sigma\epsilon T\beta+} \frac{\partial\Phi}{\partial\hat{x}_\sigma}\left|\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_{\tau\beta}}\right|\right) \\ &\quad + \frac{1}{\sqrt{2}}\frac{\partial\Phi}{\partial\hat{x}_{\tau\alpha}} - \frac{1}{\sqrt{2}}\frac{\partial\Phi}{\partial\hat{x}_{\tau\beta}}\end{aligned} \tag{2.137}$$

■

Claim 5

$$\begin{aligned}\frac{1}{2}\sum_{\sigma\epsilon T} \frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} &= -\frac{1}{2}(c''_{\tau\alpha}+c''_{\tau\beta})\delta\hat{x}_\tau - \frac{1}{2}(c''_{\tau\alpha}-c''_{\tau\beta})\hat{x}_\tau \\ &\quad + \frac{1}{\sqrt{2}}(d^*_{\tau\alpha}-d^*_{\tau\beta}) + \frac{1}{\sqrt{2}}(d''_{\tau\alpha}-d''_{\tau\beta})\end{aligned} \tag{2.138}$$

Proof. By induction. This has been shown to be true at the third level, and likewise, at the lower levels, even though at these many of the terms are zero. Suppose it is true at an arbitrary level.

$$\frac{\sqrt{2}}{2}\sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} = \left(\frac{1}{2}\sum_{\sigma\epsilon T\alpha+}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\left|\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_{\tau\alpha}}\right| - \frac{1}{2}\sum_{\sigma\epsilon T\beta+}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\left|\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_{\tau\beta}}\right|\right) + \frac{1}{2}\frac{\partial\Phi}{\partial\hat{x}_{\tau\alpha}} - \frac{1}{2}\frac{\partial\Phi}{\partial\hat{x}_{\tau\beta}} \tag{2.139}$$

$$= \frac{1}{2}\left(-2a'_{\tau\alpha}b_{\tau\alpha}\delta\hat{x}_{\tau\alpha} - 2a'_{\tau\alpha}c_{\tau\alpha}\hat{x}_{\tau\alpha}\right) + \frac{1}{2}\left(2a'_{\tau\beta}b_{\tau\beta}\delta\hat{x}_{\tau\beta} + 2a'_{\tau\beta}c_{\tau\beta}\hat{x}_{\tau\beta}\right) - \frac{1}{2}(c'_{\tau\alpha\alpha} - c'_{\tau\alpha\beta})\delta\hat{x}_{\tau\alpha} - \frac{1}{2}(c'_{\tau\alpha\alpha} + c'_{\tau\alpha\beta})\hat{x}_{\tau\alpha} + \frac{1}{2}(c'_{\tau\beta\alpha} - c'_{\tau\beta\beta})\delta\hat{x}_{\tau\beta} + \frac{1}{2}(c'_{\tau\beta\alpha} + c'_{\tau\beta\beta})\hat{x}_{\tau\beta} + \frac{1}{\sqrt{2}}(d^*_{\tau\alpha\alpha} + d^*_{\tau\alpha\beta}) + \frac{1}{\sqrt{2}}(d'_{\tau\alpha\alpha} + d'_{\tau\alpha\beta}) - \frac{1}{\sqrt{2}}(d^*_{\tau\beta\alpha} + d^*_{\tau\beta\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\beta\alpha} + d'_{\tau\beta\beta}) \tag{2.140}$$

$$= \left(-a'_{\tau\alpha}b_{\tau\alpha} - \frac{1}{2}(c'_{\tau\alpha\alpha} - c'_{\tau\alpha\beta})\right)\delta\hat{x}_{\tau\alpha} + \left(-a'_{\tau\alpha}c_{\tau\alpha} - \frac{1}{2}(c'_{\tau\alpha\alpha} + c'_{\tau\alpha\beta})\right)\hat{x}_{\tau\alpha} \left(a'_{\tau\beta}b_{\tau\beta} + \frac{1}{2}(c'_{\tau\beta\alpha} - c'_{\tau\beta\beta})\right)\delta\hat{x}_{\tau\beta} + \left(a'_{\tau\beta}c_{\tau\beta} + \frac{1}{2}(c'_{\tau\beta\alpha} + c'_{\tau\beta\beta})\right)\hat{x}_{\tau\beta} + d^*_{\tau\alpha} - d^*_{\tau\beta} \tag{2.141}$$

$$= \left(\left(a'_{\tau\alpha}b_{\tau\alpha} + \frac{1}{2}(c'_{\tau\alpha\alpha} - c'_{\tau\alpha\beta})\right)a^{\dagger-1}_{\tau\alpha}c^\dagger_{\tau\alpha} - a'_{\tau\alpha}c_{\tau\alpha} - \frac{1}{2}(c'_{\tau\alpha\alpha} + c'_{\tau\alpha\beta})\right)\hat{x}_{\tau\alpha} + \left(-\left(a'_{\tau\beta}b_{\tau\beta} + \frac{1}{2}(c'_{\tau\beta\alpha} - c'_{\tau\beta\beta})\right)a^{\dagger-1}_{\tau\beta}c^\dagger_{\tau\beta} + a'_{\tau\beta}c_{\tau\beta} + \frac{1}{2}(c''_{\tau\beta\alpha} + c''_{\tau\beta\beta})\right)\hat{x}_{\tau\beta} + \left(-a'_{\tau\alpha}b_{\tau\alpha} - \frac{1}{2}(c'_{\tau\alpha\alpha} - c'_{\tau\alpha\beta})\right)a^{\dagger-1}_{\tau\alpha}d^\dagger_{\tau\alpha} + \left(a'_{\tau\beta}b_{\tau\beta} + \frac{1}{2}(c'_{\tau\beta\alpha} - c'_{\tau\beta\beta})\right)a^{\dagger-1}_{\tau\beta}d^\dagger_{\tau\beta} + d^*_{\tau\alpha} - d^*_{\tau\beta} \tag{2.142}$$

$$= -c'_{\tau\alpha}\hat{x}_{\tau\alpha} + c'_{\tau\beta}\hat{x}_{\tau\beta} + d'_{\tau\alpha} - d'_{\tau\beta} + d^*_{\tau\alpha} - d^*_{\tau\beta} \tag{2.143}$$

$$= -c'_{\tau\alpha}\frac{1}{\sqrt{2}}\left(\delta\hat{x}_\tau + \hat{x}_\tau\right) + c'_{\tau\beta}\frac{1}{\sqrt{2}}\left(\hat{x}_\tau - \delta\hat{x}_\tau\right) + d'_{\tau\alpha} - d'_{\tau\beta} + d^*_{\tau\alpha} - d^*_{\tau\beta} \tag{2.144}$$

$$= -\frac{1}{\sqrt{2}}\left(\left(c'_{\tau\alpha} + c'_{\tau\beta}\right)\delta\hat{x}_\tau + \left(c'_{\tau\alpha} - c'_{\tau\beta}\right)\hat{x}_\tau\right) + d'_{\tau\alpha} - d'_{\tau\beta} + d^*_{\tau\alpha} - d^*_{\tau\beta} \tag{2.145}$$

$$\frac{1}{2}\sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} = -\frac{1}{2}\left(c'_{\tau\alpha} + c'_{\tau\beta}\right)\delta\hat{x}_\tau - \frac{1}{2}\left(c'_{\tau\alpha} - c'_{\tau\beta}\right)\hat{x}_\tau + \frac{1}{\sqrt{2}}\left(d'_{\tau\alpha} - d'_{\tau\beta}\right) + \frac{1}{\sqrt{2}}\left(d^*_{\tau\alpha} - d^*_{\tau\beta}\right) \tag{2.146}$$

■

Claim 6 *The general recursion uses the same steps as the third level. It generalises to the lower levels on the tree.*

Proof. In this proof, the same simplification has been made, whereby the u_τ terms in $\sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau}$ have been included in the u^*_τ terms, which are included in the definition of d_τ.

$$d_\tau = \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C\delta\hat{x}_\tau + a_\tau b_\tau\delta\hat{x}_\tau + a_\tau c_\tau\hat{x}_\tau + \frac{1}{2}\sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} \tag{2.147}$$

$$= \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C\delta\hat{x}_\tau + a_\tau b_\tau\delta\hat{x}_\tau + a_\tau c_\tau\hat{x}_\tau - \frac{1}{2}\left(c'_{\tau\alpha} + c'_{\tau\beta}\right)\delta\hat{x}_\tau$$

$$-\frac{1}{2}\left(c'_{\tau\alpha} - c'_{\tau\beta}\right)\hat{x}_\tau + \frac{1}{\sqrt{2}}\left(d'_{\tau\alpha} - d'_{\tau\beta}\right) + \frac{1}{\sqrt{2}}\left(d^*_{\tau\alpha} - d^*_{\tau\beta}\right) \tag{2.148}$$

$$d^\dagger_\tau = d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} - d'_{\tau\beta}) \tag{2.149}$$

$$= \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C\delta\hat{x}_\tau + a_\tau b_\tau\delta\hat{x}_\tau + a_\tau c_\tau\hat{x}_\tau - \frac{1}{2}\left(c'_{\tau\alpha} + c'_{\tau\beta}\right)\delta\hat{x}_\tau$$

$$-\frac{1}{2}\left(c'_{\tau\alpha} - c'_{\tau\beta}\right)\hat{x}_\tau \tag{2.150}$$

$$= \left(\sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C + a_\tau b_\tau - \frac{1}{2}\left(c'_{\tau\alpha} + c'_{\tau\beta}\right)\right)\delta\hat{x}_\tau$$

$$+\left(a_\tau c_\tau - \frac{1}{2}\left(c'_{\tau\alpha} - c'_{\tau\beta}\right)\right)\hat{x}_\tau \tag{2.151}$$

$$= a^\dagger_\tau\delta\hat{x}_\tau + c^\dagger_\tau\hat{x}_\tau \tag{2.152}$$

$$a^\dagger_\tau = \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C + a_\tau b_\tau - \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta}) \tag{2.153}$$

$$c^\dagger_\tau = a_\tau c_\tau - \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}) \tag{2.154}$$

■

The solution at the top level

The final equation concerns the top scaling function coefficient, which is part of the basis. The structure of the equation is similar to the wavelet coefficient equation at the same level, and is simplified as follows.

(2.147) becomes

$$d_{top} = \sum_{\theta \epsilon M} C^T R_{top}^{\theta-1} C\hat{x}_\tau - (a'_\tau b_\tau \delta\hat{x}_\tau + a'_\tau c_\tau \hat{x}_\tau) + \frac{1}{2}\sum_{\sigma\epsilon T} \frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\hat{x}_{top}}$$

where

$$\begin{aligned}
\frac{1}{2}\sum_{\sigma\epsilon T} \frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\hat{x}_{top}} &= \frac{1}{2}\frac{1}{\sqrt{2}}\left(\sum_{\sigma\epsilon T\alpha+}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\left|\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_{\tau\alpha}}\right| + \sum_{\sigma\epsilon T\beta+}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\left|\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_{\tau\beta}}\right|\right) \\
&\quad +\frac{1}{2}\frac{1}{\sqrt{2}}\frac{\partial\Phi}{\partial\hat{x}_{\tau\alpha}} + \frac{1}{2}\frac{1}{\sqrt{2}}\frac{\partial\Phi}{\partial\hat{x}_{\tau\beta}} && (2.155) \\
&= -\frac{1}{2}\left(c'_{\tau\alpha} - c'_{\tau\beta}\right)\delta\hat{x}_\tau - \frac{1}{2}\left(c'_{\tau\alpha} + c'_{\tau\beta}\right)\hat{x}_\tau \\
&\quad +\frac{1}{\sqrt{2}}\left(d'_{\tau\alpha} + d'_{\tau\beta}\right) + \frac{1}{\sqrt{2}}\left(d^*_{\tau\alpha} + d^*_{\tau\beta}\right) && (2.156)
\end{aligned}$$

$$\begin{aligned}
d_{top} &= \sum_{\theta\epsilon M} C^T R_{top}^{\theta-1} C\hat{x}_\tau - (a'_\tau b_\tau \delta\hat{x}_\tau + a'_\tau c_\tau \hat{x}_\tau) \\
&\quad -\frac{1}{2}\left(c'_{\tau\alpha} - c'_{\tau\beta}\right)\delta\hat{x}_\tau - \frac{1}{2}\left(c'_{\tau\alpha} + c'_{\tau\beta}\right)\hat{x}_\tau \\
&\quad +\frac{1}{\sqrt{2}}\left(d'_{\tau\alpha} + d'_{\tau\beta}\right) + \frac{1}{\sqrt{2}}\left(d^*_{\tau\alpha} + d^*_{\tau\beta}\right) && (2.157) \\
d^\ddagger_\tau &= d_{top} - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} + d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} + d'_{\tau\beta}) && (2.158) \\
&= \left(-a'_\tau b_\tau - \frac{1}{2}\left(c'_{\tau\alpha} - c'_{\tau\beta}\right)\right)\delta\hat{x}_{top} \\
&\quad \left(\sum_{\theta\epsilon M} C^T R_{top}^{\theta-1} C - a'_\tau c_\tau - \frac{1}{2}\left(c'_{\tau\alpha} + c'_{\tau\beta}\right)\right)\hat{x}_{top} && (2.159) \\
&= a^\ddagger_\tau \delta\hat{x}_\tau + c^\ddagger_\tau \hat{x}_\tau && (2.160) \\
a^\ddagger_\tau &= -a'_\tau b_\tau - \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}) && (2.161) \\
c^\ddagger_\tau &= \sum_{\theta\epsilon M} C^T R_{top}^{\theta-1} C - a'_\tau c_\tau - \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta}) && (2.162)
\end{aligned}$$

Algorithm 7 *The complete algorithm for the unconstrained multiscale state estimator for the driven case with multiple measurement sets is given below. The downsweep involves the computation of the following variables, recursively until the top node of the tree is reached.*

Initial Conditions

$$c'_\tau = d'_\tau = d^*_\tau = 0 \;\; \forall \tau \epsilon L_0 \tag{2.163}$$

$$u^*_\tau = 0 \; \forall \tau \epsilon L_0 \tag{2.164}$$

Primary Variables

$$a_\tau = (1 + A_{\tau\alpha}) \tag{2.165}$$

$$b_\tau = (\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2}B_{\tau\alpha})^{-1}(1 + A_{\tau\alpha}) \tag{2.166}$$

$$c_\tau = -(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2}B_{\tau\alpha})^{-1}(1 - A_{\tau\alpha}) \tag{2.167}$$

$$d_\tau = \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} \delta y_\tau^\theta - (1 + A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} u_{\tau\alpha} - \frac{1}{\sqrt{2}}(u^*_{\tau\alpha} - u^*_{\tau\beta}) \tag{2.168}$$

$$d_{top} = \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} y_{top}^\theta - (1 - A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} u_{\tau\alpha} - \frac{1}{\sqrt{2}}(u^*_{\tau\alpha} + u^*_{\tau\beta}) \tag{2.169}$$

Recursive Variables

$$u^*_\tau = -(1 - A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} u_{\tau\alpha} + \frac{1}{\sqrt{2}}(u^*_{\tau\alpha} + u^*_{\tau\beta}) \tag{2.170}$$

$$d^*_\tau = \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} + d^*_{\tau\beta}) + \frac{1}{\sqrt{2}}(d'_{\tau\alpha} + d'_{\tau\beta}) \tag{2.171}$$

$$a^\dagger_\tau = \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C + a_\tau b_\tau - \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta}) \tag{2.172}$$

$$c^\dagger_\tau = a_\tau c_\tau - \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}) \tag{2.173}$$

$$d^\dagger_\tau = d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} - d'_{\tau\beta}) \tag{2.174}$$

$$a'_\tau = (1 - A_{\tau\alpha}) \tag{2.175}$$

$$c'_\tau = -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta})) a_\tau^{\dagger-1} c^\dagger_\tau + (+a'_\tau c_\tau + \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta})) \tag{2.176}$$

$$d'_\tau = -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta})) a_\tau^{\dagger-1} d^\dagger_\tau \tag{2.177}$$

Scaling Function Variables (usually only used at the top node)

$$a^{\ddagger}_{\tau} = -a'_{\tau} b_{\tau} - \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}) \tag{2.178}$$

$$c^{\ddagger}_{\tau} = \sum_{\theta \epsilon M} C^T R^{\theta -1}_{top} C - a'_{\tau} c_{\tau} - \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta}) \tag{2.179}$$

$$d^{\ddagger}_{\tau} = d_{top} - \frac{1}{\sqrt{2}}(d^{*}_{\tau\alpha} + d^{*}_{\tau\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} + d'_{\tau\beta}) \tag{2.180}$$

The top node scaling function and wavelet coefficients are solved simultaneously, to obtain, $\hat{x}_{top}$ and $\delta\hat{x}_{top}$ from

$$a^{\dagger}_{top} \delta\hat{x}_{top} + c^{\dagger}_{top} \hat{x}_{top} = d^{\dagger}_{top} \tag{2.181}$$

$$a^{\ddagger}_{top} \delta\hat{x}_{top} + c^{\ddagger}_{top} \hat{x}_{top} = d^{\ddagger}_{top} \tag{2.182}$$

The downsweep involves a wavelet reconstruction from the upper levels for which optimal state estimates have been computed, followed by a backsubstitution into the dagger equations.

$$\hat{x}_{\tau\alpha} = \frac{1}{\sqrt{2}} \left(\hat{x}_{\tau} + \delta\hat{x}_{\tau}\right) \tag{2.183}$$

$$\hat{x}_{\tau\beta} = \frac{1}{\sqrt{2}} \left(\hat{x}_{\tau} - \delta\hat{x}_{\tau}\right) \tag{2.184}$$

$$a^{\dagger}_{\tau} \delta\hat{x}_{\tau} + c^{\dagger}_{\tau} \hat{x}_{\tau} = d^{\dagger}_{\tau} \tag{2.185}$$

This completes the calculation of the optimal estimates of all state variables on the tree, both wavelet and scaling function.

2.4.8 Uses of the multiscale state estimator

The current formulation works for any number of sets of measurements at any level on the tree. Each set of measurements is projected up the tree using the wavelet transform. The algorithm is effective for single or multiple sets of measurements.

For single sets of measurements, a single theta is in the set M. Multiple sets of measurements do not change the structure or the complexity of the algorithm, except for the start-up cost of the wavelet decomposition of each set.

Sparse and missing measurements are treated by setting $R^{\theta}_{\tau} = \infty$, where the missing measurements would have appeared.

2.4.9 Solution of the unconstrained estimation problem

The unconstrained multiscale state estimation problem can be solved efficiently using the algorithm. The efficiency is due to the structure of the set of normal equations. The algorithm involves a sweep up the tree, from the zeroth level to the top node, collecting model and measurement information , and summarising this into three variables per node, $a^\dagger$, the coefficient of the wavelet coefficient in the reduced equation, $c^\dagger$, the coefficient of the scaling function coefficient, and $d^\dagger$, a collection of the measurement data from all child nodes.

Since information about the measurements and the model is stored in a separate set of variables, a second estimation using the same model but different measurements would require only the d family of variables to be recomputed.

The algorithm is completely parallelisable, and is fast compared to the standard MATLAB quadratic programming solver, since it makes the most efficient use of the sparsity of the system. In a single upsweep all of the coefficients are computed recursively, using only the information on the subtree from that node. A single downsweep finds the optimal state estimates from these coefficients. At the top node of the tree, there is an inversion of a square matrix of dimension twice the state size, since the top node wavelet coefficient and scaling function coefficient are coupled and must be solved simultaneously.

2.4.10 Checking the solution

The algorithm was checked for correctness by constructing the complete quadratic program. This was solved using the standard MATLAB QP solver, and compared to the results from the multiscale state estimator. For a four state problem, the maximum deviation in the state estimates from the two methods was 10^{-13}, consistent with the computational error of MATLAB. The multiscale technique was considerably faster, by a factor of around 100 for a 2×2 system and a 32 point problem.

2.4.11 Optimal fusion of multirate measurements

The multiscale estimation algorithm works for multiple sets of measurements. This was motivated by estimation problems where different techniques are used to obtain measurements. These may be at different rates, or at different levels of resolution. Specifically, one may have a fast, cheap, but inaccurate means of measuring some process variable frequently, and a more expensive, but thorough analysis performed less frequently. The multiscale estimation algorithm fuses these sets optimally, by incorporating both into the state estimate. With multiple sets of measurements, the estimation of the current state proceeds as for the single rate measurement algorithm, but with each set of measurements indexed by a θ.

The sets of measurements will appear for the first time at the level of the tree that corresponds to their sampling frequency. A wavelet transform of each set will propagate the information at each level to nodes above it. When the multiscale state estimator of this section reaches the top node of the tree, it generates the optimal estimate at the top node of the tree, based on all of the available measurements. The downward sweep smooths the information from the complete set of measurements at all levels of resolution, and incorporates it into the optimal state estimates at all nodes of the tree.

2.4.12 Further uses for the multiscale state estimation algorithm

The algorithm extends to multivariable dynamic systems, since there is no limit on the number of state variables. In addition to multi-input-multi-output (MIMO) systems, the approach allows us to perform estimations for higher order polynomial dynamic systems, since these can be represented as first-order systems with state augmentation. A potential area for future study is the idea of parametric variation in the model, commonly solved in the time domain using the extended Kalman filter.

Another possible direction for further study may be to extend the algorithm to allow for multiple models, in the same way that it has been extended to deal with multiple sets of measurements. Of course this makes no sense for two first-order dynamic systems with different parameters as the solution will be an average of two conflicting models, fighting over control of the data. A more meaningful approach would be to truncate the multiscale state estimation algorithm at a level where the dynamic system gets lost in the input data dynamics, and thus loses its meaning. The user would be able to specify the confidence in the various models, by adjusting the Q parameters, or covariances of the modeling errors.

The algorithm provides a natural way to penalise measurement and modelling errors at various resolutions, since the R and Q values can be specified differently at the various scales. Choosing small values of R or Q at the higher levels on the tree would force the algorithm to produce a small error value at these levels, at the expense of higher errors elsewhere on the tree. This approach should be followed when most of the measurement noise is expected to be high frequency, or most of the modelling error is known to be within a particular frequency range.

The error values returned give a complete profile of the observed model uncertainty at all levels of resolution. This allows a richer error structure to be used in the dual problem to estimation - model predictive control.

2.5 The constrained state estimation problem

The extension of the multiscale state estimation problem to the constrained case follows directly from constructing a Lagrangian from the original cost function and constraints. Each constraint has an associated μ, which is appended to the cost function as a product with the constraint. A byproduct of the way the derivatives are performed is that the Lagrange multiplier terms separate out, and can be collected in their own set of variables, μ_τ, as the measurements are collected in the d variables.

This provides the same reusability as was observed with the measurements, so different sets of constraints can be used with the same set of model equations very efficiently. Industrially, this is useful when a plant switches between a number of different operating regimes. The model structure of the problem stays the same, but the set of constraints changes independently of the remainder of the model.

The μ_τ variables provide a guarantee of optimal state estimations by way of the Karush-Kuhn-Tucker conditions, but may produce algorithms that are combinatorial in the worst case [4]. The linear complimentarity condition is bilinear in the Lagrange multipliers and the state variables and an algorithm may visit each combination of constraints in turn before finding the optimum.

The multiscale state estimation algorithm allows us to find a basic feasible solution relatively quickly - essentially by grouping sets of constraints together in such a way that if a superconstraint is satisfied, then there exists a feasible solution satisfying the initial constraints. This provides an upper bound for the cost very quickly. A lower bound for the optimal cost function can be obtained from the unconstrained optimum state estimates extremely quickly, using an up-down sweep of the unconstrained algorithm.

2.5.1 Preliminaries

Our development is based on the satisfaction of the Karush-Kuhn-Tucker sufficient conditions for optimality [4]. The original optimization problem has the following form.

$$\begin{aligned} & \min_x \Phi(x) && (2.186) \\ g_j(x) &\leq 0 \;\; j = 0..r && (2.187) \end{aligned}$$

The Lagrangian formulation of the problem can be constructed that introduces a Lagrange multiplier, μ_j, corresponding to each inequality constraint g_j. The cost function is expanded to include the inequalities so that the Lagrangian is defined as follows.

$$L(x, \mu) = \Phi(x) + \sum_{j=1}^{r} \mu_j g_j(x) \tag{2.188}$$

The minimum value of the Lagrangian is guaranteed to be the minimum of the original cost function subject to the set of constraints. This is due to the KKT conditions, which guarantee that the summation term is zero. It is well known that the sufficient conditions for optimal solution, (x^*, μ^*), are the following.

$$\begin{aligned} \nabla_x L(x^*, \mu^*) &= 0 && (2.189) \\ \mu_j^* &\geq 0 \text{ for } j \epsilon A(x^*), \text{where } A(x^*) \text{ is the set of active constraints at } x^* && (2.190) \\ \mu_j^* &= 0 \text{ for } j \not\epsilon A(x^*) && (2.191) \end{aligned}$$

This is alternately represented as the linear complimentarity condition.

$$\mu_j^* g_j(x^*) = 0$$

and

$$y' \nabla_{xx}^2 L(x, \mu) y \geq 0, \forall y \epsilon V(x^*) \tag{2.192}$$

where

$$V(x^*) = \{y | \nabla_x g_j(x^*)' y = 0, j \epsilon A(x^*)\}$$

Let us explore each of these in more detail.

2.5.2 The derivative operator

We use as a basis for the problem, the set of $\left\{\hat{x}_{top}, \delta\hat{x}_{\tau}\right\}$, which is equivalent to the set of states at the zeroth, or physical, level in the original problem. We use ∇_x to describe the set of those derivatives.

2.5.3 The first KKT condition

The first sufficient condition concerns the derivative of the Lagrangian

$$\nabla_x L(x^*, \mu^*) = 0 \tag{2.193}$$

where x^* is a local minimum of the Cost function subject to the inequality constraints.

$$\begin{aligned} \nabla_x L(x^*, \mu^*) &= \nabla_x \Phi + \mu^T \nabla_x g &(2.194)\\ \mu &= [\mu_1, \mu_2, \mu_3, \dots, \mu_n] &(2.195)\end{aligned}$$

For linear constraints, $\nabla_x g$ will consist only of constants, and thus μ_{τ}, the manifestation of the Lagrange multipliers on the tree, is uniquely defined as

$$\mu_{\tau} = \mu^T \nabla_x g \tag{2.196}$$

$$\mu_{\tau} = \left[\mu_{top-sf}, \mu_{top}, \mu_{top\alpha}, \ldots, \mu_{top\beta^{num-levels}}\right] \tag{2.197}$$

It must be stressed that μ_{τ} is not the wavelet decomposition of μ, since μ are linked to specific constraints and not time points. The exception is the special case where there is a single constraint at each time point. This approach is more general - any number of constraints can be used and will be converted into a set of $\{\mu_{\tau}\}$. There will be the same number of μ_{τ} as there are nodes on the tree.

$$\nabla_x L(x^*, \mu^*) = \nabla_x \Phi + \mu_{\tau} \tag{2.198}$$

Note that for the unconstrained solution, one begins with

$$\nabla_x \Phi = 0$$

Recall that the unconstrained state estimator solves the following set of equations.

$$(\sum_{\theta \epsilon M} C^T R_{\tau}^{\theta-1} C + a_{\tau} b_{\tau}) \delta \hat{x}_{\tau} + a_{\tau} c_{\tau} \hat{x}_{\tau} = d_{\tau}$$

The equations for the constrained case have an additional term to account for the constraints.

$$\nabla_x \Phi = -\mu_{\tau} \tag{2.199}$$

The constrained state estimator solves the following set of equations.

$$(\sum_{\theta \epsilon M} C^T R_{\tau}^{\theta-1} C + a_{\tau} b_{\tau}) \delta \hat{x}_{\tau} + a_{\tau} c_{\tau} \hat{x}_{\tau} = d_{\tau} - \mu_{\tau} \tag{2.200}$$

μ_{τ} has exactly the same behaviour in the recursive solution as $-d_{\tau}$ and its subsidiary variables. There exist analogous subsidiary variables for μ_{τ} with definitions that parallel those for the d

variables.

$$\begin{aligned}
\mu'_{\tau} &= 0 \;\; \forall \tau \epsilon L_0 && (2.201)\\
\mu'_{\tau} &= -(a'_{\tau} b_{\tau} + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}))a_{\tau}^{\dagger -1}\mu_{\tau}^{\dagger} && (2.202)\\
\mu_{\tau}^{\dagger} &= \mu_{\tau} - \frac{1}{\sqrt{2}}(\mu^*_{\tau\alpha} - \mu^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(\mu'_{\tau\alpha} - \mu'_{\tau\beta}) && (2.203)\\
\mu^*_{\tau} &= \frac{1}{\sqrt{2}}(\mu^*_{\tau\alpha} + \mu^*_{\tau\beta}) + \frac{1}{\sqrt{2}}(\mu'_{\tau\alpha} + \mu'_{\tau\beta}) && (2.204)
\end{aligned}$$

The subsidiary variable, μ^*_{τ}, should not be confused with the μ^* representing the true optimal solution. Greek subscripts indicate the tree subsidiary variable. There is no change in the definition or recursion of the variables a, c or d, nor in any of their subsidiary variables.

The recursion structure produces an analogous set of equations

$$a_{\tau}^{\dagger}\delta\hat{x}_{\tau} + c_{\tau}^{\dagger}\hat{x}_{\tau} = d_{\tau}^{\dagger} - \mu_{\tau}^{\dagger} \qquad (2.205)$$

and for the top node,

$$a_{\tau}^{\ddagger}\delta\hat{x}_{\tau} + c_{\tau}^{\ddagger}\hat{x}_{\tau} = d_{\tau}^{\ddagger} - \mu_{\tau}^{\ddagger} \qquad (2.206)$$

Satisfaction of equations 2.205 at all nodes of the tree and 2.206 at the top node only, is equivalent to satisfaction of the derivative condition, and thus the first sufficient Karush-Kuhn-Tucker condition. Equality constraints can be incorporated without loss of generality as two inequality constraints of opposite sign.

2.5.4 The second and subsequent KKT conditions

The second Karush-Kuhn-Tucker condition concerns the Lagrange multipliers themselves. At the conclusion of any up-down sweep on the tree, one has a set of μ_τ variables from which a complete set of μ can be computed. These must satisfy the second Karush-Kuhn-Tucker conditions.

1. $\mu^*_j \geq 0$ for $j\epsilon A(x^*)$, where $A(x^*)$ is the set of active constraints at x^*
2. $\mu^*_j = 0$ for $j \not\epsilon A(x^*)$

The third KKT condition is often referred to as the sufficiency condition. The cost function, and the constraints are all twice continuously differentiable, thus we must check

$$y'\nabla^2_{xx}L(x,\mu)y \geq 0, \forall y \epsilon V(x^*) \tag{2.207}$$

where

$$V(x^*) = \{y | \nabla_x g_j(x^*)' y = 0, j \epsilon A(x^*)\} \tag{2.208}$$

We can compute $\nabla^2_{xx}L(x,\mu)$ easily from the cost function. The matrix $\nabla^2_{xx}L(x,\mu)$ is symmetric with a "wavelet" or block structure, and like the wavelet decomposition matrix, the eigenvalues will simply be the diagonal elements (or eigenvalues thereof, if they are matrices). These can be computed as

$$\begin{aligned}&\sum_\theta C^T R_\tau^{\theta-1} C + (1 + A_{\tau\alpha})^T (\sqrt{2} B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2} B_{\tau\alpha})^{-1} (1 + A_{\tau\alpha}) \\ &+ \sum \omega (1 - A_{\tau\alpha})^T (\sqrt{2} B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2} B_{\tau\alpha})^{-1} (1 - A_{\tau\alpha})\end{aligned} \tag{2.209}$$

where ω is some positive constant from the wavelet transform. For the scalar case, each of the terms is positive, and thus the total is positive, hence the eigenvalue from this diagonal element will be positive. For the vector case, each term is symmetric, and one would require that the individual matrices, Q and R, have positive eigenvalues. Since we specify these matrices as parameters of the cost function, we are able to guarantee this *a priori.*

Thus we have produced a Hessian that has positive eigenvalues and is symmetric (by the commutativity of the partial derivatives), and thus from Strang [20], when an $n \times n$ symmetric matrix has all eigenvalues positive, the matrix is positive definite, which concludes the Karush-Kuhn-Tucker sufficiency condition. Simply put, all terms are the squares of constant matrices or positive definite matrices. Since we are dealing with a quadratic program, it is globally convex.

2.5.5 Summary of the KKT conditions.

The sufficient conditions for optimality for the quadratic cost function of the constrained multiscale state estimator are that at each node,

$$a^{\dagger}_{\tau}\delta\hat{x}_{\tau} + c^{\dagger}_{\tau}\hat{x}_{\tau} = d^{\dagger}_{\tau} - \mu^{\dagger}_{\tau} \tag{2.210}$$

and at the top node,

$$a^{\ddagger}_{\tau}\delta\hat{x}_{\tau} + c^{\ddagger}_{\tau}\hat{x}_{\tau} = d^{\ddagger}_{\tau} - \mu^{\ddagger}_{\tau} \tag{2.211}$$

and the complimentarity conditions

$$\begin{aligned} \mu^*_j &\geq 0 \text{ for } j\epsilon A(x^*), \text{where } A(x^*) \text{ is the set of active constraints at } x^* \\ \mu^*_j &= 0 \text{ for } j\not\epsilon A(x^*) \end{aligned} \tag{2.212}$$

Satisfaction of these constraints defines the stopping condition for the algorithm.

2.5.6 The special case of upper and lower bounds on states

The special case of upper and lower bounds on states only produces a simpler structure and a more intuitive interpretation for the variables μ_{τ}.

It will illustrate how the general algorithm proceeds. Suppose we have a system where the only constraints are the upper and lower bounds on the state variables. Extending the wavelet decomposition motive, define $\underline{\mu}_t$ to be the Lagrange multiplier associated with the inequality $\underline{x} - \hat{x}(t) \leq 0$ and $\overline{\mu}_t$ with $\hat{x}(t) - \overline{x} \leq 0$. The wavelet decomposition of these μ produces a surprisingly simple equation for the first necessary condition.

For upper and lower bounds only, μ_{τ} on the tree are related to the μ in a simple way.

$$\mu_{\tau} = \overline{\mu}_{\tau} + \underline{\mu}_{\tau} \tag{2.213}$$

This is unique since only one of $\overline{\mu}_t$ and $\underline{\mu}_t$ can be non-zero at any solution - it is impossible for both constraints to be active for $\underline{x} \leq \overline{x}$.

2.5.7 Scaling function constraints in the multiscale domain

The inequalities for state upper and lower bounds can be projected onto the tree using the Haar transformation. The inequality constraints on the states exist at the zeroth level.

$$\underline{x} \leq \hat{x}(t) \leq \overline{x} \quad \forall t \tag{2.214}$$

Suppose that t is a node at the zeroth level, and τ is a node a the first level. Then we can infer a constraint at the first level by applying the Haar decomposition to the inequalities at the zeroth level.

$$\begin{aligned} \underline{x} &\leq \hat{x}_{\tau\alpha} \leq \overline{x} \quad (2.215) \\ \underline{x} &\leq \hat{x}_{\tau\beta} \leq \overline{x} \quad (2.216) \end{aligned}$$

Adding the inequalities and multiplying by $\frac{1}{\sqrt{2}}$, according to the Haar decomposition, produces the new inequality constraint for the scaling function coefficient of the states at the first level.

$$\frac{1}{\sqrt{2}}(2\underline{x}) \leq \hat{x}_{\tau} \leq \frac{1}{\sqrt{2}}(2\overline{x}) \tag{2.217}$$

This procedure can be repeated up the tree so that for t_m an arbitrary node at level m, we can construct an appropriate inequality.

$$(\sqrt{2})^m \underline{x} \leq \hat{x}_{\tau_m} \leq (\sqrt{2})^m \overline{x} \tag{2.218}$$

Clearly one can construct a solution at lower levels that violates lower bound while satisfying the upper bounds. Thus higher level constraints are necessary but not sufficient. These higher level constraints will be referred to as superconstraints later in this thesis.

2.5.8 Wavelet constraints in the multiscale domain

It is possible to construct parallel wavelet constraints on the tree, in a similar manner to the scaling function constraints. These are included here for completion, although not used in the algorithm below.

Suppose we have an $\hat{x}^*_\tau$ that satisfies (2.218), we can derive the necessary and sufficient condition for the magnitude of $\delta\hat{x}_\tau$ so that (2.218) holds at the $(m-1)$th level.

We can combine constraints from lower levels on the tree.

$$(\sqrt{2})^{m-1}\underline{x} \leq \frac{1}{\sqrt{2}}(\hat{x}^*_{\tau_m} + \delta\hat{x}_{\tau_m}) \leq (\sqrt{2})^{m-1}\overline{x} \tag{2.219}$$

$$(\sqrt{2})^{m-1}\underline{x} \leq \frac{1}{\sqrt{2}}(\hat{x}^*_{\tau_m} - \delta\hat{x}_{\tau_m}) \leq (\sqrt{2})^{m-1}\overline{x} \tag{2.220}$$

Thus

$$(\sqrt{2})^{m}\underline{x} \leq \hat{x}^*_{\tau_m} + \delta\hat{x}_{\tau_m} \leq (\sqrt{2})^{m}\overline{x} \tag{2.221}$$

$$(\sqrt{2})^{m}\underline{x} \leq \hat{x}^*_{\tau_m} - \delta\hat{x}_{\tau_m} \leq (\sqrt{2})^{m}\overline{x} \tag{2.222}$$

The inequalities for the wavelet coefficients that guarantee satisfaction of the scaling function coefficients at the level immediately below the current level can be constructed.

$$(\sqrt{2})^{m}\underline{x} - \hat{x}^*_{\tau_m} \leq \delta\hat{x}_{\tau_m} \leq (\sqrt{2})^{m}\overline{x} - \hat{x}^*_{\tau_m} \tag{2.223}$$

$$(\sqrt{2})^{m}\underline{x} - \hat{x}^*_{\tau_m} \leq -\delta\hat{x}_{\tau_m} \leq (\sqrt{2})^{m}\overline{x} - \hat{x}^*_{\tau_m} \tag{2.224}$$

These can be represented more compactly in the following non-linear form.

$$|\delta\hat{x}_{\tau_m}| \leq \min\{\hat{x}^*_{\tau_m} - (\sqrt{2})^{m}\underline{x}, (\sqrt{2})^{m}\overline{x} - \hat{x}^*_{\tau_m}\} \tag{2.225}$$

The inequalities form a necessary and sufficient condition for (2.218) to hold at the $(m-1)$th level, and thus at all lower levels, if we include corresponding wavelet constraints from the lower levels. This requirement is recursive since it requires knowledge about the upper values, before the lower values can be computed, and is thus not very useful. They may be useful for checking the final solution, but are no more so than the scaling function constraints in terms of constructing a final solution, or improving the value of the cost function. They do provide one useful corollary.

Corollary 8 *If at any point on the tree, a value of $\hat{x}^*_{\tau_m}$ in the optimal solution is at an upper or lower bound, then all values of $\delta\hat{x}_\tau$ on the inclusive subtree from that node must necessarily* *be zero.*

2.5.9 Expanding the set of constraints

The motivation for solving the constrained estimation problem in the multiscale domain is the reduction in computational complexity in certain cases that are typically found in chemical engineering control problems. The principle is to solve a sequence of simpler optimisation subproblems, check for consistency with the original problem, and either conclude optimality for the original problem, or update the modified subproblem. This can be illustrated with a set of problems with bounds on a state.

Consider the convex set A_0 to be the set defined by upper and lower bounds on the state variables at the zeroth level.

$$A_0 = \left\{x | \underline{x}_{L_o} \leq x_j \leq \overline{x}_{L_0}, \forall x_j \in L_0\right\} \quad (2.226)$$

$$K_0 = \{\text{set of constraints at } L_0\} \quad (2.227)$$

$$K_0^a = \{\text{set of constraints at } L_0 \text{ that are active}\} \quad (2.228)$$

Now suppose that this definition holds at higher levels, k, where x_j refers to the scaling function coefficients at the various levels computed as illustrated above, using Haar averaging.

$$A_k = \left\{x | \underline{x}_{L_k} \leq x_j \leq \overline{x}_{L_k}, \forall x_j \in L_k\right\} \quad (2.229)$$

$$K_k = \{\text{set of superconstraints at } L_k\} \quad (2.230)$$

$$K_k^a = \{\text{set of superconstraints at } L_k \text{ that are active}\} \quad (2.231)$$

Consider two problems, and the associated Karush-Kuhn-Tucker conditions for each.

Problem 9 *This is the original formulation*

$$\min_{x_j} \Phi(x_j) + \sum_{k \epsilon K_0} \lambda_k g_k(x_j)$$

has sufficient conditions for solution

$$\begin{aligned}\nabla\Phi(x_j) + \sum_{k\epsilon K_0} \lambda_k \nabla g_k(x_j) &= 0 \\ \lambda_k g_k(x_j) &= 0 \, \forall k \in K_0^a \quad (2.232) \\ \lambda_k &\geq 0 \, \forall k \in K_0^a \\ \{x_j\} &\in A_0 \quad (2.233)\end{aligned}$$

Problem 10 *including constraints at higher levels*

$$\min_{x_j} \Phi(x_j) + \sum_{k\epsilon K_0} \lambda_k g_k(x_j)$$

has sufficient conditions for solution

$$\begin{aligned}\nabla\Phi(x_j) + \sum_{k\epsilon K_0} \lambda_k \nabla g_k(x_j) + \sum_{k\epsilon K_1} \lambda_k \nabla g_k(x_j) &= 0 \\ \lambda_k g_k(x_j) &= 0 \, \forall k \in K_0^a \cup K_1^a \cup \ldots \cup K_m^a \quad (2.234) \\ \lambda_k &\leq 0 \, \forall k \in K_0^a \cup K_1^a \cup \ldots \cup K_m^a \\ \{x_j\} &\in A_0 \quad (2.235)\end{aligned}$$

The above two problems will have the same solutions since the new constraints added are simply restatements of collections of constraints from the older problem.

2.5.10 The implementation of the KKT conditions in the constrained state estimation algorithm

This is a description of the algorithm that is currently used for the constrained state estimation problem. It is summarised at the end of the detailed description.

The first step proceeds exactly as for the unconstrained case, with no increase in computational expense. The initial assumption made is that $\mu = 0$, or that all points computed in the downsweep will be feasible. The upsweep identifies a complete set of dagger variables, $a_\tau^\dagger, c_\tau^\dagger, d_\tau^\dagger$, which remain constant for the remainder of the problem. These variables store all of the information from the state model, the measurement model and the set of measurements.

Essentially, this part is the same as computing an unconstrained solution, and is done once for any problem statement, whereafter the variables are stored until needed.

The first step of the downsweep assumes that no constraints are active, thus the assumption that $\mu^{\dagger}_{\tau} = \mu^{\ddagger}_{\tau} = 0$ is sufficient to identify the top node solution for the unconstrained minimum, x_u, by solving the top node set of dagger equations simultaneously.

$$a^{\dagger}_{\tau}\delta\hat{x}_{\tau} + c^{\dagger}_{\tau}\hat{x}_{\tau} = d^{\dagger}_{\tau} - \mu^{\dagger}_{\tau} \qquad (2.236)$$

$$a^{\ddagger}_{\tau}\delta\hat{x}_{\tau} + c^{\ddagger}_{\tau}\hat{x}_{\tau} = d^{\ddagger}_{\tau} - \mu^{\ddagger}_{\tau} \qquad (2.237)$$

The first downsweep

The goal in the first downsweep is to produce a basic feasible solution that is as close as possible to the unconstrained minimum. This is achieved by progressing as far down the tree as possible, while tracking the unconstrained minimum, and deviating only when a constraint violation is identified. If no constraints are active at the optimum, this step will complete the algorithm, and there will be no difference between this and the unconstrained algorithm discussed earlier.

Starting with the top node, the current scaling function term $\hat{x}_{\tau}$ is compared to each of the constraints derived for the node τ. If the constraints are satisfied, the wavelet coefficient for the pertinent node is computed as for the unconstrained algorithm using the dagger equation for that node.

$$a^{\dagger}_{\tau}\delta\hat{x}_{\tau} + c^{\dagger}_{\tau}\hat{x}_{\tau} = d^{\dagger}_{\tau} - \mu^{\dagger}_{\tau} \qquad (2.238)$$

The pair $\delta\hat{x}_{\tau}$ and $\hat{x}_{\tau}$ are combined using Haar reconstruction to produce the descendant scaling function estimates, $\hat{x}_{\tau\alpha}$ and $\hat{x}_{\tau\beta}$. The algorithm then moves down the tree in parallel as nodes $\tau\alpha$ and $\tau\beta$ become the current nodes down their respective branches. Since $\hat{x}_{\tau}$ is interior, then we know by construction of the constraints that there is a feasible solution on the descendant nodes.

Suppose we discover that an upper bound on a state estimate is violated at node τ. We make the solution at this node feasible by projecting it onto the violated constraint. From the construction of the constraints at higher levels we know that if this value is part of the optimal solution, then all descendant nodes will be at the corresponding bound, and thus we choose for the initial basic feasible solution, the solution on the descendant tree that is at that bound. All descendant wavelet coefficients will be 0. This concludes the computation on the subtree descending from τ.

The parent branch of τ, uniquely defined as the collection of nodes of which τ is a descendant, must be adjusted to account for our adjustment of $\hat{x}_\tau$ in such a way that it does not affect the value of the computed states on the remainder of the tree. This is useful since it allows us to localise constraint violations without having them move the whole solution a long way from the unconstrained minimum. In adjusting the parent branch, we need to make sure that the Haar wavelet decomposition that defines the tree is satisfied at all nodes.

Suppose $\hat{x}_{\tau\beta}$ has a constraint violation for some τ, and define δ as the magnitude of the current violation. Clearly τ is an element of the parent tree. We wish to adjust $\hat{x}_{\tau\beta}$ without affecting the value of $\hat{x}_{\tau\alpha}$.

$$\begin{aligned}
x_{\tau\beta} &= \frac{1}{\sqrt{2}}\left(x_\tau - \delta x_\tau\right) \\
x_{\tau\beta} - \delta &= \frac{1}{\sqrt{2}}\left(x_\tau - \delta x_\tau\right) - \delta \\
&= \frac{1}{\sqrt{2}}\left(\left(x_\tau - \frac{1}{\sqrt{2}}\delta\right) - \left(\delta x_\tau + \frac{1}{\sqrt{2}}\delta\right)\right) \qquad (2.239) \\
x_{\tau\alpha} &= \frac{1}{\sqrt{2}}\left(x_\tau + \delta x_\tau\right) \\
&= \frac{1}{\sqrt{2}}\left(\left(x_\tau - \frac{1}{\sqrt{2}}\delta\right) + \left(\delta x_\tau + \frac{1}{\sqrt{2}}\delta\right)\right) \qquad (2.240)
\end{aligned}$$

This shows that we can adjust the parent nodes without affecting the remainder of the tree. The parent scaling function, $\hat{x}_\tau$, is decreased by $\frac{1}{\sqrt{2}}\delta$ for upper bound violations, and increased by the same amount for lower constraint violations. The parent wavelet function, $\delta\hat{x}_\tau$, is decreased by $\frac{1}{\sqrt{2}}\delta$ for upper bound violations in $\hat{x}_{\tau\beta}$ or lower bound violations in $\hat{x}_{\tau\alpha}$, but increased by $\frac{1}{\sqrt{2}}\delta$ for lower bound violations in $\hat{x}_{\tau\beta}$ and for upper bound violations in $\hat{x}_{\tau\alpha}$. This step is performed for all nodes on the parent branch. Construction of the parent node constraints guarantees that the new values will fall within the bounds of the parent nodes and thus that the solution remains feasible. The dagger equations

$$a_\tau^\dagger \delta\hat{x}_\tau + c_\tau^\dagger \hat{x}_\tau = d_\tau^\dagger - \mu_\tau^\dagger \tag{2.241}$$

will remain satisfied when we adjust the set of $\mu_\tau^\dagger$, which when non-zero indicate the presence of a constraint violation on the subtree from that node.

This approach is repeated in parallel at all nodes on the tree until the zeroth level is reached. At this point we have a set of violated constraints V and a basic feasible solution $\{\hat{x}_{\tau bfs}, \delta\hat{x}_{\tau bfs}\}$. Note that lower bounds can be dealt with in a similar way, simply reversing the signs where necessary.

The μ Conversion

For any basic feasible solution, $\{\hat{x}_{\tau bfs}, \delta\hat{x}_{\tau bfs}\}$ there is a unique set of $\left\{\mu_\tau^\dagger, \mu_{top}^\ddagger\right\}$ that will satisfy the set of dagger equations. These are easily computed from the following since all coefficients have been stored from the problem setup.

$$\mu_\tau^\dagger = a_\tau^\dagger \delta\hat{x}_\tau + c_\tau^\dagger \hat{x}_\tau - d_\tau^\dagger \tag{2.242}$$

For any set of $\left\{\mu_\tau^\dagger, \mu_{top}^\ddagger\right\}$ there is a unique set of $\left\{\mu_\tau, \mu_{top}\right\}$ which can be computed in a single upsweep from the following intermediates. This is clear since the following non-singular equations represent a linear transform. The only non-zero $\mu_\tau^\dagger$ variables will be those on the parent branch of an identified constraint, or on the subtree of such a constraint.

$$\begin{aligned}
\mu_\tau^* &= \mu_\tau' = 0 \;\; \forall \tau \epsilon L_0 && (2.243)\\
\mu_\tau^* &= \frac{1}{\sqrt{2}}(\mu_{\tau\alpha}^* + \mu_{\tau\beta}^*) + \frac{1}{\sqrt{2}}(\mu_{\tau\alpha}' + \mu_{\tau\beta}') && (2.244)\\
\mu_\tau &= \mu_\tau^\dagger + \frac{1}{\sqrt{2}}(\mu_{\tau\alpha}^* - \mu_{\tau\beta}^*) + \frac{1}{\sqrt{2}}(\mu_{\tau\alpha}' - \mu_{\tau\beta}') && (2.245)\\
\mu_\tau' &= -(a_\tau' b_\tau + \frac{1}{2}(c_{\tau\alpha}' - c_{\tau\beta}'))a_\tau^{\dagger -1}\mu_\tau^\dagger && (2.246)\\
\mu_{top} &= \mu_{top}^\ddagger + \frac{1}{\sqrt{2}}(\mu_{\tau\alpha}^* + \mu_{\tau\beta}^*) + \frac{1}{\sqrt{2}}(\mu_{\tau\alpha}' + \mu_{\tau\beta}') && (2.247)
\end{aligned}$$

Recall the definition of μ_τ.

$$\mu_\tau = \mu^T \nabla_x g \tag{2.248}$$

A unique feature of this approach is that we have not specified thus far what set of constraints we are using. The benefit is that we can eliminate any constraints known to be absent from the active set immediately, thus reducing computational complexity. Suppose we choose the set of constraints defined in the original problem - thus we ignore for the moment the constraints higher on the tree that have been used to construct the basic feasible solution. We can eliminate those corresponding to variables not on their bounds and can now convert the artificial tree variables into a set of Lagrange multipliers in the traditional sense.

$$\mu_\tau = \mu^T \nabla_x g \Rightarrow \mu^T = \mu_\tau \nabla_x^T g \left(\nabla_x g \nabla_x^T g\right)^{-1} \tag{2.249}$$

The matrix inversion can be done in advance for the complete set of original constraints, or stored for certain common or large groupings. Note that in an iterative procedure, a pivoting scheme would be more appropriate, and faster Often a solution will initially find an active constraint at a node high up on the tree, which corresponds to a large collection of nodes lower on the tree. A library of high level nodes could be stored for computational efficiency. In practice, the size of the matrix can be reduced considerably since the basic feasible solution indicates which constraints are inactive, and these can be eliminated from the set.

