UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

University of Pretoria

Department of Chemical Engineering

Master's Dissertation

Computationally efficient formulation of stochastic dynamical control within the context of switching probabilistic graphical models

Author:	*Student Number:*
St. Elmo Wilken	29034133
Co-Supervisor:	*Department:*
Mr. C Sandrock	Chemical Engineering
Co-Supervisor:	*Department:*
Dr. JP de Villiers	Electrical, Electronic and Computer Engineering

August 31, 2015

Contents

1 Introduction **4**

I Literature, theory and background material **7**

2 Literature review **8**
- 2.1 Stochastic model predictive control . . . 8
- 2.2 Switching model predictive control . . . 12

3 Background theory **16**
- 3.1 Probability theory . . . 16
  - 3.1.1 Discrete random variables . . . 17
  - 3.1.2 Continuous random variables . . . 20
- 3.2 Graph theory . . . 24
- 3.3 Probabilistic graphical models . . . 25
  - 3.3.1 Bayesian networks . . . 25
  - 3.3.2 Dynamic Bayesian networks . . . 28
- 3.4 Control . . . 30
  - 3.4.1 Linear quadratic regulator control . . . 30
  - 3.4.2 Reference tracking . . . 33
  - 3.4.3 Linear quadratic Gaussian control . . . 34
  - 3.4.4 Model predictive control . . . 35
- 3.5 Matrix identities . . . 36

4 Hidden Markov models **38**
- 4.1 Markov models . . . 38
- 4.2 Hidden Markov models . . . 39
  - 4.2.1 Filtering . . . 40
  - 4.2.2 Smoothing . . . 41
  - 4.2.3 Viterbi decoding . . . 42
  - 4.2.4 Prediction . . . 43
- 4.3 Burglar localisation problem . . . 45

5 CSTR model **48**

5.1 Qualitative analysis 49
5.2 Nonlinear model 51
5.3 Linearised models 53

II Single model systems 57

6 Inference using linear models 58
6.1 Kalman filter 59
6.2 Kalman prediction 61
6.3 Smoothing and Viterbi decoding 63
6.4 Filtering the CSTR 64

7 Inference using nonlinear models 69
7.1 Sequential Monte Carlo methods 70
7.2 Particle filter 73
7.3 Particle prediction 75
7.4 Smoothing and Viterbi decoding 75
7.5 Filtering the CSTR 76

8 Stochastic linear control 81
8.1 Unconstrained stochastic control 82
8.2 Constrained stochastic control 85
8.3 Reference tracking 91
8.4 Linear system 91
8.5 Nonlinear system 101
8.6 Conclusion 114

III Multiple model systems 116

9 Inference using linear hybrid models 117
9.1 Exact filtering 118
9.2 Rao-Blackwellised particle filter 119
9.3 Rao-Blackwellised particle prediction 120
9.4 Smoothing and Viterbi decoding 121
9.5 Filtering the CSTR 121

10 Stochastic switching linear control using linear hybrid models 130
10.1 Unconstrained switching control 133
10.1.1 Most likely model approach 133
10.1.2 Model averaging approach 140
10.2 Conclusion 140

11 Inference using nonlinear hybrid models 142

11.1 Exact filtering 143
11.2 Switching particle filter 143
11.3 Switching particle prediction 144
11.4 Smoothing and Viterbi decoding 144
11.5 Filtering the CSTR 145

12 Stochastic switching linear control using nonlinear hybrid models 150
12.1 Unconstrained switching control 151
12.2 Constrained switching control 155
12.3 Conclusion 162

13 Future work and conclusion 164
13.1 Parameter optimisation 164
13.2 Generalised graphical models 164
13.3 Filtering techniques 165
13.4 Conclusion 165

Chapter 1

Introduction

This dissertation deals with the development of predictive controllers within the framework of probabilistic graphical models, specifically dynamic Bayesian networks. Dynamic Bayesian networks are well suited to the study of stochastic processes i.e. processes where there is noise in both the state evolution and state measurements. By leveraging the natural formulation of stochastic processes within dynamic Bayesian networks we develop stochastic predictive controllers, focusing specifically on linear quadratic Gaussian (LQG) and chance constrained model predictive control (MPC).

The dissertation primarily deals with the two graphical models[1] shown in figures 1.1 and 1.2. Inference is discussed in general, but we focus on filtering and prediction because they are important for control.

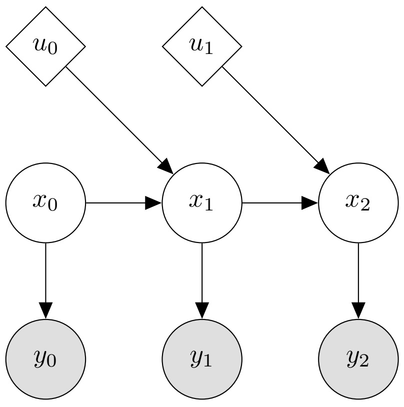

Figure 1.1: Single model probabilistic graphical model.

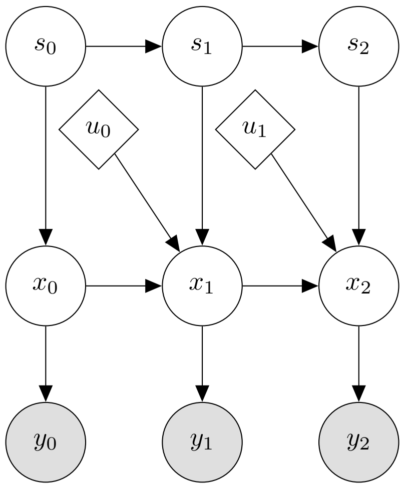

Figure 1.2: Model switching probabilistic graphical model.

The dissertation is structured in 3 parts, each composed of self contained chapters dealing

[1]Clear circular nodes represent latent variables, shaded circular nodes represent observed variables and diamond nodes are deterministic variables.

with a specific problem area. Part I contains chapters 2 to 5. The literature review, chapter 2, deals with current papers on topics related to this dissertation. Chapter 3 mainly deals with background theory found in reference materials (e.g. books). Chapter 4 deals with hidden Markov models. The goal of this chapter is to gently introduce the uninformed reader to the power of graphical models. Finally, chapter 5 introduces the continuously stirred tank reactor (CSTR) example which is used to illustrate the techniques investigated throughout the rest of the dissertation. If the reader is familiar with graphical models, predictive control and CSTR design part I may be safely skipped.

Parts II and III each follow the same pattern: a graphical model is introduced and studied after which a control scheme is implemented using the tenets of the preceding work. We detail the content and results of these two parts next.

In part II the dynamic Bayesian network, shown in figure 1.1, is investigated within the context of the Kalman filter model (linear dynamics and Gaussian noise) and the particle filter model (no assumptions about the dynamics and noise). Using the techniques endemic to the aforementioned probabilistic graphical models we show:

1. That the LQG controller reduces to the linear quadratic regulator under the assumptions of normality and linearity [2].

2. That a chance constrained MPC problem can be reduced to the standard form MPC problem (a deterministic optimisation problem with linear constraints and a quadratic objective function) under the assumptions of linearity and normality. Furthermore, since the Mahalanobis distance, a statistically important measure, is used to reduce the chance constraints to linear constraints it supports the application of the aforementioned techniques to systems which are nonlinear and non-Gaussian.

In part III the switching model filter, based on the dynamic Bayesian network shown in figure 1.2, is investigated[3]. The benefit of generalising figure 1.1 is that it allows one to infer which model, from a set of possible models, is likely to be generating the observations. This allows us to extend the stochastic MPC discussed previously to incorporate model switching. We investigate the following:

1. Using the Rao-Blackwellised particle filter as the switching model filter. In this context the resultant most likely linear model is used to move the underlying system to different regions in state space. It was found that the approach caused controller instability because the current most likely model is often not accurate enough to steer the system to the target.

2. Using a switching particle filter as the switching model filter. In this context the filter

[2]This result is not new but the derivation using probabilistic graphical models is both instructive and, more importantly, intuitive.

[3]This probabilistic graphical model uses a set of models to perform inference. The stochastic switching variables $(s_0, s_1, ...)$ are used to weight the likelihood of each model supporting the observations.

was used to identify when a process fault occurred and, based on this event, switch the model control is based upon. It was found that the algorithm successfully stabilised and regulated the nonlinear underlying system.

Perhaps most usefully, the dissertation illustrates the advantage of designing model predictive controllers from within the framework of probabilistic graphical models. While it may seem that the two fields are not related, most modern control systems perform filtering on system measurements which is a natural result of the application of probabilistic graphical models. Therefore, the motivation for this study is not purely esoteric but demonstrates a tacit relationship between the fields.

A note on implementation:
Although this dissertation spends much time investigating and developing theory there is also a significant practical aspect. All the results are supported by robust simulations. The Julia language [7] was used to implement and illustrate the effectiveness of the succeeding algorithms. Both the Ipopt [61] and Mosek [1] optimisation libraries were used in conjunction with Julia's optimisation modelling package JuMP [41]. Matplotlib [30] was used to plot the results. All the code used in this dissertation is available on Github at `https://github.com/stelmo/Stochastic-Dynamical-Control-Code`.

Part I

Literature, theory and background material

Chapter 2

Literature review

This dissertation primarily deals with stochastic model predictive control but applied within the context of probabilistic graphical models. Section 2.1 briefly covers some recent developments in stochastic model predictive control literature. Section 2.2 briefly covers control schemes, primarily based on model predictive control, where the model control is based upon is automatically adjusted based on plant measurements or manual control laws.

2.1 Stochastic model predictive control

Linear unconstrained stochastic control subject to white additive Gaussian noise is well studied in literature. The solution of the linear quadratic Gaussian (LQG) controller,

$$\begin{aligned} &\min_{\mathbf{u}} V(x_0, \mathbf{u}) = \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t \text{ with } w_t \sim \mathcal{N}(0, W), \end{aligned} \tag{2.1}$$

is one of the most fundamental results in stochastic optimal control theory [15]. The reader should note that for stochastic processes it is customary to denote the current time step x_t by x_0 within the control optimisation problem. Also, x_t is a latent variable observable only through $y_t \sim \mathcal{N}(Cx_t, V)$. Boldface is used to denote a vector of vectors over time e.g. $\mathbf{u} = (u_0, u_1, ...)$.

Using stochastic dynamic programming it is possible to show that the solution of (2.1) is merely the solution of the corresponding fully observed deterministic system, called the linear quadratic regulator (LQR), given the mean of the current state estimate x_0. A significant drawback of the LQR controller, and by extension the LQG controller, is that it is inherently linear and unconstrained.

Conventional deterministic model predictive control (MPC) is very well studied in literature [52] and can be seen as the constrained deterministic generalisation of the LQR controller,

$$\begin{aligned}
&\min_{\mathbf{u}} V(x_0, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N \\
&\text{subject to } x_{t+1} = Ax_t + Bu_t \\
&\text{and } d^T x_t + e \geq 0 \; \forall \; t = 1, 2, \ldots, N,
\end{aligned} \tag{2.2}$$

for one constraint with prediction and control horizon length N. The multiple constraint generalisation is straightforward.

A further generalisation of deterministic MPC is stochastic MPC whereby either the variables, the constraints or both have stochastic elements. In current literature the trend is to convert all the stochastic elements of the control problem into deterministic ones. This usually makes the problem more tractable from an analytic and computational point of view.

This conversion is usually achieved via two distinct approaches. In the first approach, which is also the one we employ, the probability distributions are assumed to be Gaussian and the systems linear. This allows one to greatly simplify the problem at the cost of those relatively strong assumptions. The second approach is to use a particle/sampling approach. Here the probability distributions are approximated by particles/samples and no assumptions are made of form of the distributions. It is also not necessary to assume linear dynamics. The major practical drawback of this approach is that it can quickly become computationally intractable for large problems.

This is the approach taken by [10]. They attempt to solve the stochastic MPC problem,

$$\begin{aligned}
&\min_{\mathbf{u}} \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\
&\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t \\
&\text{and } \mathbb{E}[d^T x_t + e] \geq 0 \; \forall \; t = 1, ..., N \\
&\text{and } \Pr(d^T x_t + e \geq 0) \geq p \; \forall \; t = 1, \ldots, N,
\end{aligned} \tag{2.3}$$

by approximating the current and predicted distributions with particles. Note that w_t is some stochastic variable with known parametrisation.

In their approach a particle filter is used as the state estimator; the current state distribution is approximated by particles of equal weight. An integer variable is introduced for each particle at each predicted time step. The chance constraint is then enforced by requiring that at least a certain number of particles satisfy the constraint. It is not clearly stated but this approach is only valid for particles after resampling.

By using the particle approach to model distributions it is possible to convert the stochastic optimisation problem into a deterministic one. In the case where linear dynamics are used this becomes a mixed integer linear or quadratic programming problem depending on the objective function. The chance constraint becomes an integer constraint. Their algorithm is appealing because it is not necessary to assume Gaussian distributions or linearity. However, it is possible that the algorithm can become computationally intractable due to the integer constraints which are used to approximate the chance constraints. Since it is necessary to include an integer variable for each particle at each time step in the prediction horizon the number of variables can become large. For large problems with long prediction horizons this can be problematic.

The approach taken by [40] is related to the sampling approach. They convert the stochastic chance constrained optimisation problem into a deterministic nonlinear optimisation problem. They then use a simulation approach to ensure that the chance constraints are satisfied. Their approach is numerically intensive due to the sampling and gradient estimation techniques. The approach taken by [5] uses a randomized optimisation algorithm in concert with the empirical mean of the variables. When the states approach a constraint a penalty method is used to heavily penalise the system to steer it away from the constraint. This causes the system to conservatively satisfy the constraints.

In [39] the stochastic variables are assumed to be Gaussian and the stochastic optimisation problem is transformed into a nonlinear optimisation problem. Using the Gaussian assumption they are able to ensure feasibility and constraint satisfaction albeit conservatively. In [44] the feasibility of stochastically constrained predictive control is considered. Feasibility becomes a problem when predicting under uncertainty. Since the current state estimate is not precisely known and the evolution of the system is stochastic, the certainty of the predicted states often decreases with the prediction horizon. Ensuring constraint satisfaction can become problematic in such situations because of the large predicted uncertainty in the future. In [44] an algorithm enforcing joint chance constraints and recursive feasibility is discussed using a risk allocation approach. While [55] mainly concerns stochastic parameters in the optimisation problem it is shown that chance constraints can, in theory, be rewritten as deterministic constraints if the probability distributions are known and affine constraints are used.

In [58] and [59] an ellipsoidal approximation technique is used to ensure constraint satisfaction for

$$\begin{aligned}
&\min_{\mathbf{u}} \ f(\mathbf{x}) \\
&\text{subject to } x_{t+1} = Ax_t + Bu_t \\
&\text{and } \Pr(d^T x_t + e \geq 0) \geq p \ \forall \ t = 1, \ldots, N.
\end{aligned} \tag{2.4}$$

The authors use the expected value of the stochastic variables in the objective function and system dynamics. They also only consider deterministic linear objective functions. The randomness introduced by the stochastic variables is only addressed in the chance constraint.

If one assumes that each x_t is Gaussian with sufficient statistics (μ_t, Σ_t) and dimension n, then the chance constraint can be satisfied by ensuring that the area of each ellipse $(x - \mu_t)^T \Sigma_t^{-1} (x - \mu_t) = k^2$ for $t = 1, ..., N$ is contained in the feasible region. We have that k^2

is chosen by solving the integral equation of the Chi Squared distribution

$$\frac{1}{2^{\frac{n}{2}}\Gamma(\frac{n}{2})}\int_0^{k^2} \mathcal{X}^{\frac{n}{2}-1}e^{-\frac{\mathcal{X}}{2}}d\mathcal{X} = p. \tag{2.5}$$

Using the ellipsoidal approximations the stochastic optimisation problem can transformed into a second order conic optimisation problem. The authors ensure that each ellipse is contained in the feasible region by ensuring there exists sufficient "back-off" (the distances z_1, z_2, z_3 and z_4) between the predicted ellipses and the constraints as shown in figure 2.1.

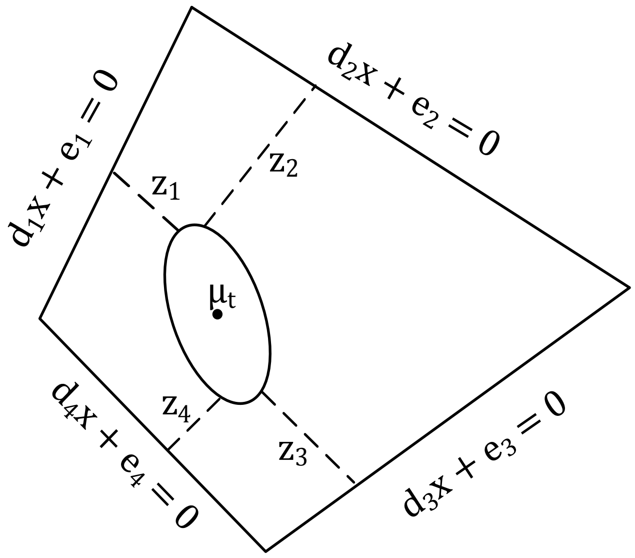

Figure 2.1: Ellipsoidal approximation to ensure chance constraint satisfaction as used by [58] and [59].

While this breakthrough is important – we build on it in our approach - the authors do not state that they are in fact using a form of the Mahalanobis distance to enforce their chance constraints. The approach of using confidence ellipsoids is further refined in [11]. The ellipsoidal approximation technique is also further investigated in [9]; they show that it is possible to reformulate joint chance constraints using univariate Gaussian distributions.

Although [64] and [65] primarily deal with univariate problems they show that if the underlying system is linear and Gaussian, it is possible to manipulate the constrained stochastic problem shown in (2.3) into a deterministic problem. Their analysis allows the stochastic objective function to be transformed into its deterministic equivalent using the properties of Gaussian integrals. This development is quite important because it allows one to directly evaluate the stochastic objective function. The constraints are handled by directly evaluating the Gaussian integral corresponding to the chance constraint in the univariate case. The authors allude to the fact that this becomes computationally intractable in higher dimensions and suggest that the approach in [58] be used. The authors also suggest a way to handle the situation where covariance matrix grows without bound in unstable systems. This is related to the feasibility problems discussed earlier.

In this dissertation the benefits gained by designing controllers within the framework of probabilistic graphical models is illustrated. This work results in the following conclusions:

1. Under the assumption of normality and linearity it is possible to convert the stochastic objective function of (2.1) into its deterministic equivalent. The analysis is closely related to the work of [64] and [65] but we show that these results are immediately obvious from within the framework of probabilistic graphical models. Thus it is possible to solve the LQG problem without resorting to stochastic dynamic programming.

2. We generalise our analysis to stochastic MPC and show that by using the statistically important metric, the Mahalanobis distance, we arrive at a technique for enforcing chance constraints which is very closely related to the approach by [58] and [59]. Under the assumption of linearity and normality we show that the constraint satisfaction is ensured. Owing to the use of the Mahalanobis distance metric we provide some theoretical support for the use of the "ellipsoidal approximation" technique if the underlying system is nonlinear or not exactly Gaussian.

3. Combining the previous results we show that it is possible to write the joint chance constrained stochastic MPC problem as a deterministic MPC problem. Additionally we show that the joint chance constraints can be written in a linear format. The entire optimisation problem can then be written in the standard form for quadratic programming optimisation. Standard deterministic MPC solution techniques can then be used to solve the stochastic problem.

4. We compare the effect different inference techniques have on the quality of the MPC. Specifically, if the underlying system is nonlinear it can be beneficial to use nonlinear particle filtering techniques as opposed to inherently linear techniques like the Kalman filter.

Lastly, measurement and system noise is ubiquitous in real life systems. Therefore most modern model predictive control systems use an inference (filtering) technique to estimate the current system state. By using a filter the control system can be seen as using a probabilistic graphical model; therefore we are not actually introducing anything exotic but rather highlighting a connection between two rich fields.

2.2 Switching model predictive control

In model predictive control the model of the plant is used to predict the future behaviour of the plant given some inputs which are optimised according to some performance criterion. Classically this model is linear and time invariant. An example of such a model is

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \\ y_t &= Cx_t + v_t. \end{aligned} \tag{2.6}$$

Since the model includes system (w_t) and measurement (v_t) noise a state estimator would typically be used to infer the state, x_t, from some observation y_t. The mean of the current state, $\mathbb{E}[x_t]$ together with the deterministic state model $x_{t+1} = Ax_t + Bu_t$ would then be used for prediction [52]. This assumption allows for the use of advanced constrained optimisation algorithms, typically quadratic programming algorithms. From a practical perspective robust and fast optimisation is crucial because it allows control inputs to be calculated on-line [43].

Unfortunately modelling errors or omissions often cause poor controller performance within the context of MPC. This is often observed as steady state offset i.e. the controller takes no more action and the system is not at the set point. In certain cases it is possible to account for plant/model mismatch or asymptotically constant disturbances by incorporating a latent disturbance model within the MPC framework. This is classically called zero offset regulation [52] and is achieved by augmenting the system model

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + B_d d_t + w_t \\ d_{t+1} &= d_t \\ y_t &= Cx_t + C_d d_t + v_t. \end{aligned} \tag{2.7}$$

By using a state estimator the latent integrating disturbance d_t can be estimated. The model used for prediction is then augmented to incorporate this disturbance $x_{t+1} = Ax_t + Bu_t + B_d\mathbb{E}[d_t]$ [37]. The problem with this approach is that it assumes that the model (A, B) is at least somewhat representative of the underlying dynamics.

Linear models (A, B) are often derived by linearising nonlinear models about a single operating point. These models are traditionally used in MPC design; a drawback with this approach is that linear models are fundamentally only accurate in a small region around the point of linearisation. The further the system moves away from the linearisation point the worse the accuracy of the model becomes because the model is no longer a good approximation of the underlying system dynamics. A possible solution to this problem is nonlinear MPC. In nonlinear MPC a full nonlinear model is used for controller design; it is hoped that this more complicated model is accurate over the entire problem domain (or a larger part thereof than the corresponding linear model). Unfortunately this approach is often computationally intensive, especially for large problems, because it inherently requires nonlinear, usually non-convex, optimisation. This is a subject of much current research [20].

Another approach is to approximate a potentially complex nonlinear system by a set of linear functions which are valid in certain ranges. An early attempt at this idea [6] integrated logical rules for switching between different system dynamics and constraints. Using that approach a set of logical rules were integrated into the optimisation problem to yield, in the setting of standard MPC, a mixed integer quadratic program with linear constraints. This approach is called mixed logical dynamical modelling (MLD) in literature; the system dynamics are then specified by

$$x_{t+1} = \sum_{i=1}^{I} \delta_i A_i x_t + \sum_{i=1}^{I} \delta_i B_i u_t \tag{2.8}$$

where $i = 1, 2, ..., I$ are the indices of the models approximating the underlying problem.

The binary variable $\delta_i \in [0, 1]$ selects which linear model to use at each step in the prediction horizon within the framework of MPC. Constraints on δ_i allow the optimisation algorithm to switch between models based on its position in the state space. The drawbacks with this approach is that the "IF-THEN-ELSE" rules need to be fully specified before hand and the mixed integer optimisation problem can become computationally intractable.

The work by [23] and [54] elaborate on this approach. Both papers deal with the control of continuously stirred tank reactors (CSTRs) throughout different operating regimes. CSTRs are a good case study because they often have multiple steady states, constraints and can be quite nonlinear. The approach of [23] and [54] is to linearise the underlying nonlinear CSTR model about different operating points and use those models to approximate the true nonlinear dynamics. Computational difficulties are reported because the complexity of the mixed integer quadratic problem scales exponentially with the number of variables.

The approach by [50] is similar except that they use ellipsoidal regions to develop a multiple model MPC. Again the approach by [36] is similar except that they attempt to reduce the complexity of the mixed integer quadratic problem by providing guidelines on selecting the number of linear models used. More linear models leads to better control but the optimisation problem becomes correspondingly more difficult. Finally, [49] also investigates hybrid systems but uses Bayes' theorem to assign weights to the different models as opposed to only having one model active in each region. In their approach δ_i is a continuous variable in the range $[0, 1]$ and $\sum_{i=1}^{I} \delta_i = 1$. While their approach is also computationally intensive, a nonlinear optimisation problem needs to be solved, an effort is made to take advantage of the problem structure to attenuate this problem.

Loosely related to the idea of model switching is model based fault detection. In [31] it is found that almost 70% of the reviewed papers dealing with fault detection use observer or parameter estimation methods. The basic idea behind this approach is to estimate the system outputs using an observer and to compare this to some model of the system. The difference, often called the residual, is then used to estimate the probability of a fault in the process [25]. The approach followed in [62] combines elements of model switching and state observations. They use a filter to construct a residual generator which is used to evaluate whether or not the system has a fault. The filter can switch between the different linear system models based on the current regime of the system.

Related to this class of model based fault detection algorithms is the switching model filter. The corresponding probabilistic graphical model is shown in figure 2.2. Classically these types of models are used to infer the current state estimate (called filtering) given a set of models. Intuitively, the model which best describes the current observation (y_t) is assigned the most weight in the state estimate. This allows for the modelling of nonlinear and multi-modal distributions by combining models based on the inferred switching variable (s) [47]. These models may be used for both filtering and event detection (see [60] for an example) which, in

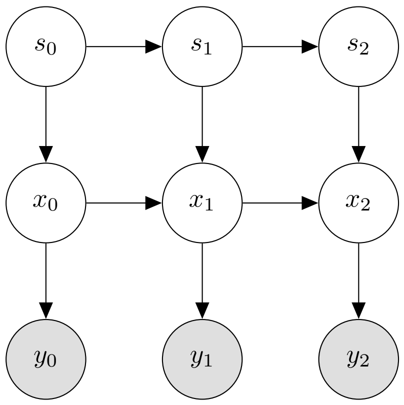

Figure 2.2: Switching filter graphical model.

the setting of control, can be interpreted as fault detection.

In this dissertation we also illustrate the potential control benefits gained by designing predictive controllers using the switching model probabilistic graphical model of figure 2.2. We investigate and show the following:

1. Using switching model filters it is possible to accurately estimate the current state as well as estimate the probability of each model supporting the current observation. The state estimate can then be used within the controller framework alluded to in section 2.1. Using this approach it is also possible to change the model control is based upon. The linear model which best describes the current observations is used for control purposes. In this way we gain the benefit of model switching but do not incur the significant computational cost when it is incorporated into the optimisation problem as is the trend in current literature. We call this approach the switching controller algorithm.

2. We investigate using the Rao-Blackwellised particle filter as the switching model filter. In this context the resultant most likely linear model is used to move the underlying system to different regions in state space. It was found that the approach caused controller instability because the current most likely model is often not accurate enough to steer the system to the target.

3. We also investigate using the switching particle filter as the switching model filter. In this context the filter was used to identify when a process fault occurred and, based on this event, switch the model control is based upon. It was found that the algorithm successfully stabilised and regulated the nonlinear underlying system. Furthermore, since the controllers of section 2.1 were used the control scheme was computationally efficient.

Even more so than section 2.1, the approach used here relies on the application of probabilistic graphical models to controller design.

Chapter 3

Background theory

This chapter is composed of five sections which introduce the main concepts and results used throughout the rest of the dissertation. Section 3.1 introduces probability theory. Section 3.2 very briefly introduces some useful nomenclature from graph theory. These two sections serve as an entry point for section 3.3 which deals with probabilistic graphical models. Section 3.4 deals with control theory and section 3.5 introduces an important result from matrix linear algebra.

It might appear as though sections 3.1 to 3.3 and section 3.4 are not related to each other. However, the foundational theory introduced here is expanded upon later and the relationship then becomes clear.

3.1 Probability theory

The calculus of probability theory was developed by Fermat and Pascal in order to better understand the problems introduced by uncertainty in gambling. From this dubious genesis a rich and incredibly powerful field has developed. We start our brief introduction of probability theory by restating Kolmogorov's three probability axioms - these axioms underpin the entire theory of probability [34].

Let the set Ω be the universe of possible events, also called the event space; that is, if we are uncertain about which of a number of possibilities are true then we let Ω represent all of them collectively. Let P be some real valued function which satisfies the three axioms stated below.

Axiom 3.1. $P(\Omega) = 1$. The probability of any event in Ω occurring is 1.

Axiom 3.2. $\forall \alpha \in \Omega,\ P(\alpha) \geq 0$. The probability of any one (or set of) event(s) in Ω occurring is non-negative.

Axiom 3.3. $\forall \alpha, \beta \in \Omega,$ if $\alpha \cap \beta = \varnothing$ then $P(\alpha \cup \beta) = P(\alpha) + P(\beta)$. The probability of two mutually disjoint sets of events in Ω occurring is equal to the sum of their probabilities.

A function P which satisfies these three axioms is known as a probability function. Based on these three axioms the theory can be extended to theorem 3.1 [34].

Theorem 3.1. $\forall \alpha, \beta \in \Omega, P(\alpha \cup \beta) = P(\alpha) + P(\beta) + P(\alpha \cap \beta)$. The probability of two events occurring in Ω is equal to the sum of their probabilities less the probability of both occurring simultaneously.

3.1.1 Discrete random variables

We now make precise what we mean by random variables: a random variable is a non-deterministic variable which is characterised by some uncertainty in its measurement. Semantically we indicate a specific value taken on by the random variable X as $X = x$ or just denote it x. Thus, the function $P(X = x) = P(x) \in \mathbb{R}$ indicates the probability of event x occurring with respect to the random variable X. We denote $P(X)$ as the probability function of the random variable X. Thus, for the discrete random variable X we have that $P(X) = (P(x_1), P(x_2), ..., P(x_n))^T$ where $x_i \in X$ for $i = 1, 2, ..., n$ and $\sum_i P(x_i) = 1$. We defer the study of the continuous case until later.

Before we proceed let us briefly discuss how we can interpret the function P for any random variable X. If $P(X = x) = 1$ we are certain of event x occurring, i.e. X will only take on the value x. If $P(x) = 0$ we are certain that event x will not occur, i.e. X will never take on the value x. Thus our certainty of event x occurring is reflected by the magnitude of $P(x)$. Attempting to make the statement "our certainty of event x occurring" more precise leads us to two different physical interpretations of $P(x)$. The first is the frequentist interpretation: to the frequentist a probability is a long term average of the observations of event x occurring in the event space. While this interpretation is satisfying if one deals with something which is easily measured e.g. the probability of a fair die rolling a 6, it fails to explain statements like: "the probability of it raining tomorrow is 50%". The reason the last statement is problematic is because the time span is ill defined. If we rather understand probabilities to mean subjective degrees of belief in event x occurring this is no longer a problem. To ensure that these subjective beliefs are rational can be problematic. One way to ensure this is by requiring that if the probabilities were used in a betting game it is impossible to exploit them to one's advantage (or disadvantage). If this is possible then there is no difference between the interpretations described above [34].

We will deal extensively with joint and marginal probability distributions. Consider the random variables X and Y. The marginal probability distribution of X is the function $P(X)$ and describes the probabilities of events involving only the variable X. The joint probability distribution of X and Y is the function $P(X, Y) = P(X \cap Y)$ and describes the intersection (and) of the probability space of X and Y. We introduce, without proof, theorem 3.2.

Theorem 3.2. Marginalisation By marginalising out X we mean we sum out X from the joint distribution $P(Y) = \sum_x P(x, Y)$. This extends to higher dimensions.

We can reduce any joint distribution to a marginal one by summing (or integrating in the case of continuous random variables) out the appropriate variable.

It is now necessary to define what we mean by conditional probability. Definition 3.1 makes precise how the knowledge that event y has occurred alters our view of event x occurring.

Definition 3.1. Conditional probability $P(X|Y) = \frac{P(X \cap Y)}{P(Y)}$

Note that if for some $y \in Y$ we have $P(Y = y) = 0$ then definition 3.1 is undefined. Additionally, the function $P(\cdot|Y)$ is a probability function. We next define what we mean by a positive probability distribution in definition 3.2.

Definition 3.2. A probability distribution is positive if $P(x) > 0 \; \forall \; x \in X$.

Clearly undefined conditional probabilities are not a problem in the setting of positive probability distributions. We also define the notion of independence, also sometimes called marginal independence, in definition 3.3.

As before, let X, Y and Z be random variables. Intuitively X and Y are independent if the outcome of X does not influence the outcome of Y. It can be shown that independence is a symmetric property [34].

Definition 3.3. Independence $X \perp\!\!\!\perp Y \equiv P(X|Y) = P(X)$

Generalising the concept of independence we define conditional independence by definition 3.4. Again this definition is symmetric [34].

Definition 3.4. Conditional independence $X \perp\!\!\!\perp Y|Z \equiv P(X|Y, Z) = P(X|Z)$

Intuitively, if X is conditionally independent of Y given Z then by observing Z one gains nothing by observing Y. Clearly if $Z = \emptyset$ we have (marginal) independence. We also introduce theorem 3.3 which naturally leads us to the formulation of Bayes' theorem (using definition 3.1) as shown in theorem 3.4.

Theorem 3.3. Chain rule Given the random variables X_1 and X_2 we have $P(X_1, X_2) = P(X_1)P(X_2|X_1)$. The generalisation to an arbitrary number of random variables is straightforward.

Theorem 3.4. Bayes' theorem $P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$

Under the Bayesian interpretation of theorem 3.4 we see that the posterior probability of some hypothesis X given some evidence Y being true is just the likelihood $P(Y|X)$ of the hypothesis supporting the evidence multiplied by the prior probability of the hypothesis $P(X)$ normalised by the prior of the evidence $P(Y)$. It is also convenient to notice that $P(Y)$ is a normalising constant and thus $P(X|Y) \propto P(Y|X)P(X)$.

To fully describe a system of random variables it is only necessary to know the joint distribution $P(X_1, X_2, ..., X_n)$. Given the joint probability distribution inference (reasoning about the variables under uncertainty) may be performed. Common probabilistic queries involve computing posterior beliefs $P(X|Y = y)$ i.e. the probability function of X given we have some information about Y. Other queries involve find the most probable explanation (called a MAP query) of some evidence i.e. finding X which maximises $P(X, Y = y)$. More on this later.

Example of Bayes' theorem in action

This section will attempt to develop some intuition behind theorem 3.4. We quote an excerpt from an article in the Economist [24] and illustrate the use of Bayes' theorem in a canonical medical example [35].

"The essence of the Bayesian approach is to provide a mathematical rule explaining how you should change your existing beliefs in the light of new evidence. In other words, it allows scientists to combine new data with their existing knowledge or expertise. The canonical example is to imagine that a precocious newborn observes his first sunset, and wonders whether the sun will rise again or not. He assigns equal prior probabilities to both possible outcomes, and represents this by placing one white and one black marble into a bag. The following day, when the sun rises, the child places another white marble in the bag. The probability that a marble plucked randomly from the bag will be white (ie, the child's degree of belief in future sunrises) has thus gone from a half to two-thirds. After sunrise the next day, the child adds another white marble, and the probability (and thus the degree of belief) goes from two-thirds to three-quarters. And so on. Gradually, the initial belief that the sun is just as likely as not to rise each morning is modified to become a near-certainty that the sun will always rise."

Now for the canonical medical example. Suppose you get tested for a certain disease. You know the disease affects 1 in 100 people. You also know that the false positive rate for the test is 20% and the false negative rate for the test is is 10%. Your test comes back positive. What are the chances of you having the disease given this information?

The information may be summarised as shown below. Let D be a binary random variable indicating the presence of the disease and $\neg D$ indicates the absence. Let T be a binary random variable indicating a positive test and $\neg T$ indicates a negative test.

1. The prior of the disease is $P(D) = 0.01$.

2. False positive rate $P(T|\neg D) = 0.2 \implies P(\neg T|\neg D) = 0.8$.

3. False negative rate $P(\neg T|D) = 0.1 \implies P(T|D) = 0.9$.

A naive approach would conclude that since $P(T|D) = 0.9$ you are 90% likely to have the disease. However, using Bayesian inference/reasoning we have:

$$
\begin{aligned}
P(D|T) &= \frac{P(T|D)P(D)}{P(T)} \\
&= \frac{P(T|D)P(D)}{\sum_D P(D,T)} \\
&= \frac{P(T|D)P(D)}{\sum_D P(D)P(T|D)} \\
&= \frac{0.9 \times 0.01}{0.01 \times 0.99 + 0.99 \times 0.2} \\
&\approx 0.04
\end{aligned}
$$

Clearly there is a big difference between the naive approach and the Bayesian (correct) approach. The power of Bayesian inference lies in the ability to reverse causal reasoning. That is, we know that the disease causes the test to be positive, $P(T|D)$, but we would like to reverse this reasoning to infer $P(D|T)$. This is immensely powerful as we shall soon discover.

3.1.2 Continuous random variables

So far in our discussion we have implicitly only used discrete random variables; that is, our probability space consisted out of a finite number of events or states. However, it is also necessary to make precise what we mean by a continuous random variable. A continuous random variable is characterised by a density function p which assigns a weight to each possible value of the variable. Intuitively this weight is *related* to the probability of that value occurring[1]. Although the density function is itself not a probability function, if it satisfies $p(x) \geq 0 \ \forall x \in X$ and $\int p(x)\mathrm{d}x = 1$, where we have implicitly integrated over the domain of p, then it can be used to generate one. The cumulative probability function $P(X \leq a) = \int_{-\infty}^{a} p(x)\mathrm{d}x$ is one such example[2].

Arguably the most well known continuous probability density distribution is the Gaussian or normal distribution. The Gaussian distribution arises naturally from a variety of different contexts and settings. For example, the central limit theorem, together with some mild assumptions, tells us that the sum of a set of N random variables is itself a random variable and in the limit can be described by a Gaussian distribution [8]. The Gaussian is regularly used because it has some very appealing analytical properties (and also often because it is physically meaningful) which we will investigate in some depth.

Since the probability of a specific value is not meaningful in the setting of continuous probability functions we abuse our notation and interchangeably denote the random variable X by x. We also do not indicate vector quantities in boldface; it can be assumed that all numbers are vectors unless otherwise noted. We will concern ourselves mostly with vector quantities and it will be obvious when we deal with scalar valued variables.

[1]Please note that strictly speaking $P(x) = 0$ for a specific point x in the domain of p. Technically it is correct to say that the probability of $P(x \in [a,b]) = \int_a^b p(y)\mathrm{d}y$; thus, if we want the probability of x occurring we could just make $[a,b]$ small to get some approximation.

[2]We have assumed that the domain of X is the entire real line.

Definition 3.5. Gaussian distribution The univariate Gaussian or normal distribution of a random variable x is defined by

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right). \tag{3.1}$$

We call μ the mean and σ^2 the variance of the distribution. The multivariate Gaussian distribution is defined by

$$\mathcal{N}(x|\mu,\Sigma) = \frac{1}{\sqrt{2\pi^{\frac{D}{2}}}} \frac{1}{|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{3.2}$$

where μ is a D dimensional mean vector and Σ is a $D \times D$ dimensional covariance matrix. Note that we often use the inverse of the covariance matrix, called the precision matrix and define it $\Lambda \equiv \Sigma^{-1}$.

It is also appropriate to define some functions which apply equally well to the discrete case as to the continuous case (just replace the integration with summation in the setting of discrete random variables). We define the expectation (or mean or average) in definition 3.6, the variance in definition 3.7 and the covariance in definition 3.8.

Definition 3.6. Expectation The average value of some integrable function f under the probability distribution p is denoted $\mathbb{E}[f] = \int p(x)f(x)dx$.

We have that $\mathbb{E}[x] = \mu$ if x is a Gaussian random variable.

Definition 3.7. Variance The variance of f is defined by $\text{var}[f] = \mathbb{E}[(f - \mathbb{E}[f])^2]$ and provides a measure of how much variability there is in f around its mean value $\mathbb{E}[f]$.

By expanding out the square we have the familiar formula $\text{var}[f] = \mathbb{E}[f^2] - \mathbb{E}[f]^2$. Also note that for a univariate Gaussian random variable x we have $\text{var}[x] = \sigma^2$.

Definition 3.8. Covariance For two random variables x, y (which may be column vectors) we define the covariance matrix $\text{cov}[x,y] = \mathbb{E}[xy^T] - \mathbb{E}[x]\mathbb{E}[y]$.

Note that $\text{cov}[x,x] = \text{cov}[x] = \text{var}[x]$. Covariance is a measure of how much two random variables vary together. If x is a D dimensional Gaussian random variable then $\text{cov}[x] = \Sigma$ as defined in definition 3.5.

The identities in theorem 3.5 will be useful in later sections. We refer the reader to [17] for justification.

Theorem 3.5. Gaussian expected value identities Suppose there exist constants $c \in \mathbb{R}^n$ and $C \in \mathbb{R}^{n\times n}$ and X is a normal random variable with statistics (μ, Σ). Then the following identities hold:

1. $\mathbb{E}[c^T X] = c^T\mu$
2. $\mathbb{E}[CX + c] = C\mu + c$

3. $\mathbb{E}[X^T CX] = \mathrm{tr}(C\Sigma) + \mu^T C\mu$

Now we are in a position to perform some manipulations assuming we are using Gaussian random variables. We state without proof theorem 3.6.

Theorem 3.6. Partitioned joint gaussians Given a Gaussian distribution $\mathcal{N}(x|\mu,\Sigma)$ with $\Lambda \equiv \Sigma^{-1}$ and $x = \begin{pmatrix} x_a \\ x_b \end{pmatrix}$, $\mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix}$, $\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$ and $\Lambda = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}$ then we have the conditional distribution

$$\begin{aligned} &p(x_a|x_b) = \mathcal{N}(x_a|\mu_{a|b}, \Lambda_{aa}^{-1}) \\ &\text{with } \mu_{a|b} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b) \end{aligned} \tag{3.3}$$

and the marginal distribution

$$p(x_a) = \mathcal{N}(x_a|\mu_a, \Sigma_{aa}). \tag{3.4}$$

It is often easier to work with the precision matrix when dealing with conditional distributions.

Next we state and then prove theorem 3.7 which we will use extensively. The proof for theorem 3.6 uses the same techniques and can be found in [8].

Theorem 3.7. Bayes' theorem for linear gaussian models Suppose we have a marginal Gaussian distribution for x and a conditional Gaussian distribution for y:

$$\begin{aligned} p(x) &= \mathcal{N}(x|\mu, \Lambda^{-1}) \\ p(y|x) &= \mathcal{N}(y|Ax + b, L^{-1}). \end{aligned} \tag{3.5}$$

Then the marginal distribution for y is

$$p(y) = \mathcal{N}(y|A\mu + b, L^{-1} + A\Lambda^{-1}A^T) \tag{3.6}$$

and the conditional distribution for x given y is

$$\begin{aligned} &p(x|y) = \mathcal{N}(x|\Sigma(A^T L(y-b) + \Lambda\mu), \Sigma) \\ &\text{with } \Sigma = (\Lambda + A^T LA)^{-1}. \end{aligned} \tag{3.7}$$

where b is a known vector or function of some deterministic variable.

Proof. We begin our proof by noticing that for a general Gaussian $\mathcal{N}(\gamma|\alpha,\beta)$ we can write the exponent as

$$-\frac{1}{2}(\gamma-\alpha)^T \beta^{-1}(\gamma-\alpha) = -\frac{1}{2}\gamma^T\beta^{-1}\gamma + \gamma^T\beta^{-1}\alpha + \text{const} \tag{3.8}$$

where const is some real number which does not depend on γ. It is known that Gaussian distributions are closed under multiplication, i.e. if one multiplies two Gaussian distributions the product is still a Gaussian distribution (of a higher dimension) [8]. To find the joint distribution we let $z = \begin{pmatrix} x \\ y \end{pmatrix}$ and consider the logarithm of the joint

$$\begin{aligned} \log(z) &= \log(p(x)) + \log(p(y|x)) \\ &= -\frac{1}{2}(x-\mu)^T\Lambda(x-\mu) - \frac{1}{2}(y - Ax - b)^T L(y - Ax - b) + \text{const} \end{aligned} \tag{3.9}$$

Here const denotes constant terms which are independent of x and y. Now we make use of (3.8) to find the mean and covariance of z. Continuing, we consider only the second order terms when (3.9) is expanded

$$\begin{aligned}
&-\frac{1}{2}x^T(\Lambda + A^TLA)x - \frac{1}{2}y^TLy + \frac{1}{2}y^TLAx + \frac{1}{2}x^TA^TLy \\
&= -\frac{1}{2}\begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} \Lambda + A^TLA & -A^TL \\ -LA & L \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \qquad (3.10) \\
&= -\frac{1}{2}z^TRz.
\end{aligned}$$

From this we immediately have the precision of z: the matrix R; we also use a matrix inversion formula found in [8] to find the covariance

$$\text{cov}[z] = R^{-1} = \begin{pmatrix} \Lambda^{-1} & \Lambda^{-1}A^T \\ A\Lambda^{-1} & L^{-1} + A\Lambda^{-1}A^T \end{pmatrix}. \qquad (3.11)$$

We now proceed in exactly the same way to find mean of z. By expanding 3.9 and only considering the first order terms in x and y we have

$$x^T\Lambda\mu - x^TA^TLb + y^TLb = \begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} \Lambda\mu - A^TLb \\ Lb \end{pmatrix}. \qquad (3.12)$$

Again, by making use of (3.8) and the fact that the covariance of z is R^{-1} it is possible to show that $\mathbb{E}[z] = \begin{pmatrix} \mu \\ A\mu + b \end{pmatrix}$ as shown in [8]. By using theorem 3.6 we immediately have the marginal and conditional distributions as required. □

We also introduce a useful metric for measuring the similarity between two distributions in definition 3.9.

Definition 3.9. Kullback-Leibler divergence Consider some unknown distribution $p(x)$ and suppose we have modelled this distribution by $q(x)$. Kullback-Leibler divergence, also known as relative entropy, is defined $\text{KL}(p||q) = -\int p(x)\ln\left(\frac{q(x)}{p(x)}\right) dx$ and measures the additional amount of information, in *nats*, needed to specify the value of x [8].

Kullback-Leibler divergence can be used to measure the dissimilarity between two distributions. If the measure is zero the distributions are identical; care needs to be taken when using Kullback-Leibler divergence because the measure is not symmetric. We introduce theorem 3.8 to measure the dissimilarity between a known distribution and a sampled approximation thereof. See [8] for the motivation.

Theorem 3.8. Kullback-Leibler sample divergence Suppose we observe a finite set of points x_n for $n = 1, 2, ..., N$ drawn from $p(x)$. Furthermore, suppose we would like to measure the information loss when p is approximated by q. We can measure this by $\text{KL}(p||q) \approx \frac{1}{N}\sum_{n=1}^{N}(-\ln(q(x_n)) + \ln(p(x_n)))$. This measurement is bounded below by zero. If, as $N \to \infty$, the information loss is zero p and q are functionally equivalent.

We also briefly introduce the Mahalanobis distance in definition 3.10.

Definition 3.10. Mahalanobis distance The Mahalanobis distance between x and a reference point $y \in \mathbb{R}^n$ given a covariance matrix $S \in \mathbb{R}^{n \times n}$, is defined by $D_M(x|y, S) = \sqrt{(x-y)^T S^{-1}(x-y)}$.

The Mahalanobis distance is a statistical distance metric which reduces to the Euclidean distance metric if $S = I$. It is found in the exponent of the Gaussian distribution density function and can be used to measure the “closeness” of points between distributions with a common covariance matrix. We will study it in more detail later.

3.2 Graph theory

A graph, G, is a data structure consisting of a set of nodes χ and edges ξ. A pair of nodes $X_i, X_j \in \chi$ can be connected by an edge. We will only consider directed graphs in this dissertation. This implies that every edge in ξ has a direction associated between the two nodes it connects i.e. $X_i \rightarrow X_j$ if there is an edge from X_i to X_j.

We now define some basic concepts which we will rely upon to further describe the types of graphs we will consider.

Definition 3.11. Directed path We say that the nodes $X_1, X_2, X_3, ..., X_n \in \chi$ form a directed path if $X_i \rightarrow X_{i+1}$ for $1 \leq i \leq n-1$.

Definition 3.12. Directed cycle A directed cycle is a non-singleton directed path which starts and ends at the same node.

Definition 3.13. Directed acyclic graph (DAG) A graph G is a DAG if it is directed and has no directed cycles.

In this dissertation we will only concern ourselves with DAGs. Figure 3.1 is an example of a DAG.

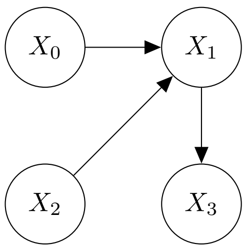

Figure 3.1: Example of a directed acyclic graph.

Next we define some nomenclature to further describe the nodes of a graph G.

Definition 3.14. Parents We say that the set of nodes $\kappa \subset \chi$ are the parents of node X_i if, for each node in κ, there exists an edge going to X_i.

Definition 3.15. Children We say that the set of nodes $\tau \subset \chi$ are the children of node X_i if, for each node in τ, there exists an edge going from X_i to that node.

Definition 3.16. Descendants We say that the set of nodes $\gamma \subset \chi$ are the descendants of node X_i if, for each node in γ, there exists a directed path from X_i to that node.

We also briefly define a structured approach to encoding a graph.

Definition 3.17. Adjacency matrix For a graph G with n nodes, the adjacency matrix A is an $n \times n$ matrix where $A_{ij} = 1$ if there is an edge from node i to node j and $A_{ij} = 0$ otherwise.

The adjacency matrix A for figure 3.1 is shown below:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

A detailed analysis of graph theory may be found in [19].

3.3 Probabilistic graphical models

Probabilistic graphical models are the union between probability theory and graph theory. Consider why, in general, it is infeasible to determine an arbitrary joint probability distribution. Suppose you have a set of n binary random variables and wish to determine their joint. This equates to finding $P(X_1, X_2, ..., X_n)$. To fully specify this model we would need to find and store $2^n - 1$ probabilities. For even moderately big n this is impractical, and this was for the simple case of a binary valued random variable. Clearly we require a more efficient way to represent the joint probability distribution.