Identification of the new set of constraints

The computed μ^T provides a set of Lagrange multipliers that indicate where improvements can be made to the cost function by relaxing a constraint. If these Lagrange multipliers are all positive, then the optimum has been found and the algorithm terminates. Further, if a group of positive Lagrange multipliers can be identified, this suggests that the super-constraint constructed from the corresponding constraints is active, and that the lower level constraints can be replaced with this single super-constraint. The identification of these groups leads to a set of nodes corresponding to the set of super-constraints, and sundry individual constraints that form the new set of active constraints.

Any negative Lagrange multiplier suggests the existence of a constraint that needs to be relaxed to reduce the value of the cost function. Constraints corresponding to negative Lagrange multipliers are thus discarded for the purposes of finding the next basic feasible solution.

Identification of further feasible solutions

We have a new set of active constraints - and let us suppose there are p of them - and can define an unknown vector μ^T, and a vector $\nabla_x g$ corresponding to these constraints. $\nabla_x g$ for the active constraints will be non-zero only on the nodes of the parent branch of a active constraint, which leads to a computational reduction. $\nabla_x g$ provides a set of coefficients for the unknown μ^T, which can be transformed in an upsweep on the parent part of the tree to a set of dagger coefficients so that at all τ, a constant matrix c can be computed so that the following equation is satisfied.

$$\mu^\dagger_\tau = \mu^T c \tag{2.250}$$

The next basic feasible solution is computed from three types of equations. The p active constraints provide p equations that set the constrained nodes to their specified bounds.

$$\begin{aligned} \overline{x}_{\tau\beta} &= \frac{1}{\sqrt{2}}(x_\tau - \delta x_\tau) && (2.251) \\ \text{or } \overline{x}_{\tau\alpha} &= \frac{1}{\sqrt{2}}(x_\tau + \delta x_\tau) && (2.252) \end{aligned}$$

Each node on the parent branches of the active constraint nodes provides two equations. The top node provides the two equations

$$\begin{aligned} a^\dagger_\tau \delta\hat{x}_\tau + c^\dagger_\tau \hat{x}_\tau &= d^\dagger_\tau - \mu^\dagger_\tau && (2.253) \\ \text{and } a^\ddagger_\tau \delta\hat{x}_\tau + c^\ddagger_\tau \hat{x}_\tau &= d^\ddagger_\tau - \mu^\ddagger_\tau && (2.254) \end{aligned}$$

with $\mu^{\dagger}_{\tau}$ and $\mu^{\ddagger}_{\tau}$ computed from 2.250. Each other node provides two equations.

$$\begin{aligned} a^{\dagger}_{\tau}\delta\hat{x}_{\tau} + c^{\dagger}_{\tau}\hat{x}_{\tau} &= d^{\dagger}_{\tau} - \mu^{\dagger}_{\tau} && (2.255) \\ \text{and } \overline{x}_{\mathcal{T}\beta} &= \frac{1}{\sqrt{2}}\left(x_{\mathcal{T}} - \delta x_{\mathcal{T}}\right) && (2.256) \\ \text{or } \overline{x}_{\mathcal{T}\alpha} &= \frac{1}{\sqrt{2}}\left(x_{\mathcal{T}} + \delta x_{\mathcal{T}}\right) && (2.257) \end{aligned}$$

This is an n equations in n unknowns set of linear equations, since for every active node there is one equation and one unknown, a element of μ^{T}. For every node on the parent tree, which by implication is non-active, there are two equations, and two unknowns, $x_{\mathcal{T}}$ and $\delta x_{\mathcal{T}}$.

The solution of this system provides a new set of μ^{T}, and a corresponding basic feasible solution, which can be used to determine where improvements can be made to the cost function.

Updating the constraint set

The presence of positive elements in μ^{T} indicates constraints that can be relaxed. Superconstraints that need to leave the basis are split into their two component constraints, and further μ^{T} computed. This is repeated until a level on the tree is reached where active constraints are identified. Once the zeroth level is reached, the constraint can be deleted from the active set and the feasible state value accepted.

Stopping condition

The algorithm continues, moving down the tree until the final set of active constraints is identified, and all KKT necessary and sufficient conditions have been satisfied. This includes checking the μ^{T} constructed from the constraints in the original problem for the final set of constraints, since satisfaction of the superconstraints does not indicate satisfaction of the zeroth level constraints.

Here follows a summary of the approach.

Algorithm 11 *The constrained multiscale state estimator.*

1. *Begin at the base of the tree and compute the coefficients* $a^{\dagger}_{\tau}, c^{\dagger}_{\tau}$, *and* $d^{\dagger}_{\tau}$ *in an upsweep similar to the unconstrained multiscale estimator.*

2. *Solve the equations for the top nodes simultaneously for* $\hat{x}_{top}$ *and* $\delta\hat{x}_{top}$.

3. *If the scaling function is within bounds, compute the descendant nodes scaling functions using the Haar reconstruction.*

4. *For each node:*

   (a) *Check state estimation constraints for the scaling functions.*

   (b) *If they are satisfied*

      i. *Compute the wavelet coefficient values for the nodes, using the coefficients computed during the upsweep.*

      ii. *Compute the scaling functions for the descendant nodes using Haar reconstruction.*

   (c) *If they are not satisfied:*

      i. *Force them to the appropriate bound, e.g. upper bound if there is an upper bound violation*

      ii. *Adjust the parent nodes so that the sibling scaling function values are unaffected.*

      iii. *Mark the node as an active bound for later computation.*

      iv. *Set all descendant scaling functions to the corresponding bound to the active node.*

      v. *Set all descendant wavelet coefficients to zero.*

   (d) *Repeat (4) for descendant nodes until the zeroth level is reached.*

5. *Construct a set of Lagrange multipliers corresponding to this basic feasible solution.*

6. *Identify nodes at which superconstraints can be used to simplify the representation.*

7. *Improve the cost function of the basic feasible solution by relaxing constraints with positive Lagrange multipliers, and compute a new basis feasible solution.*

8. *Repeat 5-7 by moving down the tree until the set of constraints remains unchanged and all KKT are satisfied and an optimal basic feasible solution is found.*

2.5.11 Comments on the constrained multiscale estimator

The upsweep of the algorithm is identical to the unconstrained case, as is the initial downsweep to identify the unconstrained solution. The algorithm begins with a small set of superconstraints and increases the set by moving down the tree until a solution is identified that satisfies the Karush-Kuhn-Tucker conditions. At this point the algorithm terminates. The third sufficiency condition is automatically satisfied for the quadratic cost function chosen, as has been shown above.

The algorithm will not improve solution time in all cases, since in the worst case, it will be necessary to solve the problem using the entire set of constraints from the worst possible point in the search space. This corresponds to solutions where there are rapid fluctuations around a constraint.

The algorithm is intended to improve average solution time in the general case. The algorithm is designed for problems where large areas of contiguous active constraints exist - such as an active input constraint for a considerable length of time. The large number of constraints at these time points can be replaced by a single constraint covering the time period.

The solution is guaranteed to converge since it is lower bounded by the unconstrained minimum, always feasible, and thus a monotonically decreasing sequence is created by the reduction in cost function at each constraint replacement. The finite number of constraints guarantees a finite number of steps. There may be problems with cycling, which have not been observed, and these would be dealt with using Bland's rule.

2.6 Prior information and Kalman filters

2.6.1 Multiscale estimator with prior information

The basic version of the multiscale estimator does not use the initial estimate of the left-most node, since it is an unwieldy complication to the algorithm. This is a common approach in state estimation problems [15], [18], [19]. .This section explains how to incorporate prior information into the estimation algorithm, and the computational cost involved in doing so.

The multiscale state estimator cost function becomes:

$$\begin{aligned}
\Phi_P &= \Phi_{orig} + \Phi_p & (2.258)\\
&= \sum_{\theta \epsilon M}((y^{\theta}_{top} - C\hat{x}_{top})^T R^{\theta-1}_{top}(y^{\theta}_{top} - C\hat{x}_{top})\\
&\quad + \sum_{\tau \epsilon T}(\delta y^{\theta}_{\tau} - C\delta\hat{x}_{\tau})^T R^{\theta-1}_{\tau}(\delta y^{\theta}_{\tau} - C\delta\hat{x}_{\tau})) +\\
&\quad + \sum_{\tau \epsilon T}((1 + A_{\tau\alpha})\delta\hat{x}_{\tau} - (1 - A_{\tau\alpha})\hat{x}_{\tau} + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})^T(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_{\tau}(\sqrt{2}B_{\tau\alpha})^{-1}\ldots\\
&\quad \ldots((1 + A_{\tau\alpha})\delta\hat{x}_{\tau} - (1 - A_{\tau\alpha})\hat{x}_{\tau} + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})\\
&\quad + \left(x_{0|-1} - \widehat{x}_0\right)^T P^{-1}_{0|-1}\left(x_{0|-1} - \widehat{x}_0\right) & (2.259)
\end{aligned}$$

The unconstrained Multiscale State Estimator must be modified for this extra term. For the unconstrained problem, the necessary and sufficient conditions for optimality are:

$$\frac{d\Phi_P}{d\hat{x}_{top}} = 0 \qquad (2.260)$$

$$\frac{d\Phi_P}{d\delta\hat{x}_{\tau}} = 0 \ \forall \tau \epsilon T \qquad (2.261)$$

For $\hat{x}_{top}$, this equation reduces to:

$$\frac{d\Phi_P}{d\hat{x}_{top}} = -2\sum_{\theta \epsilon M} C^T R^{\theta-1}_{top}(y^{\theta}_{top} - C\hat{x}_{top}) + \sum_{\sigma \epsilon T}\frac{\partial\Phi_{orig}}{\partial\hat{x}_{\sigma}}\frac{\partial\hat{x}_{\sigma}}{\partial\hat{x}_{top}} + \frac{\partial\Phi_p}{\partial\hat{x}_0}\frac{\partial\hat{x}_0}{\partial\hat{x}_{top}} = 0 \qquad (2.262)$$

and for $\delta\hat{x}_{\tau}$

$$\frac{d\Phi_P}{d\delta\hat{x}_{\tau}} = -2\sum_{\theta \epsilon M} C^T R^{\theta-1}_{\tau}(\delta y^{\theta}_{\tau} - C\delta\hat{x}_{\tau}) + \sum_{\sigma \epsilon T}\frac{\partial\Phi_{orig}}{\partial\hat{x}_{\sigma}}\frac{\partial\hat{x}_{\sigma}}{\partial\delta\hat{x}_{\tau}} + \frac{\partial\Phi_p}{\partial\hat{x}_0}\frac{\partial\hat{x}_0}{\partial\delta\hat{x}_{top}} \qquad (2.263)$$

$$+2(1 + A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_{\tau}(\sqrt{2}B_{\tau\alpha})^{-1}((1 + A_{\tau\alpha})\delta\hat{x}_{\tau} - (1 - A_{\tau\alpha})\hat{x}_{\tau} + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha}) = 0$$

Thus the additional term to be included is

$$\frac{\partial\Phi_P}{\partial\hat{x}_0} = -\,2P^{-1}_{0|-1}\left(x_{0|-1} - \widehat{x}_0\right) \qquad (2.264)$$

Note the complication of including this term in the original formulation of the estimator - the term $\hat{x}_0$ includes a contribution from nodes on the left branch of the tree makes the matrix dense, rather than lower block diagonal. An efficient solution to this problem is to construct the smallest dense matrix possible, which must be inverted, and use this as the starting point for the sparse section of the algorithm. At each level we retain the $\widehat{x}_0$ term and store it with its own set of coefficients, p_τ which behave in a similar manner to the d_τ family of variables

The new algorithm is as follows.

Define a_τ, b_τ and c_τ so that:

$$(\sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} C + a_\tau b_\tau)\delta\hat{x}_\tau + a_\tau c_\tau \hat{x}_\tau = d_\tau + p_\tau \hat{x}_0 \tag{2.265}$$

for the parent tree of $\hat{x}_0$ only, using the original equation for the remaining nodes.

$$\begin{aligned}
a_\tau &= (1 + A_{\tau\alpha}) && (2.266)\\
b_\tau &= (\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2}B_{\tau\alpha})^{-1}(1 + A_{\tau\alpha}) && (2.267)\\
c_\tau &= -(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2}B_{\tau\alpha})^{-1}(1 - A_{\tau\alpha}) && (2.268)\\
d_\tau &= \sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} \delta y_\tau^\theta - (1 + A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} u_{\tau\alpha} \\
&\quad -1/\sqrt{2}(u_{\tau\alpha}^* - u_{\tau\beta}^*) + \sqrt{2^m} P_{0|-1}^{-1} x_{0|-1} && (2.269)\\
p_\tau &= -\sqrt{2^m} P_{0|-1}^{-1} \text{ if } \tau \in \text{parent tree of } x_0 \\
&= 0 \text{ otherwise} && (2.270)
\end{aligned}$$

where m = number of levels between the zeroth level and τ, and the modified definitions are valid on the parent tree of node 0. The remaining nodes use the definitions of the original estimation algorithm.

Defining

$$
\begin{aligned}
a'_\tau &= (1 - A_{\tau\alpha}) && (2.271)\\
u^*_\tau &= -(1 - A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q_\tau^{-1}u_{\tau\alpha} + 1/\sqrt{2}(u^*_{\tau\alpha} + u^*_{\tau\beta}) && (2.272)\\
a^\dagger_\tau \delta\hat{x}_\tau + c^\dagger_\tau \hat{x}_\tau &= d^\dagger_\tau + p^\dagger_\tau \hat{x}_0 && (2.273)\\
c'_\tau &= d'_\tau = p'_\tau = 0 \ \forall \tau \epsilon L_0 && (2.274)\\
c'_\tau &= -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}))a^{\dagger -1}_\tau c^\dagger_\tau + (+a'_\tau c_\tau + \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta})) && (2.275)\\
d'_\tau &= -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}))a^{\dagger -1}_\tau d^\dagger_\tau && (2.276)\\
p'_\tau \widehat{x}_0 &= -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}))a^{\dagger -1}_\tau \hat{x}_0 && (2.277)\\
a^\dagger_\tau &= \sum_{\theta \epsilon M} C^T R^{\theta -1}_\tau C + a_\tau b_\tau - \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta}) && (2.278)\\
c^\dagger_\tau &= a_\tau c_\tau - \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}) && (2.279)\\
d^\dagger_\tau &= d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} - d'_{\tau\beta}) && (2.280)\\
p^\dagger_\tau &= p_\tau - \frac{1}{\sqrt{2}}(p^*_{\tau\alpha} - p^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(p'_{\tau\alpha} - p'_{\tau\beta}) && (2.281)\\
d^*_\tau &= \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} + d^*_{\tau\beta}) + \frac{1}{\sqrt{2}}(d'_{\tau\alpha} + d'_{\tau\beta}) && (2.282)\\
p^*_\tau &= \frac{1}{\sqrt{2}}(p^*_{\tau\alpha} + p^*_{\tau\beta}) + \frac{1}{\sqrt{2}}(p'_{\tau\alpha} + p'_{\tau\beta}) && (2.283)
\end{aligned}
$$

where the right descendant coefficients will typically be zero. For the top node, the only addition is,

$$p^\ddagger_\tau = p_{top} - \frac{1}{\sqrt{2}}(p^*_{\tau\alpha} + p^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(p'_{\tau\alpha} + p'_{\tau\beta}) \qquad (2.284)$$

This produces a simultaneous system for solution once the upsweep has been completed, and corresponds to the simultaneous solution of the top nodes' scaling function and wavelet coefficient. The matrix equation for a tree of depth 3 is given, but systems of different size can be constructed in an identical manner.

$$\begin{bmatrix} a^{\ddagger} & c^{\ddagger} & & & & & -p^{\ddagger} \\ a^{\dagger}_{top} & c^{\dagger}_{top} & & & & & -p^{\dagger}_{top} \\ & & a^{\dagger}_{top\alpha} & c^{\dagger}_{top\alpha} & & & -p^{\dagger}_{top\alpha} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & -1 & & & \\ & & & & a^{\dagger}_{top\alpha\alpha} & c^{\dagger}_{top\alpha\alpha} & -p^{\dagger}_{top\alpha\alpha} \\ & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & & -1 & \\ & & & & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -1 \end{bmatrix} \begin{bmatrix} \widehat{\delta x}_{top} \\ \widehat{x}_{top} \\ \widehat{\delta x}_{top\alpha} \\ \widehat{x}_{top\alpha} \\ \widehat{\delta x}_{top\alpha\alpha} \\ \widehat{x}_{top\alpha\alpha} \\ \widehat{x}_0 \end{bmatrix} = \begin{bmatrix} d^{\ddagger} \\ d^{\dagger}_{top} \\ d^{\dagger}_{top\alpha} \\ 0 \\ d^{\dagger}_{top\alpha\alpha} \\ 0 \\ 0 \end{bmatrix} \tag{2.285}$$

This system can be solved by matrix inversion, and the remainder of the tree can be computed as in the basic algorithm using the equations

$$a^{\dagger}_{\tau}\delta\hat{x}_{\tau} + c^{\dagger}_{\tau}\hat{x}_{\tau} = d^{\dagger}_{\tau} \tag{2.286}$$

with the wavelet reconstruction. This concludes the state estimation algorithm with prior information.

2.6.2 The Kalman filter and the multiscale state estimator

The Kalman Filter for the unconstrained estimation problem provides the motivation for the unconstrained Multiscale State Estimator, in particular in the way of algorithm design.

The Kalman filter can be viewed as the solution to the following estimation problem:

$$\min_{\{\hat{x}_j\}} \left(x_{0|-1} - \hat{x}_0\right)^T P^{-1}_{0|-1} \left(x_{0|-1} - \hat{x}_0\right) + \sum_{j=0..n} (y_j - C\hat{x}_j)^T R_j^{-1} (y_j - C\hat{x}_j) +$$
$$\sum_{j=0..n-1} (\hat{x}_{j+1} - A_j\hat{x}_j - B_j u_j)^T Q_j^{-1} (\hat{x}_{j+1} - A_j\hat{x}_j - B_j u_j) \tag{2.287}$$

The solution is obtained using a basis of $\{\hat{x}_0 ... \hat{x}_n\}$ by setting derivatives with respect to each of the variables to zero. This produces a tridiagonal set of $(n+1)$ linear equations in $(n+1)$ unknowns. The solution of these equations is obtained using Gaussian elimination to produce the Kalman filtered estimate at the *nth* point. Back-substitution produces the Rauch-Tung-Striebel smoothed estimate for all of the intermediate points.

The solution is

$$\frac{\partial \Phi}{\partial \widehat{x}_k} = 0 \ \forall k \tag{2.288}$$

For $k = 0$ this is

$$P_{0|-1}^{-1} \left(x_{0|-1} - \hat{x}_0 \right) + C^T R_0^{-1} (y_0 - C\hat{x}_0) + A_0^T Q_0^{-1} (\hat{x}_1 - A_0 \hat{x}_0 - B_0 u_0) = 0 \tag{2.289}$$

alternately,

$$\left(a_0 + e_0 \right) \hat{x}_0 + b_0 \hat{x}_1 = d_0 + f_0 \tag{2.290}$$

$$a_k = C^T R_k^{-1} C + P_{0|-1}^{-1} \tag{2.291}$$

$$b_k = -A_k^T Q_k^{-1} \tag{2.292}$$

$$d_k = C^T R_k^{-1} y_k + P_{0|-1}^{-1} x_{0|-1} \tag{2.293}$$

$$e_k = A_k^T Q_k^{-1} A_k \tag{2.294}$$

$$f_k = A_k^T Q_k^{-1} B_k u_k \tag{2.295}$$

For $0 < k < n$ this is

$$C^T R_k^{-1} (y_k - C\hat{x}_k) + A_k^T Q_k^{-1} (\hat{x}_{k+1} - A_k \hat{x}_k - B_k u_k) - Q_{k-1}^{-1} (\hat{x}_k - A_{k-1} \hat{x}_{k-1} - B_{k-1} u_{k-1}) = 0 \tag{2.296}$$

alternately,

$$c_k \hat{x}_{k-1} + \left(a_k + e_k \right) \hat{x}_k + b_k \hat{x}_{k+1} = d_k + f_k \tag{2.297}$$

where

$$\begin{aligned}
a_k &= C^T R_k^{-1} C + Q_{k-1}^{-1} && (2.298)\\
b_k &= -A_k^T Q_k^{-1} && (2.299)\\
c_k &= -Q_{k-1}^{-1} A_{k-1} && (2.300)\\
d_k &= C^T R_k^{-1} y_k + Q_{k-1}^{-1} B_{k-1} u_{k-1} && (2.301)\\
e_k &= A_k^T Q_k^{-1} A_k && (2.302)\\
f_k &= -A_k^T Q_k^{-1} B_k u_k && (2.303)
\end{aligned}$$

for $k = n$ this is

$$C^T R_n^{-1}(y_n - C\hat{x}_n) - Q_{n-1}^{-1}(\hat{x}_n - A_{n-1}\hat{x}_{n-1} - B_{n-1}u_{n-1}) = 0 \tag{2.304}$$

alternately

$$c_n\hat{x}_{n-1} + a_n\hat{x}_n = d_n \tag{2.305}$$

defined as for the general k but with

$$b_n = e_k = f_k = 0 \tag{2.306}$$

Gaussian elimination from node 0 to node 1 gives

$$\begin{aligned}
c_1\hat{x}_0 + (a_1 + e_1)\,\hat{x}_1 + b_1\hat{x}_2 - c_1 a_0^{-1}\,(a_0\hat{x}_0 + b_0\hat{x}_1) &= d_1 + f_1 - c_1 a_0^{-1} d_0\\
\left(a_1^\dagger + e_1\right)\hat{x}_1 + b_1^\dagger\hat{x}_2 &= d_1^\dagger + f_1 && (2.307)
\end{aligned}$$

Gaussian elimination from node k to node $k+1$

$$\begin{aligned}
& c_{k+1}\hat{x}_k + (a_{k+1} + e_{k+1})\,\hat{x}_{k+1} + b_{k+1}\hat{x}_{k+2} \\
& -c_{k+1}\left(a_k^{\dagger} + e_k\right)^{-1}\left(\left(a_k^{\dagger} + e_k\right)\hat{x}_k + b_k^{\dagger}\hat{x}_{k+1}\right) \qquad (2.308) \\
= \; & d_{k+1} + f_{k+1} - c_{k+1}a_k^{\dagger-1}\left(d_k^{\dagger} + f_k\right) \qquad (2.309) \\
\left(a_{k+1}^{\dagger} + e_{k+1}\right)\hat{x}_{k+1} + b_{k+1}^{\dagger}\hat{x}_{k+2} = \; & d_{k+1}^{\dagger} + f_{k+1} \qquad (2.310)
\end{aligned}$$

The algorithm has the form

$$\begin{aligned}
a_{k+1}^{\dagger} &= a_{k+1} - c_{k+1}\left(a_k^{\dagger} + e_k\right)^{-1} b_k^{\dagger} \\
b_{k+1}^{\dagger} &= b_{k+1} \\
d_{k+1}^{\dagger} &= d_{k+1} - c_{k+1}a_k^{\dagger-1}\left(d_k^{\dagger} + f_k\right) \qquad (2.311)
\end{aligned}$$

with initial conditions

$$\begin{aligned}
a_0^{\dagger} &= a_0 \\
b_0^{\dagger} &= b_0 \\
d_0^{\dagger} &= d_0 \qquad (2.312)
\end{aligned}$$

At the *nth* node

$$\begin{aligned}
b_n &= 0 \\
a_n^{\dagger}\hat{x}_n &= d_n^{\dagger} \qquad (2.313)
\end{aligned}$$

which is the first step of the back substitution algorithm which gives rise to the complete set of optimal estimates.

This turns out to be identical to the traditional linear Kalman filter in all respects, although the variables used in the Kalman filter need to be teased out of the subsidiary variables.

Gaussian elimination from level 0 to level 1 gives rise to two intermediate variables, $x_{0|0}$,and

$P_{0|0}^{-1}$.

$$x_{0|0} = \left(C^T R_0^{-1} C + P_{0|-1}^{-1}\right)^{-1} \left(C^T R_0^{-1} y_0 + P_{0|-1}^{-1} x_{0|-1}\right) \tag{2.314}$$

For comparison to the Kalman filter, let us begin with the standard form of the Ricatti equation for the discrete time Kalman filter:

$$P_{t|t} = P_{t|t-1} - P_{t|t-1} C_t^T \left(C_t P_{t|t-1} C_t^T + R_t\right)^{-1} C_t P_{t|t-1} \tag{2.315}$$

$$P_{t+1|t} = A_t P_{t|t} A_t^T + Q_t \tag{2.316}$$

and the update and prediction steps for the state estimate:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + P_{t|t-1} C_t^T \left(C_t P_{t|t-1} C_t^T + R_t\right)^{-1} \left(y_t - C_t \hat{x}_{t|t-1}\right) \tag{2.317}$$

$$\hat{x}_{t+1|t} = A_t \hat{x}_{t|t} + B_t u_t \tag{2.318}$$

A useful quantity is the gain matrix:

$$K_t = P_{t|t-1} C_t^T \left(C_t P_{t|t-1} C_t^T + R_t\right)^{-1} \tag{2.319}$$

$$= \left(C_t^T R_t^{-1} C_t + P_{t|t-1}^{-1}\right)^{-1} C_t^T R_t^{-1} \tag{2.320}$$

Lemma 12

$$\left(P_{k|k}^{-1} + A_k^T Q_k^{-1} A_k\right)^{-1} = P_{k|k} \left(P_{k|k} + A_k^{-1} Q_k A_k^{-T}\right)^{-1} A_k^{-1} Q_k A_k^{-T} \tag{2.321}$$

Proof. *Assuming all relevant matrices are invertible*

$$\begin{aligned}
I + A_k^{-1}Q_kA_k^{-T}P_{k|k}^{-1} &= A_k^{-1}Q_kA_k^{-T}P_{k|k}^{-1} + I && (2.322)\\
P_{k|k}P_{k|k}^{-1} + A_k^{-1}Q_kA_k^{-T}P_{k|k}^{-1} &= A_k^{-1}Q_kA_k^{-T}P_{k|k}^{-1} + A_k^{-1}Q_kA_k^{-T}A_k^TQ_k^{-1}A_k && (2.323)\\
\left(P_{k|k} + A_k^{-1}Q_kA_k^{-T}\right)P_{k|k}^{-1} &= A_k^{-1}Q_kA_k^{-T}\left(P_{k|k}^{-1} + A_k^TQ_k^{-1}A_k\right) && (2.324)\\
\left(P_{k|k} + A_k^{-1}Q_kA_k^{-T}\right)P_{k|k}^{-1} &= A_k^{-1}Q_kA_k^{-T}\left(P_{k|k}^{-1} + A_k^TQ_k^{-1}A_k\right) && (2.325)\\
P_{k|k}^{-1}\left(P_{k|k}^{-1} + A_k^TQ_k^{-1}A_k\right)^{-1} &= \left(P_{k|k} + A_k^{-1}Q_kA_k^{-T}\right)^{-1}A_k^{-1}Q_kA_k^{-T} && (2.326)\\
\left(P_{k|k}^{-1} + A_k^TQ_k^{-1}A_k\right)^{-1} &= P_{k|k}\left(P_{k|k} + A_k^{-1}Q_kA_k^{-T}\right)^{-1}A_k^{-1}Q_kA_k^{-T} && (2.327)
\end{aligned}$$

■

Claim 13

$$a_k^{\dagger -1} = P_{k|k} \qquad (2.328)$$

i.e. the recursions are equivalent.

Proof. *by induction. Firstly, it is true for k=0*

$$\begin{aligned}
a_0^{\dagger -1} &= a_0^{-1} && (2.329)\\
&= \left(C^TR_0^{-1}C + P_{0|-1}^{-1}\right)^{-1} && (2.330)\\
&= \left(C^TR_0^{-1}C + P_{0|-1}^{-1}\right)^{-1}P_{0|-1}^{-1}P_{0|-1} && (2.331)\\
&= \left(I - \left(C^TR_0^{-1}C + P_{0|-1}^{-1}\right)^{-1}C^TR_0^{-1}C\right)P_{0|-1} && (2.332)\\
&= \left(I - K_0C\right)P_{0|-1} && (2.333)\\
&= P_{0|0} && (2.334)
\end{aligned}$$

■

Proof. Suppose it is true for k, show that it is true for $k+1$

$$\begin{aligned}
a_{k+1}^{\dagger} &= a_{k+1} - c_{k+1}\left(a_k^{\dagger}+e_k\right)^{-1} b_k^{\dagger} && (2.335)\\
&= C^T R_{k+1}^{-1} C + Q_k^{-1} - \left(-Q_k^{-1}A_k\right)\left(P_{k|k}^{-1}+A_k^T Q_k^{-1} A_k\right)^{-1}\left(-A_k^T Q_k^{-1}\right) && (2.336)\\
&= C^T R_{k+1}^{-1} C + Q_k^{-1} \\
&\quad -Q_k^{-1} A_k P_{k|k}\left(P_{k|k}+A_k^{-1} Q_k A_k^{-T}\right)^{-1} A_k^{-1} Q_k A_k^{-T} A_k^T Q_k^{-1} && (2.337)\\
&= C^T R_{k+1}^{-1} C + Q_k^{-1} - Q_k^{-1} A_k P_{k|k}\left(P_{k|k}+A_k^{-1} Q_k A_k^{-T}\right)^{-1} A_k^{-1} && (2.338)\\
&= C^T R_{k+1}^{-1} C + Q_k^{-1} - Q_k^{-1} A_k P_{k|k} A_k^T\left(A_k P_{k|k} A_k^T + Q_k\right)^{-1} A_k A_k^{-1} && (2.339)\\
&= C^T R_{k+1}^{-1} C + Q_k^{-1} - Q_k^{-1} A_k P_{k|k} A_k^T P_{k+1|k}^{-1} && (2.340)\\
&= C^T R_{k+1}^{-1} C + Q_k^{-1} - Q_k^{-1}\left(P_{k+1|k}-Q_k\right) P_{k+1|k}^{-1} && (2.341)\\
&= C^T R_{k+1}^{-1} C + Q_k^{-1} - Q_k^{-1} + P_{k+1|k}^{-1} && (2.342)\\
&= C^T R_{k+1}^{-1} C + P_{k+1|k}^{-1} && (2.343)\\
a_k^{\dagger -1} &= \left(C^T R_{k+1}^{-1} C + P_{k+1|k}^{-1}\right)^{-1} && (2.344)\\
&= \left(C^T R_{k+1}^{-1} C + P_{k+1|k}^{-1}\right)^{-1} P_{k+1|k}^{-1} P_{k+1|k} && (2.345)\\
&= \left(I - \left(C^T R_{k+1}^{-1} C + P_{k+1|k}^{-1}\right)^{-1} C^T R_{k+1}^{-1} C\right) P_{k+1|k} && (2.346)\\
&= \left(I - K_k C\right) P_{k+1|k} && (2.347)\\
&= P_{k+1|k+1} && (2.348)
\end{aligned}$$

■

This demonstrates that the recursions in $a_k^{\dagger}$ and $P_{k|k}$ are identical.

Lemma 14

$$P_{k|k}^{-1} = C^T R^{-1} C + P_{k|k-1}^{-1}$$

Proof.

$$\begin{aligned}
I &= I \\
&= I + [C^T R^{-1} - C^T R^{-1}] C P_{k|k-1} && (2.349) \\
&= I + [C^T R^{-1} - C^T R^{-1} \left(R + C P_{k|k-1} C^T\right) \left(C P_{k|k-1} C^T + R\right)^{-1}] C P_{k|k-1} && (2.350) \\
&= I - C^T \left(C P_{k|k-1} C^T + R\right)^{-1} C P_{k|k-1} + C^T R^{-1} C P_{k|k-1} \\
&\quad - C^T R^{-1} C P_{k|k-1} C^T \left(C P_{k|k-1} C^T + R\right)^{-1} C P_{k|k-1} && (2.351) \\
&= \left(C^T R^{-1} C + P_{k|k-1}^{-1}\right) \left(P_{k|k-1} - P_{k|k-1} C^T \left(C P_{k|k-1} C^T + R\right)^{-1} C P_{k|k-1}\right) && (2.352) \\
&= \left(C^T R^{-1} C + P_{k|k-1}^{-1}\right) P_{k|k} && (2.353) \\
P_{k|k}^{-1} &= C^T R^{-1} C + P_{k|k-1}^{-1} && (2.354)
\end{aligned}$$

■

Claim 15 *Intermediate estimates for the states can be computed from the equation:*

$$a_k^\dagger x_{k|k} = d_k^\dagger \qquad (2.355)$$

Proof. *by induction*

$$\begin{aligned}
x_{0|0} &= \hat{x}_{0|-1} + P_{0|-1} C^T \left(C P_{0|-1} C^T + R_0\right)^{-1} \left(y_0 - C \hat{x}_{0|0-1}\right) && (2.356) \\
&= \hat{x}_{0|-1} + \left(C_t^T R_t^{-1} C_t + P_{t|t-1}^{-1}\right)^{-1} C_t^T R_t^{-1} \left(y_0 - C_t \hat{x}_{0|0-1}\right) && (2.357) \\
&= \left(C^T R_0^{-1} C + P_{0|-1}^{-1}\right)^{-1} C^T R_0^{-1} y_0 \\
&\quad + \left(C^T R_0^{-1} C + P_{0|-1}^{-1}\right)^{-1} \left(I - C_t^T R_t^{-1} C_t \hat{x}_{0|0-1}\right) && (2.358) \\
&= \left(C^T R_0^{-1} C + P_{0|-1}^{-1}\right)^{-1} C^T R_0^{-1} y_0 + \left(C^T R_0^{-1} C + P_{0|-1}^{-1}\right)^{-1} P_{0|-1}^{-1} \hat{x}_{0|0-1} && (2.359) \\
&= \left(C^T R_0^{-1} C + P_{0|-1}^{-1}\right)^{-1} \left(C^T R_0^{-1} y_0 + P_{0|-1}^{-1} x_{0|-1}\right) && (2.360) \\
&= a_0^{\dagger -1} d_0^\dagger && (2.361)
\end{aligned}$$

$$
\begin{aligned}
x_{k|k} &= \hat{x}_{k|k-1} + P_{k|k-1}C^T\left(CP_{k|k-1}C^T + R_k\right)^{-1}\left(y_k - C_t\hat{x}_{k|k-1}\right) && (2.362)\\
&= \left(C^TR_k^{-1}C + P_{k|k-1}^{-1}\right)^{-1}\left(C^TR_k^{-1}y_k + P_{k|k-1}^{-1}x_{k|k-1}\right) && (2.363)\\
&= a_k^{\dagger -1}\left(C^TR_k^{-1}y_k + P_{k|k-1}^{-1}x_{k|k-1}\right) && (2.364)\\
&= a_k^{\dagger -1}\left(C^TR_k^{-1}y_k + P_{k|k-1}^{-1}\left(A_{k-1}x_{k-1|k-1} + B_{k-1}u_{k-1}\right)\right) && (2.365)\\
&= a_k^{\dagger -1}\begin{pmatrix} C^TR_k^{-1}y_k \\ +\left(A_{k-1}P_{k-1|k-1}A_{k-1}^T + Q_{k-1}\right)^{-1}\left(A_{k-1}x_{k-1|k-1} + B_{k-1}u_{k-1}\right)\end{pmatrix} && (2.366)\\
&= a_k^{\dagger -1}\begin{pmatrix} C^TR_k^{-1}y_k + Q_{k-1}^{-1}\left(A_{k-1}^{-T}P_{k-1|k-1}^{-1}A_{k-1}^{-1} + Q_{k-1}^{-1}\right)^{-1} \\ A_{k-1}^{-T}P_{k-1|k-1}^{-1}A_{k-1}^{-1}\left(A_{k-1}x_{k-1|k-1} + B_{k-1}u_{k-1}\right)\end{pmatrix} && (2.367)\\
&= a_k^{\dagger -1}\begin{pmatrix} C^TR_k^{-1}y_k + Q_{k-1}^{-1}A_{k-1}\left(P_{k-1|k-1}^{-1} + A_{k-1}^TQ_{k-1}^{-1}A_{k-1}\right)^{-1} \\ \left(P_{k-1|k-1}^{-1}\left(x_{k-1|k-1} + A_{k-1}^{-1}B_{k-1}u_{k-1}\right)\right)\end{pmatrix} && (2.368)\\
&= a_k^{\dagger -1}\begin{pmatrix} C^TR_k^{-1}y_k + Q_{k-1}^{-1}A_{k-1}\left(P_{k-1|k-1}^{-1} + A_{k-1}^TQ_{k-1}^{-1}A_{k-1}\right)^{-1} \\ \begin{pmatrix} P_{k-1|k-1}^{-1}\left(x_{k-1|k-1} + A_{k-1}^{-1}B_{k-1}u_{k-1}\right) \\ -A_{k-1}^TQ_{k-1}^{-1}B_{k-1}u_{k-1} + A_{k-1}^TQ_{k-1}^{-1}B_{k-1}u_{k-1}\end{pmatrix}\end{pmatrix} && (2.369)\\
&= a_k^{\dagger -1}\begin{pmatrix} C^TR_k^{-1}y_k + Q_{k-1}^{-1}A_{k-1}\left(P_{k-1|k-1}^{-1} + A_{k-1}^TQ_{k-1}^{-1}A_{k-1}\right)^{-1} \\ \begin{pmatrix} P_{k-1|k-1}^{-1}x_{k-1|k-1} - A_{k-1}^TQ_{k-1}^{-1}B_{k-1}u_{k-1} \\ +\left(P_{k-1|k-1}^{-1} + A_{k-1}^TQ_{k-1}^{-1}A_{k-1}\right)A_{k-1}^{-1}Q_{k-1}Q_{k-1}^{-1}B_{k-1}u_{k-1}\end{pmatrix}\end{pmatrix} && (2.370)\\
&= a_k^{\dagger -1}\begin{pmatrix} C^TR_k^{-1}y_k + Q_{k-1}^{-1}B_{k-1}u_{k-1} \\ +Q_{k-1}^{-1}A_{k-1}\left(P_{k-1|k-1}^{-1} + A_{k-1}^TQ_{k-1}^{-1}A_{k-1}\right)^{-1} \\ \left(P_{k-1|k-1}^{-1}x_{k-1|k-1} - A_{k-1}^TQ_{k-1}^{-1}B_{k-1}u_{k-1}\right)\end{pmatrix} && (2.371)\\
&= a_k^{\dagger -1}\left(d_k - c_k\left(a_{k-1}^{\dagger} + e_{k-1}\right)^{-1}\left(d_{k-1}^{\dagger} + f_{k-1}\right)\right) && (2.372)\\
&= a_k^{\dagger -1}d_k^{\dagger}
\end{aligned}
$$

■

The smoothing step for the algorithm is obtained by back-substitution of equations

$$
\begin{aligned}
a_n^{\dagger}\hat{x}_n &= d_n^{\dagger}\\
\left(a_k^{\dagger} + e_k\right)\hat{x}_k + b_k^{\dagger}\hat{x}_{k+1} &= d_k^{\dagger} + f_k
\end{aligned}
$$

where at each step it is assumed that $\hat{x}_{k+1}$ has been computed at the previous step. Substituting this gives the following.

Claim 16 *The back-substitution is equivalent to the Rauch-Tung-Striebel smoother*

$$a_n^\dagger \hat{x}_n = d_n^\dagger \tag{2.373}$$

$$\left(a_k^\dagger + e_k\right) \hat{x}_k + b_k^\dagger \hat{x}_{k+1} = d_k^\dagger + f_k \tag{2.374}$$

$$\equiv$$

$$\hat{x}_k = A_k^{-1} \hat{x}_{k+1} - A_k^{-1} B_k u_k - A_k^{-1} Q_k P_{k+1|k}^{-1} \left(\hat{x}_{k+1} - x_{k+1|k}\right) \tag{2.375}$$

Proof.

$$
\begin{aligned}
\left(P_{k|k}^{-1} + A_k^T Q_k^{-1} A_k\right) \hat{x}_k &= A_k^T Q_k^{-1} \hat{x}_{k+1} + P_{k|k}^{-1} x_{k|k} - A_k^T Q_k^{-1} B_k u_k && (2.376)\\
&= A_k^T Q_k^{-1} \hat{x}_{k+1} + P_{k|k}^{-1} A_k^{-1} \left(x_{k+1|k} - B_k u_k\right) - A_k^T Q_k^{-1} B_k u_k && (2.377)\\
&= A_k^T Q_k^{-1} \hat{x}_{k+1} + P_{k|k}^{-1} A_k^{-1} x_{k+1|k} \\
&\quad - P_{k|k}^{-1} A_k^{-1} B_k u_k - A_k^T Q_k^{-1} B_k u_k && (2.378)\\
&= A_k^T \left(A_k^{-T} P_{k|k}^{-1} A_k^{-1} + Q_k^{-1}\right) \left(A_k^{-T} P_{k|k}^{-1} A_k^{-1} + Q_k^{-1}\right)^{-1} \\
&\quad \begin{pmatrix} Q_k^{-1} \hat{x}_{k+1} + A_k^{-T} P_{k|k}^{-1} A_k^{-1} x_{k+1|k} \\ -A_k^{-T} P_{k|k}^{-1} A_k^{-1} B_k u_k - Q_k^{-1} B_k u_k \end{pmatrix} && (2.379)\\
&= \left(P_{k|k}^{-1} + A_k^T Q_k^{-1} A_k\right) A_k^{-1} \left(A_k^{-T} P_{k|k}^{-1} A_k^{-1} + Q_k^{-1}\right)^{-1} \\
&\quad \begin{pmatrix} A_k^{-T} P_{k|k}^{-1} A_k^{-1} \hat{x}_{k+1} + Q_k^{-1} \hat{x}_{k+1} \\ -A_k^{-T} P_{k|k}^{-1} A_k^{-1} B_k u_k - Q_k^{-1} B_k u_k \\ -A_k^{-T} P_{k|k}^{-1} A_k^{-1} \left(\hat{x}_{k+1} - x_{k+1|k}\right) \end{pmatrix} && (2.380)\\
\left(P_{k|k}^{-1} + A_k^T Q_k^{-1} A_k\right) \hat{x}_k &= \left(P_{k|k}^{-1} + A_k^T Q_k^{-1} A_k\right) A_k^{-1} && (2.381)\\
&\quad \begin{pmatrix} \hat{x}_{k+1} - B_k u_k \\ -\left(A_k^{-T} P_{k|k}^{-1} A_k^{-1} + Q_k^{-1}\right)^{-1} A_k^{-T} P_{k|k}^{-1} A_k^{-1} \left(\hat{x}_{k+1} - x_{k+1|k}\right) \end{pmatrix} \\
\hat{x}_k &= A_k^{-1} \hat{x}_{k+1} - A_k^{-1} B_k u_k && (2.382)\\
&\quad - A_k^{-1} \left(A_k^{-T} P_{k|k}^{-1} A_k^{-1} + Q_k^{-1}\right)^{-1} A_k^{-T} P_{k|k}^{-1} A_k^{-1} \left(\hat{x}_{k+1} - x_{k+1|k}\right) \\
&= A_k^{-1} \hat{x}_{k+1} - A_k^{-1} B_k u_k \\
&\quad - A_k^{-1} Q_k \left(A_k P_{k|k} A_k^T + Q_k\right)^{-1} \left(\hat{x}_{k+1} - x_{k+1|k}\right) && (2.383)\\
\hat{x}_k &= A_k^{-1} \hat{x}_{k+1} - A_k^{-1} B_k u_k - A_k^{-1} Q_k P_{k+1|k}^{-1} \left(\hat{x}_{k+1} - x_{k+1|k}\right) && (2.384)
\end{aligned}
$$

■

This is the standard form of the discrete Rauch-Tung-Striebel backwards smoother.

This demonstrates that the Kalman filter and the proposed algorithm solve the same optimization problem, and that all Kalman filter estimates can be easily generated from the intermediate variables from the algorithm.

Since the Kalman filtered estimate is the solution to an optimisation problem, we can compare it to the multiscale state estimator in terms of the cost functions, and speed of solution. It is meaningless to use "correctness of solution" since the Kalman filtered solution will be suboptimal for the cost function of the multiscale estimator and vice versa.

2.6.3 Comparison of Kalman filter and multiscale state estimator cost functions

The Multiscale State Estimator cost function is

$$\begin{aligned}\Phi_P \;=\; & \sum_{\theta\epsilon M}((y^{\theta}_{top} - C\hat{x}_{top})^T R^{\theta-1}_{top}(y^{\theta}_{top} - C\hat{x}_{top}) + \sum_{\tau\epsilon T}(\delta y^{\theta}_{\tau} - C\delta\hat{x}_{\tau})^T R^{\theta-1}_{\tau}(\delta y^{\theta}_{\tau} - C\delta\hat{x}_{\tau})) + \\ & + \sum_{\tau\epsilon T}((1+A_{\tau\alpha})\delta\hat{x}_{\tau} - (1-A_{\tau\alpha})\hat{x}_{\tau} + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})^T(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_{\tau}(\sqrt{2}B_{\tau\alpha})^{-1}\ldots \\ & \ldots((1+A_{\tau\alpha})\delta\hat{x}_{\tau} - (1-A_{\tau\alpha})\hat{x}_{\tau} + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha}) \\ & + \left(x_{0|-1} - \hat{x}_0\right)^T P^{-1}_{0|-1}\left(x_{0|-1} - \hat{x}_0\right) \end{aligned} \tag{2.385}$$

and the Kalman filter cost function is

$$\begin{aligned}\Phi_K \;=\; & \left(x_{0|-1} - \hat{x}_0\right)^T P^{-1}_{0|-1}\left(x_{0|-1} - \hat{x}_0\right) + \sum_{\theta\epsilon M}\sum_{j=0..n}(y^{\theta}_j - C\hat{x}_j)^T R^{\theta-1}_j(y^{\theta}_j - C\hat{x}_j) + \\ & \sum_{j=0..n-1}(\hat{x}_{j+1} - A_j\hat{x}_j - B_ju_j)^T Q^{-1}_j(\hat{x}_{j+1} - A_j\hat{x}_j - B_ju_j) \end{aligned} \tag{2.386}$$

The measurement terms and the prior information terms are identical in both cost functions, thus the only difference is in the term containing the error in the dynamic system

$$\hat{x}_{j+1} - A_j\hat{x}_j - B_ju_j = w_j \tag{2.387}$$

$$(1+A_{\tau\alpha})\delta\hat{x}_{\tau} - (1-A_{\tau\alpha})\hat{x}_{\tau} + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha} = w_{\tau\alpha} \tag{2.388}$$

Thus the difference in the cost functions is

$$\begin{aligned}
&\sum_{j=0..n-1} w_j^T Q_j^{-1} w_j - \sum_{\tau \epsilon T} w_{\tau\alpha}^T Q_\tau^{-1} w_{\tau\alpha} \\
&= w_J^T Q_J^{-1} w_J - w_J^T M^T Q_\tau^{-1} M w_J \\
&= w_J^T \left(Q_J^{-1} - M^T Q_\tau^{-1} M \right) w_J \qquad (2.389)
\end{aligned}$$

where w_J is the vector of modelling errors at the zeroth level, Q_J^{-1} the block diagonal matrix of inverse covariances of the modelling errors, Q_τ^{-1} the block diagonal matrix of inverse covariances of the modelling errors on the tree and M, the modified hat transform matrix.

The two solutions will be identical for very small problems. For a single node the problems are identical since there is no state transition model and consequently no state transition modelling error. For a two node problem, the cost functions are also identical, with the state transition error in both cases being $w_0^T Q_0^{-1} w_0$. For the four node problem (for Q=1), the two cost functions diverge:

$$\begin{aligned}
KF &= w_0^2 + w_1^2 + w_2^2 && (2.390) \\
MSE &= w_0^2 + w_2^2 + w_{0/2}^T Q_{0/2}^{-1} w_{0/2} \\
&= w_0^2 + w_2^2 + (Aw_0 + (1+A)w_1 + w_2)^T \\
&\quad (2(1+A)^T(1+A) - 2A)^{-1} (Aw_0 + (1+A)w_1 + w_2) && (2.391) \\
&\approx w_0^2(1 + (1+A^{-1})^{-1}(1+A^{-1})^{-T}) + w_2^2 && (2.392) \\
&\quad + \frac{1}{2}(1 - 2(1+A^{-1})^{-1}(1+A^{-1})^{-T}) w_1^2 && (2.393)
\end{aligned}$$

There will be cross terms, but in this approximation, we can ignore them since $E[w_j w_k] = \delta_{jk}$. It turns out that higher up the tree, the even terms from the zeroth level are more heavily weighted, but the weight is spread out evenly over each level - there is no weight given to earlier or later points. The multiscale state estimator tries to spread the energy evenly over all levels of the tree by weighting each the same way. If this assumption is incorrect, then Q should be chosen to include any additional information we have on the problem.

Bibliography

[1] "Special Issue on Wavelet Transforms and Multiresoution Signal Analysis," 1992.

[2] A., Haar. "Zur Theorie der Orthogonalen Funktionensysteme," *Mathematische Annalen*, *69*:331–371 (1910).

[3] A. Benveniste, R. Nikoukhah, A. Willsky. "Multiscale system theory," *IEEE Transactions on circuits and systems-1: Fundamental theory and Applications*, *41*(1) (1994).

[4] Bazaraa, M., H. Sherali C. Shetty. *Nonlinear Programming*. Wiley, 1993.

[5] Chou, K.C. *A stochastic approach to multiscale signal processing*. PhD dissertation, Massachusetts Institute of Technology, 1991.

[6] C.K., Chui. *An Introduction to Wavelets*. Academic Press, 1992.

[7] C.K., Chui, editor. *Wavelets: A Tutorial in Theory and Applications*. Academic Press, 1992.

[8] Daubechies, I. "The wavelet transform, time-freqency localization and signal analysis," *IEEE Trans. Inf. Theory*, *36*:961 (1990).

[9] Daubechies, I. *Ten lectures on wavelets*. SIAM, 1992.

[10] Fieguth, P.W. *Application of multiscale estimation to large scale multidimensional imaging and remote sensing problems*. PhD dissertation, Massachusetts Institute of Technology, 1995.

[11] G., Strang. "Wavelets and Dilation Equations: A Brief Introduction," *SIAM Review*, *31*:614–627 (1989).

[12] Geo. Stephanopoulos, O. Karsligil. *Multi-scale model predictive control*. Technical Report, MIT-LISPE, 1998.

[13] Irving, W. *Multiscale stochastic realization and model identification with applications to large-scale estimation problems*. PhD dissertation, Massachusetts Institute of Technology, 1995.

[14] Jawerth, B. and W. Sweldens. "An Overview of Wavelet Based Multiresolution Analysis," *Technical Report, University of South Carolina* (1991).

[15] Johansson, R. *System Modelling and Identification*. Number ISBN 0-13-482308-7, Prentice-Hall, Inc, 1993.

[16] Luettgen, M. *Image Processing with multiscale stochastic models*. PhD dissertation, Massachusetts Institute of Technology, 1993.

[17] Mallat, S. G. "A theory for multiresolution signal decomposition: the wavelet representation," *IEEE Trans on PAMI*, *11*:674 (1989).

[18] Rawlings, J.B. and K.R. Muske. "The Stability of Constrained Receeding Horizon Control," *IEEE Transactions on Automatic Control*, *38*(10):1512–1516 (1993).

[19] Robertson, D.L. and J.B. Rawlings. "A Moving Horizon-Based Approach for Least Squares State Estimation," *AICHE Journal*, *42*:2209–2223 (1994).

[20] Strang, G. and T. Nguyen. *Wavelets and filter banks*. Wellesley-Cambridge Press, 1996.

Chapter 3

Uncertainty Aspects

3.1 Introduction

In this chapter, we develop the uncertainty analysis associated with the state estimates generated by the algorithm in Chapter 2. An uncertainty analysis cannot be done without making some assumptions about the structure of the uncertainty. In this context, the uncertainty that we estimate in the state estimate is only as accurate as our assumptions about the underlying model. We begin by defining the error in the state estimate (subsequently abbreviated to estimation error) in terms of the state estimate itself, and in the assumed underlying process. The unconstrained algorithm is rewritten in terms of the estimation errors which are represented explicitly in terms of the underlying uncertainties in the model. If we provide a model for the underlying uncertainties in terms of probability distribution functions, then this is sufficient information to produce a probability distribution function in terms of the estimation error itself.

We explore three different approaches to the construction of probability functions in this chapter. We investigate (a), the approach of polynomial chaos expansions, (b), the approach of direct integration of the underlying probability distribution functions, and (c), a Monte Carlo approach for the generation of the distributions. The chapter concludes with an illustration of these approaches using a fifth Rayleigh distribution, primarily to illustrate the distribution of uncertainty on the tree, but also to illustrate the propagation of a different distribution through the Haar wavelet transform.

Further approaches to eliminate outliers involve median based estimators, such as those presented by Huber, [2]. These provide an increase in robustness for parameter estimation algorithms, and are useful for small numbers of parameters, however for the state estimation problem, they become computationally intractable quickly.

3.2 Modifying the multiscale state estimation algorithm

The state estimation algorithm derived in Chapter 2 shows how measurements, y, are used to generate a set of optimal state estimates $\hat{x}$. The state, x, is governed by a process equation $x = f(x, u, w)$, where u are the known inputs, and w are used to describe any unknown inputs, or unknown dynamics in the process. The measurements, y, are governed by a measurement equation $y = g(x, v)$, where v represents the uncertainty in the measurement process.

The uncertainties in the states, and consequently the measurements, can be computed in a similar way if we have access to an assumed model for the two underlying uncertainty processes, v and w. The multiscale state estimator makes no assumptions about the probability distribution functions of the underlying uncertainties - it simply provides a weight for each variable in the cost function. Because of this, we can compute error statistics for a far larger family of probability distributions.

For the purposes of this chapter, it is assumed that the underlying physical process consists of the following first order autoregressive process, with model uncertainty, and exogenous input.

$$x(t+1) = Ax(t) + Bu(t) + w(t) \tag{3.1}$$

Clearly, we do not have access to these underlying states, only measurements of them, and thus the state, x, must be kept distinct from the estimation of this state, $\hat{x}$. The following section will derive the estimation error form of the algorithm, that corresponds to the solution of the multiscale state estimator. Ultimately, we will need to incorporate prior information about these measurements, however the bulk of this derivation is independent of this restriction.

3.3 Estimation error on the multiscale tree

The state estimation error is defined as the difference between the true, underlying state and the optimal state estimate.

$$e_t = x_t - \hat{x}_t \tag{3.2}$$

Clearly the wavelet transforms of the estimation error will be identical, since the Haar transform is linear.

$$\begin{aligned} e_\tau &= x_\tau - \hat{x}_\tau && (3.3) \\ \delta e_\tau &= \delta x_\tau - \delta\hat{x}_\tau && (3.4) \end{aligned}$$

The measurement and modelling equations transform from a basis of state estimates to one of estimation errors. The state transition equation in tree form is:

$$\left(I + A_{\tau\alpha}\right)\left(\delta x_\tau\right) = \left(I - A_{\tau\alpha}\right) x_\tau - \sqrt{2} B u_{\tau\alpha} - \sqrt{2} w_{\tau\alpha} \tag{3.5}$$

The state transition equation is written in error form by substitution.

$$\begin{aligned} \left(I + A_{\tau\alpha}\right)\left(\delta e_\tau + \delta\hat{x}_\tau\right) &= \left(I - A_{\tau\alpha}\right)\left(e_\tau + \hat{x}_\tau\right) - \sqrt{2} B u_{\tau\alpha} - \sqrt{2} w_{\tau\alpha} \\ \left(I + A_{\tau\alpha}\right)\delta\hat{x}_\tau - \left(I - A_{\tau\alpha}\right)\hat{x}_\tau + \sqrt{2} B u_{\tau\alpha} &= -\left(I + A_{\tau\alpha}\right)\delta e_\tau + \left(I - A_{\tau\alpha}\right) e_\tau - \sqrt{2} w_{\tau\alpha} && (3.6) \end{aligned}$$

The measurement equation transforms similarly

$$\begin{aligned} y_\tau &= C x_\tau + v_\tau \\ &= C\left(e_\tau + \hat{x}_\tau\right) + v_\tau \\ y_\tau - C\hat{x}_\tau &= C e_\tau + v_\tau && (3.7) \\ \text{in wavelet terms } \delta y_\tau - C\delta\hat{x}_\tau &= C\delta e_\tau + \delta v_\tau && (3.8) \end{aligned}$$

The multiscale estimation algorithm can be rewritten in terms of the estimation errors of the multiscale state estimates. The goal of this approach is to find the optimal estimation errors.

We begin at the point where the derivative of the following cost function is about to be taken.

$$\begin{aligned}
&\min_{\{\hat{x}_{top},\delta\hat{x}_\tau\}} \sum_{\theta\epsilon M}((y^\theta_{top} - C\hat{x}_{top})^T R^{\theta-1}_{top}(y^\theta_{top} - C\hat{x}_{top}) + \sum_{\tau\epsilon T}(\delta y^\theta_\tau - C\delta\hat{x}_\tau)^T R^{\theta-1}_\tau(\delta y^\theta_\tau - C\delta\hat{x}_\tau)) + \\
&+\sum_{\tau\epsilon T}((1 + A_{\tau\alpha})\delta\hat{x}_\tau - (1 - A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})^T(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_\tau(\sqrt{2}B_{\tau\alpha})^{-1}\ldots \\
&\ldots((1 + A_{\tau\alpha})\delta\hat{x}_\tau - (1 - A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha}) \qquad (3.9)
\end{aligned}$$

3.3.1 The derivative conditions

For the unconstrained problem, the necessary and sufficient conditions for optimality are the derivatives with respect to each basis variable, the set of $\hat{x}_{top}, \delta\hat{x}_\tau$, set equal to zero.

$$\begin{aligned}
\frac{d\Phi}{d\hat{x}_{top}} &= 0 && (3.10) \\
\frac{d\Phi}{d\delta\hat{x}_\tau} &= 0 \;\forall\tau\epsilon T && (3.11)
\end{aligned}$$

For $\hat{x}_{top}$, this equation can be rewritten in terms of estimation errors.

$$\begin{aligned}
\frac{d\Phi}{d\hat{x}_{top}} &= -2\sum_{\theta\epsilon M} C^T R^{\theta-1}_{top}(y^\theta_{top} - C\hat{x}_{top}) + \sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\hat{x}_{top}} && (3.12) \\
&= -2\sum_{\theta\epsilon M} C^T R^{\theta-1}_{top}(v^\theta_{top} + Ce_{top}) + \sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\hat{x}_{top}} && (3.13) \\
&= 0 && (3.14)
\end{aligned}$$

and for $\delta\hat{x}_\tau$, the substitution is as follows.

$$\begin{aligned}
\frac{d\Phi}{d\delta\hat{x}_\tau} &= -2\sum_{\theta\epsilon M} C^T R^{\theta-1}_\tau(\delta y^\theta_\tau - C\delta\hat{x}_\tau) + \sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} && (3.15) \\
&\quad +2(1 + A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_\tau(\sqrt{2}B_{\tau\alpha})^{-1}((1 + A_{\tau\alpha})\delta\hat{x}_\tau - (1 - A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha}) \\
&= -2\sum_{\theta\epsilon M} C^T R^{\theta-1}_\tau(\delta v^\theta_\tau + C\delta e_\tau) + \sum_{\sigma\epsilon T}\frac{\partial\Phi}{\partial\hat{x}_\sigma}\frac{\partial\hat{x}_\sigma}{\partial\delta\hat{x}_\tau} && (3.16) \\
&\quad +2(1 + A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_\tau(\sqrt{2}B_{\tau\alpha})^{-1}(-\left(I + A_{\tau\alpha}\right)\delta e_\tau + \left(I - A_{\tau\alpha}\right)e_\tau - \sqrt{2}w_{\tau\alpha}) \\
&= 0 && (3.17)
\end{aligned}$$

The scaling function terms contribute terms of the form:

$$\frac{\partial \Phi}{\partial \hat{x}_\tau} = -2(1 - A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2}B_{\tau\alpha})^{-1} (-\left(I + A_{\tau\alpha}\right) \delta e_\tau + \left(I - A_{\tau\alpha}\right) e_\tau - \sqrt{2} w_{\tau\alpha}) \tag{3.18}$$

The system of equations in terms of estimation error is structurally identical to the system for the state equations. It follows that the estimation errors corresponding to the optimal state estimate can be obtained from an iterative algorithm that is structurally identical to the algorithm for the optimal state estimates. Everywhere that $\hat{x}$ appears in the original formulation, there is a $-e$ in the final formulation. The same holds for (u, w) and (y, v) pairs. Thus iteration will proceed in an identical manner with the following recursion.

The following three substitutions are necessary.

$$\hat{x} \longrightarrow -e \tag{3.19}$$

$$u \longrightarrow -w \tag{3.20}$$

$$y \longrightarrow v \tag{3.21}$$

Algorithm 17 *The complete algorithm for the error in the unconstrained multiscale state estimator for the driven case with multiple measurement sets is given below. The downsweep involves the computation of the following variables, recursively until the top node of the tree is reached.*

Initial Conditions

$$c'_\tau = d'_\tau = d^*_\tau = 0 \;\; \forall \tau \epsilon L_0 \tag{3.22}$$

$$w^*_\tau = 0 \; \forall \tau \epsilon L_0 \tag{3.23}$$

Primary Variables

$$a_\tau = (1 + A_{\tau\alpha}) \tag{3.24}$$

$$b_\tau = (\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2}B_{\tau\alpha})^{-1}(1 + A_{\tau\alpha}) \tag{3.25}$$

$$c_\tau = -(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} (\sqrt{2}B_{\tau\alpha})^{-1}(1 - A_{\tau\alpha}) \tag{3.26}$$

$$d_\tau = \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} \delta v_\tau^\theta - (1 + A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} w_{\tau\alpha} - \frac{1}{\sqrt{2}}(w^*_{\tau\alpha} - w^*_{\tau\beta}) \tag{3.27}$$

$$d_{top} = \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} v_{top}^\theta - (1 - A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} w_{\tau\alpha} - \frac{1}{\sqrt{2}}(w^*_{\tau\alpha} + w^*_{\tau\beta}) \tag{3.28}$$

Recursive Variables

$$w^*_\tau = -(1 - A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} w_{\tau\alpha} + \frac{1}{\sqrt{2}}(w^*_{\tau\alpha} + w^*_{\tau\beta}) \tag{3.29}$$

$$d^*_\tau = \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} + d^*_{\tau\beta}) + \frac{1}{\sqrt{2}}(d'_{\tau\alpha} + d'_{\tau\beta}) \tag{3.30}$$

$$a^\dagger_\tau = \sum_{\theta\epsilon M} C^T R_\tau^{\theta-1} C + a_\tau b_\tau - \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta}) \tag{3.31}$$

$$c^\dagger_\tau = a_\tau c_\tau - \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}) \tag{3.32}$$

$$d^\dagger_\tau = d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} - d'_{\tau\beta}) \tag{3.33}$$

$$a'_\tau = (1 - A_{\tau\alpha}) \tag{3.34}$$

$$c'_\tau = -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}))a_\tau^{\dagger-1} c^\dagger_\tau + (+a'_\tau c_\tau + \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta})) \tag{3.35}$$

$$d'_\tau = -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}))a_\tau^{\dagger-1} d^\dagger_\tau \tag{3.36}$$

Scaling Function Variables (usually only used at the top node)

$$a^\ddagger_\tau = -a'_\tau b_\tau - \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta}) \tag{3.37}$$

$$c^\ddagger_\tau = \sum_{\theta\epsilon M} C^T R_{top}^{\theta-1} C - a'_\tau c_\tau - \frac{1}{2}(c'_{\tau\alpha} + c'_{\tau\beta}) \tag{3.38}$$

$$d^\ddagger_\tau = d_{top} - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} + d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} + d'_{\tau\beta}) \tag{3.39}$$

The top node error scaling function and error wavelet coefficients are solved simultaneously, to *obtain, e_{top} and δe_{top} from*

$$\begin{aligned} a^{\dagger}_{top}\delta e_{top} + c^{\dagger}_{top} e_{top} &= d^{\dagger}_{top} &\quad (3.40)\\ a^{\ddagger}_{top}\delta e_{top} + c^{\ddagger}_{top} e_{top} &= d^{\ddagger}_{top} &\quad (3.41)\end{aligned}$$

The downsweep involves a wavelet reconstruction from the upper levels for which optimal state estimates have been computed, followed by a backsubstitution into the dagger equations.

$$\begin{aligned} e_{\tau\alpha} &= \frac{1}{\sqrt{2}}\left(e_{\tau} + \delta e_{\tau}\right)\\ e_{\tau\beta} &= \frac{1}{\sqrt{2}}\left(e_{\tau} - \delta e_{\tau}\right)\\ a^{\dagger}_{\tau}\delta e_{\tau} + c^{\dagger}_{\tau} e_{\tau} &= d^{\dagger}_{\tau} &\quad (3.42)\end{aligned}$$

This completes the calculation of the estimation error in all state variables on the tree, both wavelet and scaling functions, in terms of d, which we will show implies that they can be expressed in terms of the complete set of $\{w_k\}$ and $\{v_k\}$ at the zeroth level. At this point, no assumptions have been made about the structure of the uncertainty distributions.

3.4 Calculation of error statistics at the top node

We can assume that all model uncertainty is included in w and v. This is equivalent to assuming that the model parameters, a, b, c, are either known exactly, or that their uncertainty can be expressed as a probability distribution function in the w parameter, and that the R and Q, are meaningfully specified. The state uncertainty at the top nodes is an affine function of $d^{\dagger}_{top}$ and $d^{\ddagger}_{top}$. Explicitly,

$$\begin{aligned} \begin{bmatrix} e_{top} \\ \delta e_{top} \end{bmatrix} &= \begin{bmatrix} c^{\dagger}_{top} & a^{\dagger}_{top} \\ c^{\ddagger}_{top} & a^{\ddagger}_{top} \end{bmatrix}^{-1} \begin{bmatrix} d^{\dagger}_{top} \\ d^{\ddagger}_{top} \end{bmatrix} &\quad (3.43)\\ &= \widetilde{A}^{-1} \begin{bmatrix} d^{\dagger}_{top} \\ d^{\ddagger}_{top} \end{bmatrix} &\quad (3.44)\end{aligned}$$

and since the matrix is constant, the function is affine.

Now consider $d^{\dagger}_{top}$ and $d^{\ddagger}_{top}$ as functions of w and v. This point will be expanded upon later.

3.5 Propagation of uncertainty down the tree.

Claim 18 *$e_{top} = d^T_{w,topsc,e}w + d^T_{v,topsc,e,}v$ where d_w and d_v are known constant matrices.*

Since we know that the uncertainty in the top node can be expressed as a linear combination of the uncertainties at the zeroth level, we can compute similar coefficients for lower nodes on the tree.

$$\begin{aligned}
e_{top} &= d^T_w w + d^T_v v \\
\delta e_\tau &= a^{\dagger -1}_\tau \left(d^{\dagger}_\tau - c^{\dagger}_\tau e_\tau \right) \\
\delta e_{top} &= a^{\dagger -1}_{top} \left(d^{\dagger}_{top} - c^{\dagger}_{top} e_{top} \right) \\
\text{but } d^{\dagger}_{top} &= d^T_{w,top,d} w + d^T_{v,top,d} v && (3.45) \\
\text{so } \delta e_{top} &= a^{\dagger -1}_{top} \left(d^T_{w,top,d} w + d^T_{v,top,d} v - c^{\dagger}_{top} \left(d^T_{w,topsc,e} w + d^T_{v,topsc,e} v \right) \right) \\
d^T_{w,top,\delta e} &= a^{\dagger -1}_{top} d^T_{w,top,d} - a^{\dagger -1}_{top} c^{\dagger}_{top} d^T_{w,topsc,e} && (3.46) \\
d^T_{v,top,\delta e} &= a^{\dagger -1}_{top} d^T_{v,top,d} - a^{\dagger -1}_{top} c^{\dagger}_{top} d^T_{v,topsc,e} && (3.47)
\end{aligned}$$

This pattern continues down the tree in a similar manner.

$$\begin{aligned}
e_{\tau\alpha} &= \frac{1}{\sqrt{2}} \left(e_\tau + \delta e_\tau \right) \\
&= \frac{1}{\sqrt{2}} \left(d^T_{w,\tau,e} w + d^T_{v,\tau,e} v + d^T_{w,\tau,\delta e} w + d^T_{v,\tau,\delta e} v \right) \\
d^T_{w,\tau\alpha,e} &= \frac{1}{\sqrt{2}} \left(d^T_{w,\tau,e} + d^T_{w,\tau,\delta e} \right) && (3.48) \\
d^T_{w,\tau\beta,e} &= \frac{1}{\sqrt{2}} \left(d^T_{w,\tau,e} - d^T_{w,\tau,\delta e} \right) && (3.49) \\
d^T_{w,\tau,\delta e} &= a^{\dagger -1}_\tau d^T_{w,\tau,d} - a^{\dagger -1}_\tau c^{\dagger}_\tau d^T_{w,\tau,e} && (3.50)
\end{aligned}$$

A similar recursion exists for the coefficients of the measurement error.

At this point, we have a complete set of $\left\{ d^T_{w,\tau,\delta e}, d^T_{v,\tau,\delta e}, d^T_{w,\tau,e}, d^T_{v,\tau,e} \right\}$ for all nodes on the tree, where v and w at the zeroth level form a set of independent random variables with assumed statistics. This implies that every e and δe can be expressed in terms of the uncertainty variables at the zeroth level as

$$\begin{aligned} e_\tau &= \sum d^T_{w,\tau,e} w + d^T_{v,\tau,e} v \\ \delta e_\tau &= \sum d^T_{w,\tau,\delta e} w + d^T_{v,\tau,\delta e} v \end{aligned}$$

As an aside, if the w and v are correlated, the approach will still work, although one would need to construct a correlation matrix, and thus uncover the underlying independent process that drives the uncertainty. The moments of the errors in the state estimates can be computed using the formula calculated in the following sections for propagation of moment information, and all that remains is to find suitable moments for the assumed probability distribution function. We can reformulate this to represent all estimation errors explicitly in terms of w and v at the zeroth level.

3.6 Calculation of coefficients of w and v in the error variables

Coefficients in the error variables can be constructed in an iterative process from level to level. We need to go through a series of transformations to obtain expressions in terms of variables for which we have statistics. w and v are related to their tree variables using the modified Hat and the Haar transform respectively.

$$\begin{aligned}
d^{\dagger}_{\tau} &= \sum_k \frac{\partial d^{\dagger}_{\tau}}{\partial v_k} v_k + \sum_k \frac{\partial d^{\dagger}_{\tau}}{\partial w_k} w_k && (3.51)\\
\frac{\partial d^{\dagger}_{\tau}}{\partial v_k} &= \sum_{\sigma \in T} \frac{\partial d^{\dagger}_{\tau}}{\partial v_\sigma} \frac{\partial v_\sigma}{\partial v_k} && (3.52)\\
\frac{\partial d^{\dagger}_{\tau}}{\partial v_\sigma} &= \frac{\partial}{\partial v_\sigma} \left(d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} - d'_{\tau\beta}) \right) && (3.53)\\
& \quad \text{note that down at least one of the branches, the} \\
& \quad \text{derivative will always be zero.} \\
\frac{\partial d_{\tau}}{\partial v_\sigma} &= \sum_{\theta \epsilon M} C^T R_{\tau}^{\theta-1} \text{ for } \tau = \sigma && (3.54)\\
&= 0 \text{ otherwise} && (3.55)\\
\frac{\partial d^*_{\tau}}{\partial v_\sigma} &= \frac{1}{\sqrt{2}} \frac{\partial}{\partial v_\sigma} \left(d^*_{\tau\alpha_k} + d'_{\tau\alpha_k} \right) && (3.56)\\
\frac{\partial d'_{\tau}}{\partial v_\sigma} &= -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta})) a^{\dagger -1}_{\tau} \frac{\partial d^{\dagger}_{\tau}}{\partial v_\sigma} && (3.57)
\end{aligned}$$

Thus, starting from the bottom of the tree, there is a recursive, parallelisable formula for the coefficients of v_τ as functions of the all of the estimation error wavelet coefficients. A similar recursion exists for w_τ.

$$\begin{aligned}
\frac{\partial d^{\dagger}_{\tau}}{\partial w_k} &= \sum_{\sigma \in T} \frac{\partial d^{\dagger}_{\tau}}{\partial w_\sigma} \frac{\partial w_\sigma}{\partial w_k} && (3.58)\\
\frac{\partial d^{\dagger}_{\tau}}{\partial w_\sigma} &= \frac{\partial}{\partial w_\sigma} \left(d_\tau - \frac{1}{\sqrt{2}}(d^*_{\tau\alpha} - d^*_{\tau\beta}) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} - d'_{\tau\beta}) \right) && (3.59)\\
\frac{\partial d_{\tau}}{\partial w_\sigma} &= (1 + A_{\tau\alpha})(\sqrt{2} B_{\tau\alpha})^{-T} Q_{\tau}^{-1} + \frac{1}{\sqrt{2}} \frac{\partial}{\partial w_\sigma}(w^*_{\tau\alpha} - w^*_{\tau\beta}) \text{ for } \tau\alpha = \sigma && (3.60)\\
&= \frac{1}{\sqrt{2}} \frac{\partial}{\partial w_\sigma}(w^*_{\tau\alpha} - w^*_{\tau\beta}) \text{ otherwise} && (3.61)\\
\frac{\partial d^*_{\tau}}{\partial w_\sigma} &= \frac{1}{\sqrt{2}} \frac{\partial}{\partial w_\sigma} \left(d^*_{\tau\alpha_k} + d'_{\tau\alpha_k} \right) && (3.62)\\
\frac{\partial d'_{\tau}}{\partial w_\sigma} &= -(a'_\tau b_\tau + \frac{1}{2}(c'_{\tau\alpha} - c'_{\tau\beta})) a^{\dagger -1}_{\tau} \frac{\partial d^{\dagger}_{\tau}}{\partial w_\sigma} && (3.63)
\end{aligned}$$

Finally, for the top node,

$$\begin{aligned}
d_{top} &= \sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} v_{top}^\theta - (1-A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} w_{\tau\alpha} - \frac{1}{\sqrt{2}}(w_{\tau\alpha}^* + w_{\tau\beta}^*) && (3.64)\\
\frac{\partial d_\tau^\ddagger}{\partial v_\sigma} &= \frac{\partial}{\partial v_\sigma}\left(d_{top} - \frac{1}{\sqrt{2}}(d_{\tau\alpha}^* + d_{\tau\beta}^*) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} + d'_{\tau\beta})\right) && (3.65)\\
\frac{\partial d_{top}}{\partial v_\sigma} &= \sum_{\theta \epsilon M} C^T R_\tau^{\theta-1} \quad \sigma = \text{top scaling function only} && (3.66)\\
&= 0 \text{ otherwise} \\
\frac{\partial d_\tau^\ddagger}{\partial w_\sigma} &= \frac{\partial}{\partial w_\sigma}\left(d_{top} - \frac{1}{\sqrt{2}}(d_{\tau\alpha}^* + d_{\tau\beta}^*) - \frac{1}{\sqrt{2}}(d'_{\tau\alpha} + d'_{\tau\beta})\right) && (3.67)\\
\frac{\partial d_{top}}{\partial w_\sigma} &= (1-A_{\tau\alpha})(\sqrt{2}B_{\tau\alpha})^{-T} Q_\tau^{-1} + \frac{1}{\sqrt{2}}(w_{\tau\alpha}^* + w_{\tau\beta}^*) \quad \sigma = \text{top scaling only} && (3.68)\\
&= \frac{1}{\sqrt{2}}(w_{\tau\alpha}^* + w_{\tau\beta}^*) \text{ otherwise} && (3.69)
\end{aligned}$$

At this point it is clear that all d, and thus all $d_\tau^\ddagger$ can be written explicitly as a linear combination of $\{w_k\}$ and $\{v_k\}$ at the zeroth level.

3.7 Higher order statistical calculations

Then the moments of the top node error process are defined, [1]

$$\begin{aligned}
E\left[e_{top}\right] &= d_w^T E\left[w\right] + d_v^T E\left[v\right] && (3.70)\\
\sigma_e^2 &= E\left[e_{top} e_{top}^T\right] - E\left[e_{top}\right] E\left[e_{top}\right]^T && (3.71)\\
&= E\left[\left(d_w^T w + d_v^T v\right)\left(d_w^T w + d_v^T v\right)^T\right] - E\left[e_{top}\right] E\left[e_{top}\right]^T && (3.72)\\
\text{and } E\left[v w^T\right] &= E\left[v\right] E\left[w\right]^T \text{ since } w \text{ and } v \text{ are independent,} \\
\sigma_e^2 &= d_w^T E\left[w w^T\right] d_w + d_v^T E\left[v v^T\right] d_v + d_v^T E\left[v\right] E\left[w\right]^T d_w + d_w^T E\left[w\right] E\left[v\right]^T d_v \\
&\quad - d_w^T E\left[w\right] E\left[w\right]^T d_w - d_v^T E\left[v\right] E\left[v\right]^T d_v \\
&\quad - d_w^T E\left[w\right] E\left[v\right]^T d_v - d_v^T E\left[v\right] E\left[w\right]^T d_w && (3.73)\\
\sigma_e^2 &= d_w^T \sigma_w^2 d_w + d_v^T \sigma_v^2 d_v && (3.74)
\end{aligned}$$

and higher order moments follow in a similar manner. It follows that the moments of estimation errors to arbitrary order can be computed at each node. A similar scheme using the coefficients for the error in the top node wavelet coefficient can be constructed, and thus we have a full set of moments (or as high as we care to calculate) for the top nodes.

3.7.1 Propagation of third order statistics

$$\begin{aligned} \mu_3 &= E\left[(X-\mu)^3\right] && (3.75)\\ &= \mu_3' - 3\mu\sigma^2 - \mu^3 && (3.76) \end{aligned}$$

So for e

$$\begin{aligned} e &= d_w w + d_v v && (3.77)\\ \mu_{e3} &= E\left[(e-\mu_e)^3\right] && (3.78)\\ &= \mu_{e3}' - 3\mu_e\sigma_e^2 - \mu_e^3 && (3.79)\\ &= d_w^3\mu_{w3}' - 3d_w\mu_w d_w^2\sigma_w^2 - d_w^3\mu_w^3 + d_v^3\mu_{v3}' - 3d_v\mu_v d_v^2\sigma_v^2 - d_v^3\mu_v^3 \\ &\quad +3E\left[d_w^2w^2d_vv\right] + 3E\left[d_wwd_v^2v^2\right] - 3d_w\mu_w d_v^2\sigma_v^2 - 3d_v\mu_v d_w^2\sigma_w^2 \\ &\quad -3d_w^2\mu_w^2 d_v\mu_v - 3d_w\mu_w d_v^2\mu_v^2 && (3.80) \end{aligned}$$

but since u and v are independent,

$$\begin{aligned} &= d_w^3\mu_{w3} + d_v^3\mu_{v3} + 3d_w^2\mu_{w2}'d_v\mu_v + 3d_w\mu_w d_v^2\mu_{v2}' && (3.81)\\ &\quad -3d_w\mu_w d_v^2\sigma_v^2 - 3d_v\mu_v d_w^2\sigma_w^2 - 3d_w^2\mu_w^2 d_v\mu_v - 3d_w\mu_w d_v^2\mu_v^2 \\ \text{but } \sigma^2 &= \mu_2' - \mu^2 && (3.82)\\ \mu_{e3} &= d_w^3\mu_{w3} + d_v^3\mu_{v3} && (3.83) \end{aligned}$$

3.7.2 Propagation of fourth order statistics

$$\begin{aligned} \mu_4 &= E\left[(X-\mu)^4\right] && (3.84)\\ &= \mu_4' - 4\mu_3'\mu + 6\mu_2'\mu^2 - 4\mu^4 + \mu^4 && (3.85)\\ &= \mu_4' - 4\mu\mu_3' - 6\mu^2\sigma^2 - \mu^4 && (3.86) \end{aligned}$$

and similarly at the fourth level for e

$$\begin{aligned} e &= d_w w + d_v v && (3.87)\\ \mu_{e4} &= E\left[(e-\mu_e)^4\right] && (3.88)\\ &= \mu'_{e4} - 4\mu_e\mu'_{e3} + 6\mu'_{e2}\mu_e^2 - 3\mu_e^4 && (3.89)\\ &= d_w^4\mu_{w4} + d_v^4\mu_{v4} + 6d_w^2 d_v^2\sigma_w^2\sigma_v^2 && (3.90)\end{aligned}$$

The appendix contains more details of the above computations. At this point, we have an explicit form for the estimation errors, both scaling and wavelet coefficients, in terms of the probability distribution functions of the underlying uncertainties. This translates to knowing the moments of the uncertainty in the estimation error.

3.8 Approaches to explicit computation of estimation error

The preceding paragraphs show how to generate an explicit representation of the estimation error in terms of the uncertain variables in the original formulation. Since these uncertain variables are not known - indeed at best a probability distribution approximation will be known, we need to convert this information into an explicit error estimation. The three approaches listed here serve to demonstrate how one would produce meaningful numbers from the given probability distribution functions.

3.8.1 The polynomial chaos expansion coefficients approach

The principle behind the polynomial chaos expansion is to reduce computational complexity by the generation of an approximation to the original probability distribution, in terms of a well known, and computationally efficient basis function. Much of the following can be found in the theses of Tatang, Wang and Engel, [3].

The argument used is that an exact probability distribution is often excessive for the purposes of the computation, or the question that we want answered, and a good approximation will give use a reasonable answer. Since the moments to arbitrary high order of the normal distribution are well known, it is common to approximate random variables in terms of functions of normal distributions. Suppose we have a random variable, θ, with an approximation for its probability distribution, and the knowledge that it shares some features with the normal distribution, such as being unimodal, two-tailed, and relatively smooth. We can expand this random variables, either in terms of multiple normally distributed functions, or in terms of orthogonal functions of a single normally distributed random variable, which for the purposes of this discussion we shall call ξ. The set of Hermite polynomials gives a set of mutually orthogonal polynomials to order n. The following represents our variable in terms of the first four Hermite polynomials with four undetermined coefficients

$$\theta = \theta_0 + \theta_1 \xi + \theta_2 \left(\xi^2 - 1\right) + \theta_3 \left(\xi^3 - 3\xi\right) \tag{3.91}$$

The selection of these coefficients requires some goodness of fit test, so that the resulting probability distribution function looks like the underlying distribution. The choice made by Tatang et al, [3], was the matching of the first four moments, the justification for this is that the first four moments of a probability distribution function are well known statistical quantities. Clearly one could add more orthogonal polynomials to the expression and increase the accuracy, however the goal of the approach is to reduce the computational complexity of a numerical integral, and a large number of terms. A large number of terms suggests that a better selection of polynomials, or of underlying distributions may produce better results. We begin by computing the first four moments of the new random variable in terms of its parameters.

$$\begin{aligned} E\left[\xi^k\right] &= 0 \;\forall k \text{ odd} && (3.92) \\ E\left[\xi\right] &= 1 && (3.93) \\ E\left[\xi^k\right] &= (k-1)\, E\left[\xi^{k-2}\right] && (3.94) \end{aligned}$$

$$\begin{aligned}
\theta &= \theta_0 + \theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right) && (3.95)\\
E\left[(\theta-\theta_0)^2\right] &= \theta_1^2 + 2\theta_2^2 + 6\theta_3^2 && (3.96)\\
E\left[(\theta-\theta_0)^3\right] &= 6\theta_1^2\theta_2 + 108\theta_2\theta_3^2 + 8\theta_2^3 + 36\theta_1\theta_2\theta_3 && (3.97)\\
E\left[(\theta-\theta_0)^4\right] &= 3\theta_1^4 + 60\theta_2^4 + 3348\theta_3^4 + 576\theta_1\theta_2^2\theta_3 + 24\theta_1^3\theta_3\\
&\quad +60\theta_1^2\theta_2^2 + 252\theta_1^2\theta_3^2 + 1296\theta_1\theta_3^3 + 2232\theta_2^2\theta_3^2 && (3.98)
\end{aligned}$$

These coefficients allow us to match previously known moments of a probability distribution function. Note that the expression above is fourth order for the four parameter expansion, and it should not be surprising that the exact matching problem may have no solution, or one that is multi-valued, and thus dependent on the initial guess used in a numerical algorithm.

The approach suggested by Tatang is to solve the minimisation problem

$$\min \sum_i \left(m_{i,observed} - m_{i,calculated}\right)^2 \tag{3.99}$$

where m_i represents the ith moment of the distribution $f(\theta)$, defined

$$m_i = \int_{-\infty}^{\infty} \theta^i f(\theta)\, d\theta \tag{3.100}$$

Again, this may be non-convex, and thus a true minimum may be hard to find.

3.9 Computation of upper and lower error bars

We assume at this stage that the probability distribution is completely described by the four moments - i.e. that the higher order moments have a negligible effect on the shape of the function.

To generate the probability distribution function of the derived variable, we can make use of knowledge of the statistics of the underlying random variable, and propagate these through an integral equation.

$$\theta = \theta_0 + \theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right) \tag{3.101}$$

The function θ is cubic in ξ. Consider the limiting behaviour for $\theta_3 > 0$, without loss of generality since $\theta_3 < 0$ requires the signs to be reversed. $\theta_3 = 0$ can be treated as a special case with one of the infinities treated as a turning point.

The turning points of this polynomial satisfy the following relation, with notation defined in Figure 3-1.

$$\begin{aligned}
\frac{d\theta}{d\xi} &= \frac{d}{d\xi}\left(\theta_0 + \theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right)\right) && (3.102)\\
&= \theta_1 + 2\theta_2\xi + 3\theta_3\left(\xi^2 - 1\right) && (3.103)\\
&= 0 && (3.104)\\
\xi_{TP} &= \frac{-\theta_2 \pm \sqrt{\theta_2^2 - 3\theta_3\left(\theta_1 - 3\theta_3\right)}}{\theta_3} && (3.105)\\
\theta_U &= \theta\left(\xi_R\right) = \theta\left(\xi_{LP}\right) && (3.106)\\
\theta_L &= \theta\left(\xi_L\right) = \theta\left(\xi_{RP}\right) && (3.107)
\end{aligned}$$

The region of non-montonicity is defined by $\{\theta\epsilon\left[\theta_L, \theta_U\right], \xi\epsilon\left[\xi_L, \xi_R\right]\}$.

$$\begin{aligned}
0 &= \theta - \theta_L = \left(\xi - \xi_L\right)\left(\xi - \xi_{RP}\right)^2 && (3.108)\\
&= \left(\xi - \xi_L\right)\left(\theta_3\xi + \theta_2 + \sqrt{\theta_2^2 - 3\theta_3\left(\theta_1 - 3\theta_3\right)}\right)^2 && (3.109)\\
&= \left(\xi - \xi_L\right)\left(\begin{array}{c} \theta_3^2\xi^2 + 2\theta_2\theta_3\xi + \theta_2^2 \\ +2\left(\theta_3\xi + \theta_2\right)\sqrt{\theta_2^2 - 3\theta_3\left(\theta_1 - 3\theta_3\right)} + \theta_2^2 - 3\theta_3\left(\theta_1 - 3\theta_3\right)\end{array}\right) && (3.110)\\
&= \left(\xi - \xi_L\right)\left(\begin{array}{c} \theta_3\xi^2 + 2\theta_2\xi + 2\theta_2^2/\theta_3 \\ +2\left(\xi + \theta_2/\theta_3\right)\sqrt{\theta_2^2 - 3\theta_3\left(\theta_1 - 3\theta_3\right)} - 3\left(\theta_1 - 3\theta_3\right)\end{array}\right) && (3.111)
\end{aligned}$$

Equating the constant terms in the above expression gives

$$\begin{aligned}
\theta_0 - \theta_L &= -\xi_L \left(2\theta_2^2/\theta_3 + 2\left(\theta_2/\theta_3\right)\sqrt{\theta_2^2 - 3\theta_3\left(\theta_1 - 3\theta_3\right)} - 3\left(\theta_1 - 3\theta_3\right) \right) && (3.112)\\
\xi_L &= \left(\theta_U - \theta_0\right) / \left(2\theta_2^2/\theta_3 + 2\left(\theta_2/\theta_3\right)\sqrt{\theta_2^2 - 3\theta_3\left(\theta_1 - 3\theta_3\right)} - 3\left(\theta_1 - 3\theta_3\right) \right) && (3.113)\\
\xi_R &= \left(\theta_L - \theta_0\right) / \left(2\theta_2^2/\theta_3 - 2\left(\theta_2/\theta_3\right)\sqrt{\theta_2^2 - 3\theta_3\left(\theta_1 - 3\theta_3\right)} - 3\left(\theta_1 - 3\theta_3\right) \right) && (3.114)
\end{aligned}$$

From this we can produce a probability distribution function for θ.

$$\begin{aligned}
P(\theta\left(\xi\right) &< \theta\left(\xi_k\right)) = P\left(\xi < \xi_k\right) \text{ for } \theta < \theta_L, \theta > \theta_R && (3.115)\\
P(\theta\left(\xi\right) &< \theta) = P\left(\xi < \xi_{k1}\right) - P\left(\xi < \xi_{k2}\right) + P\left(\xi < \xi_{k3}\right) \text{for } \theta_L < \theta < \theta_R && (3.116)
\end{aligned}$$

where $k1, k2$ and $k3$ are the roots of $\theta\left(\xi\right) = \theta$.

This expression illustrates the problem with the approach. There is a bifurcation point at θ_L and θ_R, such that $f\left(\theta\right)$ has an infinite derivative at these bifurcation points. To elaborate further, suppose that the underlying probability distribution in ξ is peaked around ξ_{LP}, and negligible near ξ_R. The probability of $\theta > \theta_U$ will be negligible, but will increase sharply immediately after θ_U with an infinite derivative at the bifurcation point. When one is doing a numerical derivative, spurious peaks appear around these bifurcation points, due to approximation of the infinite integrals. Since the derivative is infinite, one cannot overcome this problem by reducing interval, indeed it will make the problem worse.

For the purposes of the approximation of the underlying probability distribution function, it would not matter if the significant values of the underlying probability distribution function were far from the turning points in the $\theta - \xi$ graph, however in many cases, the approximation works because we are using these points.

A way around this problem is to solve the problem requiring that the bifurcation points do not exist.

$$\xi_{TP} = \frac{-\theta_2 \pm \sqrt{\theta_2^2 - 3\theta_3\left(\theta_1 - 3\theta_3\right)}}{\theta_3} \qquad (3.117)$$

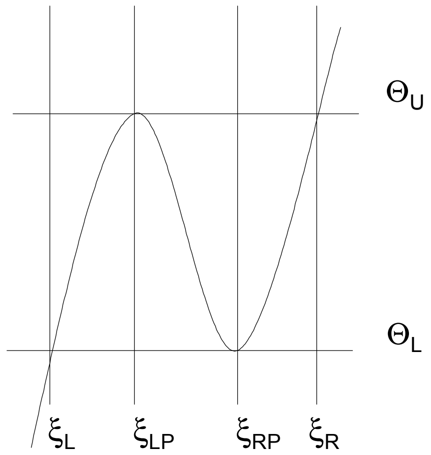

Figure 3-1: Relevant points on an arbitrary cubic function

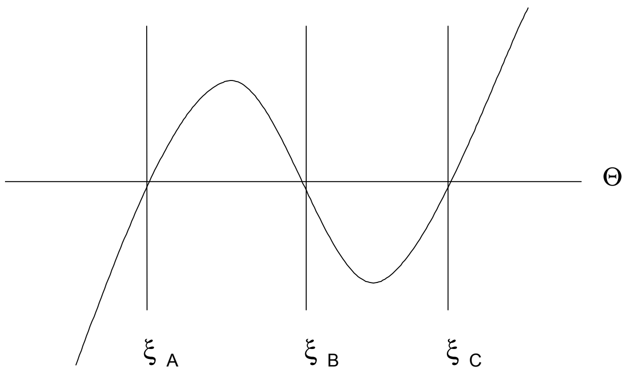

Figure 3-2: The roots of an arbitrary cubic function

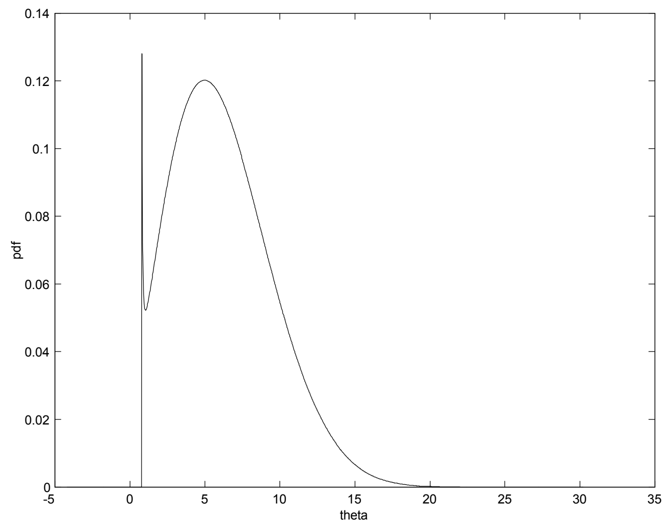

Figure 3-3: Rayleigh 5 regenerated from its moments featuring bifurcation problems

This is equivalent to requiring that no real turning points exist in the cubic function, alternately

$$\theta_2^2 - 3\theta_3 \left(\theta_1 - 3\theta_3\right) < 0 \tag{3.118}$$

This restriction provides us with a minimisation problem with a quadratic constraint, and a non-convex surface. While the problem may produce good solutions in certain classes of problems, experience with the moments obtained from the multiscale state estimation problem has shown that the solution to the optimisation problem is highly dependent on the initial guess, suggesting that the non-convexity of the cost surface causes significant problems for the solver.

The conclusion drawn is that the technique is useful if an approximation to the probability distribution function is sought, and we are not concerned about numerical aberrations around the bifurcation points, and in any case, the mass contained in these aberrations should be measured to determine that they do not contain a significant fraction of the mass of the probability distribution function. There will be probability distribution functions for which the technique of density function generation from moments is very effective, and the technique will be numerically efficient for these. This does not appear to be the case for the combinations of Rayleigh distribution tested.

An illustration of this problem is included in figure 3-3 which demonstrates a probability density function with the correct moments, but containing ”batman ears” due to numerical approximations at the bifurcation points. The illustration was demonstrated using the following set of coefficients.

$$\begin{aligned} \theta_0 &= 6.259 \\ \theta_1 &= 3.214 \\ \theta_2 &= 0.382 \\ \theta_3 &= -0.070 \end{aligned}$$

3.9.1 Direct convolution approach

A second approach that was considered was the brute force approach of directly evaluating the desired intervals[3], .

Suppose that we want to evaluate the probability of e which is a function of a set of random variables $\{w_k\}$. Suppose that the functional form of e is known to be

$$e = \sum_{k=1}^{n} d_k w_k \tag{3.119}$$

The probability density function of e can be computed as

$$p(e) = \int \cdots \int_{-\infty}^{\infty} \prod_{k=1}^{n-1} p\left(w_k\right) p\left(\frac{1}{d_n}\left(e - \sum_{k=1}^{n-1} d_k w_k\right)\right) dw_1 \ldots dw_n \tag{3.120}$$

This integral can be evaluated reasonably accurately using Gaussian quadrature, however we must use more than one point for each integral, and using three means that the number of multiplications required for the integral is 3^n. Since our state estimation calculations typically contain a support of upwards of 32 points, we rapidly reach computational intractability if we wish to compute error statistics for every point. For small numbers of points, such as parameter estimation problems with fewer than ten parameters, or cases where the integrands reduce to analytic expressions, such as the normal distribution, the direct convolution approach would be highly appropriate. For the general case of an unspecified distribution, the approach is inadequate for fast computation.

3.9.2 Monte Carlo simulation approach

Once all technically interesting avenues have been exhausted, we too resort to a fast, reliable approach to discovering the region of interest in our probability density function. The plots illustrated below are generated using the expressions of the form

$$e_\tau = \sum_{k=1}^{n} d_{w\tau k} w_k + d_{v\tau k} v_k \tag{3.121}$$

A probability distribution function is specified for the $\{w_k\}$ and $\{v_k\}$ at the zeroth levels and these are used to generate a typical value of e by using a uniform random number generator, specifically ran1 in Numerical Recipes, creating the appropriate $\{w_k\}$ and $\{v_k\}$. For n values of $\{w_k\}$ and $\{v_k\}$, the computational complexity is mn where m is the number of points computed during the simulation. Note that in all cases there is some numerical noise from the number of points used - this is due to the trade off between computational accuracy and simulation time.

The specific plots here are generated by identifying a useful range for binning from the first 100 points, then using 100 equally sized bins for the remainder of the simulation. Occasionally there will be no outliers in the first few points, in which case the tails will be more aggressively truncated. The tails are omitted from the plots since a relatively small number of points is generated, which makes the uncertainty comparable to the value of the function. The tails could be generated if desired, simply by increasing the number of bins from the original set.

For each plot, the vertical lines illustrate the equivalent of σ and 2σ for the normal distribution, although any specified confidence interval could be selected and plotted from the probability distribution functions. The points that correspond to

$$\begin{aligned}
p(\theta &< \theta_k) = 0.0228 \\
p(\theta &< \theta_k) = 0.1587 \\
p(\theta &< \theta_k) = 0.8413 \\
p(\theta &< \theta_k) = 0.9772 \qquad (3.122)
\end{aligned}$$

The σ lines would be used as representative error bars for a computation. The example used was a data set where the $\{w_k\}$ and $\{v_k\}$ are generated by a Rayleigh 5 distribution, and state estimation is done for the case study number three in Chapter 4. The Rayleigh five distribution was selected since it is non zero mean, and is positively skewed. A number of plots are illustrated here.

Plots 3-4 to 3-8 show the scaling functions of n equally weighted Rayleigh-5 error distributions moving up the tree. These are plotted from the physical level to the highest scaling function node on the tree. In all cases, the distributions are skewed towards positive infinity, consistent with the original Rayleigh distribution function. Note that as we move up the tree, they tend towards a normal distribution. This is due to relative infrequency of the points in the tail of the underlying distribution, and their correspondingly small contribution, in particular when we are summing ten distributions together. Note too that the spread increases as we move up the tree, however the relative spread decreases, since the orthonormality of the Haar transform causes larger numbers to appear at higher scales.