3.3.1 Bayesian networks

A Bayesian network is a representation of the joint probability distribution of a set of random variables parametrised by:

1. A graph depicting local independence relationships.

2. Conditional probability distributions.

The fundamental assumption behind Bayesian networks, and more generally probabilistic graphical models, is that there is a useful underlying structure to the problem being modelled which can be captured by the Bayesian network. This underlying structure is available via conditional independence relationships between the variables.

Suppose P is the joint distribution of some set of random variables we require to do inference on.

Definition 3.18. I-Map The I-Map of P, denoted by $\mathcal{I}(P)$, is the set of independence assertions of the form $X \perp\!\!\!\perp Y|Z$ which hold over P.

Let G be a Bayesian network graph over the random variables $X_1, X_2, ..., X_n$ where each random variable is a node. We say that the distribution P factorises over the same space if P can be expressed as the product defined by the chain rule for Bayesian networks.

Definition 3.19. Chain Rule for Bayesian Networks The chain rule for Bayesian networks specifies that the joint factorises according to $P(X_1, ..., X_n) = \Pi_{i=1}^{n} P(X_i|\text{Parents}(X_i))$.

Each of the individual factors of P, as factorised by the chain rule for Bayesian networks, represents the conditional probability distributions required to parametrise the Bayesian network. It can be shown that a Bayesian network graph G over which P factorises is not unique. However, if the graph explicitly models the causality inherent in the system being modelled the representation is often much sparser [34]. A Bayesian network is then defined as the tuple (G, P) such that the joint P factorises over the graph G. We state without proof theorem 3.9.

Theorem 3.9. Let G be a Bayesian network graph over a set of random variables χ and let P be a joint distribution over the same space. If P factorises according to G then G is an I-Map for P. Conversely, if G is an I-Map for P then P factorises according to G.

Thus, the conditional independences imply factorisation of P. Conversely, factorisation according to G implies the associated conditional independences.

To illustrate computational benefit of using Bayesian networks, consider again our simple system of n binary random variables $X_1, X_2, ..., X_n$. Suppose the Bayesian network in figure 3.2 models the system.

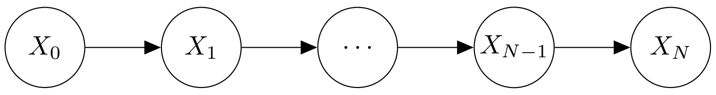

Figure 3.2: Example of a simple Bayesian network.

Without knowing any structure $2^n - 1$ parameters were needed to specify the joint. However, using the chain rule for Bayesian networks we can factorise the joint $P(X_1, ..., X_n) = P(X_1)P(X_2|X_1)...P(X_n|X_{n-1})$. This implies that we only require $2n - 1$ parameters. From a modelling perspective this is a significant gain.

The primary reason we would want to have a model of the joint distribution of a set of random variables is to reason with. To achieve this we invariably manipulate the joint distribution by either some form of marginalisation or optimisation. To make inference computationally tractable it is desirable to leverage the independence assertions implied by the network graph. To this end we expand on the independence assertions implied by the graph. Recall theorem 3.9: since we have that the joint factorises over the graph we also have that any independence assertions implied by the graph's connectivity also apply to the joint.

We introduce the concept of d-separation as a method of determining whether a set of nodes X are conditionally independent of another set Y given the set E. Firstly we generalise the concept of a directed path to an undirected path between sets of variables.

Definition 3.20. Undirected path An undirected path between two sets of nodes X and Y is any sequence of nodes between a member of X and a member of Y such that every adjacent pair of nodes is connected by an edge regardless of direction and no node appears twice.

Definition 3.21. Blocked path A path is blocked, given a set of nodes $\mathbf{E}$, if there is a node Z on the path for which at least one of the three conditions holds:

1. Z is in E and Z has one edge leading into it from the path and one edge leading out of it on the path.

2. Z is in E and Z has both edges leading out of it from the path.

3. Neither Z nor any descendant of Z is in E and both path edges lead into Z.

Definition 3.22. D-separation A set of nodes E d-separates two other sets of nodes X and Y if every path from a node in X to a node in Y is blocked given E.

To shed some more light on d-separation consider figure 3.3. The first diagram depicts the first blocked condition, i.e. a causal chain. Node E blocks relevance of X to Y. The second diagram illustrates the second blocked condition, i.e. a common cause. Node E blocks X from being relevant to Y. Finally, the third diagram illustrates the third blocked condition or, more aptly, illustrates how lack of knowledge of the nodes in the path from X to Y implies that they are conditionally independent [35].

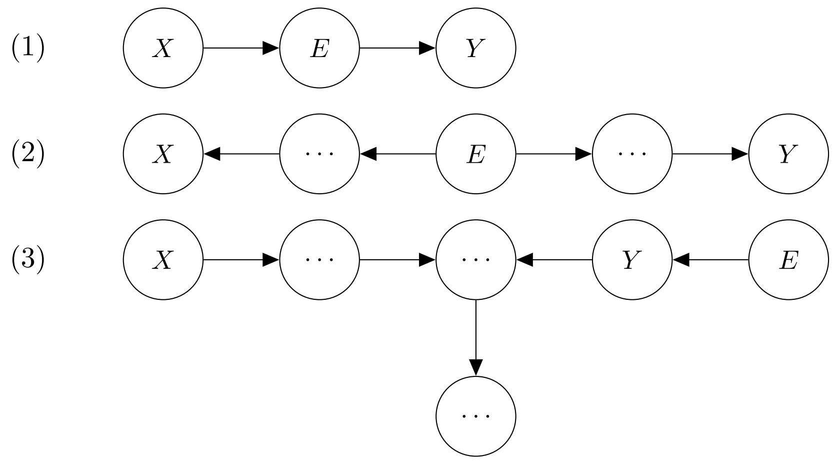

Figure 3.3: Examples of d-separation.

Using d-separation we can efficiently reason about the conditional independences implied by the graph and the observed variables (E). This becomes incredibly useful when one attempts to apply inference techniques because it can simplify the joint calculations significantly. More on this later.

Bayesian networks are commonly used to model situations which are not time dependent. We will primarily restrict ourselves to time series modelling in this dissertation. As such we will not delve deeper into static Bayesian network theory.

3.3.2 Dynamic Bayesian networks

Dynamical Bayesian networks generalise the conventional static Bayesian networks of the previous section. Dynamic, or temporal, Bayesian networks model systems which evolve with time. Since sequential, or temporal, data is abundant in most engineering applications we will primarily concern ourselves with such models. Notationally we denote a time dependent vector by $x_{1:t} = x_1, x_2, ..., x_t$, for example the joint $P(x_{1:3}) = P(x_1, x_2, x_3)$.

There are two important classes of analysis one may perform on sequential data using graphical models. On-line analysis, including prediction and filtering and off-line analysis, including smoothing and the most probable explanation (sometimes called Viterbi decoding). In both cases we are generally interested in learning something about a set of hidden state variables by performing inference on some set of observed variables.

A state space model assumes that there is some underlying hidden state (x_t) of the world which generates observations (y_t). These hidden states may evolve with time and may be functions of some inputs (u_t). The hidden states and observations are most generally assumed to be random variables. Any state space model is fully parametrised by the following information:

1. A prior probability distribution over the states: $P(x_0)$
2. A state transition function: $P(x_t|x_{0:t-1}, u_{0:t-1})$
3. An observation function: $P(y_t|x_{0:t}, u_{0:t-1})$

For the purposes of this dissertation we will assume that the state space model is known. If this model is not known machine learning techniques may be used find these models [47]. To simplify notation we will sometimes omit the dependence of the probability functions on the inputs $u_{0:t}$.

We will assume that all the systems we model satisfy the first order Markov assumption.

Definition 3.23. N$^{\text{th}}$-order Markov assumption A system satisfies the N$^{\text{th}}$ Markov assumption if $P(x_t|x_{0:t}) = P(x_t|x_{t-n:t-1})$. For example, a first order Markov system satisfies $P(x_t|x_{0:t}) = P(x_t|x_{t-1})$. Similarly with the observation function.

This is not as restrictive as it may seem at first. It is always possible to transform an N$^{\text{th}}$-order

Markov system into a first order Markov system by modifying the state space [47]. We also assume that the state and observation functions remain the same for all time i.e. they are time invariant or homogeneous or stationary.

Intuitively, a state space model is a model of how x_t generates or causes y_t and x_{t+1}. The goal of inference is to invert this mapping. The four types of inference we will consider in this dissertation are:

1. Filtering: we attempt to infer $P(x_t|y_{0:t})$, i.e. we attempt to estimate the current state given all past observations.

2. Smoothing: we attempt to infer $P(x_{t-m}|y_{0:t})$ with $m > 0$, i.e. we attempt to estimate some past state given all the past and future observations. A more apt description of this process is applying hindsight to state estimation.

3. Prediction: we attempt to infer either $P(x_{t+m}|y_{0:t})$ or $P(y_{t+m}|y_{0:t})$ with $m > 0$, i.e. we attempt to estimate the future hidden states or observations given all the past observations.

4. Viterbi decoding: we attempt to perform $x^*_{1:t} = \arg\max_{x_{1:t}} P(x_{1:t}|y_{1:t})$, i.e. we attempt to infer the most likely sequence of states which best explain the observations.

It is customary to denote hidden (latent) variables by a clear node, observed (visible) variables by a shaded node and deterministic variables by a diamond shaped node. Additionally, it is also customary to separate the input, state and observation variables from each other: $z_t = (u_t, x_t, y_t)$.

To fully specify a dynamic Bayesian network we require the pair $(B_0, B_{\rightarrow})$. The Bayesian network B_0 defines the prior over the random variables being modelled and $B_{\rightarrow}$ defines the transition and observation functions by means of a Bayesian network graph, typically over two time slices, see figure 3.4 for an example. This Bayesian network graph may be factorised according to the Bayesian network chain rule such that at each time slice

$$P(z_t|z_{t-1}) = \Pi_{i=1}^{N} P(z_t^i|\text{Parents}(z_t^i)). \tag{3.13}$$

A dynamic Bayesian network may be unrolled (temporally) into a (long) Bayesian network. If one views dynamic Bayesian networks as an extension of Bayesian networks all the previous theory applies. Using the chain rule for Bayesian networks again we can specify the full joint over time as

$$P(z_{0:T}) = \Pi_{t=1}^{T} \Pi_{i=1}^{N} P(z_t^i|\text{Parents}(z_t^i)). \tag{3.14}$$

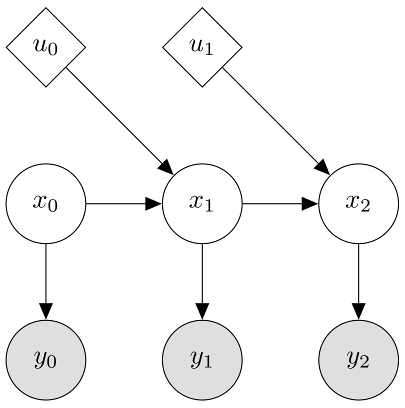

Figure 3.4: Example of a dynamic Bayesian network unrolled for 3 time slices.

3.4 Control

In this section we briefly introduce three fundamental control strategies. First, the linear quadratic regulator (LQR) which deals with the optimal control of linear discrete time invariant systems. Second, we deal with the stochastic generalisation of the LQR controller: the famous linear quadratic Gaussian (LQG) controller. Third and finally, we introduce deterministic model predictive control (MPC). The aim of this section is to introduce and illustrate the relationship between these controllers.

3.4.1 Linear quadratic regulator control

We start our analysis by assuming we have an accurate, linear, discrete, time invariant state space representation of a system

$$x_{t+1} = Ax_t + Bu_t. \tag{3.15}$$

The control sequence N steps into the future is denoted $\mathbf{u} = (u_0, u_1, ..., u_{N-1})$. It is our goal to derive a linear quadratic regulator (controller) given the system in (3.15) and the initial state x_0. Note that it is customary to assign $x_0 \leftarrow x_t$ at each time step to simplify the succeeding optimisation problem's notation.

Definition 3.24. Linear quadratic regulator (LQR) objective function The controller minimising the quadratic objective function

$$V(x_0, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N \tag{3.16}$$

is called the LQR controller.

The optimisation of (3.16) is implicitly subject to the state dynamics (3.15). The matrices Q, R and Q_f are tuning parameters affecting the relative importance of the state and control inputs to the objective function respectively. We also assume that Q, P_f and R are real and symmetric matrices with the additional assumption that Q, P_f are positive semidefinite and R is positive definite.

We assume the reader is familiar with dynamic programming, and present theorem 3.10 because it will be useful later. The proof may be found in [52] and follows from algebraic manipulations.

Theorem 3.10. Sum of quadratics Suppose two quadratic functions $V_1(x) = \frac{1}{2}(x-a)^T A(x-a)$ and $V_2(x) = \frac{1}{2}(x-b)^T B(x-b)$ are given. Then the sum $V_1(x) + V_2(x) = V(x)$ is also quadratic and $V(x) = \frac{1}{2}(x-v)^T H(x-v) + d$ with $H = A + B$, $v = H^{-1}(Aa + Bb)$ and $d = V_1(v) + V_2(v)$.

We now state the complete LQR problem for finite horizon linear systems

$$\begin{aligned} &\min_{\mathbf{u}} V(x_0, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t \end{aligned} \tag{3.17}$$

and analytically solve it using backward dynamic programming. Expanding the objective function to examine its structure we have

$$\begin{aligned} \min_{\mathbf{u}} V(x_0, \mathbf{u}) &= \min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^t R u_k\right) + \frac{1}{2}x_N^T P_f x_N \\ &= \min_{u_0, u_1, \ldots, u_{N-1}} \frac{1}{2}\left(x_0^T Q x_0 + u_0^T R u_0 + x_1^T Q x_1 + u_1^T R u_1 + \ldots + x_N^T P_f x_N\right) \\ &= \min_{u_0, u_1, \ldots, u_{N-2}} \frac{1}{2}\left(x_0^T Q x_0 + u_0^T R u_0 + \ldots + x_{N-2}^T Q x_{N-2} + u_{N-2}^T R u_{N-2}\right) \ldots \\ &\quad + \min_{u_{N-1}} \frac{1}{2}\left(x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + x_N^T P_f x_N\right). \end{aligned} \tag{3.18}$$

Note that given x_0 and the system dynamics all succeeding states are unknown only in the control input. The expansion of the objective function is recursive; this structure motivates the use of dynamic programming. By using theorem 3.10 and the constraint $x_N = Ax_{N-1} + Bu_{N-1}$ we can simplify the last term in the separated minimisation problem of (3.16) by writing

$$\begin{aligned} \min_{u_{N-1}} V_{N-1}(x_N, u_{N-1}) &= \min_{u_{N-1}} \frac{1}{2}\left(x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + x_N^T P_f x_N\right) \\ &= \min_{u_{N-1}} \frac{1}{2}\left(x_{N-1}^T Q x_{N-1} + (u_{N-1} - v)^T H (u_{N-1} - v)\right) + d \\ &\text{with } H = R + B^T P_f B \\ &\text{and } v = K_{N-1} x_{N-1} \\ &\text{and } d = \frac{1}{2} x_{N-1}^T \left(K_{N-1}^T R K_{N-1} + (A + BK_{N-1})^T P_f (A + BK_{N-1})\right) x_{N-1} \\ &\text{and } K_{N-1} = -(B^T P_f B + R)^{-1} B^T P_f A. \end{aligned} \tag{3.19}$$

This is the first step of backward dynamic programming used to solve the problem. Given the form of the objective function we see that the optimal input u_{N-1} is v and consequently that the optimal control law at time $N-1$ is a linear function, K_{N-1}, of x_{N-1}. We also see that the cost function of the last stage is quadratic. The optimal stage cost and controller action is

$$\begin{aligned}
u^0_{N-1}(x) &= K_{N-1}x \\
x^0_{N-1}(x) &= (A + BK_{N-1})x \\
V^0_{N-1}(x) &= \frac{1}{2}x^T\Pi_{N-1}x \\
K_{N-1} &= -(B^TP_fB + R)^{-1}B^TP_fA \\
\Pi_{N-1} &= Q + A^TP_fA - A^TP_fB(B^TP_FB + R)^{-1}B^TP_fA.
\end{aligned} \tag{3.20}$$

The function $V^0_{N-1}(x)$ defines the optimal cost to go from state x for the last stage under the optimal control law $u^0_{N-1}(x)$. Now we proceed with the backward dynamic programming and solve

$$\min_{u_{N-2}} \frac{1}{2}\left(x^T_{N-2}Qx_{N-2} + u^T_{N-2}Ru_{N-2}\right) + V^0_{N-1}(x_{N-1}). \tag{3.21}$$

But now we note the similarity between (3.19) and (3.21). Using $x_{N-1} = Ax_{N-2} + Bu_{N-2}$ and the same procedure as before we have

$$\begin{aligned}
u^0_{N-2}(x) &= K_{N-2}x \\
x^0_{N-2}(x) &= (A + BK_{N-2})x \\
V^0_{N-2}(x) &= \frac{1}{2}x^T\Pi_{N-2}x \\
K_{N-2} &= -(B^T\Pi_{N-1}B + R)^{-1}B^T\Pi_{N-1}A \\
\Pi_{N-2} &= Q + A^T\Pi_{N-1}A - A^T\Pi_{N-1}B(B^T\Pi_{N-1}B + R)^{-1}B^T\Pi_{N-1}A.
\end{aligned} \tag{3.22}$$

The recursion to go from Π_{N-1} to Π_{N-2} is known as backward Ricatti iteration and is defined by

$$\Pi_{k-1} = Q + A^T\Pi_kA - A^T\Pi_kB(B^T\Pi_kB + R)^{-1}B^T\Pi_kA. \tag{3.23}$$

With terminal condition $\Pi_N = P_f$. We see that to find the optimal control policy we need to continue with the backward dynamic programming recursion relationships until $k = 1$. We summarise one of the most fundamental results in optimal control theory in theorem 3.11.

Theorem 3.11. Solution of the finite horizon LQR control problem Given a finite horizon N and a discrete linear system as shown in (3.15) the optimal control policy which minimises the LQR objective function of defintion 3.24 is given by iterating

$$\begin{aligned}
u^0_k(x) &= K_kx \\
K_k &= -(B^T\Pi_{k+1}B + R)^{-1}B^T\Pi_{k+1}A
\end{aligned} \tag{3.24}$$

backwards for $k = N-1, N-2, ..., 1$ using backward Ricatti iteration as shown in (3.23). The optimal cost to go from time k to time N is $V^0_k(x) = \frac{1}{2}x^T\Pi_kx$.

After the optimal input $\mathbf{u}$ is found only u_0 is applied. For a treatment of the continuous case see [28].

Unfortunately optimal control in the setting described above does not guarantee stable control [52]. It can be shown that the finite horizon controller is not guaranteed to be stable i.e. there exist non-trivial systems for which the controller is unstable. This problem is fixed by considering the infinite horizon LQR problem.

Definition 3.25. Infinite horizon LQR problem Find the optimal control sequence $\mathbf{u}$ which solves

$$\begin{aligned} &\min_{\mathbf{u}} V(x, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{\infty}\left(x_k^T Q x_k + u_k^T R u_k\right) \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t \\ &\text{and } x_0 = x. \end{aligned} \tag{3.25}$$

The same restrictions on the tuning parameters apply as before.

By assuming that the system under consideration is controllable it is possible to show that the infinite horizon LQR solution shown in theorem 3.12 is convergent and stabilising [52].

Definition 3.26. Controllability A system is controllable is, for any pair of states x, z in the state space, z can be reached in finite time from x. That is, x can be controlled to z. It is possible to characterise a controllable system further. A system with n variables (which require control) is controllable if and only if rank $\begin{pmatrix}\lambda I - A & B\end{pmatrix} = n$ for all $\lambda \in \text{eig}(A)$.

Theorem 3.12. Solution of the infinite horizon LQR control problem Given the infinite horizon LQR problem of definition 3.25 it can be shown that the optimal control is given by

$$\begin{aligned} &u_k^0(x) = Kx \\ &K = -(B^T \Pi B + R)^{-1} B^T \Pi A \\ &\Pi = Q + A^T \Pi A - A^T \Pi B (B^T \Pi B + R)^{-1} B^T \Pi A. \end{aligned} \tag{3.26}$$

The optimal cost is given by $V^0(x) = \frac{1}{2}x^T Kx$. The matrix Π can be found by iterating the Ricatti equation. This solution is stabilising if the system is controllable [52].

3.4.2 Reference tracking

The LQR control problem, as discussed in the previous section, applies to deterministic systems where the goal is to drive the controlled variables to the origin. It is straightforward to extend this approach to systems where it is desired to drive the states to a reference (set) point r_{sp}.

To achieve this we simply redefine the objective function in terms of deviation variables

$$\begin{aligned} &\tilde{x}_t = x_t - x_{sp} \\ &\tilde{u}_t = u_t - u_{sp}. \end{aligned} \tag{3.27}$$

The constants x_{sp} and u_{sp} are the state and corresponding controller set point one would like to drive the system to. The deviation variables are then used in the objective function

$$\begin{aligned} \min_{\tilde{\mathbf{u}}} V(x_0, \tilde{\mathbf{u}}) &= \frac{1}{2}\sum_{k=0}^{N-1} \left(\tilde{x}_k^T Q \tilde{x}_k + \tilde{u}_k^T R \tilde{u}_k\right) + \frac{1}{2}\tilde{x}_N^T P_f \tilde{x}_N \\ \text{subject to } \tilde{x}_{t+1} &= A\tilde{x}_t + B\tilde{u}_t \end{aligned} \tag{3.28}$$

as opposed to x_t and u_t. The system dynamics remain the same [52] and only $\tilde{u}_0$ is used as before. We apply $u_0 = \tilde{u}_0 + u_{sp}$ to the system. All that is required is that we specify x_{sp} and u_{sp}. This is done by solving

$$\begin{pmatrix} I - A & -B \\ HC & 0 \end{pmatrix} \begin{pmatrix} x_{sp} \\ u_{sp} \end{pmatrix} = \begin{pmatrix} 0 \\ r_{sp} \end{pmatrix}. \tag{3.29}$$

Note that H relates the observed variables to the controlled variables. If there are more measured outputs than manipulated variables (3.29) cannot be solved directly. It is possible to cast (3.29) into an optimisation problem. We refer the reader to [52] for a full treatise on the subject.

3.4.3 Linear quadratic Gaussian control

The LQR problem dealt with deterministic systems where the states were known exactly. However, this is problematic from a practical perspective because:

1. The system model is almost never known exactly.

2. The state measurements are almost always noisy.

The linear quadratic Gaussian (LQG) controller for stochastic systems of the form

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \\ y_t &= Cx_t + v_t \end{aligned} \tag{3.30}$$

was developed to handle these problems. Additionally we assume $w_t \sim \mathcal{N}(0, W)$ and $v_t \sim \mathcal{N}(0, V)$ which are independent white noise terms. Because the states and measurements are stochastic variables we cannot use the LQR objective function as before. Instead we use the LQG objective function which is a generalisation of the former as shown in definition 3.27.

Definition 3.27. Linear quadratic Gaussian (LQG) objective function The controller minimising the quadratic objective function

$$V(x_0, \mathbf{u}) = \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1} \left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \tag{3.31}$$

is called the LQG controller. The restrictions on the tuning parameters are the same as before. It is assumed that y_0 is used to infer x_0 which is then used in the optimisation problem.

The full LQG control problem is

$$\begin{aligned}&\min_{\mathbf{u}} V(x_0, \mathbf{u}) = \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t.\end{aligned} \tag{3.32}$$

It is indeed possible to solve this controller analytically using stochastic dynamic programming but the derivation is tedious. We rather employ the separation principle [15]. We do however re-derive the optimal controller in a later chapter.

Definition 3.28. Separation principle The solution of the LQG problem is obtained by combining the solution of the deterministic LQR problem and the optimal state estimation problem. The optimal current state estimate is used as the current deterministic state within the framework of the LQR controller. This is also sometimes called certainty equivalence.

The optimal state estimate of linear systems under Gaussian noise is known as the Kalman filter. In later a chapter we devote much time to its derivation but for now we merely introduce it loosely.

Definition 3.29. Kalman filter The optimal linear state estimator for Gaussian random variables is called the Kalman filter.

A schematic diagram of the solution of LQG control problem is shown in figure 3.5.

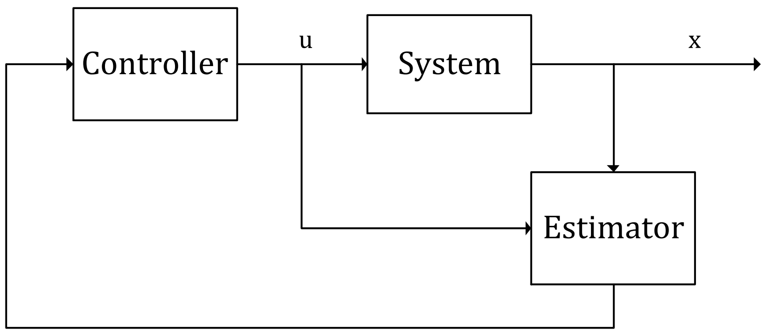

Figure 3.5: LQG control schematic. The estimator is the Kalman filter and the controller is the deterministic LQR.

The LQR and LQG controller solutions are two of the most fundamental results in optimal control theory [28]. In the next section we discuss model predictive control which is a generalisation of the controllers we have discussed so far.

3.4.4 Model predictive control

Model predictive control (MPC) is the constrained generalisation of the LQR controller. It is widely used industry and has been the subject of a significant amount of scholarly research.

We introduce the classic deterministic linear MPC

$$\begin{aligned}&\min_{\mathbf{u}} V(x_0, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\\ &\text{subject to } x_{t+1} = Ax_t + Bu_t\\ &\text{and } d^T x_t + e \geq 0 \;\forall\; t = 1, 2, ..., N\end{aligned} \tag{3.33}$$

with coincidental control and prediction horizons N.

It is straightforward to incorporate more constraints of the form $d_i^T x_t + e_i \geq 0$ if required. The structure of (3.33) is important: the objective function is quadratic and the constraints are linear. There are two primary benefits of this structure. Firstly, the problem is provably convex which means that if a minimum is found it is the global minimum. Secondly, (and as a consequence of the first benefit) this allows for the use of specialised quadratic programming techniques which are fast and reliable.

Unfortunately there is no tractable way to analytically compute the control law offline, except in trivial cases, like the LQR controller. It is invariably necessary to use some optimisation algorithm to solve for $\mathbf{u}$ given x_0.

Recent advances in quadratic programming algorithms, like the interior point algorithm, have made it possible to solve quadratic programming problems almost as fast as linear programming problems. These solvers typically exploit the sparseness structure inherent in problems like (3.33) [43]. Thus, it is desirable to make use of these solvers wherever possible by ensuring that the underlying quadratic programming structure is not lost when modifications are made to the algorithm.

Finally, the stability and robustness (e.g. when there is plant/model mismatch) of deterministic linear MPC is well studied and understood in modern literature. See [52] and [43] for details. Thus, it is likewise desirable to exploit this knowledge rather than attempt to derive an MPC with completely different or new characteristics.

We will continue our study of linear MPC in later sections.

3.5 Matrix identities

It will be useful in a later section to have access to a block matrix inversion formula. We state the result without proof and refer the reader to [33] for more details.

Theorem 3.13. Block matrix inversion Suppose we have a square block matrix of the form $\begin{pmatrix} A & b \\ c^T & d \end{pmatrix}$ where A is an invertible matrix; b and c are conforming vectors; and d is a real number. Then the identity

$$\begin{pmatrix} A & b \\ c^T & d \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1}(I + p^{-1}bc^T A^{-1}) & -p^{-1}A^{-1}b \\ -p^{-1}c^T A^{-1} & p^{-1} \end{pmatrix} \tag{3.34}$$

holds with $p = (d - c^T A^{-1} b)$.

Chapter 4

Hidden Markov models

In this chapter we consider probabilistic graphical models of the form shown in figure 4.1. We assume that $(X_0, X_1, \ldots)$ are each n state discrete random variables and that $(Y_0, Y_1, \ldots)$ are each m state discrete random variables. Models of this form are classically called hidden Markov models.

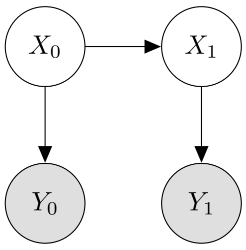

Figure 4.1: Graphical model used in this chapter.

Intuitively, this model represents the situation where we are not sure about the state of the world but we can observe some facet of it. At each time step our state changes stochastically according to the transition function. A new observation is (stochastically) generated from our new state. Generally, we attempt to infer the state of the world given the observations. Clearly two new random variables X_t, Y_t are created at each time step.

In this chapter we briefly describe Markov models because they link back to previous work done by the chemical engineering department at the University of Pretoria. We focus on hidden Markov models for the remainder of the chapter because the techniques we develop here generalise to linear latent dynamical systems which we discuss in chapter 6.

4.1 Markov models

A first order Markov model (sometimes called a Markov chain) is shown in figure 4.2. Using the chain rule for Bayesian networks (definition 3.19) we can immediately write down the joint probability distribution

$$P(X_{0:T}) = \Pi_{t=0}^{T} P(X_t|X_{t-1}) \text{ with } P(X_0|X_{-1}) = P(X_0). \tag{4.1}$$

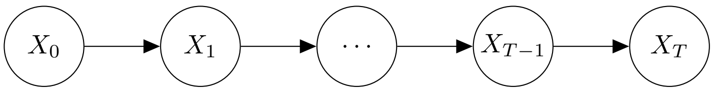

Figure 4.2: First order markov chain.

This model describes the forward propagation of a discrete random variable through time. It is interesting to study the marginal distribution of $P(X_T)$ as it evolves through time. By d-separation we know that $X_t \perp\!\!\!\perp X_{0:t-2}|X_{t-1}$. Thus, we only have to marginalise out the previous time step to compute the required distribution

$$P(X_T) = \sum_{x_{T-1}} P(X_T, x_{T-1}) = \sum_{x_{T-1}} P(x_{T-1})P(X_T|x_{T-1}). \tag{4.2}$$

Since we know that the transition function is a row stochastic $n \times n$ matrix (the random variable has n discrete states) we can write (4.2) in vector notation

$$P(X_t) = \mathbf{p}_t = \mathbf{A}\mathbf{p}_{t-1} = \mathbf{M}^{t-1}\mathbf{p}_1. \tag{4.3}$$

Note that $P(X_T)$ is a discrete random variable and can thus be expressed as a stochastic column vector i.e. $\sum_i P(x_t = i) = 1$. We have implicitly rewritten (4.3) in recursive format. Thus, we have a recursive expression for the marginal distribution of X. If, as $T \to \infty$, we have that $\mathbf{p}_{t\to\infty} = \mathbf{p}_\infty$ exists and is independent of $\mathbf{p}_0$ we call $\mathbf{p}_\infty$ the equilibrium distribution of the chain.

We define the stationary distribution, in matrix notation, by

$$\mathbf{p}_\infty = \mathbf{A}\mathbf{p}_\infty. \tag{4.4}$$

Recalling the definition of the eigenvalue problem we see that the stationary distribution is just the eigenvector corresponding to the unit eigenvalue of $\mathbf{A}$. While this model may seem simplistic it is the foundation of Google's PageRank algorithm [63]. Intuitively $\mathbf{p}_\infty$ represents the steady state probability distribution of the random variable X as it is propagated through time by the transition function A. See the work in [57] for an application specifically geared towards chemical engineering.

4.2 Hidden Markov models

Hidden Markov models extend Markov models by incorporating the observed random variables $(Y_0, Y_1, \ldots)$ as shown in figure 4.1. At each time step it is now possible to observe the random variable Y_t which gives more information about the state of X_t. We are still in the setting of discrete random variables. It is not necessary to restrict $(Y_0, Y_1, \ldots)$ to be discrete but we do so for the sake of simplicity here. In later chapters we will model both hybrid and purely continuous systems.

In general a hidden Markov model is just a specific case of the general dynamic Bayesian network class of graphical models. As such we already know that to fully specify the model we only require a prior state distribution $P(X_0)$, the transition probability function $P(X_t|X_{t-1})$, the observation (or sometimes called the emission) probability function $P(Y_t|X_t)$ and the Bayesian network graphs of the initial time step and the next two time steps. We assume that the model's structure repeats at each time step and thus we only require the graph as shown in figure 4.1.

We assume that the transition and observation probability functions are stationary. Consequently they may be represented by the row stochastic square matrices $P(x_t = i|x_{t-1} = j) = A$ and $P(y_t = i|x_t = j) = B$. Intuitively this means that the probability of state $x_{t-1} = j$ going to state $x_t = i$ is A_{ij}. Similarly, B_{ij} is the probability of observing $y_t = i$ if the underlying state is $x_t = j$.

For the purposes of this dissertation we will always assume that the model parameters are known. In section 3.3.2 the four primary inference techniques were briefly mentioned. We now derive recursive expressions for each inference technique for discrete models of the form shown in figure 4.1. The tools and techniques we develop here will be useful in the following chapters.

4.2.1 Filtering

The goal of filtering is to find $P(X_t|y_{0:t})$: the distribution of the current state given all the past and current observations. The corresponding graphical model is shown in figure 4.3.

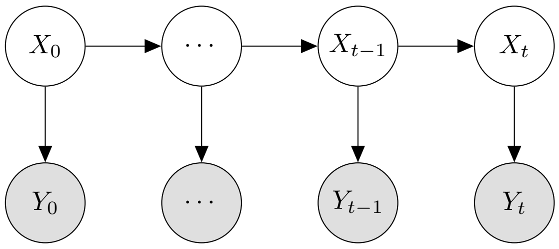

Figure 4.3: Filtering graphical model.

We start the derivation by noting that X_{t-1} d-separates X_t from $X_{0:t-2}$. Thus X_{t-1} contains all the hidden state information of the system up to and including $t-1$. This is not surprising since we have assumed a first order Markov system. This is why we only marginalise over the reduced state joint $P(X_t, X_{t-1}|y_{0:t})$ in

$$\begin{aligned} P(X_t|y_{0:t-1}) &= \sum_{x_{t-1}} P(X_t, x_{t-1}|y_{0:t-1}) \\ &= \sum_{x_{t-1}} P(x_{t-1}|y_{0:t-1})P(X_t|y_{0:t-1}, x_{t-1}) \\ &= \sum_{x_{t-1}} P(x_{t-1}|y_{0:t-1})P(X_t|x_{t-1}). \end{aligned} \tag{4.5}$$

The expansion followed from the chain rule of Bayesian networks (definition 3.19) and the cancellation followed from the conditional independence assumption of the transition function. Next we define $\alpha(X_t) \equiv P(X_t|y_{0:t})$; then

$$\begin{aligned} \alpha(X_t) &= \frac{P(y_t|X_t, y_{0:t-1})P(X_t|y_{0:t-1})}{P(y_t|y_{0:t-1})} \\ &= \frac{P(y_t|X_t)P(X_t|y_{0:t-1})}{P(y_t|y_{0:t-1})} \\ &= \frac{P(y_t|X_t)\sum_{x_{t-1}} P(x_{t-1}|y_{0:t-1})P(X_t|x_{t-1})}{P(y_t|y_{0:t-1})} \\ &= \frac{P(y_t|X_t)\sum_{x_{t-1}} \alpha(x_{t-1})P(X_t|x_{t-1})}{P(y_t|y_{0:t-1})} \end{aligned} \tag{4.6}$$

follows from (4.5) and by application of Bayes' theorem (theorem 3.4). It is not actually necessary to calculate $p(y_t|y_{0:t-1})$ as it is only a normalisation constant. We thus have a recursion relation for the filtered posterior distribution X_t with initial condition $\alpha(X_1) = P(X_1, y_1) = P(X_1)P(y_1|X_1)$ by

$$\alpha(X_t) \propto P(y_t|X_t) \sum_{x_{t-1}} \alpha(x_{t-1})P(X_t|x_{t-1}). \tag{4.7}$$

One often uses logarithms to perform the filter calculations as machine precision errors become a problem for large t due to the multiplication of small fractions. The recursive filtering algorithm we derived is often called the forwards algorithm in literature [4].

4.2.2 Smoothing

The goal of smoothing is to find $P(X_t|y_{0:T})$ for $t \leq T$: the distribution of the state at t given all the past and future observations to T. The smoothing algorithm we study here is called the parallel smoothing algorithm. The recursion expression we derive is often called the backwards algorithm in literature [47]. The graphical model corresponding to this situation is shown in figure 4.4.

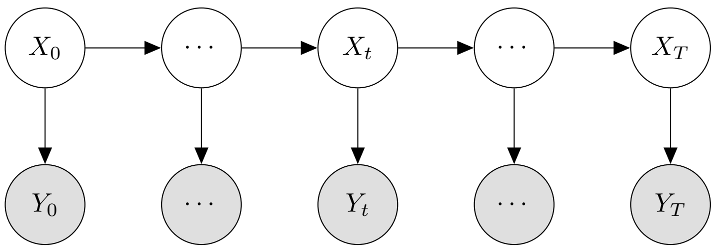

Figure 4.4: Smoothing graphical model.

We start by splitting the joint $P(X_t, y_{0:T}) = P(X_t, y_{0:t})P(y_{t+1:T}|X_t, y_{0:t})$ by the chain rule and using d-separation to reduce it further $P(X_t, y_{0:T}) = P(X_t, y_{0:t})P(y_{t+1:T}|X_t)$. Effectively the last step implies that future observations are independent of past observations given the current state. We defined $\beta(X_t) \equiv P(y_{t+1:T}|X_t)$ and continue

$$\begin{aligned} P(y_{t:T}|X_{t-1}) &= \sum_{x_t} P(y_t, y_{t+1:T}, x_t|X_{t-1}) \\ &= \sum_{x_t} P(y_{t+1:T}, x_t|X_{t-1})P(y_t|y_{t+1:T}, x_t, X_{t-1}) \\ &= \sum_{x_t} P(y_{t+1:T}, x_t|X_{t-1})P(y_t|x_t) \\ &= \sum_{x_t} P(x_t|X_{t-1})P(y_{t+1:T}|x_t, X_{t-1})P(y_t|x_t) \\ &= \sum_{x_t} P(x_t|X_{t-1})P(y_{t+1:T}|x_t)P(y_t|x_t). \end{aligned} \tag{4.8}$$

We have made judicious use of the implied independence assertions (via d-separation) of figure 4.1. Making use of the definition of β we have

$$\begin{aligned} &\beta(X_{t-1}) = \sum_{x_t} P(x_t|X_{t-1})\beta(x_t)P(y_t|x_t) \text{ for } 1 \leq t \leq T \\ &\text{with } \beta(X_T) = 1. \end{aligned} \tag{4.9}$$

The recursion initial condition $\beta(X_T) = 1$ stems from Bayes' theorem (theorem 3.4) and the definition of α as illustrated by

$$\begin{aligned} P(X_T|y_{0:T}) &= \frac{P(X_T, y_{0:T})}{P(y_{0:T})} \\ &= \alpha(X_T)\beta(X_T) \\ &= P(X_T|y_{0:T})\beta(X_T) \\ &\implies \beta(X_T) = 1. \end{aligned} \tag{4.10}$$

Note that β is not a probability function. The smoothed posterior is given by applying Bayes' theorem

$$P(X_t|y_{0:T}) = \frac{\alpha(X_t)\beta(X_t)}{\sum_{x_t} \alpha(x_t)\beta(x_t)}. \tag{4.11}$$

Together the $\alpha - \beta$ recursions are called the forwards-backwards algorithm and find extensive use in general purpose exact inference of dynamic Bayesian networks [47].

Numerical issues may also become problematic due to the multiplication of small positive numbers. In practice it is often necessary to work in the log space to attenuate these problems [4].

4.2.3 Viterbi decoding

The goal of Viterbi decoding is to find $x^*_{0:T} = \underset{x_{0:T}}{\arg\max}\, P(x_{0:T}|y_{0:T})$: finding the most likely sequence of states which best describe the observations by attempting to find the sequence

$x_{0:T}$ such that the joint probability function $P(x_{0:T}, y_{0:T})$ is maximised. This is equivalent to finding $\arg\max_{x_{0:T}} P(x_{0:T}|y_{0:T})$ because, if one uses the chain rule on the joint, the observations will just be a constant factor. The graphical model used here is similar to that of figure 4.4.

Intuitively we first attempt to find the maximum of the joint and then determine which sequence of states led to this maximal joint. By using the chain rule for Bayesian networks we can rewrite the joint maximisation problem as

$$\begin{aligned}\max_{x_{0:T}} P(x_{0:T}, y_{0:T}) &= \max_{x_{0:T}} \Pi_{t=1}^{T} P(y_t|x_t)P(x_t|x_{t-1}) \\ &= \left(\max_{x_{0:T-1}} \Pi_{t=1}^{T-1} P(y_t|x_t)P(x_t|x_{t-1})\right) \max_{x_T} P(y_T|x_T)P(x_T|x_{T-1}).\end{aligned} \tag{4.12}$$

Defining $\mu(X_{t-1}) \equiv \max_{x_t} P(y_t|x_t)P(x_t|x_{t-1})$ we can rewrite (4.12) as

$$\max_{x_{0:T}} P(x_{0:T}, y_{0:T}) = \max_{x_{0:T-1}} \Pi_{t=1}^{T-1} P(y_t|x_t)P(x_t|x_{t-1})\mu(x_{T-1}). \tag{4.13}$$

Thus we have a recursive expression,

$$\begin{aligned}&\mu(x_{t-1}) = \max_{x_t} P(y_t|x_t)P(x_t|x_{t-1})\mu(x_t) \text{ for } 2 \le t \le T \\ &\text{with } \mu(x_T) = 1,\end{aligned} \tag{4.14}$$

to find the value of the joint under the most likely sequence of states given the observations.

The recursive expression in (4.14) implies that the effect of maximising over the previous time step can be compressed into a message (a function) of the current time step. Effectively we pass theses messages backward in time to find the maximum joint in terms of x_0. We then find the state which maximises this joint and pass this message forward. Continuing in this way we have

$$\begin{aligned}x_1^* &= \arg\max_{x_1} P(y_1|x_1)P(x_1)\mu(x_1) \\ x_2^* &= \arg\max_{x_2} P(y_2|x_2)P(x_2|x_1^*)\mu(x_2) \\ &\vdots \\ x_t^* &= \arg\max_{x_t} P(y_t|x_t)P(x_t|x_{t-1}^*)\mu(x_t).\end{aligned} \tag{4.15}$$

This algorithm is called the Viterbi algorithm. It is computationally efficient since the optimisations occur only on a single variable. Readers familiar with dynamic programming will recognise that we have effectively performed a variant of dynamic programming in the preceding derivation.

4.2.4 Prediction

The goal of prediction is to find $P(X_{t+1}|y_{0:t})$ and $P(Y_{t+1}|y_{0:t})$: the predicted hidden and observed state given all the previous observations. The one step ahead prediction expression for both the states and observations is derived here. The graphical model corresponding to the state prediction is shown in figure 4.5.

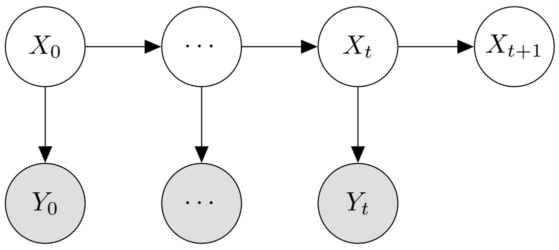

Figure 4.5: State prediction graphical model.

We again start by noticing that given all the observations up to time t the current state d-separates all previous states. Thus, to infer information about the next state we only need to marginalise out the current state X_t. Furthermore, the next state d-separates the next observation from all the previous states. Thus, to infer information about the next observation we additionally only need to marginalise out X_{t+1}.

We start with predicting the next state distribution. We have applied the chain rule, the independence assertions and used the definition of α to write

$$\begin{aligned} P(X_{t+1}|y_{0:t}) &= \sum_{x_t} P(X_{t+1}, x_t|y_{0:t}) \\ &= \sum_{x_t} P(x_t|y_{0:t})P(X_{t+1}|x_t, y_{0:t}) \\ &= \sum_{x_t} P(x_t|y_{0:t})P(X_{t+1}|x_t) \\ &= \sum_{x_t} \alpha(x_t)P(X_{t+1}|x_t). \end{aligned} \tag{4.16}$$

Clearly the state prediction uses the filtered state estimate and projects that forward using the transition function. Next we derive the observation prediction. Again we apply the chain rule, use the independence assertions and use the definition of α to write

$$\begin{aligned} P(Y_{t+1}|y_{0:t}) &= \sum_{x_t,x_{t+1}} P(x_{t+1}, x_t, Y_{t+1}|y_{0:t}) \\ &= \sum_{x_t,x_{t+1}} P(x_t|y_{0:t})P(x_{t+1}, Y_{t+1}|y_{0:t}, x_t) \\ &= \sum_{x_t,x_{t+1}} P(x_t|y_{0:t})P(x_{t+1}|y_{0:t}, x_t)P(Y_{t+1}|y_{0:t}, x_t, x_{t+1}) \\ &= \sum_{x_t,x_{t+1}} P(x_t|y_{0:t})P(x_{t+1}|x_t)P(Y_{t+1}|x_{t+1}) \\ &= \sum_{x_t,x_{t+1}} \alpha(x_t)P(x_{t+1}|x_t)P(Y_{t+1}|x_{t+1}). \end{aligned} \tag{4.17}$$

Clearly the observation prediction is just an extension of the state prediction. We effectively just predict the next state and use the observation function to predict the observation distribution.

It is pleasing that the prediction expressions are closely related to each other and effectively only depend on the filtered state estimate and the transition or observation functions. This realisation hold for more general problems and will become important when we consider controlling the system. More on this later.

4.3 Burglar localisation problem

While the focus of this dissertation is not on hidden Markov model type problems it is nevertheless instructive to consider a simple example to build some intuition about inference of random variables. Thus, it is desirable to conduct a numerical experiment using the previously derived inference techniques. The type of problem we consider here is a localisation problem. This type of problem (and its extensions) has many applications in robotics and object tracking.

The problem is taken from chapter 23 in Barber's book *Bayesian Reasoning and Machine Learning* [4]. Briefly, it is necessary to infer the location of a burglar in your house given observations (noises) you perceive from an adjoining room. You discretise the room the burglar is in into n^2 blocks. The room is then the discrete random variable X. You observe two distinct types of noises: creaks and bumps. From the knowledge of your house you know which blocks are likely to creak and which are likely to bump if the burglar is on that block i.e. if the random variable X is in a specific state. This is shown in figure 4.6: a dark block indicates it is likely to emit a noise with probility 0.9 and a light block will emit a noise with probability 0.01 if the burglar is on that block. The noises are independent of each other.

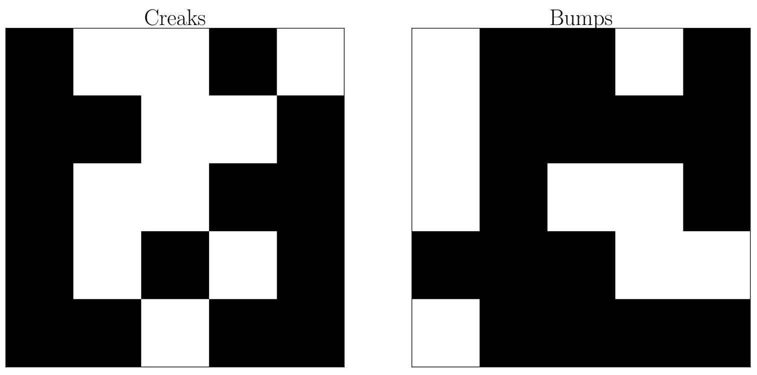

Figure 4.6: Burglar problem observations.

The burglar moves up, down, left and right with equal probability where appropriate. See Barber for more details on the example. It is necessary to perform inference to determine the path of the burglar both in real time and with the benefit of hindsight. Applying the inference techniques we developed earlier results in figure 4.7.

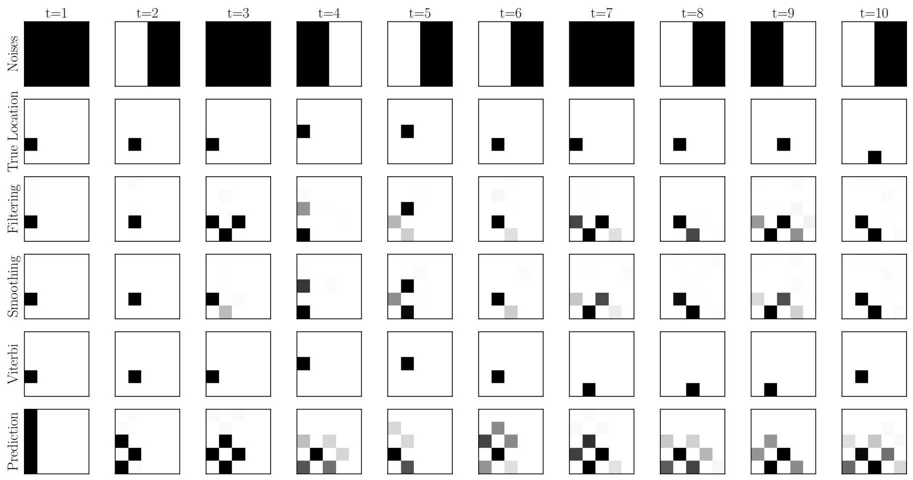

Figure 4.7: Burglar problem: filter, smoothing, Viterbi decoding and prediction.

In this context filtering means we estimate the location of the burglar given all the past observations at the current time step. This inference can be done on-line. Smoothing means we attempt to estimate his position with hindsight given all the observations starting from the first time step and moving forwards. In Viterbi decoding we attempt to estimate the most likely path path of the burglar. Finally the prediction algorithm is self-explanatory.