These plots should be compared to the normal distribution passing through the Haar transform, where a normal distribution at the zeroth level implies a normal distribution at any level of the tree, with the same standard deviation, σ, skewness, 0, and kurtosis, 3σ. The second group of plots is figure 3-4 and 3-9, the distribution of the first wavelet coefficient of two equally weighted Rayleigh-5 variables. Not that the skewness is completely gone as there is an equal contribution of tails from both sides - one term will have a positive tail and the other a negative tail. Compare this to the lowest order scaling function, where the positive skewness from the two distributions affects the sum with the same sign. Plots 3-10 to 3-15 show the estimation error distribution from a multiscale state estimation. Qualitatively, the plots are similar to those

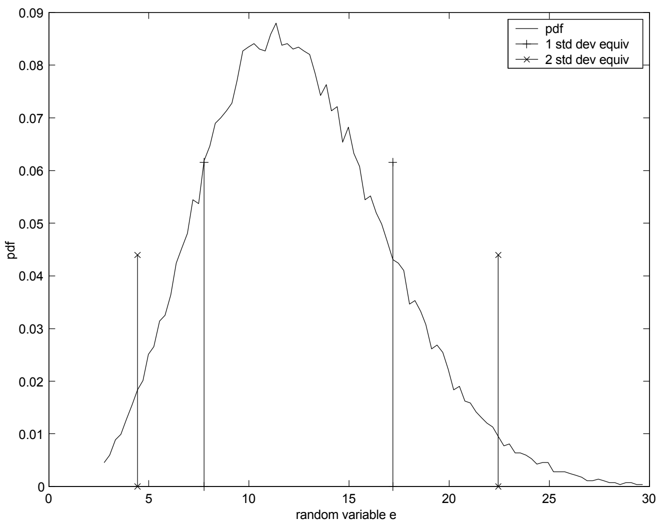

Figure 3-4: Error distribution in scaling function coefficients at the first level for equally weighted Rayleigh-5 distributions

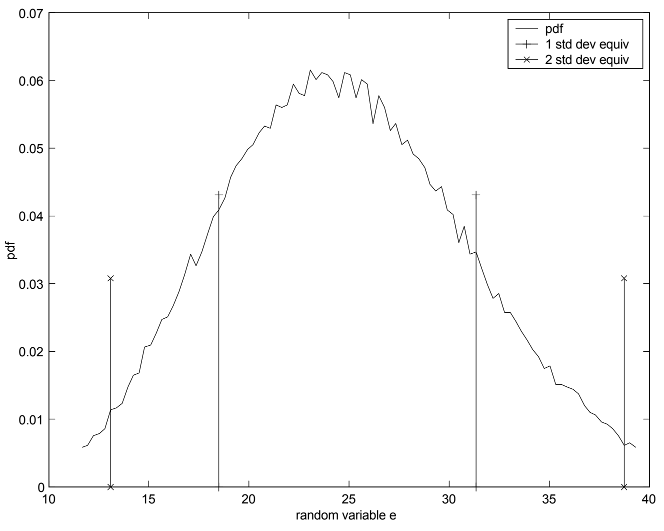

Figure 3-5: Error distribution in scaling function coefficient at the 2nd level for equally weighted Rayleigh-5 distributions

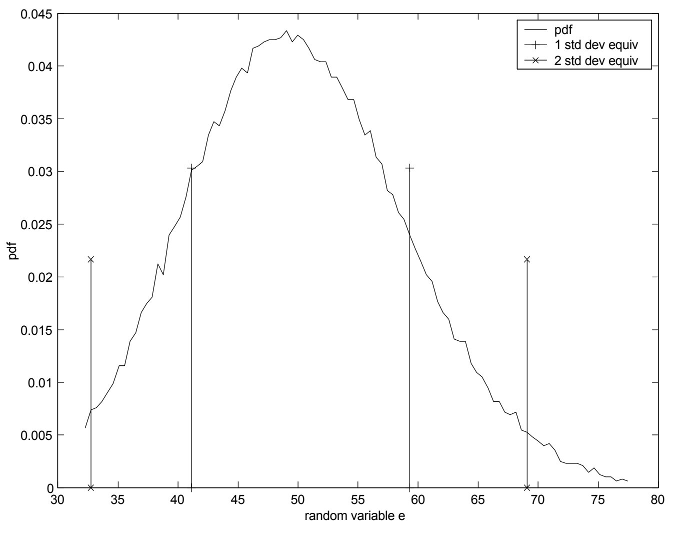

Figure 3-6: Error distribution in scaling function at the 3rd level for equally weighted Rayleigh-5 distributions

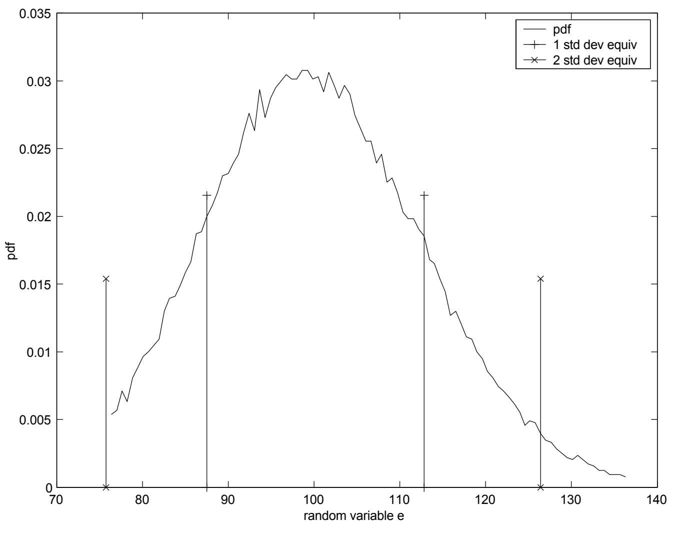

Figure 3-7: Error distribution in scaling function coefficient at the 4th level for equally weighted Rayleigh-5 distributions

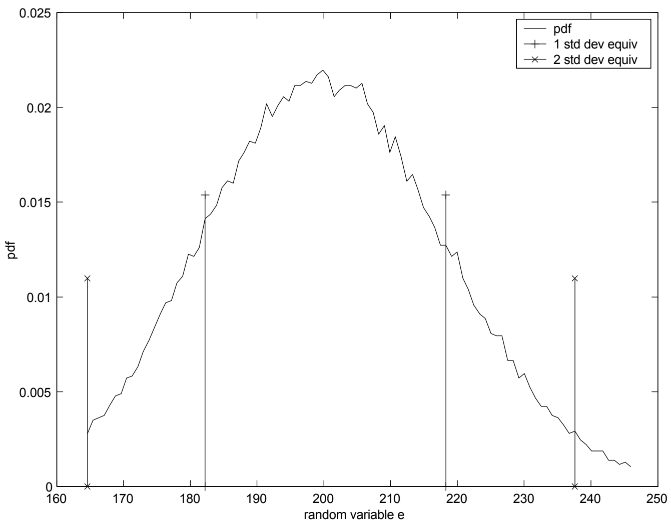

Figure 3-8: Error distribution in scaling function coefficient at the 5th level for equally weighted Rayleigh-5 distributions

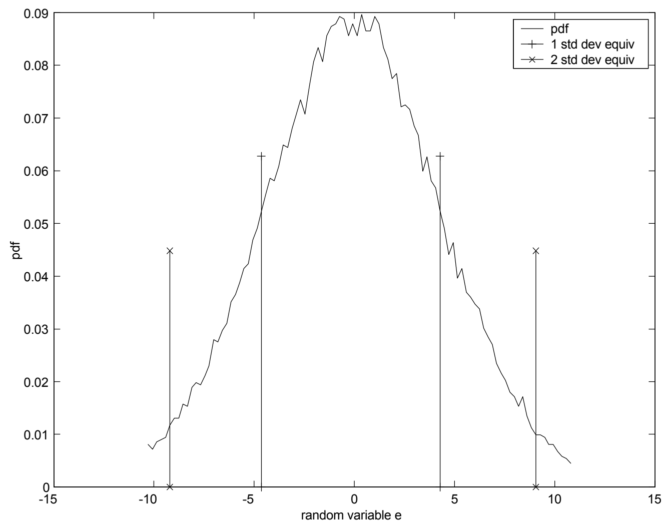

Figure 3-9: Error distribution in wavelet coefficient at the first level for equally weighted Rayleigh-5 distributions

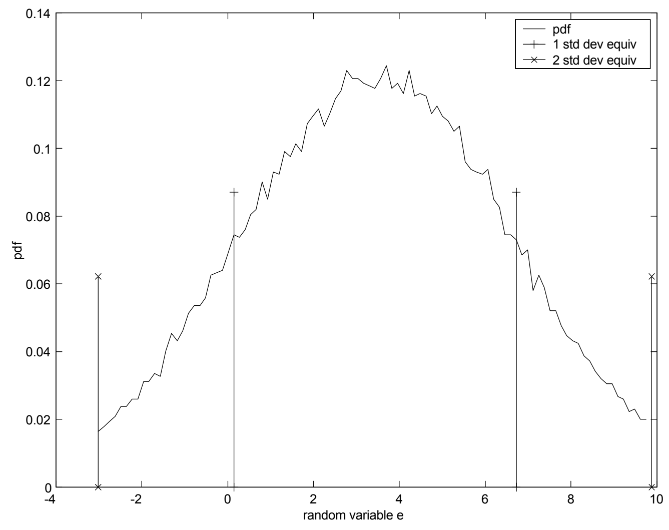

Figure 3-10: Error distribution in state estimate - topmost node

generated by a pure Rayleigh distribution - more significant skewing lower on the tree, and a tendency towards a normal distribution at the top of the tree. Quantitatively, there is more of a difference, and the absolute spread remains roughly constant at a σ of about 6 at all levels of the tree. This suggests that the uncertainty is spread evenly over all levels of the tree - or over all frequency bands. The final batch of plots, figure 3-15 to 3-20 represent a sample of plots across the zeroth level, specifically the left-most node of each right-most subtree of 2^n points. The zeroth level spans points 33 to 64, and thus 49 is just past the midpoint, 57 just past the 3/4 point. All plots look roughly the same, suggesting that there is no significant difference, despite the fact that the w variables are weighted very differently moving from left to right on the tree. This suggests that the multiscale state estimation algorithm provides a reasonable smoothing of uncertainty over all zeroth level points, which is the point of an estimator and smoother.

3.10 Conclusions

In this chapter we have derived an explicit form for the estimation error variables in the multi-scale state estimation algorithm. The error is explicitly represented in terms of the measurement

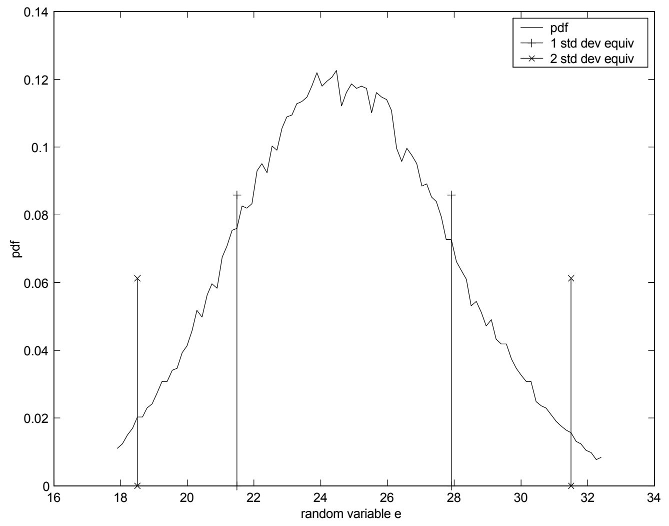

Figure 3-11: Error distribution in state estimate at leftmost node - level 4

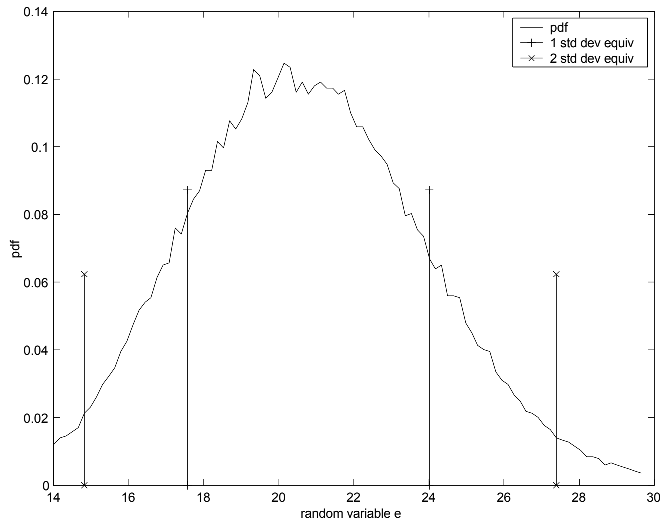

Figure 3-12: Error distribution in state estimate at leftmost node - level 3

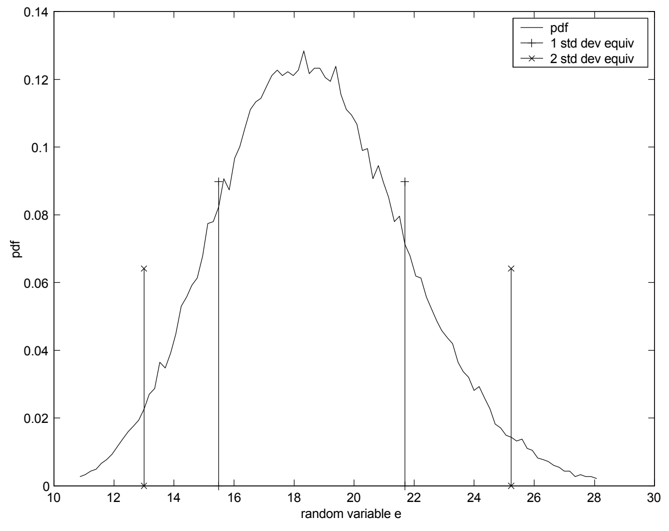

Figure 3-13: Error distribution in state estimate at leftmost node - level 2

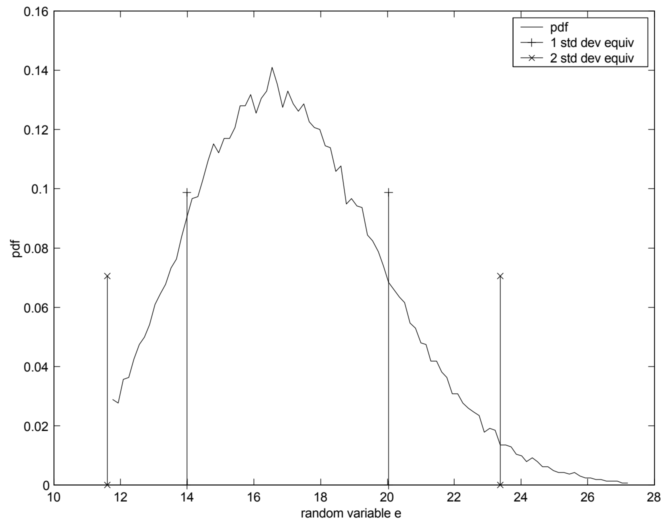

Figure 3-14: Error distribution in state estimate at leftmost node - level 1

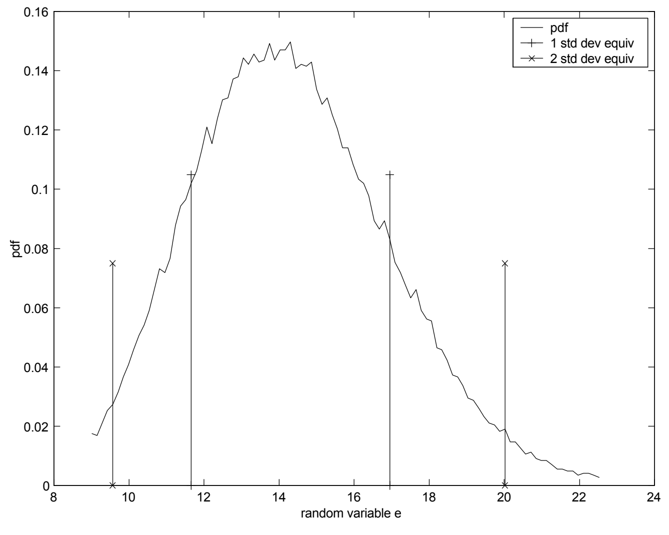

Figure 3-15: Error distribution in state estimate at leftmost node - level 0

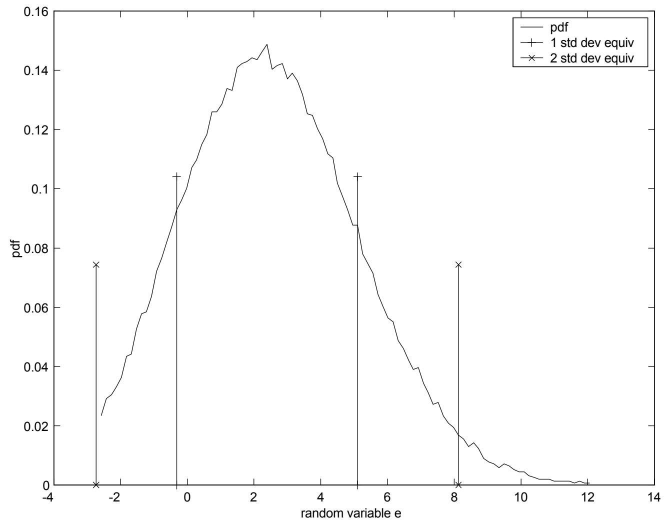

Figure 3-16: Error distribution in state estimate at node 49

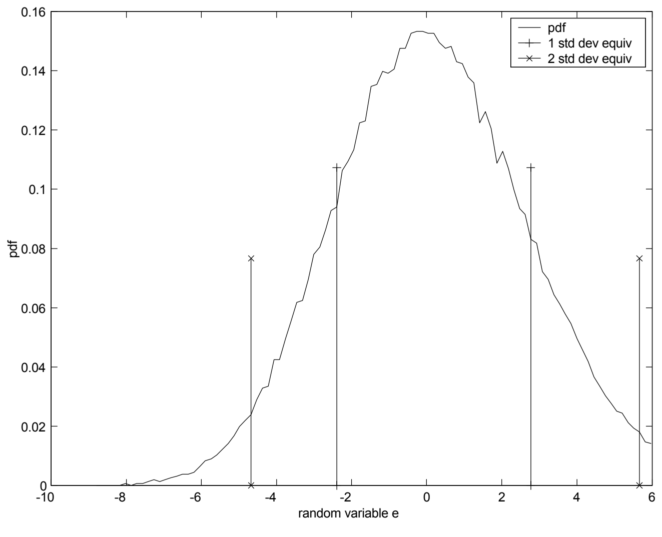

Figure 3-17: Error distribution in state estimate at node 57

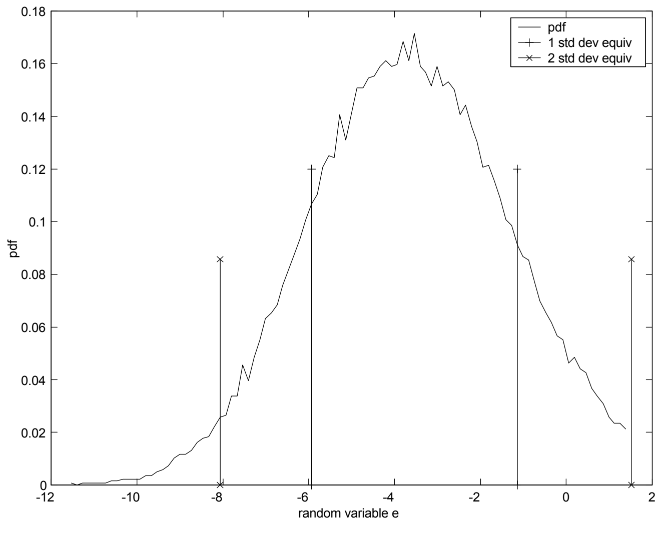

Figure 3-18: Error distribution in state estimate at node 61

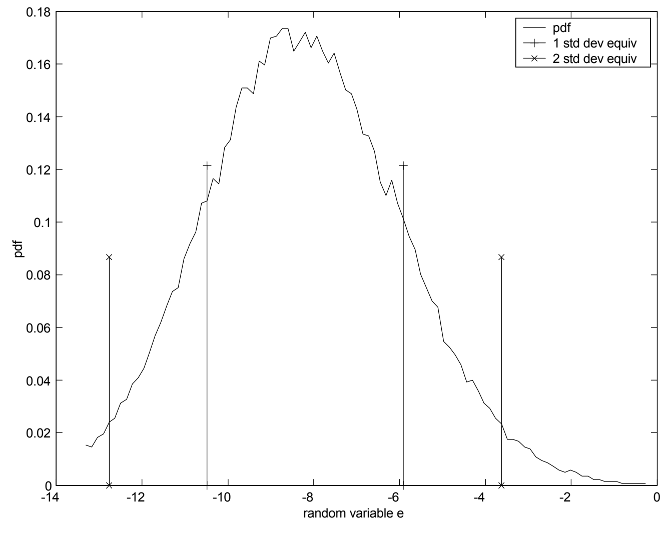

Figure 3-19: Error distribution in state estimate at node 63

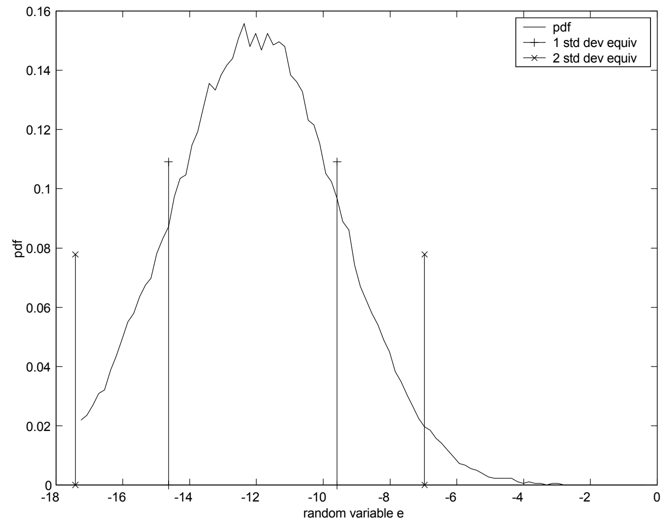

Figure 3-20: Error distribution in state estimate at node 64

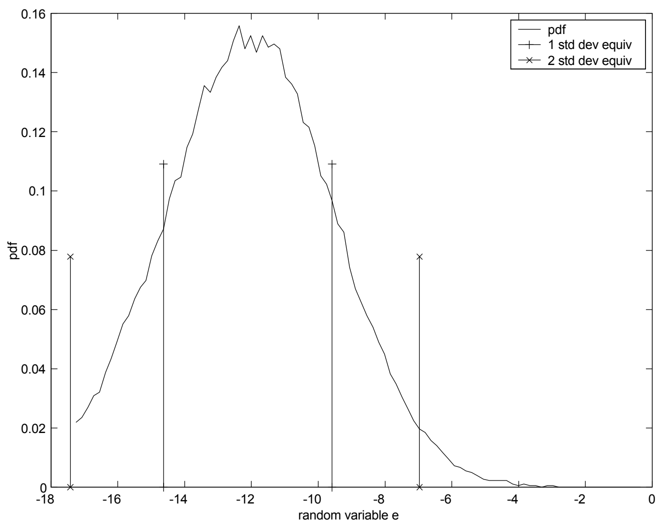

Figure 3-21: Error distribution in state estimate at node 64 - rightmost node

and modelling error variables, at every point on the tree. We have produced an algorithm to generate this estimate recursively and in a parallelisable manner on the multiscale tree. We have derived recursion formulae for the moments of these error variables. These moments can be used as they are as a measure of uncertainty, however this means that we are discarding information which may be useful. Thus we have shown how a probability distribution function can be generated.

We have examined three different approaches for the generation of probability distribution functions, and a measure of error that is comparable to the standard deviation for the normal distribution. The polynomial chaos expansion can be used to generate an approximation to the probability distribution function in a short period of time, but it is sensitive to initial guesses and causes spurious results for many underlying probability distribution functions. The complete integral approach using convolution of the underlying probability distribution functions will give complete and accurate probability density functions, but the amount of computation required for a meaningful answer makes it suitable for small problems only. Since the state estimates from the multiscale state estimation algorithm are dependent on a large number of underlying uncertainty variables, this approach is unsuitable in our case. Finally we have shown how a probability distribution function can always be generated using Monte Carlo simulation, and will converge reasonably quickly. It provides estimates to error bounds quickly, and should be used in the most general cases, where the underlying probability density functions are not susceptible to analytical integration, and the integrals cannot be accurately represented by a polynomial chaos expansion.

This suggests that the Monte Carlo approach is by far the safest, and should be used in any automatic implementation.

3.11 Appendix

3.11.1 Higher order statistical calculations in graphic detail

Then the moments of the top node error process are as follows

$$\begin{aligned}
E\left[e_{top}\right] &= d_w^T E\left[w\right] + d_v^T E\left[v\right] && (3.123)\\
\sigma_e^2 &= E\left[e_{top}e_{top}^T\right] - E\left[e_{top}\right]E\left[e_{top}\right]^T && (3.124)\\
&= E\left[\left(d_w^T w + d_v^T v\right)\left(d_w^T w + d_v^T v\right)^T\right] - E\left[e_{top}\right]E\left[e_{top}\right]^T && (3.125)\\
\text{and } E\left[vw^T\right] &= E\left[v\right]E\left[w\right]^T \text{ since } w \text{ and } v \text{ are independent,} \\
\sigma_e^2 &= d_w^T E\left[ww^T\right]d_w + d_v^T E\left[vv^T\right]d_v + d_v^T E\left[v\right]E\left[w\right]^T d_w + d_w^T E\left[w\right]E\left[v\right]^T d_v \\
&\quad - d_w^T E\left[w\right]E\left[w\right]^T d_w - d_v^T E\left[v\right]E\left[v\right]^T d_v \\
&\quad - d_w^T E\left[w\right]E\left[v\right]^T d_v - d_v^T E\left[v\right]E\left[w\right]^T d_w && (3.126)\\
\sigma_e^2 &= d_w^T \sigma_w^2 d_w + d_v^T \sigma_v^2 d_v && (3.127)
\end{aligned}$$

and higher order moments follow in a similar manner. It follows that the moments of estimation errors to arbitrary order can be computed at each node. A similar scheme using the coefficients for the error in the top node wavelet coefficient can be constructed, and thus we have a full set of moments (or as high as we care to calculate) for the top nodes.

Propagation of third order statistics

$$\begin{aligned}
\mu_3 &= E\left[(X-\mu)^3\right] \\
&= E\left[X^3 - 3X^2\mu + 3X\mu^2 - \mu^3\right] \\
&= \mu_3' - 3\mu_2'\mu + 3\mu^3 - \mu^3 \\
&= \mu_3' - 3\mu\sigma^2 - \mu^3
\end{aligned}$$

So for e

$$\begin{aligned}
e &= d_w w + d_v v \\
\mu_{e3} &= E\left[(e-\mu_e)^3\right] \\
&= \mu_{e3}' - 3\mu_e\sigma_e^2 - \mu_e^3 \\
&= E\left[d_w^3 w^3\right] + 3E\left[d_w^2 w^2 d_v v\right] + 3E\left[d_w w d_v^2 v^2\right] + E\left[d_v^3 v^3\right] \\
&\quad -3\left(d_w\mu_w + d_v\mu_v\right)\left(d_w^2\sigma_w^2 + d_v^2\sigma_v^2\right) \\
&\quad -d_w^3\mu_w^3 - 3d_w^2\mu_{w2}' d_v\mu_v - 3d_w\mu_w d_v^2\mu_{v2}' - d_v^3\mu_v^3 \\
&= d_w^3\mu_{w3}' - 3d_w\mu_w d_w^2\sigma_w^2 - d_w^3\mu_w^3 + d_v^3\mu_{v3}' - 3d_v\mu_v d_v^2\sigma_v^2 - d_v^3\mu_v^3 \\
&\quad +3E\left[d_w^2 w^2 d_v v\right] + 3E\left[d_w w d_v^2 v^2\right] - 3d_w\mu_w d_v^2\sigma_v^2 - 3d_v\mu_v d_w^2\sigma_w^2 \\
&\quad -3d_w^2\mu_w^2 d_v\mu_v - 3d_w\mu_w d_v^2\mu_v^2 \\
&\quad \text{but since } u \text{ and } v \text{ are independent,} \\
&= d_w^3\mu_{w3} + d_v^3\mu_{v3} + 3d_w^2\mu_{w2}' d_v\mu_v + 3d_w\mu_w d_v^2\mu_{v2}' \\
&\quad -3d_w\mu_w d_v^2\sigma_v^2 - 3d_v\mu_v d_w^2\sigma_w^2 - 3d_w^2\mu_w^2 d_v\mu_v - 3d_w\mu_w d_v^2\mu_v^2 \\
\text{but } \sigma^2 &= \mu_2' - \mu^2 \\
\mu_{e3} &= d_w^3\mu_{w3} + d_v^3\mu_{v3}
\end{aligned}$$

Propagation of fourth order statistics

$$\begin{aligned}
\mu_4 &= E\left[(X-\mu)^4\right] \\
&= E\left[X^4 - 4X^3\mu + 6X^2\mu^2 - 4X\mu^3 + \mu^4\right] \\
&= \mu_4' - 4\mu_3'\mu + 6\mu_2'\mu^2 - 4\mu^4 + \mu^4 \\
&= \mu_4' - 4\mu\left(\mu_3 + 3\mu\sigma^2 + \mu^3\right) + 6\mu^2\left(\sigma^2 + \mu^2\right) - 4\mu^4 + \mu^4 \\
&= \mu_4' - 4\mu\mu_3' - 6\mu^2\sigma^2 - \mu^4
\end{aligned}$$

and for e

$$\begin{aligned}
e &= d_w w + d_v v \\
\mu_{e4} &= E\left[(e-\mu_e)^4\right] \\
&= \mu_{e4}' - 4\mu_e\mu_{e3}' + 6\mu_{e2}'\mu_e^2 - 3\mu_e^4 \\
&= E\left[d_w^4 w^4\right] + 4E\left[d_w^3 w^3 d_v v\right] + 6E\left[d_w^2 w^2 d_v^2 v^2\right] + 4E\left[d_w w d_v^3 v^3\right] + E\left[d_v^4 v^4\right] \\
&\quad -4\left(d_w\mu_w + d_v\mu_v\right)\left(d_w^3\mu_{w3} + d_v^3\mu_{v3}\right) - 6\left(d_w\mu_w + d_v\mu_v\right)^2\left(d_w^2\sigma_w^2 + d_v^2\sigma_v^2\right) \\
&\quad -d_w^4\mu_w^4 - 4d_w^3\mu_w^3 d_v\mu_v - 6d_w^2\mu_w^2 d_v^2\mu_v^2 - 4d_w\mu_w d_v^3\mu_v^3 - d_v^4\mu_v^4 \\
&= d_w^4\mu_{w4}' + 4d_w^3\mu_{w3}' d_v\mu_v + 6d_w^2\mu_{w2}' d_v^2\mu_{v2}' + 4d_w\mu_w d_v^3\mu_{v3}' + d_v^4\mu_{v4}' \\
&\quad -4d_w^4\mu_w\mu_{3w} - 4d_w\mu_w d_v^3\mu_{3v} - 4d_v\mu_v d_w^3\mu_{3w} - 4d_v^4\mu_v\mu_{3v} - 6d_w^4\mu_w^2\sigma_w^2 \\
&\quad -6d_w^2\mu_w^2 d_v^2\sigma_v^2 - 12d_w^3\mu_w d_v\mu_v\sigma_w^2 - 12d_w\mu_w d_v^3\mu_v\sigma_v^2 - 6d_v^2\mu_v^2 d_w^2\sigma_w^2 \\
&\quad -6d_v^4\mu_v^2\sigma_v^2 - d_w^4\mu_w^4 - 4d_w^3\mu_w^3 d_v\mu_v - 6d_w^2\mu_w^2 d_v^2\mu_v^2 - 4d_w\mu_w d_v^3\mu_v^3 - d_v^4\mu_v^4 \\
&= \left(4d_w^3\mu_{w3}' d_v\mu_v - 4d_v\mu_v d_w^3\mu_{3w} - 12d_w^3\mu_w d_v\mu_v\sigma_w^2 - 4d_w^3\mu_w^3 d_v\mu_v\right) \\
&\quad + \left(4d_w\mu_w d_v^3\mu_{v3}' - 4d_w\mu_w d_v^3\mu_{3v} - 12d_w\mu_w d_v^3\mu_v\sigma_v^2 - 4d_w\mu_w d_v^3\mu_v^3\right) \\
&\quad + \left(d_w^4\mu_{w4}' - 4d_w^4\mu_w\mu_{3w} - 6d_w^4\mu_w^2\sigma_w^2 - d_w^4\mu_w^4\right) \\
&\quad + \left(d_v^4\mu_{v4}' - 4d_v^4\mu_v\mu_{3v} - 6d_v^4\mu_v^2\sigma_v^2 - d_v^4\mu_v^4\right) \\
&\quad +6d_w^2\mu_{w2}' d_v^2\mu_{v2}' - 6d_w^2\mu_w^2 d_v^2\sigma_v^2 - 6d_v^2\mu_v^2 d_w^2\sigma_w^2 - 6d_w^2\mu_w^2 d_v^2\mu_v^2 \\
&= d_w^4\mu_{w4} + d_v^4\mu_{v4} + 6d_w^2 d_v^2\left(\mu_w^2 + \sigma_w^2\right)\left(\mu_v^2 + \sigma_v^2\right) - 6d_w^2\mu_w^2 d_v^2\sigma_v^2 - 6d_v^2\mu_v^2 d_w^2\sigma_w^2 - 6d_w^2\mu_w^2 d_v^2\mu_v^2 \\
&= d_w^4\mu_{w4} + d_v^4\mu_{v4} + 6d_w^2 d_v^2\sigma_w^2\sigma_v^2
\end{aligned}$$

3.11.2 Construction of polynomial chaos expansion coefficients

$$\theta = \theta_0 + \theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right) \tag{3.128}$$

$$\begin{aligned}
(\theta - \theta_0)^2 &= \left(\theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right)\right)^2 \\
&= 6\theta_2\theta_3\xi + 2\theta_1\xi^3\theta_2 + 2\theta_1\xi^4\theta_3 - 6\theta_1\xi^2\theta_3 + 2\theta_2\xi^5\theta_3 \\
&\quad -8\theta_2\xi^3\theta_3 - 2\theta_1\xi\theta_2 + \theta_1^2\xi^2 + \theta_2^2\xi^4 - 2\theta_2^2\xi^2 + \theta_2^2 \\
&\quad +\theta_3^2\xi^6 - 6\theta_3^2\xi^4 + 9\theta_3^2\xi^2
\end{aligned} \tag{3.129}$$

$$\begin{aligned}
(\theta - \theta_0)^3 &= \left(\theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right)\right)^3 \\
&= 3\theta_1^2\xi^4\theta_2 + 3\theta_1^2\xi^5\theta_3 - 9\theta_1^2\xi^3\theta_3 - 3\theta_1^2\xi^2\theta_2 + 3\theta_1\xi^5\theta_2^2 \\
&\quad -6\theta_1\xi^3\theta_2^2 + 3\theta_1\xi\theta_2^2 + 3\theta_1\xi^7\theta_3^2 - 18\theta_1\xi^5\theta_3^2 + 27\theta_1\xi^3\theta_3^2 \\
&\quad +21\theta_2^2\xi^3\theta_3 + 3\theta_2^2\xi^7\theta_3 - 15\theta_2^2\xi^5\theta_3 + 3\theta_2\xi^8\theta_3^2 - 21\theta_2\xi^6\theta_3^2 \\
&\quad +45\theta_2\xi^4\theta_3^2 - 9\theta_2^2\theta_3\xi + \theta_1^3\xi^3 + \theta_2^3\xi^6 - 3\theta_2^3\xi^4 + 3\theta_2^3\xi^2 + \theta_3^3\xi^9 \\
&\quad -9\theta_3^3\xi^7 + 27\theta_3^3\xi^5 - 27\theta_3^3\xi^3 - \theta_2^3 + 18\theta_1\xi^2\theta_2\theta_3 + 6\theta_1\xi^6\theta_2\theta_3 \\
&\quad -24\theta_1\xi^4\theta_2\theta_3 - 27\theta_2\theta_3^2\xi^2
\end{aligned} \tag{3.130}$$

$$\begin{aligned}
(\theta - \theta_0)^4 &= \left(\theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right)\right)^4 \\
&\quad \theta_1^4\xi^4 + \theta_2^4\xi^8 - 4\theta_2^4\xi^6 + 6\theta_2^4\xi^4 - 4\theta_2^4\xi^2 + \theta_3^4\xi^{12} - 12\theta_3^4\xi^{10} + 54\theta_3^4\xi^8 \\
&\quad -108\theta_3^4\xi^6 + 81\theta_3^4\xi^4 + \theta_2^4 + 84\theta_1\xi^4\theta_2^2\theta_3 + 12\theta_1\xi^8\theta_2^2\theta_3 \\
&\quad -60\theta_1\xi^6\theta_2^2\theta_3 + 12\theta_1\xi^9\theta_2\theta_3^2 - 84\theta_1\xi^7\theta_2\theta_3^2 + 180\theta_1\xi^5\theta_2\theta_3^2 \\
&\quad -36\theta_1\xi^2\theta_2^2\theta_3 + 36\theta_1^2\xi^3\theta_2\theta_3 + 4\theta_1^3\xi^5\theta_2 + 4\theta_1^3\xi^6\theta_3 - 12\theta_1^3\xi^4\theta_3 \\
&\quad -4\theta_1^3\xi^3\theta_2 + 6\theta_1^2\xi^6\theta_2^2 - 12\theta_1^2\xi^4\theta_2^2 + 6\theta_1^2\xi^2\theta_2^2 + 6\theta_1^2\xi^8\theta_3^2 - 36\theta_1^2\xi^6\theta_3^2 \\
&\quad +54\theta_1^2\xi^4\theta_3^2 + 4\theta_1\xi^7\theta_2^3 - 12\theta_1\xi^5\theta_2^3 + 12\theta_1\xi^3\theta_2^3 + 4\theta_1\xi^{10}\theta_3^3 \\
&\quad -36\theta_1\xi^8\theta_3^3 + 108\theta_1\xi^6\theta_3^3 - 108\theta_1\xi^4\theta_3^3 - 4\theta_1\xi\theta_2^3 + 48\theta_2^3\xi^5\theta_3 \\
&\quad +4\theta_2^3\xi^9\theta_3 - 24\theta_2^3\xi^7\theta_3 + 6\theta_2^2\xi^{10}\theta_3^2 - 48\theta_2^2\xi^8\theta_3^2 + 132\theta_2^2\xi^6\theta_3^2 \\
&\quad -40\theta_2^3\xi^3\theta_3 + 4\theta_2\xi^{11}\theta_3^3 - 40\theta_2\xi^9\theta_3^3 + 144\theta_2\xi^7\theta_3^3 - 216\theta_2\xi^5\theta_3^3 \\
&\quad +12\theta_1^2\xi^7\theta_2\theta_3 - 48\theta_1^2\xi^5\theta_2\theta_3 - 108\theta_1\xi^3\theta_2\theta_3^2 - 144\theta_2^2\xi^4\theta_3^2 + 12\theta_2^3\theta_3\xi \\
&\quad +54\theta_2^2\theta_3^2\xi^2 + 108\theta_2\theta_3^3\xi^3
\end{aligned} \tag{3.131}$$

Now eliminate odd powers of ξ :

$$\begin{aligned}
\theta &= \theta_0 + \theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right) && (3.132)\\
E\left[(\theta-\theta_0)^2\right] &= E\left[\left(\theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right)\right)^2\right]\\
&= E[2\theta_1\xi^4\theta_3 - 6\theta_1\xi^2\theta_3 + \theta_1^2\xi^2 + \theta_2^2\xi^4 - 2\theta_2^2\xi^2\\
&\quad +\theta_2^2 + \theta_3^2\xi^6 - 6\theta_3^2\xi^4 + 9\theta_3^2\xi^2] && (3.133)\\
E\left[(\theta-\theta_0)^3\right] &= E\left[\left(\theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right)\right)^3\right]\\
&= E[3\theta_1^2\xi^4\theta_2 - 3\theta_1^2\xi^2\theta_2 + 3\theta_2\xi^8\theta_3^2 - 21\theta_2\xi^6\theta_3^2\\
&\quad +45\theta_2\xi^4\theta_3^2 + \theta_2^3\xi^6 - 3\theta_2^3\xi^4 + 3\theta_2^3\xi^2\\
&\quad -\theta_2^3 + 18\theta_1\xi^2\theta_2\theta_3 + 6\theta_1\xi^6\theta_2\theta_3 - 24\theta_1\xi^4\theta_2\theta_3 - 27\theta_2\theta_3^2\xi^2] && (3.134)\\
E\left[(\theta-\theta_0)^4\right] &= E\left[\left(\theta_1\xi + \theta_2\left(\xi^2 - 1\right) + \theta_3\left(\xi^3 - 3\xi\right)\right)^4\right]\\
&= E[\theta_1^4\xi^4 + \theta_2^4\xi^8 - 4\theta_2^4\xi^6 + 6\theta_2^4\xi^4 - 4\theta_2^4\xi^2 + \theta_3^4\xi^{12} - 12\theta_3^4\xi^{10} + 54\theta_3^4\xi^8\\
&\quad -108\theta_3^4\xi^6 + 81\theta_3^4\xi^4 + \theta_2^4 + 84\theta_1\xi^4\theta_2^2\theta_3 + 12\theta_1\xi^8\theta_2^2\theta_3 - 60\theta_1\xi^6\theta_2^2\theta_3\\
&\quad -36\theta_1\xi^2\theta_2^2\theta_3 + 4\theta_1^3\xi^6\theta_3 - 12\theta_1^3\xi^4\theta_3 + 6\theta_1^2\xi^6\theta_2^2 - 12\theta_1^2\xi^4\theta_2^2\\
&\quad +6\theta_1^2\xi^2\theta_2^2 + 6\theta_1^2\xi^8\theta_3^2 - 36\theta_1^2\xi^6\theta_3^2 + 54\theta_1^2\xi^4\theta_3^2 + 4\theta_1\xi^{10}\theta_3^3\\
&\quad -36\theta_1\xi^8\theta_3^3 + 108\theta_1\xi^6\theta_3^3 - 108\theta_1\xi^4\theta_3^3 + 6\theta_2^2\xi^{10}\theta_3^2 - 48\theta_2^2\xi^8\theta_3^2\\
&\quad +132\theta_2^2\xi^6\theta_3^2 - 144\theta_2^2\xi^4\theta_3^2 + 54\theta_2^2\theta_3^2\xi^2] && (3.135)
\end{aligned}$$

Substituting

$$
\begin{aligned}
E\left[\xi^k\right] &= 0 \ \forall k \text{ odd} && (3.136)\\
E\left[\xi^k\right] &= (k-1)\, E\left[\xi^{k-2}\right] && (3.137)\\
E\left[\xi^0\right] &= 1 && (3.138)\\
E\left[\xi^2\right] &= 1 && (3.139)\\
E\left[\xi^4\right] &= 3 && (3.140)\\
E\left[\xi^6\right] &= 15 && (3.141)\\
E\left[\xi^8\right] &= 105 && (3.142)\\
E\left[\xi^{10}\right] &= 945 && (3.143)\\
E\left[\xi^{12}\right] &= 10395 && (3.144)\\
E\left[\xi^{14}\right] &= 135135 && (3.145)\\
E\left[\xi^{16}\right] &= 2027025 && (3.146)\\
E\left[\xi^{18}\right] &= 34459425 && (3.147)\\
E\left[\xi^{20}\right] &= 654729075 && (3.148)
\end{aligned}
$$

Gives:

$$
\begin{aligned}
\theta &= \theta_0 + \theta_1\xi + \theta_2\left(\xi^2-1\right) + \theta_3\left(\xi^3-3\xi\right) && (3.149)\\
E\left[(\theta-\theta_0)^2\right] &= 6\theta_1\theta_3 - 6\theta_1\theta_3 + \theta_1^2 + 3\theta_2^2 - 2\theta_2^2 + \theta_2^2 + 15\theta_3^2 - 18\theta_3^2 + 9\theta_3^2\\
&= \theta_1^2 + 2\theta_2^2 + 6\theta_3^2 && (3.150)\\
E\left[(\theta-\theta_0)^3\right] &= 9\theta_1^2\theta_2 - 3\theta_1^2\theta_2 + 105*3\theta_2\theta_3^2 - 15*21\theta_2\theta_3^2\\
&\quad +3*45\theta_2\theta_3^2 + 15*\theta_2^3 - 3*3\theta_2^3 + 3\theta_2^3 && (3.151)\\
&= -\theta_2^3 + 18\theta_1\theta_2\theta_3 + 15*6\theta_1\theta_2\theta_3 - 3*24\theta_1\theta_2\theta_3 - 27\theta_2\theta_3^2\\
&= 6\theta_1^2\theta_2 + 108\theta_2\theta_3^2 + 8\theta_2^3 + 36\theta_1\theta_2\theta_3 && (3.152)\\
E\left[(\theta-\theta_0)^4\right] &= E\left[\left(\theta_1\xi + \theta_2\left(\xi^2-1\right) + \theta_3\left(\xi^3-3\xi\right)\right)^4\right]\\
&= E[3\theta_1^4 + 105\theta_2^4 - 15*4\theta_2^4 + 18\theta_2^4 - 4\theta_2^4 + 10395\theta_3^4 - 945*12\theta_3^4 + 105*54\theta_3^4\\
&\quad -15*108\theta_3^4 + 3*81\theta_3^4 + \theta_2^4 + 3*84\theta_1\theta_2^2\theta_3 + 105*12\theta_1\theta_2^2\theta_3 - 15*60\theta_1\theta_2^2\theta_3\\
&\quad -36\theta_1\theta_2^2\theta_3 + 15*4\theta_1^3\theta_3 - 3*12\theta_1^3\theta_3 + 15*6\theta_1^2\theta_2^2 - 3*12\theta_1^2\theta_2^2\\
&\quad +6\theta_1^2\theta_2^2 + 105*6\theta_1^2\theta_3^2 - 15*36\theta_1^2\theta_3^2 + 3*54\theta_1^2\theta_3^2 + 945*4\theta_1\theta_3^3\\
&\quad -105*36\theta_1\theta_3^3 + 15*108\theta_1\theta_3^3 - 3*108\theta_1\theta_3^3 + 945*6\theta_2^2\theta_3^2 - 105*48\theta_2^2\theta_3^2\\
&\quad +15*132\theta_2^2\theta_3^2 - 3*144\theta_2^2\theta_3^2 + 54\theta_2^2\theta_3^2]\\
&= 3\theta_1^4 + 60\theta_2^4 + 3348\theta_3^4 + 576\theta_1\theta_2^2\theta_3 + 24\theta_1^3\theta_3\\
&\quad +60\theta_1^2\theta_2^2 + 252\theta_1^2\theta_3^2 + 1296\theta_1\theta_3^3 + 2232\theta_2^2\theta_3^2 && (3.153)
\end{aligned}
$$

These coefficients allow us to match previously known moments of a probability distribution function.

Bibliography

[1] Dougherty, E. *Probability and Statistics for the Engineering, Computing and Physcial Sciences*. Prentice-Hall, 1990.

[2] Huber, P. *Robust Statistical Procedures*. Philadelphia: SIAM, 1977.

[3] Tatang, Menner A. *Direct Incorporation of Uncertainty in Chemical and Environmental Engineering Systems*. PhD dissertation, Massachusetts Institute of Technology, 1995.

Chapter 4

Various Uses of the Multiscale State Estimator

4.1 Introduction

In this chapter, we demonstrate the main features of the multiscale state estimation algorithm. We show how the algorithm can be adapted to solve the control problem, how it works for various multiple measurement sets. The algorithm contains two weighting variables, R, and Q, and we discuss the effects of choosing these poorly, and how to identify when they have been chosen poorly. We discuss how to adapt the algorithm for higher order autoregressive processes. We demonstrate the types of solution produces by the constrained state estimator. We discuss how prior estimates can be used in moving horizon estimators to reduce computation and the avoid problems unreliable measurements. We make a comparison with the celebrated Kalman filter and the Rauch-Tung-Striebel smoother, as a classical example of a state estimator.

4.2 Alternative interpretations of the cost function

The cost function derived in Chapter 2 is designed for the state estimation problem, whereby a set of measurements is used to generate a set of underlying states that satisfies a certain dynamic system. A certain class of control problems has a cost function that is structurally identical, where instead of fitting to a set of measurements, we would like the outputs of the system to track a previously specified reference path.

Recall the cost function

$$\min_{\{\widehat{x}_{top},\delta\widehat{x}_\tau\}} \sum_{\theta \epsilon M}((y^\theta_{top} - C\hat{x}_{top})^T R^{\theta -1}_{top}(y^\theta_{top} - C\hat{x}_{top}) + \sum_{\tau\epsilon T}(\delta y^\theta_\tau - C\delta\hat{x}_\tau)^T R^{\theta-1}_\tau(\delta y^\theta_\tau - C\delta\hat{x}_\tau)) +$$
$$+\sum_{\tau\epsilon T}((1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})^T(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_\tau(\sqrt{2}B_{\tau\alpha})^{-1}\dots$$
$$\dots((1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha}) + \left(x_{0|-1} - \widehat{x}_0\right)^T P^{-1}_{0|-1}\left(x_{0|-1} - \widehat{x}_0\right) \tag{4.1}$$

Suppose we change the definition of y from being a set of measurements of the states to

$$y \equiv Cx \tag{4.2}$$

where y is some set of output variables of the plant described by the states, x. We would like these outputs to track some reference path R. There will be a steady state set of inputs, u, around which there will be a fluctuation, w, so that the input to be used for the plant is $(u+w)$. w can be defined from the dynamic system.

$$w_{\tau\alpha} \equiv (1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha} \tag{4.3}$$

This interpretation allows the multiscale state estimation algorithm, complete with prior estimate, to be used directly for the control problem, where both an initial condition and a model are known, and a reference path is provided for the plant output to track. The goal is to produce a set of inputs from the optimisation problem. The cost function has the following equivalent form.

$$\min_{\{\widehat{x}_{top},\delta\widehat{x}_\tau\}} (r_{top} - C\hat{x}_{top})^T R_{top}(r_{top} - C\hat{x}_{top}) + \sum_{\tau\epsilon T}(\delta r_\tau - C\delta\hat{x}_\tau)^T R_\tau(\delta r_\tau - C\delta\hat{x}_\tau) +$$
$$+\sum_{\tau\epsilon T}(w^T_{\tau\alpha}(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_\tau(\sqrt{2}B_{\tau\alpha})^{-1}w_{\tau\alpha} + \left(x_{0|-1} - \widehat{x}_0\right)^T P^{-1}_{0|-1}\left(x_{0|-1} - \widehat{x}_0\right) \tag{4.4}$$

The solution algorithm will be identical to the one derived in Chapter 2, and can be equipped with the same constraint handling mechanisms, and all other features of the original algorithm.

4.3 Optimal fusion of multiple measurement sets

Optimal fusion deals with the appropriate weighting of various sets of measurements to produce an optimal state estimate. Since every measurement technique is subject to its own uncertainty, a better state estimate can be produced by acknowledging the differences in confidence that we have in the various measurements. The multiscale state estimator makes it possible to specify the uncertainties in each measurement explicitly.

Within the cost function, each individual measurement is described by a subscript indicating its position in the tree, and a superscript, θ, denoting the set of measurements to which it belongs.

$$\begin{aligned}
&\min_{\{\widehat{x}_{top},\delta\widehat{x}_\tau\}} \sum_{\theta\epsilon M}((y^\theta_{top} - C\hat{x}_{top})^T R^{\theta-1}_{top}(y^\theta_{top} - C\hat{x}_{top}) + \sum_{\tau\epsilon T}(\delta y^\theta_\tau - C\delta\hat{x}_\tau)^T R^{\theta-1}_\tau(\delta y^\theta_\tau - C\delta\hat{x}_\tau)) + \\
&+\sum_{\tau\epsilon T}((1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha})^T(\sqrt{2}B_{\tau\alpha})^{-T}Q^{-1}_\tau(\sqrt{2}B_{\tau\alpha})^{-1}\dots \\
&\dots((1+A_{\tau\alpha})\delta\hat{x}_\tau - (1-A_{\tau\alpha})\hat{x}_\tau + \sqrt{2}B_{\tau\alpha}u_{\tau\alpha}) + \left(x_{0|-1} - \widehat{x}_0\right)^T P^{-1}_{0|-1}\left(x_{0|-1} - \widehat{x}_0\right) \qquad (4.5)
\end{aligned}$$

The physical motivation for this is that we may have sets of measurements taken using different techniques, with different uncertainties. These sets of measurements may be complete, in the case of continuously sampled variables, or incomplete, such as a more accurate set of measurements taken in a region in which we have a particular interest. This is reflected in the choice of R^θ_τ for each measurement. R^θ_τ is typically chosen to be the standard deviation of the set of measurements, based on the Bayesian least squares estimate of a normally distributed measurement error. In many cases, it will provide a reasonable estimate of the magnitude of the error, and thus weight it appropriately in the cost function, although care should be taken when the uncertainty in the measurements differs considerably from the normal distribution.

Missing measurements are easily dealt with since they have no expected value, and infinite uncertainty. Setting $R^\theta_\tau = \infty$ is an appropriate way to represent this uncertainty, and thus removes all terms containing this measurement from the cost function.

Measurements taken at a slower frequency than the fastest set can be considered to have incomplete measurements at the fastest sampling frequency.

The following two diagrams illustrate a rather extreme form of sensor fusion. The plot in

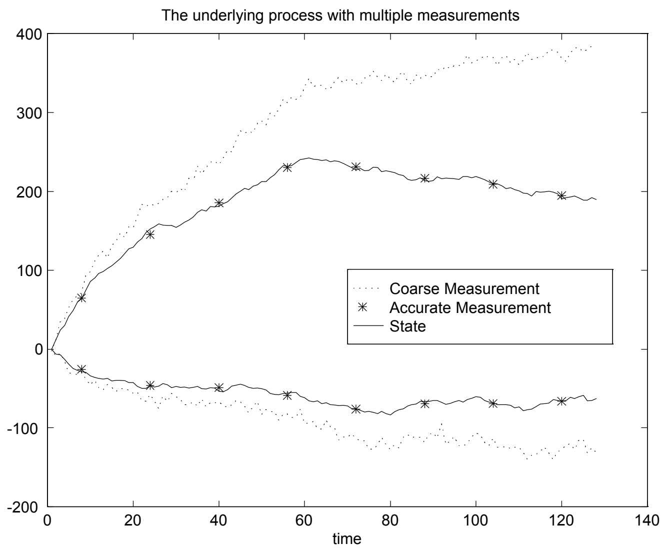

Figure 4-1: Generation of Multiple Measurement Sets

figure 4-1 shows the state, which was generated from a first order driven model using MATLAB. The state is measured using two measurement techniques - one a reliable sensor that is expensive to use, and thus used infrequently. This is represented by the * points. The second provides a coarser measurement, and captures rapid changes efficiently, but is subject to a drift. In this case study, a linear drift has been applied to the data.

The state estimator weights these measurements appropriately, with uncertainty estimates based on previous observations, and the result of the state estimation is illustrated in figure 4-2. The state estimate tracks the accurate measurements well, but contains jumps about halfway between each of the accurate measurements. The estimator is trying to fit to the local shape of the coarser measurements, while being forced to pass through the more accurate measurements, weighted by their smaller R_τ^θ. The blockiness is due to the nature of the Haar transform. The Haar basis functions have a span equal to the powers of two, thus, at the border between two basis functions, it is common to see an edge effect. This is pronounced in extreme cases, where there is an unmodelled disturbance such as the linear drift term.

The state estimator still manages to produce a reasonable approximation of the states, even though the coarse measurement set violates the assumption that the measurement error is zero mean white noise.

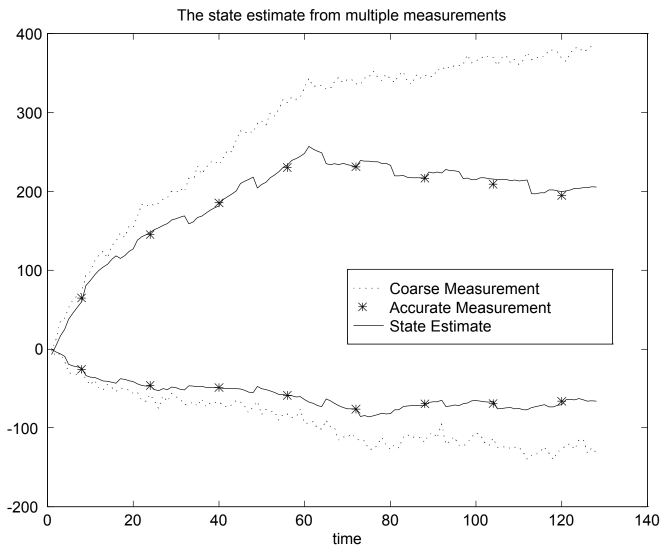

Figure 4-2: Estimation from Multiple Measurement Sets

4.4 The effect of different weighting functions

The weights R_τ^θ and Q_τ used in the cost functions are determined physically based on prior knowledge of the uncertainties associated with the measurement technique, and the uncertainty in the underlying model of the dynamic system. The following case study demonstrates the importance of this by using the same set of data to generate the state estimates, but with these different tuning parameters in the cost function.

The data in figures 4-3, 4-4 and 4-5 were generated using the following procedure.

The physical data was generated according to the first order model

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{4.6}$$

with $A = 0.95$, u_k, measured and known, and w_k a zero mean white noise process with covariance of $Q_{\text{real}} = 5$, which represents to any uncertainty in the inputs or the model. The true states that were generated by simulation are represented by plus signs in the plots. These are described in the legend as the physical results. Measurements of the process were generated by applying a zero mean white noise process with $R_{\text{real}} = 100$ to the simulated states. The measurements are represented by x's in the plots. These measurements were used to produce optimal state estimates using the multiscale state estimator. The state estimates are represented by the solid line in all of the plots. Figure 4-3 shows the state estimate produced using the true values of R and Q.

The two plots that follow illustrate the effect of poor choices of R and Q. The ratio of R and Q is the key issue here rather than the absolute values, since they will scale in the cost function. Plot 4-4 demonstrates the effect of a large R on the state estimator. Since R^{-1} is used to weight the measurements in the state estimator, the presence of a reasonably small $R = 5 \times 10^{-4}$ has the unsurprising effect of tracking the measurements very carefully with little regard to the underlying dynamics. An extreme value for R has been used to illustrate the point, clearly the estimate will vary continuously with R, for any function with continuous derivatives.

Plot 4-5 shows the effect of a large $R = 500$ on the state estimate. Note in comparison to figure 4-4 that a difference of only two orders of magnitude is sufficient to render the state estimate virtually useless - the measurements are essentially ignored, and the goal becomes to produce a small estimate of modelling uncertainty. The set of modelling uncertainties is defined in the multiscale sense and thus the left nodes on the tree dominate the state estimation algorithm, since we no longer have the weighting effect of the measurement at the right-most node. This effect is clearly seen in figure 4-5 with the large values of the state estimates at time points 64 and 128. The modified hat transform weights the left-most nodes on the tree more than the right nodes, and thus the more right nodes in the ancestry of a node, then that node will receive less weight the modelling uncertainty at the zeroth level.

The uncertainty estimates are essentially tuning parameters in the state estimation algorithm. The conclusion to be drawn from this case study is that reasonably accurate estimates of the uncertainties are required for meaningful state estimates, but also that incorrect estimates produce predictable behaviour, and thus can be identified from a comparison with the measurements.

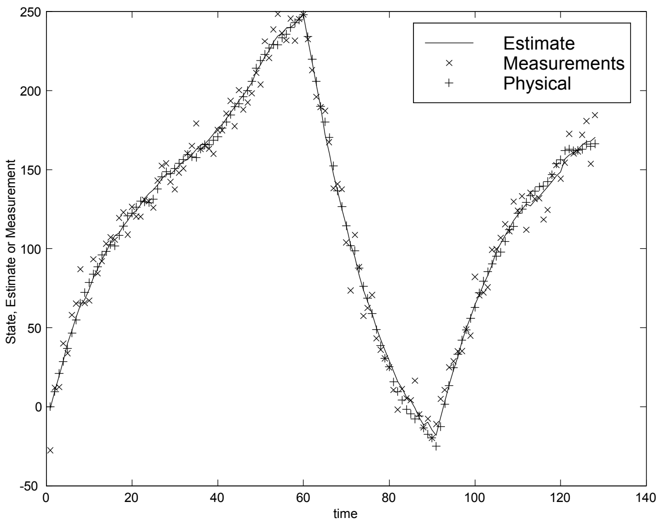

Figure 4-3: The effect of using matched R and Q on state estimation

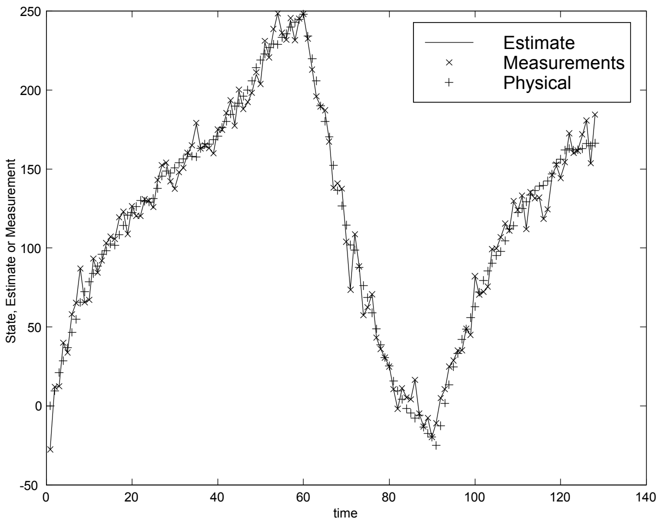

Figure 4-4: The effect of using a large R on state estimation

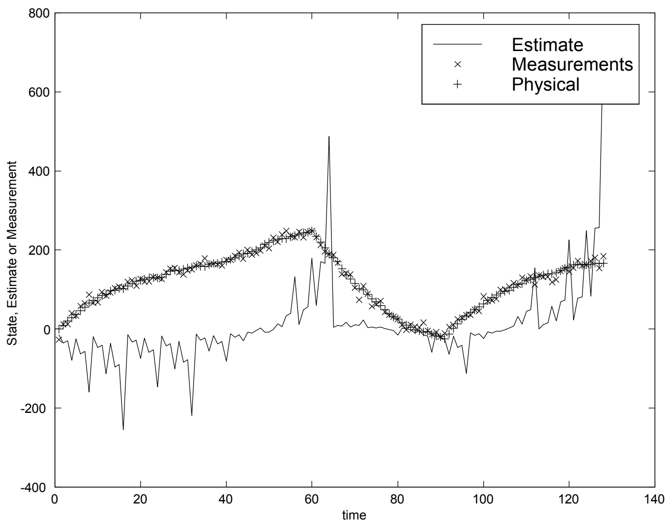

Figure 4-5: The effect of using a large Q on state estimation

4.5 Higher order polynomial systems

The multiscale state estimator deals equally well with higher order autoregressive processes. The case study presented in figure 4-6 shows the state estimate, measurements and underlying process for a fourth order polynomial system with a known input.

In this case study, data was generated using an initial condition of the origin, and the state dynamic equation

$$x_k = 0.95x_{k-1} - 0.4x_{k-2} - 0.1x_{k-3} - 0.05x_{k-4} + Bu_k + w_k \tag{4.7}$$

The dynamic system is converted by augmenting the state to produce the following augmented system.

$$\begin{bmatrix} x_k \\ x_{k-1} \\ x_{k-2} \\ x_{k-3} \end{bmatrix} = \begin{bmatrix} 0.95 & -0.4 & -0.1 & -0.05 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ x_{k-2} \\ x_{k-3} \\ x_{k-4} \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \\ 0 \end{bmatrix} u_k + \begin{bmatrix} w_k \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.8}$$

This fits the standard form for state estimation for the multiscale state estimator. for this

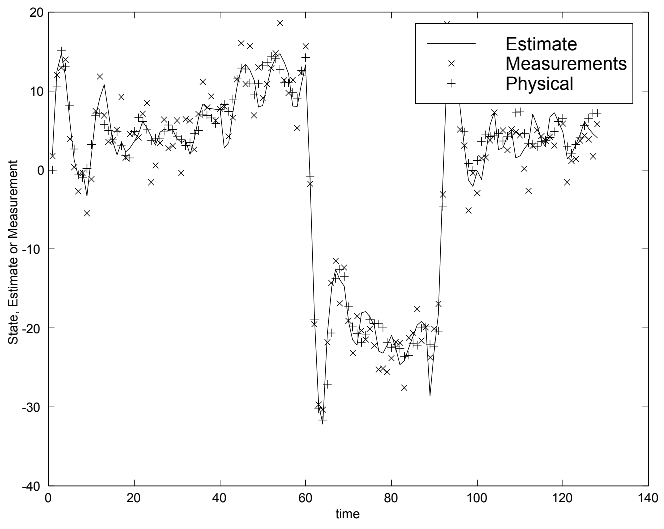

Figure 4-6: A higher order polynomial system using the multiscale state estimator

case study, $Q = 1$ and $R = 10$ were used to generate the data, and for the state estimation and a successful state estimation is illustrated in figure 4-6.

4.6 The constrained multiscale optimiser

Figure 4-7 illustrates the use of the constrained multiscale state estimator in a useful context.

The crosses represent a state trajectory specified by the controller or some other external consideration. The state is constrained by an upper bound of 100, and the optimisation problem takes the following form.

$$\min_{\{\widehat{x}_{top},\delta\widehat{x}_\tau\}}(r_{top}-C\hat{x}_{top})^T R_{top}(r_{top}-C\hat{x}_{top})+\sum_{\tau\epsilon T}(\delta r_\tau - C\delta\hat{x}_\tau)^T R_\tau(\delta r_\tau - C\delta\hat{x}_\tau)+$$
$$+\sum_{\tau\epsilon T}(u_{\tau\alpha}^T(\sqrt{2}B_{\tau\alpha})^{-T}Q_\tau^{-1}(\sqrt{2}B_{\tau\alpha})^{-1}u_{\tau\alpha}+\left(x_{0|-1}-\widehat{x}_0\right)^T P_{0|-1}^{-1}\left(x_{0|-1}-\widehat{x}_0\right) \tag{4.9}$$

where

$$\hat{x}_t \quad < \quad 100 \tag{4.10}$$
$$(1+A_{\tau\alpha})\delta\hat{x}_\tau-(1-A_{\tau\alpha})\hat{x}_\tau+\sqrt{2}B_{\tau\alpha}u_{\tau\alpha} \quad = \quad w_{\tau\alpha} \tag{4.11}$$

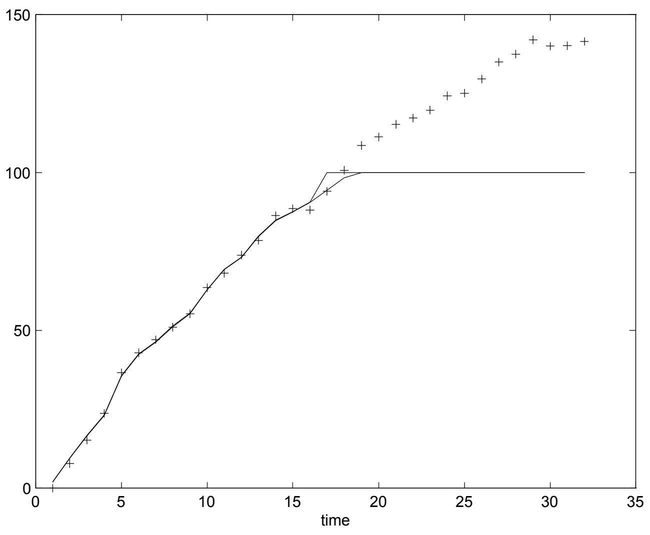

Figure 4-7: The constrained multiscale state estimator

Suppose we know the expected value of u, we can solve for a trajectory, x, and hence obtain a set of w - the set control variables. The notation presented in this case study is consistent with the multiscale state estimator, but not with the traditional control literature, where u and w have different interpretations. The solution presented in fig 4-7 contains the final two passes of the iteratively obtained solution. The multiscale state estimator initially generates the unconstrained minimum and moving down the tree until it reaches a constraint at the node that spans points 17-32. This intermediate solution is illustrated as pass one by one of the solid lines. A downsweep in lambda identifies that improvement can be made by relaxing the constraint at nodes 17 and 18. These are added to the basis, and a second solution is computed, which turns out to be optimal. Optimality can be shown since the KKT conditions are satisfied.

4.7 The use of prior estimates

Prior estimates are an extremely useful addition to the multiscale state estimator in many problem formulations.

Prior estimates allow the incorporation of additional information, including state estimates from earlier computations, [1], [2]. A major computational reduction can be achieved by using this in a moving horizon. This is particularly useful for systems where we wish to use a constant model of the estimation horizon, but modify it between estimations.

The left-most node of the tree is particularly sensitive to the values of u predicted by the state estimator, as illustrated in figure 4-5. A problem arises when the measurements are unreliable, and much of the energy in the cost function goes into the modelling uncertainty. The solution that minimises the cost function in the absence of a prior estimate will often choose a set of u's that produces a reasonable estimate everywhere except the left-most node. This node is directly affected by the largest set of input variables. The solution is considerably improved by increasing the weight of the first measurement, or by using a prior estimate from a previous estimation problem.

4.8 Comparison with the Kalman filter

A final demonstration compares the state estimate from the multiscale state estimator with the estimate provided by a Kalman filter and Rauch-Tung-Striebel smoother. This case study illustrated in figure 4-3 is used and the reader is referred to section 4.4 for details of the original case study. Plotting the Kalman filter state estimate is relatively uninformative since the lines essentially overlap. Deviation variables from the physical value are used to illustrate the difference between the methods. The measurement deviation predictably fluctuates randomly around a mean of zero since this was the distribution chosen to generate the measurement data. The Kalman filter and the multiscale state estimates are essentially indistinguishable from one another, as one would expect from the claim that minor changes in the shape of the cost function, and in the weighting parameters should not affect the quality of the estimate. When there is an outlier in the measurement error, both the Kalman filter and the multiscale state estimator will tend to underestimate the measurement error, meaning that the estimate regresses towards the mean for outliers. Neither algorithm directly rejects outliers, which can cause problems in the case of extreme failures in the measurement techniques. The problem can be corrected with more robust state estimators such as median based estimators, and Huber's estimator, although these are considerably slower than two norm based state estimators, since they typically rely on cost functions that are only piecewise continuous and not differentiable,

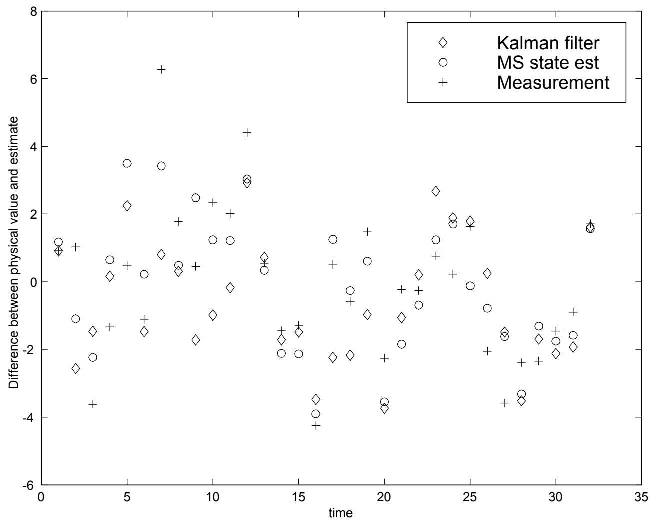

Figure 4-8: The difference between a state estimation using Kalman filter and the multiscale state esitmator

and thus cannot make use of gradient information reliably. They typically use some version of bisection.

4.9 Summary

The multiscale state estimation algorithm can be adapted to solve the control problem by noting the structural similarity between the state estimation and control optimisation problems. Multiple measurement sets can be incorporated by direct addition to the cost function, and missing measurements can be dealt with by using infinite uncertainty variables. R, and Q, the estimation algorithm tuning parameters behave in a predictably way, where if R is to dominant, the estimate will track the measurements too closely, while if Q is dominant, the estimate will largely ignore the measurements, and attempt to weight sections of the tree that have low weights. Higher order autoregressive processes can be estimated using state augmentation. Prior estimates are a useful addition since moving horizon estimators can be used to reduce computation, and to avoid the problems unreliable measurements. The multiscale state estimator produces a qualitatively similar solution to the Kalman filter and the Rauch-Tung-Striebel smoother, since it is essentially the same thing, with warps in its cost function.

Bibliography

[1] Kalman, R. and R. Bucy. “New Results in Linear Filtering and Prediction Theory,” *Journal of Basic Engineering:Trans of ASME*, 95–108 (1961).

[2] Rawlings, J.B. and K.R. Muske. “The Stability of Constrained Receeding Horizon Control,” *IEEE Transactions on Automatic Control*, *38*(10):1512–1516 (1993).

Chapter 5

The Paper Rolling Case Study

5.1 Introduction

The goal of this section is to demonstrate the differences between a multiscale state estimator and the classical Karhunen-Loeve transform as a means of state estimation. The specific class of problems under investigation is the class of continuous sheet forming processes, where a sheet with a spatial dimension is produced subject to some uniformity specification. Common industrial examples of this would be sheet metal production, aluminium foil, or can production, polymer film extrusion. A common uniformity requirement is to produce a sheet of constant thickness. In sheet dyeing processes such as those found in the textile industries, the uniform feature would typically be colour or shade variation, while in the paper rolling industry, which inspired the specific case study that will be discussed in this chapter, the uniformity specification is constant basis weight, caliper or moisture content. The assumption that we will make is that there is a measurement technique available that produces a correlation with the desired property - a light sensor could be used to measure colour or shade variations, while an array of height sensors would be used to measure thickness.

The industrial objective is to produce a uniform sheet in the presence of upstream production uncertainties, to identify when the sheet is within specification, and to suggest what to do when it is not. There is a strong economic motivation for this problem since improvements in production quality in these large scale industries will reduce raw material and energy costs significantly.

Measurement sets from these processes will typically contain a spatial component - usually from an array of sensors spaced across the sheet, or from a single sensor that sweeps from side to side across the sheet. This is referred to as the cross direction. Since production is continuous, there will be a temporal component to the data, corresponding directly to a second perpendicular spatial variable. This is referred to as the machine direction.

The goal of this case study is to identify a disturbance model from the full profile of data.

5.2 The Karhunen Loeve Transform

The Karhunen Loeve Transform, or principal component analysis was developed in 1963 by Karhunen and Loeve, [6], as a means of identifying a non-parametric model of random processes based on experimental data. It has been used successfully in a wide range of applications - including the currently blossoming field of bioinformatics.

The goal of the Karhunen-Loeve transform is to identify an optimal orthonormal basis, based on empirical data, with which to represent some random field. This orthonormal basis is frequently used to transform the data into a new set of coordinates, in which it is evident that many of the axes contain useless or inconclusive data, and can thus be ignored. This is particularly useful for identifying useful features, and numbers of degrees of freedom, in very large data sets[2].

Suppose we begin with a set of measurements $z_w(x,t)$, that represent the realization of some random vector field, where x refers to the spatial component and t to the temporal, in the general sense. The measurements may be a true samples of a continuous spatial and temporal domain, but may also be a set of time series of spatially unrelated data, such as temperature and pressure measurements in a plant, or may be various samples of spatial data, such as gene expression data from various parts of the body. In this case study, we are dealing with a truly continuous process in space and time, although it will be discretised in space by the fixed number of spatial sensors, and in time by the sampling frequency of the sensors.

We can define $\langle \bullet \rangle$ an ensemble average of any quantity over the observed realizations. For a discrete set this is

$$\langle f \rangle = \frac{1}{n} \sum_{k=1}^{n} f_n \tag{5.1}$$

and for continuous systems

$$\langle f \rangle \quad = \int f d \tag{5.2}$$

This can be updated constantly as new information becomes available.

With the ensemble average, one can define a deviation

$$\overline{z} = \overline{z}(x,t) = z_w(x,t) - \langle z_w(x,t) \rangle_w \tag{5.3}$$

from this we are able to generate the principal components which represent the most efficient basis for the representation of the data.

The following treatment follows that of Tatang [4] fairly closely. The general form of the transform can be written as

$$z(x,t) = \sum_{n=1}^{\infty} c_n \alpha_n(t) \beta_n(x) \tag{5.4}$$

where c_n is the coefficient of the nth term and $\alpha_n(t)$ is the nth temporal orthonormal function defined on the range of times $(0,T)$. $\beta_n(x)$ denotes the nth uncorrelated spatial random variable defined by

$$\int_0^T z(x,t) \alpha_n(t) dt = c_n \beta_n(x) \tag{5.5}$$

Since we can always subtract the expected value of $z(x,t)$, we can assume that it is a zero mean random process, with covariance denoted $C(t,s)$. Multiplying by $z(s,x)$ gives

$$\int_0^T z(s,x) z(x,t) \alpha_n(t) dt = z(s,x) c_n \beta_n(x) \tag{5.6}$$

And taking the expected value gives:

$$\int_0^T C(t,s)\alpha_n(t)dt = c_n^2\alpha_n(s) \tag{5.7}$$

This integral equation is an eigenvalue problem with respect to the covariance kernel, which can be solved using a singular value decomposition and thus $\alpha_n(t)$ and c_n^2 can be obtained, while $\beta_n(x)$ are obtained from equation 5.5.

Suppose now that we have an empirical realization of this random variable $E(x,t)$, again with realization in space and time. The Karhunen-Loeve expansion of this will be take the form

$$E(x,t) = \sum_{n=1}^{N} c_n\alpha_n(t)\beta_n(x) \tag{5.8}$$

Since there is a symmetry in the treatment of time and space, we can choose to generate a set of spatial eigenfunctions, and then generate temporal eigenfunctions from these, or the other way around. Thus covariance matrices can be constructed

$$\begin{aligned} C(t,s) &= \int E(x,t)E(x,s)dx && (5.9)\\ K(x,y) &= \int E(x,t)E(y,t)dt && (5.10) \end{aligned}$$

For large spatial dimensions, the first equation is preferable, for large temporal dimensions, the second. In the paper rolling problem with 20 sensors and a considerable set of temporal data, we elect to solve the second of these equations. $K(x,y)$ is computed empirically

$$K(x,y) = \frac{1}{N}\sum_{t=t_1}^{t_N} E(x,t)E(y,t) \tag{5.11}$$

The singular value decomposition is

$$\begin{aligned} K(x,y) &= UWV^T && (5.12)\\ W &= diag\{c_n^2\} && (5.13) \end{aligned}$$

where U is an orthogonal matrix containing the values of the eigenfunctions, or principal components, based on the first N points. The temporal eigenfunctions can be calculated from the spatial eigenfunctions as

$$\alpha_n(t) = \frac{1}{c_n} \int E(x,t)\beta_n(x)dx \tag{5.14}$$

And from the definition of these eigenvalues, they will be self-normalised. Further theoretical details are provided in a number of sources, specifically that error does goes to zero as we increase the number of terms.

5.3 The paper rolling case study

The practical example used here is taken from a 1996 case study by Rigopoulos and Arkun, [3]. Since their sponsor was in the pulp and paper industry we can use this to obtain a picture of the process. The plant produces sheets of paper at a constant rate by producing wet paper from a headbox, which then forms a sheet, which passes below an array of twenty sensors arranged across the sheet, perpendicular to the motion of the paper. These sensors send a signal to our control centre, which tries to identify when faults are occurring, and thus when product needs to be discarded, or recycled. The concern in this case study is the identification of the problems.

They deal with a sheet forming process, with a uniformity requirement, and an unknown uncertainty structure. Thus there is no a priori model for the uncertainty, only, possibly for the measurement process - since means and standard deviations of the measurement process could be obtained by moving sensors around, and repeating experiments. The maximum entropy representation for a known mean and standard deviation is the normal distribution, which is the assumption we will make for any measurement noise, although as shown in Chapter four, we could develop additional statistics with a more detailed description of the measurement uncertainty.

The measurement data are constructed using the disturbance profile composed of sinusoids, although of course any profile could be used. The variations presented are regarded as typical of inconsistencies that arise from pulp flowrate and chemistry coming out of the headbox. The model for the underlying paper structure is given in terms of sensor number, n, for a time

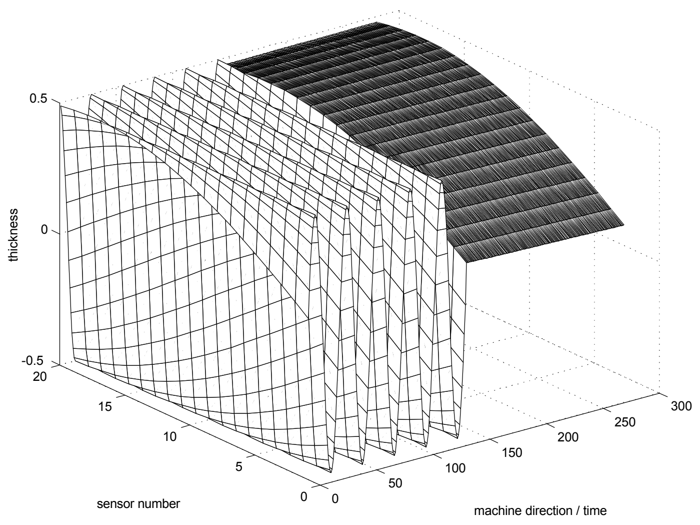

Figure 5-1: Actual pattern on the paper surface

horizon, T.

$$\theta_n = \frac{(n-1)\pi}{2(N-1)} \tag{5.15}$$

$$E_1(n,t) = 0.5\sin\left(\frac{20\pi k}{K} + \theta_n\right) \tag{5.16}$$

$$E_2(n,t) = 0.5\sin\left(\frac{(20 - 0.05 * \lfloor (k-499)/2 \rfloor)\pi}{K} + \theta_n\right) \tag{5.17}$$

This data is converted to measurement data by adding a Gaussian zero mean, white noise, with standard deviation of 0.1, and the data is then analysed.

5.4 The case study results

The case study is generated using the algorithm above. The original data, which is assumed to be the profile coming out of the headbox, and thus what we are trying to measure, is illustrated in Figure 5-1. The measurement data has the white noise added to it, and a reasonable amount to test the power of the methods. This data is illustrated in Figure 5-2.

Principal components are identified using the Karhunen-Loeve expansion. It turns out that a substantial fraction of the energy of the signals is captured in the first three principal

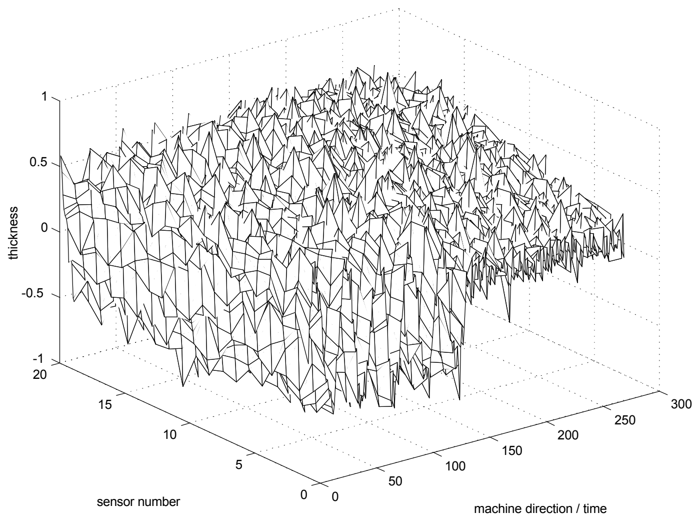

Figure 5-2: Raw measurements of paper rolling case study

components. The first three spatial eigenfunctions are illustrated in Figure 5-3. Note that the third spatial eigenvalue is beginning to look like a white noise process, and thus it seems that most of the important information has been captured in the first two principal components, or that the white noise is covering any further information.

The magnitude of the principal component coefficients are plotted as a function of sensor number in Figure 5-4. Not visible in this view is the alternation between components 1 and 2, which is illustrated more clearly in Figure 5-10.

This data can be viewed from a different point of view to illustrate the temporal variation in Figure 5-5. Note that the white noise gives rise to a sprinkling of non-zero components spread over the remaining principal components. The principal components of a white noise process turn out to be the Haar wavelet decomposition, and the energy is evenly spread over all components. thus the presence of two real components suggests more structure than white noise.

The data can be reconstructed using the first three principal components to give a reasonable resemblance to the underlying data. This is illustrated in Figure 5-6.

There is a clear difference in the nature of the underlying process in the first and second parts. Suppose principal components are generated using the second half alone. Figure 5-7

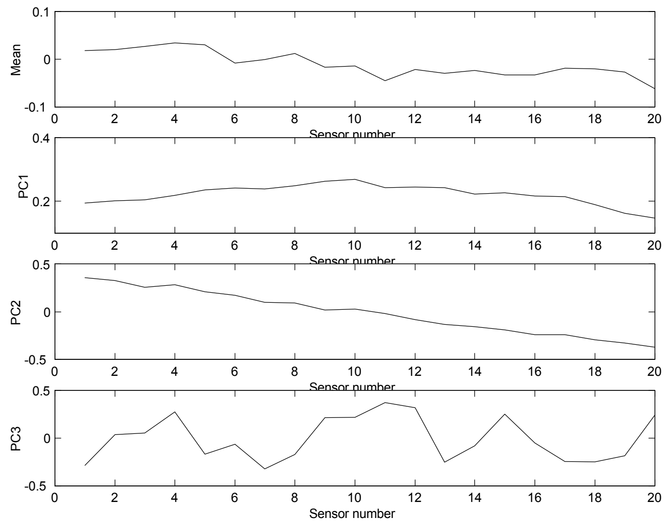

Figure 5-3: First three principal components for the paper rolling data

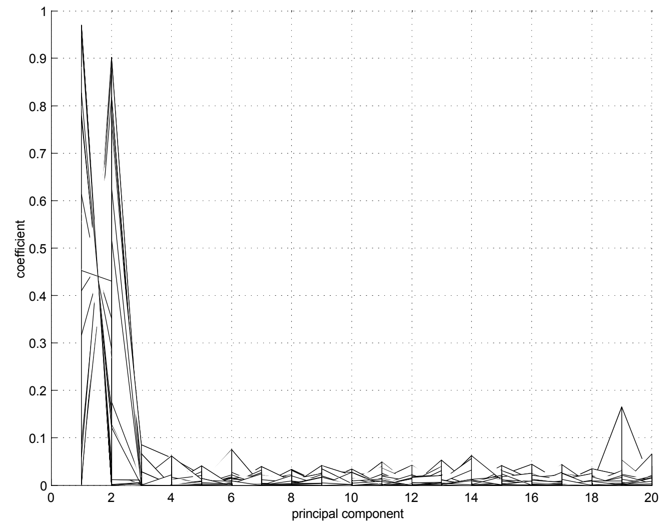

Figure 5-4: Coefficients of the principal components in decreasing order of importance.

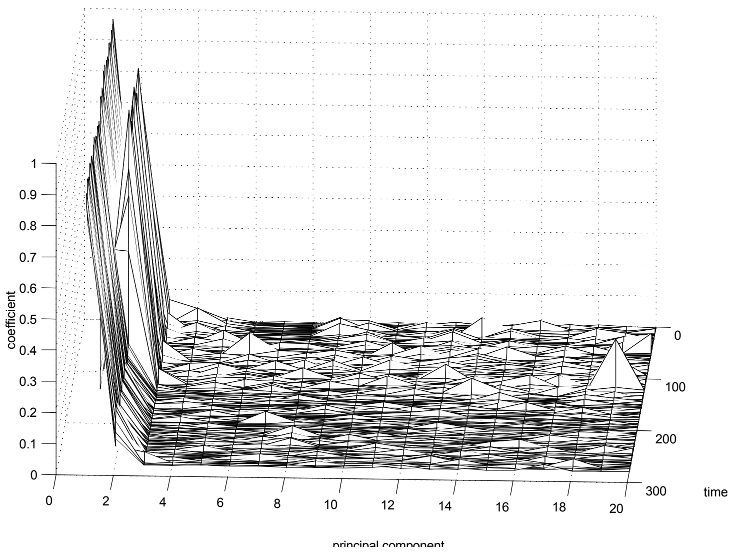

Figure 5-5: Magnitude of the principal component coefficients

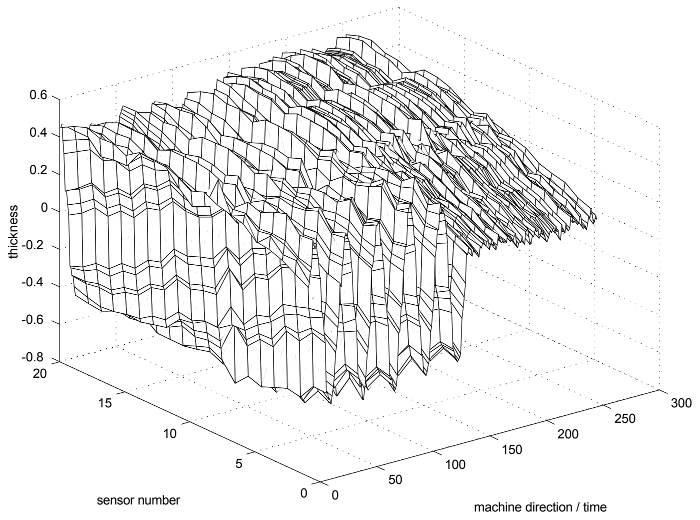

Figure 5-6: State estimation from measurements based on first three principal components only

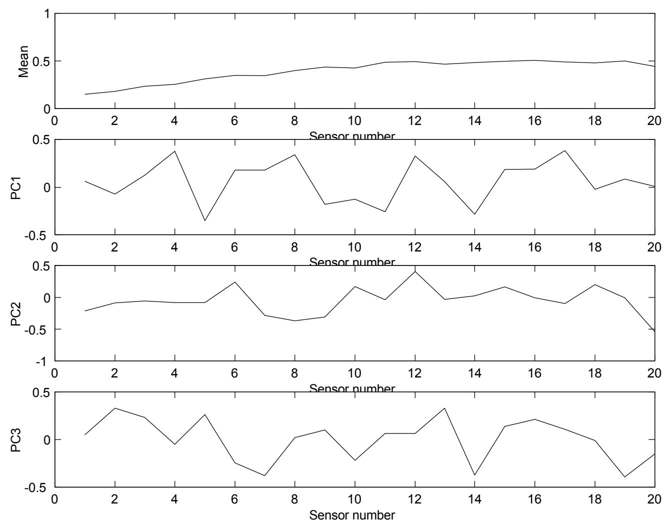

Figure 5-7: Principal components computed using the second half of the process

represents the coefficient structure based on principal components from the second half. Note that most of the structure is captured in the mean, and the principal components essentially capture and model the white noise.

The temporal coefficients are plotted in Figure 5-8. Note the significant difference between this plot and Figure 5-5, where we are decomposing using eigenfunctions from the first part of the data, where there is significant structure, and from the second part of the data, where there is considerably less. The first set of eigenfunctions describes the second set of data reasonably, but the converse is not true. The modelling of white noise is illustrated by the roughly even distribution of coefficients across the coefficient space for the second, or front, half of the data, while the back data, or first half, has a number of important components, but not concentrated at the low end of the principal component set. This illustrates the danger of using an inappropriate set of principal components to decompose the data, and suggests a measure for the detection of changes, where other coefficients of the principal component analysis become excited above a specified threshold. This would suggest that the most recent data needs to be used to generate a new set of principal components.

These principal components can be used to construct the data, again using the most significant components, which is illustrated in Figure 5-9. Unsurprisingly, the second half, for which

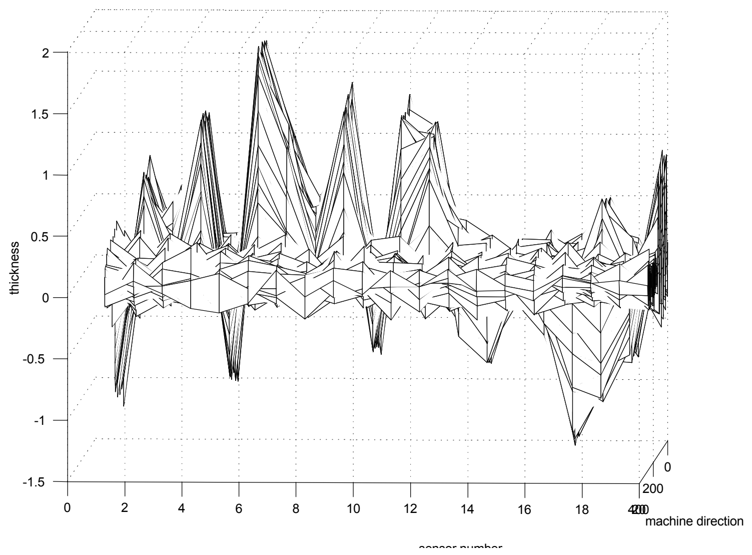

Figure 5-8: Coefficients of the principal components computed from the second half of the data

the principal components have been designed - note that the second half is well represented, while the first half fails fairly dramatically, due to the low concentration of energy in the first three components, evident from Figure 5-8 .

We can plot the trajectory of the state estimate in coefficient space, illustrate in Figure 5-10. This illustrates the linked cyclic nature of the first two components for the first half of the time period, followed by the random noise apparent in the second half in this coordinate system.

Finally, the same estimation problem is solved using the multiscale state estimator, the state estimation comparable to the Karhunen-Loeve approach is illustrated in Figure 5-11. This should be directly compared to the data in Figure 5-2, from which the state estimation is made. Note that denoising, and feature identification is successful, while the solution retains more of the noise that the Karhunen-Loeve chopping does, illustrated in figure 5-6.

The multiscale state estimator produces a solution that is more hierarchical in nature and can be stopped at any level of resolution. The solution at eight levels of resolution is illustrated in figures 5-11 to 5-18. The levels retain a decreasing amount of detail, corresponding to various frequency bands. Once the level is passed corresponding to the frequency of the sine-like

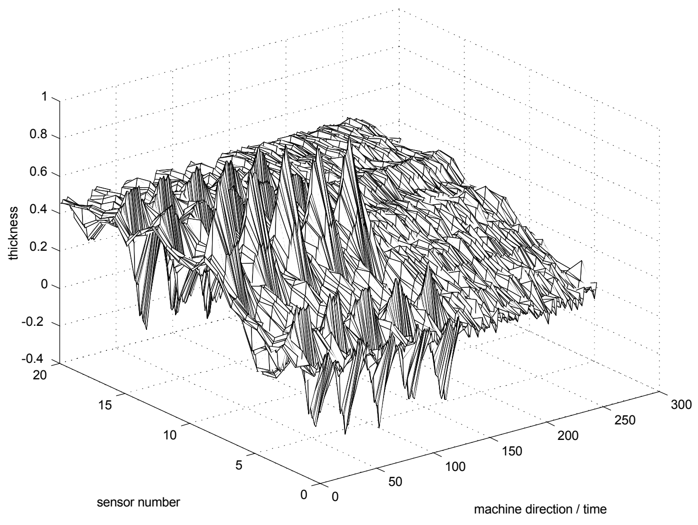

Figure 5-9: State estimation using principal components from the second half of the data.

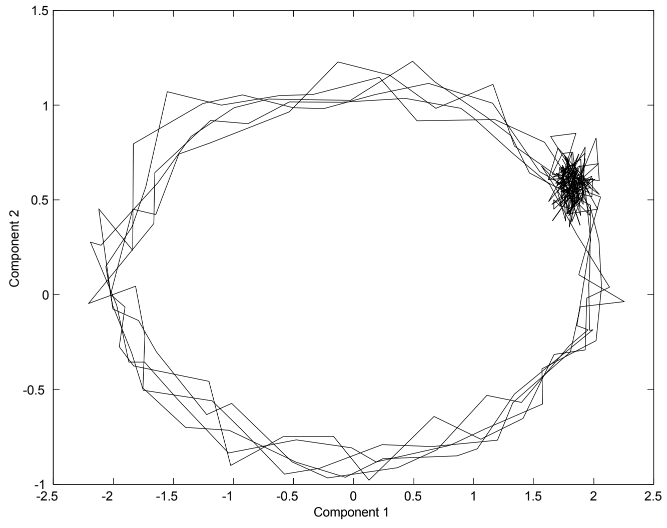

Figure 5-10: First two principal component coefficients in coefficient space

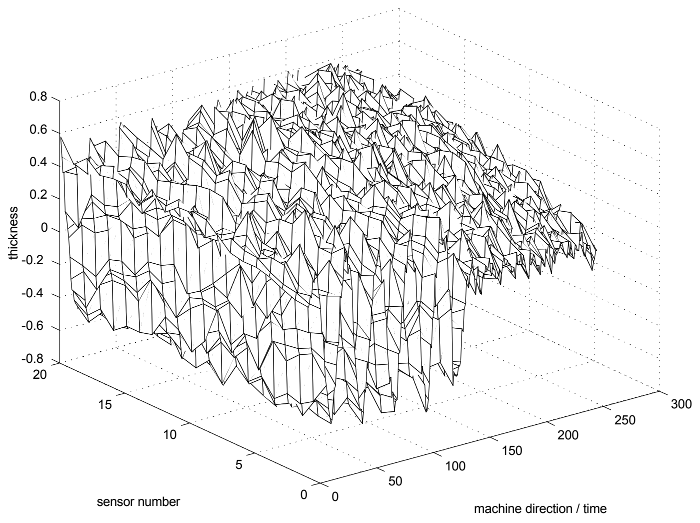

Figure 5-11: Multiscale state estimate of paper rolling measurements - most detailed level 8.

waves, there is little qualitative difference between levels, suggesting that further computation is pointless. This suggests that the multiscale state estimator would be very useful for identifying coherent structures within a specified frequency band - incorporating all the data from a time series, and summarising it in an optimal way. The wavelet decomposition used to generate these state estimates is indeed the Karhunen Loeve expansion for an integrated white noise process, as shown by Wornell, [5].

5.5 Conclusions

The Karhunen Loeve transform and the multiscale state estimator provide two different approaches to the generation of state estimates. The Karhunen Loeve transform aims to reduce computation by identifying the optimal basis with which to represent the data, and thus to eliminate unimportant information in the irrelevant part of the space, while the multiscale state estimator reduces computation in a hierarchical way, by identifying a level of interest, and halting computation once that level is reached.

Both approaches provide reasonable state estimates of the underlying data, and they reduce to each other in the event of pure white noise, for which the Haar decomposition is the Karhunen

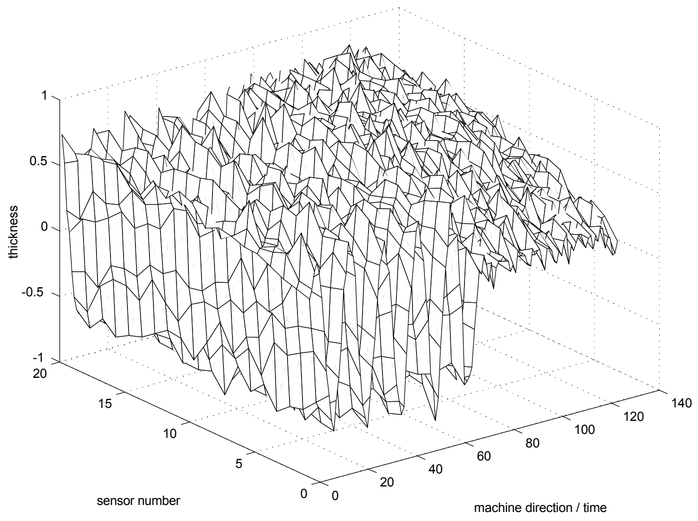

Figure 5-12: Multiscale state estimate of paper rolling measurements at level 7

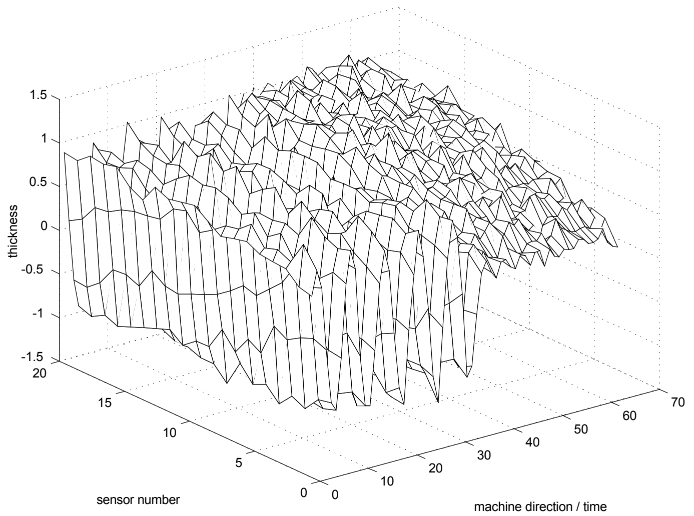

Figure 5-13: Multiscale state estimate of paper rolling measurements - level 6

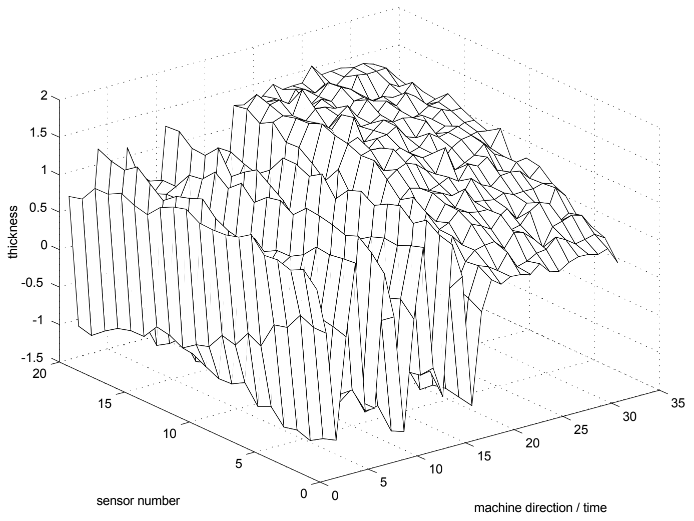

Figure 5-14: Multiscale state estimate of paper rolling measurements - level 5

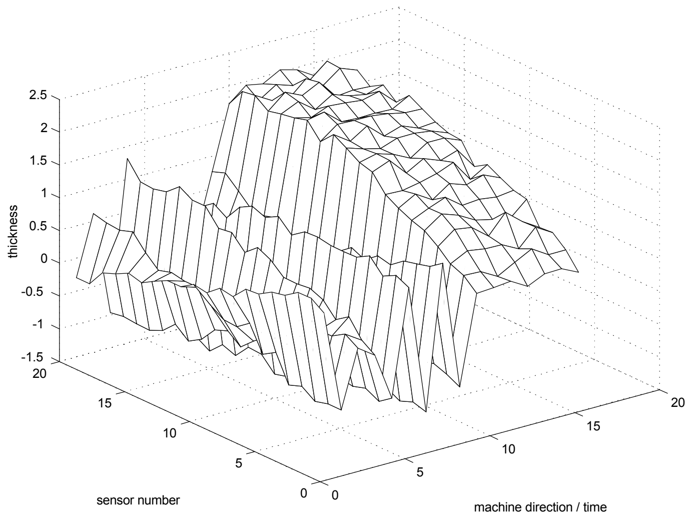

Figure 5-15: Multiscale state estimate of paper rolling measurements - level 4

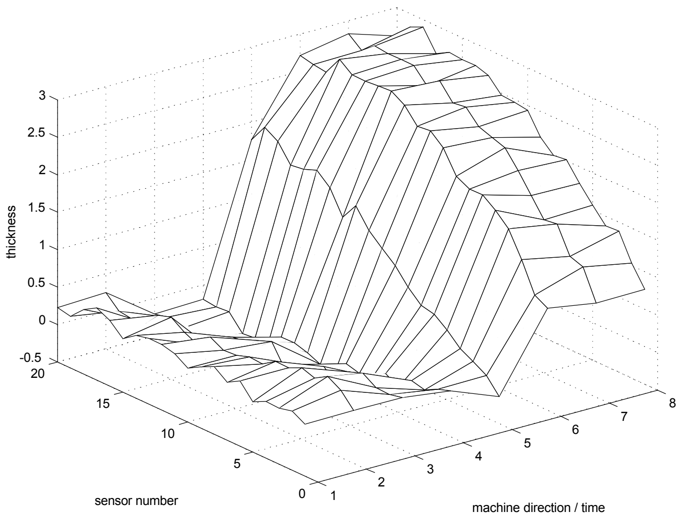

Figure 5-16: Multiscale state estimate of paper rolling measurements - level 3

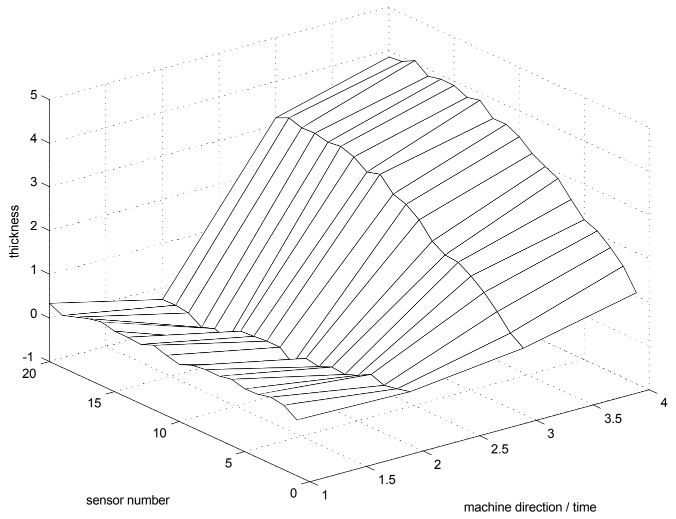

Figure 5-17: Multiscale state estimate of paper rolling measurements - level 2

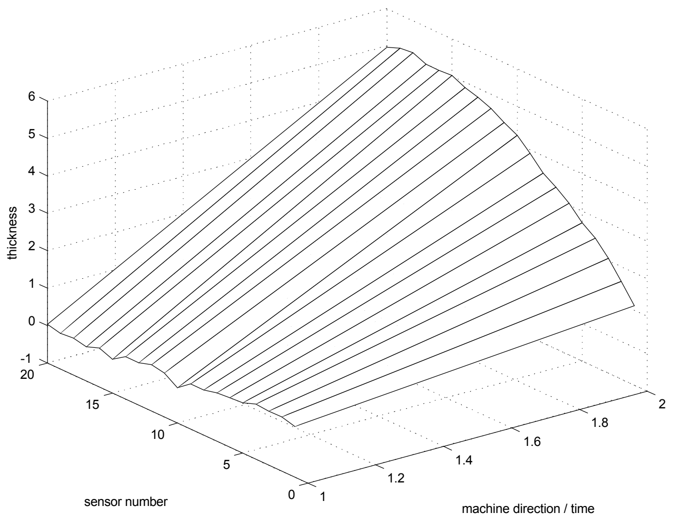

Figure 5-18: Multiscale state estimate of paper rolling measurements - top level

Loeve basis set, [1], [5].