It is interesting to note that smoothed posterior converges to the filtered posterior near the end of the time window. Reflecting on the expression for smoothing this is not surprising since at $t = T$ the smoothing component of the forwards-backwards algorithm is unity. Therefore we see that the smoothed state estimate converges to the filtered state estimate as t approaches T.

It is also interesting to note that the prediction algorithm is very much dependent on the quality of the transition function. The four block pattern readily apparent in the prediction distribution originates from the transition function (the burglar is equally likely to move in any direction). This strongly implies that the closer the transition function is to reality the better our predictions will be.

Finally, it is important to understand the benefit of using this approach as opposed to the exhaustive “if this then that” approach. Firstly, the latter approach scales exponentially with the number of variables because one would need to fully consider all the possibilities to infer any sort of belief. Secondly, the former approach has an associated probability distribution: the certainty of our inferred belief is automatically quantified e.g. the darker the blocks the more sure we are about our inference. Thus, the techniques we developed make room for uncertainty about the correctness of the answer.

Hidden Markov models are very powerful and have found many uses e.g. speech recognition, object tracking and bio-informatics [4]. Many extensions of the basic model (see figure 4.1) exist which are much more expressive. However, we are interested in modelling and reasoning about continuous random variables. For such applications the hidden Markov model, due to the discrete assumption, is inappropriate. Fortunately, the techniques investigated in this chapter carry over to the continuous case as we will see in chapter 6.

Chapter 5

CSTR model

In this chapter we introduce the model we will use to illustrate the techniques we develop in this dissertation. The model is a simple continuously stirred tank reactor (CSTR) undergoing an exothermic, irreversible first order reaction where $A \rightarrow B$. A schematic diagram of the reactor is shown in figure 5.1. The model is taken from literature [45].

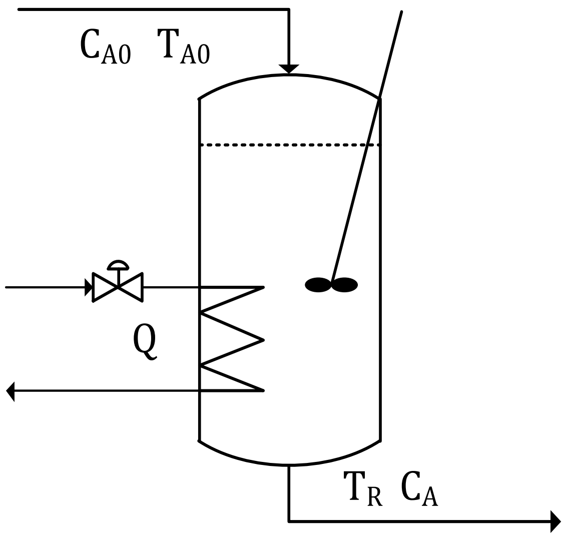

Figure 5.1: Diagram of a simple CSTR where the heat added to system is the only manipulated variable.

The state space equations describing the reactor are

$$
\begin{aligned}
\dot{C_A} &= f(C_A, T_R) = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT_R}} C_A \\
\dot{T_R} &= g(C_A, T_R) = \frac{F}{V}(T_{A0} - T_R) + \frac{-\triangle H}{\rho C_p} k_0 e^{\frac{-E}{RT_R}} C_A + \frac{Q}{\rho C_p V}
\end{aligned}
\tag{5.1}
$$

with parameters shown in table 5.1. The meaning of the variables is what one would expect from an intuitive understanding: C_A is the concentration of species A, T_R is the temperature of the CSTR and Q is the heat added (or removed for negative Q) from the CSTR.

Table 5.1: CSTR parameters

V	$5.0\ m^3$	R	$8.314\ \frac{kJ}{kmol.K}$
C_{A0}	$1.0\ \frac{kmol}{m^3}$	T_{A0}	$310\ K$
$\triangle H$	$-4.78 \times 10^4\ \frac{kJ}{kmol}$	k_0	$72 \times 10^7\ \frac{1}{min}$
E	$8.314 \times 10^4\ \frac{kJ}{kmol}$	C_p	$0.239\ \frac{kJ}{kg.K}$
ρ	$1000\ \frac{kg}{m^3}$	F	$100 \times 10^{-3}\ \frac{m^3}{min}$

The CSTR model is a familiar control example. Similar models may be found in [23], [12], [51] and [66]. We use this model because it is low dimensional yet complex enough to illustrate the principles we investigate. CSTR models are also very popular examples to illustrate control techniques because they invariably have nonlinear dynamics and multiple steady states. Note that we have increased the volume of the reactor and reduced the rate constant from the reactor we quoted in literature. This is primarily to adjust the time scale of the transient response to be in the order of minutes and not milliseconds when moving to high temperature regions of operation.

5.1 Qualitative analysis

In this section we use standard mathematical tools, as found in [26], to analyse the qualitative behaviour of the CSTR. By inspecting (5.1) we see that the model is coupled and nonlinear. By solving

$$
\begin{aligned}
0 &= \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT_R}} C_A \\
0 &= \frac{F}{V}(T_{A0} - T_A) + \frac{-\triangle H}{\rho C_p} k_0 e^{\frac{-E}{RT_R}} C_A + \frac{Q}{\rho C_p V}
\end{aligned}
\tag{5.2}
$$

we see that for nominal operating conditions ($Q = 0$) there exist 3 operating points (critical points) as shown in table 5.2.

Table 5.2: Nominal operating points for the CSTR

Critical Point	C_A	T_R	Stability
(C_A^1, T_R^1)	0.0097	508	Stable Improper Node
(C_A^2, T_R^2)	0.4893	412	Unstable Saddle Point
(C_A^3, T_R^3)	0.9996	310	Stable Improper Node

The stability of the operating points were found by linearising (5.1) and computing the eigenvalues of the jacobian,

$$
J(C_A, T_R) = \begin{pmatrix} -\frac{F}{V} - k_0 e^{\frac{-E}{RT_R}} & -k_0 e^{\frac{-E}{RT_R}} C_A \left(\frac{E}{RT_R^2}\right) \\ \frac{-\triangle H}{\rho C_p} k_0 e^{\frac{-E}{RT_R}} & -\frac{F}{V} + \frac{-\triangle H}{\rho C_p} k_0 e^{\frac{-E}{RT_R}} C_A \left(\frac{E}{RT_R^2}\right) \end{pmatrix}, \tag{5.3}
$$

at each critical point. In figure 5.2 we see the operating curve for the CSTR. The curve resembles the classical CSTR operating curve with all the associated potential control complexity e.g. it is possible for one set of control inputs to result in two stable operating points. This occurs due to the two stable critical points (for $Q \in [-906, 1145]$) of the system and is called input multiplicity [42].

Also note that the obvious bifurcation parameter for this system is the heat input Q. For $Q = -906$ kJ/min we see that we no longer have three critical points but only two, and for $Q < -906$ kJ/min we only have one critical point. Likewise, for $Q = 1145$ kJ/min we see that we only have two critical points and for $Q > 1145$ kJ/min we only have one critical point. The stability of these points are shown in table 5.3.

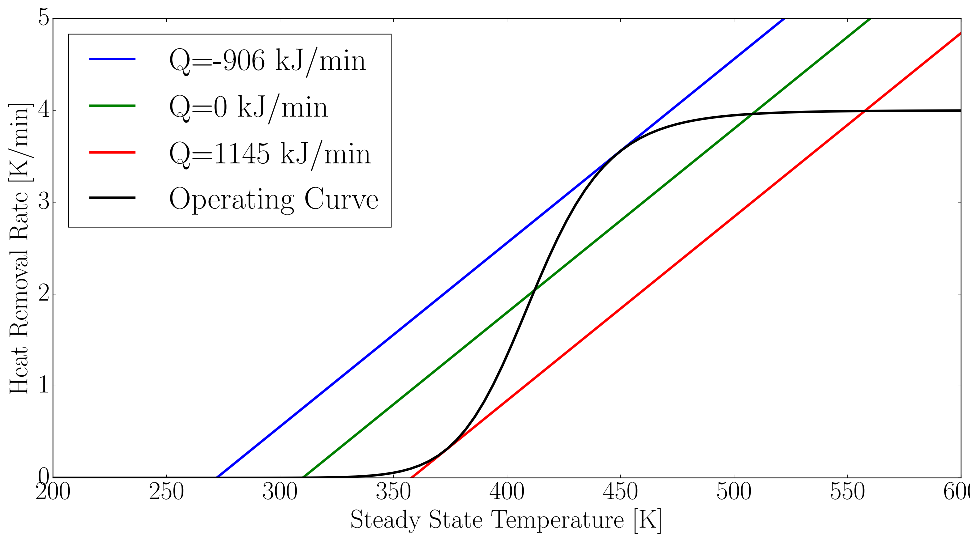

Figure 5.2: CSTR operating curve with different input curves. Nominal operating conditions are $Q = 0$ kJ/min.

Table 5.3: Bifurcation analysis of the CSTR at different heat input values. The $\backsim$ notation indicates that the operating point depends on the precise value of Q.

Heat Input	C_A	T_R	Stability
$Q = -906$ kJ/min	0.1089	450	Stable Improper Node
$Q = -906$ kJ/min	0.9999	272	Stable Improper Node
$Q < -906$ kJ/min	$\backsim$	$\backsim$	Stable Improper Node
$Q = 1145$ kJ/min	0.0017	558	Stable Improper Node
$Q = 1145$ kJ/min	0.9263	373	Stable Improper Node
$Q > 1145$ kJ/min	$\backsim$	$\backsim$	Stable Improper Node

The (potentially) multiple stable critical points separated by the unstable critical point makes control of this system challenging.

5.2 Nonlinear model

In this section we evaluate the transient response of the CSTR. The nonlinear differential equation shown in (5.1) is intractably difficult to solve analytically. For this reason we will use a numerical method, specifically the Runge-Kutta method [26], to simulate the transient response. We chose the Runge-Kutta method because it is an explicit, fourth order accurate method which is easy to implement.

For completeness we show the method here. Suppose we have an autonomous ordinary differential equation,

$$\begin{aligned} &\dot{x}(t) = f(x(t)) \\ &\text{with } x(t) = x_a \text{ for } t = t_a, \end{aligned} \tag{5.4}$$

and we require its solution over $[t_a, t_b]$. This is an initial value problem; for the sake of brevity we assume that a unique solution always exists. Furthermore, suppose we discretise the time domain such that $[t_a, t_b] = [t_0 = t_a, t_1 = t_a + h, t_2 = t_a + 2h, ..., t_T = t_b]$. Then the scheme

$$\begin{aligned} x_{t+1} &= x_t + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k4) \\ k_1 &= f(x_t) \\ k_2 &= f(x_t + \frac{h}{2}k_1) \\ k_3 &= f(x_t + \frac{h}{2}k_2) \\ k_4 &= f(x_t + hk_3) \end{aligned} \tag{5.5}$$

is called the Runge-Kutta method. For sufficiently small time steps, h, the method is stable and convergent.

By applying the Runge-Kutta method to the CSTR we have figures 5.3 and 5.4. It is clear that the dynamics are faster (almost two orders of magnitude) when moving to the higher temperature operating point than they are when moving to the lower temperature operating point. The impact of the nonlinear kinetics is seen here.

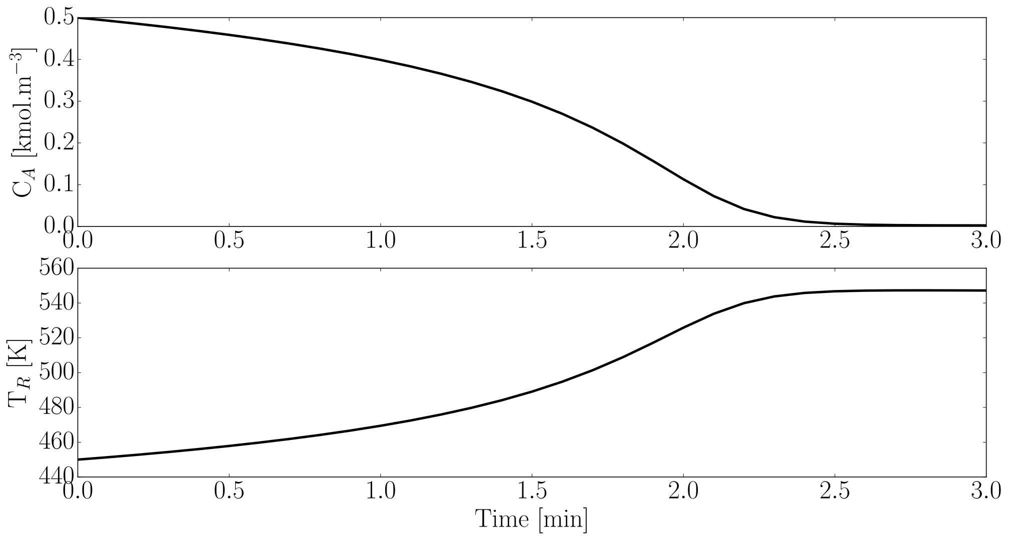

Figure 5.3: Transient response of the CSTR under nominal operating conditions with initial condition $(0.5, 450)$ and $h = 0.1$.

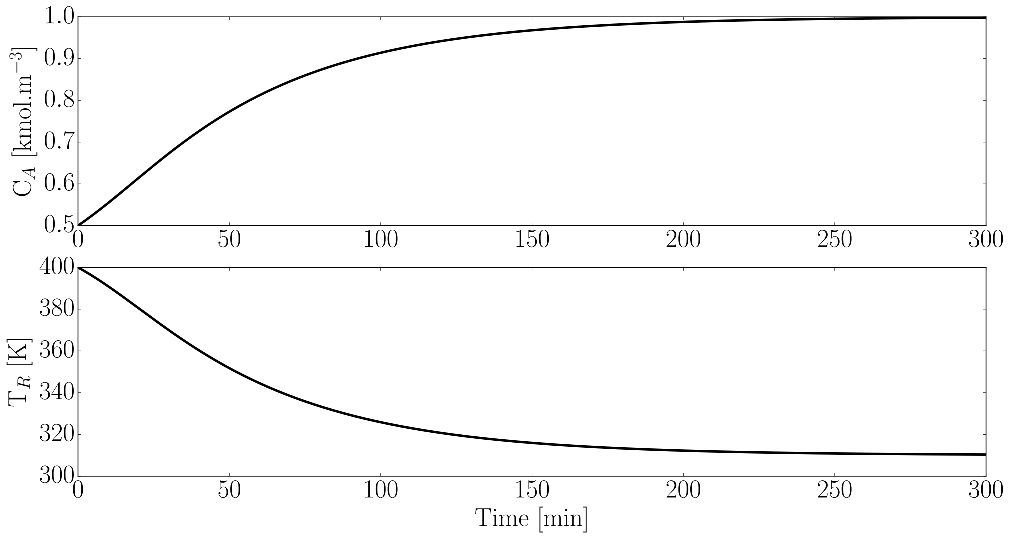

Figure 5.4: Transient response of the CSTR under nominal operating conditions with initial condition $(0.5, 400)$ and $h = 0.1$.

It is often desirable to linearise a nonlinear system about some point, usually the operating point, to simplify the model. Computationally this is advantageous because linear techniques (e.g. linear optimisation) are usually significantly faster than nonlinear techniques. Practically linearisation is only valid in a small region around the point of linearisation. If the system moves away from the linearisation point the linear approximation can become grossly inaccurate.

Based on figure 5.3, where the dynamics are fast, we can venture a guess that linearisation will be a bad approximation, except for a very small time period, of plant behaviour because the states will rapidly move away from the point of linearisation.

On the other hand, based on figure 5.4, we can venture a guess that linearisation will be a fair approximation of plant behaviour for a meaningful period of time because the dynamics are slow.

5.3 Linearised models

Linear approximations used for modelling and control are very common both in practice and literature [42]. Furthermore, the approach of using multiple piecewise affine (linear) functions for control, based on linearisation around critical points, has also been investigated in literature [23], [36]. Typically the state domain is discretised into regimes and the linear approximation of the model in each regime is used for control. We will also use linear models for the purposes of control. It is therefore prudent to investigate linearisation techniques.

First we present a linearisation technique. Consider an arbitrary point in the state space (C_A^*, T_R^*). Then

$$\begin{pmatrix} \dot{C}_A \\ \dot{T}_R \end{pmatrix} = \begin{pmatrix} f(C_A^*, T_R^*) \\ g(C_A^*, T_R^*) \end{pmatrix} + J(C_A^*, T_R^*)\left(\begin{pmatrix} C_A \\ T_R \end{pmatrix} - \begin{pmatrix} C_A^* \\ T_R^* \end{pmatrix}\right) \tag{5.6}$$

is the general linearised model around (C_A^*, T_R^*) using the notation of (5.1). It is often desirable to change the variables such that (5.6) has no constant terms. This change of variables, which holds even if the linearisation point is not a critical point of the model, is

$$\begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix} = \begin{pmatrix} C_A \\ T_R \end{pmatrix} - J(C_A^*, T_R^*)^{-1}\left(J(C_A^*, T_R^*)\begin{pmatrix} C_A^* \\ T_R^* \end{pmatrix} - \begin{pmatrix} f(C_A^*, T_R^*) \\ g(C_A^*, T_R^*) \end{pmatrix}_{Q=0}\right). \tag{5.7}$$

We then have

$$\frac{d}{dx}\begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix} = J(C_A^*, T_R^*)\begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix} + B(C_A^*, T_R^*)Q. \tag{5.8}$$

Note that the input term B originates from $\begin{pmatrix} f(C_A^*, T_R^*) \\ g(C_A^*, T_R^*) \end{pmatrix}$ and in (5.7) we specifically set it to zero so that it is not removed. We now use the bilinear transform (also known as Tustin's transform) to convert (5.8) into the discrete equation

$$\begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix}_{t+1} = A(C_A^*, T_R^*)\begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix}_t + B(C_A^*, T_R^*)Q. \tag{5.9}$$

Observe that (5.9) implicitly depends on the sampling time. Recall that Q is the heat input to the system. Note that we need to add back the offset we removed in the change of variables step (5.7) when we want to physically interpret the results of applying (5.9).

Next we briefly investigate the accuracy of the linear models. In all the subsequent figures the first subplot illustrates the accuracy of the linear model if the initial value is close to the point of linearisation and the second subplot illustrates the accuracy when the initial value is further away from the point of linearisation.

Figure 5.5 shows the state space response of the linear model which was linearised around the high temperature, low concentration stable operating point (C_A^1, T_R^1) as defined in table 5.2.

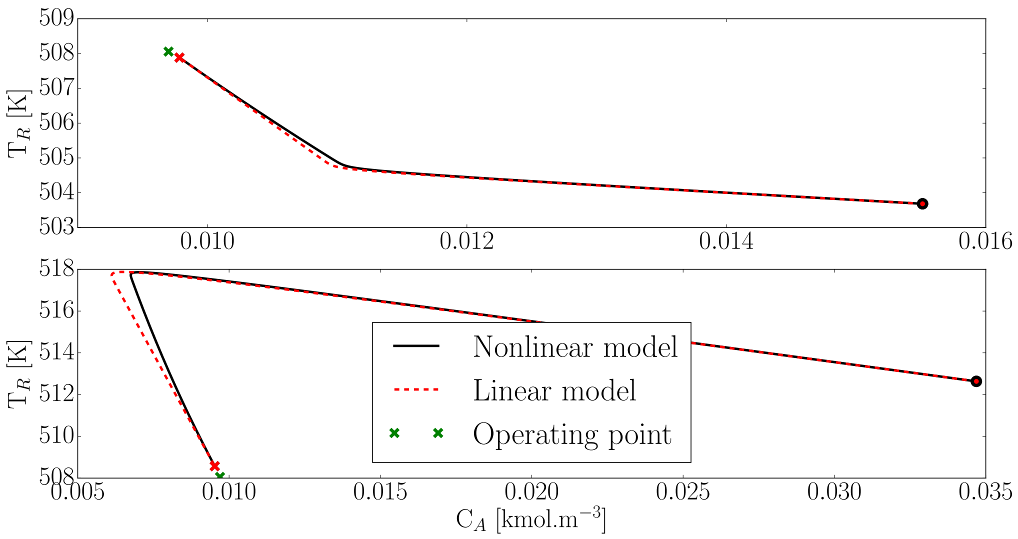

Figure 5.5: State space response of the CSTR under nominal operating conditions linearised around (C_A^1, T_R^1) with different initial conditions. The dot indicates where the simulation started and the cross where it finished. Total simulation time was 30 seconds.

We see that the linear approximation is quite accurate if the initial condition is close to the linearisation point (as expected). If the initial condition is further away the approximation is less accurate.

In figure 5.6 we see the state space response of the CSTR using the linear model linearised around the unstable operating point. Clearly this approximation is less accurate because the system tend to move away from operating point rather than towards it.

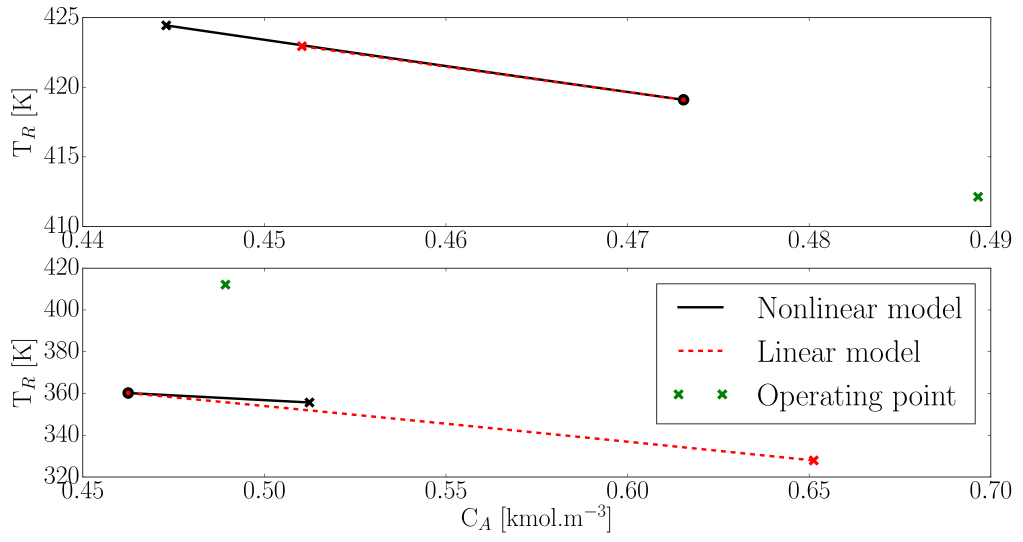

Figure 5.6: State space response of the CSTR under nominal operating conditions linearised around (C_A^2, T_R^2) with different initial conditions. The dot indicates where the simulation started and the cross where it finished. Total simulation time was 5 minutes.

If we want to use the linear model around the unstable operating point we will need to effect control to keep it within some region where the model is accurate.

In figure 5.7 we have the state space response of the CSTR under nominal conditions, like before, except that we have now linearised around the low temperature high concentration operating point.

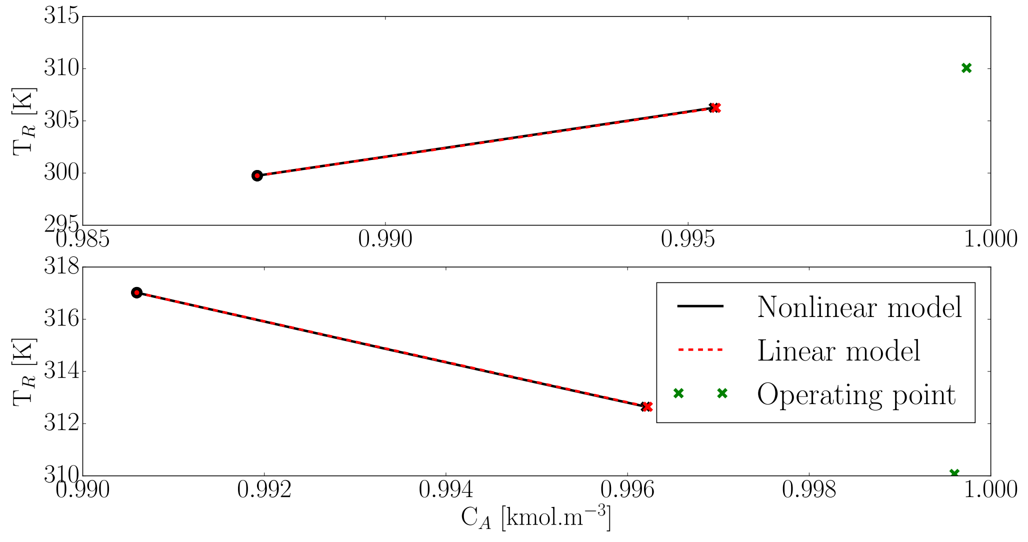

Figure 5.7: State space response of the CSTR under nominal operating conditions linearised around (C_A^3, T_R^3) with different initial conditions. The dot indicates where the simulation started and the cross where it finished. Total simulation time was 50 minutes.

Like figure 5.5, the linear model is quite accurate. The slow dynamics and stability of the operating point cause this desirable behaviour.

Based on the general linearisation formula in (5.6) there is no reason why one cannot linearise about an arbitrary point in the state space. It stands to reason that the more linear models at different linearisation points one has, the better one will be able to model the reactor. By selecting a model to use based on some metric (taking into account the current observation) it is reasonable to suppose that one will be able to model the system more accurately than if only one linear model was available. In part III we take this idea further.

However, in part II we restrict our attention to systems where one linear model is sufficient for our purposes.

Part II

Single model systems

Chapter 6

Inference using linear models

In this chapter we consider probabilistic graphical models of the form shown in figure 6.1. This model is a generalisation of the graphical model seen in chapter 4. We now assume that the states $(x_0, x_1, x_2, \ldots)$ and observations $(y_0, y_1, y_2, \ldots)$ are continuous random variables but the inputs $(u_0, u_1, u_2, \ldots)$ are continuous deterministic variables. Models of this form are called latent dynamical systems - if one assumes linearity and normality the famous Kalman filter model falls into this category.

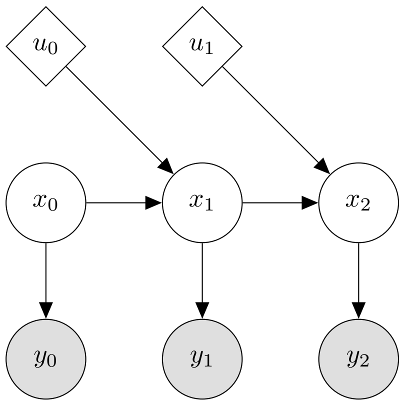

Figure 6.1: Graphical model considered in this chapter.

In chapter 4 we developed inference algorithms but assumed that the transition and observation functions were discrete. We also noted that this assumption is not appropriate for continuous systems. The reason is that one would invariably need to discretise the domain of the continuous random variable under consideration. This would result in intractably large discrete systems if one requires fine resolution. To address this issue we extend the previous model to include both continuous states and observations.

In this chapter we assume linearity and that all the random variables are Gaussian. While these are strong assumptions they form the building blocks of much more expressive models as we will discover in the next chapter. We also assume that the transition and observation functions are time invariant and that the state space model is of the form

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_{t+1} \text{ with } \mathcal{N}(w_{t+1}|0, W) \\ y_{t+1} &= Cx_{t+1} + v_{t+1} \text{ with } \mathcal{N}(v_{t+1}|0, V). \end{aligned} \tag{6.1}$$

From the graphical model in figure 6.1 we know that the latent variable x_t is observed through y_t. Rewriting the state space model we see that the transition and observation probability density functions are given by

$$\begin{aligned} p(x_{t+1}|x_t, u_t) &= \mathcal{N}(x_{t+1}|Ax_t + Bu_t, W) \\ p(y_{t+1}|x_t + 1) &= \mathcal{N}(y_{t+1}|Cx_{t+1}, V). \end{aligned} \tag{6.2}$$

We also assume that the system is first order Markov (see definition 3.23). We have implicitly assumed that the noise is Gaussian and white[1]. Intuitively one can think of V as the noise associated with state measurements and W being a form of the uncertainty associated with the linear model of the plant. Additionally, W can also model any zero mean unmeasured disturbances which may influence the system[2]. Thus, larger V and W indicate more uncertainty in the system.

To fully specify the system we require the transition and observation probability density functions (these implicitly depend on the internal structure of the graphical model in figure 6.1) as well as the prior (initial) distribution $p(x_0)$.

6.1 Kalman filter

The goal of filtering is to find the posterior distribution $p(x_t|y_{0:t}, u_{0:t-1})$. It is pleasing to note that this derivation will follow in an analogous manner to the filtering derivation in section 4.2.1 albeit with continuous Gaussian distributions. The motivation for taking the joint of only the preceding hidden time step is the same as before. The graphical model corresponding to filtering is shown in figure 6.2.

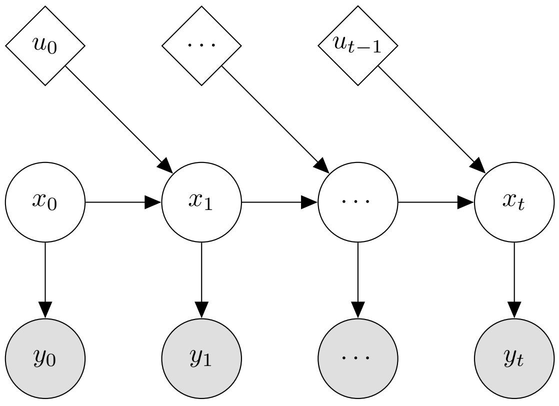

Figure 6.2: Extended - for illustration - graphical model used for filtering.

[1]The noise is temporally independent, has zero mean and finite variance.

[2]Note that for the purposes of this dissertation plant is a synonym for the system.

We start with the prediction expression $p(x_t|y_{0:t-1}, u_{0:t-1})$ and assume, due to the closure of linear conditional Gaussian distributions, that $\alpha(x_{t-1}) = p(x_{t-1}|y_{0:t-1}, u_{0:t-2}) = \mathcal{N}(x_{t-1}|\mu_{t-1}, \Sigma_{t-1})$ is available. This allows us to write

$$\begin{aligned}
p(x_t|y_{0:t-1}, u_{0:t-1}) &= \int_{x_{t-1}} p(x_t, x_{t-1}|y_{0:t-1}, u_{0:t-1}) \\
&= \int_{x_{t-1}} p(x_{t-1}|y_{0:t-1}, u_{0:t-1}) p(x_t|x_{t-1}, y_{0:t-1}, u_{0:t-1}) \\
&= \int_{x_{t-1}} p(x_{t-1}|y_{0:t-1}, u_{0:t-2}) p(x_t|x_{t-1}, u_{t-1}) \\
&= \int_{x_{t-1}} \alpha(x_{t-1}) \mathcal{N}(x_t|Ax_{t-1} + Bu_{t-1}, W) \\
&= \int_{x_{t-1}} \mathcal{N}(x_{t-1}|\mu_{t-1}, \Sigma_{t-1}) \mathcal{N}(x_t|Ax_{t-1} + Bu_{t-1}, W).
\end{aligned} \tag{6.3}$$

Now we use theorem 3.7 (Bayes' theorem for linear Gaussian models) to evaluate the marginal expression

$$\begin{aligned}
\int_{x_{t-1}} \mathcal{N}(x_{t-1}|\mu_{t-1}, \Sigma_{t-1}) \mathcal{N}(x_t|Ax_{t-1} + Bu_{t-1}, W) &= \mathcal{N}(x_t|A\mu_{t-1} + Bu_{t-1}, W + A^T\Sigma_{t-1}A) \\
&= \mathcal{N}(x_t|\mu_{t|t-1}, \Sigma_{t|t-1}).
\end{aligned} \tag{6.4}$$

Intuitively, (6.4) is the one step ahead prediction for the hidden state given all the past observations and the past and present inputs. Now we make use of theorem 3.4 (Bayes' theorem) to update our view of x_t given the current observation

$$\begin{aligned}
p(x_t|y_{0:t}, u_{0:t-1}) &= p(x_t|y_t, y_{0:t-1}, u_{0:t-1}) \\
&= \frac{p(y_t|x_t, y_{0:t-1}, u_{0:t-1}) p(x_t|y_{0:t-1}, u_{0:t-2}, u_{t-1})}{p(y_t|y_{0:t-1}, u_{0:t-1})} \\
&= \frac{p(y_t|x_t) p(x_t|y_{0:t-1}, u_{0:t-1})}{p(y_t|y_{0:t-1}, u_{0:t-1})} \\
&\propto p(y_t|x_t) p(x_t|y_{0:t-1}, u_{0:t-1}) \\
&= p(y_t|x_t) \mathcal{N}(x_t|A\mu_{t-1} + Bu_{t-1}, W + A^T\Sigma_{t-1}A) \\
&= \mathcal{N}(y_t|Cx_t, V) \mathcal{N}(x_t|\mu_{t|t-1}, \Sigma_{t|t-1}).
\end{aligned} \tag{6.5}$$

Now we again make use of theorem 3.7 to evaluate the conditional expression

$$\begin{aligned}
p(x_t|y_{0:t}, u_{0:t-1}) &= \mathcal{N}(y_t|Cx_t, V) \mathcal{N}(x_t|\mu_{t|t-1}, \Sigma_{t|t-1}) \\
&= \mathcal{N}(x_t|\Gamma(C^TV^{-1}y + \Sigma_{t|t-1}^{-1}\mu_{t|t-1}), \Gamma) \\
&\text{with } \Gamma = (\Sigma_{t|t-1}^{-1} + C^TV^{-1}C)^{-1}.
\end{aligned} \tag{6.6}$$

By using the matrix identity $(A + BD^{-1}C)^{-1} = A^{-1} - A^{-1}B(D + CA^{-1}B)^{-1}CA^{-1}$ and defining $K_t = \Sigma_{t|t-1}C^T(C\Sigma_{t|t-1}C^T + V)^{-1}$ we can simplify Γ to the recursive posterior covariance estimate

$$\Sigma_t = (I - K_tC)\Sigma_{t|t-1}. \tag{6.7}$$

Similarly, using the same matrix identity together with $(P^{-1}B^TR^{-1}B)^{-1})^{-1}B^TR^{-1} = PB^T(BPB^T + R^{-1})$ and the definition of K_t we have the posterior mean estimate

$$\mu_t = \mu_{t|t-1} + K_t(y_t - C\mu_{t|t-1}). \tag{6.8}$$

Together (6.7) and (6.8) are known as the Kalman filter equations [48]. For the first time step only the update expression is evaluated as the prediction is the prior of x_0.

Intuitively, the Kalman filter equations use the state space model to predict the new state distribution and then adjust it by a correction factor $K_t(y_t - C\mu_{t|t-1})$. This factor depends on the difference between the actual observation and the predicted observation. The Kalman gain, K_t, represents the inferred confidence of the model. If the model is deemed accurate then the predictions make up most of μ_t but if the model is bad at predicting the observations then the observations play a bigger part in the next mean estimate [8].

6.2 Kalman prediction

The goal of prediction is to find an expression for the distributions $p(x_{t+h}|y_{0:t}, u_{0:t+h-1})$ and $p(y_{t+h}|y_{0:t}, u_{0:t+h-1})$ with $h \geq 1$. Note that these derivations follow in exactly the same way as the prediction derivations did in section 4.2.4 given the current posterior state estimate of x_0. The reason for this is because the graphical models are the same (the deterministic inputs don't change the structure of the underlying random variable network). The graphical model corresponding to the two step ahead state prediction is shown in figure 6.3.

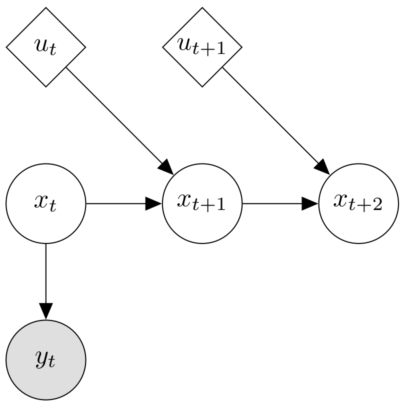

Figure 6.3: Graphical model used for state prediction.

We start the derivation by considering the one step ahead state prediction

$$\begin{aligned} p(x_{t+1}|y_{0:t}, u_{0:t}) &= \int_{x_t} p(x_{t+1}, x_t|y_{0:t}, u_{0:t}) \\ &= \int_{x_t} p(x_t|y_{0:t}, u_{0:t-1})p(x_{t+1}|x_t, y_{0:t}, u_{0:t}) \\ &= \int_{x_t} p(x_t|y_{0:t}, u_{0:t-1})p(x_{t+1}|x_t, u_t) \\ &= \int_{x_t} \alpha(x_t)p(x_{t+1}|x_t, u_t) \\ &= \int_{x_t} \mathcal{N}(x_t|\mu_t, \Sigma_t)\mathcal{N}(x_{t+1}|Ax_t + Bu_t, W) \\ &= \mathcal{N}(x_{t+1}|Ax_t + Bu_t, W + A\Sigma_t A^T) \\ &= \mathcal{N}(x_{t+1}|\mu_{t+1|t}, \Sigma_{t+1|t}). \end{aligned} \tag{6.9}$$

Note that μ_t and Σ_t is the filtered mean and covariance found by the Kalman filter. We have again relied upon theorem 3.7 to evaluate the marginal integral. We now consider the two step ahead state prediction

$$\begin{aligned} p(x_{t+2}|y_{0:t}, u_{0:t+1}) &= \int_{x_{t+1}} p(x_{t+2}, x_{t+1}|y_{0:t}, u_{0:t+1}) \\ &= \int_{x_{t+1}} p(x_{t+1}|y_{0:t}, u_{0:t})p(x_{t+2}|x_{t+1}, y_{0:t}, u_{0:t+1}) \\ &= \int_{x_{t+1}} p(x_{t+1}|y_{0:t}, u_{0:t})p(x_{t+2}|x_{t+1}, u_{t+1}) \\ &= \int_{x_t} \mathcal{N}(x_{t+1}|\mu_{t+1|t}, \Sigma_{t+1|t})\mathcal{N}(x_{t+2}|Ax_{t+1} + Bu_{t+1}, W) \\ &= \mathcal{N}(x_{t+2}|A\mu_{t+1|t} + Bu_{t+1}, W + A\Sigma_{t+1|t}A^T) \\ &= \mathcal{N}(x_{t+2}|\mu_{t+2|t}, \Sigma_{t+2|t}). \end{aligned} \tag{6.10}$$

It is clear that we have derived a recursive algorithm to estimate the h^{th}-step ahead state prediction as shown in

$$\begin{aligned} &p(x_{t+h}|y_{0:t}, u_{0:t+h}) = \mathcal{N}(x_{t+h}|\mu_{t+h|t}, \Sigma_{t+h|t}) \\ &\text{with } \mu_{t+h|t} = A\mu_{t+h-1|t} + Bu_{t+h-1} \\ &\text{and } \Sigma_{t+h|t} = W + A\Sigma_{t+h-1|t}A^T \\ &\text{and } \mu_{t+1|t} = A\mu_t + Bu_t \\ &\text{and } \Sigma_{t+1|t} = W + A\Sigma_t A^T. \end{aligned} \tag{6.11}$$

Inspecting (6.11) we see that the predictive distribution is just the forward projection, using the transition function, of the filtered distribution. Note that it is possible for $\Sigma_{t+h|t}$ to become smaller, in some normed ($|\cdot|$) sense, for increasing h (obviously bounded by Q below). For, if the eigenvalues of A are less than unity we have that $|A\Sigma_{t+h|t}A^T| \leq |A\Sigma_{t+h-1|t}A^T|$.

Next we consider the observation prediction, $p(y_{t+h}|y_{0:t}, u_{0:t+h-1})$. Again consider the one step ahead prediction

$$\begin{aligned}
p(y_{t+1}|y_{0:t}, u_{0:t}) &= \int_{x_t, x_{t+1}} p(y_{t+1}, x_{t+1}, x_t|y_{0:t}, u_{0:t}) \\
&= \int_{x_t, x_{t+1}} p(x_t|y_{0:t}, u_{0:t-1}) p(y_{t+1}, x_{t+1}|x_t, y_{0:t}, u_{0:t}) \\
&= \int_{x_t, x_{t+1}} p(x_t|y_{0:t}, u_{0:t-1}) p(x_{t+1}|x_t, y_{0:t}, u_{0:t}) p(y_{t+1}|x_{t+1}, x_t, y_{0:t}, u_{0:t}) \\
&= \int_{x_t, x_{t+1}} \alpha(x_t) p(x_{t+1}|x_t, u_t) p(y_{t+1}|x_{t+1}) \\
&= \int_{x_t, x_{t+1}} \mathcal{N}(x_t|\mu_t, \Sigma_t) \mathcal{N}(x_{t+1}|Ax_t + Bu_t, W) \mathcal{N}(y_{t+1}|Cx_{t+1}, V) \\
&= \mathcal{N}(y_{t+1}|C\mu_{t+1|t}, V + C\Sigma_{t+1|t}C^T).
\end{aligned} \tag{6.12}$$

We have again used theorem 3.7 and used the nomenclature of the one step ahead state prediction derivation. For the sake of brevity we trust that the reader will see the similarity between the two derivations and allow us to conclude, without proof, that the h^{th}-step ahead observation prediction is

$$p(y_{t+h}|y_{0:t}, u_{0:t+h-1}) = \mathcal{N}(y_{t+h}|C\mu_{t+h|t}, R + C\Sigma_{t+h|t}C^T). \tag{6.13}$$

It is reassuring to note that the observation prediction is just the state prediction transformed by the observation function.

6.3 Smoothing and Viterbi decoding

For the sake of completeness we state the Kalman smoothing equations and briefly discuss Viterbi decoding within the context of conditional linear Gaussian systems.

The reason we do not go into detail with the smoothing algorithm is because it follows much the same structure as the hidden Markov model smoothing algorithm (see section 4.2.2) except that we make use of theorem 3.7 to simplify the algebra. We are also primarily only interested in filtering and prediction because they are important for the purposes of control which is the focus of this dissertation.

The smoothing algorithm, also called the Rauch, Tung and Striebel (RTS) algorithm, for $p(x_t|y_{0:T}, u_{0:T-1})$ is also a Gaussian distribution of the form $\mathcal{N}(\hat{\mu}_t, \hat{\Sigma}_t)$. The recursion expressions for the posterior mean and covariance are

$$\begin{aligned}
&\hat{\mu}_t = \mu_t + J_t\left(\hat{\mu}_{t+1} - (A\mu_t + Bu_{t-1})\right) \\
&\hat{\Sigma}_t = \Sigma_t + J_t(\hat{\Sigma}_{t+1} - P_t)J_t^T \\
&\text{with } P_t = A\Sigma_t A^T + W \\
&\text{and } J_t = \Sigma_t A^T (P_t)^{-1} \\
&\text{and } \hat{\mu}_T = \mu_T \\
&\text{and } \hat{\Sigma}_T = \Sigma_T.
\end{aligned} \tag{6.14}$$

Finally, we know from the definition 3.19 (chain rule for Bayesian networks) and figure 6.1 that the joint distribution for $p(x_{0:T}, y_{0:T}, u_{0:T-1}) = p(x_1)p(y_1|x_1)\Pi_{t=2}^{T} p(y_t|x_t)p(x_t|x_{t-1}, u_{t-1})$. Since Gaussian distributions are closed under multiplication this joint distribution is also a Gaussian distribution. It can be shown that maximising with respect to all latent variables jointly or maximising with respect to the marginal distributions of the latent variables is the same because the mean and the mode of a Gaussian distribution coincide [4]. This implies that Viterbi decoding is just the sequence of means found by the smoothing algorithm.

6.4 Filtering the CSTR

In this chapter we apply the Kalman filter to the nonlinear CSTR introduced in chapter 5. We use the linear model linearised around the unstable operating point (C_A^2, T_R^2) as shown in (6.15) to describe the nonlinear underlying model. Note that the matrix A and vectors B, b depend on the step size and should be recalculated for different h. To make things concrete we have used $h = 0.1$ here. Note that V indicates that we only measure temperature for now.

$$\begin{aligned} &A = \begin{pmatrix} 0.9959 & -6.0308 \times 10^{-5} \\ 0.4186 & 1.0100 \end{pmatrix} \quad B = \begin{pmatrix} 0 \\ 8.4102 \times 10^{-5} \end{pmatrix} \quad C = \begin{pmatrix} 0 & 1 \end{pmatrix} \\ &W = \begin{pmatrix} 1 \times 10^{-6} & 0 \\ 0 & 0.1 \end{pmatrix} \quad V = \begin{pmatrix} 10 \end{pmatrix} \end{aligned} \tag{6.15}$$

The system noise W indicates that the standard deviation of the concentration component of the model is 0.001 kmol/m^{-3} and the temperature component is 0.32 K. While these variances may seem small, bear in mind that noise is added at each time step which compounds its effect. The measurement noise implies that 68% of the measurements will fall between $\pm\sqrt{10}$ of the actual state. We use an initial state with mean at the initial condition and covariance W.

The focus of this dissertation is on the application of probabilistic graphical models to control, therefore our investigation into the various aspects which improve or degrade filtering performance will be relatively superficial and will target factors which are most relevant only. In this chapter we will primarily only investigate the benefit gained by including more state measurements.

Before we begin it is prudent to introduce the metric we will use to quantify filter performance. We define the average estimation error by $\frac{1}{N}\sum_{t=0}^{N} |\frac{\hat{x}_t - x_t}{x_t}|$ where $\hat{x}_t$ is the inferred state and and x_t the true underlying state at time t.

In figure 6.4 we illustrate the strengths and weaknesses of the Kalman filter. Since we derived the recursion equations analytically it is computationally efficient to use, the biggest cost is a matrix inversion which needs to be computed at each time step[3]. During the initial part

[3]It is even possible to avoid this step by noticing that the posterior covariance quickly converges to a constant covariance which can be precomputed off line.

of the simulation the filter very accurately estimates the current system states because the model is accurate in this region. Thus the filter is able to infer the true state in the presence of noisy measurements.

Unfortunately the recursion equations assumed that the system can be described by a linear model throughout the state space. With time the trajectories move away from the linearisation point (because the linearisation point is unstable) and thus the linear model becomes less accurate. This has a detrimental effect on the quality of the Kalman filter estimate as the filter effectively starts to solely rely on the measurements to infer the states. This works reasonably well for the measured states (T_R), but since we do not measure concentration the filter is forced to incorporate the linear model prediction which is grossly inaccurate.

The average concentration estimation error throughout the run is 22.73% while the average temperature estimation error is 0.47%. Clearly there is a significant benefit to measuring the state one wishes to infer.

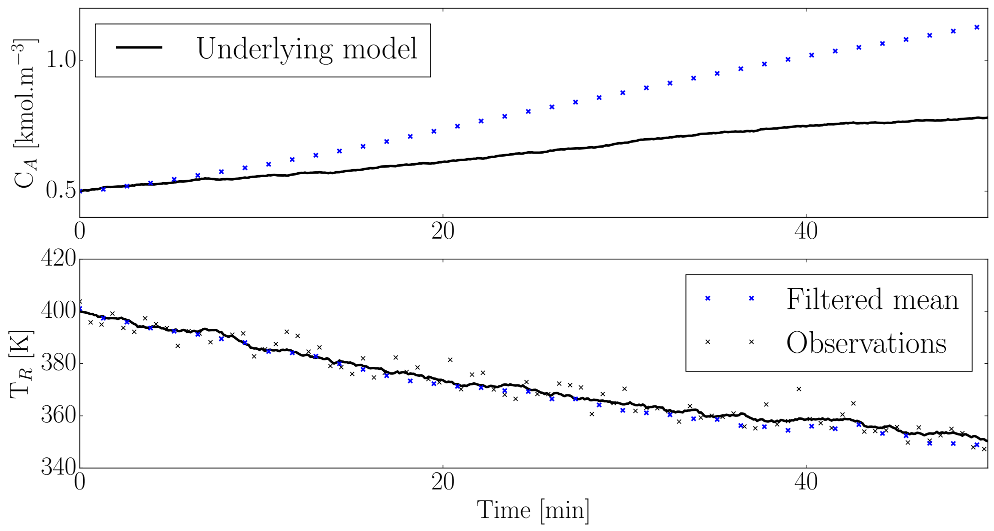

Figure 6.4: Kalman filter superimposed on the time series evolution of the CSTR with initial condition $(0.50, 400)$ and measuring only temperature.

In figure 6.5 we see another interesting property of Kalman filters. The posterior covariance quickly converges to a constant value (the confidence region quickly stops changing shape/size) which is independent of the observations. This is a general property of linear Gaussian systems [4] and is evident from the recursion expression. The modelled system dynamics and noise are the only factors affecting the covariance. If the model is accurate this is not a problem but we see that as the model becomes less accurate the filter maintains the same level of confidence in its estimate. This is quite undesirable behaviour because the confidence in the estimate is not a function of the observations.

It is also interesting to consider the shape of the confidence region. Notice that it is short vertically - indicating less uncertainty in the temperature state dimension but wide horizontally - indicating more uncertainty in the concentration state dimension. Intuitively this is plausible because, since we do not measure concentration, we are less sure about the underlying state.

In figure 6.5 we see that while the temperature estimate is still trustworthy (the temperature mean estimate lines up horizontally with the true underlying state) the concentration estimate diverges.