Bibliography

[1] G.W, Wornell and A.V.. Oppenheim. "Estimation of Fractal Signals from Noisy Measurements Using Wavelets.," *IEE Trans. Signal Process*, *40*:611–623 (1992).

[2] Moore, Bruce. "Principal Component Analysis in Linear Systems: Controllability, Observability and Model Reduction," *IEEE Transactions on Automatic Control*, *26*:17–32 (1981).

[3] Rigopoulos, A. and Y. Arkun. "Identification of Full Profile Disturbance Models for Sheet Forming Processes," *AIChE Journal preprint* (2000).

[4] Tatang, Menner A. *Direct Incorporation of Uncertainty in Chemical and Environmental Engineering Systems*. PhD dissertation, Massachusetts Institute of Technology, 1995.

[5] Wornell, G.W. "Karhunen-Loeve Like Exapnsion for 1/F Processes Via Wavelets," *IEEE Trans. Inform. Theory*, *36*:859–861 (1990).

[6] Wyckoff, P. *Numerical Solution of Differential Equations Through Empirical Eigenfunction Expansions*. PhD dissertation, Massachusetts Institute of Technology, 1995.

Chapter 6

Model Predictive Control with Multiscale State Estimation

6.1 Introduction

Model predictive control is a well established optimisation-based technique for the control of chemical processes [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], and [17]. The basic structure of the algorithm is that at each time point, an open-loop optimal control problem is solved using a model of the process. The control strategy obtained as the solution to this problem is implemented until further measurements become available. The model itself is constructed from prior information. In many cases, direct state measurements are not available and it becomes necessary to perform state estimation using the prior measurements, in order to produce useful information for the controller.

Since the construction of the control policy is optimisation based, there is a natural way to incorporate inequality constraints - a major benefit of the model predictive control approach. Rao and Rawlings, [16], make the point that in many industrial situations, the desire for maximum profit regularly causes one or more of the process constraints to be active at the optimal operation point. Therefore it seems that the operation at constraints should be regarded as the normal mode of operation rather than as an exception in chemical engineering.

Most industrial processes are motivated purely by profit, but constrained by the physical equipment such as in this example. Production rates of exothermic chemical processes are usually dependent on the flowrate of raw materials, and these are usually constrained by the heat duty that can be removed by the cooling system. Operating at maximum production rate is typically done at the heat duty constraint. Many disturbances may enter the process, suggesting that a control strategy incorporating this constraint would be useful.

Previous work in Model Predictive Control has recognised that stabilizibilty of the closed loop problem may not be guaranteed by a number of control algorithms, and the problem is usually dealt with by invoking an infinite horizon argument in the controller. The use of the multiscale model predictive controller as a stabilizing closed loop control is discussed at length in the Ph.D. thesis of Orhan Karsligil, [6].

6.2 The Shell Standard Control Problem

In December 1986, Shell Development, describing themselves as "a leader in Process Control", brought together some of the leading academic and industrial researchers in process control [14], [15]. The Shell Process Control Workshop provided a standard control problem that incorporated sufficient modelling detail to test various control design methodologies.

The column has a gaseous feed stream which carries all of the heat required by the column. There are three intermediate reflux loops which are used to remove heat from the column to produce the desired product purity. Since these heat exchangers are used to heat other parts of the plant, the heat that they remove is variable. The bottom heat duty is controlled by the production of steam, which can be specified, thus bottom heat duty is a manipulated variable, while the other two heat duties are regarded as disturbances. There are three draws, of which the top and intermediate are controlled for product purity, and the bottom has unspecified product.

The problem itself contains considerable complexity, and in this case study, we will be dealing with a reduced set of complexity, for the purposes of illustrating the working of the estimator and model predictive controller.

The original standard control problem has the following control objectives:

1. Top and side product draws must be kept at specification of 0.0 ± 0.005 at steady state.

2. The heat removal in the bottom circulating reflux is to be maximised, thereby maximising

Figure 6-1: Heavy Oil Fractionator for the Shell Standard Control Problem

the amount of steam produced by the process. In the formulation given, heat duty represents heat input to the system, thus one must minimize heat duty to the bottom reflux.

3. Unmeasured reflux disturbances should be rejected. These enter the column through the top and intermediate columns and are the result of disturbances in the heat duty requirements of other columns. The disturbances are assumed to be in the range $[-0.5, 0.5]$, and it is desired that disturbances be rejected even when one or both end point analyzers fail.

   Further, the Shell standard control problem is equipped with constraints and a set of inputs and outputs.

   All draws must be within $[-0.5, 0.5]$.

4. The bottom reflux heat duty must be $[-0.5, 0.5]$.

5. All derivatives of manipulated variables must have a maximum of 0.05/minute.

6. The fastest sampling time is 1 minute.

7. The bottom reflux draw has a lower bound of -0.5.

8. The top endpoint must lie $[-0.5, 0.5]$.

The case study that we will be performing utilises all of these constraints, and concentrates on objectives 1 and 3 from the original control objectives. We have not concentrated on the maximisation of steam production since the goal of this case study is to demonstrate the state estimator.

The process models for the plant are described as transfer functions of first order processes with dead time of the standard form:

$$G = \frac{Ke^{-\theta s}}{\tau s + 1} \tag{6.1}$$

The inputs for this system are the top draw, the side draw, and the three reflux duties. The complete set of outputs is the top end point, the side end point, and the temperatures of the top draw, the upper reflux, the side draw, the intermediate reflux and the bottom draw. The complete set of inputs and outputs is given in Prett and Morari's workshop proceedings.

				Reflux Duties	
Output	**Top Draw**	**Side Draw**	**Bottom**	**Intermediate**	**Upper**
Top End Point	$\frac{4.05e^{-27s}}{50s+1}$	$\frac{1.77e^{-28s}}{60s+1}$	$\frac{5.88e^{-27s}}{50s+1}$	$\frac{1.20e^{-27s}}{45s+1}$	$\frac{1.44e^{-27s}}{40s+1}$
Side End Point	$\frac{5.39e^{-18s}}{50s+1}$	$\frac{5.72e^{-14s}}{60s+1}$	$\frac{6.90e^{-15s}}{40s+1}$	$\frac{1.52e^{-15s}}{25s+1}$	$\frac{1.83e^{-15s}}{20s+1}$
Bottoms Reflux T	$\frac{4.38e^{-20s}}{33s+1}$	$\frac{4.42e^{-22s}}{44s+1}$	$\frac{7.20}{19s+1}$	$\frac{1.14}{27s+1}$	$\frac{1.26}{32s+1}$

Table 6.1: Table Process Model Parameters for the Shell Heavy Fractionator

				Reflux Duties	
Output	**Top Draw**	**Side Draw**	**Bottom**	**Intermediate**	**Upper**
Top End Point	$4.05 + 2.11\varepsilon_1$	$1.77 + 0.39\varepsilon_2$	$5.88 + 0.59\varepsilon_3$	$1.20 + 0.12\varepsilon_4$	$1.44 + 0.16\varepsilon_5$
Side End Point	$5.39 + 3.29\varepsilon_1$	$5.72 + 0.57\varepsilon_2$	$6.90 + 0.89\varepsilon_3$	$1.52 + 0.13\varepsilon_4$	$1.83 + 0.13\varepsilon_5$
Bottoms Reflux T	$4.38 + 3.11\varepsilon_1$	$4.42 + 0.73\varepsilon_2$	$7.20 + 1.33\varepsilon_3$	$1.14 + 0.18\varepsilon_4$	$1.26 + 0.18\varepsilon_5$

Table 6.2: Table Uncertainty in the Gains for the Shell Heavy Fractionator

In our case study, we will focus on a reduced set of outputs - specifically, the top end point, the side end point and the bottoms reflux temperature, and our transfer functions will reflect this.

The parameters for the various transfer functions are listed in Table 6.1.

In addition to this model, the standard problem contains estimates for the gain of the models as shown in Table 6.2.

In all cases, ε_k is in the range $[-1, 1]$.

The five prototype test cases from the workshop are listed below.

Demonstrate through simulation that the proposed controller satisfies the control objectives without violating the control constraints for the following plants within the uncertainty set. Assume that all inputs and outputs are initially at zero, magnitudes for the upper and intermediate reflux duty step changes are indicated below.

1. $\varepsilon_k = 0$. Upper reflux duty = 0.5. Intermediate reflux duty = 0.5.

2. $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = -1; \varepsilon_4 = \varepsilon_5 = 1$. Upper reflux duty = -0.5. Intermediate reflux duty=-0.5.

3. $\varepsilon_1 = \varepsilon_3 = \varepsilon_4 = \varepsilon_5 = 1, \varepsilon_2 = -1$. Upper reflux duty = -0.5. Intermediate reflux duty=-0.5.

4. $\varepsilon_1 = \varepsilon_2 = \varepsilon_3 = \varepsilon_4 = \varepsilon_5 = 1$. Upper reflux duty = -0.5. Intermediate reflux duty = -0.5.

5. $\varepsilon_1 = -1, \varepsilon_2 = 1, \varepsilon_3 = \varepsilon_4 = \varepsilon_5 = 0$. Upper reflux duty = 0.5. Intermediate reflux duty

= -0.5.

6.3 The multiscale approach

The original presentation of the problem has all of the models in transfer function form. The multiscale state estimator, and multiscale model predictive controller require that the models be in state space format, that is the dynamic system should have the form

$$x_{t+1} = Ax_t + Bu_t + w_t \tag{6.2}$$

$$y_t = Cx_t + Du_t \tag{6.3}$$

The formulation provided has a steady state at the origin, suggesting that deviation variables are used. These are useful since the steady state behaviour is separated from the dynamic elements of the system. Once the dynamic system has been transformed, the constraints, inputs and outputs will be in a form suitable for the multiscale state estimation algorithm .

Rawlings and Rao [16] make a number of points about the structure of the constraints in their paper. They note that the incorporation of constraints can produce problems in the presence of large disturbances. In a disturbance rejection control problem with constrained input variables, one can always find a disturbance large enough so that the inputs cannot reject it without violating constraints. Thus care must be taken to check for problems that become inherently infeasible.

In the Shell control problem, the origin is always feasible and is the steady state solution for the undistrubed plant, due to the use of deviation variables. Steady state disturbances in the uncontrolled inputs may mean that the range of the plant shifts from its centre around the origin, or that it is necessary to have a non-zero control variable to reject these external unmodelled distrubances. Input constraints may make the plant uncontrollable for a given set of disturbances. This aspect should be addressed by identifying a stable feasible solution by simulation, before implementing the controller on a live plant.

Our case study begins by converting the input-output models to state space, by approximating each of the time delay elements by the Pade approximation.

$$e^{-\theta s} \approx 1 - \theta s \tag{6.4}$$

These transfer functions are multiplied by the numerators of the relevant transfer functions.

A set of matrices (A, B, C, D) is defined to be a realisation of the transfer function $H(s)$ if

$$H(s) = C(sI - A)^{-1}B + D \tag{6.5}$$

The transformation from the Laplace domain to state space is non-unique, and our selection has been the minimum realisation, the realisation that has (A, B) reachable and (A, C) observable. The procedure is automated in MATLAB, and the reader is referred to the source code, or a book on linear systems theory full detailed descriptions on the generation of realisations. It is taken directly from a set of notes by Dahleh

Here follows a brief description of Gilbert's realization, a technique for generating a minimal realisation for MIMO systems. It is taken directly from a set of notes by Dahleh.

Suppose we have a matrix transfer function and we factor out the least common denominator polynomial $d(s)$. If it has no repeated roots then it is possible to construct a minimal realization. If we apply a partial fraction expansion to each of the elements of H(s) and collect residues for each distinct pole, then the transfer function can be written in the following form:

$$H(s) = \sum_{i=1}^{k} \frac{1}{s - s_i} H_i \tag{6.6}$$

where H_i is $p \times m$, for p outputs and m inputs. H_i has rank r_i where r_i is the minimum number of independent poles with location s_i required to realize $H(s)$. It follows from the rank of H_i that it can be decomposed as te product of two matrices with full column and full row rank respectively:

$$H_i = C_i B_i \tag{6.7}$$

The elements of each H_i are just the residues of the termwise partial fraction expansion of $H(s)$

at each pole s_i.

$$\begin{aligned}
A &= \begin{bmatrix} s_1 I_{r1} & 0 & 0 & 0 & 0 \\ 0 & s_2 I_{r2} & 0 & 0 & 0 \\ 0 & 0 & s_3 I_{r3} & 0 & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & 0 & s_k I_{rk} \end{bmatrix} && (6.8) \\
B &= \begin{bmatrix} B_1 & B_2 & B_3 & \cdots & B_k \end{bmatrix}^T && (6.9) \\
C &= \begin{bmatrix} C_1 & C_2 & C_3 & \cdots & C_k \end{bmatrix} && (6.10) \\
I_x &= \text{identity matrix of dimension } x && (6.11)
\end{aligned}$$

Further minimal realisations can be constructed using similarity matrices.

The resulting matrix may be poorly conditioned - which produces condition number problems in the state estimation algorithm, particularly in the computation of top level nodes, which rely on the inversion of the state matrix raised to a large power. Thus we try to eliminate small states by choosing a tolerance for the size of state that we regard as relevant, and eliminate these. Note that the full system can still be used for simulation, but the reduced system is more robust for estimation since some function of it must be inverted.

The actual model produced in this simulation has sixteen states, which are automatically reduced to fourteen by MATLAB, due to small singular values. Since the matrix is large, and barely invertible, which we need for the state estimation. we thus eliminate the next six states by setting their derivatives to zero, and solving the resulting system for the remaining state equations. The reduced system is used for the estimation, and in the control algorithm, and it has three outputs, three inputs and eight states. With this dynamic system, the matrices are ready to be used in the multiscale state estimator algorithm.

The states in this system represent the model outputs and various derivatives of these outputs, and combinations of these. This becomes clear if one inspects the structure of the A matrix of the minimal realisation, which is block diagonal except for the bottom few rows. This is typical for higher order systems reduced to first order using state augmentation, with higher order states representing various derivatives.

The origin is known to be a steady state solution to the dynamic system, and this is used as an initial condition, from which all state estimation and control will be done. In practice, one would have historical data from previous operations, such as the moving horizon approach discussed in Section 3.7, from the start up or from experiments. In the theoretical world, this must be created before the algorithm can commence. Alternately a steady state assumption can be made up for start up. Typically one would need a number of points equal to 2^n where n is the number of levels on the tree.

The uncertainty parameters in the model arise from the uncertainty in the gain matrix, as well as from the presence of unmodelled disturbances. These are lumped into the w variables in the state estimation algorithm. Further, it is assumed that measurements will be subject to some measurement uncertainty, the order of magnitude of which can be estimated from prior experiments, or from knowledge about the equipment.

We construct the tree dynamic system for the model, and produce a set of simulated data to initialise the state estimator. At each point after this, the most recent measurement is received, and the multiscale state estimator is used to produce an optimal state estimation of data from a fixed moving horizon in the past. The first point in the state estimation uses a prior estimate with uncertainty, obtained from previous state estimation calculations. The details of this section of the algorithm can be found in Section 2-6 This allows for changes in the plant to occur without adversely affecting state estimates for long periods of time. Clearly, the length of this horizon will be a tuning parameter, and we choose it to be comparable to the length of the control horizon.

Results of the state estimator are illustrted in Figure6-2. The dotted lines present the state estimates, while the solid lines represent the true state generated through simulation. The offsets are due to the modelling errors. The state estimation produces a set of uncertainties in the model that will be used in the prediction estimates in the controller. The uncertainties and the state estimates are sent to the controller, which returns a set of inputs to be implemented before the next measurement is available. This procedure is repeated at each time step.

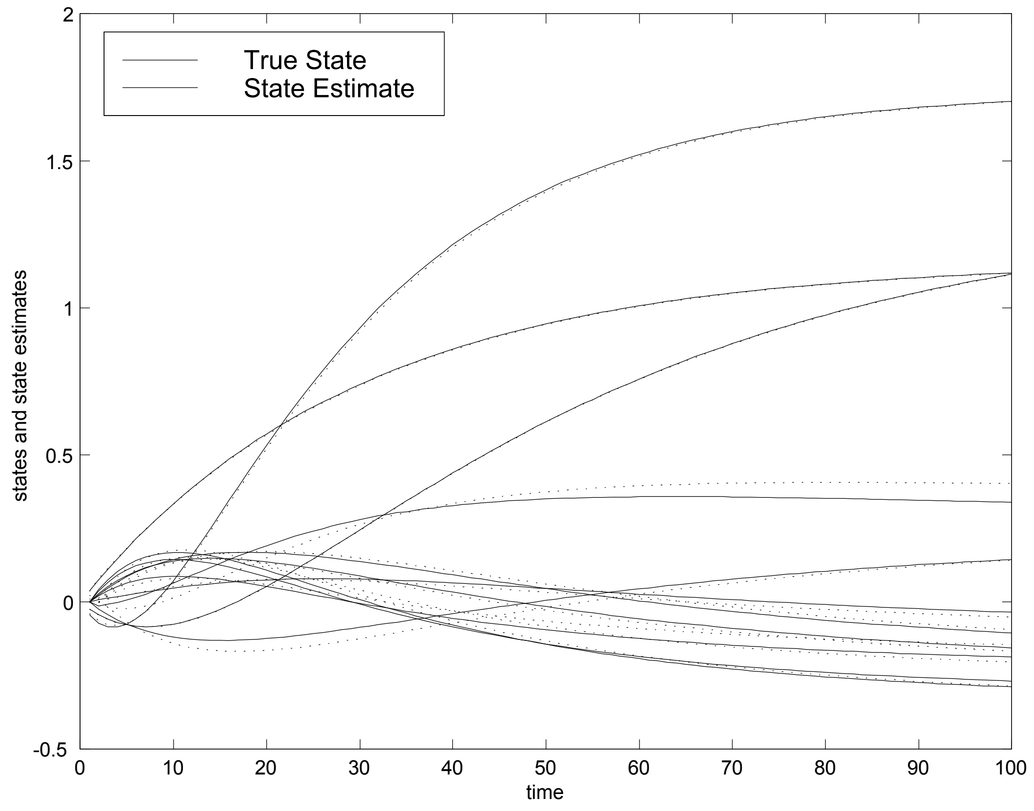

Figure 6-2: Successful state estimation by the multiscale state estimator

6.4 The multiscale model predictive controller

This controller was designed as part of a larger project within the LISPE group, to develop multiscale tools for chemical engineering problems [6], [2], [1]. The model predictive controller in multiscale is presented *in toto* in the Ph.D. thesis of Orhan Karsligil. A brief description of the process is given here.

It is assumed that a state estimate at the current point is available, as well as a set of prediction errors at all points on the tree. These are provided by the state estimator. The controller has a target path - the reference path - which the state trajectories aim to track, either due to set point changes or due to unmodelled disturbances in the plant. Note that the state trajectory may be expressed in terms of the output variables - in this case we want the output variables to follow some trajectory, not the states. This requires some recasting of the control algorithm.

A feature of the multiscale model predictive control algorithm is the construction of a reference path that is smooth so that constraints on the input variables will be satisfied in the absence of disturbances.

The model predictive controller selects an optimal length for the controller horizon in an iterative way. At each level of the multiscale tree from the zeroth level, a simple, blocked input strategy is sought that satisfies the input constraints, and achieves the desired control objective in the absence of future disturbances. If this cannot be achieved at the current level, another level is added, until a level on the tree is reached, where a feasible solution exists. At this level, referred to as the variable horizon level, the complete quadratic program is solved for the optimal open loop control strategy. This strategy is sent to the real plant (or simulator) and implemented until another measurement becomes available.

The blocked solution, which will generally be suboptimal, is characterised by repeated input variables of the same value, simply to reduce computation time by reducing the number of variables, purely for this variable horizon computation which is not required accurately..

The thesis examines this problem in considerably greater detail and provides stability arguments.

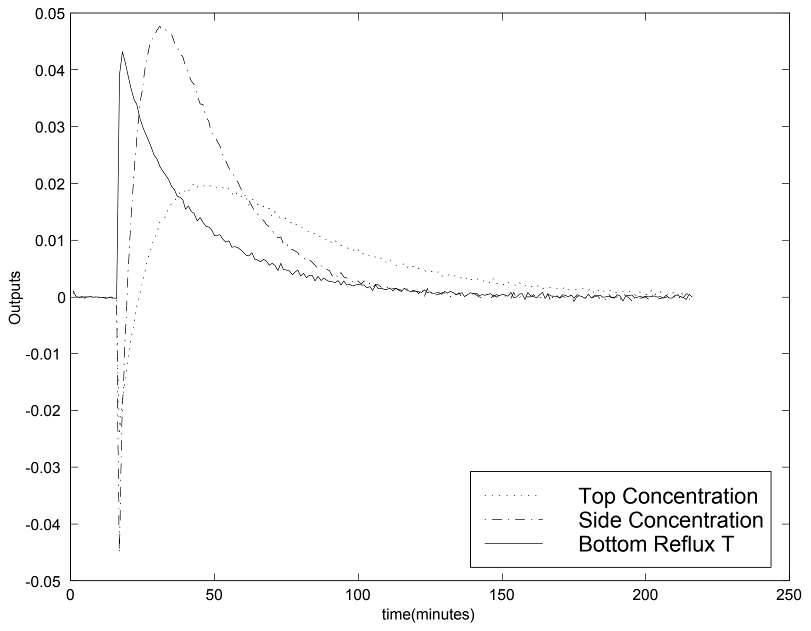

Figure 6-3: Shell Standard Case Study One Outputs

6.5 Case studies

The five cases studies presented here represent the cases suggested by the Shell Process Control team as a representative set of problems, where pertinent problems for controllers would be expected to show up.

The models used to drive the case studies were constructed using the minimal realisations discussed above. The full set of states was used for simulation, while the control algorithm uses a reduced number of states for its system. Further, the Standard Control problem suggests uncertainties for the plant. Again, the controller would be unaware of plant uncertainty, but would be required to deal with it, so real plant data uses the model with the nominal uncertainty, while the controller assumes the steady state, or model, version of the plant.

The strategy in all cases was to simulate 20 minutes at steady state, the to apply the step change in the relevant input variable, and then to use model predictive control every minute to reject the disturbance, in the presence of plant uncertainty..

Figures 6-4 and 6-3 show the first case study original plant with perfect plant model, and positive disturbances in both the upper and intermediate reflux duties. The outputs produce an

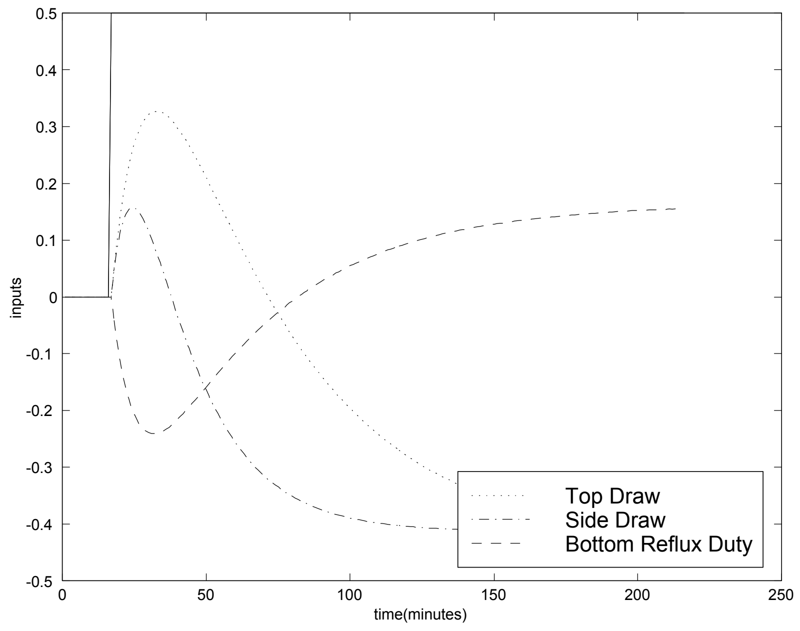

Figure 6-4: Shell Standard Case Study One Inputs

inverse response initially, but this settles to a steady state in roughly 200 minutes, and possibly shorter if we follow the design spec of 0.005 at steady state.

Case study two is illustrated in figures 6-5 and 6-6. This time there is some uncertainty in all of the plant variables, leading to a slightly longer decay than for the perfectly known model. Nevertheless the disturbance is almost completely rejected within about 200 minutes.

Figures 6-7 and 6-8 illustrate the rejection of the same disturbance, but with different model uncertainties. The plots are qualitatively the same as 6-5 and 6-6 suggesting that the model uncertainty is being correctly captured by the model predictive control algorithm. Figures 6-9 and 6-10 show a slightly different feature, where the disturbances to be rejected are of opposite sign, and thus cancel out some of the effect of each other. Note however that the outputs exhibit a clear oscillatory behaviour for the first time in the sequence. Nevertheless, there is a good rejection of the disturbance.Figures 6-11 and 6-12 return to reflux disturbances in the same direction, but with a reduced set of model uncertainties restricted to the top and the side draw. This gives slightly improved performance over case studies 2 and 3, where all models were uncertain.

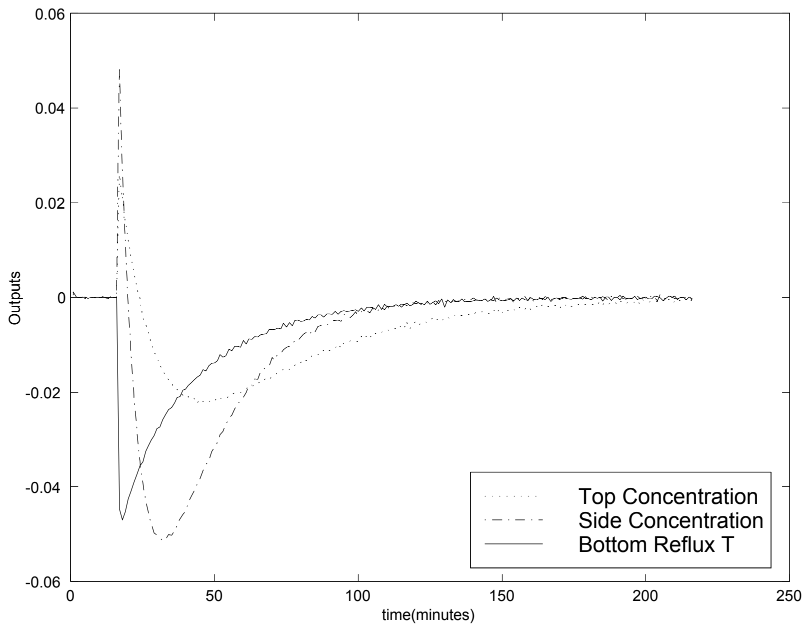

Figure 6-5: Shell Standard Case Study Two Outputs

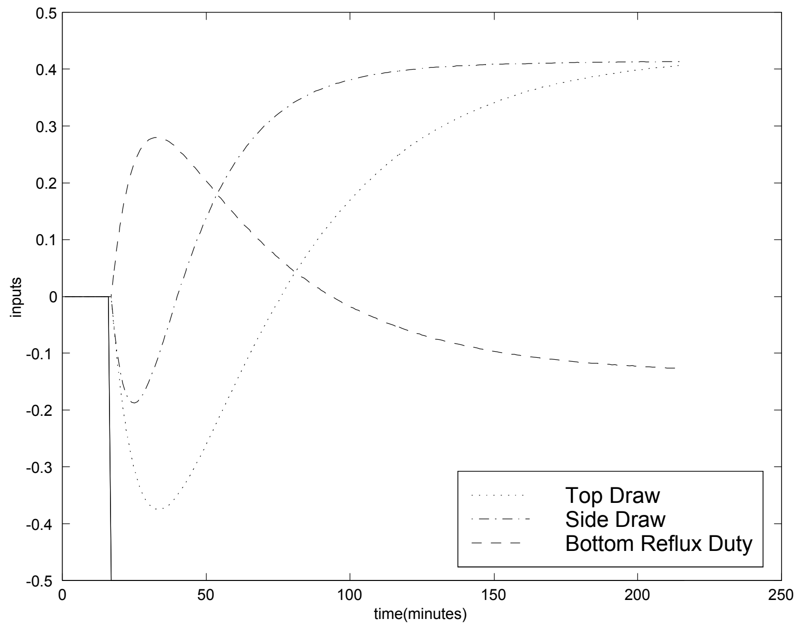

Figure 6-6: Shell Standard Case Study Two Inputs

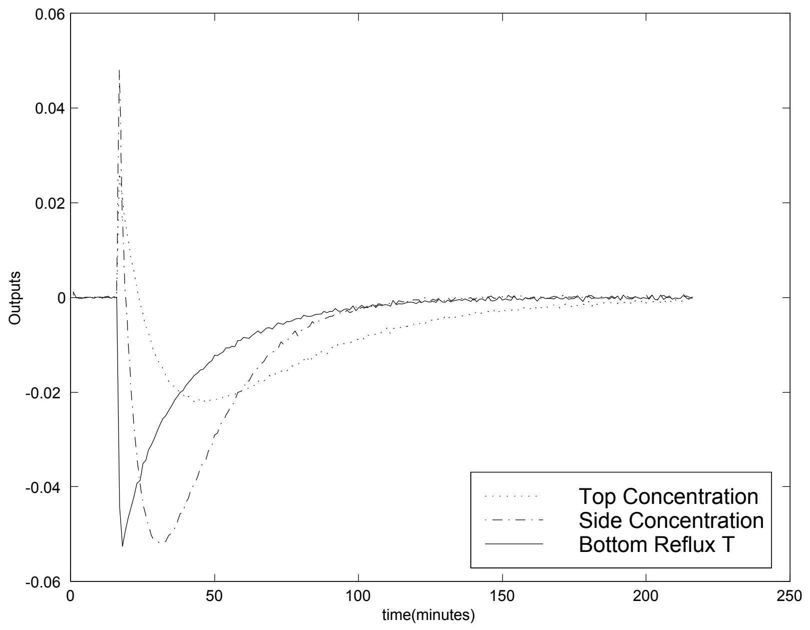

Figure 6-7: Shell Standard Case Study Three Outputs

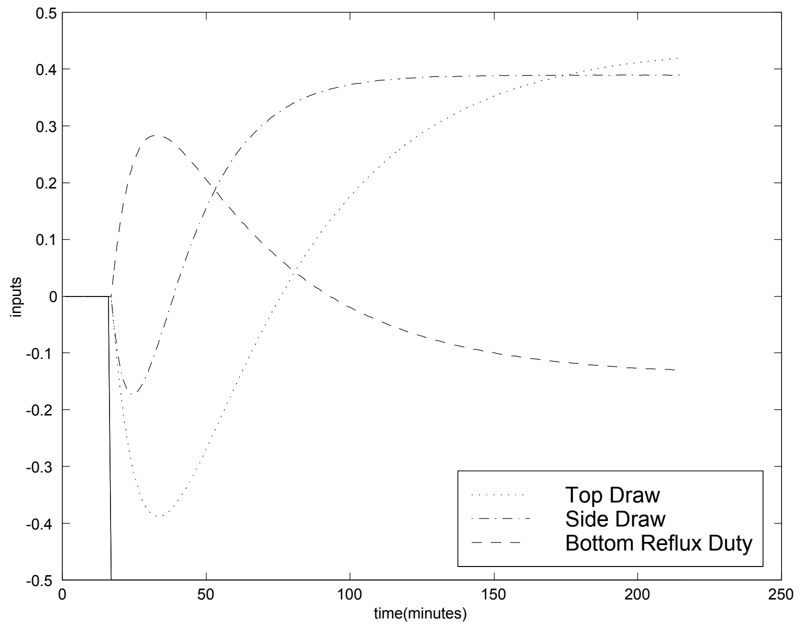

Figure 6-8: Shell Standard Case Study Three Inputs

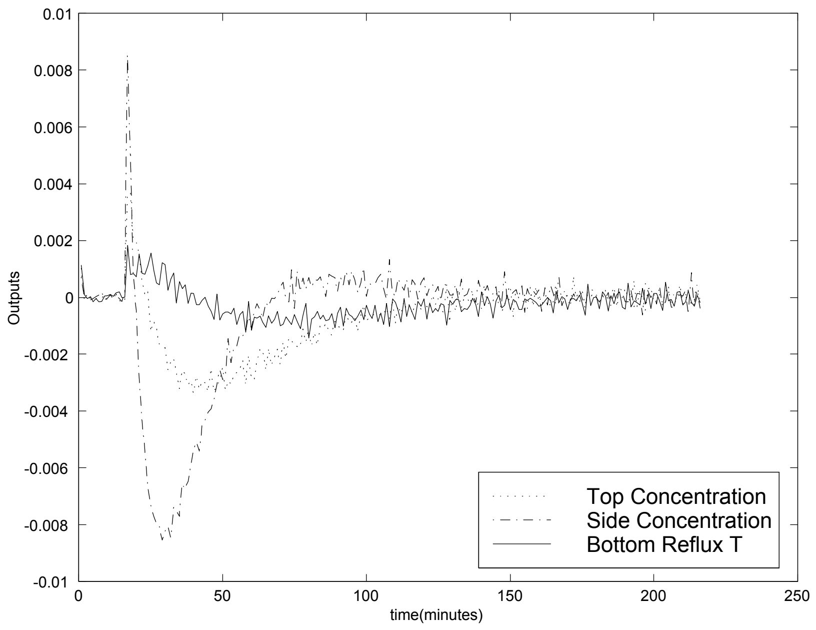

Figure 6-9: Shell Standard Case Study Four Outputs

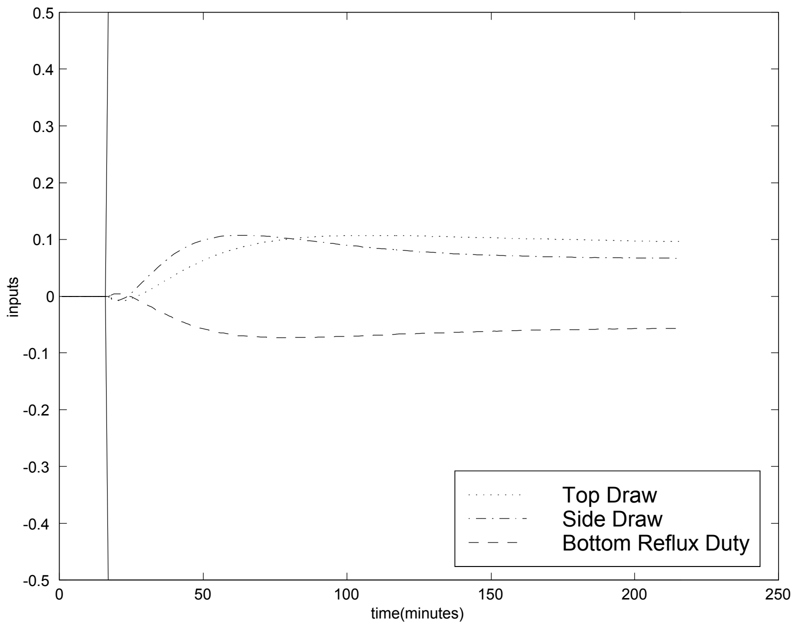

Figure 6-10: Shell Standard Case Study Four Inputs

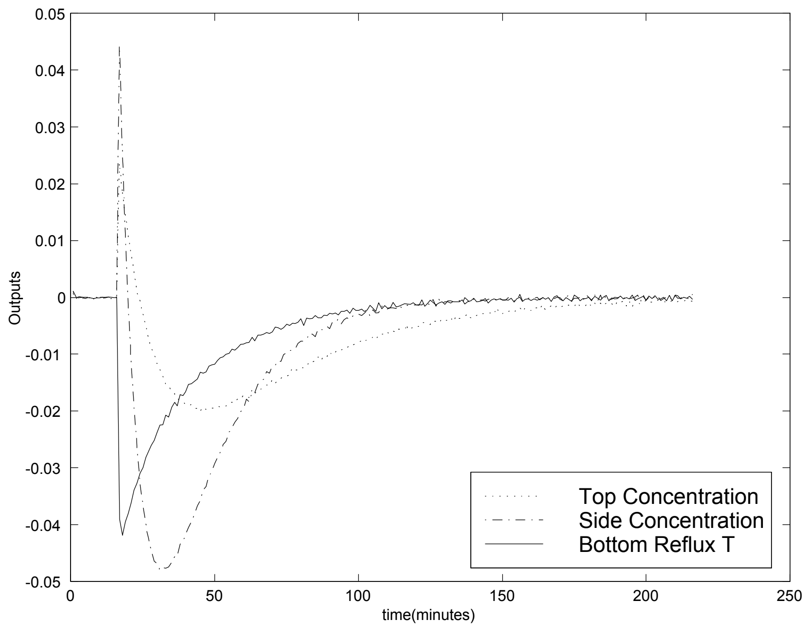

Figure 6-11: Shell Standard Case Study Five Outputs

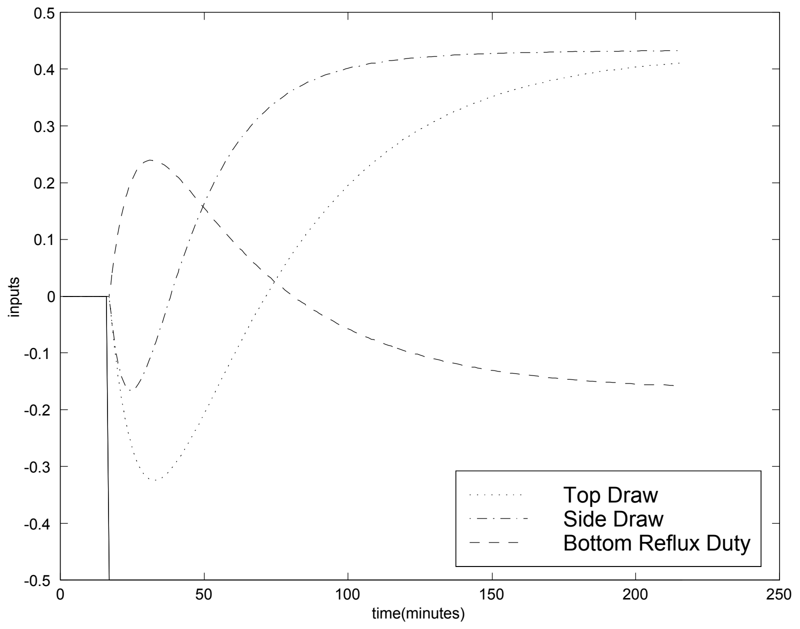

Figure 6-12: Shell Standard Case Study Five Inputs

6.6 Summary

In this chapter we have discussed the use of the multiscale state estimator in a model predictive control algorithm. There is little difference between the results obtained for the various different model uncertainty variables. The multiscale state estimator has been used to assist a model predictive control algorithm applied to a state space implementation of the Shell Standard Control Problem - all five parts of the problem. In all cases, steady state was successfully reached roughly 200 minutes after the application of the step change disturbances. Since these disturbances were at the maximum allowed bounds by the physical constraints imposed on the problem, it seems reasonable to expect this to be an upper bound for any disturbance. Case study 4, despite its relatively uncertain plants compared to 1 and 5, produced the smallest response to the disturbances.

Bibliography

[1] Bakshi, B. and G. Stephanopoulos. "Wave-Net: A Multiresolution, Hierarchical Neural Network with Localized Learning," *AICHE Journal*, *39* (1993).

[2] Bakshi, B. and G. Stephanopoulos. "Represenatation of Process Trends-III: Multiscale Extraction of Trends from Process Data," *Computers Chem. Engng*, *18*:267–302 (1994).

[3] Cutler, C.R. and B.L. Ramaker. "Dynamic Matrix Control- a Computer Control Algorithm," *Proceedings of the Joint Automatics Control Conference* (1980).

[4] Geo. Stephanopoulos, O. Karsligil. *Multi-scale model predictive control*. Technical Report, MIT-LISPE, 1998.

[5] J. Eaton, J.B. Rawlings. "Model predictive control," *Chemical Engineering Science* (1991).

[6] Karsligil, O. *Multiscale Model Predictive Control*. PhD dissertation, Massachusetts Institute of Technology, 2000.

[7] Lee, J. "Recent Advances in Model Predictive Control and Other Related Areas," *CPC-V Conference Proceedings*, 1–21 (1994).

[8] Lee, J.H. "Recent Advances in Model Predictive Control and Other Related Areas," *CPC-V Conference Proceedings* (1995).

[9] Lee, J., M. Morari C. Garcia. "State-Space Interpretation of Model Predictive Control," *Automatica*, *30*:707–717 (1994).

[10] Mayne, D.Q. "Nonlinear Model Predictive Control: An Assessment," *Preprints of Chemical Process Control -V* (1996).

[11] Morari, M. “Model Predictive Control: Multivariable Control Technique of CHoice in the 1990’s?,” (1994).

[12] Muske, K.R. and J.B. Rawlings. “Model Predictive Control with Linear Models,” *AICHE Journal*, *39*(2):262–287 (1993).

[13] Nikolaou, M. “Model Predictive Controllers: A Crytical Synthesis of Theory and Industrial Needs,” (1998). Preprint.

[14] Prett, D.M. and Morari M., editors. *Shell Process Control Workshop*, Houston, TX: Butterworth, 1987.

[15] Prett, G. and C.E. Garcia, editors. *The Shell Process Control Workshop*, Buterworth Publishers, Stoneham, MA, 1989.

[16] Rao, C.V. and J.B. Rawlings. “Steady States and Constraints in Model Predictive Control,” *AICHE Journal*, *45*(6):1266–1278 (1999).

[17] Yu, Zheng H., Wei Li Jay Lee. “State Estimation Based Model Predictive Control Applied to Shell Control Problem: A Case Study,” *Chemical Engineering Science*, *49*:285–301 (1994).

Chapter 7

The Terephthalic Acid Plant Case Study

7.1 Introduction

The motivation for this case study was to find a chemically relevant use of the multiscale state estimator, that went further than the initial linear assumption of the state dynamic model in the construction of the estimator. In this case study, the primary reactor in a continuous terephthalic acid plant is initially described by a nonlinear model, due to products of composition and functions of temperature, all of which are changing continuously. A further complication is the nonlinear nature of the phase equilibrium calculation, which affects the mass and energy balances, and must be inferred from indirect measurements.

7.2 Overview of the terephthalic acid plant

The plant is designed to produce the terephthalic acid monomer continuously, using the plant represented in Figure 7-1, [1]. The reaction is occurs as a series of three oxidation reactions that use a platinum catalyst in an acetic acid medium. The primary chemical species are listed in Table The three reactions occur in an initial reactor, at high pressure. The high temperature in this reactor is due to the highly exothermic nature of the reactions, and cooling is used to prevent unwanted side reactions that occur above $225°C$. The reaction is followed by a

Figure 7-1: Schematic of Terephthalic Acid Plant

flash tank, a separation system to remove the monomer from the e- uent, a series of distillation columns to remove the unwanted heavies and the unused catalyst, and to concentrate the acetic acid for recycling back to the reactor. This case study will focus on the first reactor.

The model we are using consists of a reactor in which the three oxidation reactions occur in series. The liquid feed stream consists of a combination of pure xylene, and a recycle stream from the separations section of the plant consisting of acetic acid with traces of water and toluic acid, the byproducts and intermediates from the oxidation reaction. Essentially all of the acetic acid is recycled for economic reasons. The gas feed stream consists of high pressure air from a multistage compressor, fed from below through a sparger. There is an overhead condenser which is used to recycle the acetic acid and water back to the reactor, and to separate it from the inert nitrogen and unused oxygen. These leave to a stripper which removes further acetic acid, before atmospheric release. The condenser is a countercurrent heat exchanger that uses

AA	acetic acid
TA	toluic acid
CBA	carboxybebzaldehyde
TPA	terephthalic acid
XY	xylene
H_2O	water
O_2	oxygen
N_2	nitrogen

Table 7.1: Names of the Principle Components

Reaction	$A(min^{-1})$	$E(kJ/mol)$	$k(min^{-1})$	$\Delta H^{rxn}(kJ/mol)$
1	1.03×10^{6}	61.1	0.403	-691
2	3.31×10^{5}	61.1	0.129	-377
3	3.47×10^{17}	168	0.965	-295

Table 7.2: Reaction Constants

cooling water at $40°C$ and returns it at a maximum of $50°C$.

The reactor itself is assumed to be a continuously stirred tank reactor, thus the concentration throughout the reactor and in the exiting liquid stream is assumed constant. The oxygen is assumed to be well distributed through the tank, and is in excess.

7.2.1 Physical data

The three reactions in series are the oxidation of xylene to toluic acid, toluic acid to carboxybenzaldehyde, and carboxybenzaldehyde to terephthalic acid. For the remainder of the discussion they will be numbered 1 to 3. The reactions occur in an acetic acid solution, with a platinum catalyst, assumed to be in constant concentration, and continuously fed with the feed xylene. The oxidation reactions satisfy the following equations, with physical constants given in the accompanying tables.

$$XY + \frac{3}{2}O_2 \xrightarrow{1} TA + H_2O \tag{7.1}$$

$$TA + O_2 \xrightarrow{2} CBA + H_2O \tag{7.2}$$

$$CBA + \frac{1}{2}O_2 \xrightarrow{3} TPA \tag{7.3}$$

Compound	AA	XY	TA	CBA	TPA	$Water$	N_2	O_2
Heat Capacity ($J/molK$)	123	182	169	175	199	75	31	31
Heat of Vaporisation (kJ/mol)	23.70	35.67	-	-	-	40.65	-	-

Table 7.3: Chemical Properties

7.3 Setting up the mass balance equations

The reactor is assumed to be continuously stirred and thus the output flow is identical in composition to the contents of the reactor. Thus molar balances over the reactor for each component can be constructed. N_{XY} is the number of moles in the reactor of component XY. k_1, k_2, and k_3 are the reaction rates of the three series reactions. F_{OUT}, in units of $/hr$, is the flowrate out of the reactor, normalised by the number of moles of liquid in the reactor. F_{OUT} is a function of the composition, the densities of each component, the volume of liquid in the reactor. The reactor is assumed to be well stirred, thus the mole fractions in the fluid within the reactor are the same as those in the exit stream. F_G is the flowrate in $moles/hr$ of the gas stream leaving in overhead to the condenser, with composition given by y_i. F_C is the flowrate in $moles/hr$ of the stream returning from the condenser, with composition $x_{i,C}$.

$$Accumulation = In - Out + Generation - Depletion \tag{7.4}$$
$$\frac{d}{dt}N_{XY} = N_{XY,IN} - F_{OUT}N_{XY} - k_1N_{XY} \tag{7.5}$$
$$\frac{d}{dt}N_{TA} = N_{TA,IN} - F_{OUT}N_{TA} + k_1N_{XY} - k_2N_{TA} \tag{7.6}$$
$$\frac{d}{dt}N_{CBA} = -F_{OUT}N_{CBA} + k_2N_{TA} - k_3N_{CBA} \tag{7.7}$$
$$\frac{d}{dt}N_{TPA} = -F_{OUT}N_{TPA} + k_3N_{CBA} \tag{7.8}$$
$$\frac{d}{dt}N_{H_2O} = -F_{OUT}N_{H_2O} + k_1N_{XY} + k_2N_{TA} - F_Gy_{H_2O} + F_Cx_{H_2O,C} \tag{7.9}$$
$$\frac{d}{dt}N_{AA} = N_{AA,IN} - F_{OUT}N_{AA} - F_Gy_{AA} + F_Cx_{AA,C} \tag{7.10}$$

The energy balance uses molar heat capacities, C_{pi} in $J/molK$, temperature of the reactor,

T in K, heats of reaction, ΔH^{rxn}, in J/mol, and heats of vaporisation, ΔH^{vap}, also in J/mol.

$$\begin{aligned}
Accumulation &= In - Out + Generation - Depletion \\
\frac{d}{dt}\left(\sum N_i C_{pi} T\right)_R &= \left(T_{IN} - T_{REF}\right)\sum N_{i,IN} Cp_i \\
&\quad - \left(T_{OUT} - T_{REF}\right) F_{OUT} \sum N_{i,OUT} Cp_i \\
&\quad + F_{OUT} \sum N_{i,OUT} \Delta H^{rxn}\left(T_{REF}\right) \\
&\quad - F_G \sum y_{i,G} \Delta H^{vap} \qquad (7.11)
\end{aligned}$$

The reaction rates, k, are non-linear in temperature, and appear in product terms with the number of moles in the reactor. Since a steady state operating region exists, it is possible to linearise around the operating temperature, T_0 and operating compositions, N_0.

$$\begin{aligned}
k(T) &= A \exp(-E_i/RT) \\
k(T)N &\approx \left(k(T_0) + (T - T_0)\frac{\partial k}{\partial T}|_{T_0}\right)(N_0 + \Delta N) \\
&\approx k(T_0)N + N_0(T - T_0)\frac{\partial k}{\partial T}|_{T_0} \qquad (7.12) \\
&= k(T_0)N + N_0(T - T_0)k(T_0)\frac{E_i}{RT_0^2} \\
&= k(T_0)N + N_0 k(T_0)\frac{E_i}{RT_0^2}T - N_0 k(T_0)\frac{E_i}{RT_0} \qquad (7.13)
\end{aligned}$$

Now construct this as a dynamic system

$$\frac{d}{dt}x = Ax + Bu + w$$

$$\frac{d}{dt}\begin{bmatrix} \left(\sum N_i C_{pi}\right)_R T \\ N_{XY} \\ N_{TA} \\ N_{CBA} \\ N_{TPA} \\ N_{H_2O} \\ N_{AA} \end{bmatrix} = A\begin{bmatrix} T \\ N_{XY} \\ N_{TA} \\ N_{CBA} \\ N_{TPA} \\ N_{H_2O} \\ N_{AA} \end{bmatrix} + \begin{bmatrix} -F_G \sum y_{i,G} \Delta H^{vap} \\ 0 \\ 0 \\ 0 \\ 0 \\ -F_G y_{H_2O} + F_C x_{H_2O,C} \\ -F_G y_{AA} + F_C x_{AA,C} \end{bmatrix} + \begin{bmatrix} \sum \Delta H_{IN,i} \\ N_{XY,IN} \\ N_{TA,IN} \\ 0 \\ 0 \\ 0 \\ N_{AA,IN} \end{bmatrix}$$

$$- \begin{bmatrix} T_{REF} F_{OUT} \sum N_{i,0} Cp_i \\ -N_{XY,0} k_1(T_0) \frac{E_1}{RT_0} \\ N_{XY,0} k_1(T_0) \frac{E_1}{RT_0} - N_{TA,0} k_2(T_0) \frac{E_2}{RT_0} \\ N_{TA,0} k_2(T_0) \frac{E_2}{RT_0} - N_{CBA,0} k_3(T_0) \frac{E_3}{RT_0} \\ N_{CBA,0} k_3(T_0) \frac{E_3}{RT_0} \\ N_{XY,0} k_1(T_0) \frac{E_1}{RT_0} + N_{TA,0} k_2(T_0) \frac{E_2}{RT_0} \\ 0 \end{bmatrix}$$

$$\Delta H_{IN,i} \equiv \left(T_{IN} - T_{REF}\right) N_{IN,i} Cp_i$$

$$\Delta H_{OUT,i} \equiv -\left(T_{OUT} - T_{REF}\right) F_{OUT} Cp_i$$

where

$$A = \begin{bmatrix} -F_{OUT}\sum N_{i,0}Cp_i & \Delta H_{OUT,XY} & \begin{array}{c}\Delta H_{OUT,TA}\\ +F_{OUT}\Delta H^{rxn}_{T_{REF}}\end{array} \\ -N_{XY,0}k_1(T_0)\frac{E_1}{RT_0^2} & -F_{OUT}-k_1(T_0) & 0 \\ N_{XY,0}k_1(T_0)\frac{E_1}{RT_0^2} - N_{TA,0}k_2(T_0)\frac{E_2}{RT_0^2} & k_1(T_0) & -F_{OUT}-k_2(T_0) \\ N_{TA,0}k_2(T_0)\frac{E_2}{RT_0^2} - N_{CBA,0}k_3(T_0)\frac{E_3}{RT_0^2} & 0 & k_2(T_0) \\ N_{CBA,0}k_3(T_0)\frac{E_3}{RT_0^2} & 0 & 0 \\ N_{XY,0}k_1(T_0)\frac{E_1}{RT_0^2} + N_{TA,0}k_2(T_0)\frac{E_2}{RT_0^2} & k_1(T_0) & k_2(T_0) \\ 0 & 0 & 0 \end{bmatrix} \cdots$$

$$\cdots \begin{bmatrix} \begin{array}{c}\Delta H_{OUT,CBA}\\ +F_{OUT}\Delta H^{rxn}_{T_{REF}}\end{array} & \begin{array}{c}\Delta H_{OUT,TPA}\\ +F_{OUT}\Delta H^{rxn}_{T_{REF}}\end{array} & \Delta H_{OUT,H_2O} & \Delta H_{OUT,AA} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -F_{OUT}-k_3(T_0) & 0 & 0 & 0 \\ k_3(T_0) & -F_{OUT} & 0 & 0 \\ 0 & 0 & -F_{OUT} & 0 \\ 0 & 0 & 0 & -F_{OUT} \end{bmatrix}$$

The next step is to deal with the non-linearity between temperature and composition in the energy balance. Again, we can linearise around our selected steady state to obtain the following linearised energy balance.

$$\begin{aligned} \frac{d}{dt}\left(\sum N_i C_{pi} T\right)_R &= \left(\sum N_i C_{pi}\right)_0 \frac{dT}{dt} + \left(\sum C_{pi} T_0 \frac{dN_i}{dt}\right)_R \\ \frac{dT}{dt} &= \left(\sum N_i C_{pi}\right)_0^{-1} \left(\frac{d}{dt}\left(\sum N_i C_{pi} T\right)_R - \left(\sum C_{pi} T_0 \frac{dN_i}{dt}\right)_R\right) \quad (7.14) \end{aligned}$$

All of the parameters on the right hand of the equation are obtained from the matrix A, and the first row of both A and the input vectors are replaced by linear combinations of the original entries, according to this linearised energy balance.

The input vectors are assumed to be fluctuating around a known steady state. The flowrates into the reactor are assumed constant, but in reality fluctuate according to decisions upstream. We specify a set point for the input flowrate of the xylene monomer and top-up acetic acid, consistent with our desired production rate, and the expected loss of acetic acid from the system through the waste and product streams. We are unable to control the flowrate from the separations unit. This stream consists largely of acetic acid, and is essentially equal to the amount of acetic acid leaving the reactor with come amount lost through the product and liquid and vapour waste streams. We can thus predict a steady state value for this stream.

The control inputs consist of the flowrates to and from the overhead condenser, the operation of which is defined by the temperature and pressure selected for the steady state. The pressure can be controlled directly, by adjusting the pressure of the entering gases, while the temperature can be adjusted by altering the flowrate of coolant through the condenser, and thus the temperature of the returned stream. We are thus able to control the flowrates through the condenser. The following section describes this in more detail, since the relationship between the quantities is non-linear.

7.4 The vapour-liquid equilibrium

Assume that the flowrate of the overhead gas is unknown, temperature of the liquid stream from the condenser is measured. Rotameters and thermocouples may be used for the flowrate and temperature, while the composition, which is expected to be a predominantly water and acetic acid could be measured using conductivity.

The physics of the evaporation, condensation, and described sequentially. The Antoine Equation provides vapour pressure in atmospheres as a function of temperature. This equation is valid in the reactor, where the operating point is $P = 30atm$, $T = 225°C$, and at the end point of the condenser. In the reactor:

$$\ln P_{AA}^{vap}(T) = A - \frac{B}{T+C} = 11.8359 - \frac{4457.86}{T(C) + 258.451} \tag{7.15}$$
$$= 2.61 \text{ at } 225^\circ C \tag{7.16}$$
$$P_{AA}^{vap}(T) = 13.6 atm \tag{7.17}$$
$$\ln P_{H_2O}^{vap}(T) = A - \frac{B}{T+C} = 11.9517 - \frac{3984.95}{T(C) + 233.426} \tag{7.18}$$
$$= 3.259 \text{ at } 225^\circ C \tag{7.19}$$
$$P_{H_2O}^{vap}(T) = 26.0 atm \tag{7.20}$$

The liquid mole fractions in the reactor are either obtained from the state estimator for previous points if these are available, or are set to the steady state under the current operating conditions. These will be periodically measured, and would be updated whenever a discrepancy was detected.

$$x_{AA} = 0.60 \tag{7.21}$$
$$x_{H_2O} = 0.25 \tag{7.22}$$

The Margules Equation is used to determine the activity coefficients of the volatile components.

$$\ln \gamma_{AA} = x_{H_2O}^2 \left[A_{AA/H_2O} + 2\left(A_{H_2O/AA} - A_{AA/H_2O}\right) x_{AA}\right] \tag{7.23}$$
$$= (0.25)^2 [0.9960 + 2(0.4552 - 0.9660)(0.6)] \tag{7.24}$$
$$= 0.02394 \tag{7.25}$$
$$\gamma_{AA} = \exp(0.02394) = 1.0242 \tag{7.26}$$
$$\ln \gamma_{H_2O} = x_{AA}^2 \left[A_{H_2O/AA} + 2\left(A_{AA/H_2O} - A_{H_2O/AA}\right) x_{H_2O}\right] \tag{7.27}$$
$$= (0.6)^2 [0.4552 + 2(0.9660 - 0.4552)(0.25)] \tag{7.28}$$
$$= 0.2558 \tag{7.29}$$
$$\gamma_{H_2O} = \exp(0.2558) = 1.2915 \tag{7.30}$$

Modified Raoult's Law gives the vapour mole fractions in equilibrium with the liquid mole fractions within the reactor. Since the overhead gas leaves to the condenser, this is assumed to be the composition of the gas stream that goes to the condenser.

$$\begin{aligned}
y_{AA} &= \frac{\gamma_{AA} x_{AA} P_{AA}^{vap}(T)}{P_{tot}} = \frac{(1.0242)(0.60)(13.6)}{30} && (7.31)\\
&= 0.279 && (7.32)\\
y_{H_2O} &= \frac{\gamma_{H_2O} x_{H_2O} P_{H_2O}^{vap}(T)}{P_{tot}} = \frac{(1.2915)(0.25)(26.0)}{30} && (7.33)\\
&= 0.280 && (7.34)
\end{aligned}$$

The components of this stream will be surplus oxygen, inert nitrogen and all of the components of the liquid phase, although the vapour pressures of most of these are considerably lower that of acetic acid and water, or in such small quantities in the liquid phase, that they can essentially be ignored in this calculation. The input of oxygen is selected to be in molar ratio of 5:1 with the incoming xylene, since air is the source, we can predict the flowrate of nitrogen into the system. We can estimate the amount of oxygen that reacts, and hence the output of the nitrogen and oxygen. Here is a sample calculation that would be changed if there was a change in the xylene flowrate.

$$\begin{aligned}
F_G &= F_{O_2} + F_{N_2} + F_{AA} + F_{H_2O} + F_{TA} + F_{XY} && (7.35)\\
&= \frac{F_{O_2} + F_{N_2}}{1 - (y_{AA} + y_{H_2O} + y_{TA} + y_{XY})} && (7.36)\\
&= \frac{(133 + 1290)\, kmol/h}{1 - (0.279 + 0.280 + small)}\\
&= 3226 kmol/h
\end{aligned}$$

We can compute the amount of AA and H_2O lost from the top of the condenser by computing the vapour liquid equilibrium between oxygen, nitrogen and the acetic acid and water. Assume most of the acetic acid and water condenses to $50°C$, the liquid fractions are in roughly the same mole fractions as in the entering stream to the condenser, which is approximately 50%.

$$\begin{aligned}
\ln P_{AA}^{vap}(50) &= 11.8359 - \frac{4457.86}{50+258.451} && (7.37)\\
P_{AA}^{vap}(50) &= 0.073 atm && (7.38)\\
\ln P_{H_2O}^{vap}(50) &= 11.9517 - \frac{3984.95}{50+233.426} && (7.39)\\
P_{H_2O}^{vap}(T) &= 0.122 atm && (7.40)
\end{aligned}$$

$$\begin{aligned}
\ln\gamma_{AA} &= x_{H_2O}^2\left[A_{AA/H_2O} + 2\left(A_{H_2O/AA} - A_{AA/H_2O}\right)x_{AA}\right] && (7.41)\\
&= (0.5)^2\left[0.9960 + 2(0.4552 - 0.9660)(0.5)\right] && (7.42)\\
&= 0.1213 && (7.43)\\
\gamma_{AA} &= \exp(0.1213) = 1.129 && (7.44)\\
\ln\gamma_{H_2O} &= x_{AA}^2\left[A_{H_2O/AA} + 2\left(A_{AA/H_2O} - A_{H_2O/AA}\right)x_{H_2O}\right] && (7.45)\\
&= (0.5)^2\left[0.4552 + 2(0.9660 - 0.4552)(0.5)\right] && (7.46)\\
&= 0.2415 && (7.47)\\
\gamma_{H_2O} &= \exp(0.2415) = 1.273 && (7.48)
\end{aligned}$$

$$\begin{aligned}
y_{AA} &= \frac{\gamma_{AA} x_{AA} P_{AA}^{vap}(T)}{P_{tot}} = \frac{(1.129)(0.50)(0.073)}{30} && (7.49)\\
&= 0.0014 && (7.50)\\
y_{H_2O} &= \frac{\gamma_{H_2O} x_{H_2O} P_{H_2O}^{vap}(T)}{P_{tot}} = \frac{(1.273)(0.5)(0.122)}{30} && (7.51)\\
&= 0.0026 && (7.52)
\end{aligned}$$

$$\begin{aligned}
F_{EX} &= F_{O_2} + F_{N_2} + F_{AA} + F_{H_2O} && (7.53)\\
&= \frac{F_{O_2} + F_{N_2}}{1 - (y_{AA} + y_{H_2O})} && (7.54)\\
&= \frac{(133 + 1290)\, kmol/h}{1 - 0.004}\\
&= 1429 kmol/h
\end{aligned}$$

of which $6.56 kmol/h$ is AA and H_2O, in a molar ratio of roughly 1:2.

This gives us enough information to compute the input vector, since $-F_G y_{H_2O} + F_C x_{H_2O,C} \approx 4.3 kmol/h$ and $-F_G y_{AA} + F_C x_{AA,C} \approx 2.3 kmol/h$.

$-F_G \sum y_{i,G} \Delta H^{vap}$ comes from the above values, and the values of ΔH^{vap} in the tables of physical data.

7.5 The case study

An implementation of the above plant has been coded and run using MATLAB. A simulator for the plant was constructed using the model equations given above. Initially, a set of operating parameters was selected - a fixed flowrate of the monomer, a desired operating temperature and pressure, and a functional steady state was selected. The plant was simulated at steady state to produce a set of molar compositions within the reactor, specific to this set of operating conditions.

The simulator was used to generate a set of plant data under non-steady state conditions. The state data - temperatures and number of moles within the reactor were generated using the "true" plant, with actual reactor temperature, and actual compositions used for the transition equations. This state data was measured according to a selected model, and recorded. The case study has been built with a number of flexible parameters. It was assumed that the input flowrate of the monomer was varying, uncontrolled and measured, that the input temperature was varying, uncontrolled and measured, and that the exit temperature from the condenser is variable - consistent with the physical reality that cooling water has a fluctuating temperature.

The measured data were stored and sent to the multiscale state estimator to recover the underlying state variables from the measurements.

The estimation of the plant from measurement data suffers from the fact that the model is inherently non-linear - and linearised around the selected operating point.

The original intention was to recover the concentration profile from a measurement of temperature only. The problem with this is the observability of the system - the observability

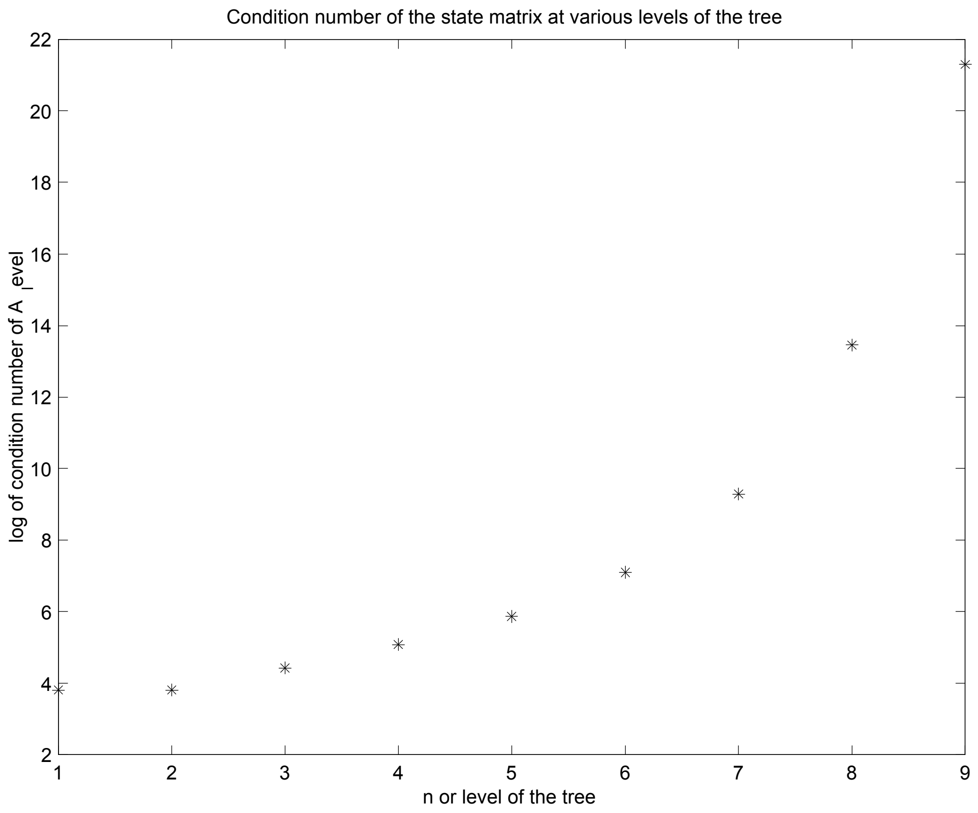

Figure 7-2: The condition number of the TPA state matrix

matrix $O = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix}$ has rank of five instead of the seven that is required for complete observability. It is clear then that we need some concentration data. A reasonable approximation can be made by assuming that the concentrations are close to their steady state values, and performing state observation using these "measurements" to perform the state estimation.

There is a problem with dealing with large systems of data in multidimensional systems. Since the matrix A is inverted high up on the tree, we are concerned about the condition number of this matrix. It turns out that the condition number roughly squares with each level, as illustrated in the plot of condition number with level. this provides us with a natural upper limit on the number of points we wish to use for the state estimation.

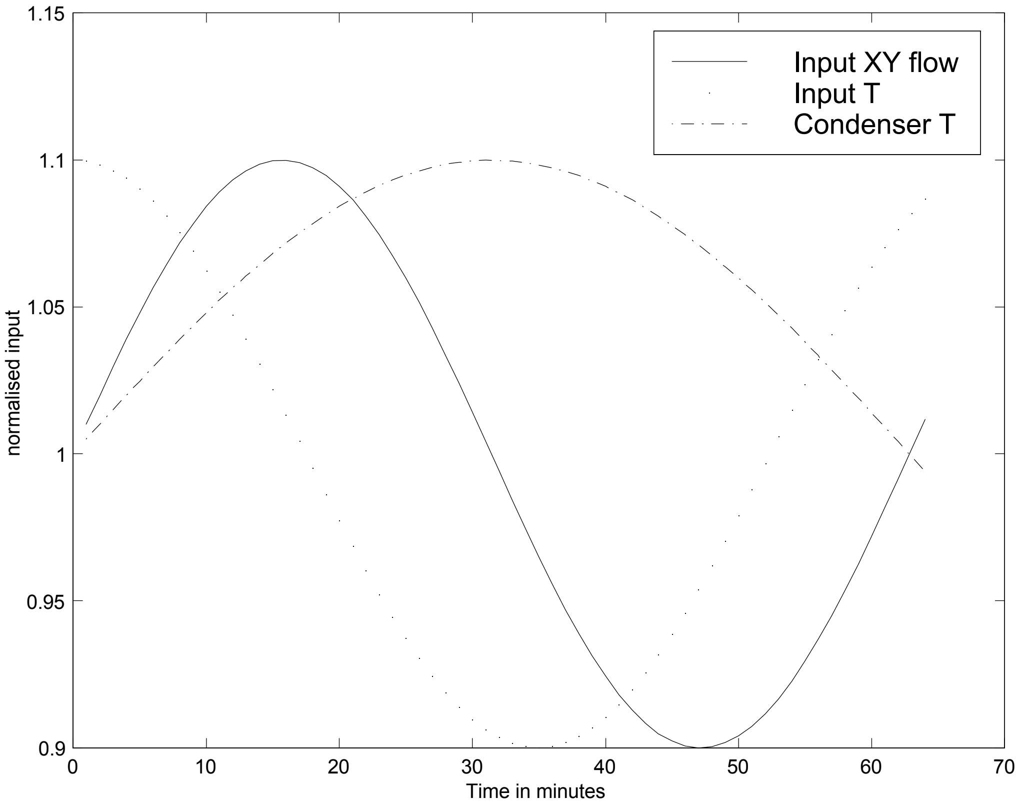

Figure 7-3: Input variables for TPA plant

The plots from the case studies illustrate a number of features of the state estimator, and how we have dealt with the difficulties confronting us. The first plot shows the input variables used. These are plotted as normalised variables around their steady states. The amount of xylene flowing into the reactor was assumed to be fluctuating in a measurable way, and this has been modeled with the profile below. The temperature of the input stream depends on the upstream processes and the length of time it has been stored, so it is assumed to be varying in a measurable way. The condenser temperature is dependent on the control system used, the temperature of the coolant, and is assumed to be measurable and varying as shown in Figure 7-3.

A base case for comparison of the other plots is illustrated in Figure 7-4. In all cases, we are plotting the state variables normalised by their steady state values, and then plotted on the same scale as temperature, which is set around 485K. Temperature fluctuates around 485K, and is bundled together with the acetic acid, TPA and water reactor masses. The mass of toluic acid rises initially, while of the lower curves, xylene is closer to 500 than the concentration of CBA. In all of the plots, the true state value from the simulation is represented by a a solid line for each species, while the measurements are represented by crosses, and the state estimates by dots.

We have performed a simulation, and state estimation using noise free measurements. Initially, there were problems due to the free estimate of the left most node in time. These have not been illustrated, but essentially the initial point was very large compared to all other points, suggesting that most of the energy in the cost function was concentrated in the initial point. The reason for this is that the left node is not dependent on any inputs, and thus has less effect on the surrounding points than any other nodes on the tree. Hence, the entire left-most branch of the tree will be unconstrained by inputs from the left, and consequently the leftmost point is free to float. This can be resolved by applying a prior estimate to the left-most node of the tree, using the technique discussed at length in Chapter 2.

The plot in Figure 7-5 shows a more realistic case study. Here we have used the same simulation data as for the case study without noise, but assumed that all measurements contain measurement noise. This data is used to produce the state estimates, and plotted here as described above for Figure 7-4.

The case study in Figure 7-6 illustrates a different scenario. Here, the simulation proceeds as for the previous case studies, but the estimation only has access to some of the concentration data. In this case study, the remaining data is assumed to be at the steady state values. The estimator performs less well since the measurements are inconsistent with the state model. As would be expected, the measured states are estimated better than the unmeasured states, but with three concentrations measured, and the remaining three estimated, it is possible to produce a full set of state estimates.

7.6 Summary

The terephthalic acid plant provides a chemical engineering example of a process that can be analysed using the multiscale state estimator. The original system of equations is non-linear and

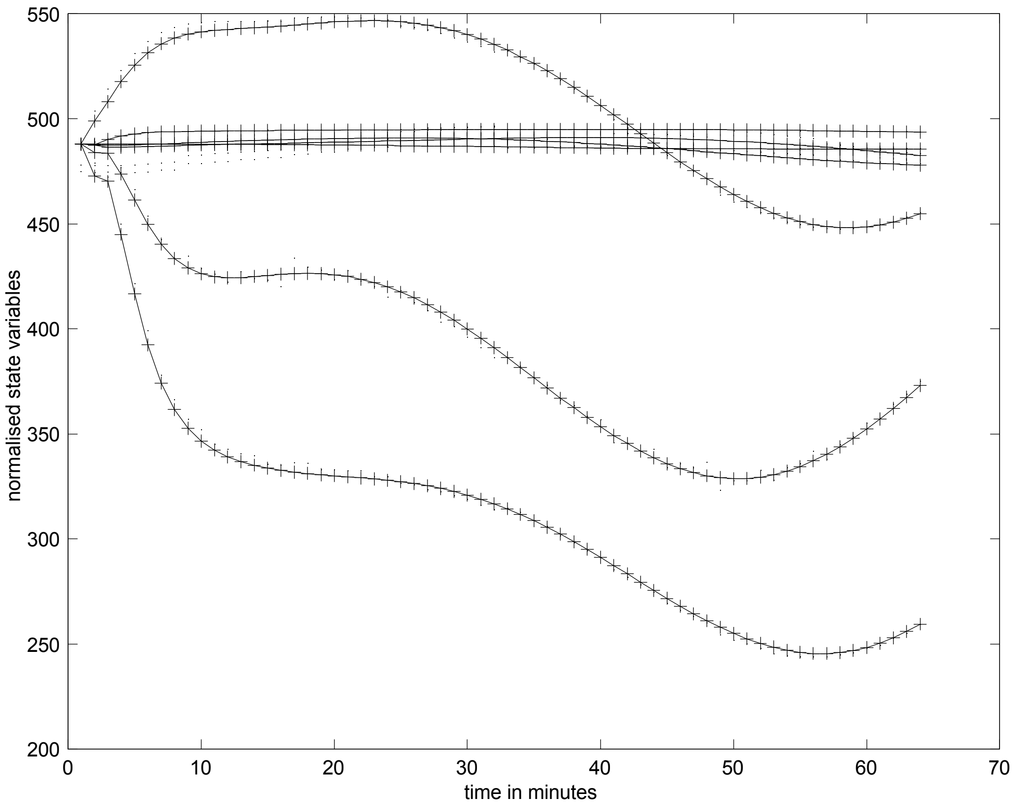

Figure 7-4: State Estimation with Noise-free Measurements

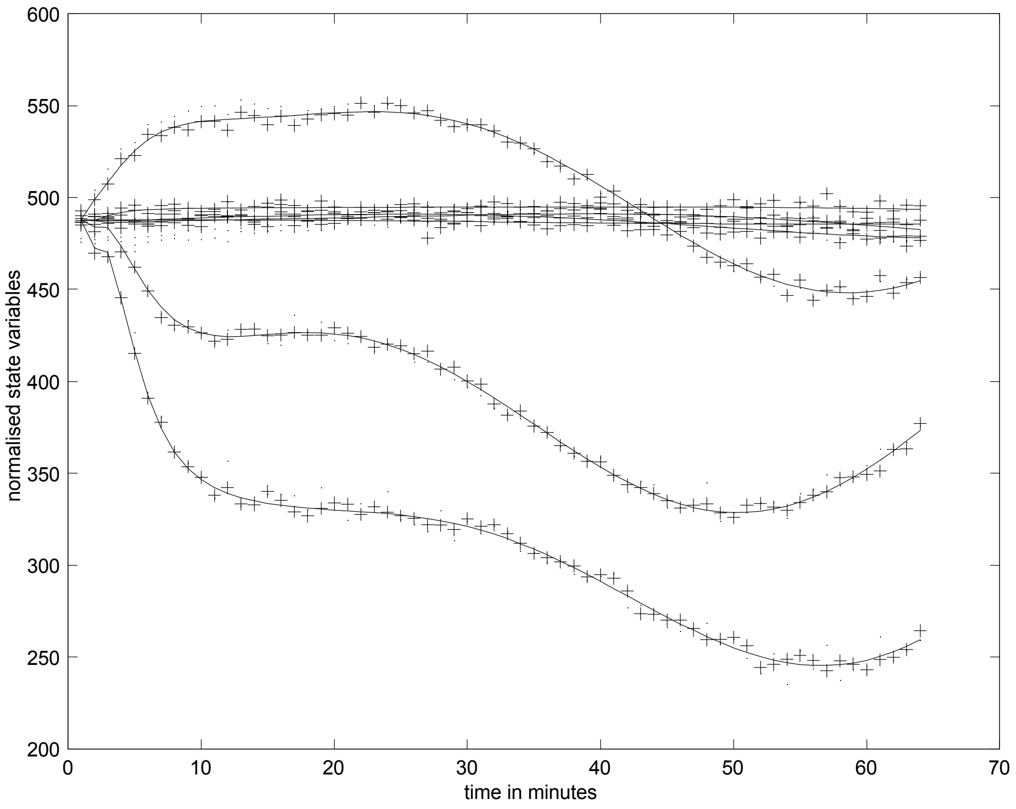

Figure 7-5: State Estimation using Complete Sets of Measurements

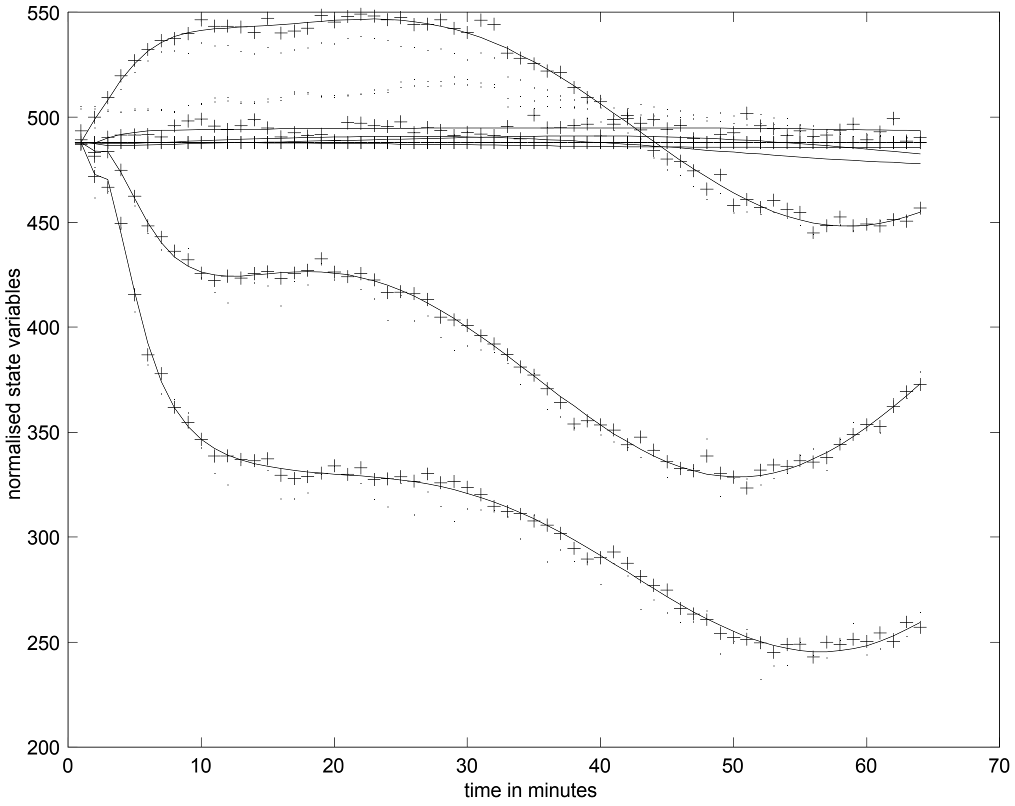

Figure 7-6: State Estimation using Steady States as Estimates

has been linearised successfully around a given steady state, which is used as the plant model. In the absence of complete information, the specified steady state can be used as a measurement subject to a large estimate of measurement error. In this case a good prior estimate is useful, since the large measurement errors cause spurious values at the first point, as discussed in chapter four.

Bibliography

[1] Huang, S., Kim S. Munro J. *Preliminary Evaluation of the Manufacture of Terephthalic Acid*. Technical Report, Massachusetts Institute of Technology, 1998.

Chapter 8

Conclusions and Suggestions for Future Work

This chapter discusses the contributions of this thesis and suggests directions for future work.

The state of knowledge at the outset of this project was an absence of multiscale models with a physical basis. The field of wavelet theory had developed to deal effectively with image compression, and signal decomposition, but there was no attempt to describe models at the different levels of representation. Within this context, the original direction of the work was to develop a meaningful and useful concept of a multiscale model, and its relation to a physical model, to explore ways in which this could be used within estimation, optimisation and control problems, and finally to explore physical and chemical applications where these tools may be useful.

8.1 Thesis contributions

There are two main aspects in which the multiscale state estimator differs qualitatively from the time based state estimator. The first is the shape of the cost space. The introduction of multiscale models produces a cost function with a different minimum to the standard two norm minimiser in time. This suggests that the error structure is somewhat arbitrary, yet in the absence of perfect state information, it is impossible to say which estimator produces the better estimate - it is a function of the accuracy of the error models used to generate the two state estimators. It follows that a system for which multiscale knowledge exists should benefit from the richer, multiscale description of error used in the multiscale state estimator.

The thesis presents the development of multiscale models in an efficient way for use in state estimation algorithms. Multiscale models are developed by applying the Haar transform to polynomial order autoregressive systems in time. The modified hat transform is the natural consequence of this approach, and is the correct data structure for representing inputs and modelling uncertainty on the multiscale tree. Various representations of multiscale model are presented - those that describe relationships between states at the same level, those describing the relationship between states at adjacent levels, and those describing the relationship between wavelet and scaling functions of the same quantity. These representations are summarised in a compact form.

We have successfully produced a state estimator that follows the spirit of the Kalman filter and Rauch-Tung-Striebel smoother. The matrix equations obtained are highly structured due to the nature of the wavelet transform, and the localised relationship between nodes and their subtrees. This is allows the development of a sequential algorithm that makes maximum use of the sparse structure of the matrix equations. The algorithm has been developed to include varying levels of complexity. The algorithm has the feature that the model part is separate from the measurement part, meaning that the model part is reusable for different sets of measurements. Similarly, prior state estimates can be included using a separable set of coefficients. The relationship between the Kalman filter and the Rauch-Tung-Striebel estimator is demonstrated by deriving the Kalman filter equations using the same procedure as for the multiscale state estimation algorithm.

We have successfully demonstrated the extension of this algorithm to include linear constraints on the states, and on linear combinations of the states - thus estimates in the measurement and modelling errors. The algorithm is also designed so that the constraint related coefficients are separable from the model and measurement coefficients. The constrained multiscale state estimator is an iterative parallelisable algorithm that works well when the solution contains large contiguous areas of active constraints that can be lumped together, thereby reducing computational complexity considerably. Like its counterpart in the time domain, there are problems for which a combinatorial search will be necessary, and the multiscale problem will typically inherit this property from the time domain problem.

The multiscale estimators have been derived to deal with an arbitrary number of states, although algorithmic speed will scale with the number of states, an arbitrary number of measurements, and measurements sets, although only the startup cost of computation will scale with the number of measurement sets. The ability to use multiple measurement sets implies that multirate input and state data can be fused in an optimal and transparent way. The simple version of the algorithm is for first order state and input dynamics, although polynomial order multiscale systems can be dealt with easily.

The state estimation algorithms are computationally efficient, and parallelisable, scaling as O(log(N)) for N points, compared to O(N) for time based algorithms. While a parallel implementation of the code has not been written, it will clearly be faster than the current single processor code, and will have fewer serial operations than a time based estimator. They are thus suited to real-time implementations. For systems where a righer description of the underlying error processes is available, this can be incorporated easily, and more importantly, knowledge about both frequency and temporal aspects of the uncertainty can be incorporated using the uncertainty parameters at the various levels of the tree.

The multiscale models have been analysed for the propagation of error up and down the tree. This is performed completely for the normal distribution, and then extended using the polynomial chaos expansion to include systems with non-normal error distributions. One needs to provide the algorithm with estimates of the moments to arbitrary order, but four works well, and one obtains estimates for the error in the state estimates, and the moments of this error to the number specified *a priori*. Specifically, this approach provides a considerable improvement to systems where the infinite tail assumption of the normal distribution is problematic.

These concepts have been demonstrated using a collection of case studies - numerical experiments to demonstrate data fusion, the use of higher order dynamic systems, and the comparison with the Kalman filter; and chemically based examples, the paper rolling case study, the terephthalic acid reactor, and finally the heavy oil fractionator. The final case study demonstrates the link between the multiscale state estimator and the corresponding multiscale model predictive control algorithm, which forms the intellectual other half to this work, and is discussed at length in the soon-to-be-published thesis of Orhan Karsligil.

Theoretically, the algorithm inherits its stability and solvability aspects from quadratic programming and linear algebra theories. It is possible to construct systems for which matrices will be singular, and for which no minima will exist. A quadratic programming problem which has no solution in the time domain will not have one in the wavelet domain, and vice versa, since the time domain and the wavelet domain are linked by an orthonormal transform.

In conclusion, multiscale state estimation is most suitable for very large estimation problems, in particular those for which estimates are required at a level of resolution coarser than that at which measurements are taken. If access to a parallel machine exists, then considerable reductions in the time to state estimation will be achieved.

8.2 Suggestions for future work

The primary thrust of future work should be to search widely and thoroughly for physical, chemical, environmental or other processes that contain multiscale features that could be used naturally in a wavelet based algorithm. While a collection of case studies has been collected for this work, the multiscale element has been in the algorithmic treatment, rather than in the underlying process. Wavelet techniques have been successful for compression based technologies, where a reduction of computation follows from the elimination of redundant data.

A parameter estimation algorithm specific to the relevant model should be investigated using the tree framework. Simultaneous state and parameter estimation is necessarily a non-linear problem, and thus the multiscale tree may provide a computational advantage by eliminating repeated calculations on certain sections of the tree that are not changing. The problem that is likely is that the non-linear nature of the problem convolves the frequency components, meaning that no section of the tree is likely to settle faster than any other. Nevertheless, there may be certain non-linear systems may be more conducive to tree-based approaches than others.

The parallel nature of the code has never been demonstrated, due to the lack of access to a parallel processor. This could be demonstrated using a suitably interesting problem.

The algorithm has been developed for the Haar wavelet, since it is the most computationally efficient, having the smallest support. There could be improvements by adapting the algorithm for other special types of wavelets, where there are recognisable structures that are better modelled using a different wavelet. The Poisson wavelet is an example of such a wavelet, derived for, and extremely good at modelling systems with exponential decay. Essentially the final exponential decay is reduced to a single wavelet, while the Haar domain requires truncation and approximation, and a large number of terms.

Appendix A

Supporting MATLAB Code

The appendix contains code for the various tasks.

A.1 Utility Programs

A.1.1 wt.m

```
% wt receives a vector of arbitrary length, pads it with zeros and returns the
% wavelet transform, packed with scaling function at the top,
% and wavelets after.

% Modified for multidimensional vectors

function w = wt(y)

flag = 0;
if(size(y,1)<size(y,2))
    disp('Problem with geometry of matrix in wavelet transform, make it a ...
             column vector')
    y = y';
    flag = 1;
end
```

```
yorig = y;
lly = log(length(y))/log(2);
% pad with zeros
if floor(lly)~=lly
   lly = ceil(lly);
   lyo = length(y);
   ly = 2^lly;
   y(ly,:) = 0;
   y((lyo+1):ly,:) = 0*y(1:ly-lyo,:);
end

% y = y(:);
waveletfilter = -[1/sqrt(2) -1/sqrt(2)];
scalingfilter = [1/sqrt(2) +1/sqrt(2)];

w=[];
for counter2 = 1:size(y,2)
  W = [];
  y = y(:,counter2);
  for counter1 = 1:lly
    waveletdecomposition = filter(waveletfilter,1,y);
 waveletdecomposition = waveletdecomposition(2:2:length(waveletdecomposition));
    scalingfunction = filter(scalingfilter,1,y);
    scalingfunction = scalingfunction(2:2:length(scalingfunction));

    W = [ waveletdecomposition; W];
    y = scalingfunction;
  end;
  w = [w [scalingfunction; W]];
  y = yorig;
end;

if flag==1
  w = w';
end;

flag = 0;
```

A.1.2 mwt.m

```
% wt receives a vector of arbitrary length, pads it with zeros and
% returns the modified wavelet transform, packed with scaling function
% at the top, and wavelets after.
% Matrix A is used for the transform.

% Modified for multidimensional vectors

function w = mwt(y,A)

r2 = sqrt(2);

flag = 0;

if(size(y,1)<size(y,2))
    disp('Problem with geometry of matrix in MWT wavelet transform, make it a
column  vector')
    y = y';
    flag = 1;
end
```

```
yorig = y;
lly = log(length(y))/log(2);
% pad with zeros
if floor(lly)~=lly
   y(length(y)+1:exp(ceil(lly))) = 0*(1:exp(ceil(lly))-length(y));
end

% y = y(:);
Aorig = A;
scalingfilter = 1/sqrt(2)*[eye(size(A)) eye(size(A))+A A];

z = zeros(1,size(y,2));
w=[];
W = [];
leftnodes = y(1:2:size((y),1),:);

for counter1 = 1:lly
    scalingfunction=[y;z;z]*eye(size(A))'+[z;y;z]*(eye(size(A))+A)'+[z;z;y]*A';
    scalingfunction = 1/r2*scalingfunction(3:2:size((scalingfunction),1),:);
    A = A*A;
    W = [scalingfunction;W];
    y = scalingfunction;
    leftnodes = [scalingfunction(1:2:size((scalingfunction),1),:); leftnodes];
end;

A = Aorig;
w = [w; leftnodes];
y = yorig;

if flag==1
```

```
  w = w';
end;
flag = 0;
```

A.1.3 iwt.m

```
% iwt is the inverse wavelet transform for a multidimensional vector,
% and replaces the original 1-D iwt.m
% iwt receives a vector of arbitrary length,  packed with scaling function at
% the top, and wavelets sorted by scale and  returns the inverse Haar wavelet
%  transform

function w = iwt(y)

flag = 0;
if(size(y,1)<size(y,2))
  disp('Problem with geometry of matrix in inverse wavelet transform, make ...
            it a column vector')
  y = y';
  flag =1;
end

r2 = sqrt(2);
yorig = y;
lly = log(length(y))/log(2);
% pad with zeros
if floor(lly)~=lly
   y(length(y)+1:exp(ceil(lly))) = 0*(1:exp(ceil(lly))-length(y));
end

% y = y(:);
```

```
w = [];
for counter2 = 1:size(yorig,2)
  y = y(:,counter2);
  sf = y(1);
  a = sf;
  wf = y(2);
  y = y(2:length(y));

  while(length(y)>0);
    la = length(a);
    wf = y(1:la);
    y = y(la+1:length(y));
    a(1:2:2*la) = 1/r2*(sf+wf);
    a(2:2:2*la) = 1/r2*(sf-wf);
    sf = a';
  end;
  w = [w sf];
  y = yorig;
counter2;
end;

if flag==1
   w = w';
end;
```

A.1.4 sf.m

```
% sf receives a vector of arbitrary length, pads it with zeros and returns the
% wavelet transform, packed with scaling function coefficients by scale.

% Modified for multidimensional vectors
% Returns scaling functions

function s = sf(y)

flag = 0;

if(size(y,1)<size(y,2))
    disp('Problem with geometry of matrix in wavelet transform, make it a ...
           column vector')
    y = y';
    flag = 1;
end

yorig = y;
lly = log(length(y))/log(2);
% pad with zeros
if floor(lly)~=lly
   y(length(y)+1:exp(ceil(lly))) = 0*(1:exp(ceil(lly))-length(y));
end

% y = y(:);
waveletfilter = -[1/sqrt(2) -1/sqrt(2)];
scalingfilter = [1/sqrt(2) +1/sqrt(2)];

w=[];
s = [];
for counter2 = 1:size(y,2)
  S = [];
  y = y(:,counter2);
  for counter1 = 1:lly
```

```
    waveletdecomposition = filter(waveletfilter,1,y);
  waveletdecomposition = waveletdecomposition(2:2:length(waveletdecomposition));
    scalingfunction = filter(scalingfilter,1,y);
    scalingfunction = scalingfunction(2:2:length(scalingfunction));

    S = [scalingfunction; S];
    y = scalingfunction;
  end;
  s = [s [scalingfunction; S]];
  S = [ scalingfunction; S];
  y = yorig;
end;

if flag==1
  s = s';
end;
```

A.1.5 mkhaar.m

```
% mkhaar.m returns a Haar matrix, H,  of specified size, so that w=Hx returns
% wavelet transform of vector x.

function [mk] = mkhaar(n);

H = zeros(n,n);

numlevels = log(n)/log(2);
temp = 1;
level = n;
for j = numlevels:-1:1
   temp = 1/sqrt(2)*[abs(temp) -abs(temp)];
   for k = n-length(temp)+1:-length(temp):1
       H(level,k:k+length(temp)-1) = temp;
       level = level-1;
   end;
end;
H(1, :) = abs(temp);
mk = H;
```

A.1.6 mkmhat.m

```
% mkhaar returns the modified hat transform matrix M, so that w=Mx returns
%\ the modifed hat transform of vector x.

 function [mk] = mkmhat(n,A);

r2 = sqrt(2);
H = eye(n);
H = [H; 0*H];
bigG = H(1:2:n,:);
numlevels = log(n)/log(2);
temp = 1;
nl = n;
for j = 1:numlevels-1
   G = 1/r2*(A*H(1:2:nl,:) + (1+A)*H(2:2:nl+1,:) + H(3:2:nl+2,:));
   H = [ G; 0*G];
   G = G(1:2:nl/2,:);
   bigG = [G; bigG];
   nl = nl/2;
   A = A*A;
end;
```

```
mk = bigG;
```

A.1.7 parfind.m

```
% parfind  find the parents of a set of nodes


function pall = parfind(actives,pN)
j = 1;
pall = [];
for k = 1:length(actives);
 p = actives(k);
 ptest = 6;
 while((ptest>1)&(j<1000))
   ptest = pN(p(1));
   p = [ptest; p];
   j = j+1;
 end;
 pall = [pall; p(1:length(p)-1)];
end;
if(j>998)
  disp('loop in parfind terminated prermaturely')
end


pall = sort(pall);
pall = deldups(pall);  % deletes the duplicates
```

A.1.8 parsign.m

```
% Produces a sign matrix - + for left nodes, - for rights
% This is used when constraint violations are identified
```

```
function parsign=parsign(p,pN,rN)


pp = parfind(p,pN);

prn = rN(pp);

ppones = diag(ones(length(pp)));

pch = [pp(2:length(pp)); p];

ii = find(prn==pch');

ppones(ii) = -1*diag(ones(length(ii)));


parsign = ppones;
```

A.1.9 subtree.m

```
% subtree(n)

% This returns the nodes on the subtree of node n


function s = subtree(n,lN,rN)


if(n>0.5*rN(length(rN)))   % This checks for zeroth level nodes

   s = [n];

else

   s = [n];

   k = 1;

   steam = 1;   % Escape valve

   while (k<99999&steam<1000)

     t = [lN(s(k)) rN(s(k))];

     s = [s t];

     if(t(1)>0.5*rN(length(rN)))   % Note this is changed to be consistent with

                    % new lN, rN vectors that fill with zeroth.

       k = 99999;

     else
```

```
        k = k+1;
      end;
    steam = steam +1;
    end; % while
    s = s(1:length(s)-2);  % And we don't want zeroth level nodes here.
end; % if
```

A.2 The unconstrained multiscale state estimator

A.2.1 motmmain.m

```
% Calculates unconstrained multiscale state estimation
% Main driving program for motmseries

% This is the setup program that defines the model, and generates the simulated
% data.
motmsp;

% This converts the model parameters into parameters relevant for use on the
% tree.
motmparm;

% This performs the up sweep.
motmup;

% This performs the down sweep.
motmdown;
```

A.2.2 motmsp.m

```
% A sample program to setup the model and parameters.
% Multiple measurements and constraints
% B=1 throughout - other B's must be incorporated into U's.
% Define problem parameters
% This version is for multiple states

r2 = sqrt(2);
numlevels = 5;          % number of levels
N = 2^numlevels;        % number of points
   % Arrange all vectors {xtop, dx_top, dx_topa ...}
   %    store things for bookkeeping purposes

xdim = 1;  % state size
ydim = 1;  % measurement size

A = 0.95;  % A at the zeroth level - a few to choose from.
%A = [0.95 -0.4 -0.1 -0.05; 1 0 0 0; 0 1 0 0; 0 0 1 0];
% A = [0.95 0.1 0 0; 0 0.95 0 0 ; 0 0 0.95 0; 0 0 -0.1 0.95];
% A = [0.95 0 ; 0 0.95 ];

B = eye(xdim);          % Input dynamics
% B = diag([1 0.1 0.01 0.001]);

Q = 5*eye(xdim);                  % Plant input covariance (assumed)
U = randn(N,xdim)*sqrt(Q);        % Unmeasured plant inputs for simulation
% U(:,2:4) = 0*U(:,2:4);           For non first order systems

% Measured inputs specified here.
BU = ones(N,xdim)*diag([10 ]);
BU(10:15,1) = (10:5/5:15)';
BU(16:20,1) = ones(5,1)*15;
BU(21:26,1) = (-3:-2/5:-5)';
BU(27:32,1) = ones(6,1)*(-5);



C = eye(xdim);   % Measurement Model
% C = [1 0 0 0]; % 0 0 0 0; 0 0 0 0; 0 0 0 0];
R1 = 5*eye(ydim);
% Measurement covariance for first set (assumed)
R1inv=inv(R1);
R2 = 10^89*eye(ydim);  % Measurement covariance for second set.
R2inv=inv(R2);

Qinv = inv(Q);

% Constraints

xubar = 100*[1;1];
xlbar = -100*[1;1];
% Upper and lower constraints at the zeroth level

Atemp = A;   % Construct tree dynamic system, A first.
for counter2 = 1:numlevels % Matrices stored as vectors
  Alevels(counter2,:) = reshape(Atemp',1,xdim*xdim);
  Atemp = Atemp*Atemp;
end;
    % Construct higher level constraints
xubar(1:numlevels) = r2.^(0:numlevels-1)*xubar(1);
xlbar(1:numlevels) = r2.^(0:numlevels-1)*xlbar(1);
```

```
% Getting the multiscale Q right takes a bit of effort
% through a recursion

QRM = [];
QRM(1,:) = reshape(Q,1,xdim*xdim);
QDM(1,:) = 0* QRM;
QRold = Q;
QDold = 0*Q;
for counter = 2:numlevels+1
   QA = reshape(Alevels(counter-1,:),xdim,xdim)';
   QR = 0.5*(QA*QRold*QA'+(eye(xdim)+QA)*QRold*(eye(xdim)+QA)'+QRold + ...
      QA*QDold*(eye(xdim)+QA)' + (eye(xdim)+QA)*QDold + ...
      (eye(xdim)+QA)*QDold'*QA' + QDold'*(eye(xdim)+QA)');
   QR = 0.5*(QR+QR');
   QD = 0.5*QRold*QA' + 0.5*((QA+eye(xdim))*QDold*QA'+QDold*(QA+eye(xdim))');
   QRM(counter,:) = reshape(QR,1,xdim*xdim);
   QDM(counter,:) = reshape(QD,1,xdim*xdim);
   QRold = QR;
   QDold = QD;
end;


% Simulate data
[y,x] = dlsim(A,B,C,0*C,BU+U);
% Discrete simulation, measurements y, states x

% Note these assume a diagonal structure for R1, R2 since off diags are not
% stored.
% This could be changed if necessary by coding an alternative noise structure.
```

```
V1 = randn(N,ydim)*sqrt(R1);    % Construct zeroth level measurement noise
% V1 = V1 + [1.5*(1:N)' , -0.5*(1:N)'];   adding sensor drift for 2-D
wV1 = wt(V1);   % Take wavelet transform of measurements
Rm1 = ones(size(V1))*R1; % Setting uncertainties for measurements
  % Note this is where you would insert varying measurement uncertainty
  % accross the tree.

V2 = randn(N/16,ydim)*sqrt(R2); % Measurement noise for second set of
                                % measurements
wV2 = wt(V2);   % For this set, I introduce a fake uncertainty for
wV2(N,:) = zeros(1,ydim); % points with no measurement.
Rm2real = ones(size(V2))*R2;
Rm2 = ones(size(V1))*10^89;
Rm2(1:N/16,:) = Rm2real; % [Rm2 ; 10^89*Rm2 ; 10^89*Rm2; 10^89*Rm2];
% Replace this with R2factor for verstility - R2factor*R2inv
R2factor = 0*wV2;
R2factor(1:N/16,:) = ones(N/16,ydim);


xw = wt(x);             % wavelet transform of true state data
yw1 = wt(y) +wV1; % and the measurements
                          % wt returns the wavelet transform of the functions
yw2 = R2factor.*(wt(y) +wV2);
Uw = mwt(BU,A);           % modified hat transform of control inputs

lN = [3,3,5:2:N-1, (N+1)*ones(1,N/2), N+1];
   % This vector contains the left descendant node of each
rN = [4,4,6:2:N, (N+1)*ones(1,N/2), N+1];     % likewise right descendant
pN = [(2:N) ; (2:N)];    % and the parent nodes
pN = [0; 1; pN(:)];
```

```
% This section sets up parameters following the notation of thesis
% motmsetup defines the parameters and does some preliminary calcs
%  Derivation of Recursion System for Multiscale Optimisation on the
%  Tree for Measurements at Multiple Scales by Matthew Dyer, LISPE, MIT, 1998
% Note these are operations on the 3-D matrices a and Qtree, performed in 2-D
% This leads to some reshaping of vectors.

% The goal here is to produce a unique A, Q and upper and lower constraint
% at each point.  For this example, they are constant across a level, but
% this is where they would be changed if they varied in time.

% Start with the top level, scaling function, then wavelet coefficient.

a = [Alevels(numlevels,:) ; Alevels(numlevels,:)];
Qtree = [QRM(numlevels,:) ; QRM(numlevels,:)];
Xubar = [xubar(numlevels) ; xubar(numlevels)];
Xlbar = [xlbar(numlevels) ; xlbar(numlevels)];

% And continue down the tree, adding to the matrix from below.

for counter = numlevels-1:-1:1
   a = [a; ones(size(a,1),1)*Alevels(counter,:)];
   Qtree = [Qtree; ones(size(Qtree,1),1)*QRM(counter,:)];
   Xubar = [Xubar; xubar(counter)*ones(length(Xubar),1)];
   Xlbar = [Xlbar; xlbar(counter)*ones(length(Xlbar),1)];
end;

% Finally a layer for the zeroth level is included.
```

```
xubar(counter) = r2*xubar(counter);
xlbar(counter) = r2*xlbar(counter);
Xubar = [Xubar; xubar(counter)*ones(length(Xubar),1)];
Xlbar = [Xlbar; xlbar(counter)*ones(length(Xlbar),1)];
a(size(a,1)+1,:) = zeros(1,xdim*xdim);
Qtree(size(Qtree,1)+1,:) = zeros(1,xdim*xdim);

% Taking transform for data consistency.

Xubar = Xubar';
Xlbar = Xlbar';
```