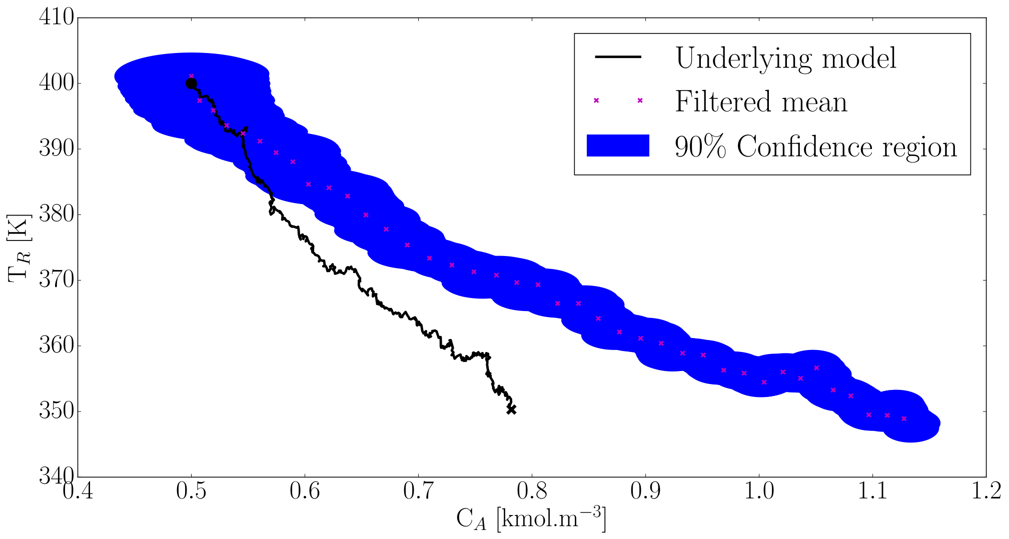

Figure 6.5: State space diagram of the CSTR with mean and 90% confidence region superimposed thereupon. Only temperature is measured.

The root of the problem lies in the unsuitability of the model rather than our inference technique. It can be shown that for linear systems with Gaussian noise the Kalman filter is the optimal state estimator [2].

Based on our discussion in chapter 5, where the CSTR example was introduced, we know that the linear models will not be accurate in regions far removed from their linearisation points. We therefore modify (6.15) to also incorporate concentration measurements to offset this weakness. In this case we have that $C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and $V = \begin{pmatrix} 1 \times 10^{-3} & 0 \\ 0 & 10 \end{pmatrix}$ with everything else the same. The time evolution of the states is shown in figure 6.6 and the state space representation is shown in figure 6.7.

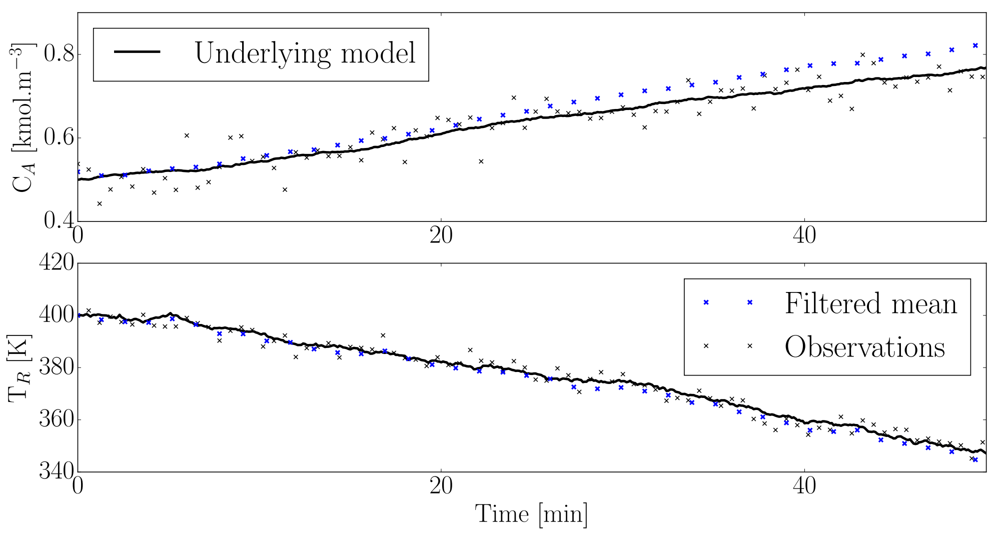

Figure 6.6: Kalman filter superimposed on the time series evolution of the CSTR with initial condition $(0.50, 400)$ and measuring both temperature and concentration.

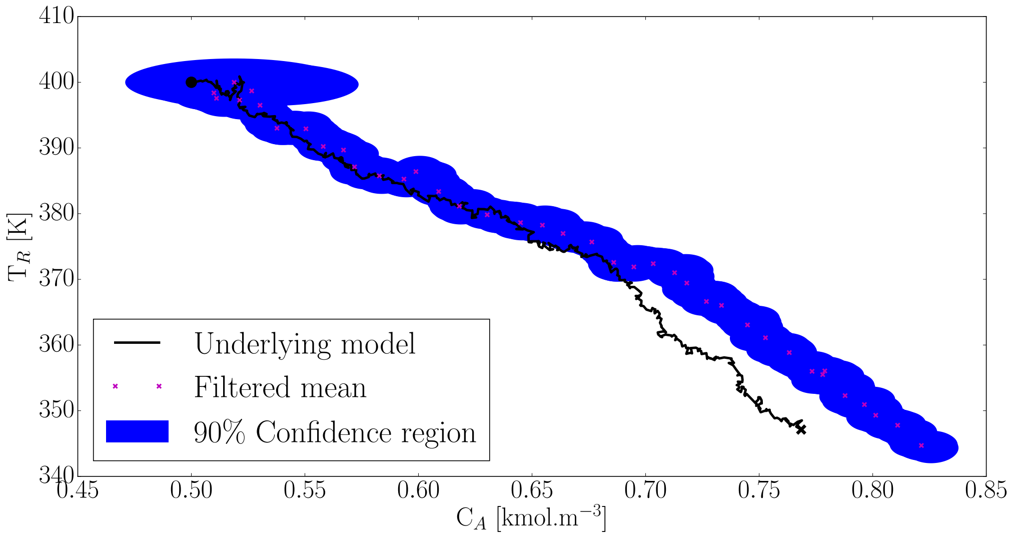

Figure 6.7: State space diagram of the CSTR with mean and 90% confidence region superimposed thereupon. Both concentration and temperature are measured.

Comparing figures 6.5 and 6.7 we see that by incorporating the state measurement the state estimation is much more accurate. The average concentration and temperature estimation error is only 4.09% and 0.45% respectively. It is not necessary to directly measure concentration as we have done: any measurement which depends on C_A (or even both C_A and T_R) would suffice. The second measurement reduces our uncertainty in the concentration state estimate because we have more to base our inference on than just a bad model.

In the next chapter we drop the linearity and normality assumptions at the cost of the closed form, exact inference solutions. Sections 6.1 and 6.2 form the basis of chapter 8 and we encourage the reader to familiarise themselves with it.

Chapter 7

Inference using nonlinear models

In this chapter we consider probabilistic graphical models of the form shown in figure 7.1. These models have exactly the same form as the models in chapter 6. The variables retain their meaning as before but we generalise the model by dropping the linearity assumption. Unfortunately, this generalisation, although allowing us to expand our investigation to a much more expressive class of models, makes closed form solutions to the inference problem intractable in general.

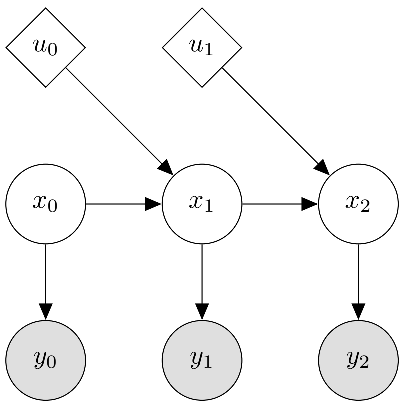

Figure 7.1: Graphical model used in this chapter.

We again assume that the transition and observation functions are time invariant. The state space model is now of the form

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, w_{t+1}) \\ y_{t+1} &= g(x_{t+1}, v_{t+1}). \end{aligned} \tag{7.1}$$

As in chapter 6, we observe y_t to infer the latent x_t. Note that we make no assumption about the functional form of the noise terms w_t, v_t. In practice it is customary to assume that they have zero mean but otherwise are not restricted. Additionally, to simplify notation we will omit the dependence on u of f and g and their associated distributions. Since u is a deterministic variable, by assumption, it is straightforward to incorporate it into later analysis.

7.1 Sequential Monte Carlo methods

Many approximate inference techniques exist in literature, the most notable ones include Gaussian sum filters [32] and particle based methods. We shall focus only on sequential Monte Carlo methods, of which particle based methods are a subset, because it is simple to implement and generalises well (and easily) to more complex graphical models.

Sequential Monte Carlo methods are a general class of Monte Carlo methods which sample sequentially from a growing target distribution $\pi_t(x_{0:t})$. By only requiring that γ_t be known point-wise we have the framework of sequential Monte Carlo methods in

$$\begin{aligned}\pi_t(x_{0:t}) &= \frac{\gamma_t(x_{0:t})}{Z_t} \\ Z_t &= \int_{x_{0:t}} \gamma_t(x_{0:t}).\end{aligned} \tag{7.2}$$

Note that Z_t is some normalisation constant [21]. For example, in the context of filtering we have that $\gamma_t(x_{0:t}) = p(x_{0:t}, y_{0:t})$ and $Z_t = p(y_{0:t})$ so that $\pi_t(x_{0:t}) = p(x_{0:t}|y_{0:t})$.

It is possible to approximate the distribution $\pi_t(x_{0:t})$ by drawing N samples $X^i_{0:t} \backsim \pi_t(x_{0:t})$ and using the Monte Carlo method to find the approximation $\hat{\pi}_t(x_{0:t})$ by

$$\pi_t(x_{0:t}) \approx \hat{\pi}_t(x_{0:t}) = \frac{1}{N}\sum_{i=1}^{N} \delta(X^i_{0:t}, x_{0:t}). \tag{7.3}$$

We denote the Dirac delta function of x with mass located at x_0 by $\delta(x_0, x)$. It is easy to approximate the marginal $\pi_t(x_t)$ as

$$\hat{\pi}_t(x_t) = \frac{1}{N}\sum_{i=1}^{N} \delta(X^i_t, x_t). \tag{7.4}$$

It can be shown that the variance of the approximation error of π_t decreases at rate $\mathcal{O}(\frac{1}{N})$. Unfortunately there are two significant drawbacks to the Monte Carlo approximation. The first is that often we cannot sample from $\pi_t(x_{0:t})$ directly and the second is that even if we could it is often computationally prohibitive.

We use the importance sampling method to address the first problem. We do this by introducing an importance (sometimes called proposal) density $q_t(x_{0:t})$ such that $\pi_t(x_{0:t}) > 0 \implies q_t(x_{0:t}) > 0$. By substituting this into the sequential Monte Carlo framework (7.2) we have

$$\begin{aligned}\pi_t(x_{0:t}) &= \frac{w_t(x_{0:t})q_t(x_{0:t})}{Z_t} \\ Z_t &= \int_{x_{0:t}} w_t(x_{0:t})q_t(x_{0:t}).\end{aligned} \tag{7.5}$$

Where we have defined the unnormalised weight function $w_t(x_{0:t}) = \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t})}$. It is possible, for example, to set q_t to a multivariate Gaussian which is easy to sample from. By drawing N

samples $X^i_{0:t} \backsim q_t(x_{0:t})$ and using (7.5) we have

$$\begin{aligned}\hat{\pi}_t(x_{0:t}) &= \frac{1}{N}\sum_{i=1}^{N} W^i_t \delta(X^i_{0:t}, x_{0:t}) \\ \hat{Z}_t &= \frac{1}{N}\sum_{i=1}^{N} w_t(X^i_{0:t}) \\ W^i_t &= \frac{w_t(X^i_{0:t})}{\sum_{i=1}^{N} w_t(X^i_{0:t})}.\end{aligned} \tag{7.6}$$

Now we will attempt to modify the importance sampling method to address the second problem of computational cost incurred by the sampling routine.

We do this by selecting an importance/proposal distribution which factorises according to $q_t(x_{0:t}) = q_{t-1}(x_{0:t-1})q_t(x_t|x_{0:t-1}) = q_0(x_0)\Pi^t_{k=1}q_k(x_k|x_{0:k-1})$. In this way we only need to sample sequentially at each time step: at time $t = 0$ we sample $X^i_0 \backsim q_0(x_0)$, at time $t = 1$ we sample $X^i_1 \backsim q_1(x_1|x_0)$ and so we build up $X^i_{0:t} \backsim q_t(x_{0:t})$ factor by factor.

The weights can be written in the form

$$\begin{aligned}w_t(x_{0:t}) &= \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t})} \\ &= \frac{\gamma_{t-1}(x_{0:t-1})}{q_{t-1}(x_{0:t-1})}\frac{\gamma_t(x_{0:t})}{\gamma_{t-1}(x_{0:t-1})q_t(x_t|x_{0:t-1})} \\ &= w_{t-1}(x_{0:t-1})\alpha_t(x_{0:t-1}) \\ &= w_0(x_0)\Pi^t_{k=1}\alpha_k(x_{0:k}).\end{aligned} \tag{7.7}$$

Thus, at any time t we can obtain the estimates $\hat{\pi}_t(x_{0:t})$ and Z_t. The major limitation of this approach is that the variance of the resulting estimates typically increases exponentially with t [21].

We overcome this problem by resampling and thus introduce the sequential importance resampling method. So far we have a set of weighted samples generated from $q_t(x_{0:t})$ which builds the approximation $\hat{\pi}_t(x_{0:t})$. However, sampling directly from $\hat{\pi}_t(x_{0:t})$ does not approximate $\pi_t(x_{0:t})$. To obtain an approximate distribution of $\pi_t(x_{0:t})$ we need to sample from the weighted distribution $\hat{\pi}_t(x_{0:t})$. This is called resampling because we are sampling from a sampled distribution. Many techniques exist to perform this step efficiently. The crudest and most widely used one is to simply use the discrete multinomial distribution based on $W^i_{0:t}$ to draw samples from $\hat{\pi}_t(x_{0:t})$ [21].

The benefit of resampling is that it allows us to remove particles with low weight and thus keeps the variance of the estimate in check. We are finally ready to consider the general sequential importance resampling algorithm:

Sequential importance resampling algorithm

For $t = 0$:

1. Sample $X^i_0 \backsim q_0(x_0)$.

2. Compute the weights $w_0(X_0^i)$ and $W_0^i \propto w_0(X_0^i)$.

3. Resample (W_0^i, X_0^i) to obtain N equally weighted particles $(\frac{1}{N}, \bar{X}_0^i)$.

For $t \geq 1$:

1. Sample $X_t^i \backsim q_t(x_t|\bar{X}_{0:t-1}^i)$ and set $X_{0:t}^i \leftarrow (\bar{X}_{0:t-1}^i, X_t^i)$.

2. Compute the weights $\alpha_t(X_{0:t}^i)$ and $W_t^i \propto \alpha_t(X_{0:t}^i)$.

3. Resample $(W_t^i, X_{0:t}^i)$ to obtain N equally weighted particles $(\frac{1}{N}, \bar{X}_{0:t}^i)$.

At any time t we have two approximations for $\pi(x_{0:t})$:

$$\begin{aligned} \hat{\pi}(x_{0:t}) &= \sum_{i=1}^{N} W_t^i \delta(X_{0:t}^i, x_{0:t}) \\ \bar{\pi}(x_{0:t}) &= \frac{1}{N}\sum_{i=1}^{N} \delta(\bar{X}_{0:t}^i, x_{0:t}). \end{aligned} \tag{7.8}$$

The latter approximation represents the resampled estimate and the former represents the sampled estimate [21]. We prefer the former because in the limit as $N \to \infty$ it is a better approximation of π_t. However, as we have mentioned the variance of $\hat{\pi}(x_{0:t})$ tends to be unbounded and thus we often have that most of the particles in the particle population have very low weight. From a computational point of view this is wasteful. To ameliorate this we use the latter, resampled, estimate. However, the problem with the resampled estimate is that it effectively culls low weight particles and this reduces the diversity of the particle population [47].

We attempt to get the benefit of both worlds by only performing resampling when the weight variance of the particles becomes large. The effective sample size (ESS) is a method whereby one determines when to perform resampling according to

$$\text{ESS} = \frac{1}{\sum_{i=1}^{N}(W_n^i)^2}. \tag{7.9}$$

If the effective sample size becomes smaller than some threshold (typically $\frac{N}{2}$) we resample to cull low weight particles and replace them with high weight particles. In this manner we have a computationally feasible method. This is called adaptive resampling and is a straightforward extension of the sequential Monte Carlo algorithm as shown below.

Adaptive sequential importance resampling algorithm

For $t = 0$:

1. Sample $X_0^i \backsim q_0(x_0)$.

2. Compute the weights $w_0(X_0^i)$ and $W_0^i \propto w_0(X_0^i)$.

3. If resample criterion is satisfied then resample (W_0^i, X_0^i) to obtain N equally weighted particles $(\frac{1}{N}, \bar{X}_0^i)$ and set $(\bar{W}_0^i, \bar{X}_0^i) \leftarrow (\frac{1}{N}, \bar{X}_0^i)$ otherwise set $(\bar{W}_0^i, \bar{X}_0^i) \leftarrow (W_0^i, X_0^i)$.

For $t \geq 1$:

1. Sample $X_t^i \backsim q_t(x_t|\bar{X}_{0:t-1}^i)$ and set $X_{0:t}^i \leftarrow (\bar{X}_{0:t-1}^i, X_t^i)$.

2. Compute the weights $\alpha_t(X_{0:t}^i)$ and $W_t^i \propto \bar{W}_{t-1}^i \alpha_t(X_{0:t}^i)$.

3. If the resample criterion is satisfied then resample $(W_t^i, X_{0:t}^i)$ to obtain N equally weighted particles $(\frac{1}{N}, \bar{X}_{0:t}^i)$ and set $(\bar{W}_1^i, \bar{X}_t^i) \leftarrow (\frac{1}{N}, \bar{X}_t^i)$ otherwise set $(\bar{W}_t^i, \bar{X}_t^i) \leftarrow (W_t^i, X_t^i)$.

Numerous convergence results exist for the sequential Monte Carlo methods we have discussed but the fundamental problem with this scheme is that of sample impoverishment. It is fundamentally impossibly to accurately represent a distribution on a space of arbitrarily high dimension with a finite set of samples [21]. We attempt to mitigate this problem by using resampling but degeneracy inevitably occurs for large enough t. Fortunately, for our purposes we will not be dealing with arbitrarily large dimensional problems because of the Markov assumption.

7.2 Particle filter

We now apply the adaptive sequential importance resampling algorithm in the setting of filtering. We set $\pi_t(x_{0:t}) = p(x_{0:t}|y_{0:t})$, $\gamma_t(x_{0:t}) = p(x_{0:t}, y_{0:t})$ and consequently $Z_t = p(y_{0:t})$. We use the recursive proposal distribution $q_t(x_{0:t}|y_{0:t}) = q(x_t|x_{0:t-1}, y_{0:t})q_{t-1}(x_{0:t-1}|y_{0:t-1})$. We then have the unnormalised weights

$$\begin{aligned} w_t(x_{0:t}) &= \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t}|y_{0:t})} \\ &= \frac{p(x_{0:t}, y_{0:t})}{q_t(x_{0:t}|y_{0:t})} \\ &\propto \frac{p(x_{0:t}|y_{0:t})}{q_t(x_{0:t}|y_{0:t})} \\ &\propto \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q_t(x_t|x_{0:t-1}, y_{0:t})} \frac{p(x_{0:t-1}|y_{0:t-1})}{q_{t-1}(x_{0:t-1}|y_{0:t-1})} \\ &= \alpha_t(x_{0:t})w_{t-1}(x_{0:t-1}). \end{aligned} \tag{7.10}$$

For filtering we only care about $p(x_t|y_{0:t})$ and thus we do not need the entire trajectory $x_{0:t}$. This allows us to choose the proposal distribution $q_t(x_t|x_{0:t-1}, y_{0:t}) = q_t(x_t|x_{t-1}y_t)$. In this case the incremental weight α_t simplifies to

$$\alpha_t(x_{0:t}) = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q_t(x_t|x_{t-1}, y_t)}. \tag{7.11}$$

The most common proposal distribution is, the suboptimal, $q_t(x_t|x_{t-1}|y_t) = p(x_t|x_{t-1})$ because it is easy to sample from. This implies that the incremental weights simplify to $\alpha_t(x_{0:t}) = p(y_t|x_t)$. Using such a proposal distribution was initially proposed in [29] in the setting of the non-adaptive sequential importance resampling method.

For general purpose filtering this is not very efficient because it amounts to "guessing until you hit". If the transitions are very stochastic inference can be improved by using the optimal proposal distribution $q_t(x_t|x_{t-1}, y_t) = p(x_t|x_{t-1}, y_t)$. While this is optimal it introduces some difficulty because, in general, it is more difficult to sample from. The focus of dissertation is not on optimal filtering and for the purposes of prediction the suggested proposal distribution is sufficiently good [47]. We thus restrict ourselves to the proposal distribution $p(x_t|x_{t-1})$ for simplicity.

Finally, we have mentioned that resampling kills off unlikely particles. An unfortunate consequence of this is that some particle diversity is lost; this is sometimes called sample impoverishment. An empirical method used to attenuate this problem is to resample from a kernel around the particle selected by the resampling process. This is called roughening in [29]. We thus make a final modification to the adaptive sequential importance resampling algorithm. We select a particle from the population in the standard way but resample from a normal distribution centred around that particle and with a diagonal covariance matrix where the standard deviation of each diagonal is $KEN^{-\frac{1}{d}}$. We define E as the range of the particle's relevant component, N as the number of particles and d as the dimension of the problem. K is a tuning factor which specifies how broad the kernel we sample from should be.

For the sake of completeness we present the particle filter algorithm we used here. Recall that f and g are the transition and observation functions respectively. The algorithm is applied to each particle $i = 1, 2, ..., N$.

Particle filter algorithm

For $t = 0$:

1. Sample $X_0^i \backsim p(x_0)$.

2. Compute the weights $w_0(X_0^i) = p(y_0|X_0^i) = \mathcal{N}(y_0|g(X_0^i), V)$ where y_0 is the observation. Normalise $W_0^i \propto w_0(X_0^i)$.

3. If the number of effective particles is below some threshold apply resampling with roughening (W_0^i, X_0^i) to obtain N equally weighted particles $(\frac{1}{N}, \bar{X}_0^i)$ and set $(\bar{W}_0^i, \bar{X}_0^i) \leftarrow (\frac{1}{N}, \bar{X}_0^i)$ otherwise set $(\bar{W}_0^i, \bar{X}_0^i) \leftarrow (W_0^i, X_0^i)$

For $t \geq 1$:

1. Sample $X_t^i = f(\bar{X}_{t-1}^i, w_t) \backsim p(x_t|\bar{X}_{t-1}^i, W)$.

2. Compute the weights $\alpha_t(X_t^i) = p(y_t|X_t^i) = \mathcal{N}(y_t|g(X_t^i), V)$ and normalise $W_t^i \propto \bar{W}_{t-1}^i \alpha_t(X_t^i)$.

3. If the number of effective particles is below some threshold apply resampling with roughening (W_t^i, X_t^i) to obtain N equally weighted particles $(\frac{1}{N}, \bar{X}_t^i)$ and set $(\bar{W}_1^i, \bar{X}_t^i) \leftarrow (\frac{1}{N}, \bar{X}_t^i)$ otherwise set $(\bar{W}_t^i, \bar{X}_t^i) \leftarrow (W_t^i, X_t^i)$.

The algorithm presented above is a slight generalisation of the bootstrap particle filter as initially proposed by Gordon et. al. [29].

Intuitively the algorithm may be summarised like this: particle filters predict the next hidden state by projecting all the current particles forward using the transition function and associated noise. For each particle the likelihood of the observation is calculated given the particle and measurement noise. This likelihood is related to the weight of each particle. Particles with a relatively high weight are then deemed to more accurately represent the posterior distribution and thus we infer the posterior state estimate based on the relative weights of each particle. The graphical model of particle filtering is exactly the same as that of the Kalman filter graphical model as shown in figure 6.2. This should not come as a surprise because the general graphical model of this chapter, figure 7.1, is exactly the same as the general graphical model of chapter 6 as shown in figure 6.1.

7.3 Particle prediction

We are primarily interested in predicting the future hidden states but we also show how the future visible states may be predicted within the framework of particle based methods. Recalling the prediction derivations of chapter 4 and chapter 6 we expect the hidden state prediction to merely be an n step ahead projection of the current filtered particles. Likewise, we expect the visible state prediction to just be transformation of the predicted hidden states under the observation function.

Inspecting the bootstrap particle filter algorithm presented in section 7.2 we are relieved to find that this is the case. One just removes the observation update steps (steps 2 and 3) from the algorithm because we cannot observe the future. We illustrate the two step ahead predictions and trust that the reader can generalise from here.

Particle Prediction Algorithm

1. Sample $X^i_{t+1} = f(\bar{X}^i_t, w_{t+1}) \backsim p(x^i_{t+1}|y_t, \bar{X}^i_t)$
2. Project $X^i_{t+2} = f(X^i_{t+1}, w_{t+2}) \backsim p(x^i_{t+2}|y_t, X^i_{t:t+1})$
3. Project $Y^i_{t+2} = g(X^i_{t+2}, v_{t+2}) \backsim p(y^i_{t+2}|y_t, X^i_{t:t+1})$

Once again, the graphical model depicting this situation is exactly the same as figure 6.3 for the same reasons as mentioned before.

7.4 Smoothing and Viterbi decoding

In the context of nonlinear transition and observation functions smoothing and Viterbi decoding are much more difficult than before. For the purposes of this dissertation it is not important to consider inferences of that type and thus we merely refer the reader to literature where this is discussed [4], [21], [32], [47] and [48].

7.5 Filtering the CSTR

In this chapter we apply the particle filter to the nonlinear CSTR problem introduce in chapter 5. We first demonstrate the effectiveness of the particle filter by performing inference using the full nonlinear CSTR model measuring only temperature. Next we use the full nonlinear model again but measure both temperature and concentration. Finally, we compare the particle filter and the Kalman filter measuring both states. These investigations are by no means thorough but serve to illustrate important aspects of probabilistic graphical models which will affect control.

We do not investigate the effect the number of particles used for inference has on the particle filter. It is well known that increasing the number of particles increases the accuracy of particle based methods [47] but at the cost of increased computational complexity. The number of particles used in this dissertation reflects this trade-off i.e. we use a relatively small number of particles so that the simulations run quickly but are still accurate enough for practical purposes.

Although it is not necessary we assume that the process and measurement noise is Gaussian with the same distributions as those found in section 6.4. This holds for all the other parameters as well. We have used 200 particles to represent the state posterior. In figure 7.2 we see the state estimates as a function of time.

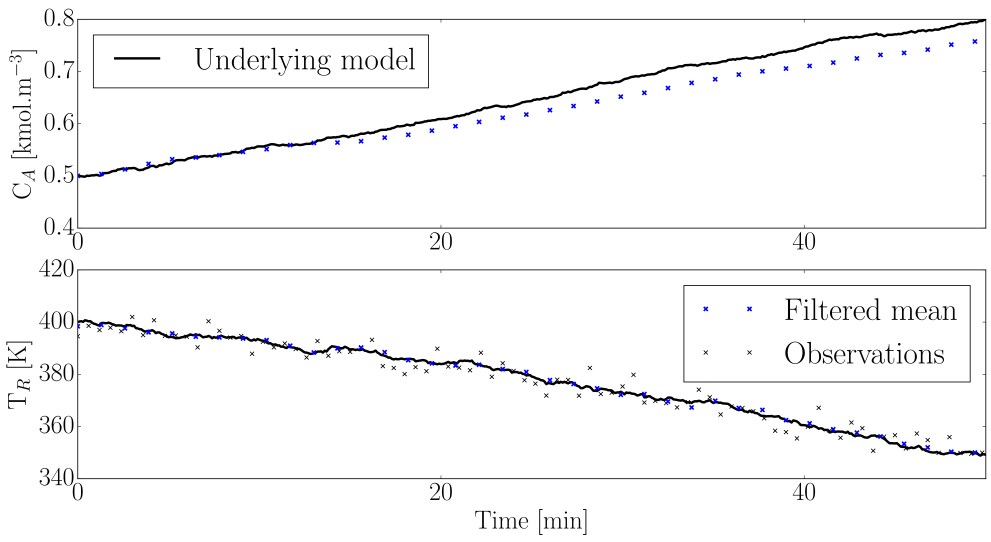

Figure 7.2: Time series state estimates using the particle filter on the nonlinear CSTR model with initial condition $(0.5, 400)$ and measuring only temperature. The filter uses 200 particles.

The filter tracks both states reasonable well with a little more variance evident in the unmeasured state. The benefit of using the full nonlinear model is evident here - since the model is more accurate than the previously used linear model the filter infers the concentration more accurately. The average concentration and temperature estimation error is 3.15% and 0.20% respectively. Compare this to 22.73% and 0.47% over the same simulation time using a Kalman filter measuring only temperature. The increased accuracy is also reflected in the state space evolution curve in figure 7.3.

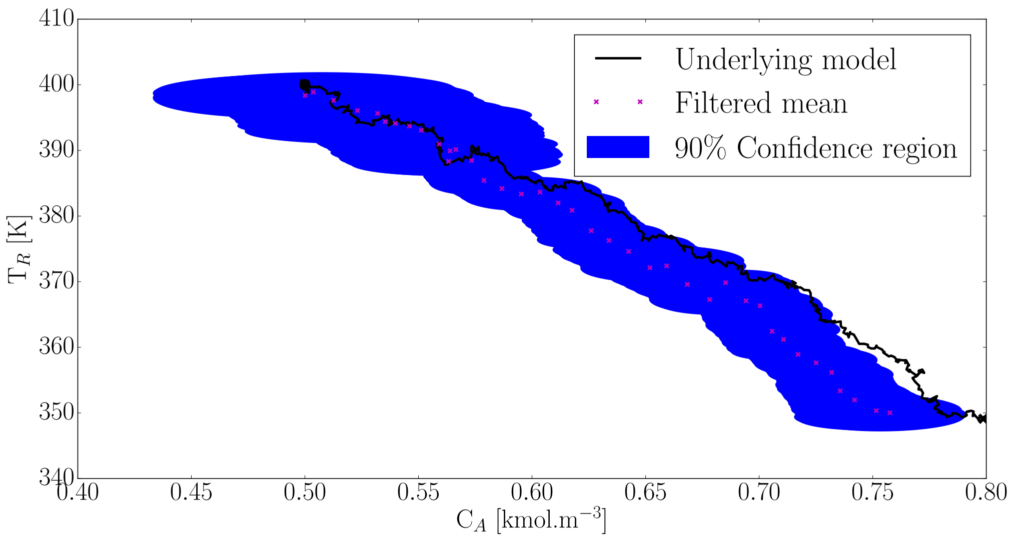

Figure 7.3: State space evolution of the particle filter on the nonlinear CSTR model with initial condition $(0.5, 450)$ and measuring only temperature. The filter uses 200 particles.

We also see in figure 7.3 that the variance of the estimates is quite high (the confidence region is quite big). We expect that by also measuring concentration this will decrease. In figures 7.4 and 7.5 we incorporate concentration measurement to aid inference.

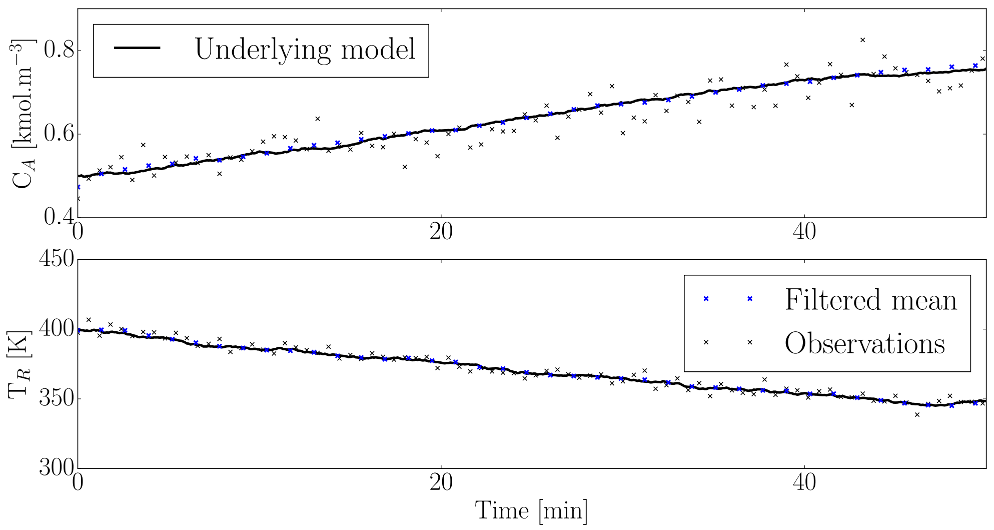

Figure 7.4: Time series state estimates using the particle filter on the nonlinear CSTR model with initial condition $(0.5, 450)$ and measuring both states. The filter uses 200 particles.

It is clear that that the particle filter reliably tracks the state evolution in the presence of plant and measurement noise. The average concentration and temperature estimation error is 0.81% and 0.21% respectively. We see that by also measuring the concentration the size of the confidence region decreases in figure 7.5.

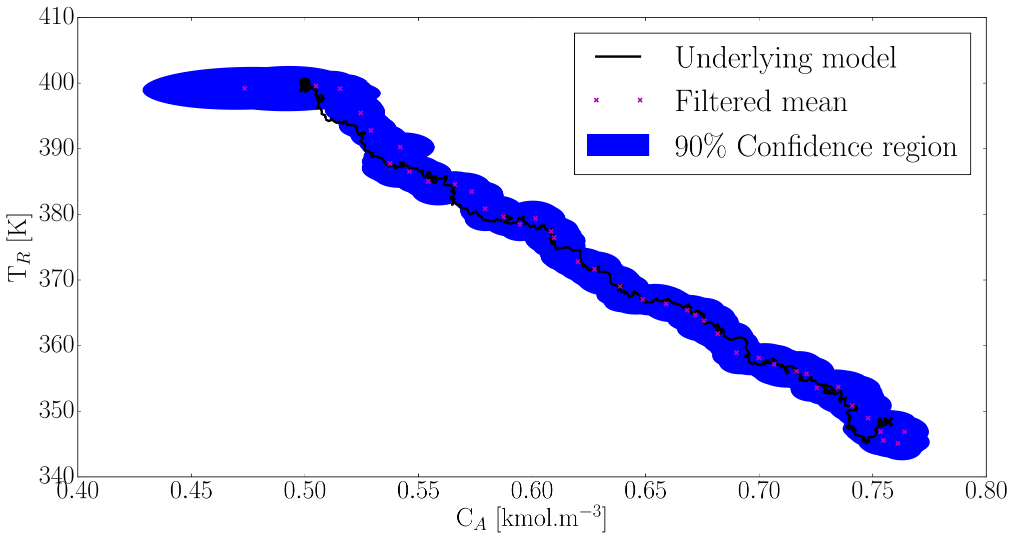

Figure 7.5: State space evolution of the particle filter on the nonlinear CSTR model with initial condition $(0.5, 450)$ and measuring both states. The filter uses 200 particles.

Finally we compare the particle filter to the Kalman filter using both temperature and concentration measurements. First we illustrate that if the underlying model is linear and the noise Gaussian the particle filter does no better than the Kalman filter. In figure 7.6 we see that both the particle filter and the Kalman filter are able to accurately estimate the posterior state distribution over time. Note that we have used 500 particles to meaningfully compare the distribution estimates (the more particles one uses in the particle filter the more accurate it becomes).

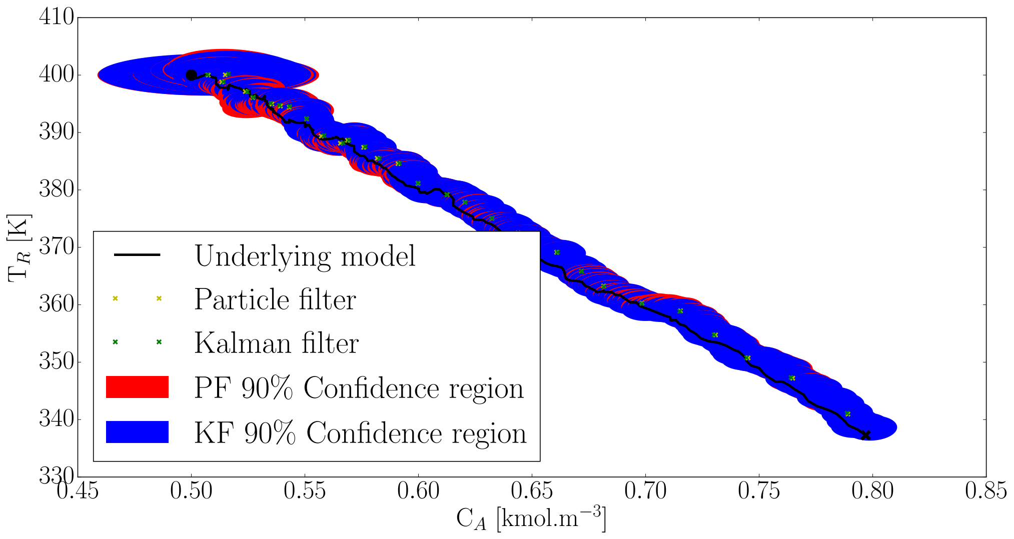

Figure 7.6: State space evolution of the particle filter and the Kalman filter on the linear CSTR model with initial condition $(0.5, 400)$ and measuring both temperature and concentration. The particle filter uses 500 particles.

The average concentration and temperature estimation errors for the particle filter is 0.93% and 0.23% respectively while the corresponding estimation errors for the Kalman filter is 0.97% and 0.23% respectively. Since the confidence region overlaps throughout the entire simulation it is clear that if the underlying model is linear there is no great difference between the two filters from an accuracy point of view. It does however makes sense, from a computational point of view, to use the Kalman filter: it is well known that the particle filter does not perform well in high dimensional problems [56].

Next we consider the same comparison but change the underlying model to the full nonlinear CSTR as shown in figure 7.7. The average concentration and temperature estimation errors for the particle filter is 0.83% and 0.19% respectively while the corresponding estimation errors for the Kalman filter is 4.50% and 0.41% respectively.

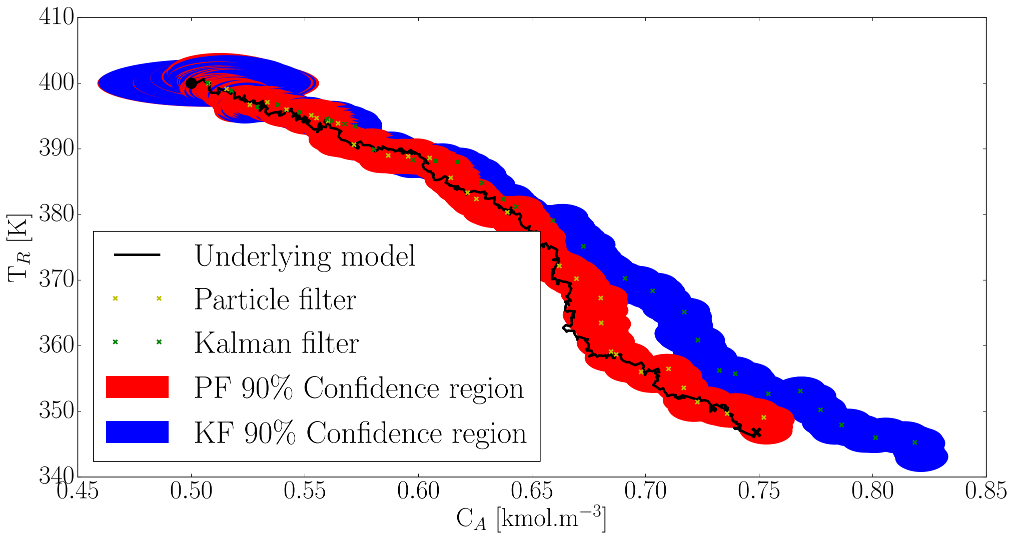

Figure 7.7: State space evolution of the particle filter and the Kalman filter on the non-linear CSTR model with initial condition $(0.5, 400)$ and measuring both temperature and concentration. The particle filter uses 500 particles.

Inspecting figure 7.7 we see that throughout the simulation the particle filter's confidence region is smaller. Since we are using a significantly more accurate model for the particle filter this is not surprising. Additionally we see that the Kalman filter state estimates diverge from the true states as the model moves away from the region where the linear model is accurate. The same weakness in the Kalman filter was discussed in section 6.4 concerning the usage of the linear model.

Therefore, while the particle filter may be computationally more expensive to use it is a better filter if the system exhibits nonlinearity or non-normality. But, if the system is linear and normal one is better off using the standard Kalman filter.

In the next chapter we design model predictive controllers from within the framework of probabilistic graphical models.

Chapter 8

Stochastic linear control

In this chapter we consider the stochastic reference tracking problem. It is required to move the states and manipulated variables of the system,

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + w_{t+1} \\ y_{t+1} &= g(x_{t+1}) + v_{t+1}, \end{aligned} \tag{8.1}$$

to the set point (x_{sp}, u_{sp}) by manipulating the input variables u. It is assumed that x_t is a latent stochastic variable and y_t is an observed stochastic variable.

We assume uncorrelated zero mean additive Gaussian noise (w_t and v_t) in both the state function f and the observation function g with known covariances W and V respectively. Clearly it is not possible to achieve perfect control (zero offset at steady state) because of the noise terms, specifically w_t. For this reason we need to relax the set point goal a little bit. We will be content if our controller is able to achieve definition 8.1.

Definition 8.1. Stochastic reference tracking goal: Suppose we have designed a controller and set $\delta > 0$ as a controller benchmark. If there exists some positive number $t^* < \infty$ such that $\forall\, t > t^*$ the controller input causes $\mathbb{E}[(x_t - x_{sp})^T Q (x_t - x_{sp}) + (u_t - u_{sp})^T R (u_t - u_{sp})] < \delta$ we will have satisfied the stochastic reference tracking goal given δ.

While definition 8.1 is pleasing from a theoretical point of view, it is not easy to design a controller to specifically satisfy a given δ. We again simplify our goal somewhat: we will be content if the controller we design (without a specific δ in mind) can satisfy definition 8.1 for some suitably small resultant δ. Intuitively, we would like the mean of the states and inputs to be "close enough" to the set point.

In this chapter we limit ourselves by only considering controllers developed using a single linear model of the underlying, possibly nonlinear, system functions f and g. The linearised model control is based upon is

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_{t+1} \\ y_{t+1} &= Cx_{t+1} + v_{t+1} \end{aligned} \tag{8.2}$$

and is subject to the same noise as (8.1). We will endeavour to develop predictive controllers using the graphical models of chapters 6 and 7.

8.1 Unconstrained stochastic control

Our first goal is to solve the problem

$$\begin{aligned}&\min_{\mathbf{u}} J_{LQG}(x_0, \mathbf{u}) = \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t\end{aligned} \tag{8.3}$$

given the current state estimate x_0[1]. If the system is controllable then solving (8.3) will satisfy the linear unconstrained stochastic reference tracking goal.

Note that the future inputs $\mathbf{u} = (u_0, u_1, \dots, u_{T-1})$ are denoted in boldface to emphasise that it could be a vector of vectors. Inspecting (8.3) we see that this is none other than the LQG control problem of section 3.4.3. Therefore we know what the optimal solution should look like.

We start our analysis using the results of chapter 6. We assume that at every sequential time step we have the current state estimate, which is assumed to be Gaussian, and denote this by x_0.

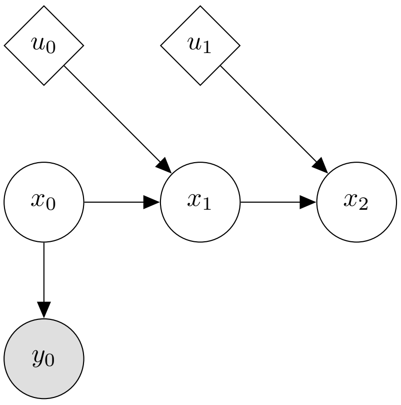

Figure 8.1: Graphical model for state prediction.

Inspecting figure 8.1 we note that the state prediction equations derived in section 6.2 are applicable. Thus we can predict the state distributions given the future inputs $\mathbf{u}$.

Before we proceed we prove a very intuitive result in theorem 8.1. We will use this to link the predictive controller we derive here, using the results of sections 6.1 and 6.2, to the LQR controller derived in section 3.4.1.

[1]It is customary to assign $x_0 \leftarrow x_t$ at each time step to simplify the controller optimisation problem's notation.

Theorem 8.1. Optimisation Equivalence Suppose we have two real valued convex objective functions $f(x_0, \mathbf{u})$ and $g(x_0, \mathbf{u})$ and we are required to minimise them with respect to $\mathbf{u}$ over the same space where they are both defined: $\mathbf{u} \in \mathcal{U}$ and $x_0 \in \mathcal{X}$. Furthermore, suppose there exists a real number k such that $\forall \mathbf{u} \in \mathcal{U}$ we have that $g(x_0, \mathbf{u}) + k = f(x_0, \mathbf{u})$. Finally, assume the existence and uniqueness of the global minimiser for each problem. Then the global minimiser $\mathbf{u}^*$ of $g(x_0, \mathbf{u})$ is also the global minimiser of $f(x_0, \mathbf{u})$.

Proof. This proof only holds over functions which are at least twice differentiable. By assumption we know that $\mathbf{u}^*$ is the minimiser of $g(x_0, \mathbf{u})$ given x_0. By the necessary conditions for optimality [27] we know that $\nabla g(x_0, \mathbf{u}^*) = 0$ and that $\nabla^2 g(x_0, \mathbf{u}^*)$ is positive semi-definite. Since f and g are both twice differentiable and $g(x_0, \mathbf{u}^*) + k = f(x_0, \mathbf{u}^*)$ it must hold that $\nabla g(x_0, \mathbf{u}^*) = \nabla f(x_0, \mathbf{u}^*)$ and $\nabla^2 g(x_0, \mathbf{u}^*) = \nabla^2 f(x_0, \mathbf{u}^*)$. Since $\nabla^2 g(x_0, \mathbf{u}^*)$ is positive semi-definite it must be that $\nabla^2 f(x_0, \mathbf{u}^*)$ is also positive semi-definite. Therefore $\mathbf{u}^*$ is necessarily a minimum of f. Since f is convex the minimum must also be a global minimum. $\square$

Now we are in a position to show the equivalence between the LQR control problem and the LQG control problem using the results of sections 6.1 and 6.2. Theorem 8.2 shows how this is possible. It is quite reassuring to note that by starting within the framework of graphical models we arrive at the most important contribution of [64] and [65] in an intuitively simple manner.

Theorem 8.2. LQR and LQG objective function difference Consider the LQR,

$$\begin{aligned} J_{LQR}(x_0, \mathbf{u}) &= \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N \\ \text{with } x_{t+1} &= A x_t + B u_t, \end{aligned} \tag{8.4}$$

and LQG,

$$\begin{aligned} J_{LQG}(x_0, \mathbf{u}) &= \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\ \text{with } x_{t+1} &= A x_t + B u_t + w_{t+1}, \end{aligned} \tag{8.5}$$

objective functions. Suppose x_0 is the state estimate, assumed to be Gaussian, given the latest observation in the stochastic case. In the deterministic case we have that $x_0 = \mathbb{E}[x_0] = \mu_0$ because we exactly observe the state. Given any input sequence $\mathbf{u} \in \mathcal{U}$, where $\mathcal{U}$ is the shared admissible input space, we have that $J_{LQR}(x_0, \mathbf{u}) + \frac{1}{2}\sum_{k=0}^{N} \operatorname{tr}(Q\Sigma_k) = J_{LQG}(x_0, \mathbf{u})$ where $\Sigma_{t+1} = W + A\Sigma_t A^T$ and Σ_0 is the covariance matrix of the current state given by the observer.

Proof. Expanding the LQG objective function (the expected value operator is linear) and noting that $\mathbf{u}$ is deterministic we have

$$\begin{aligned} J_{LQG}(x_0, \mathbf{u}) = &\frac{1}{2}\mathbb{E}\left[x_0^T Q x_0\right] + \frac{1}{2}u_0^T R u_0 + \frac{1}{2}\mathbb{E}\left[x_1^T Q x_1\right] + \frac{1}{2}u_1^T R u_1 + \dots \\ &+ \frac{1}{2}\mathbb{E}\left[x_{N-1}^T Q x_{N-1}\right] + \frac{1}{2}u_{N-1}^T R u_{N-1} + \frac{1}{2}\mathbb{E}\left[x_N^T P_f x_N\right]. \end{aligned} \tag{8.6}$$

We know that $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ because the current state estimate was assumed to be Gaussian. This means that we can evaluate the first expected value in (8.6) using theorem 3.5 to write

$$\mathbb{E}\left[x_0^T Q x_0\right] = \text{tr}(Q\Sigma_0) + \mu_0^T Q \mu_0. \tag{8.7}$$

Now we turn our attention to the second expected value in (8.6). First note that because we have x_0 and $\mathbf{u}$ we can use the result from section 6.2 to predict the distribution of x_1. Therefore we know that $x_1 \sim \mathcal{N}(A\mu_0 + Bu_0, W + A\Sigma_0 A^T)$. Now we let $\mu_1 = A\mu_0 + Bu_0$ and $\Sigma_0 = W + A\Sigma_0 A^T$. Then by using theorem 3.5 we have

$$\mathbb{E}\left[x_1^T Q x_1\right] = \text{tr}(Q\Sigma_1) + \mu_1^T Q \mu_1. \tag{8.8}$$

Note that $\text{tr}(Q\Sigma_1)$ does not depend on u_0 but only on the initial state estimate x_0 which is independent of the future inputs $\mathbf{u}$. Notice that we can continue in this manner to simplify the LQG objective function to

$$\begin{aligned} J_{LQG}(x_0, \mathbf{u}) &= \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q \mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N + \frac{1}{2}\sum_{k=0}^{N}\text{tr}(Q\Sigma_k) \\ \text{with } & \mu_{t+1} = A\mu_t + Bu_t \\ \text{and } & \Sigma_{t+1} = W + A\Sigma_t A^T. \end{aligned} \tag{8.9}$$

Now note that except for the last term $J_{LQG}(x_0, \mathbf{u})$ is exactly the same as $J_{LQR}(x_0, \mathbf{u})$. The conclusion follows because $\frac{1}{2}\sum_{k=0}^{N}\text{tr}(Q\Sigma_k)$ is independent of $\mathbf{u}$. $\square$

Finally we combine theorems 8.1 and 8.2 to produce theorem 8.3.