A.2.3 motmparm.m

```
% This program constructs further parameters in preparation for the upsweep.
% motmparm

% First we construct an a and aprime at each node on the tree.

aprime = -a;

for counter = 1:xdim
   a(1:size(a,1)-1,xdim*(counter-1)+counter) = ...
                              a(1:size(a,1)-1,xdim*(counter-1)+counter)+1;
   aprime(1:size(a,1)-1,xdim*(counter-1)+counter) = ...
1+aprime(1:size(a,1)-1,xdim*(counter-1)+counter);
end;

b = 0*a; c = 0*a;

% Find modified hat transform of the known inputs
```

```
Uw = mwt(BU,A);

% The contribution of the known inputs is stored in DU, the contribution
% of u to d.
DU(N+1,:) = zeros(1,xdim);

% DU requires u-stars as intermediate variables
Ustar(N+1,:) = zeros(1,xdim);

% b and c are constructed at each level, and then stored as vectors.
% They are model parameters, independent of measurements.

for counter = N:-1:1        % Must go up the tree for recursion
  ib = 1/r2*inv(B')*inv(reshape(Qtree(counter,:),xdim,xdim)')*1/r2*inv(B);
  ic = -ib;
  DU(counter,:) = -r2*Uw(counter,:)*ic'*reshape(a(counter,:),xdim,xdim) -
1/r2*(Ustar(lN(counter),:)-Ustar(rN(counter),:));
  Ustar(counter,:) = r2*Uw(counter,:)*ib'*reshape(aprime(counter,:),xdim,xdim)+
1/r2*(Ustar(lN(counter),:)+Ustar(rN(counter),:));
  ib = ib*reshape(a(counter,:),xdim,xdim)';
  ic = ic*reshape(aprime(counter,:),xdim,xdim)';
  b(counter,:) = reshape(ib',1,xdim*xdim);
  c(counter,:) = reshape(ic',1,xdim*xdim);
end;


% DU is computed for the top node.
DU(1,:) =
-r2*Uw(2,:)*(1/r2*inv(B')*inv(reshape(Qtree(1,:),xdim,xdim)')*1/r2*inv(B))'*...
(2*eye(xdim)-reshape(a(2,:),xdim,xdim)) - 1/r2*(Ustar(lN(1),:)+Ustar(rN(1),:));
```

```
clear ic ib counter

% final values
% This must be modified here for non-diagonal R's and multiple C's

d =  (yw1*R1inv)*C + R2factor.*yw2*R2inv*C - DU(1:N,:);

% Initialisation of remaining variables.

cprime = 0*a;
dprime = zeros(N+N,xdim);
dstar = zeros(N+N,xdim);

% These appear for reusability with the constrained case.

muprime = zeros(N+N,xdim);
mustar = zeros(N+N,xdim);
mu = zeros(N+N,xdim);
```

A.2.4 motmup.m

```
% motmup
% This file is the engine of the upsweep, and follows the equations in
% the thesis carefully.
% Progression up the tree is done level by level, which would be split up if
% parallel processors were available.
% The computation begins at the lowest level of the tree, and moves up,
% halving the number of nodes at each level.

Nlevel = N;
```

```
for counter = 1:numlevels
  Nlevel = Nlevel/2;             % number of nodes at each level
  node = [(Nlevel+1):Nlevel*2];  % vector positions of these nodes

% Matlab allows us to do the nodes as a batch
% The variable names follow those used in the thesis, a,b,c,d and mu (for the
% constrained case), modified with primes, daggers and stars.
% Where an intermediate matrix is computed, it is stored with a at as in
% cdaggert is the precursor to cdagger, but has not been stored as a vector.

  for counter2 = node(1):node(length(node))
     R1inv = diag(1./Rm1(counter2,:));
     R2inv = diag(1./Rm2(counter2,:));
     R2inv = R2factor(counter2)*R2inv;
     ab = reshape(a(counter2,:),xdim,xdim)'*reshape(b(counter2,:),xdim,xdim)';
     ac = reshape(a(counter2,:),xdim,xdim)'*reshape(c(counter2,:),xdim,xdim)';
     aprimeb = reshape(aprime(counter2,:),xdim,xdim)'*reshape(b(counter2,:),...
                xdim,xdim)';
     aprimec = reshape(aprime(counter2,:),xdim,xdim)'*reshape(c(counter2,:),...
                xdim,xdim)';
     adaggert = C'*R1inv*C  + C'*R2inv*C+ab -
0.5*reshape((cprime(lN(counter2),:)+cprime(rN(counter2),:)),xdim,xdim)';
     cdaggert = ac -
0.5*reshape((cprime(lN(counter2),:)-cprime(rN(counter2),:)),xdim,xdim)';
     ddagger(counter2,:) = d(counter2,:) -
inv(r2)*(dstar(lN(counter2),:)-dstar(rN(counter2),:)) -
inv(r2)*(dprime(lN(counter2),:)-dprime(rN(counter2),:));
     mudagger(counter2,:) = mu(counter2,:) -
inv(r2)*(mustar(lN(counter2),:)-mustar(rN(counter2),:)) -
inv(r2)*(muprime(lN(counter2),:)-muprime(rN(counter2),:));
```

```
      invadaggercdagger =inv(adaggert)*cdaggert;
      invadaggerddagger(counter2,:) =(inv(adaggert)*ddagger(counter2,:)')';
      invadaggermudagger(counter2,:) =(inv(adaggert)*mudagger(counter2,:)')';

  dstar(counter2,:) =  inv(r2)*(dstar(lN(counter2),:)+dstar(rN(counter2),:)) +
inv(r2)*(dprime(lN(counter2),:)+dprime(rN(counter2),:));
  mustar(counter2,:) =inv(r2)*(mustar(lN(counter2),:)+mustar(rN(counter2),:))+
inv(r2)*(muprime(lN(counter2),:)+muprime(rN(counter2),:));
  cprimet = -(aprimeb +0.5*reshape((cprime(lN(counter2),:)...
      -cprime(rN(counter2),:)),xdim,xdim)') *invadaggercdagger...
      +(aprimec+0.5*reshape((cprime(lN(counter2),:)...
      +cprime(rN(counter2),:)),xdim,xdim)');
  dprime(counter2,:) = (-(aprimeb +
0.5*reshape((cprime(lN(counter2),:)-cprime(rN(counter2),:)),xdim,xdim)')
*invadaggerddagger(counter2,:)')';
  muprime(counter2,:) = (-(aprimeb +
0.5*reshape((cprime(lN(counter2),:)-cprime(rN(counter2),:)),xdim,xdim)')
*invadaggermudagger(counter2,:)')';

% And now the necessary storage from the temporary variables

  cprime(counter2,:) = reshape(cprimet',1,xdim*xdim);
  adagger(counter2,:) = reshape(adaggert',1,xdim*xdim);
  cdagger(counter2,:) = reshape(cdaggert',1,xdim*xdim);

  end;
end;

% The scaling function variables must be computed for the top level.
% In the thesis, these are represented by double daggers
```

```
     counter2 = 1;   % Node number for the scaling function equation
     aprimeb = reshape(aprime(counter2,:),xdim,xdim)'*reshape(b(counter2,:),...
                 xdim,xdim)';
     aprimec = reshape(aprime(counter2,:),xdim,xdim)'*reshape(c(counter2,:),...
                 xdim,xdim)';
     cdaggert = C'*R1inv*C  + R2factor(counter2)*C'*R2inv*C-aprimec -
0.5*reshape((cprime(lN(counter2),:)+cprime(rN(counter2),:)),xdim,xdim)';
     adaggert = -aprimeb -
0.5*reshape((cprime(lN(counter2),:)-cprime(rN(counter2),:)),xdim,xdim)';
     ddagger(counter2,:) = d(counter2,:) -
inv(r2)*(dstar(lN(counter2),:)+dstar(rN(counter2),:)) -
inv(r2)*(dprime(lN(counter2),:)+dprime(rN(counter2),:));
     mudagger(counter2,:) = mu(counter2,:) -
inv(r2)*(mustar(lN(counter2),:)+mustar(rN(counter2),:)) -
inv(r2)*(muprime(lN(counter2),:)+muprime(rN(counter2),:));

  adagger(counter2,:) = reshape(adaggert',1,xdim*xdim);
  cdagger(counter2,:) = reshape(cdaggert',1,xdim*xdim);
```

A.2.5 motmdown.m

```
% motmdown
% This program is the engine of the down sweep, where the measurements are
% used to produce optimal estimates at each point.

% Zeroing

xest = 0*ones(1,xdim); dxest = xest;
```

```
% Now solve for the scaling function and wavelet coefficient state estimates
% as a 2x2 system at the top

TSA = [reshape(cdagger(2,:),xdim,xdim)' reshape(adagger(2,:),xdim,xdim)' ;
reshape(cdagger(1,:),xdim,xdim)' reshape(adagger(1,:),xdim,xdim)'];
TSD = [ddagger(2,:)'+mudagger(2,:)' ; ddagger(1,:)'+mudagger(1,:)'];

xpair = inv(TSA)*TSD ;    % The pairwise solution of the top node.

% Note this is where condition number problems typically cause
% problems where present.

% The estimates for the top node estimates are stored.

xest(2,:) = xpair(1:xdim)';
dxest(2,:) = xpair((xdim+1):(xdim+xdim))'  ;
dxest(1,:) = xest(2,:);

% The child nodes are computed using wavelet reconstruction.

xpair = 1/r2*[(dxest(2,:))+(xest(2,:)); -(dxest(2,:))+(xest(2,:))];
xest(3:4,:) = xpair;

% The downsweep continues recursively until the bottom of the tree is reached.

Nlevel = 1;
counter2 = 3;  % Starting parent node
counter3 = 5;  % Starting child node

while(counter2<N+1)
```

```
  dxest(counter2,:) = (   inv( reshape(adagger(counter2,:),xdim,xdim)' )
*(ddagger(counter2,:)'+mudagger(counter2,:)'...
-reshape(cdagger(counter2,:),xdim,xdim)'*(xest(counter2,:))')))';
  xpair = 1/r2*[dxest(counter2,:)+xest(counter2,:); -dxest(counter2,:)...
      +xest(counter2,:)];
  xest(counter3:counter3+1,:) = xpair;
  counter2 = counter2 +1;  % Move to next parent node
  counter3 = counter3+2;  % Each parent has two child nodes
end;

clear counter2 counter3
```

A.3 The constrained multiscale state estimator

In most cases, the "mg"-series tracks the "motm"-series, with a parallel naming system. Programs that differ significantly are listed here.

A.3.1 mg.m

```
%  mg series - for constrained case based on an identification of violations
% on the way down the tree.

% Strategy
xestrec= [];
% Set up constants
mgsafetyvalve = 0;   % Checks that we don't have too many iterations
mgsp;
mgparm;
mmurec = zeros(1,20);
motmup;
```

```
% Adjust for constraints


mglam;
xorig = xest(N+1:N+N);
xestrec = xest(N+1:N+N);
disp('Original set of constraints activated')
counter4   % This is a vector listing the currently active constraints,
           %by node number.


% get x-> lambda conversion
mgcon;
mgwtbch;
```

A.3.2 mglam.m

```
% mglam - part of the mg series that converts the counter4 matrix to
% the initial xest soln.
disp('Running mglam')


% Incoporates and adapts old mgdown.


% motmdown
xest = 0*ones(1,xdim); dxest = xest;
TSA = [reshape(cdagger(2,:),xdim,xdim)' reshape(adagger(2,:),xdim,xdim)' ;...
        reshape(cdagger(1,:),xdim,xdim)' reshape(adagger(1,:),xdim,xdim)'];
TSD = [ddagger(2,:)'+mudagger(2,:)' ; ddagger(1,:)'+mudagger(1,:)'];
xpair = inv(TSA)*TSD ;    % The pairwise solution of the top node.
xest(2,:) = xpair(1:xdim)';
dxest(1,:) = xest(2,:);
```

```
% Now for the downsweep - this is modified since we identify constraint
% violations while looking for optimal solutions
% dataset contains all constraints to be checked - with 99999 as a
% place holder for the last.
% counter4 is the set of violated constraints.

Nlevel = 1;
counter4 = [];
dataset = [2:N 99999];
dsindex = 1;
counter2 = dataset(dsindex);
counter3 = lN(counter2);

% Eliminate the trivial solution first
if min(Xubar(2)*diag(ones(xdim))-xest(2,:)<1e-15)
  disp('top node is currently on bound')
% Sp we set the top node to it's bound, set all remaining wavelet coefficients
% to zero
  xest(2,:) = Xubar(2)*diag(ones(xdim));
  dxest(2,:) = 0*Xubar(2)*diag(ones(xdim));
    counter4 = [counter4; 2];  % Storing node two as a violated constraint
    st = subtree(2,lN,rN);           % Identifying all nodes on the subtree
                                     % below node two
    for counter6 = 1:length(st)
       xest(st(counter6),:) = ones(1,xdim)*Xubar(st(counter6));
       dxest(st(counter6),:) = zeros(1,xdim);
       ff = find(dataset==st(counter6));
       if isempty(ff)
          disp('Zeroth node found in counter6 loop 1')
       else
```

```
          dataset = [dataset(1:ff-1) dataset(ff+1:length(dataset))];
       end;
    end;
  dataset
end

while(dsindex<length(dataset))
  counter2 = dataset(dsindex); % moving to the next node without a violation
  counter3 = lN(counter2);     % identifying the next child node to be computed
  dxest(counter2,:) = ...      % solving for the next wavelet coefficient
    (   inv( reshape(adagger(counter2,:),xdim,xdim)' )*(ddagger(counter2,:)'...
     +mudagger(counter2,:)'-reshape(cdagger(counter2,:),xdim,xdim)'...
     *(xest(counter2,:))')))';
  xpair = 1/r2*[dxest(counter2,:)+xest(counter2,:); -dxest(counter2,:)+...
           xest(counter2,:)]
% Using wavelet reconstruction to identify apparently optimal scaling function

% When a constraint violation is identified we must
% - correct the sibling point
% - set all values on subtree to the bound
% - add violating point to constraint set
% - delete subtree from search set

  if min(Xubar(counter3)*diag(ones(xdim))-xpair(1,:)<1e-15)
% ie upper constraint is violated for the left node
    counter4 = [counter4; counter3];

    st = subtree(counter3,lN,rN);   % Identify potentially active subtree

% And set the subtree to its bound
```

```
    for counter6 = 1:length(st)
       xest(st(counter6),:) = ones(1,xdim)*Xubar(st(counter6));
       dxest(st(counter6),:) = zeros(1,xdim);
       ff = find(dataset==st(counter6));
       if isempty(ff)
          disp('Zeroth level node in counter6 loop 2')
       else
          dataset = [dataset(1:ff-1) dataset(ff+1:length(dataset))];
       end;
    end;
    xpairdiffn = 1;
    xpairdiff = xpair(1,:)-ones(1,xdim)*Xubar(counter3);
    mgcp;      % This changes the values of the parent nodes to adjust
               % for the change
    xpair(1,:) = ones(1,xdim)*Xubar(counter3);
  elseif max(Xlbar(counter3)*diag(ones(xdim))-xpair(1,:)>0)
    counter4 = [counter4; -counter3];
    disp('Lower constraint violation identified')
    % And reverse all of the signs and replace upper with lower bounds
  end

% Then the argument is repeated for the right node

  if min(Xubar(counter3+1)*diag(ones(xdim))-xpair(2,:)<1e-15)
    counter4 = [counter4; counter3+1];
    st = subtree(counter3+1,lN,rN);
    for counter6 = 1:length(st)
       xest(st(counter6),:) = ones(1,xdim)*Xubar(st(counter6));
       dxest(st(counter6),:) = zeros(1,xdim);
       ff = find(dataset==st(counter6));
```

```
        if isempty(ff)
           disp('Zeroth level node found in counter6 loop 3')
        else
           dataset = [dataset(1:ff-1) dataset(ff+1:length(dataset))];
        end;
     end;
     xpairdiffn = 2;
     xpairdiff = xpair(2,:)-ones(1,xdim)*Xubar(counter3+1);
     mgcp;      % This changes the values of the parent nodes to adjust
                % for the change
     xpair(2,:) = ones(1,xdim)*Xubar(counter3+1);
  elseif max(Xlbar(counter3+1)*diag(ones(xdim))-xpair(2,:)>0)
     counter4 = [counter4; -counter3-1];
     disp('Lower constraint violation identified')
  end

% Finally we have the correct state estimates
  xest(counter3:counter3+1,:) = xpair;

% And are ready to move onto the next node for consideration
  dsindex = dsindex +1;
end;  % while

% At the bottom of the tree, the zeroth level can be computed without
% rigorous checking

counter2 = N/2+1;
counter3 = N+1;
while(counter2<N+1)
  xpair = 1/r2*[dxest(counter2,:)+xest(counter2,:);...
```

```
                 -dxest(counter2,:)+xest(counter2,:)];
  xest(counter3:counter3+1,:) = xpair;
  counter2 = counter2 +1;
  counter3 = counter3+2;
end;

clear counter2 counter3

% counter4 contains a list of all constraint violations, from which a
% constraint tree can be constructed.
% positive if it is an upper bound violation, negative if lower.

counter4 = [counter4 ; 99999];
disp('Finished mglam')
```

A.3.3 mgcp.m

```
% mgcp - part of the mglam series, changes the parent nodes in response to
% a constraint violation.

% The parent branch of the tree must be adjusted so that no other nodes on
% the tree are affected, ie increasing co-nodar wavelet and scaling function
% estimates simultaneously

tempnode = counter3-1+xpairdiffn;    % coutner3 for lN, ..+1for rN
tempnodep = pN(counter3);

xpairdiff = 1/r2*xpairdiff;

while(tempnodep>1)
  if(rem(tempnode,2)==0)
```

```
    signnode = -1;  % Right node
  else
    signnode=1;     % Left node
  end
  xest(tempnodep,:) = xest(tempnodep,:)-xpairdiff;
  dxest(tempnodep,:) = dxest(tempnodep,:)-signnode*xpairdiff;
  tp = tempnodep;
  tempnodep = pN(tempnode);
  tempnode = tp;
  xpairdiff = 1/r2*xpairdiff;
end;

clear tempnode tempnodep signnode tp
```

A.3.4 mgcon.m

```
% mgcon.m converts a set of x's to a set of lambdas

% File converts unconstrained state estimate to first round of constrained
% state estimate and constraints to produce the closest feasible solution
% and then checks to see whether constraints are satisfied.

% Program needs motmmain and motmqpc run before

% Receive solution
% since constraints are in terms of zeroth level data, use these.

xuc = xest(N+1:N+N,:);
xucwt = wt(xuc);
xucsf = sf(xuc);
```

```
% Now since we have dagger equations precomputed, we can obtain mudagger's
% Equation to be solved is adag*dx + cdag*x = ddag+mudag
% Transform is xuc -> mudaguc
% This is a modified form of motmdown, essentially just for the mu variables

mudag = 0*ones(1,xdim);
counter2 = 1;
  mudag(counter2,:) = ( reshape(adagger(counter2,:),xdim,xdim)'...
*(xucwt(counter2+1,:))'  -ddagger(counter2,:)' +reshape(cdagger(counter2,:),...
      xdim,xdim)'*(xucsf(counter2+1,:))')';
counter2 = counter2+1;
while(counter2<N+1)
  mudag(counter2,:) = ( reshape(adagger(counter2,:),xdim,xdim)'...
      *(xucwt(counter2,:))'-ddagger(counter2,:)'...
      +reshape(cdagger(counter2,:),xdim,xdim)'*(xucsf(counter2,:))')';
  counter2 = counter2 +1;
end;

clear counter2

% End of motmdown.
% The mudag need to be converted into mustars and mus
% This is from motmup, modified to compute only the mu variables

mustarint = mustar;
muprimeint = muprime;
Nlevel = N;
for counter = 1:numlevels
  Nlevel = Nlevel/2;             % number of nodes at each level
  node = [(Nlevel+1):Nlevel*2];  % vector positions of these nodes
  for counter2 = node(1):node(length(node))
     R1inv = diag(1./Rm1(counter2,:));
     R2inv = diag(1./Rm2(counter2,:));
     R2inv = R2factor(counter2)*R2inv;
     ab = reshape(a(counter2,:),xdim,xdim)'*reshape(b(counter2,:),xdim,xdim)';
     aprimeb = reshape(aprime(counter2,:),xdim,xdim)'*reshape(b(counter2,:),...
              xdim,xdim)';
     adaggert = C'*R1inv*C  + C'*R2inv*C+ab...
      -0.5*reshape((cprime(lN(counter2),:)+cprime(rN(counter2),:)),xdim,xdim)';
     muint(counter2,:)=mudag(counter2,:)+inv(r2)*(mustarint(lN(counter2),:)...
           -mustarint(rN(counter2),:)) +inv(r2)*(muprimeint(lN(counter2),:)...
           -muprimeint(rN(counter2),:));

     invadaggermudagger(counter2,:) =(inv(adaggert)*mudag(counter2,:)')';

  mustarint(counter2,:) =  inv(r2)*(mustarint(lN(counter2),:)...
     +mustarint(rN(counter2),:)) +inv(r2)*(muprimeint(lN(counter2),:)...
     +muprimeint(rN(counter2),:));
  muprimeint(counter2,:) = (-(aprimeb + 0.5*reshape((cprime(lN(counter2),:)...
    -cprime(rN(counter2),:)),xdim,xdim)')   *invadaggermudagger(counter2,:)')';
 end;
end;

% And mus for the top nodes represented by double daggers
 counter2 = 1;   % For the scaling function equation
 muint(counter2,:) = mudag(counter2,:) + inv(r2)*(mustarint(lN(counter2),:) ...
        +mustarint(rN(counter2),:)) +inv(r2)*(muprimeint(lN(counter2),:)...
        +muprimeint(rN(counter2),:));

% End of motmup - modified
% So here, muint is the set of lagrangian variables
% These are reconstructed to form a set of mu at the zeroth level

muinttim = iwt(muint);
muintsf = sf(muinttim);

% Then all that remains is to check whether KKT are satisfied, ie,
```

A.3.5 mgwtbch.m

```
% mgwtbch converts the counter4 vector to a new counter4 vector by inspecting
% the child nodes and the computed mu's into real mu's

% It essentially checks for constraint violations identified at upper levels,
%  that the level has been identified correctly, and that lagrangeans all have
% the correct sign on the subtree.

% based on mu_tau = mu'*div_x(g)

numconstr = length(counter4)-1;

basis = zeros(1,N)';
delg = [];
counter4new = [];

for k = 1:numconstr
 if(counter4(k)<N+1)
   st = subtree(counter4(k), lN, rN);
   st = sort([lN(st((length(st)+1)/2:length(st)))...
               rN(st((length(st)+1)/2:length(st)))] );
 else
   st = counter4(k);
 end;
 for j = 1:length(st)
   node = st(j);
   if(node<N+1)
     if(node==2)
       basisnode = basis;
       basisnode(1) = 1;
     else
       basisnode = basis;
       basisnode(node) = 1;
       basisnode = sf2wt(basisnode);
     end;
   else
     basisnode = basis;
     basisnode(node-N) = 1;
     basisnode = wt(basisnode);
   end;
   delg = [delg ; basisnode'];
   counter4new = [counter4new ; st(j)];
 end;
end;

% This creates a set of lagrangean variables in terms of the zeroth level
% variables.

mufinal = inv(delg*delg')*delg*muint

% These must be negative for the supernode to be active.
% Now sort the list to find the highest node on the tree where all in
% subtree are negative.

[cci, ccj] = find(mufinal<1e-15);

counter4new = sort(counter4new(cci)); % keep only the negatives)
counter4new = [counter4new; 99999];
for j = 1:numlevels
   k = 1;
   while(counter4new(k)~=99999)
       if(rem(counter4new(k),2)==1)
         if(counter4new(k)+1==counter4new(k+1))
           counter4new = [counter4new(1:k-1) ; pN(counter4new(k)) ; ...
                          counter4new(k+2:length(counter4new))];
         end;
       end;
    k = k+1;
    end;
    counter4new = sort(counter4new);
end;

counter4new
counter4 = counter4new;
% And insert an artificial set of lagrangeans to continue the algorithm.
mufinal = -diag(eye(length(counter4)-1));

clear st cc1 cc2 counter4new
```

A.3.6 mmu.m

```
%Mmu
```

```
%  Mmu comes straight from delg

% counter4 gives the raw list of previously tried mu
if(counter4(length(counter4))==99999)
   counter4 =    counter4(1:length(counter4)-1);  % Kill that 99999
end
% Note counter4 not mufinal
negs = find(mufinal<0);    %satisfied constraints
numnegs = length(negs);

% Choose set of constraints to be satisfied, then set appropriate
% counter4 for next iteration.

% Strategy:

% Find constraints with positive mu

% For the set of constraints there are various options
% Assume we bound right node first, (arbitrary, based on approach
%     to a steady state at a bound)
% 1. try bounding right node - success -> stop
%                left node violation -> try bounding left node
%                mu still positive -> repeat, going deeper

% 2. try bounding left node - success -> stop
%                right node violation -> problem with cycling
%                         (can we prove this won't happen)
%                mu still positive -> repeat, going deeper
% We can prove that left node and right node cannot both be bounded,
% but we can't prove that the non bounded node has lam=0, since
% child nodes may have bound violations to be discovered.  This
% is ok though, and will show up in a later iteration.

% find constraints

actives = [counter4(negs)];
mmucomp

% We've computed the right nodes at this point.  Find all of the right nodes
%\ that are unsatisfied.

mufinal = mmusol(length(mmusol)-length(actives)+1:length(mmusol))

negls = find(mufinal<0);    %satisfied constraints
posls = find(mufinal>1e-16);  % need to be sat

% Find which appear twice and move down the tree

counter4 = [counter4; 99999];

newactives = [];
newmus = [];

% First check initially satisfied constraints.
c1=1;
while(c1<length(negs)+1)
  if(mufinal(c1)>0)    % Then constraint satisified at least once
          % constraint not satisified for node, relax
```

```
    if(actives(c1)<N+1)
      % change to  left node
      actives(c1) = lN(actives(c1));
       % and add right node
      newactives =  [newactives; rN(actives(c1))];
      newmus = [newmus; mufinal(c1)];
     else  % we're at the bottom of the tree - just delete it from set
       actives = [actives(1:c1-1); actives(c1+1:length(actives))];
       mufinal = [mufinal(1:c1-1); mufinal(c1+1:length(mufinal))];
       negs = negs(1:length(negs)-1)
     end;
  end;
  c1=c1+1;
end;

% Parents will not cause problems.

mgsafetyvalve = mgsafetyvalve+1;

counter4 = [actives ; newactives]
mufinal = [mufinal; newmus];

[counter4, iii] = sort(counter4);
mufinal = mufinal(iii);

if(max(mufinal)>0)
   mmurec(size(mmurec,1)+1,1:length(counter4)) = counter4';
   mmu;
end;
```

```
% Next step is the stopping condition.  Having found the final set of
% constraints, check to produce the final set of values before proceding to
% the final solution.
% This is necessary because even though the active constraints have been
% identified, haven't computed all state values using these constraints yet.

% counter4 gives the raw list of previously tried mu
if(counter4(length(counter4))==99999)
   counter4 =    counter4(1:length(counter4)-1);  % Kill that 99999
end
actives = counter4;
mmucomp

mufinal = mmusol(length(mmusol)-length(actives)+1:length(mmusol))

if(mufinal<0)
   % find final solution
   mglam2;  % This will go down the tree and compute the next iteration
else
   % Try again
   counter4 = [counter4; 99999];
   mmu
end;
```

A.3.7 mglam2.m

```
% mglam - part of the mg series that converts the counter4 matrix to an
% updated xest soln.

disp('Running mglam2')
```

```
% This is run after we have identified a new set of active constraints, with
% upper levels of the tree taken care of.  Strategy is to begin at sibling
% nodes to the constraints and move down the tree.  The key then is that the
% starting set is set to the siblings of the active nodes.

if(counter4(length(counter4))==99999)
   counter4old = counter4(1:length(counter4)-1);
else
   counter4old = counter4;
end;
mufinalold = mufinal;

dxest(parents(2:lp),:) = mmusol(1:2:2*length(parents)-2);
xest(parents(2:lp),:) = mmusol(2:2:2*length(parents)-2);

% All is fine if we compute the correct mudaggers

numconstr = length(counter4);
basis = zeros(1,N)';
delg = [];
for j = 1:numconstr
   node = counter4(j);
   if(node<N+1)
     basisnode = basis;
     basisnode(node) = 1;
     basisnode = sf2wt(basisnode);
   else
     basisnode = basis;
     basisnode(node-N) = 1;
     basisnode = wt(basisnode);
   end;
   delg = [delg ; basisnode'];
end;

mu = delg'*mufinal;

% And now convert to mudaggers
% from motmup

Nlevel = N;
for counter = 1:numlevels
  Nlevel = Nlevel/2;             % number of nodes at each level
  node = [(Nlevel+1):Nlevel*2];  % vector positions of these nodes
  for counter2 = node(1):node(length(node))
     R1inv = diag(1./Rm1(counter2,:));
     R2inv = diag(1./Rm2(counter2,:));
     R2inv = R2factor(counter2)*R2inv;
     ab = reshape(a(counter2,:),xdim,xdim)'*reshape(b(counter2,:),xdim,xdim)';
     aprimeb = reshape(aprime(counter2,:),xdim,xdim)'*reshape(b(counter2,:),...
                xdim,xdim)';
     adaggert = C'*R1inv*C+C'*R2inv*C+ab-0.5*reshape((cprime(lN(counter2),:)...
                +cprime(rN(counter2),:)),xdim,xdim)';
     mudagger(counter2,:) = mu(counter2,:) - inv(r2)*(mustar(lN(counter2),:)...
              -mustar(rN(counter2),:)) -    inv(r2)*(muprime(lN(counter2),:)...
              -muprime(rN(counter2),:));
     invadaggermudagger(counter2,:) =(inv(adaggert)*mudagger(counter2,:)')';

  mustar(counter2,:)=inv(r2)*(mustar(lN(counter2),:)+mustar(rN(counter2),:))...
                 +inv(r2)*(muprime(lN(counter2),:)+muprime(rN(counter2),:));
  muprime(counter2,:) = (-(aprimeb + 0.5*reshape((cprime(lN(counter2),:)...
      -cprime(rN(counter2),:)),xdim,xdim)')*invadaggermudagger(counter2,:)')';
  end;
end;

% In the reports, these are represented by double daggers

     counter2 = 1;   % For the scaling function equation
     aprimeb = reshape(aprime(counter2,:),xdim,xdim)'*reshape(b(counter2,:),...
                 xdim,xdim)';
     adaggert = -aprimeb - 0.5*reshape((cprime(lN(counter2),:)...
                 -cprime(rN(counter2),:)),xdim,xdim)';
     mudagger(counter2,:) = mu(counter2,:) - inv(r2)*(mustar(lN(counter2),:)...
            +mustar(rN(counter2),:)) -    inv(r2)*(muprime(lN(counter2),:)...
            +muprime(rN(counter2),:));

% Incoporates and adapts old mgdown.

% Now for the downsweep

Nlevel = 1;

counter4 = [];
dataset = [2:N 99999];
dsindex = 1;
counter2 = dataset(dsindex);
counter3 = lN(counter2);

while(dsindex<length(dataset))
  counter2 = dataset(dsindex);
  counter3 = lN(counter2);
  dxest(counter2,:) = (   inv( reshape(adagger(counter2,:),xdim,xdim)' ) ...
   *(ddagger(counter2,:)'+mudagger(counter2,:)'-reshape(cdagger(counter2,:),...
          xdim,xdim)'*(xest(counter2,:))')))';
  xpair = 1/r2*[dxest(counter2,:)+xest(counter2,:); ...
                 -dxest(counter2,:)+xest(counter2,:)];



% When a constraint violation is identified we must
% - correct the sibling point
% - set all values on subtree to the bound
% - add violating point to constraint set
% - delete subtree from search set

  if min(Xubar(counter3)*diag(ones(xdim))-xpair(1,:)<1e-15)
    counter4 = [counter4; counter3];

    st = subtree(counter3,lN,rN);
    for counter6 = 1:length(st)
       xest(st(counter6),:) = ones(1,xdim)*Xubar(st(counter6));
       dxest(st(counter6),:) = zeros(1,xdim);
       ff = find(dataset==st(counter6));
       if ff==[]
          disp('Zeroth node in counter6 loop 1')
       else
          dataset = [dataset(1:ff-1) dataset(ff+1:length(dataset))];
       end;
    end;
    xpairdiffn = 1;
    xpairdiff = xpair(1,:)-ones(1,xdim)*Xubar(counter3);
    mgcp;      % This changes the values of the parent nodes to adjust for
```

```
             % the change
  xpair(1,:) = ones(1,xdim)*Xubar(counter3);

elseif max(Xlbar(counter3)*diag(ones(xdim))-xpair(1,:)>0)
  counter4 = [counter4; -counter3];
  disp('Lower constraint hit')
end
if min(Xubar(counter3+1)*diag(ones(xdim))-xpair(2,:)<1e-15)
  counter4 = [counter4; counter3+1];

  st = subtree(counter3+1,lN,rN);
  for counter6 = 1:length(st)
     xest(st(counter6),:) = ones(1,xdim)*Xubar(st(counter6));
     dxest(st(counter6),:) = zeros(1,xdim);
     ff = find(dataset==st(counter6));
     if ff==[]
        disp('Zeroth node found in counter6 loop 2')
     else
        dataset = [dataset(1:ff-1) dataset(ff+1:length(dataset))];
     end;
  end;
  xpairdiffn = 2;
  xpairdiff = xpair(2,:)-ones(1,xdim)*Xubar(counter3+1);
  mgcp;      % This changes the values of the parent nodes to adjust for
             % the change
  xpair(2,:) = ones(1,xdim)*Xubar(counter3+1);

elseif max(Xlbar(counter3+1)*diag(ones(xdim))-xpair(2,:)>0)
  counter4 = [counter4; -counter3-1];
  disp('Lower constraint hit')
  end

  xest(counter3:counter3+1,:) = xpair;
  dxest(counter2,:) = 1/r2*(xpair(1,:)-xpair(2,:));
  dsindex = dsindex +1;
end;

counter2 = N/2+1;
counter3 = N+1;
while(counter2<N+1)
  xpair = 1/r2*[dxest(counter2,:)+xest(counter2,:); -dxest(counter2,:)...
        +xest(counter2,:)];
  xest(counter3:counter3+1,:) = xpair;
  counter2 = counter2 +1;
  counter3 = counter3+2;
end;

xestrec = [xestrec  xest(N+1:N+N)];

clear counter2 counter3

% counter4 arrives the first time with a list of all constraint violations.
% negative if it is an upper bound violation, positive if lower.
 counter4 = [counter4 ; 99999];

mgcon;
mgwtbch;
counter4 = counter4(1:length(counter4)-1);

% At this point ready to going mmu again if necessary
```

```
if(length(counter4)==length(counter4old))
  if(counter4==counter4old)
     disp('the solution has converged')
  end;
else
   disp('The solution has not converged')
   disp('Old and new constraint sets')
   [counter4; counter4old]
   counter4 = [counter4 ; 99999];
   mmu;
end;
% At this point ready to going mmu again if necessary
```

A.3.8 mmucomp.m

```
% mmucomp.m


% This is a subprog of mmu - it takes a set of active nodes and returns the
% state and lagrangean solution for the given set.


disp('computing solution with basis')
actives'


parents = parfind(actives,pN);
lp = length(parents);


% Construct a delg - see mgwtb


numconstr = length(actives);


basis = zeros(1,N)';
```

```
delg = [];

for j = 1:numconstr
   node = actives(j);
   if(node<N+1)
     basisnode = basis;
     basisnode(node) = 1;
     basisnode = sf2wt(basisnode);
   else
     basisnode = basis;
     basisnode(node-N) = 1;
     basisnode = wt(basisnode);   % This is for actives at zeroth level
   end;
   delg = [delg ; basisnode'];
end;

% delgp = delg(:,parents)';
d2mudag;

% delgp needs to be converted to mu doubledaggers

mmuA = [adagger(1) cdagger(1) ; adagger(2) cdagger(2)];
mmub = [ddagger(1) ; ddagger(2)];

for k = 3:length(parents)
  lmmuA = size(mmuA,1);
  mmuA = [mmuA zeros(lmmuA,2*xdim) ; ...
    zeros(xdim,size(mmuA,2)) adagger(parents(k)) cdagger(parents(k))];
  mmub = [mmub; ddagger(parents(k))];
end
```

```
mmuA = [mmuA -delgp];

% And now put the wavelet transforms at the bottom
mmuwt = zeros(length(parents)-1,size(mmuA,2));
mmuwtb = zeros(length(parents)-1,1);
k = 1;
for j = 2:length(parents)
   [mmui] = find(parents(2:lp)==lN(parents(j)));
   if(mmui~=[])
     mmuwt(k,2*(j-1)-1:2*(j-1)) = 1/r2*[1 1];
     mmuwt(k,2*(mmui)) = -1;
     k = k+1;
   end
   [mmui] = find(parents(2:lp)==rN(parents(j)));
   if(mmui~=[])
     mmuwt(k,2*(j-1)-1:2*(j-1)) = 1/r2*[-1 1];
     mmuwt(k,2*(mmui)) = -1;
     k = k+1;
   end
end;
for j = 1:length(actives)
   [mmui] = find(lN(parents(2:lp))==actives(j));
   if(mmui~=[])
     mmuwt(k,2*(mmui)-1:2*(mmui)) = 1/r2*[1 1];
     mmuwtb(k) = Xubar(actives(j));
     k = k+1;
   end
   [mmui] = find(rN(parents(2:lp))==actives(j));
   if(mmui~=[])
```

```
      mmuwt(k,2*(mmui)-1:2*(mmui)) = 1/r2*[-1 1];
      mmuwtb(k) = Xubar(actives(j));
      k = k+1;
    end
end;

mmuA = [mmuA; mmuwt];
mmub = [mmub; mmuwtb];

mmusol = inv(mmuA)*mmub

mmubranch = mmusol(1:length(mmusol)-length(actives));
mmubranch = 1/r2*[mmubranch(1:2:length(mmubranch)-1)...
      + mmubranch(2:2:length(mmubranch)),-mmubranch(1:2:length(mmubranch)-1)...
      + mmubranch(2:2:length(mmubranch)), r2*parents(2:lp)]
xubar
```

A.3.9 d2mudag.m

```
% d2mudag

% This program takes the delg produced in mmu.m, used as mu and converts it
% to a set of mudag

% The delg is of the form mu(tau) = delg*mu(t)
% For each mu(t), we need to construct mudagger(tau), then we can concatenate.

delgp = [];
delgold = delg;

for munum = 1:size(delg,1)
```

```
mustar = 0*xest;
muprime = 0*xest;
mu = delg(munum,:)';

Nlevel = N;
for counter = 1:numlevels
  Nlevel = Nlevel/2;             % number of nodes at each level
  node = [(Nlevel+1):Nlevel*2];  % vector positions of these nodes
  for counter2 = node(1):node(length(node))
     R1inv = diag(1./Rm1(counter2,:));
     R2inv = diag(1./Rm2(counter2,:));
     R2inv = R2factor(counter2)*R2inv;
     ab = reshape(a(counter2,:),xdim,xdim)'*reshape(b(counter2,:),xdim,xdim)';
     aprimeb = reshape(aprime(counter2,:),xdim,xdim)'*reshape(b(counter2,:)...
                  ,xdim,xdim)';
     mudagger(counter2,:) = mu(counter2,:) - inv(r2)*(mustar(lN(counter2),:)...
               -mustar(rN(counter2),:)) -    inv(r2)*(muprime(lN(counter2),:)...
               -muprime(rN(counter2),:));
 mustar(counter2,:) =  inv(r2)*(mustar(lN(counter2),:)+mustar(rN(counter2),:))...
                   +inv(r2)*(muprime(lN(counter2),:)+muprime(rN(counter2),:));
 adaggert = C'*R1inv*C  + C'*R2inv*C+ab - 0.5*reshape((cprime(lN(counter2),:)...
                         +cprime(rN(counter2),:)),xdim,xdim)';
     invadaggermudagger(counter2,:) =(inv(adaggert)*mudagger(counter2,:)')';

     muprime(counter2,:) = (-(aprimeb + 0.5*reshape((cprime(lN(counter2),:)...
    -cprime(rN(counter2),:)),xdim,xdim)')   *invadaggermudagger(counter2,:)')';
  end;
end;
```

```
% In the reports, these are represented by double daggers

     counter2 = 1;   % For the scaling function equation

     mudagger(counter2,:) = mu(counter2,:) - inv(r2)*(mustar(lN(counter2),:)...
        +mustar(rN(counter2),:)) -    inv(r2)*(muprime(lN(counter2),:)...
        +muprime(rN(counter2),:));

delgp = [delgp mudagger];
end;

delgp = delgp(parents,:);
```

A.4 Direct solving programs

A.4.1 mpcqp.m

```
% mpcqp.m

% this program produces the full quadratic program solution
% and constructs the full matrix by utilising the wavelet structure.
% convenient for looking at the matrix explicitly, and checking small problems.
% considerably slower than up-down type approaches.
% Requires parameter set up programs to be run first.

% xqp = [dxest(top), xest(top), dxest(topalpha) etc]
% Ax<=b are the wavelet decompositions

lev = [];
lev = [numlevels; numlevels];
for counter = numlevels-1:-1:1
```

```
   lev = [lev; counter*ones(2^(1+numlevels-counter),1)];
end;

% node references back to original program numbering scheme.
node = [2:N ; 2:N];
node = node(:);
Hqp = zeros(2*N-2);
for counter = 1:2:(2*N-3)
    matrixpos = xdim*(counter-1)+1:xdim*counter;
    R1inv = diag(1./Rm1(node(counter),:));
    R2inv = diag(1./Rm2(node(counter),:));
    R2inv = R2factor(node(counter))*R2inv;
    qpa = reshape(Alevels(lev(counter),:),xdim,xdim);
    qpbu = Uw(node(counter),:);
    midbit = 0.5*inv(B')*inv(reshape(QRM(lev(counter),:),xdim,xdim))*inv(B);
    Hqp(matrixpos,matrixpos) =2*(eye(xdim)+qpa)'*midbit*(eye(xdim)+qpa) +
2*C'*R1inv*C+2*C'*R2inv*C;
    Hqp(matrixpos+xdim,matrixpos+xdim)  = 2*(eye(xdim)-qpa)'*midbit*...
            (eye(xdim)-qpa);
    Hqp(matrixpos,matrixpos+xdim) =
-(eye(xdim)-qpa)'*midbit*(eye(xdim)+qpa)-(eye(xdim)+qpa)'*midbit*(eye(xdim)-qpa);
    Hqp(matrixpos+xdim,matrixpos) = Hqp(matrixpos,matrixpos+xdim)';
    fqp(matrixpos) = (-2*yw1(node(counter),:)*R1inv*C ...
-2*yw2(node(counter),:)*R2inv*C)'+2*r2*(eye(xdim)+qpa)*midbit*qpbu';
    fqp = fqp(:);
    fqp(matrixpos+xdim) = 0*fqp(matrixpos)-2*r2*(eye(xdim)-qpa)*midbit*qpbu';
end;

% Getting top node things right.
counter = 2;
```

```
    R1inv = diag(1./Rm1(node(counter),:));
    R2inv = diag(1./Rm2(node(counter),:));
    R2inv = R2factor(node(counter))*R2inv;
    qpa = reshape(Alevels(lev(counter),:),xdim,xdim);
    qpbu = Uw(node(counter),:);
Hqp(xdim+1:2*xdim,xdim+1:2*xdim) = Hqp(xdim+1:2*xdim,xdim+1:2*xdim) +
2*C'*diag(1./Rm1(2,:))*C + 2*C'*diag(1./Rm2(2,:))*C;

fqp(xdim+1:2*xdim) =fqp(xdim+1:2*xdim) +(-2*yw1(1,:)*R1inv*C...
         -2*yw2(1,:)*R2inv*C)';

% Now we need to add prior information to control first point
if(isempty(ixmqpc))
   Po1 = 1e-10*eye(xdim);
   Po1x = inv(Po1)*0.001*ones(xdim,1);
else
   Po1 = 1e-10*Q;
   Po1x = inv(Po1)*ixmqpc(1,:)';
end;

% LEft node sf and wt are N-3 and N-4

% Hqp = 0*Hqp;   For testing purposes, must be commented out.

Hqp((xdim*(N-1)-xdim+1):(xdim*(N-1)),(xdim*(N-1)-xdim+1):(xdim*(N-1))) =
Hqp((xdim*(N-1)-xdim+1):(xdim*(N-1)),(xdim*(N-1)-xdim+1):(xdim*(N-1)))...
     + inv(Po1);
Hqp((xdim*(N)-xdim+1):(xdim*(N)),(xdim*(N-1)-xdim+1):(xdim*(N-1))) =
Hqp((xdim*(N)-xdim+1):(xdim*(N)),(xdim*(N-1)-xdim+1):(xdim*(N-1))) + inv(Po1);
Hqp((xdim*(N-1)-xdim+1):(xdim*(N-1)),(xdim*(N)-xdim+1):(xdim*(N))) =
Hqp((xdim*(N-1)-xdim+1):(xdim*(N-1)),(xdim*(N)-xdim+1):(xdim*(N))) + inv(Po1);
Hqp((xdim*(N)-xdim+1):(xdim*(N)),(xdim*(N)-xdim+1):(xdim*(N))) =
Hqp((xdim*(N)-xdim+1):(xdim*(N)),(xdim*(N)-xdim+1):(xdim*(N))) + inv(Po1);

fqp((xdim*(N-1)-xdim+1):(xdim*(N-1))) = fqp((xdim*(N-1)-xdim+1):(xdim*(N-1)))...
     -r2*Po1x;
fqp((xdim*(N)-xdim+1):(xdim*(N))) = fqp((xdim*(N)-xdim+1):(xdim*(N)))-r2*Po1x;

% This section eliminates the scaling function coefficients except the top one

H = Hqp;
f = fqp;
counterpar = N-2;
counterch = 2*N-4;
while(counterpar>1)
   par = xdim*(counterpar-1)+1:xdim*counterpar;
   chL = xdim*(counterch-1)+1:xdim*(counterch);
   counterchR = counterch + 2;
   chR = xdim*(counterchR-1)+1:xdim*(counterchR);
   H(:,par) = H(:,par)+1/r2*(H(:,chL)+H(:,chR));
   H(:,par-xdim) = H(:,par-xdim)+1/r2*(H(:,chL)-H(:,chR));
   H = [H(:,1:chL(1)-1) H(:,chL(xdim)+1:chR(1)-1) H(:,chR(xdim)+1:size(H,2))];
   H(par,:) = H(par,:)+1/r2*(H(chL,:)+H(chR,:));
   H(par-xdim,:) = H(par-xdim,:)+1/r2*(H(chL,:)-H(chR,:));
   f(par) = f(par)+1/r2*(f(chL)+f(chR));
   f(par-xdim) = f(par-xdim)+1/r2*(f(chL)-f(chR));
   counterch = counterch -4;
   counterpar = counterpar -2;

   H = [H(1:chL(1)-1,:); H(chL(xdim)+1:chR(1)-1,:); H(chR(xdim)+1:size(H,1),:)];
   f = [f(1:chL(1)-1); f(chL(xdim)+1:chR(1)-1); f(chR(xdim)+1:length(f))];
end;

H = 0.5*(H+H');


clear qpparent qpchild




% Now to set up the constraints that are the wavelet decomps


% xqp = QP(Hqp, fqp, Aqp, bqp,1e12*ones(2*(N-1)*xdim,1),-1e12*ones(2*(N-1)*xdim,1),
0*ones(2*(N-1)*xdim,1), (N-2)*xdim);


% Can constrain begin point.


xqp = QP(H,f, [], []);
%tqp = etime(clock, t0);
xqpoutput = xqp;
%disp('done with qp')


% Now to reshape appropriately
xmqp = -inv(H)*f;
xmqpq = reshape(xmqp,xdim,N);
xmqpq = xmqpq';
temp = xmqpq(2,:);
xmqpq(2,:) = xmqpq(1,:);
xmqpq(1,:) = temp ;
ixmqpq = iwt(xmqpq);
```

```
% These two prove the correctness of the Hqp->H decimation

% Checked for f too.
xmqpc = reshape(xqp,xdim,N);
xmqpc = xmqpc';
temp = xmqpc(2,:);
xmqpc(2,:) = xmqpc(1,:);
xmqpc(1,:) = temp ;
ixmqpc = iwt(xmqpc);
```