Theorem 8.3. Solution of the finite horizon LQG control problem We wish to solve the LQG control problem within the framework of graphical models. The full problem is

$$\begin{aligned} \min_{\mathbf{u}} V(x_0, \mathbf{u}) &= \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\ \text{subject to } & x_{t+1} = Ax_t + Bu_t + w_t. \end{aligned} \tag{8.10}$$

We assume that x_0 is the current posterior state estimate, which is Gaussian, supplied by some observer. The solution of (8.10) is equivalent to solving the LQR problem with initial state equal to the mean of the initial state estimate supplied by some observer.

Proof. We assume that we have a Gaussian posterior state estimate for x_0. We use theorem 8.2 to prove that given x_0 and $\forall \mathbf{u} \in \mathcal{U}$ we have that $J_{LQR}(x_0, \mathbf{u}) + \frac{1}{2}\sum_{k=0}^{N}\text{tr}(Q\Sigma_k) = J_{LQG}(x_0, \mathbf{u})$ with $\frac{1}{2}\sum_{k=0}^{N}\text{tr}(Q\Sigma_k) \in \mathbb{R}$ a constant depending only on x_0. Thus we can use theorem 8.1 to prove that we only need to solve for the optimal controller input $\mathbf{u}$ using the LQR objective function to solve (8.10). $\square$

As we have mentioned before, the separation theorem (sometimes called certainty equivalence) implies that the solution of the LQG control problem is achieved by using the Kalman filter to optimally estimate the current state and then using that state estimate in the optimal

LQR controller. It is reassuring that theorem 8.3 is confirmed by this result if we use the Kalman filter as the observer. The primary benefit of the graphical model approach is clear: we have solved the LQG problem without resorting to stochastic dynamical programming.

Under some circumstances it is also possible to extend the result of theorem 8.3 to the infinite horizon case as shown in theorem 8.4.

Theorem 8.4. Solution of the infinite horizon LQG control problem If the linear model of (8.2) is stable then, using, with some minor adjustments, theorems 8.1 and 8.2 it is possible to show that the infinite horizon LQG problem is solved in a similar manner: a Gaussian state estimate is used in conjunction with the infinite horizon LQR solution. This result can also be obtained by using the separation theorem.

To clarify why it is important that the linear system (i.e. the matrix A) is stable, consider the quantity $\frac{1}{2}\sum_{k=0}^{N}\text{tr}(Q\Sigma_k)$. If it is unbounded the optimisation problem will be ill posed because the minimum will tend to some infinity. Inspecting $\Sigma_{t+1} = W + A\Sigma_t A^T$ we see that $||\Sigma_\infty||$ will be unbounded if $||A\Sigma_t A^T||$ becomes unbounded (W is a constant) as $t \to \infty$. Note that $||\cdot||$ is some matrix norm. It can be shown that only if the eigenvalues of A are less than unity i.e. the linear model is stable, then $||A\Sigma_{t+1}A^T|| \leq ||A\Sigma_t A^T||$ which implies that $\frac{1}{2}\sum_{k=0}^{N}\text{tr}(Q\Sigma_k)$ is bounded and the optimisation is reasonable.

8.2 Constrained stochastic control

The goal of this section is to solve the stochastically constrained MPC optimisation problem

$$\begin{aligned}
&\min_{\mathbf{u}} \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\
&\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t \qquad (8.11)\\
&\text{and } \mathbb{E}[d^T x_t + e] \geq 0 \; \forall \; t = 1, \ldots, N \\
&\text{and } \Pr(d^T x_t + e \geq 0) \geq p \; \forall \; t = 1, \ldots, N.
\end{aligned}$$

We assume that the underlying system is linear and the probability distributions are Gaussian. From the results of chapter 6 the probability distributions will be Gaussian if the system dynamics are linear. However, it is well known [43] that MPC is not in general a linear controller. From an analytical point of view this is problematic. We assume that the nonlinearity introduced by the MPC is negligible. We also restrict our analysis to affine constraints. Note that we only include one affine constraint in the succeeding examples however, the multiple constraint generalisation is addressed in the theory and are handled in exactly the same way as the examples using a single constraint. Furthermore, we assume that the current state estimate x_0 is supplied by some observer.

It might seem that the last constraint is a duplicate of the preceding one. Closer inspection reveals their different character. The first inequality constraint, which we will call the expected value constraint, requires that the predicted states satisfy the constraint "on average" while the second inequality constraint, which we will call the chance constraint, requires that the predicted states satisfy the constraint with at least some probability p.

For navigational convenience we supply a link to theorem 8.8 - when the reader reaches that point the reason will become obvious.

Theorem 8.5 succinctly shows that it is simple to convert the expected value constraint in (8.11) to a linear deterministic constraint.

Theorem 8.5. Affine expected value constraints Suppose we have a stochastic variable x with a known Gaussian distribution and we also have $d \in \mathbb{R}^n$ and $e \in \mathbb{R}$. Then the stochastic constraint $\mathbb{E}[d^T x + e] \geq 0$ simplifies to the deterministic constraint $d^T \mu + e \geq 0$ where $\mathbb{E}[x] = \mu$ is the mean of the stochastic variable.

Furthermore, suppose we have the set of n stochastic constraints $\mathbb{E}[Dx + e] \geq 0$. In this case $D \in \mathbb{R}^{n \times m}$ and $e \in \mathbb{R}^n$. Then $\mathbb{E}[Dx + e] \geq 0$ simplifies to $D\mu + e \geq 0$.

Proof. We know that x is a Gaussian stochastic variable. By theorem 3.5 we know that $\mathbb{E}[d^T x + e] = d^T \mu + e$ and $\mathbb{E}[Dx + e] = D\mu + e$. This immediately implies the result. □

It is interesting to pause here for a moment and consider theorem 8.3 and theorem 8.5 applied to a problem of the form

$$\begin{aligned} &\min_{\mathbf{u}} \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2} x_N^T P_f x_N\right] \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t \\ &\text{and } \mathbb{E}[d^T x_t + e] \geq 0 \ \forall \ t = 1, \ldots, N. \end{aligned} \tag{8.12}$$

We assume that the current state estimate is available at each time step, that it is Gaussian and that $\mathbb{E}[x_0] = \mu_0$. It is clear that by applying theorems 8.3 and 8.5 it is possible to rewrite (8.12) as

$$\begin{aligned} &\min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q \mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N + \frac{1}{2}\sum_{k=0}^{N} \text{tr}(Q\Sigma_k) \\ &\text{subject to } \mu_{t+1} = A\mu_t + Bu_t \\ &\text{and } d^T \mu_t + e \geq 0 \ \forall \ t = 1, \ldots, N. \end{aligned} \tag{8.13}$$

This implies that the standard deterministic MPC problem (8.13) is equivalent to the stochastic MPC problem with affine expected value constraints (8.12) under the assumptions of linearity and normality. This suggests that if chance constraints are not required the standard deterministic MPC will be sufficient for the control of stochastic processes. The generalisation of (8.12) to multiple expected value constraints is straightforward due to the second part of theorem 8.5.

Theorem 8.6 is a necessary step before we can convert the chance constraint of (8.11) into a nonlinear deterministic constraint.

Theorem 8.6. Shortest squared Mahalanobis distance between a hyperplane and a point Suppose we are given a symmetric positive semi-definite matrix S and a point y. The shortest squared Mahalanobis distance between y and the hyperplane $b^Tx + c = 0$ is given by $\frac{(b^Ty+c)^2}{b^TSb}$.

Proof. It is natural to formulate theorem 8.6 as an optimisation problem

$$\begin{aligned}&\min_x \; (x-y)^TS^{-1}(x-y)\\ &\text{subject to } b^Tx + c = 0.\end{aligned} \tag{8.14}$$

Note that S and therefore also S^{-1} is symmetric. Using conventional calculus we have $\nabla f(x) = (S^{-1} + S^{-1^T})x - 2S^{-1}y = 2S^{-1}x - 2S^{-1}y$ and $\nabla g(x) = b^T$. Using the method of Lagrangian multipliers [27] we have the system of equations

$$\begin{aligned}2S^{-1}x - 2S^{-1}y + \lambda b &= 0\\ b^Tx + c &= 0.\end{aligned} \tag{8.15}$$

This can be rewritten in block matrix form

$$\begin{pmatrix}2S^{-1} & b\\ b^T & 0\end{pmatrix}\begin{pmatrix}x\\ \lambda\end{pmatrix} = \begin{pmatrix}2S^{-1}y\\ -c\end{pmatrix}. \tag{8.16}$$

The special structure of the left hand side matrix in (8.16) allows us to analytically compute the inverse (see theorem 3.13 in section 3.5)

$$\begin{pmatrix}2S^{-1} & b\\ b^T & 0\end{pmatrix}^{-1} = \begin{pmatrix}\frac{1}{2}S(I - \frac{bb^TS}{b^TSb}) & \frac{Sb}{b^TSb}\\ \frac{b^TS}{b^TSb} & -\frac{2}{b^TSb}\end{pmatrix}. \tag{8.17}$$

To find the arguments which satisfy (8.15) we solve

$$\begin{pmatrix}\frac{1}{2}S(I - \frac{bb^TS}{b^TSb}) & \frac{Sb}{b^TSb}\\ \frac{b^TS}{b^TSb} & -\frac{2}{b^TSb}\end{pmatrix}\begin{pmatrix}2S^{-1}y\\ -c\end{pmatrix} = \begin{pmatrix}S(I - \frac{bb^TS}{b^TSb})S^{-1}y - c\frac{Sb}{b^TSb}\\ 2(\frac{b^TS}{b^TSb} + \frac{c}{b^TSb})\end{pmatrix} \tag{8.18}$$

which is equivalent to solving the system of linear equations in (8.16). Therefore, the arguments which minimise (8.14) are $x^* = S(I - \frac{bb^TS}{b^TSb})S^{-1}y - c\frac{Sb}{b^TSb}$. Substituting this into the objective function we have

$$\begin{aligned}&(x^*-y)^T S^{-1}(x^*-y)\\
&= \left(S\left(I - \frac{bb^TS}{b^TSb}\right)S^{-1}y - c\frac{Sb}{b^TSb} - y\right)^T S^{-1}\left(S\left(I - \frac{bb^TS}{b^TSb}\right)S^{-1}y - c\frac{Sb}{b^TSb} - y\right)\\
&= \left(\frac{Sbb^Ty}{b^TSb} + c\frac{Sb}{b^TSb}\right)^T S^{-1}\left(\frac{Sbb^Ty}{b^TSb} + c\frac{Sb}{b^TSb}\right)\\
&= \frac{(Sb)^T}{b^TSb}\left(b^Ty + c\right)^T S^{-1}\frac{Sb}{b^TSb}\left(b^Ty + c\right)\\
&= \frac{b^TS}{b^TSb}\left(b^Ty + c\right)^T \frac{b}{b^TSb}\left(b^Ty + c\right)\\
&= \left(b^Ty + c\right)^T \frac{b^TSb}{(b^TSb)^2}\left(b^Ty + c\right)\\
&= \frac{(b^Ty+c)^T(b^Ty+c)}{b^TSb}\\
&= \frac{(b^Ty+c)^2}{b^TSb}.\end{aligned} \tag{8.19}$$

We can conclude that the shortest squared Mahalanobis distance between a point y and the constraint of (8.14) is $\frac{(b^T y+c)^2}{b^T Sb}$. □

In theorem 8.7 we apply theorem 8.6 to convert the chance constraints into nonlinear deterministic constraints.

Theorem 8.7. Gaussian affine chance constraints Suppose the underlying distribution of a random variable x is Gaussian with sufficient statistics (μ, Σ) and we have that $d \in \mathbb{R}^n$ and $e \in \mathbb{R}$. Also, suppose that $d^T\mu + e > 0$ is ensured.

Then the chance constraint $\Pr(d^T x + e \geq 0) \geq p$ can be rewritten as the deterministic constraint $\frac{(d^T\mu+e)^2}{d^T\Sigma d} \geq k^2$ where k^2 is the (constant) critical value of the inverse cumulative Chi Squared distribution with the degrees of freedom equal to the dimensionality of x such that $\Pr(\mathcal{X} \leq k^2) = p$ with $\mathcal{X}$ a Chi Squared random variable. Note that $p > 0.5$ due to the fact that $d^T\mu + e > 0$ is ensured.

Furthermore, suppose $D \in \mathbb{R}^{n\times m}$, $e \in \mathbb{R}^n$ and $D\mu + e > 0$ which, again, implies that $p > 0.5$. Then joint chance constraint $\Pr(Dx + e \geq 0) \geq p$ can be written: for all $i = 1, \ldots, n$ we require $\frac{(d_i^T\mu+e_i)^2}{d_i^T\Sigma d_i} \geq k^2$. Note d_i is each row in D and e_i each corresponding element in e.

Proof. Intuitively theorem 8.7 posits that if the shortest squared Mahalanobis distance is further away than some threshold k^2 the chance constraint $\Pr(d^T x + e \geq 0) \geq p$ will be satisfied. Figure 8.2 depicts the idea that the minimum squared Mahalanobis distance must be greater than k^2 i.e. $\epsilon > 0$. The use of the Mahalanobis distance is important because it takes into account the uncertainty associated with each component of the random variable x.

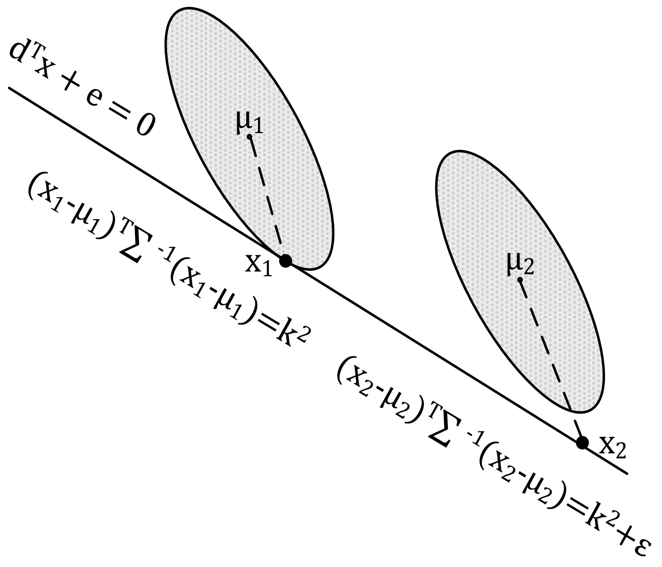
$d^Tx + e = 0$
μ_1
$(x_1-\mu_1)^T\Sigma^{-1}(x_1-\mu_1)=k^2$
x_1
μ_2
$(x_2-\mu_2)^T\Sigma^{-1}(x_2-\mu_2)=k^2+\varepsilon$
x_2

Figure 8.2: The ellipse centred around μ_1 intersects the constraint while the ellipse centred around μ_2 is wholly above (satisfies) the constraint. The shortest Mahalanobis distance will not necessarily be perpendicular to the constraint.

Since x is a Gaussian stochastic variable we have that $\mathbb{E}[x] = \mu$ and $\text{var}[x] = \Sigma$. Let

$\Omega = \{x \in \mathbb{R}^n \mid (x-\mu)^T\Sigma^{-1}(x-\mu) \leq k^2\}$ and k^2 be the critical value such that for the Chi Squared distribution with degrees of freedom equal to the dimension of x we have that $\Pr(\mathcal{X} \leq k^2) = p$. Then it is well known [53] that $\int_\Omega p(x|\mu,\Sigma)dx = p$ where $p(\cdot|\mu,\Sigma)$ is the multivariate Gaussian probability distribution function of x. If the shortest squared Mahalanobis distance from the mean of x is further away from the affine constraint $d^Tz+e=0$ than k^2 it implies that the curve of the function $h(z) = (z-\mu)^T\Sigma^{-1}(z-\mu) = k^2$ does not intersect with the constraint - a positive ϵ in the context of figure 8.2. Therefore Ω is wholly contained within the feasible region because $d^T\mu + e > 0$. This implies, with at least a probability of p, that the chance constraint will not be violated because the "confidence ellipse" is contained in the feasible region. Therefore, by applying theorem 8.6 to find the shortest squared Mahalanobis distance between μ and the constraint we have that $\frac{(d^T\mu+e)^2}{d^T\Sigma d} \geq k^2 \implies \Pr(d^Tx+e \geq 0) \geq p$.

The generalisation to more than one constraint, which should be jointly satisfied, is shown in figure 8.3. It is required that the ellipse, up to confidence level p, is wholly contained in the feasible region $Dx + e > 0$. If this requirement is satisfied we will satisfy the joint chance constraint by the same reasoning as before.

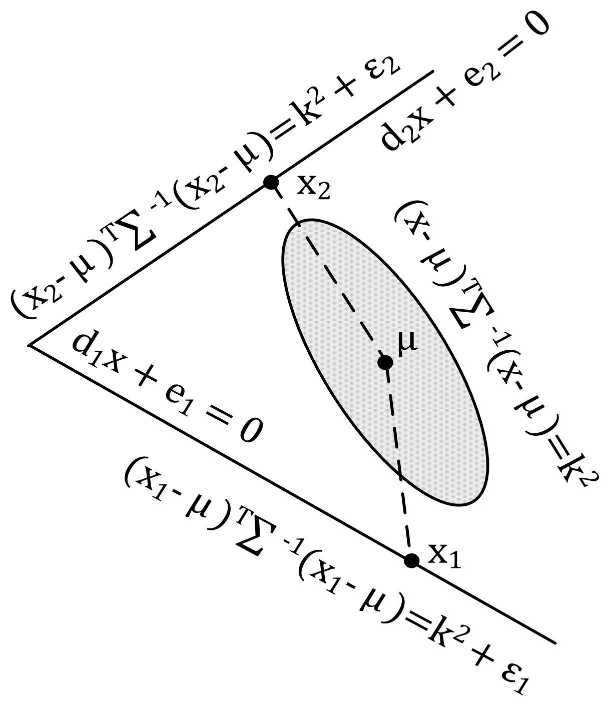
$(x_2-\mu)^T\Sigma^{-1}(x_2-\mu)=k^2+\varepsilon_2$
$d_2x+e_2=0$
x_2
$(x-\mu)^T\Sigma^{-1}(x-\mu)=k^2$
μ
$d_1x+e_1=0$
x_1
$(x_1-\mu)^T\Sigma^{-1}(x_1-\mu)=k^2+\varepsilon_1$

Figure 8.3: Illustration of the fact that theorem 8.7 holds jointly for multiple constraints. The ellipse centred around μ must lie wholly within the feasible region. This is guaranteed if the Mahalanobis constraint is satisfied.

By ensuring that $D\mu + e > 0$ and that the shortest squared Mahalanobis distance between μ and the constraints is bigger than k^2 this is guaranteed. This implies that if for all $i = 1, \ldots, n$ $\frac{(d_i^T\mu+e_i)^2}{d_i^T\Sigma d_i} \geq k^2$ then $\Pr(Dx+e \geq 0) \geq p$ will be satisfied. $\square$

In theorem 8.7 it is important that μ lie within the feasible region. If that is not required then it is possible for the ellipse to be wholly outside the feasible region while still satisfying the requirement that the shortest squared Mahalanobis distance between μ and the constraints is at least k^2. Clearly the chance constraint will then not be satisfied.

It is interesting to note the striking similarity between the work in [59] and [58] and theorem 8.7 given that we started analysing the problem within the framework of graphical models. The additional benefit of theorem 8.7 is that it formulates the chance constraint as a function of the statistical Mahalanobis distance measure. This is useful because it lends a statistical interpretation to theorem 8.7 even if the underlying distribution is non-Gaussian.

In theorem 8.8 we combine and elaborate on the work of [64], [59] to yield a QP MPC which exactly satisfies the stochastic MPC in (8.11) given the assumptions of linearity and normality. Note that the dimensionality of the problem is arbitrary.

Theorem 8.8. Conversion of the stochastic MPC formulation to the standard deterministic QP MPC formulation Under the assumptions of linearity and Gaussian distributions we can reformulate the stochastic MPC problem shown in (8.11) as a standard deterministic quadratic programming MPC problem

$$\begin{aligned}
&\min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q \mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N + \frac{1}{2}\sum_{k=0}^{N}\mathrm{tr}(Q\Sigma_k)\\
&\text{subject to } \mu_{t+1} = A\mu_t + Bu_t \\
&\text{and } \Sigma_{t+1} = W + A\Sigma_t A^T \\
&\text{and } d^T\mu_t + e \geq k\sqrt{d^T\Sigma_t d}\ \forall\ t = 1, \ldots, N.
\end{aligned} \tag{8.20}$$

Other deterministic constraints, e.g. on the input, can be added as usual. Note that we have assumed that the initial state estimate x_0 is available in the form of its mean, μ_0, and covariance, Σ_0. It is straightforward to include more chance constraints. Since each chance constraint is reduced to an inequality constraint which measures the Mahalanobis distance between the predicted distributions, the resultant feasible points will jointly satisfy all chance constraints.

Proof. Let the admissible set of controller inputs $\mathcal{U}$ be the same for both the stochastic MPC and the deterministic MPC formulations. Furthermore, let the current state estimate x_0 be given. Then by theorem 8.2 the objective function and equality constraints follow. By theorem 8.5 the expected value inequality constraint in (8.11) can be reformulated to require that $d^T\mu_t + e \geq 0$ for each $t = 1, 2, \ldots, N$. The chance constraint in (8.11) can be reformulated by using theorem 8.7:

$$\begin{aligned}
\frac{(d^T\mu_t + e)^2}{d^T\Sigma_t d} \geq k^2 &\implies (d^T\mu_t + e)^2 \geq k^2 d^T\Sigma_t d \\
&\implies d^T\mu_t + e \geq k\sqrt{d^T\Sigma_t d}\ \forall\ t = 1, 2, \ldots, N.
\end{aligned} \tag{8.21}$$

The first line of (8.21) follows because Σ_t is positive semi-definite for all $t = 1, 2, \ldots, N$ and $d \neq 0$ (otherwise it would not be a constraint), therefore it can be multiplied over the inequality sign like a positive number. By theorem 8.3 we have that $\Sigma_{t+1} = W + A\Sigma_t A^T$, therefore the predicted covariance matrices used in (8.21) are well defined. The second line follows because of the first inequality constraint: we know that $d^T\mu_t + e \geq 0 \; \forall \; t = 1, 2, \ldots, N$ and therefore we can square root both sides of the inequality constraint. Thus we have the two inequality constraints $d^T\mu_t + e \geq 0$ and $d^T\mu_t + e \geq k\sqrt{d^T\Sigma_t d}$ for each $t = 1, 2, \ldots, N$. Since $k > 0$ (otherwise we do not have a meaningful chance constraint) we can condense the two inequality constraints into a single constraint: $d^T\mu_t + e \geq k\sqrt{d^T\Sigma_t d} > 0$ for each $t = 1, 2, \ldots, N$ from which the conclusion follows. □

The beauty of theorem 8.8 is that no new theory is necessary to analyse the stability and convergence results of the new MPC. This is highly desirable because it allows one to merely "add" the inequality constraint in (8.20) to your existing MPC formulation. Most practical MPCs will have some form of state estimation and thus no new parameters are introduced either. Since the problem is in standard QP form it is straightforward to implement and, even more importantly, it is computationally fast because the problem is trivially convex.

In the work by [64] and [65], which primarily dealt with univariate problems, it was found that feasibility problems might arise if one uses the predicted covariance estimates in the controller. If the system is unstable the uncertainty in the estimates can grow with time as discussed in theorem 8.4. This can cause the ellipses used in theorem 8.7 to become too large to fit inside the feasible region. The approach adopted by [64] and [65] is to only use the one step ahead predicted covariance (Σ_1) over the entire prediction horizon. This does not ensure feasibility but restricts the growth associated with infeasibility. The drawback of this approach is that it might cause constraint violation because the controller will be more aggressive.

8.3 Reference tracking

So far we have only dealt with controllers which drive the system to the origin. The more general situation we are interested in is arbitrary reference point tracking. Fortunately, section 3.4.2 applies without modification because the stochastic MPC problem was reduced to the *standard* deterministic MPC problem.

8.4 Linear system

In this section we consider the problem of controlling a linear system using the stochastic MPC formulation of theorem 8.8. More precisely, we assume the linear model linearised about the unsteady operating point of our CSTR example from chapter 5 accurately represents the underlying system. For the purposes of illustration we will only use the (somewhat unrealistic) system where both states are measured. However, there is no theoretical reason why we cannot use the system where only temperature is measured. The drawback of using the latter system is that inference, as discussed previously, will be worse. The control goal is to steer the concentration, in the sense of definition 8.1, to the unsteady operating point ($C_A = 0.49$ kmol.m^{-3}) about which the system was linearised. See chapter 5 for more details.

We repeat the relevant system dynamics

$$\begin{aligned} &A = \begin{pmatrix} 0.9959 & -6.0308 \times 10^{-5} \\ 0.4186 & 1.0100 \end{pmatrix} \quad B = \begin{pmatrix} 0 \\ 8.4102 \times 10^{-5} \end{pmatrix} \quad C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ &W = \begin{pmatrix} 1 \times 10^{-6} & 0 \\ 0 & 0.1 \end{pmatrix} \quad V = \begin{pmatrix} 1 \times 10^{-3} & 0 \\ 0 & 10 \end{pmatrix} \end{aligned} \tag{8.22}$$

using a time step $h = 0.1$. The tuning parameters used in this and all subsequent chapters are

$$Q = P_f = \begin{pmatrix} 1 \times 10^4 & 0 \\ 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 1 \times 10^{-6} \end{pmatrix} \tag{8.23}$$

Note that the magnitude difference between the components of Q, P_f and R is necessary because the units of concentration and heat input are not scaled. We assume that a Kalman filter supplies the current posterior state estimate x_0 at each time step. Additionally, we assume a zero order hold of 1 min between controller inputs.

It is useful to introduce the measures we will use to quantify the effectiveness of the control techniques used in this and the following chapters. We define the average energy input by $\frac{1}{N}\sum_{t=0}^{N}|u_t - u_s|$ and the average concentration error by $\frac{1}{N}\sum_{t=0}^{N}|\frac{C_{At}-y_{sp}}{y_{sp}}|$. Additionally, to confirm that the results hold true over multiple simulations we will, at the end of sections 8.4 and 8.5, include a Monte-Carlo simulation. The metrics we use for the Monte Carlo simulations will be the time each simulation spent in violation of the constraints and the total state space area (measured within the context of the Mahalanobis distance) in violation of the constraint.

First we illustrate the approach of using only the result of theorem 8.3 i.e. we apply the LQG regulator to the system. Based on our previous results we know that given the Kalman filter's current posterior state estimate mean, μ_0, we only need to solve the LQR problem,

$$\begin{aligned} &\min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q \mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N \\ &\text{subject to } \mu_{t+1} = A\mu_t + Bu_t, \end{aligned} \tag{8.24}$$

to solve the LQG problem. The prediction horizon N is set at 150 time steps i.e. 15 minutes into the future. Figure 8.4 shows that the system does indeed converge, noisily, to the set point. This is not surprising because we have effectively just implemented the very well studied LQG controller. The primary drawback of this method is that there is no easy way to constrain the system. From a practical perspective this can be problematic.

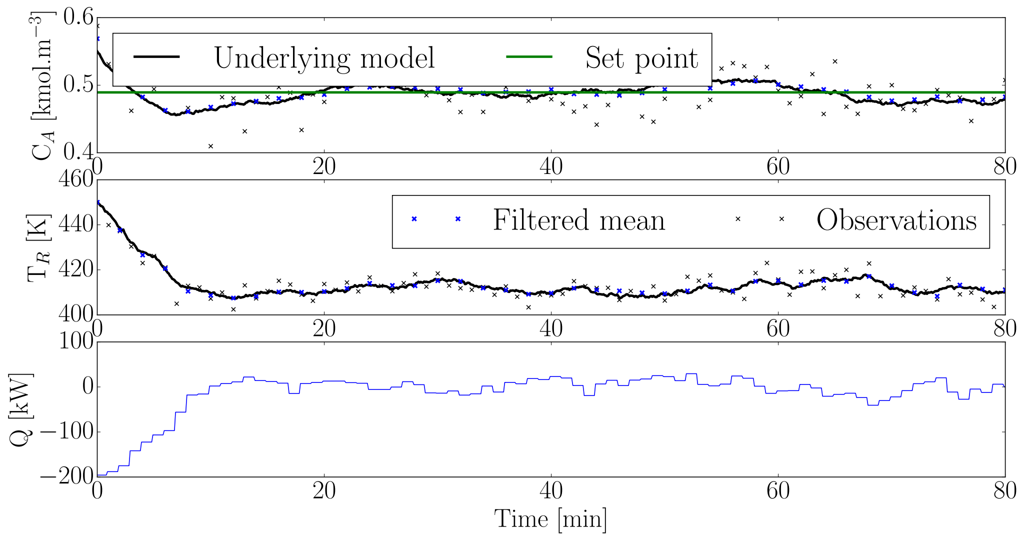

Figure 8.4: Unconstrained LQG regulator tracking with initial condition $(0.55, 450)$ and measuring both states.

The average heat energy usage (controller input) over the simulation run is 24.99 kW. The average set point error is 2.29% over the same 80 min time period.

Next we illustrate the approach of using conventional deterministic MPC to control the stochastic system. The MPC problem is

$$\begin{aligned}
&\min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q\mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N \\
&\text{subject to } \mu_{t+1} = A\mu_t + Bu_t \\
&\text{and } \begin{pmatrix}10\\1\end{pmatrix}^T \mu_t + 412 \geq 0\ \forall\ t = 1,\dots,N \\
&\text{and } |u_t| \leq 165\ \forall\ t = 0,\dots,N-1.
\end{aligned} \tag{8.25}$$

Using MPC allows us to easily add state and input constraints; this is a significant improvement over conventional LQG as discussed previously. We only use a single state constraint in this dissertation but the extension to multiple constraints is straightforward. The prediction and control horizon are equal to each other and set at 150 time steps i.e. 15 minutes into the future. We additionally constrain the magnitude of the inputs.

Due to assumption of normality and linearity and by theorem 8.3 and 8.5 we can also interpret the deterministic MPC as a stochastic MPC with an affine expected value constraint. Therefore the controller in (8.25) is not inappropriate for the control of the stochastic CSTR process under consideration.

In figure 8.5 we see the reference tracking and controller input for the deterministic MPC. The average heat energy input and set point error over the simulation run is 28.22 kW and 2.43%

respectively. Interestingly the average error is not much different but the controller requires more energy to track the set point. This is reasonable because the additional constraints make problem (8.25) a harder problem than (8.24).

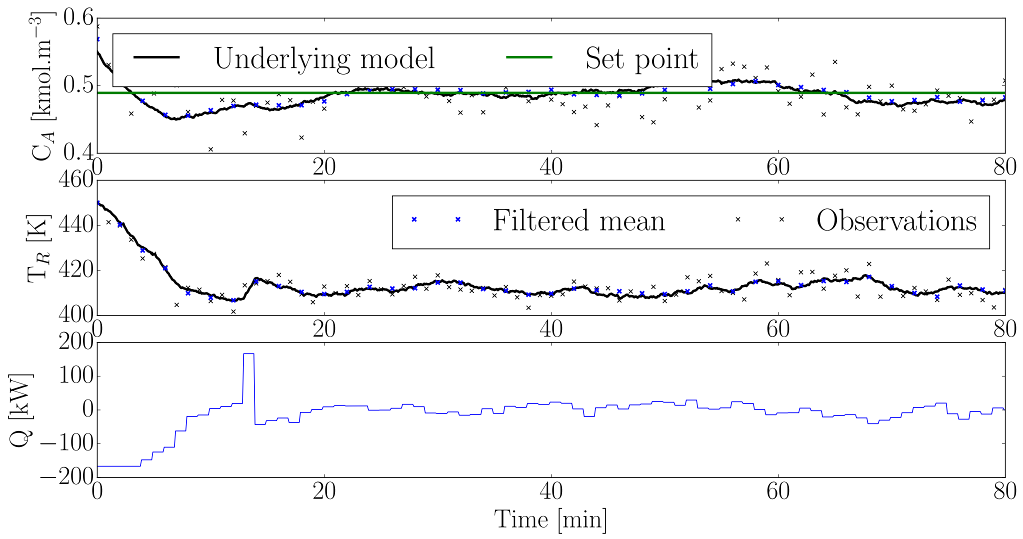

Figure 8.5: Deterministic constrained MPC tracking with initial condition $(0.55, 450)$ and measuring both states.

Like the LQG controller, it is clear that we have noisy convergence to the set point. As mentioned previously we will never be able to achieve zero set point offset because of the noise term in the system dynamics (8.1). Note that we have constrained the maximum magnitude of the inputs such that $|u_t| \leq 165$ kW. In the unconstrained case the controller required a maximum absolute input magnitude of over 200 kW; the ability to naturally constrain the inputs can be practically very useful. The benefit of MPC is apparent here.

In figure 8.6 we see the corresponding state space trajectory of the system together with the state constraint.

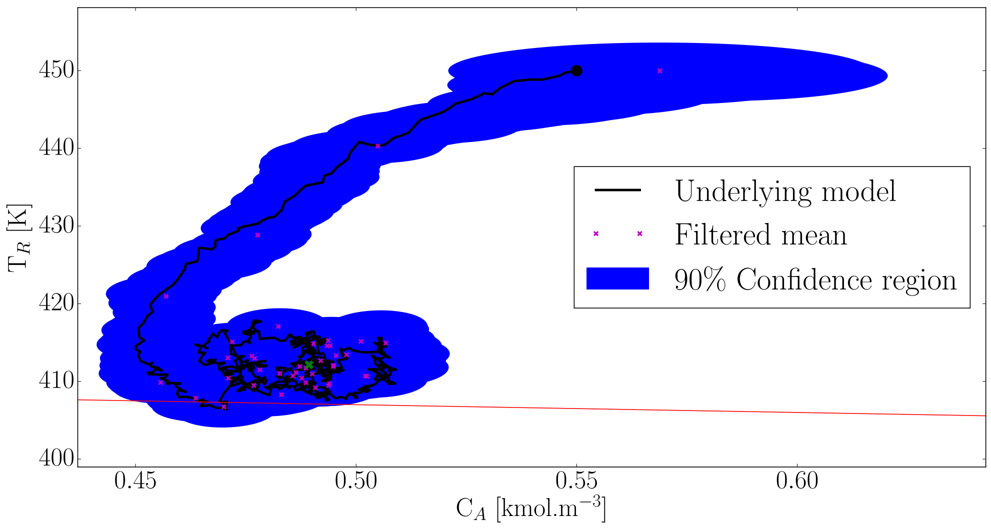

Figure 8.6: Deterministic MPC state space trajectory with initial condition $(0.55, 450)$ and measuring both states. The red line is the constraint.

While the predicted mean state estimates do not violate the constraint (due to the optimisation constraints) the actual underlying system does. This is clearly seen if one inspects the confidence region around the lower state estimates in figure 8.6. The confidence region is deeply violated by the constraint which implies that it is likely that the underlying system might. This is clearly a problem from a control point of view; the deterministic MPC cannot ensure that the constraint is satisfied.

We remedy this situation by introducing the chance constrained MPC as discussed in theorem 8.8

$$\begin{aligned}
&\min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q\mu_k + u_k^T Ru_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N + \frac{1}{2}\sum_{k=0}^{N}\mathrm{tr}(Q\Sigma_k)\\
&\text{subject to } \mu_{t+1} = A\mu_t + Bu_t\\
&\text{and } \Sigma_{t+1} = W + A\Sigma_t A^T\\
&\text{and } d^T\mu_t + e \geq k\sqrt{d^T\Sigma_t d}\ \forall\ t = 1,\dots,N\\
&\text{and } |u_t| \leq 165\ \forall\ t = 0,\dots,N-1.
\end{aligned} \tag{8.26}$$

Note that $d^T = (10, 1)$ and $e = 412$ as before. By consulting a Chi Squared distribution table we set $k^2 = 4.6052$ which corresponds to the chance constraint $\Pr(d^T x_t + e \geq 0) \geq 90\%\ \forall\ t = 1,\dots,N$. Note that problem (8.26) is harder than (8.25) due to the added constraint and thus we expect that the system will require greater controller input to satisfy the constraint.

In figure 8.7 we see that the stochastic MPC is able to track the set point in a similar manner as the LQG controller and deterministic MPC. The total average heat input and set point error over the simulation run is 38.58 kW and 2.58% respectively. This problem is harder than the preceding one due to the additional constraint and thus more energy is required.

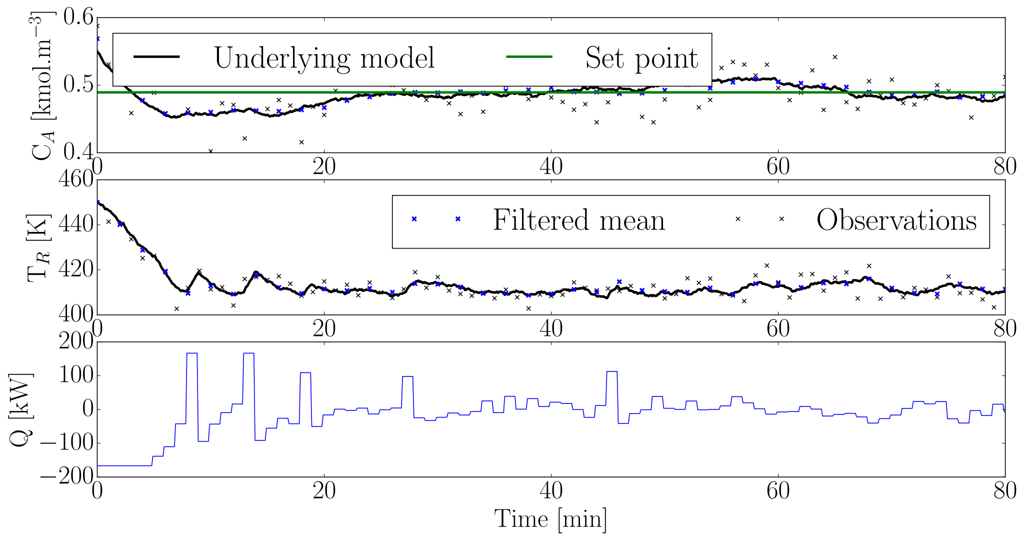

Figure 8.7: Chance constrained MPC tracking with initial condition $(0.55, 450)$ and measuring both states. A Kalman filter is used for inference and the chance constraint is set at 90%.

However, the benefit of adding the chance constraint is apparent in figure 8.8. It is clear that the constraint on the underlying state is not violated.

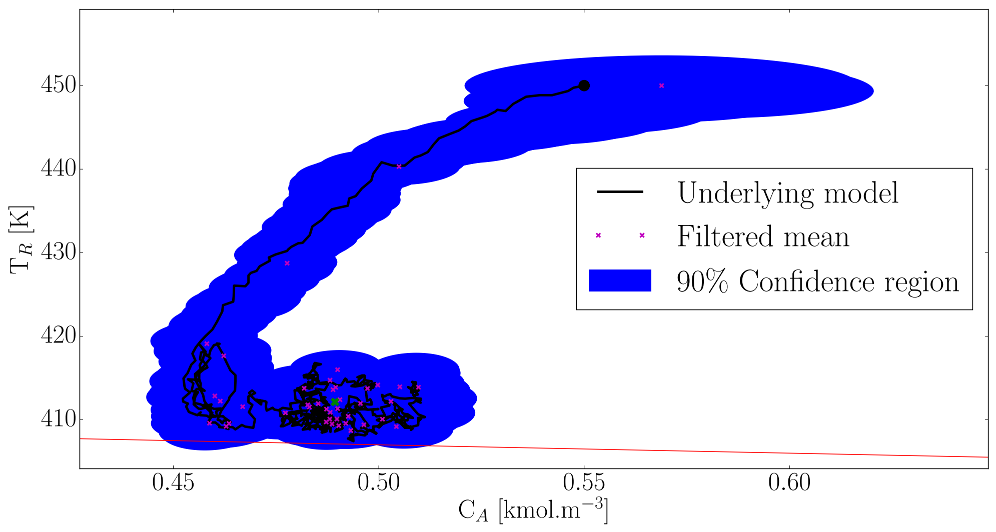

Figure 8.8: Chance constrained MPC state space trajectory with initial condition $(0.55, 450)$ and measuring both states. A Kalman filter is used for inference and the chance constraint is set at 90%.

Since the chance constraint is only enforced with probability 90% it is possible that the underlying system can come "close" to the constraint. This then has the consequence that the posterior confidence region marginally violates (spills over) the constraint as seen in the lower regions of figure 8.8.

It is interesting to investigate what effect increasing the probability that the chance constraint is satisfied will have on the system. To this end we modify the chance constraint of (8.26) such that $k^2 = 9.21$ which corresponds to the chance constraint $\Pr(d^T x_t + e \geq 0) \geq 99\% \; \forall \; t = 1, \ldots, N$. We expect that the underlying system will be moved further away from constraint due to this added level of conservativeness.

In figure 8.9 we see that the stochastic MPC still tracks the set point and figure 8.10 shows that the expected behaviour is realised.

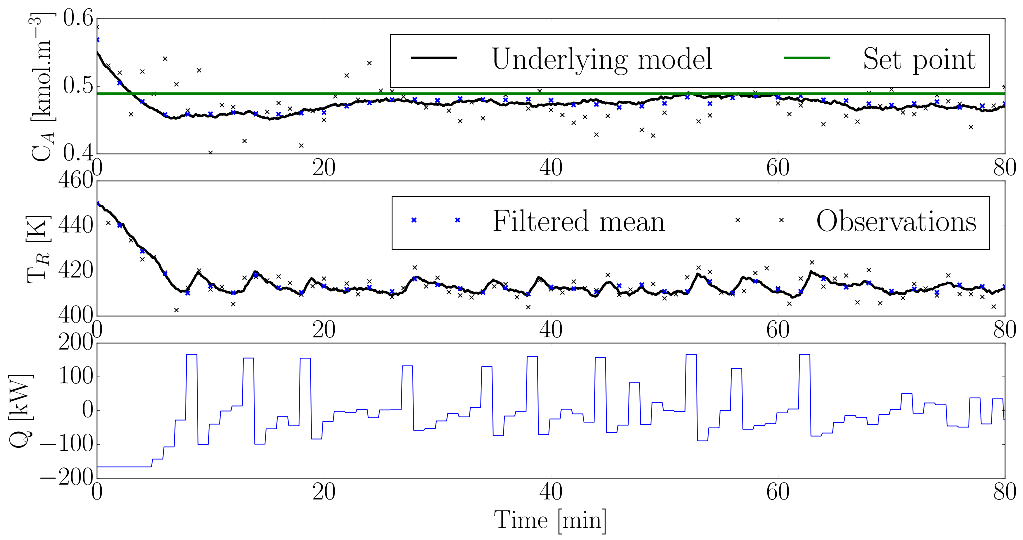

Figure 8.9: Chance constrained MPC tracking with initial condition $(0.55, 450)$ and measuring both states. A Kalman filter is used for inference and the chance constraint is set at 99%.

The average heat input and average set point error is 3.49% and 58.39 kW. The added conservativeness of the MPC prevents it from attempting to get to the set point as fast as the previous stochastic MPC; consequently there is no set point overshoot in figure 8.9. This causes the higher average error but the constraints are satisfied more robustly. As before, the control problem is harder and thus requires more energy.

In figure 8.10 we see the 90% confidence region is above the constraint. Since the probability that the predicted states are close to the constraint is much lower than before we see that the confidence region satisfies the constraint more robustly.

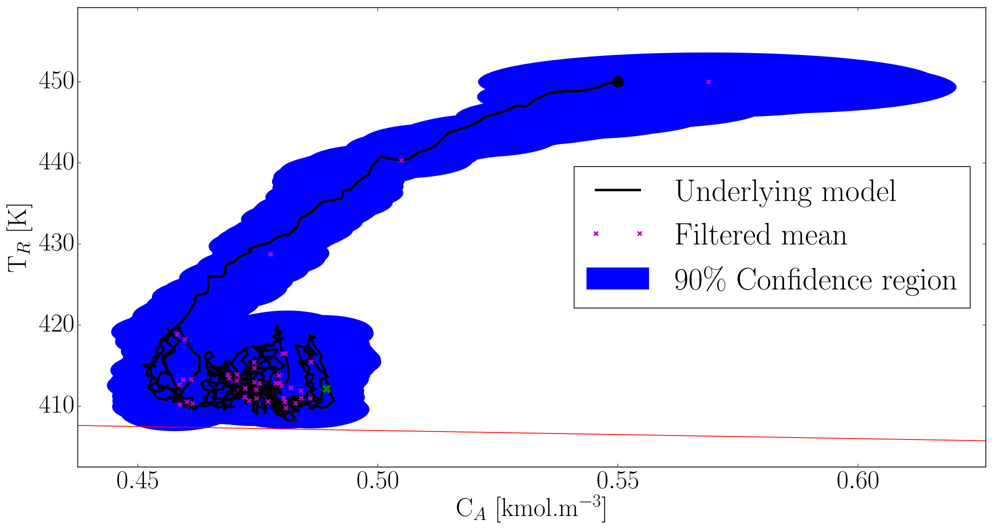

Figure 8.10: Chance constrained MPC state space trajectory with initial condition $(0.55, 450)$ and measuring both states. A Kalman filter is used for inference and the chance constraint is set at 99%.

It would also be correct to infer that k can be used as an empirical measure of the inherent stochastic conservativeness of the resulting controller. That is, lower values of k indicate a more aggressive controller which may violate the chance constraints and higher values of k indicate a more conservative controller. This can be useful for systems where the normal assumption is not valid but one would still like to enforce chance constraints in some empirical sense.

We have made the strong assumption that the system dynamics remain linear and Gaussian even under the MPC control law which is not necessarily linear [43]. Clearly if the system is far from Gaussian the Gaussian approach to simplifying the chance constraint will not be valid. Fortunately this is relatively simple to check and serves as a good way of measuring controller health. That is, the more Gaussian the filtered distributions are, the better the linear stochastic controller will work.

Kullback-Leibler Divergence was introduced in theorem 3.8 to estimate the degree to which samples match a given distribution. From chapter 7 we know that given enough particles a particle filter can accurately represent any distribution. Thus we temporarily replace the Kalman filter with a particle filter and use theorem 3.8 to estimate the degree of normality of the posterior state distributions.

Figure 8.11 shows the degree to which the underlying distribution is Gaussian. Since we cannot use an infinite number of particles we compare the sampled Gaussian distribution approximation to a baseline.

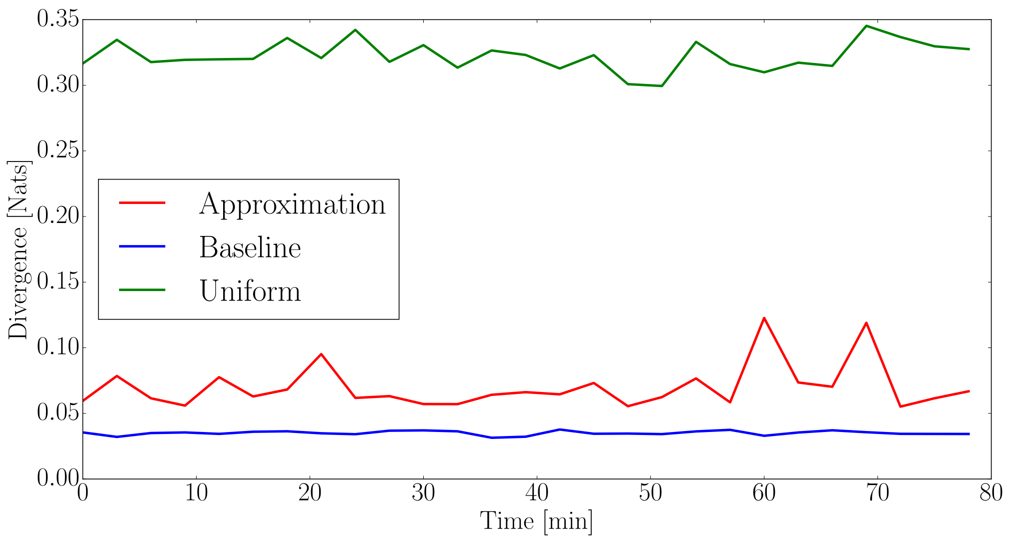

Figure 8.11: Kullback-Leibler Divergence between the assumed Gaussian distribution and different sampled distributions using 5000 particles. The underlying model is linear.

The approximation curve in figure 8.11 shows how much the samples diverge from the Gaussian distribution approximated using the samples. The baseline curve shows how much the Gaussian distribution diverges from samples of the same distribution. The uniform curve shows how much a Gaussian approximation of a Uniform distribution drawn in the interval $(\mu_i - 2\sigma_{ii}, \mu_i + 2\sigma_{ii})$ (for each i in the dimension of the underlying distribution) diverges; this serves to illustrate the divergence one would expect if attempting to model a distribution which is decidedly not normal. One would expect the baseline curve to tend to zero as the number of particles tends to infinity. Sampling error causes divergence from zero for the baseline curve. Thus we can use the baseline and uniform curves as a crude measure of normality.

In figure 8.11 we see that the approximation is relatively close to the baseline. Additionally it is far removed from the uniform curve. The average divergence for the baseline, approximation and uniform curve (in nats) is: 0.035, 0.069 and 0.322 respectively. This implies that even though we are using a nonlinear control technique the posterior state distributions are still approximately Gaussian.

In figure 8.12 we investigate the effect k^2 has on the chance constraint. We expect that by increasing k^2 the system becomes more conservative and less likely to violate the constraint. Under the assumptions of linearity and normality we are assured of this behaviour due to theorem 8.8. However, as figure 8.11 indicates our assumptions do not strictly hold. In figures 8.8 and 8.10 we reached the conclusion that the chance constraint was effective by only considering a single simulation. In figure 8.12 we show the compilation of over 2000 runs to illustrate the veracity of the claim that the chance constraints are effective.

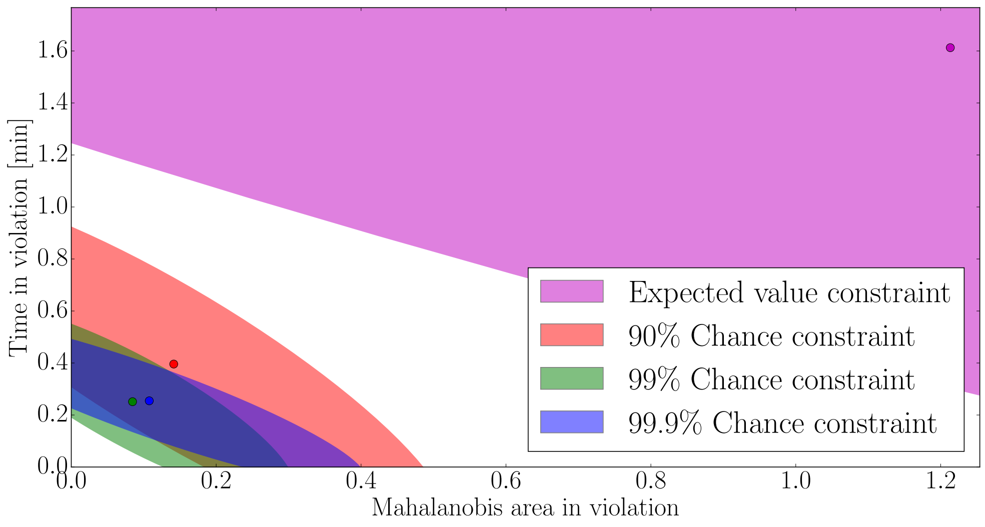

Figure 8.12: Monte-Carlo simulation to investigate the effect increasing k^2 has on the constraint violation characteristics of the MPC controllers. Each region indicates where 90% of the simulations scored. The mean is indicated by a solid point.

The Mahalanobis area dimension indicates the degree to which the constraint was violated: larger numbers imply the constraint was deeply violated in state space. The time in violation dimension indicates the length of time the system violated the constraint. It is clear that the chance constraints dramatically reduce the violation characteristics of the system.

Interestingly, there seems to be diminishing returns on increasing k^2 beyond the 99% chance interval. Feasibility issues plague the system under the more conservative chance constraint levels. Intuitively the predicted ellipses become too big to fit inside the feasible region - especially when projected forward in time. Given that the controller input - the only way to move them inside the feasible region - is constrained the system reaches a performance deadlock. In the work by [64] they also noted this problem and solved it by not letting the confidence ellipses grow as they were projected into the future (see the discussion after theorem 8.4 and in section 2.2). For this system the issue is not severe enough to warrant further concern.

Finally, we illustrate that the MPC controllers we developed, using the linear underlying model, actually drive the system to the set point for many runs - not just the single realisation we considered so far. In figure 8.13 a Monte Carlo simulation was conducted using 50 runs per controller.

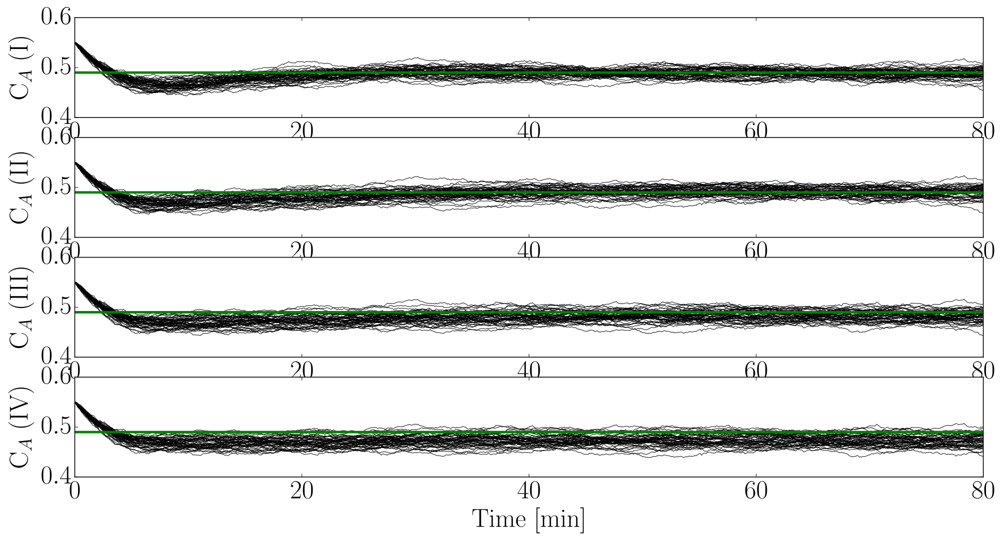

Figure 8.13: Monte-Carlo simulation showing the set point tracking of the MPC controllers developed in this chapter. Subplot (I) corresponds to the expected value constrained MPC, subplots (II), (III) and (IV) correspond to the 90%, 99% and 99.9% chance constrained MPC controllers. The green line is the set point.

It is clear that each controller causes the system to approach set point. However, the more conservative the systems become (i.e. 99% and 99.9% chance constrained) the longer they take to reach the set point. We did not include the LQG controller in this analysis because the theory supporting its convergence is very well studied in literature. The Monte Carlo absolute average set point errors over the last 10 minutes for each controller are: 1.54%, 1.73%, 2.15% and 3.5% respectively. The controllers come close enough to the set point to suggest they work (a small δ as per definition 8.1). The increasing average error is caused by the conservativeness of the system - at the end of the simulation the controllers have not yet come as close to their set point as the more aggressive controllers.

8.5 Nonlinear system

In this section we consider the problem of controlling the full nonlinear system with a linear model linearised around the unsteady operating point. The control goal is the same as before; the only difference between this section and section 8.4 is that the underlying plant is nonlinear.

The linear control model and noise parameters are the same as (8.22). The control tuning parameters are the same as (8.23).

As before we first investigate the LQG controller. Figure 8.14 shows the unconstrained reference tracking results.

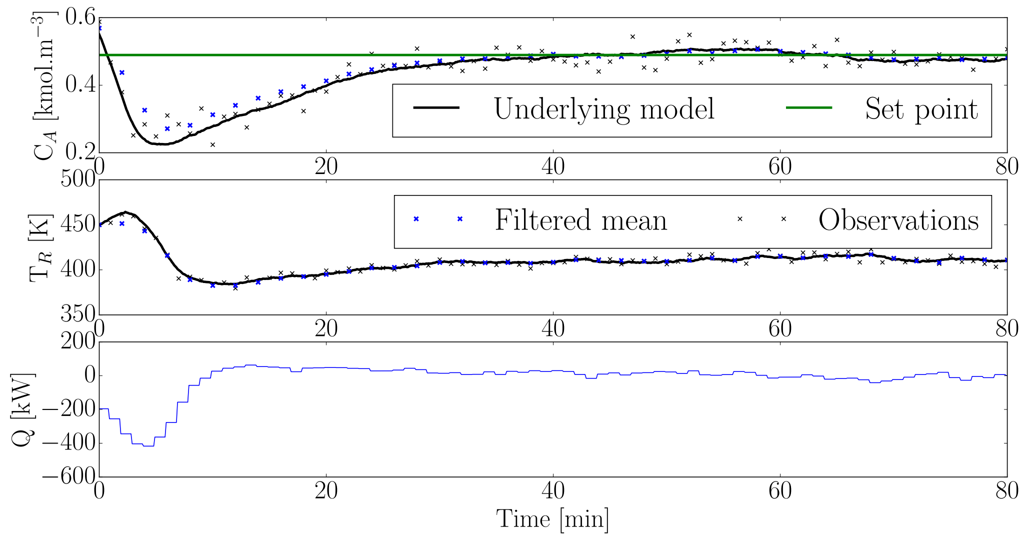

Figure 8.14: LQG regulator tracking with initial condition (0.55, 450) and measuring both states.

The average energy usage and concentration error was 50.88 kW and 11.66% respectively over the 80 min simulation time. Comparing figures 8.4 and 8.14 we see that the maximum absolute input energy is much greater with the nonlinear underlying dynamics. This is not unexpected because the controller in both cases is linear: one expects the plant-model mismatch to have a detrimental effect on control.

In section 8.4 we had a linear underlying model and an almost linear controller. Figure 8.11 also demonstrated that the posterior state distributions were approximately Gaussian. Thus there was no reason to use nonlinear inference algorithms like the particle filter introduced in chapter 7. However, in this chapter we are using a nonlinear underlying model and it might be advantageous to use a more sophisticated inference tool. We investigate using both a Kalman filter and a particle filter for inference. In the setting of the particle filter we approximate the samples as Gaussian and use that for control.

As before we first investigate the deterministic MPC. The control problem is

$$\begin{aligned} &\min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q \mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N \\ &\text{subject to } \mu_{t+1} = A\mu_t + Bu_t \\ &\text{and } \begin{pmatrix}10\\1\end{pmatrix}^T \mu_t + 406 \geq 0 \ \forall\ t = 1,\dots,N \\ &\text{and } |u_t| \leq 330 \ \forall\ t = 0,\dots,N-1. \end{aligned} \tag{8.27}$$

Note that the constraints are different due to the expected extra difficulty introduced by the nonlinear underlying model. In figure 8.15 we see that the deterministic MPC using the Kalman filter for state inference does converge to the set point. The ability to naturally constrain the system is again highlighted in the input: as opposed to the peak |420| kW required by the LQG controller the MPC manages to control the system while never requiring more than |330| kW.

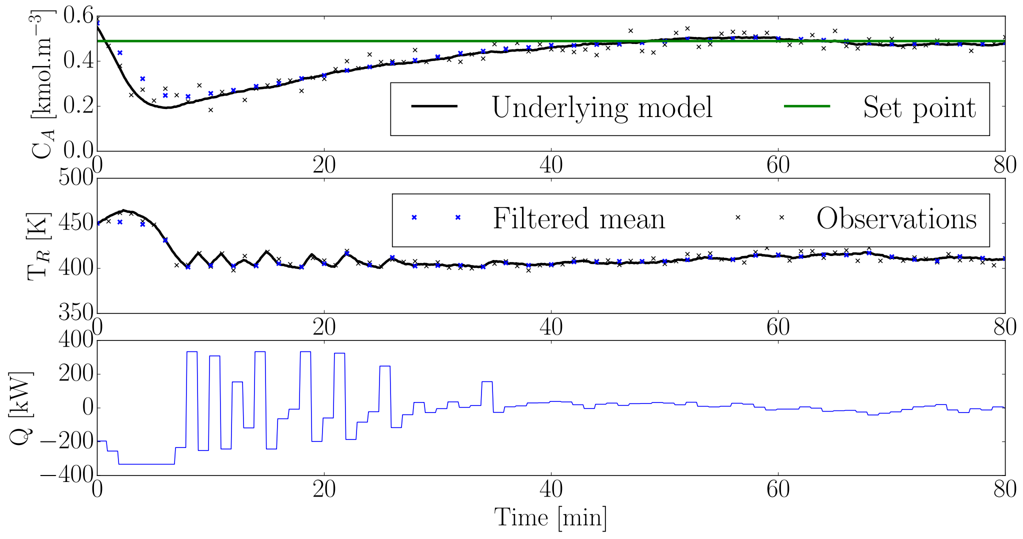

Figure 8.15: Deterministic constrained MPC reference tracking with initial condition $(0.55, 450)$ and measuring both states. The Kalman filter is used for inference.

In figure 8.16 we see that the state constraint is violated just like figure 8.6. We also see a somewhat unrealistic jagged state trajectory but this is just a numerical artefact. The average energy input and concentration error over the simulation run is 88.68 kW and 16.11% respectively. The added constraints explain why the performance is degraded when compared to the LQG controller.

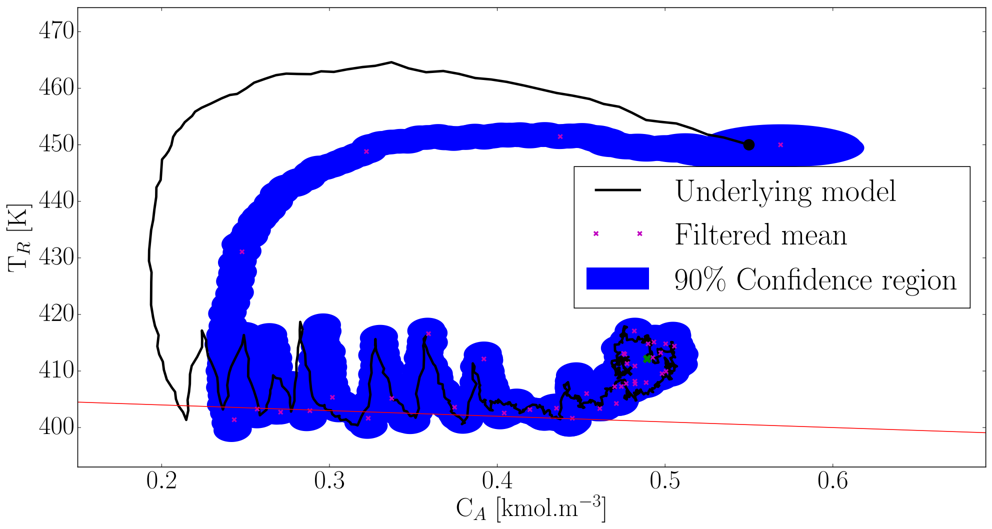

Figure 8.16: Deterministic constrained MPC state space trajectory with initial condition $(0.55, 450)$ and measuring both states. The Kalman filter is used for inference.

Since we are not using a chance constrained MPC the state constraint violation is not surprising in figure 8.16. However, a more significant issue is the inability of the Kalman filter to accurately track the states throughout the simulation (the underlying system briefly diverges from the state estimates). This can be significantly problematic if a constraint existed in the left hand side of the state space: the controller wouldn't know that it was violating the constraint because the state estimate is poor. This behaviour is caused by the linear model used by the Kalman filter. The state trajectory moves away from the region close to the linearisation point and thus, as explained in chapter 6, the state estimate becomes poor.

We can remedy this situation by using a more sophisticated inference algorithm. In figure 8.17 we see the deterministic MPC using a particle filter with 200 particles for inference.

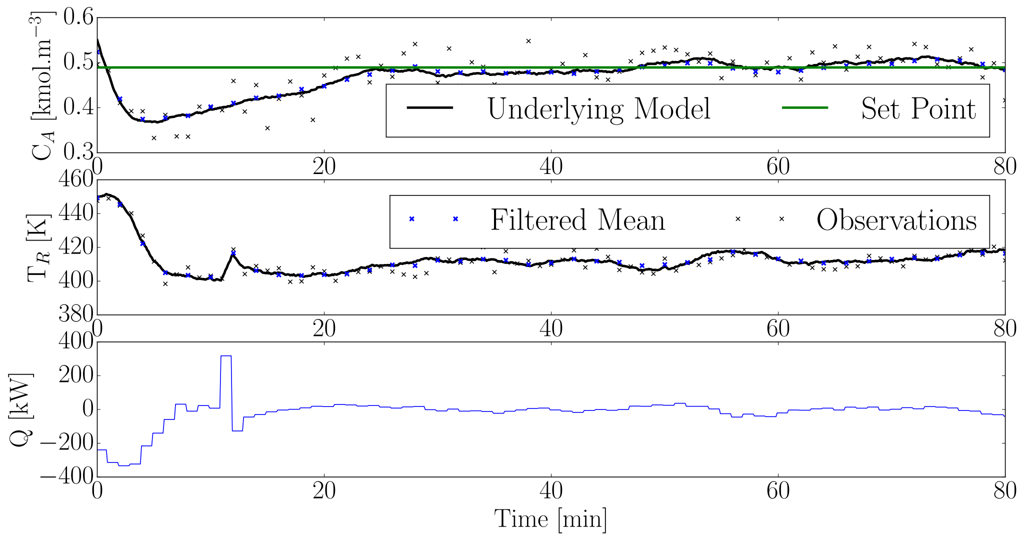

Figure 8.17: Deterministic constrained MPC reference tracking with initial condition $(0.55, 450)$ and measuring both states. A particle filter with 200 particles is used for inference.

The average energy input and concentration error is 38.83 kW and 5.81% respectively. This is a vast improvement over the same controller where the Kalman filter was used for inference. The benefit of accurate state estimation is apparent here and also in figure 8.18.

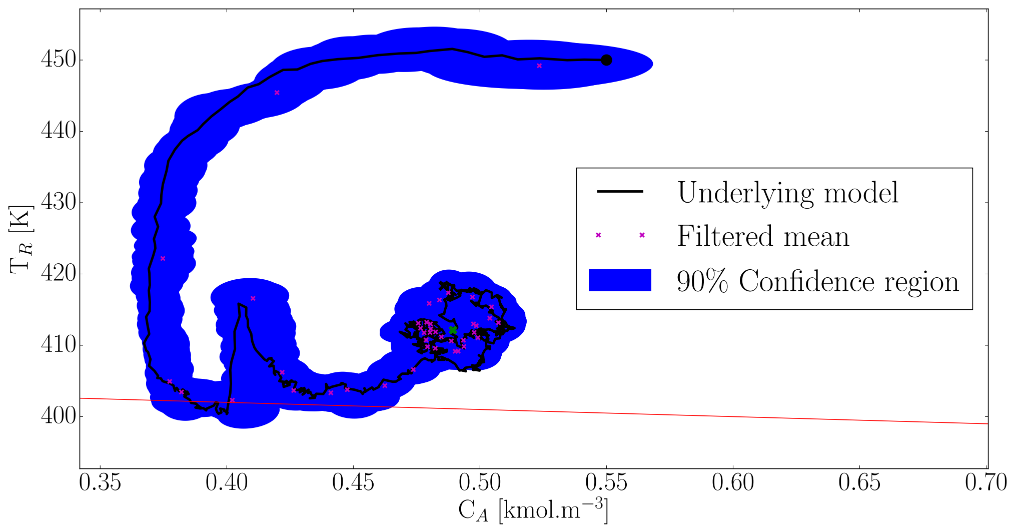

Figure 8.18: Deterministic constrained MPC state space trajectory with initial condition $(0.55, 450)$ and measuring both states. A particle filter with 200 particles is used for inference.

In figure 8.16 we saw significant estimation deviation from the true underlying state, while in figure 8.18 the deviation is negligible. We still have that the state constraint is violated but this is due to the stochastic nature of the underlying system. It is clear that the particle filter

MPC combination is superior to the Kalman filter MPC combination in this case. However, the benefit of using the particle filter should be weighed against the cost of the algorithm especially in higher dimensions where it is known that the particle filter does not perform well (recall the discussion following figure 7.7).

Now we introduce the chance constrained MPC

$$\begin{aligned}
&\min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q\mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N + \frac{1}{2}\sum_{k=0}^{N}\mathrm{tr}(Q\Sigma_k)\\
&\text{subject to } \mu_{t+1} = A\mu_t + Bu_t\\
&\text{and } \Sigma_{t+1} = W + A\Sigma_t A^T\\
&\text{and } d^T\mu_t + e \geq k\sqrt{d^T\Sigma_t d}\ \forall\ t = 1,\ldots,N\\
&\text{and } |u_t| \leq 330\ \forall\ t = 0,\ldots,N-1.
\end{aligned} \tag{8.28}$$

Note that the constraints are different than (8.26) for the same reason as (8.27). We have $d^T = (10, 1)$ and $e = 406$ as before. By consulting the Chi Squared distribution table we set $k^2 = 4.6052$ which corresponds to the chance constraint $\Pr(d^T x_t + e \geq 0) \geq 90\%\ \forall\ t = 1,\ldots,N$ exactly as in the previous chapter.

As with the deterministic case we first investigate the system where a Kalman filter is used for inference. Figure 8.19 illustrates that the chance constrained system does indeed converge to the set point. The average energy input and concentration error is 80.75 kW and 14.12%. The average energy usage and average concentration error is reduced compared to the deterministic system.

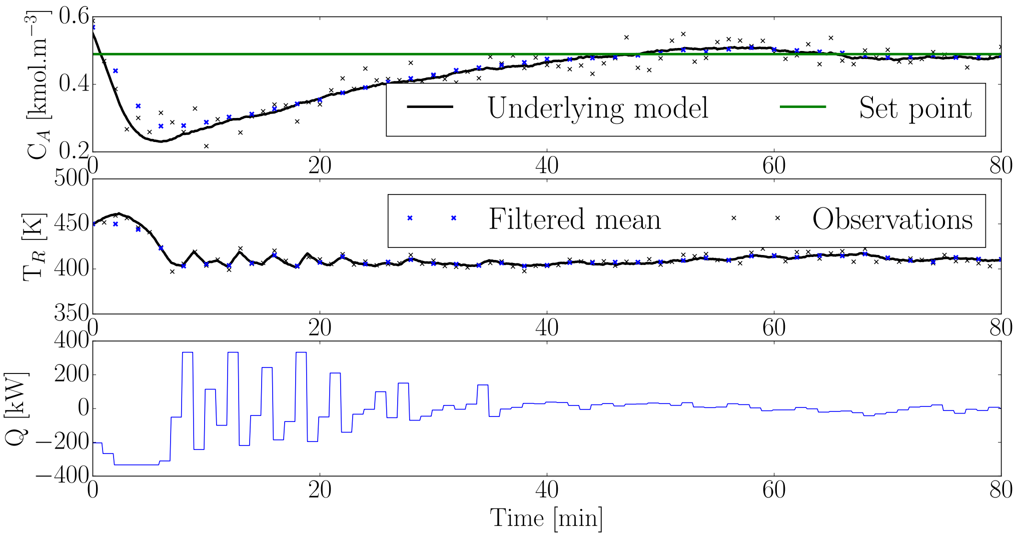

Figure 8.19: Chance constrained MPC tracking with initial condition (0.55, 450) and measuring both states. A Kalman filter is used for inference and the chance constraint is set at 90%.

The benefit of adding the chance constraint is apparent in figure 8.20. It is clear that the constraint on the underlying state is not violated as much as in figure 8.16.

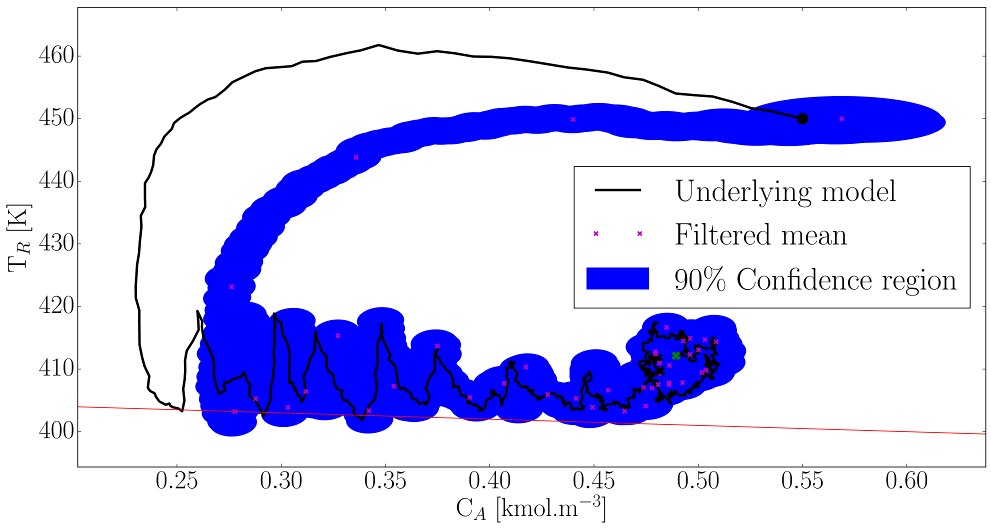

Figure 8.20: Chance constrained MPC state space trajectory with initial condition $(0.55, 450)$ and measuring both states. A Kalman filter is used for inference and the chance constraint is set at 90%.

It is clear that the nonlinearity of the underlying system makes stochastic control difficult. By increasing $k^2 = 9.21$ which corresponds to changing the chance constraint such that $\Pr(d^T x_t + e \geq 0) \geq 99\% \ \forall \ t = 1, \ldots, N$ we hope to increase the minimum distance between the constraint and the underlying state.

Figure 8.21 shows the tracking of the modified system. The average energy input and average concentration error is 69.67 kW and 12.44% respectively. It is quite interesting that the more conservative system performs better with regard to these to metrics than the less conservative system.

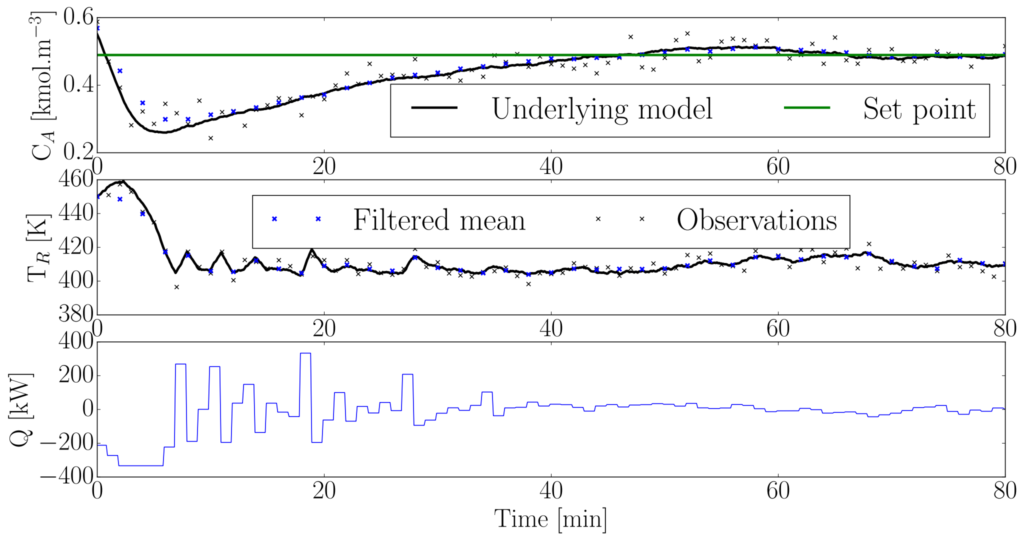

Figure 8.21: Chance constrained MPC tracking with initial condition $(0.55, 450)$ and measuring both states. A Kalman filter is used for inference and the chance constraint is set at 99%.

In figure 8.22 we see that the margin of safety is increased although the confidence region still spills over the constraint. However, there is no constraint violation in this case.

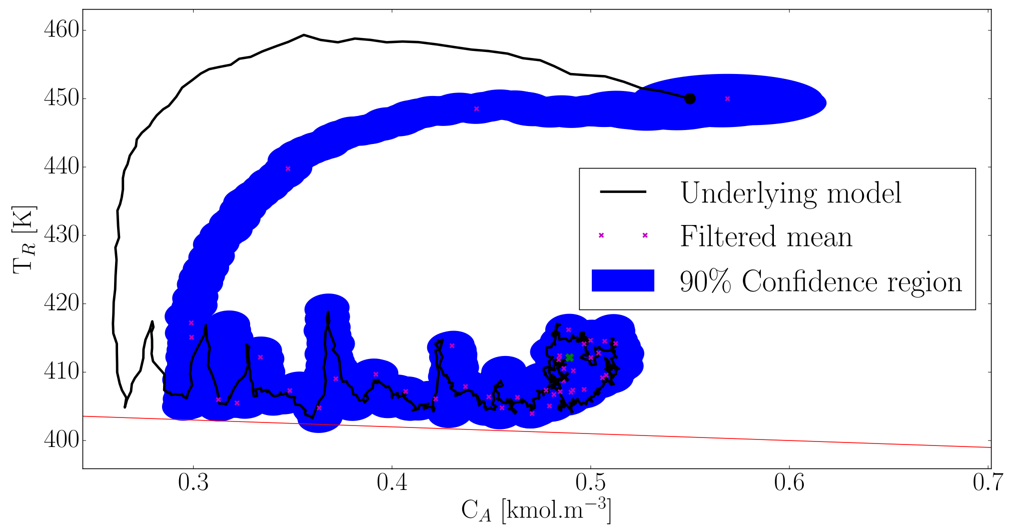

Figure 8.22: Chance constrained MPC state space trajectory with initial condition $(0.55, 450)$ and measuring both states. A Kalman filter is used for inference and the chance constraint is set at 99%.

The ability of k to increase or decrease the conservativeness of the constraint lends credibility to its value, if the system is non-normal, at the very least as an empirical measure to include stochastic robustness to the MPC in an efficient way.

Figure 8.12 illustrates the effect increasing k^2 (i.e. the threshold for the chance constraint) has on the system. Over 2000 simulations were used to illustrate that these tendencies hold not just for a specific realisation.

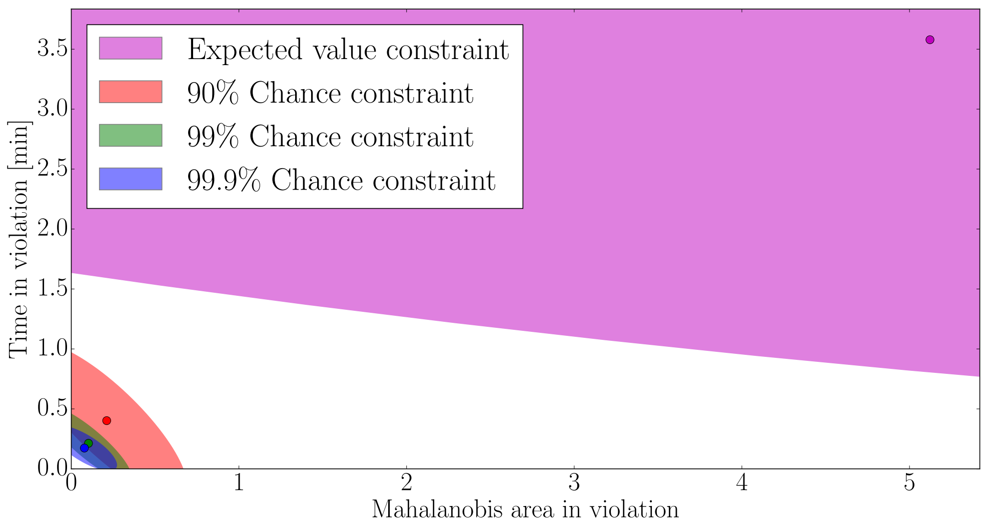

Figure 8.23: Monte-Carlo simulation to investigate the effect increasing k^2 has on the constraint violation characteristics of the MPC controllers. Each region indicates where 90% of the simulations scored. The mean is indicated by a solid point. A Kalman filter was used for inference.

It is clear that increasing k^2 causes the constraints to be violated less (as per the theory). It is also clear that the expected value constrained MPC (i.e. the deterministic MPC in figure 8.12) performed significantly worse with the nonlinear underlying model when compared to figure 8.23. The value of the chance constrained MPC is apparent here. Unlike figure 8.12 there seems to be more value in increasing the chance threshold above the 99% interval.

Figure 8.13 illustrates that the controllers we developed in this chapter, using the Kalman filter on the nonlinear underlying system, do indeed drive the system to the set point for many runs (not just the single realisation of our earlier examples). We again do not include a Monte Carlo simulation for the LQG controller because it is very well studied in literature.

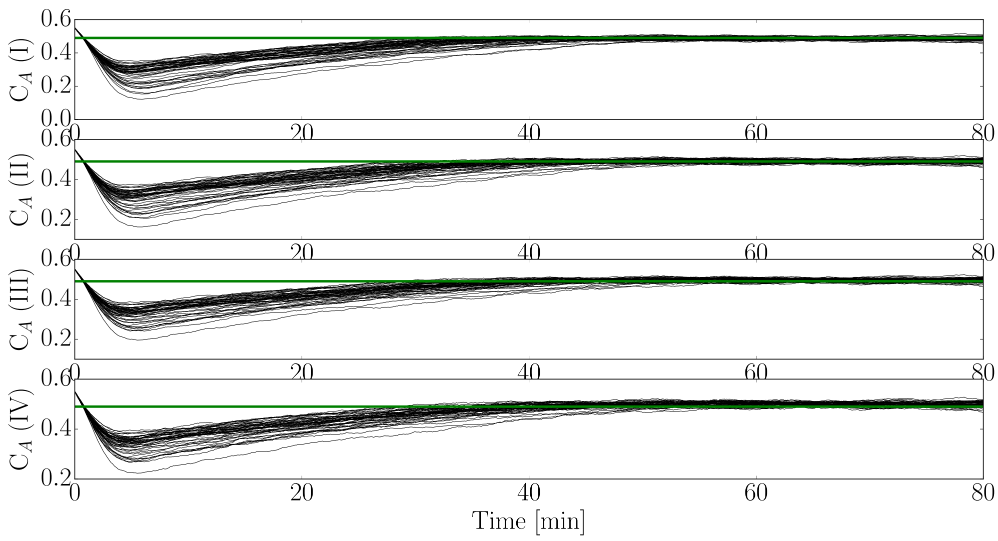

Figure 8.24: Monte-Carlo simulation showing the set point tracking of the MPC controllers developed in this chapter. Subplot (I) corresponds to the expected value constrained MPC, subplots (II), (III) and (IV) correspond to the 90%, 99% and 99.9% chance constrained MPC controllers. The green line is the set point. A Kalman filter was used for inference.

The Monte Carlo simulations (50 per controller) indicated that all the controllers tracked the set point. The Monte Carlo absolute average set point error over the last 10 minutes for each controller are: 1.56%, 1.47%, 1.70% and 2.24% respectively. The controllers come close enough to the set point to suggest they work (a small δ as per definition 8.1). We again see that the more conservative controllers have larger errors because they take longer to reach set point. However, this tendency is much less pronounced here than in figure 8.13.

Figures 8.19 to 8.22 all display the undesirable property originally seen in figure 8.15: the poor state estimation and associated control problems. We attempt to rectify this situation by using a more sophisticated filter with the chance constrained MPC. We use the MPC as shown in (8.28) with a particle filter using 200 particles. The tracking results are shown in figure 8.17.

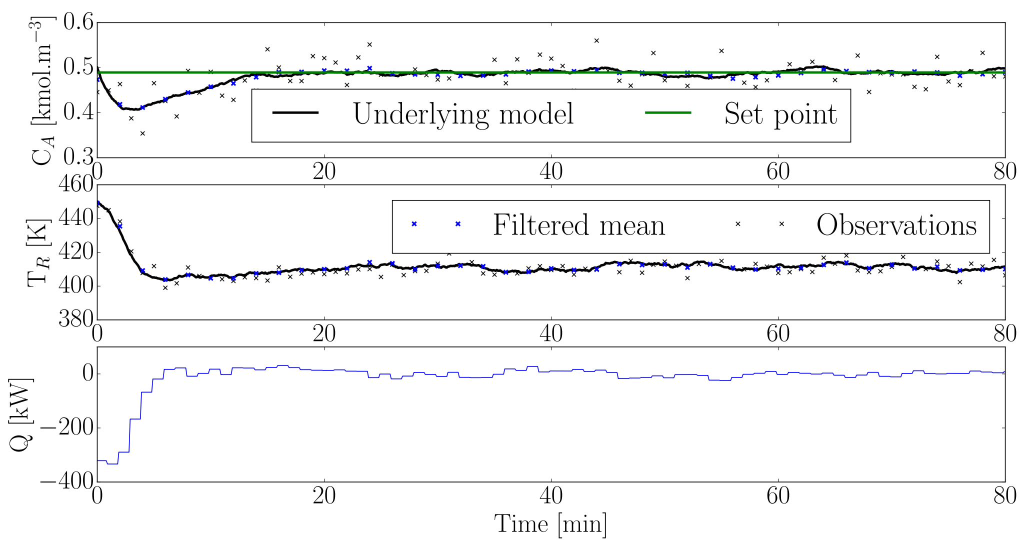

Figure 8.25: Chance constrained MPC tracking with initial condition $(0.55, 450)$ and measuring both states. A particle filter with 200 particles is used for inference and the chance constraint is set at 90%.

One could see the discussion in section 7.5 as a justification for using the particle filter instead of the Kalman filter for state estimation. Since the underlying model is nonlinear we expect the particle filter to outperform the Kalman filter (recall the discussion following figure 7.7).

The average input energy and average concentration error is 24.26 kW and 2.34% over the course of the simulation. This is a vast improvement over the Kalman filter MPC combination. Clearly the more accurate state estimation is immensely beneficial for control. Figure 8.26 also illustrates that the chance constraint is easily satisfied using the more accurate state estimator.

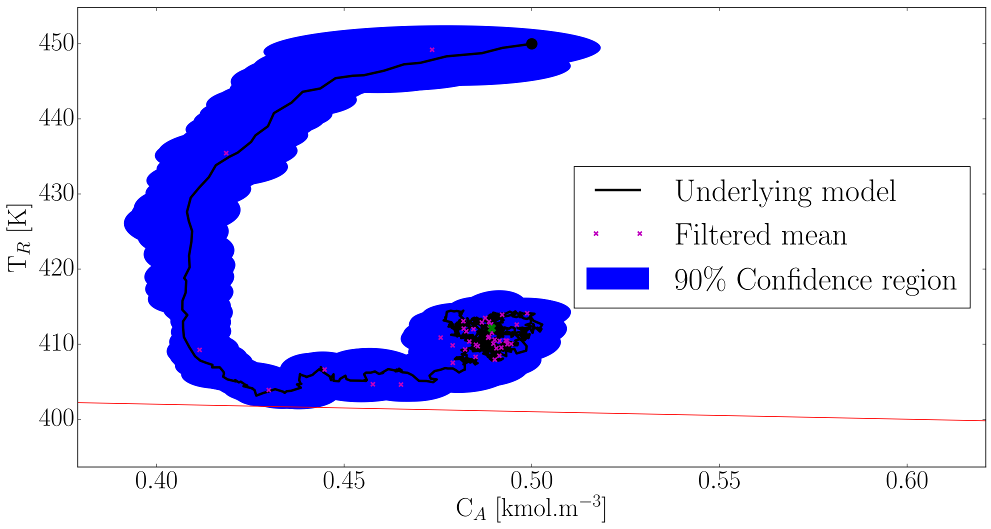

Figure 8.26: Chance constrained MPC state space trajectory with initial condition $(0.55, 450)$ and measuring both states. A particle filter with 200 particles is used for inference and the chance constraint is set at 90%.

In figure 8.27 we compare the constraint violation characteristics of the Kalman filter and particle filter based MPCs using over 2000 simulations.

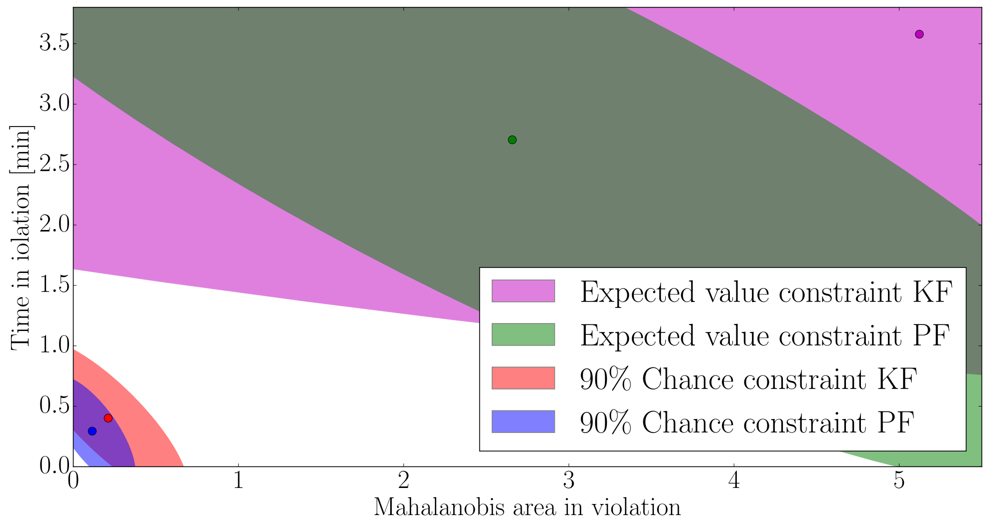

Figure 8.27: Monte-Carlo simulation to investigate the effect increasing k^2 has on the constraint violation characteristics of the MPC controllers. Each region indicates where 90% of the simulations scored. The mean is indicated by a solid point. KF indicates the Kalman Filter MPC and PF indicates the particle filter MPC.

It is clear that the particle filter based MPC outperforms the Kalman filter based MPC in terms of constraint violation characteristics. This is not surprising because, as we discussed for the single realisation results, the state estimation ability of the Kalman filter is inferior due to the nonlinear underlying system.

In figure 8.28 we illustrate that the particle filter based stochastic MPC controllers we developed track the set point.

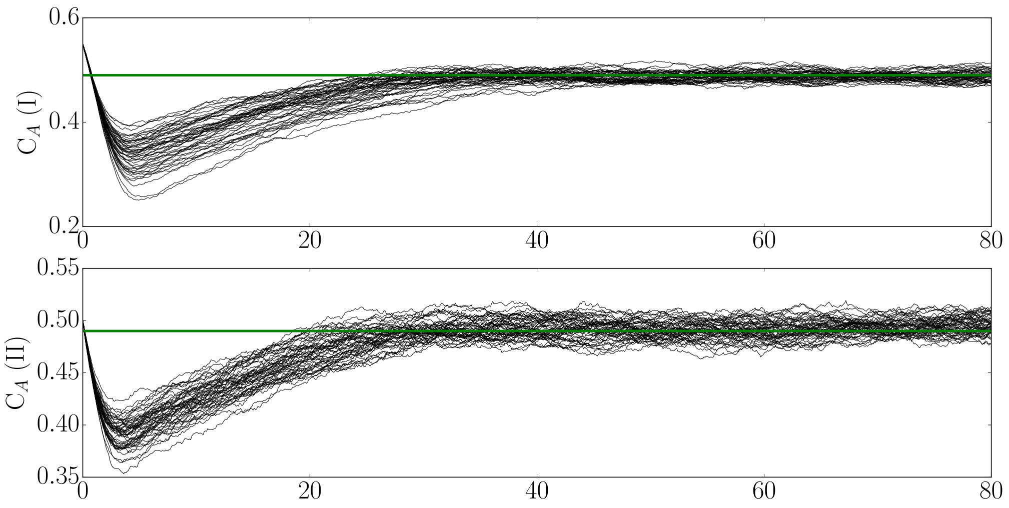

Figure 8.28: Monte-Carlo simulation showing the set point tracking of the MPC controllers developed in this chapter. Subplot (I) corresponds to the expected value constrained MPC and subplot (II) corresponds to the 90% chance constrained MPC controller. The green line is the set point. A particle filter was used for inference.

The Monte Carlo average set point error over the last 10 minutes of each controller (50 simulations per controller) are: 1.51% and 1.59% respectively. The controllers come close enough to the set point to suggest they work (a small δ as per definition 8.1).

As before we need to investigate the normal assumption underpinning the theory behind the chance constraint simplification. We investigate it in exactly the same manner as figure 8.11 using Kullback-Leibler Divergence. To this end, figure 8.29 shows the degree to which the underlying posterior state distributions are Gaussian given the nonlinear underlying system.

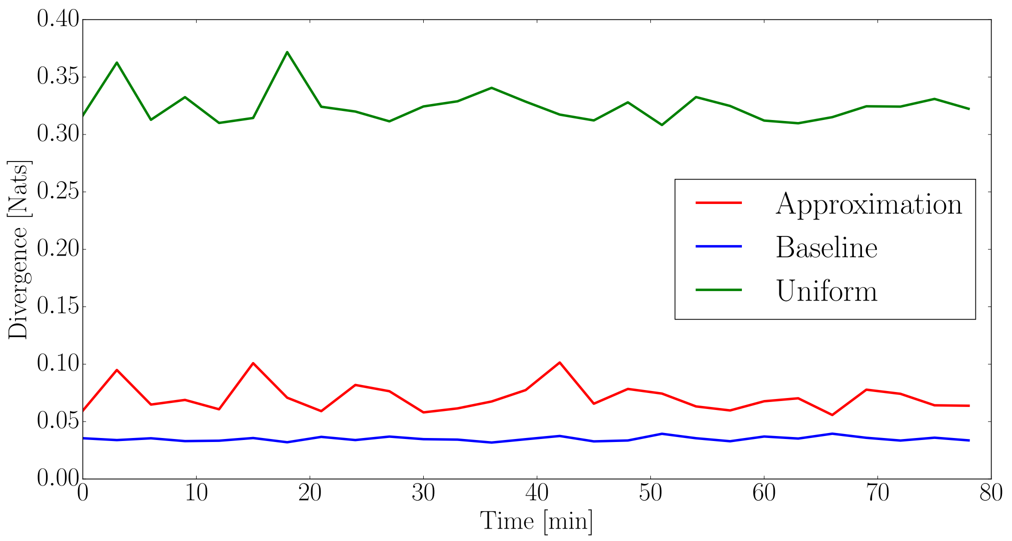

Figure 8.29: Kullback-Leibler Divergence between the assumed Gaussian distribution and actual distribution using 5000 particles. The underlying model is nonlinear.

It is a relief that the normal assumption seems to hold almost as well in the nonlinear underlying system case. The average divergence for the baseline, approximation and uniform curves are: 0.035, 0.071 and 0.324. It is not surprising that the underlying nonlinearity reduced the degree of normality of the distributions. However, the distribution were not significantly non-Gaussian to raise any concern - the difference between figures 8.11 and 8.29 is almost negligible.

8.6 Conclusion

We have illustrated the benefits gained by designing model predictive controllers within the framework of probabilistic graphical models by showing:

1. Under the assumption of normality and linearity it is possible to convert stochastic quadratic objective functions into their deterministic equivalents. The analysis is closely related to the work of [64] and [65] but we have shown that these results are immediately obvious from within the framework of probabilistic graphical models. Thus it is possible to solve LQG objective type problems without resorting to stochastic dynamic programming.

2. We have generalised our analysis to stochastic MPC and shown that by using the statistically important metric, the Mahalanobis distance, we arrive at a technique for enforcing chance constraints which is very closely related to the approach taken by [58] and [59]. Under the assumptions of linearity and normality we have shown that constraint satisfaction is ensured. Due to the use of the Mahalanobis distance metric we

have provided some theoretical support for the use of the “ellipsoidal approximation” technique if the underlying system is nonlinear or not exactly Gaussian.

3. Combining the previous results we have shown that it is possible to write the joint chance constrained stochastic quadratic MPC problem as a deterministic quadratic MPC problem. Additionally we show that the joint chance constraints can be written in a linear format. The entire optimisation problem can then be written in the standard form for quadratic programming optimisation. Standard deterministic MPC solution and analysis techniques can then be used to solve the stochastic problem.

4. We have compared the effect different inference techniques have on the quality of the MPC. If the system is linear and Gaussian the Kalman filter is adequate. If there is significant departure from linearity or normality it can be beneficial to use the particle filter. The computational burden of the inference technique must also be taken into account when designing a controller.

In part III we generalise our graphical models to incorporate switching states. This allows us to design switching model predictive controllers.

Part III

Multiple model systems

Chapter 9

Inference using linear hybrid models

In this chapter we generalise the probabilistic graphical models of chapters 6 and 7 as shown in figure 9.1. We have added the discrete random variables $(s_0, s_1, s_2, \ldots)$, where each variable has M states, which we will call the switching variables. The goal of adding switching variables is to allow our graphical models to switch (or more precisely, choose based on the observation) between M different dynamical models. For the moment we restrict ourselves to linear transition functions i.e. we use linear state space models. The other variables retain their meaning as before. Models of this form are usually called switching Kalman filter models [47].

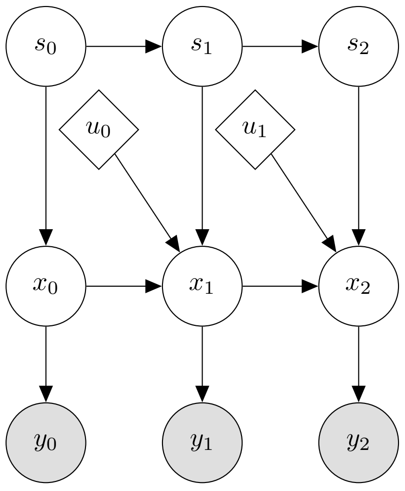

Figure 9.1: Graphical model used in this chapter.

One of the benefits of combining discrete switching variables with linear dynamical models is that it allows us to model nonlinear, even multi-modal, processes with linear models. Intuitively, we glue together linear models which each describe a nonlinear model in some region and use the switch to determine which one (or combination of) to use. The switch assigns a weight to each model based on its ability to explain the evidence.

We model this system as follows. Let s_t denote a discrete, time homogeneous, M state first order Markov chain with transition matrix P as discussed in chapter 4. Let each state $s_t = i$ be associated with a parameter set $(A_i, B_i, W_i, C_i, V_i)$ used to evaluate the dynamical model

$$\begin{aligned} x_{t+1} &= A_i x_t + B_i u_t + w_t \text{ with } \mathcal{N}(w_t|0, W_i) \\ y_{t+1} &= C_i x_{t+1} + v_{t+1} \text{ with } \mathcal{N}(v_{t+1}|0, V_i). \end{aligned} \tag{9.1}$$

If s_t were observed then (9.1) would simplify to a set of latent linear dynamical systems we could perform inference on using the methods investigated in chapter 6. However, we assume that s_t is a hidden random variable.

To fully specify the system we also require the prior distributions $p(s_0)$ and $p(x_0|s_0)$ as well as the switch transition matrix P. For the purposes of this dissertation we assumed that the switch transition matrix is available. This matrix can be inferred using the Baum-Welch algorithm [47] or it can be set using operator expertise as we will show later.

9.1 Exact filtering

The switching variables $(s_0, s_1, s_2, \ldots)$ are discrete random variables exactly like the ones seen in chapter 4. There we derived recursive analytic expressions for inference which were computationally inexpensive. The structure of the stochastic variables $(x_0, x_1, x_2, \ldots)$ and $(y_0, y_1, y_2, \ldots)$ are exactly the same as those found chapter 6. There we derived the famous Kalman filter equations which were also analytic, recursive and computationally inexpensive. Taking this into consideration, it seems plausible to believe that inference, specifically filtering, for hybrid systems like (9.1) can be formulated in a computationally feasible manner.

Unfortunately, it can be shown that this is not possible in general [38], [46] because the memory requirements scale exponentially with time. Loosely speaking one can see this by noting that at the first time step the system is described by a weighted set of M Kalman filter models (due to the linear assumption and the M switching indices). At time step two the system is described by a weighted set of M^2 Kalman filter models. Continuing in this manner we see that at time step t the memory requirement is M^t. Clearly this is computationally infeasible and calls for approximate methods to be used.

In literature many approximate filtering algorithms exist and it is not clear which is best. Two of the more popular methods include Gaussian sum filtering [3] and particle filtering based methods (specifically the Rao-Blackwellisation approach, see [13], [22]). Both of these methods take advantage of the Gaussian structure of the system and operate in a fixed memory space making them computationally attractive. We focus on particle based methods because it can be extended to nonlinear systems with ease.

9.2 Rao-Blackwellised particle filter

It is our objective to find the joint posterior distribution $p(s_{0:t}, x_{0:t}|y_{0:t})$. This joint posterior admits filtering of figure 9.1 if we discard the trajectory and focus only on s_t, x_t. By the chain rule (definition 3.19) we immediately have that $p(s_{0:t}, x_{0:t}|y_{0:t}) = p(s_{0:t}|y_{0:t})p(x_{0:t}|y_{0:t}, s_{0:t})$. Given $s_{0:t}$ we see that $p(x_{0:t}|y_{0:t}, s_{0:t})$ can be evaluated using the Kalman filter equations (see chapter 6) and thus we are only concerned with finding some approximation for $p(s_{0:t}|y_{0:t})$. This is the essence of the Rao-Blackwellised particle filter - taking advantage of the conditionally linear Gaussian nature of the system to analytically evaluate a part of the posterior distribution [22].

Using the formulation of the adaptive sequential importance sampling algorithm discussed in chapter 7 we can apply it to find an approximation of $p(s_{0:t}|y_{0:t})$. We set $\gamma_t(s_{0:t}) = p(s_{0:t}, y_{0:t})$ and $Z_t = p(y_{0:t})$ and then have that $\frac{\gamma_t(s_{0:t})}{Z_t} = p(s_{0:t}|y_{0:t})$ as desired. We then choose our proposal distribution $q_t(s_{0:t}|y_{0:t})$ to be recursive and follow the same procedure as before:

$$\begin{aligned}
w_t(s_{0:t}) &= \frac{\gamma_t(s_{0:t}, y_{0:t})}{q_t(s_{0:t}|y_{0:t})} \\
&= \frac{p(s_{0:t}, y_{0:t})}{q_t(s_{0:t}|y_{0:t})} \\
&\propto \frac{p(s_{0:t}|y_{0:t})}{q_t(s_{0:t}|y_{0:t})} \\
&\propto \frac{p(y_t|s_t)p(s_t|s_{t-1})}{q_t(s_t|s_{0:t-1}, y_{0:t})}\frac{p(s_{0:t-1}|y_{0:t-1})}{q_t(s_{0:t-1}|y_{0:t-1})} \\
&= \alpha_t(s_{0:t})w_{t-1}(s_{0:t-1}).
\end{aligned} \tag{9.2}$$

As before, we are not interested in the whole trajectory of the switching variable because we only need to perform filtering. Thus our proposal distribution can be chosen to be the prior i.e. $q_t(s_t|s_{0:t-1}, y_{0:t}) = p(s_t|s_{t-1})$. This is suboptimal but easy to sample from [22]. The incremental weight then simplifies to $\alpha_t(s_{0:t}) = p(y_t|s_t)$. We can evaluate this distribution by marginalising out x_t and using the properties of the Gaussian distributions

$$\begin{aligned}
\alpha_t(s_{0:t}) &= p(y_t|s_t) \\
&= \int_{x_t} p(y_t|x_t, s_t)p(x_t|s_{0:t}, y_{0:t-1}) \\
&= p(y_t|y_{0:t-1}, s_{0:t}) \\
&= \mathcal{N}\left(y_t|C_{s_t}A_{s_t}\mu_{t-1}, C_{s_t}\left(A_{s_t}\Sigma_{t-1}A_{s_t}^T + Q_{s_t}\right)C_{s_t} + R_{s_t}\right).
\end{aligned} \tag{9.3}$$

Where the subscript s_t denotes the state of the switching variable at time t [47]. Upon inspection we see that (9.3) is just the one step ahead prediction likelihood as discussed in section 6.2 [47]. Note that we will still need to resample the switching state from P periodically to prevent sample impoverishment.

We now have an efficient particle approximation of $p(s_t|y_t)$. To find the filtered posterior distribution as desired we note that $p(s_t, x_t|y_{0:t}) = \sum_i w_t(S_t^i)\delta(S_t^i, s_t)p(x_t|y_{0:t}, S_t^i)$ where S_t^i is the i$^{\text{th}}$ particle within the framework of section 7.1. Each particle thus consists of a weight, a switch sample and the sufficient statistics generated by the Kalman filter for a Gaussian i.e. a mean and covariance. The complete algorithm is shown below.

Rao-Blackwellised particle filter algorithm

For $t = 0$:

1. Sample $S_0^i \backsim p(s_0)$ and $\mu_{0|0}^i \backsim p(x_0|s_0)$.

2. Compute the weights $w_0(S_0^i) = p(y_0|S_0^i)$ where y_0 is the observation. Normalise $W_0^i \propto w_0(S_0^i)$.

3. Apply the update step of the Kalman filter to each particle i and associated parameters to find μ_0^i and Σ_0^i.

4. If the number of effective particles is below some threshold apply resampling with roughening $(W_0^i, S_0^i, \mu_0^i, \Sigma_0^i)$ to obtain N equally weighted particles $(\frac{1}{N}, \bar{S}_0^i, \bar{\mu}_0^i, \bar{\Sigma}_0^i)$ and set $(\bar{W}_0^i, \bar{S}_0^i, \bar{\mu}_0^i, \bar{\Sigma}_0^i) \leftarrow (\frac{1}{N}, \bar{S}_0^i, \bar{\mu}_0^i, \bar{\Sigma}_0^i)$ otherwise set $(\bar{W}_0^i, \bar{S}_0^i, \bar{\mu}_0^i, \bar{\Sigma}_0^i) \leftarrow (W_0^i, S_0^i, \mu_0^i, \Sigma_0^i)$

For $t \geq 1$:

1. Sample $S_t^i \backsim p(S_t^i|\bar{S}_{t-1}^i)$.

2. Compute the weights $\alpha_t(S_t^i) = p(y_t|S_t^i)$ and normalise $W_t^i \propto \bar{W}_{t-1}^i \alpha_t(S_t^i)$.

3. Apply the Kalman filter algorithm to μ_{t-1} and Σ_{t-1} for each particle i to find the sufficient statistics μ_t and Σ_t using the parameters corresponding to the state of S_t^i.

4. If the number of effective particles is below some threshold apply resampling with roughening $(W_t^i, S_t^i, \mu_t^i, \Sigma_t^i)$ to obtain N equally weighted particles $(\frac{1}{N}, \bar{S}_t^i, \bar{\mu}_t^i, \bar{\Sigma}_t^i)$ and set $(\bar{W}_t^i, \bar{S}_t^i, \bar{\mu}_t^i, \bar{\Sigma}_t^i) \leftarrow (\frac{1}{N}, \bar{S}_t^i, \bar{\mu}_t^i, \bar{\Sigma}_t^i)$ otherwise set $(\bar{W}_t^i, \bar{S}_t^i, \bar{\mu}_t^i, \bar{\Sigma}_t^i) \leftarrow (W_t^i, S_t^i, \mu_t^i, \Sigma_t^i)$

9.3 Rao-Blackwellised particle prediction

Like the particle predictor studied in the section 7.3, performing prediction using Rao-Blackwellisation is straightforward because there is no weighting (updating the particles based on the observation) step. Each particle's switching state is merely propagated forward using the proposal distribution (the transition matrix P) and the Kalman prediction algorithm is used to evaluate the predicted mean and covariance. For the sake of brevity we do not supply an algorithm because it is a straightforward simplification of the Rao-Blackwellised particle filter algorithm as shown above. The corresponding probabilistic graphical model is shown in figure 9.2.

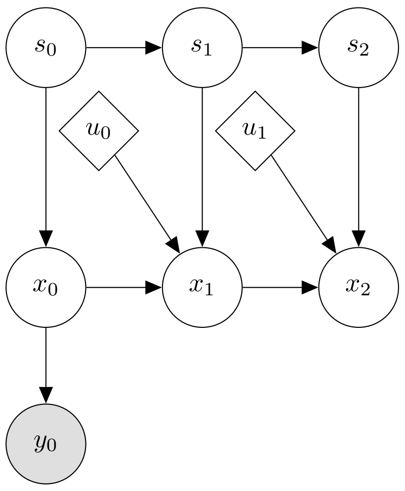

Figure 9.2: Rao-Blackwellised particle prediction graphical model.

9.4 Smoothing and Viterbi decoding

It is also possible to take advantage of the Gaussian structure in figure 9.1 to derive a so-called Rao-Blackwellised smoothing algorithm. We do not include it here because it is not necessary for the purposes of this dissertation. We refer the reader to the relevant literature [13], [22].

Viterbi decoding is likewise not within the scope of this dissertation and as such we refer the reader to [47] for more information. Suffice to say, by increasing the complexity of figure 9.1 we increase the difficulty of inference in general.

9.5 Filtering the CSTR

We now apply the Rao-Blackwellised particle filter to the CSTR introduced in chapter 5. The focus of this dissertation is on the application of probabilistic graphical models to control, therefore our investigation into the various aspects which improve or degrade filtering performance will be relatively superficial and will target factors which are most relevant only. We will briefly investigate 3 aspects influencing the accuracy of the filter:

1. The switch transition matrix P.
2. Using more state measurements.
3. Using more models.

Like in chapter 7 we do not investigate the effect increasing the number of particles will have on inference. The same reasons apply and we use the same motivation in selecting the number of particles we use in this chapter.

Note that we use the same parameters (e.g. noise covariances etc.) unless otherwise noted as used in chapter 6. Additionally we assume that the underlying model is the nonlinear CSTR model of chapter 5.

We begin our investigation by only measuring temperature and using 3 linear models, derived by linearising the nonlinear CSTR model at each nominal operating point. Since the CSTR has 3 nominal operating points we have 3 linear models. We compare the use of 2 different switch transition matrices P_1 and P_2 defined by

$$P_1 = \begin{pmatrix} 0.50 & 0.25 & 0.25 \\ 0.25 & 0.50 & 0.25 \\ 0.25 & 0.25 & 0.50 \end{pmatrix} \quad P_2 = \begin{pmatrix} 0.99 & 0.01 & 0.00 \\ 0.01 & 0.98 & 0.01 \\ 0.00 & 0.01 & 0.99 \end{pmatrix}. \tag{9.4}$$

The first index corresponds to the high temperature operating point ($M_1 = (A_1, B_1)$), the second index to the unstable operating point ($M_2 = (A_2, B_2)$) and the third index to the low temperature operating point ($M_3 = (A_3, B_3)$). Note that we assume $C_1 = C_2 = C_3$ because we do not change the way we measure the states between models; additionally we assume that the state and measurement noise is common to all models i.e. $W_1 = W_2 = W_3$ and $V_1 = V_2 = V_3$.

Intuitively P_1 indicates that we are less sure about the underlying dynamical transitions i.e. we believe it is possible for the system to jump from the dynamics of the low temperature operating point to the dynamics of the high temperature operating point. Conversely, P_2 indicates that we believe it is impossible for the system dynamics to jump from the low temperature operating point to the high temperature operating point without first transitioning through the unstable operating point. Clearly one could use operator expertise to set the switch transition matrix; alternatively, if an automatic algorithm is used to set P then operator expertise could be used to fault check and refine it.

Figure 9.3 shows the state space trajectory of the system we are attempting to perform filtering on. The operating points (points of linearisation) are superimposed on the state space.

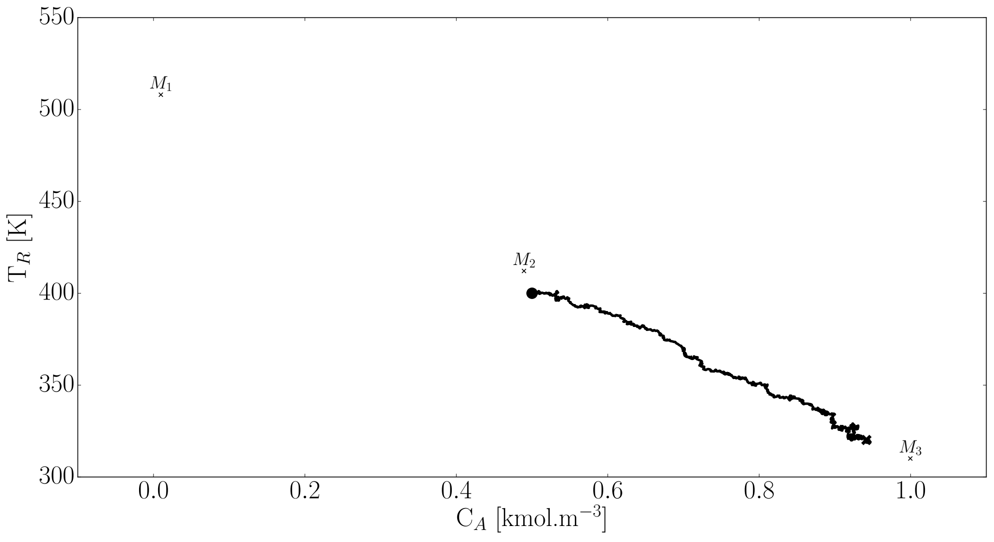

Figure 9.3: State space of the CSTR problem with the position of the 3 linear models superimposed thereupon. The trajectory followed by the system is also shown, the dot is the initial point and the cross the final point.

It is clear from figure 9.3 that we expect the filter to use M_2 initially and then switch to M_3 as time progresses. Figure 9.4 shows how the Rao-Blackwellised particle filter filters the CSTR over a simulation window of 150 minutes. The average concentration error is 22.03% and the average temperature error is 0.43% for the state estimator.

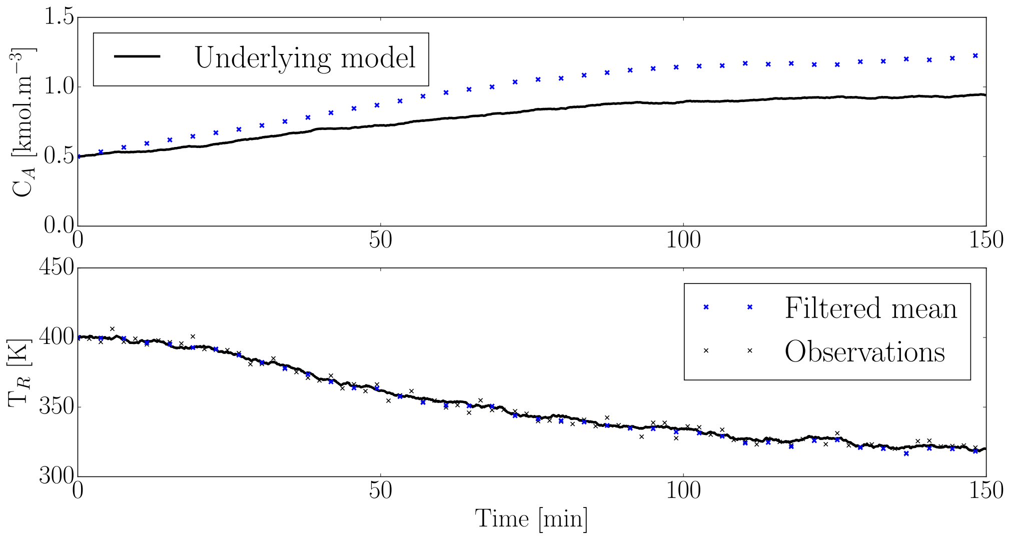

Figure 9.4: Filtering with the Rao-Blackwellised particle filter using 3 linear models and 500 particles. Switch transition matrix P_1 was used.

Figure 9.5 shows the state of the corresponding switching variable s_t over time. Since s_t is a discrete random variable we have that at each time slice $\sum_{i=1}^{M=3} s_t^i = 1$.

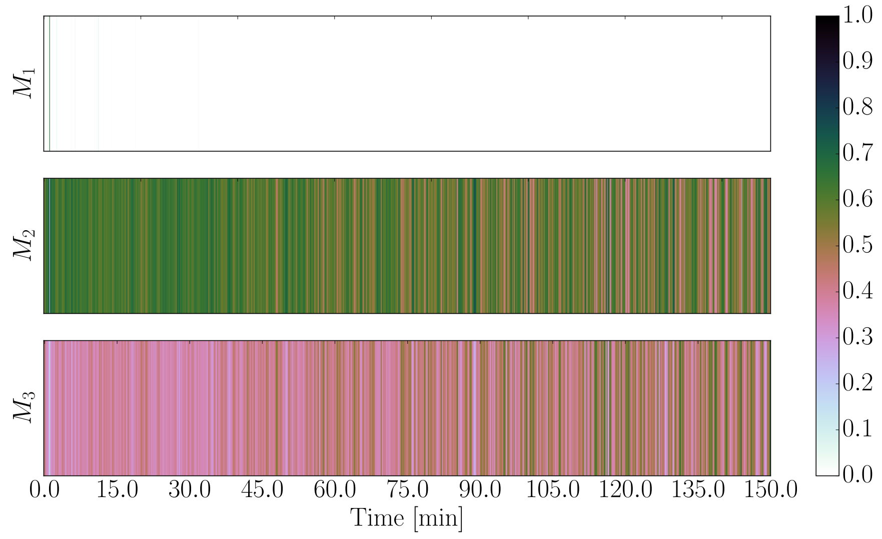

Figure 9.5: State of the switching variable s_t over time. The weight indicates the sum of the particle weights per model.

From figure 9.4 we see that the filtering error is quite large in the unmeasured state. This has been the trend when performing inference on an unmeasured state, however the magnitude of the error does not justify the use of the more complicated graphical model. Additionally, we see that there is no clear switching point in figure 9.5 - the filter relies on both M_2 an M_3 to estimate the state throughout the simulation. This is contrary to what we expected based on figure 9.3.

Figure 9.6 shows how the Rao-Blackwellised particle filter filters the CSTR over a simulation window of 150 minutes using P_2. The average concentration error is 4.56% and the average temperature error is 0.23% for the state estimator. This is a vast improvement over the case where P_1 was used.

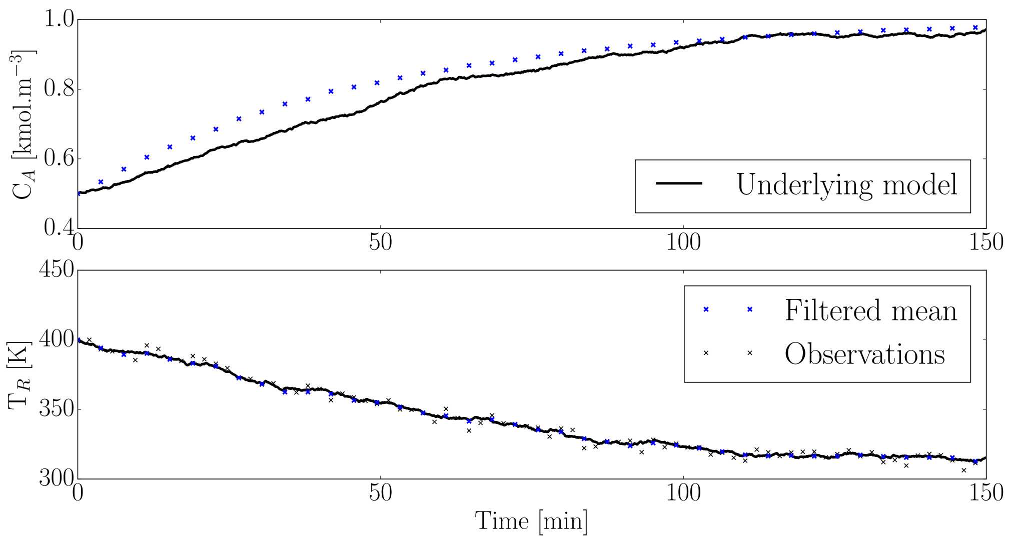

Figure 9.6: Filtering with the Rao-Blackwellised particle filter using 3 linear models and 500 particles. Switch transition matrix P_2 was used.

Figure 9.7 shows the state of the corresponding switching variable s_t over time.

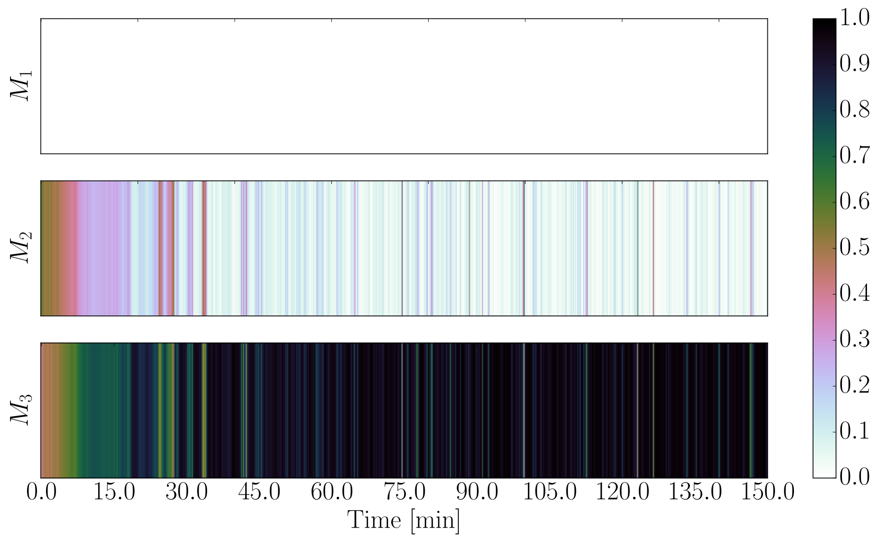

Figure 9.7: State of the switching variable s_t over time. The weight indicates the sum of the particle weights per model.

Unlike figure 9.5 we do see a clear model transition around the 20 minute mark in figure 9.7. This is the behaviour we expected - as the system moves away from the unstable operating point the corresponding graphical model becomes less important.

These results suggest that the switch transition matrix sets how "sticky" the model transitions are. The more vague they are, as in the case of P_1, the more unsure the filter is about which model is probably generating the observations. On the other hand, in the case of P_2, once the filter switched to the higher probability model it stayed there. The reason for this behaviour is simple: in the case of P_1 the approximate split of particles per model is equal. Since a relatively large number of particles represent M_2 it is reasonable that one of those particle's "guesses" will be near the observation. The filter then assigns a high weight to the inappropriate model and the cycle continues. In the case of P_2, once the model switches the transition matrix ensures that the number of particles representing M_2 is greatly reduced. This then improves filtering performance because the unlikely model does not have a large share of the available particles while the more likely model has more.

This behaviour is also desirable because it makes the switching variable have physical significance (not to mention that it improves filtering performance). However, the immediate drawback of the "sticky" approach is that the filter may be slow to switch models. Additionally if a machine learning approach is not used to infer the values of P it could become a tedious task to set P for a large system. Clearly the values used in P_2 were set by hand - more investigation is necessary to determine proper heuristics if this approach should be adopted in practice.

Next we investigate the effect measuring both states has on the accuracy of the filter. Due to the work in chapters 6 and 7 we expect that by measuring concentration we will increase the filter accuracy. We use the 3 model filter with P_2 to demonstrate that this is the case.

In figure 9.8 we see the filtering performance of the Rao-Blackwellised particle filter measuring both states. The average concentration and temperature error is 0.66% and 0.22%. This is a significant improvement over the tracking we saw in figure 9.6.

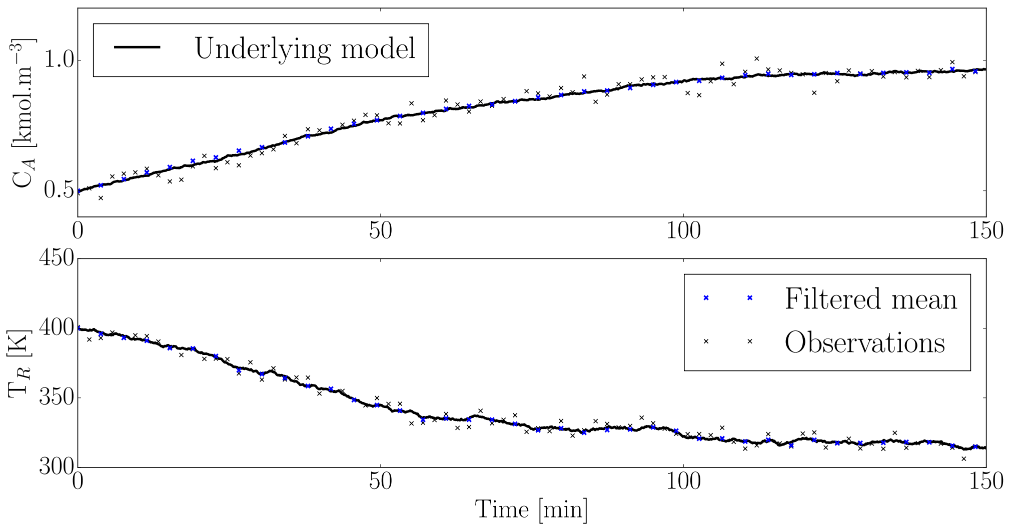

Figure 9.8: Filtering with the Rao-Blackwellised particle filter using 3 linear models and 500 particles. Switch transition matrix P_2 was used. Both states are measured.

In figure 9.9 we see the state of the switching variable over the simulation run. Like figure 9.7 we also see a clear switch occurring at approximately 15 minutes (actually at 20 minutes when measuring only one state). However, comparing figures 9.7 and 9.9 closely we see less "switching noise" in the latter. The second measurement allows the filter to compare both state predictions to discern between models. This is clearly beneficial.

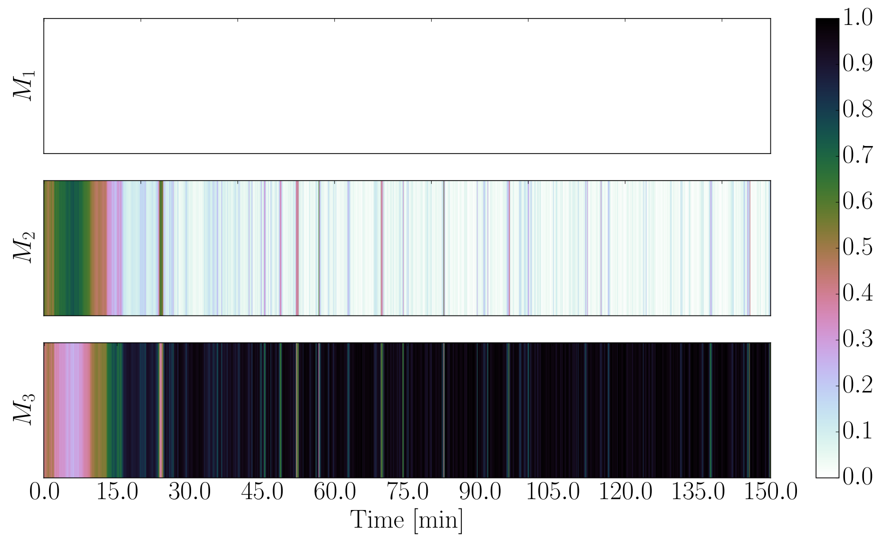

Figure 9.9: State of the switching variable s_t over time. The weight indicates the sum of the particle weights per model.

Finally, we investigate the effect using more models has on the filter which measures both states. We use the same 3 model filter as before (using P_2) but compare it to a 7 model filter. The state transition matrix for the 7 model filter is

$$P_3 = \begin{pmatrix} 0.98 & 0.00 & 0.00 & 0.00 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.98 & 0.00 & 0.01 & 0.01 & 0.01 & 0.00 \\ 0.01 & 0.00 & 0.98 & 0.00 & 0.00 & 0.01 & 0.01 \\ 0.00 & 0.00 & 0.00 & 0.98 & 0.00 & 0.01 & 0.00 \\ 0.00 & 0.01 & 0.00 & 0.00 & 0.99 & 0.00 & 0.00 \\ 0.01 & 0.01 & 0.01 & 0.01 & 0.00 & 0.96 & 0.00 \\ 0.00 & 0.00 & 0.01 & 0.00 & 0.00 & 0.00 & 0.99 \end{pmatrix}. \tag{9.5}$$

We again assume that the state and measurement noise is common to all models and that they all share the same observation matrix C. The values of P_3 were set using the same reasoning as before. Figure 9.10 show state trajectory of the system (like figure 9.3) but with the additional models superimposed thereupon. Clearly M_5, M_6 and M_7 correspond to high temperature, unstable and low temperature operating points.

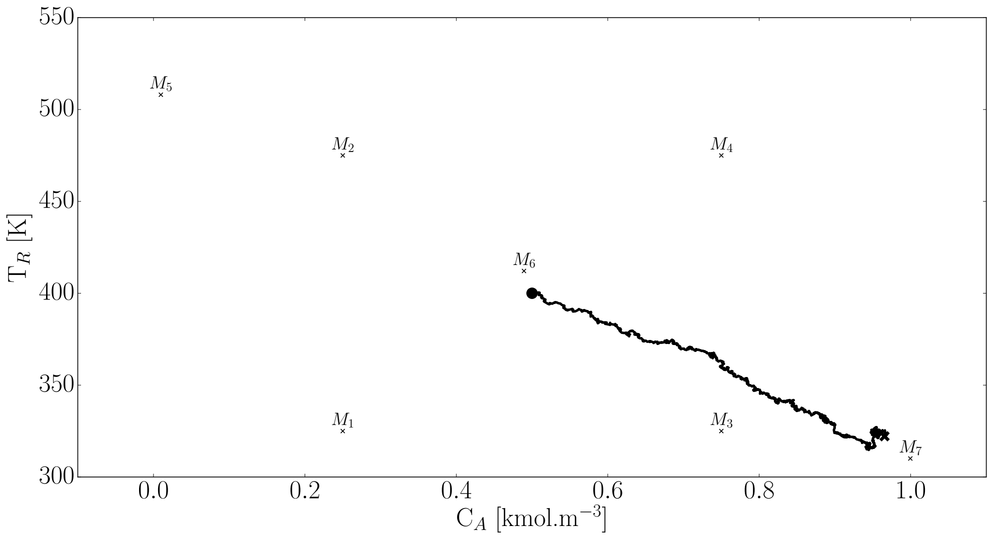

Figure 9.10: State space of the CSTR problem with the position of the 7 linear models superimposed thereupon. The trajectory followed by the system is also shown, the dot is the initial point and the cross the final point.

Figure 9.11 shows the effectiveness of the filter over the simulation window. The average concentration and temperature error is 0.71% and 0.23% respectively.

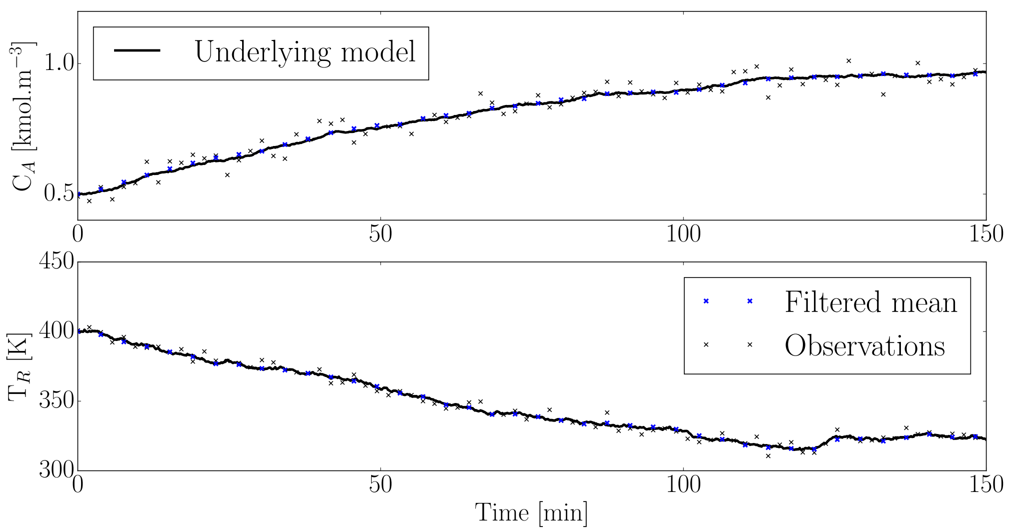

Figure 9.11: Filtering with the Rao-Blackwellised particle filter using 7 linear models and 500 particles. Switch transition matrix P_3 was used.

Interestingly enough we actually observe worse tracking performance when more models are used compared to the 3 model case with P_2. We expected the additional models to increase the effectiveness of the filter. Figure 9.12 shows the state of the corresponding switching variable s_t over time. A possible explanation for the performance degradation is evident here.

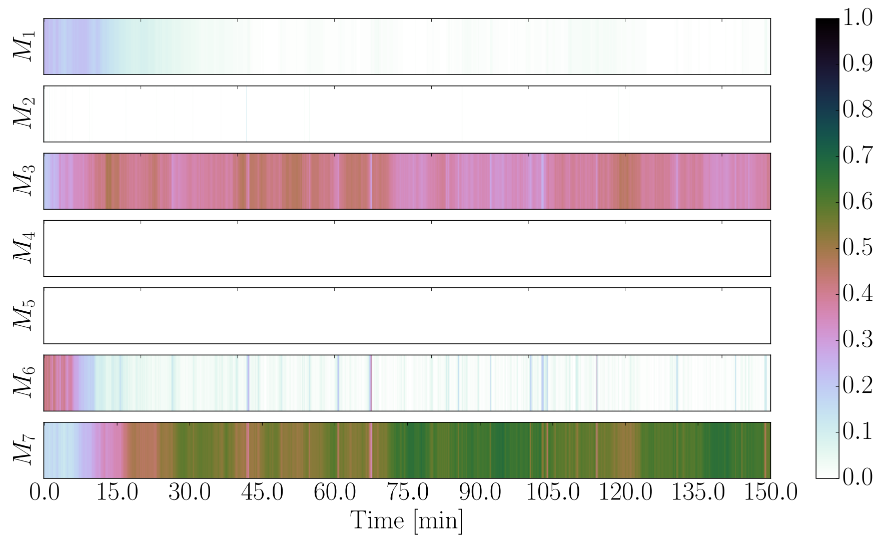

Figure 9.12: State of the switching variable s_t over time. The weight indicates the sum of the particle weights per model.

We certainly expected M_7 to be the dominant model near the end of the simulation; however M_3, while close to the low temperature operating point, played a significant role in the state estimate throughout the simulation. The filter uses a weighted combination of model predictions to estimate the current state; it is evident that there were a non-negligible number of particles which maintained the M_3 hypothesis. This implies that less particles were available to use the M_7 model and thus we see worse performance.

The crux of the problem is model overlap. While it is clear to a human that the system should only use M_7 near the end of the simulation the algorithm has no way of knowing this. It infers this based on the predictive ability of the models. Clearly M_7, M_3 and to a lesser extent M_1 and M_6 were all able to accurately predict the current state. For this reason they have non-negligible weights in figure 9.12.

We have in fact already come across this problem in figures 9.4 and 9.5. We saw that it is possible to attenuate this problem by making the switching transition matrix "stickier". Unfortunately this does not solve the underlying problem - the models are not different enough. Using more models would only make this problem worse.

It should be added that this problem is not necessarily bad for inference. The model switching and weighting allows the filter to accurately track regions between the linear models i.e. regions where no one model is accurate. From a filtering perspective this can be beneficial.

In the next chapter we implement switching model predictive controllers using the Rao-Blackwellised particle filter to select the best model for prediction.

Chapter 10

Stochastic switching linear control using linear hybrid models

In section 2.2 model switching MPC was introduced. In short, a set of models with corresponding binary integer variables are incorporated into the MPC optimisation problem. The optimisation algorithm changes the model it uses for prediction based on the location of the previous predicted state. In this way a number of models can potentially be used for prediction. It is desirable to change models if the system states move far away from the linearisation point of current linear model. It is hoped that the significant computational burden this introduces is offset by the increased predictive accuracy of the controller.

In chapter 8 we developed efficient stochastic controller algorithms (LQG and MPC) which use a single linear model for control. While it is possible to attempt to extend these algorithms to the aforementioned approach, the computational problems will persist because mixed integer programming is fundamentally more difficult than quadratic programming [27]. From a practical perspective one would like to reduce computational complexity because, especially for large problems, on-line optimisation can become problematic.

In chapter 9 the Rao-Blackwellised particle filter was introduced. Briefly, the filter uses a set of linear models, $M_i = (A_i, B_i)$ for each model i, to estimate the current state (we assume the system and measurement noise is common across all models as well as the observation matrix). The ability of each model to explain the observations is calculated in a Bayesian sense. This is used to weight the importance of each model's contribution to the current state estimate.

In this chapter we will attempt to combine the ideas of section 2.2, and chapters 8 and 9 to create a computationally efficient switching model controller algorithm. We assume that the underlying process dynamics are described by

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + w_{t+1} \\ y_{t+1} &= g(x_{t+1}) + v_{t+1} \end{aligned} \tag{10.1}$$

where f and g are the nonlinear transition and observation functions of the CSTR process introduced in chapter 5. It is assumed that x_t is a latent stochastic variable and y_t is an observed stochastic variable. We also assume that the models used for inference and control are linear and of the form

$$\begin{aligned} x_{t+1} &= A_i x_t + B_i u_t + w_{t+1} \\ y_{t+1} &= C x_{t+1} + v_{t+1} \end{aligned} \tag{10.2}$$

for model $M_i = (A_i, B_i)$. The noise terms retain their meaning from chapter 8. It is our aim to move the system states from the unstable (nominal) operating point to another operating point. This will clearly cause the system to traverse the state space and necessitate model switching. We first describe the intuition behind the proposed switching controller algorithm and then state the algorithm.

As mentioned before, it becomes desirable to have a mechanism to switch the underlying controller model if the system states move far away from the linearisation point of the current model. However, it is computationally difficult to perform this switching within the framework of the optimisation algorithm because it invariably necessitates the introduction of integer variables. We propose an algorithm which uses the Rao-Blackwellised particle filter to estimate the current state as well as the models which best describe the current observation. Based on the results of chapter 9 we expect the weight assigned to each model to skew in favour of the models which were linearised closest to the current state. Now we have two options to implement control at each time step[1]:

1. Use only the most likely model (the model with the highest switch weight) for controller prediction i.e. use $A^* = A_{\text{indmax}[s_t]}$. See section 10.1.1.

2. Use the weighted average (from the switch weight) of the models for controller prediction i.e. use $A^* = \sum_{i=1}^{M} s_t^i A_i$. See section 10.1.2.

Since it is not clear which approach is best we investigate both. This "best current model" is then used in the single model controller algorithms discussed in chapter 8. This approach falls squarely between the purely single model controllers, as discussed in chapter 8, and the switching model controllers, where the model switching occurs inside the optimisation problem, as discussed in section 2.2. By switching models outside the optimisation problem the scheme will necessarily be more computationally efficient than those found in section 2.2.

Switching controller algorithm:

1. Use a switching filter algorithm, e.g. the Rao-Blackwellised particle filter, to update the state estimates of the particle population given the current observation. See chapter 9 for more details.

2. Select the best current model to for control based on the model weights also supplied by the switching filter algorithm.

[1]Note that A^* is the model used for control in this explanation.

3. Use the mean and covariance information from the current posterior state estimate and the best current model (from step 2) within the context of the stochastic controller (LQG or MPC) formulation of chapter 8.

4. Repeat for the next observation.

The astute reader will notice that we are implicitly using the graphical model shown in figure 10.1 for state estimation (filtering) but the graphical model of figure 10.2 for model based prediction in step 3.

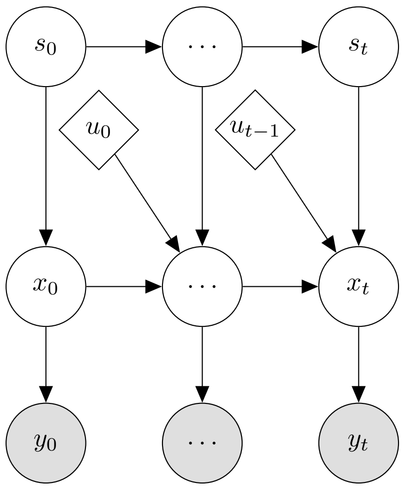

Figure 10.1: Graphical model used for state estimation.

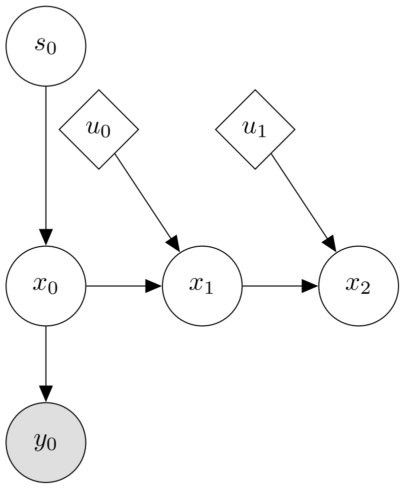

Figure 10.2: Simplified graphical model used for prediction. Within the context of prediction we have that $x_0 \leftarrow x_t$ and $s_0 \leftarrow s_t$ at each successive time step to simplify notation.

We are not using the graphical model associated with Rao-Blackwellised particle prediction (see section 9.3) because that would require that we incorporate stochastic model switching within the optimisation algorithm.

For the remainder of this chapter we assume that we have a bank of M linear models and that we measure both states. Each model is derived by linearising the non-linear CSTR model, found in chapter 5, around the nominal operating points discussed in the same chapter as well as section 9.5. We also use the switching transition matrices P_2, P_3 found in (9.4) where appropriate. All other parameters are the same as those found in chapter 8.

10.1 Unconstrained switching control

As mentioned earlier, we will investigate two approaches which can be used to implement the switching controller algorithm. The first approach, used in section 10.1.1, makes use of only the most likely model within the controller. The second approach, discussed in section 10.1.2, makes use of model averaging to construct a model for control.

10.1.1 Most likely model approach

Due to the analysis of section 8.1 we know that it is possible to convert the stochastic optimisation problem

$$\begin{aligned} &\min_{\mathbf{u}} \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\ &\text{subject to } x_{t+1} = A_i x_t + B_i u_t + w_t \end{aligned} \tag{10.3}$$

into the deterministic optimisation problem

$$\begin{aligned} &\min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q \mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N + \frac{1}{2}\sum_{k=0}^{N}\text{tr}(Q\Sigma_k) \\ &\text{with } \mu_{t+1} = A_i \mu_t + B_i u_t \\ &\text{and } \Sigma_{t+1} = W + A_i \Sigma_t A_i^T \end{aligned} \tag{10.4}$$

for each linear model (M_1, M_2, M_3) given that we have the current state estimate x_0, the model dynamics are linear and the underlying distributions are Gaussian. Throughout this chapter we make these assumptions[2]. As before, we also denote the mean and covariance of the current state estimate x_0 by $\mathbb{E}[x_0] = \mu_0$ and $\text{var}[x_0] = \Sigma_0$. We use a prediction horizon of $N = 150$ i.e. 15 minutes into the future. We have that (10.3) is equivalent to (10.4) under the aforementioned assumptions.

Given this we apply the switching controller algorithm within the context of the LQG controller i.e. given a model M_i from the filter we solve the LQG problem and implement that input. In light of our analysis in chapter 8 it is clear that the switching controller algorithm is straightforward to implement because it simplifies to M deterministic LQR controllers.

As mentioned before we only use the most likely model for control purposes here. By only selecting one model to use for control we dramatically simplify the control problem. It allows us to use the controllers of chapter 8 directly.

[2]Note that the underlying model is clearly nonlinear but the model used for prediction and inference is linear.

We study 4 control problems using the switching controller algorithm in this chapter. Problems 1 and 2 allow the controller to switch between 3 linear models and problems 3 and 4 allow the controller to switch between 7 linear models. Furthermore, problems 1 and 3 seek to drive the system to the low temperature operating point i.e. a concentration set point of 0.998 kmol.m^{-3} while problems 2 and 4 seek to drive the CSTR to a concentration set point of 0.90 kmol.m^{-3}. In all cases we use the switching LQG controller as shown in (10.3).

We investigate the first problem in figures 10.3 to 10.5. In figure 10.3 we see the state space trajectory of the system under control. We expect M_2 to be active initially after which only M_3 should be active.

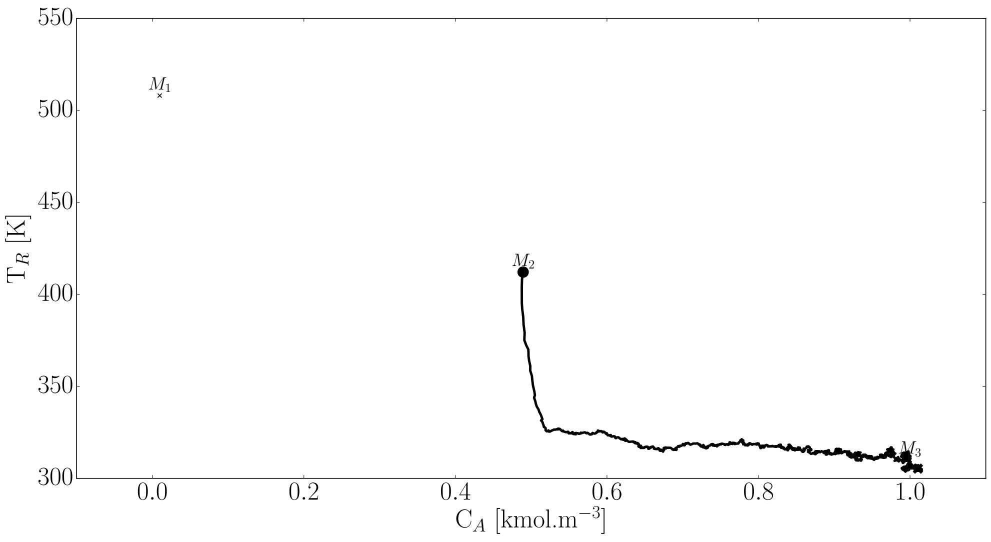

Figure 10.3: State space trajectory of the non-linear CSTR under control of the LQG switching controller algorithm. The initial point was $(0.49, 412)$

Figure 10.4 confirms the behaviour we expected: initially M_2 best explained the observations but M_2 gives way to M_3 throughout the rest of the simulation.

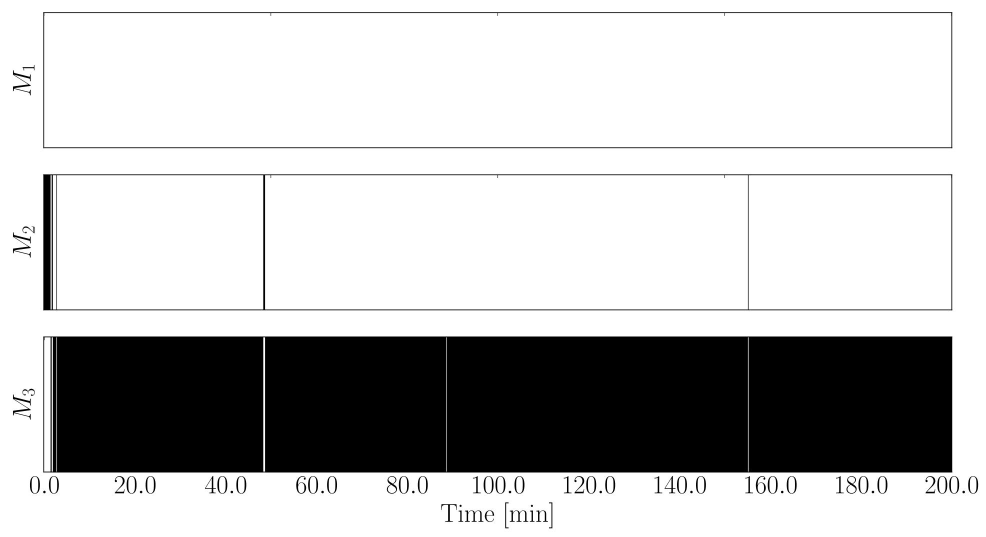

Figure 10.4: Most likely model used for control at each time step over the simulation. Black indicates the model is active.

However, it is clear that there are some problems in figure 10.4. There does seem to be some slight switching noise. Given that we are using the sticky switching transition matrix P_2 it is clear that model overlap is causing problems. The same problem was identified in section 9.5. In figure 10.5 we see the set point tracking performance of the switching controller. It is clear that the controller tracks the set point.

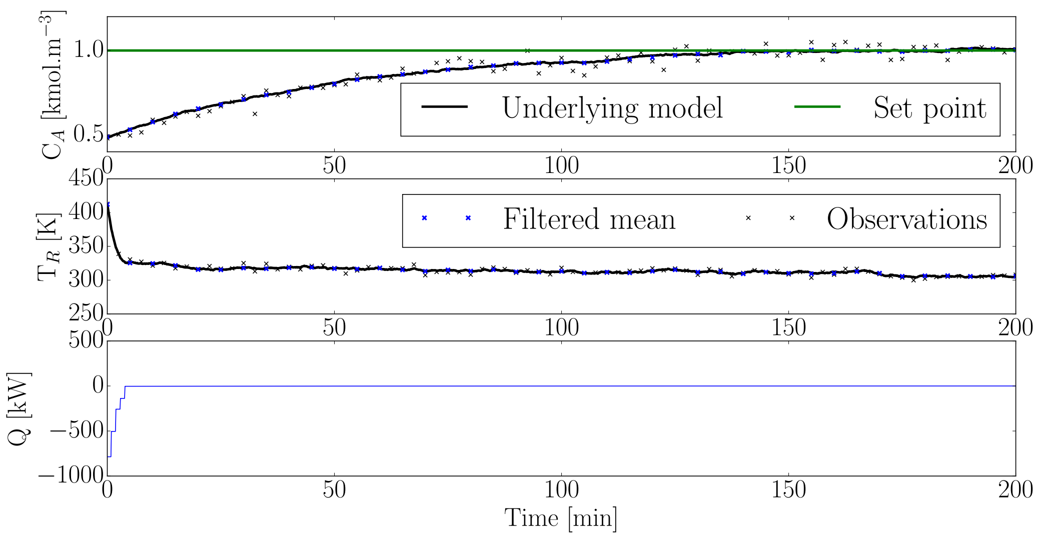

Figure 10.5: Set point tracking and controller input for the LQG 3 model switching controller algorithm. The initial point was $(0.49, 412)$.

Based on figures 10.4 and 10.5 it would be too easy to surmise that the controller algorithm works. Unfortunately this is not the case in general. In figures 10.6 and 10.7 we study problem 2: tracking the concentration set point of 0.90 kmol.m^{-3}. In figure 10.6 we see significant model switching noise.

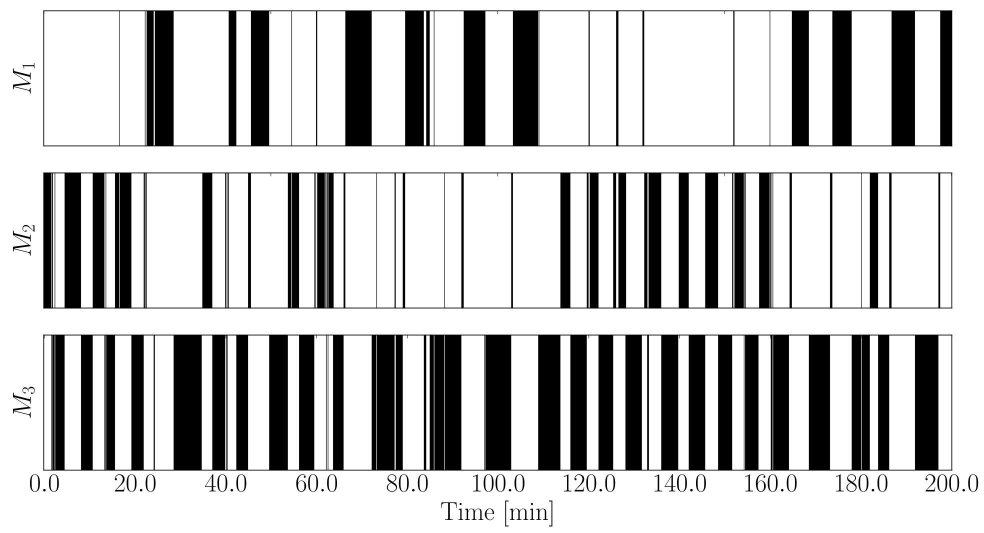

Figure 10.6: Most likely model used for control at each time step over the simulation. Black indicates the model is active.

In figure 10.7 the detrimental consequence of the switching noise is evident. The controller is completely unstable and oscillates.

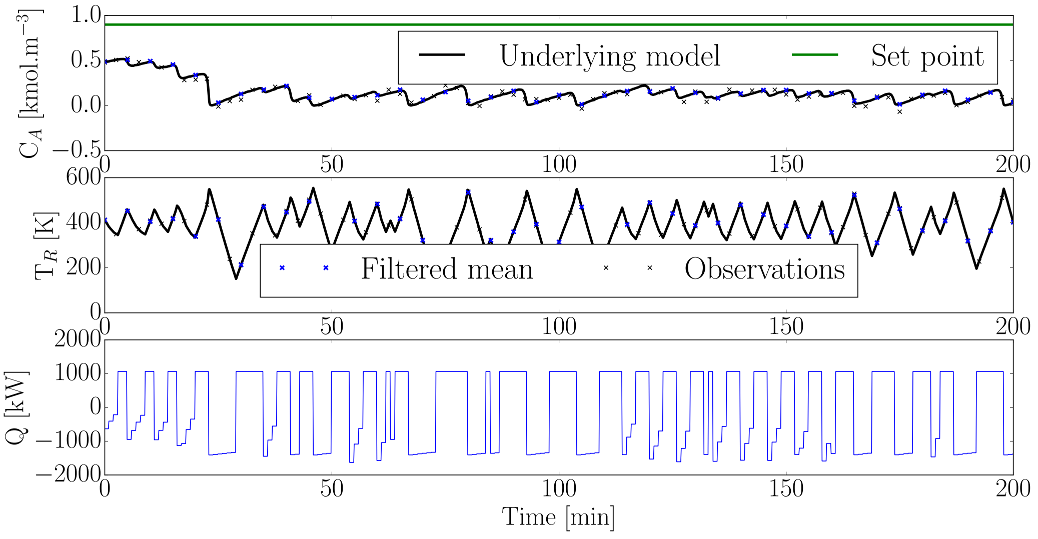

Figure 10.7: Set point tracking and controller input for the LQG 3 model switching controller algorithm. The initial point was $(0.49, 412)$.

It is clear that the oscillations are caused by the filter's inability to stick to a model. There are two major problems with the switching controller algorithm as adopted in this chapter:

1. Fundamentally we are using an inappropriate model for controller prediction during the initial period of the simulation. If we stayed near the current position in state space then the most likely model would predict the future well and thus result in good control. However, we are projecting the controlled states into regions where the current model control is based upon may not a good approximation at all. This is a fundamental problem of our approach - it would be better to incorporate the model switching within the controller prediction (optimisation) process, but this is exactly what we want to avoid due to the computational burden this introduces!

2. The switching noise is problematic because it can cause the controller to use a model which is good locally (for the current observation) but inappropriate with regard to the true underlying position of the system in state space. However, from an inference perspective the noise is not necessarily undesirable. Switching noise can improve the state estimate accuracy - especially in regions between models. It is also important in allowing the filter to switch punctually: making the switching transition matrix too static retards the sensitivity the filter has to model changes. It is possible to address the issue of switching noise without making the switch transition matrix too static. The augmented switching filter model, shown in figure 10.8, is a candidate for this.

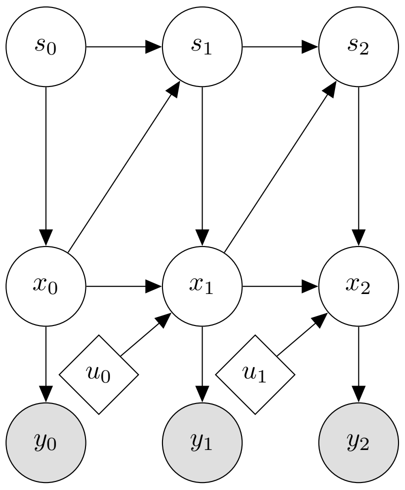

Figure 10.8: Augmented switching filter graphical model.

Using this model it is possible to model the influence the state variables $(x_0, x_1, ...)$ have on the switching variables $(s_0, s_1, ...)$. Plausibly this could ameliorate switching noise while not making the switching transition matrix too sticky. For example, the switch transition matrix could, in this case, be a function of the location of the current system state. However, since this would require that we modify the graphical model of figure 10.1 we leave it for future work.

We rather attempt to ameliorate these problems by extending the number of models available to the filter. We use more models to bridge the gap between the current most likely model and where it projects the states during prediction. In figure 10.9 we illustrate the state space trajectory followed by the system when it has 7 models available for inference and control.

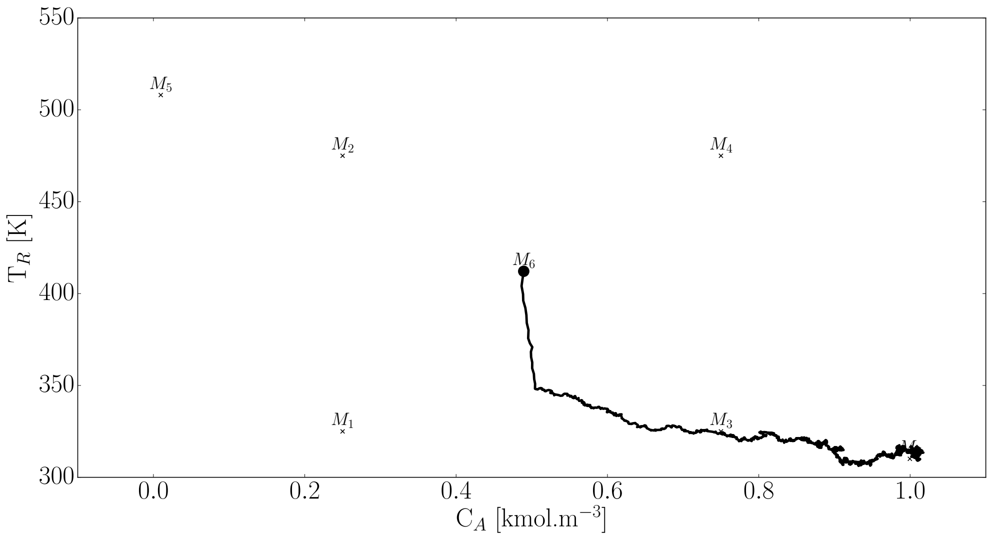

Figure 10.9: State space trajectory of the non-linear CSTR under control of the LQG 7 model switching controller algorithm. The initial point was $(0.49, 412)$

In figures 10.10 and 10.11 we again attempt to steer the system from the unstable operating point to the low temperature operating point. Figure 10.10 shows which models were active over the simulation window.

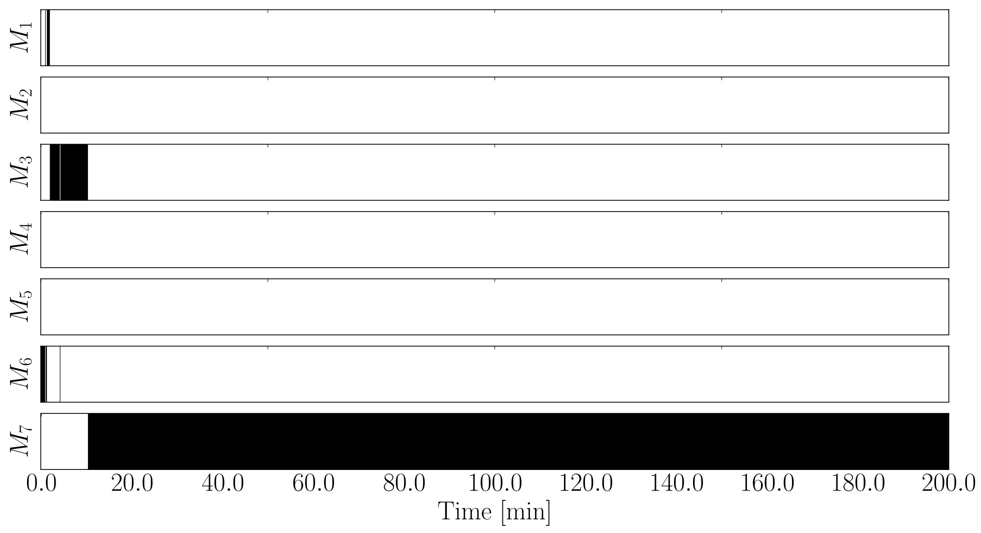

Figure 10.10: Most likely model used for control at each time step over the simulation. Black indicates the model is active.

While there is slight switching noise the model selection is exactly what one would expect.

Figure 10.11 shows the controller set point tracking performance over the simulation window.

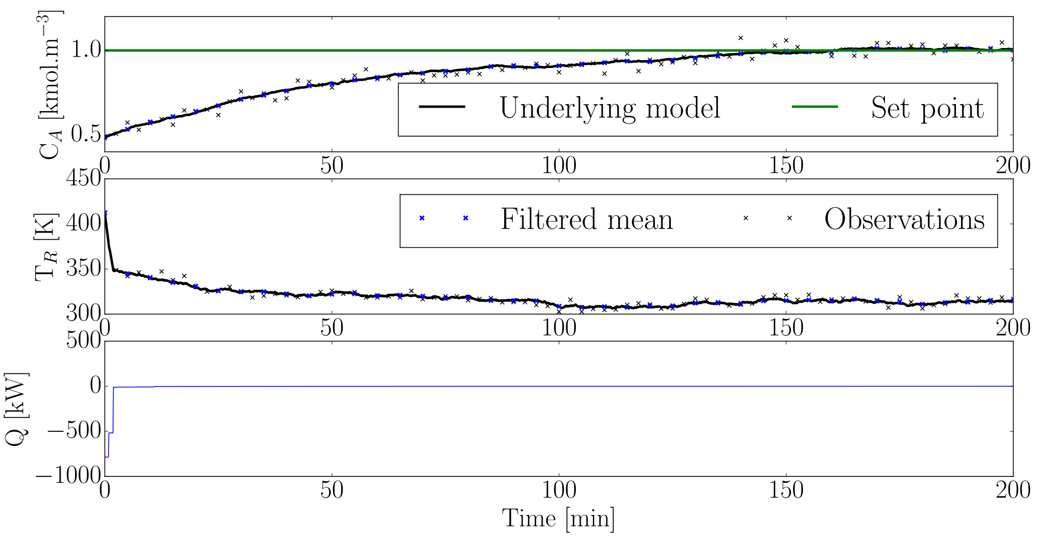

Figure 10.11: Set point tracking and controller input for the LQG 7 model switching controller algorithm. The initial point was $(0.49, 412)$.

Like figure 10.5 we also have a stable, reference tracking controller. This is not surprising because the model switching/selection was reasonable. In figure 10.12 and 10.13 we again attempt to steer the system to a concentration set point of 0.9 kmol.m^{-3}. Since we introduced M_3 to serve as a bridge between M_6 and M_7 (since the set point is between these two models) we expect better performance than in figure 10.6 and 10.7. Unfortunately this is not the case as may be seen in figures 10.12 and 10.13.

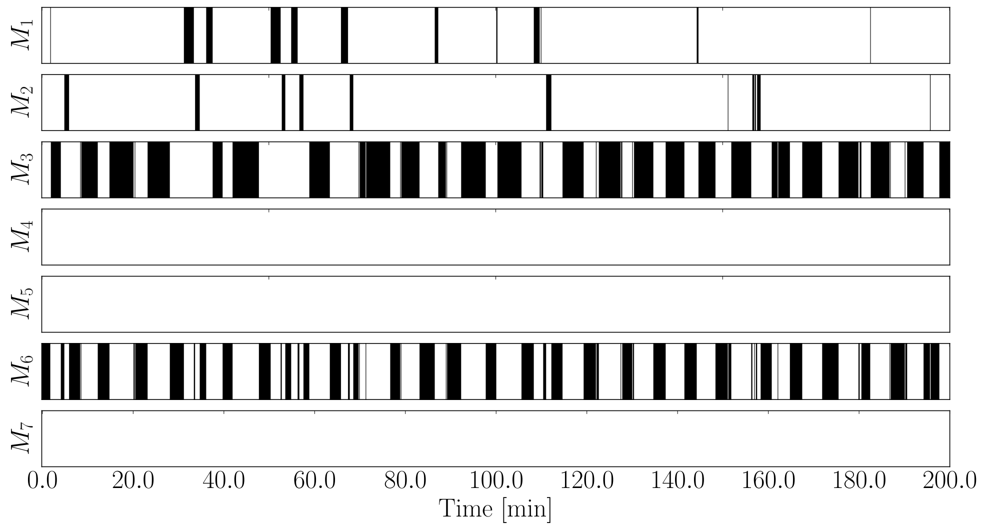

Figure 10.12: Most likely model used for control at each time step over the simulation. Black indicates the model is active.

The same oscillating switching noise and unstable control is present here as there was in figures 10.6 and 10.7.

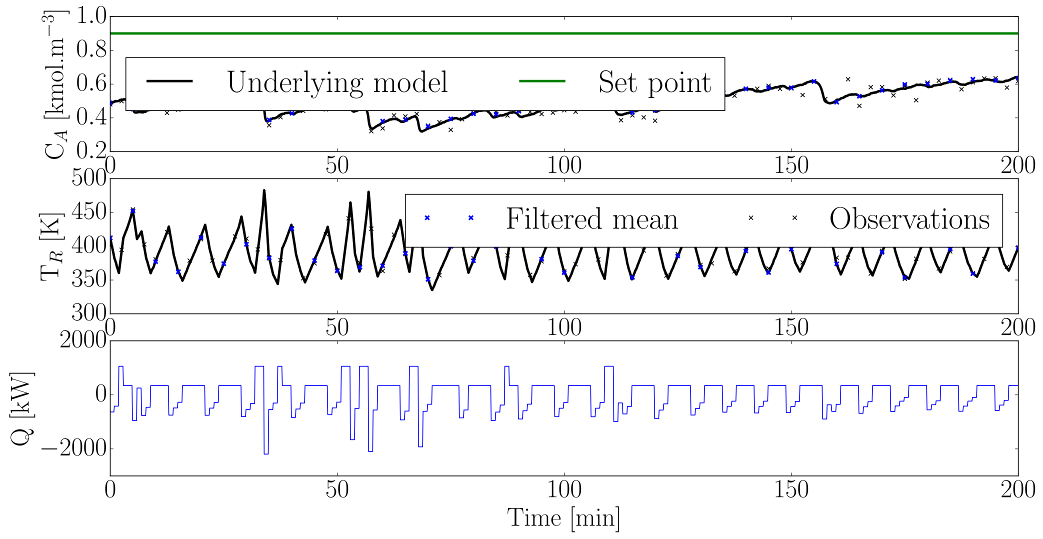

Figure 10.13: Set point tracking and controller input for the LQG 7 model switching controller algorithm. The initial point was $(0.49, 412)$.

The underlying reasons for the instability are the same: we are using models to predict into regions where they are inaccurate and the filter does not robustly enough isolate the model closest to the current system location in state space. The combination of these two problems make effective control impossible.

10.1.2 Model averaging approach

The fundamental problem with the switching controllers of section 10.1.1 is that an inappropriate model was used for prediction. Unfortunately using a weighted average of all the models, based on their probability with respect to the switching variable s_t, will exacerbate this problem. For this reason we do not explore this approach.

10.2 Conclusion

The goal of the switching controller algorithm was to drastically reduce the computational burden introduced by modern switching controller implementations as discussed in section 2.2. The approach of switching the model used for control outside the optimisation algorithm, while attractive intuitively, suffers from the fundamental problem that a bad model is used to predict the future states.

The switching LQG controllers introduced in this chapter were not robust against this type of problem. Additionally, we also had severe switching noise which led to controller oscillation. From a fundamental point of view it seems as if the current approach of incorporating the model switching inside the optimisation algorithm has the most promise of yielding good control.

It should be investigated whether it is feasible to incorporate the Rao-Blackwellised particle prediction (see section 9.3) within the controller optimisation process.

Chapter 11

Inference using nonlinear hybrid models

In this chapter we study the same graphical model as chapter 9, shown in figure 11.1 for convenience, but we drop the assumption that the dynamic models used for inference are linear. The variables retain their meaning as before.

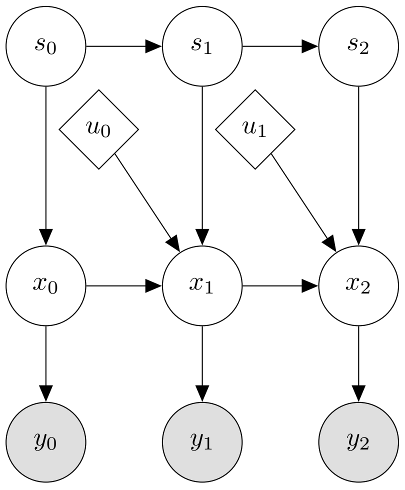

Figure 11.1: Graphical model used in this chapter.

Intuitively we are now using the switching variables to decide which nonlinear model (or combination of) better describes the observed system behaviour. At each point in time we desire a weighted set of nonlinear models with the weight proportional to the ability of the model to explain the plant behaviour. Such a system could be used to describe significant model changes e.g. catalyst degradation in our CSTR or a reactor which breaks suddenly etc..

We model this system as follows. Let s_t denote a discrete M state first order Markov chain with transition matrix P as discussed in chapter 9. Let each state $s_t = i$ be associated with a model set (f_i, g_i, W_i, V_i) used to evaluate the dynamical model

$$\begin{aligned} x_{t+1} &= f_i(x_t, u_t, w_{t+1}) \text{ with } w_{t+1} \backsim \mathcal{N}(0, W_i) \\ y_{t+1} &= g_i(x_{t+1}, v_{t+1}) \text{ with } v_{t+1} \backsim \mathcal{N}(0, V_i). \end{aligned} \tag{11.1}$$

In this dissertation we assume that the the noise distributions are Gaussian but there is no fundamental reason why they cannot be arbitrary. To fully specify the system we again require the prior distributions $p(s_0)$ and $p(x_0|s_0)$ as well as the stochastic matrix P. In this chapter we manually specify the matrix P but it can be computed using the Baum-Welch algorithm.

11.1 Exact filtering

By extending the model to incorporate nonlinear models it becomes even more difficult to perform inference. It is clear that for the type of systems we consider here no exact inference algorithm, which is computationally feasible, exists [47]. We again turn to approximate inference algorithms.

Note that we cannot apply Rao-Blackwellisation (i.e. analytically evaluate the stochastic dynamical system component and approximate the switching component) as before because the dynamic models used for inference are no longer linear. We use the adaptive sequential importance resampling, i.e. the bootstrap particle filter, algorithm as discussed in chapter 7.

11.2 Switching particle filter

We cannot analytically evaluate any part of the desired posterior distribution $p(s_{0:t}, x_{0:t}|y_{0:t})$ in a computationally feasible manner, so we must apply the adaptive sequential importance resampling algorithm to the entire state space of figure 11.1. The algorithm follows straightforwardly from our discussion in section 7.1 [47]. We merely state the proposal distribution and incremental weight function we sample from:

$$\begin{aligned} q_t(s_t, x_t|s_{0:t-1}, x_{0:t-1}, y_{0:t}) &= p(s_t|s_{t-1})p(x_t|s_t, x_{t-1}) \\ \alpha_t(s_{0:t}, x_{0:t}) &= p(y_t|x_t, s_t). \end{aligned} \tag{11.2}$$

Applying the algorithm is a straightforward extension of the bootstrap particle filter introduced in section 7.2 given the weighting function and proposal distribution as shown below.

Switching particle filter algorithm
For $t = 0$:

1. Sample $S_0^i \backsim p(s_0)$ and $X_0^i \backsim p(x_0|s_0)$.
2. Compute the weights $w_0(S_0^i, X_0^i) = p(y_0|S_0^i, X_0^i)$ where y_0 is the observation. Normalise $W_0^i \propto w_0(S_0^i, X_0^i)$.

3. If the number of effective particles is below some threshold apply resampling with roughening (W_0^i, S_0^i, X_0^i) to obtain N equally weighted particles $(\frac{1}{N}, \bar{S}_0^i, \bar{X}_0^i)$ and set $(\bar{W}_0^i, \bar{S}_0^i, \bar{X}_0^i) \leftarrow (\frac{1}{N}, \bar{S}_0^i, \bar{X}_0^i)$ otherwise set $(\bar{W}_0^i, \bar{S}_0^i, \bar{X}_0^i) \leftarrow (W_0^i, S_0^i, X_0^i)$

For $t \geq 1$:

1. Sample $S_t^i \backsim p(S_t^i|\bar{S}_{t-1}^i)$ and $X_t^i \backsim p(X_t^i|S_t^i, \bar{X}_{t-1}^i)$.
2. Compute the weights $\alpha_t(S_t^i, X_t^i) = p(y_t|S_t^i, X_t^i)$ where y_t is the observation. Normalise $W_t^i \propto W_{t-1}^i \alpha_t(S_t^i, X_t^i)$.
3. If the number of effective particles is below some threshold apply resampling with roughening (W_t^i, S_t^i, X_t^i) to obtain N equally weighted particles $(\frac{1}{N}, \bar{S}_t^i, \bar{X}_t^i)$ and set $(\bar{W}_t^i, \bar{S}_t^i, \bar{X}_t^i) \leftarrow (\frac{1}{N}, \bar{S}_t^i, \bar{X}_t^i)$ otherwise set $(\bar{W}_t^i, \bar{S}_t^i, \bar{X}_t^i) \leftarrow (W_t^i, S_t^i, X_t^i)$

11.3 Switching particle prediction

The prediction of the hybrid nonlinear states follows in an analogous manner to the prediction algorithm found in section 7.3. We do not supply an algorithm because it is a straightforward simplification of the switching particle filter algorithm seen above: effectively there is no weight update step because there is no observation. The corresponding graphical model is shown in figure 11.2.

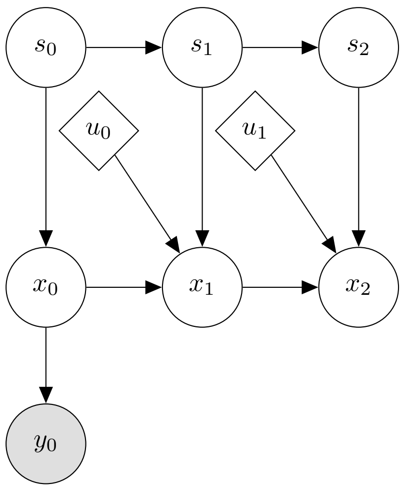

Figure 11.2: Switching particle prediction graphical model.

11.4 Smoothing and Viterbi decoding

Like chapter 9 we refer the reader elsewhere for a detailed discussion on both smoothing and Viterbi decoding [47]. It suffices to say that given the nonlinear dynamics the aforementioned inference will be beyond the scope of this dissertation.

11.5 Filtering the CSTR

In this chapter we illustrate the use of the switching particle filter using two nonlinear dynamical model derived from the familiar CSTR example of chapter 5. Since the graphical model of chapter 9 is identical to that of figure 11.1 we expect that the general trends discussed in section 9.5 to hold here as well.

For the purposes of illustration we assume a scenario where the rate constant of the CSTR decreases by an order of magnitude. This scenario is not completely arbitrary, for example, this could be caused by catalyst degradation due to some environmental factor. It is our aim to infer when this happens and to be able to track the states accurately despite the significant model change. Therefore we will have one nonlinear model of the healthy plant M_1 and one nonlinear model of the faulty plant M_2.

Note that the character of the inference we are attempting to do is fundamentally different from that of chapter 9 but that the underlying graphical models are the same. In chapter 9 the Rao-Blackwellised particle filter switched between models which all attempt to describe the same physical system albeit in different regions of the state space. In this chapter the switching particle filter will attempt to switch between models which describe completely different physical systems. This difference informs our choice of the switch transition matrix.

We use 500 particles during all runs for the switching particle filter and use the switching transition matrix $P_1 = \begin{pmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{pmatrix}$. For the particle filter, used for a comparative base, we use 200 particles. We spent much time in section 9.5 discussing the effect the switch transition matrix has on model selection. The form of the matrix is motivated by physical considerations as well: once the catalyst denatures it is unlikely to fix itself. Thus, once the model breaks, switches from M_1 to M_2, it is unlikely to switch back. In all the simulations the catalyst denatures at 50 minutes into the run.

We conduct two brief, but illustrative, investigations comparing the effectiveness of the switching particle filter and the particle filter. In both cases the particle filter uses the healthy system model - the benefit of the additional complexity of the switching particle filter model is to be highlighted here. In the first investigation we measure only temperature and in the second we measure both concentration and temperature.

In figure 11.3 we illustrate the tracking performance[1] of the particle filter on the system. Note that the simulation window is very long - 600 minutes.

[1]Unfortunately we cannot use the average tracking performance measures used previously. Since the filter approaches a concentration of 0 kmol.m^{-3} the average error estimates are not accurate: there is division by very small numbers. We thus rely on a visual comparison.

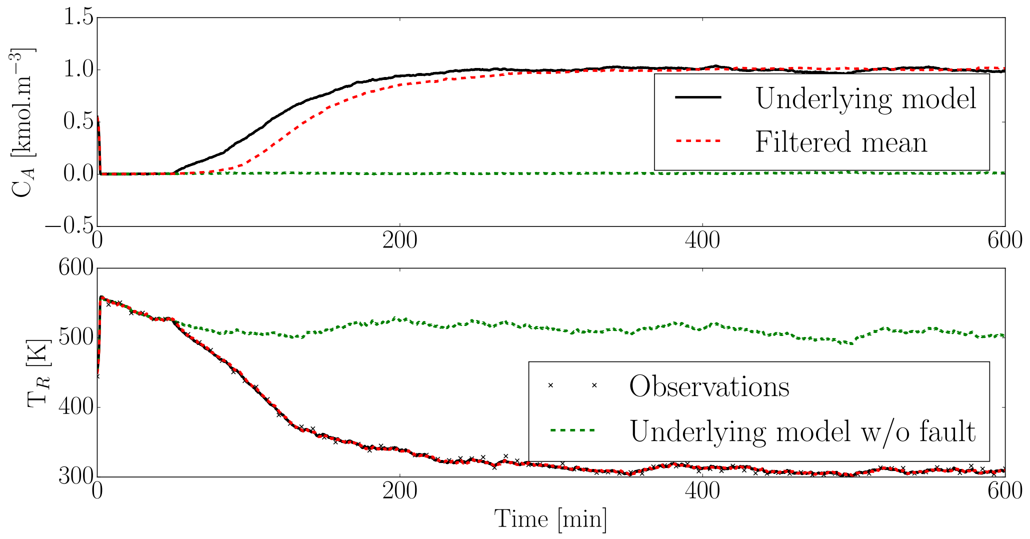

Figure 11.3: Particle filter using 200 particles tracking the CSTR where the catalyst denatures at 50 minutes. Only temperature is measured.

It is clear that the particle filter tracks the temperature well, because it is measured, but tracks the concentration very poorly. It takes almost 300 minutes before the particle filter estimates the concentration reliably. Clearly the model mismatch causes the filter's poor performance - compare this to the excellent tracking in figure 7.5.

In figure 11.4 we see the tracking performance of the switching particle filter measuring only temperature.

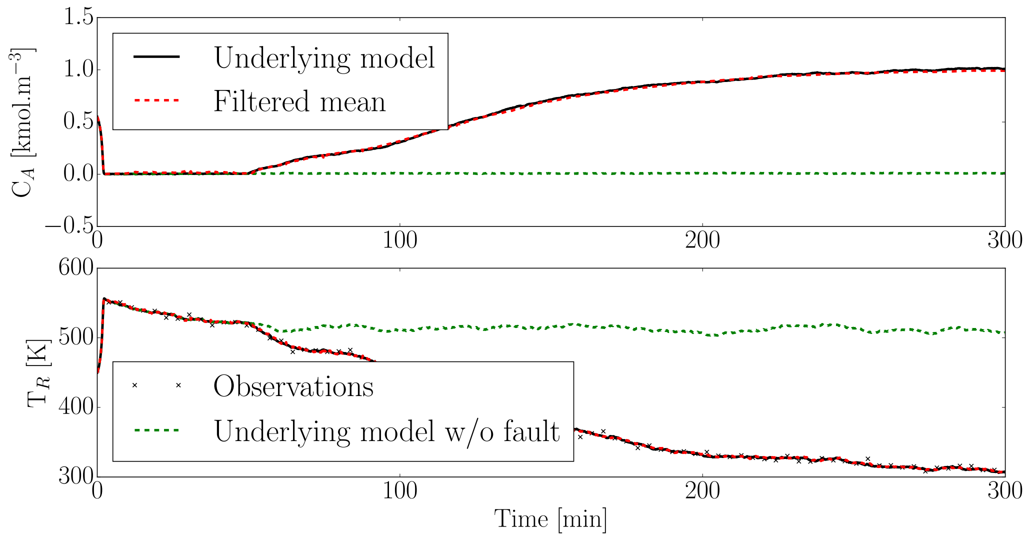

Figure 11.4: switching particle filter using 500 particles tracking the CSTR where the catalyst denatures at 50 minutes.

It is clear that the switching particle filter tracks the states very well. However, figure 11.5 indicates that we have the some switching noise problems.

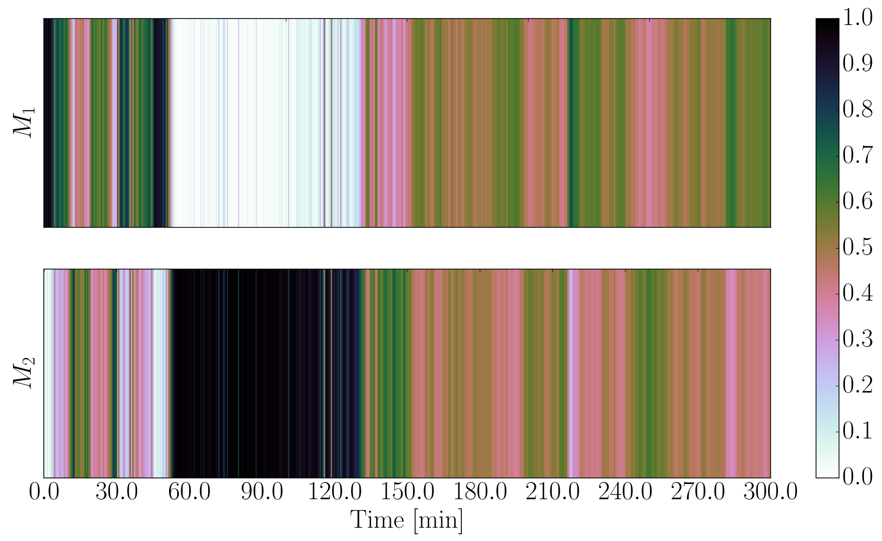

Figure 11.5: Switching particle filter measuring only temperature. The switching weight of each particle is shown per time step. M_1 corresponds to the healthy plant and M_2 to the broken plant.

It is not surprising that there is switching noise: the graphical models of this and chapter 9 are the same. However, we do see that the switching particle filter switches models at about 50 minutes. This indicates that the filter effectively identifies when the process fault occurs. It seems that after the fault has been identified there is a period where the filter reliably isolates the correct underlying model thereafter, from about 130 minutes, both models are equally likely. It seems there is a regime near the unstable operating point where the models are maximally different. Conversely, near the low temperature operating point (near the end of the simulation) the models are quite similar. This is physically believable because both those operating points correspond to a system where almost no conversion occurs; therefore broken or not the models would generate the similar predictions. Therefore, switching noise caused by model overlap is only a problem in this region.

Figure 11.6 illustrates the filtering performance of the particle filter using both state measurements.

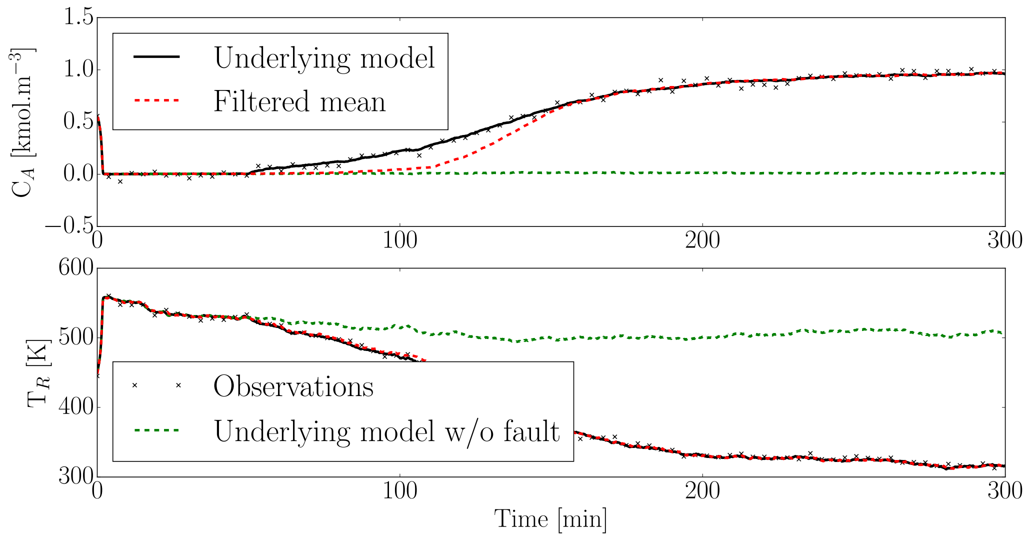

Figure 11.6: Particle filter using 200 particles tracking the CSTR where the catalyst denatures at 50 minutes. Both states are measured.

Clearly measuring both states is beneficial in terms of filter performance. The state estimation deviation seen in figure 11.3 is significantly less here - by approximately 150 minutes the particle filter is tracking the underlying system.

Figure 11.7 shows the filtering performance of the switching particle filter.

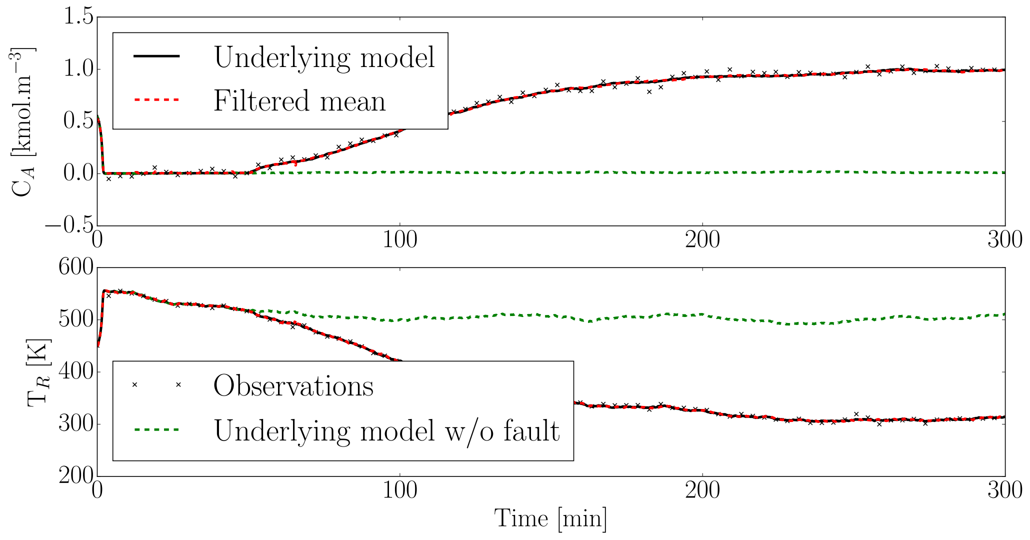

Figure 11.7: Switching particle filter using 500 particles tracking the CSTR where the catalyst denatures at 50 minutes. Both states are measured.

Again the performance is very good - the filter accurately tracks the underlying states. In figure 11.8 there is significantly less switching noise in the first 100 minutes of the simulation than in figure 11.5.

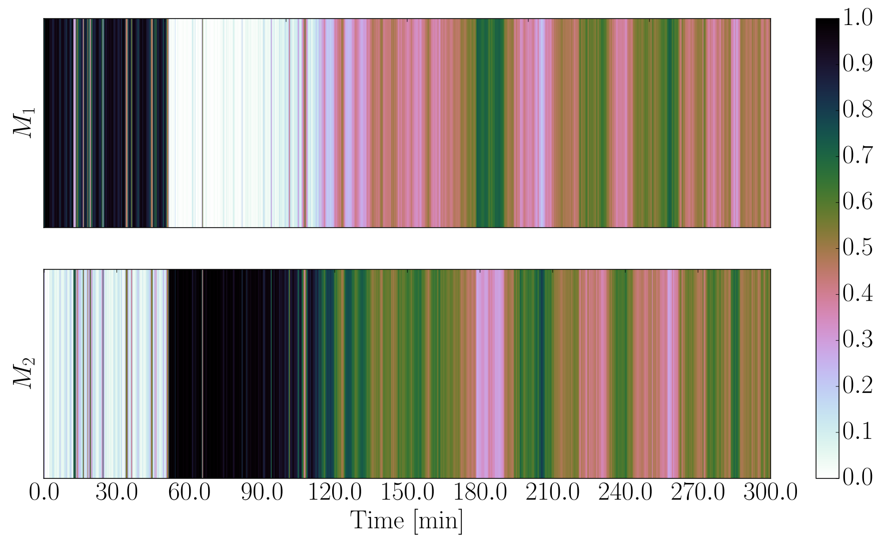

Figure 11.8: Switching particle filter measuring both concentration and temperature. The switching weight of each particle is shown per time step.

It is clear that the second measurement helps the filter differentiate between the regimes of the healthy and broken plant when the system is near the high temperature and unstable operating points. However, we see the same behaviour near the low temperature operating point - the models are clearly similar here and we again have the problem of model overlap.

In the next chapter we implement control using the switching particle filter to identify when the underlying system's dynamics have changed.

Chapter 12

Stochastic switching linear control using nonlinear hybrid models

We continue our discussion of switching controller algorithms from chapter 10 here. In this chapter we use the switching particle filter to identify the best model to use in the stochastic controllers we developed in chapter 8. More precisely, let $M_i = (A_i, B_i)$ be the linearised model corresponding to the nonlinear models (f_i, g_i) as discussed in chapter 11. By finding the most likely nonlinear model, using the switching particle filter, we aim to design a linear controller (based on the most likely nonlinear model) which is robust against system faults.

The fundamental difference between the controllers we develop in this chapter and those of chapter 10 is that the linear model used for state prediction is, if the filter behaves as intended, an accurate approximation of the underlying dynamics. In chapter 10 the controllers performed poorly because they were used to predict the system states into regions where they were not accurate. In this chapter we use the switching particle filter to identify when the underlying dynamics change. The more accurate model is then used for control; however, the crucial difference is that we do not attempt to traverse the state space as in chapter 10. We rather solve the more modest goal of keeping the system at set point in the presence of system faults.

We assume the same scenario as introduced in section 11.5 i.e. we assume we have 2 nonlinear plant models available. Model M_1 corresponds to the healthy CSTR and model M_2 corresponds to the CSTR with denatured catalyst (the faulty model). We will again avail ourselves of the switching controller algorithm repeated here for convenience.

Switching controller algorithm:

1. Use a switching filter algorithm, e.g. the switching particle filter, to update the state estimates of the particle population given the current observation. See chapter 11 for more details.

2. Select the particle with the highest switching weight. Since each particle corresponds to

a certain model we implicitly have the most probable model M_i.

3. Use the mean and covariance information encoded by this particle within the context of the stochastic controller (LQG or MPC) formulation of chapter 8. Use the most likely model, M_i from step 2, in this setting.

4. Repeat for the next observation.

A coincidental benefit of this approach is that the filter/controller combination will automatically detect the modelled fault. We do not consider the model averaging approach (in finding the best linear model to use for control) because it will not make physical sense: the plant is either healthy or broken but cannot be a mixture between the two.

Since the underlying graphical model in chapter 9 and chapter 11 is the same, we expect the filtering trends to be the same as those found in chapter 9. For the sake of illustration we exclusively use both state measurements. There is no fundamental reason why one cannot use only one state measurement except that the filter performance will be worse.

For the remainder of this chapter we assume the control goal is to keep the system at the unsteady concentration operating point of the healthy model, even in the presence of the denatured catalyst. In all the simulations the catalyst denatures at 100 minutes. This allows us to demonstrate that the switching controller is able to regulate both the healthy and faulty plant.

12.1 Unconstrained switching control

In this chapter we compare the LQG controller (discussed in sections 3.4.3 and 8.1) to the switching controller algorithm implemented within the context of the LQG controller

$$\begin{aligned}&\min_{\mathbf{u}} \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^TQx_k + u_k^TRu_k\right) + \frac{1}{2}x_N^TP_fx_N\right] \\ &\text{subject to } x_{t+1} = A_ix_t + B_iu_t + w_t.\end{aligned} \tag{12.1}$$

For the non-switching LQG controller we use a particle filter to estimate the current state. The same control parameters as those found in section 8.4 are used. Note that the current state estimate x_0 is inferred from the respective observers. Note that we select the most likely model, $M_i = (A_i, B_i)$, based on the switch weight supplied by the switching particle filter at each time step. This model is then used in (12.1) and solved using the techniques of section 8.1.

In figure 12.1 we see the performance of the LQG controller applied to the CSTR system. At 100 minutes the catalyst denatures and the model used to design the controller becomes grossly inaccurate. The inappropriateness of the model also affects the particle filter's performance.

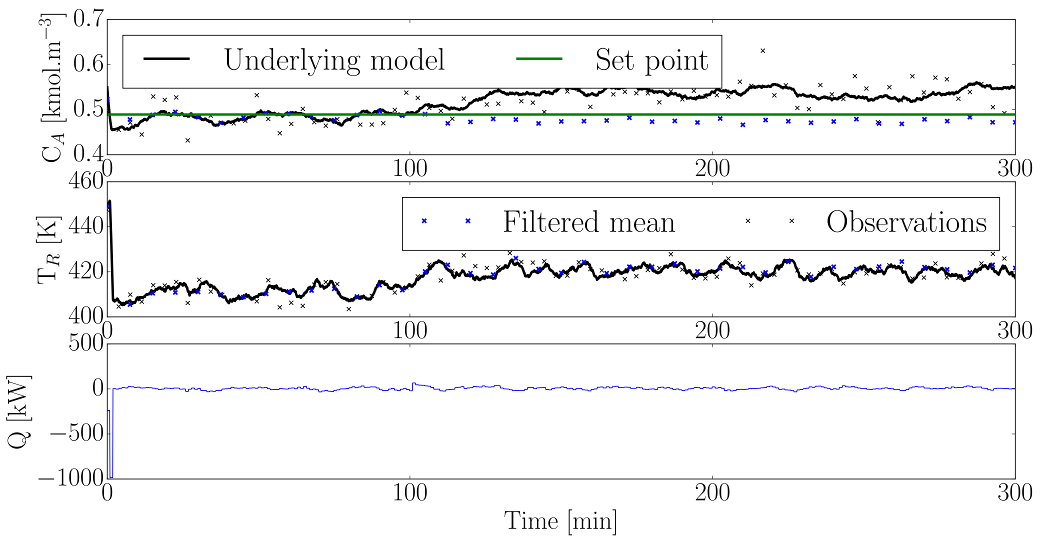

Figure 12.1: Standard LQG controller applied to the CSTR where the catalyst denatures at 100 minutes. The bootstrap particle filter was used for inference and the Gaussian approximation of the particles was used.

The average concentration error is 6.79% and the average controller input is 17.08 kW over the course of the simulation. We can clearly see that there is non-zero set point offset and control is bad in the sense of definition 8.1. Clearly the LQG controller is ineffective in this scenario.

We used a constant disturbance model to infer the plant/model mismatch[1]. This was used to accordingly adjust the controller predictions as discussed in section 2.2. It is interesting to note that the state estimator infers that the plant reaches set point but in reality there is non-zero offset. This is a consequence of using an inappropriate model in the controller/observer.

This motivates the use of a controller which intelligently changes the model control is based upon, as discussed previously. In figure 12.2 we see the set point tracking ability of the switching controller algorithm using the LQG controller. We have used the switching transition matrix $P_1 = \begin{pmatrix} 0.99 & 0.01 \\ 0.01 & 0.99 \end{pmatrix}$ as in section 11.5.

[1]The results of this chapter implement the constant disturbance model to achieve zero set point offset. To keep notation the same we do not explicitly show it in (12.2) but mention it here.

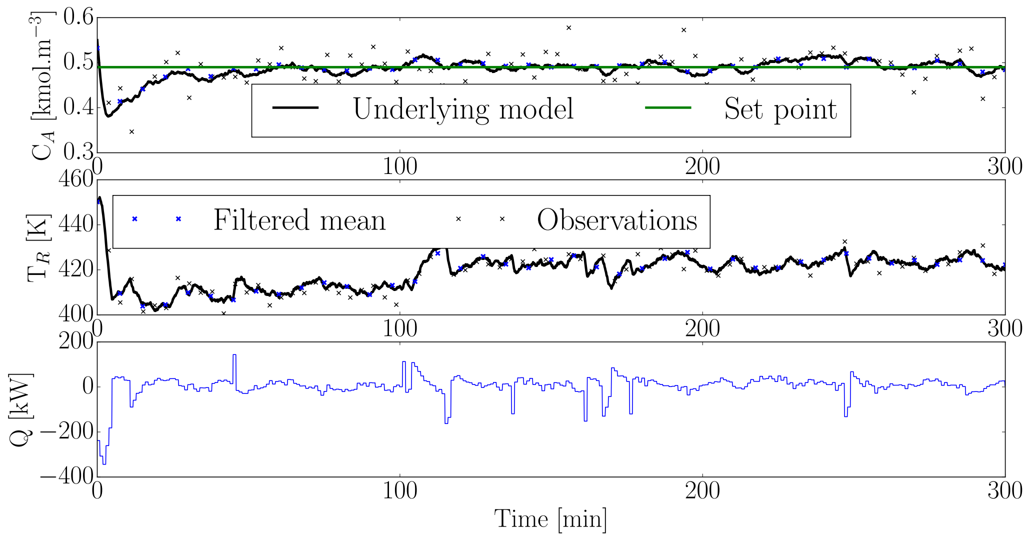

Figure 12.2: Switching LQG controller applied to the CSTR where the catalyst denatures at 100 minutes.

The average concentration error is 2.77% and the average controller input is 28.07 kW. It is clear that we have set point tracking even after the catalyst denatures. By inspecting figure 12.3 we see that this is not surprising: the filter correctly (for the most part) identifies when the underlying model changes and then uses the better model for control.

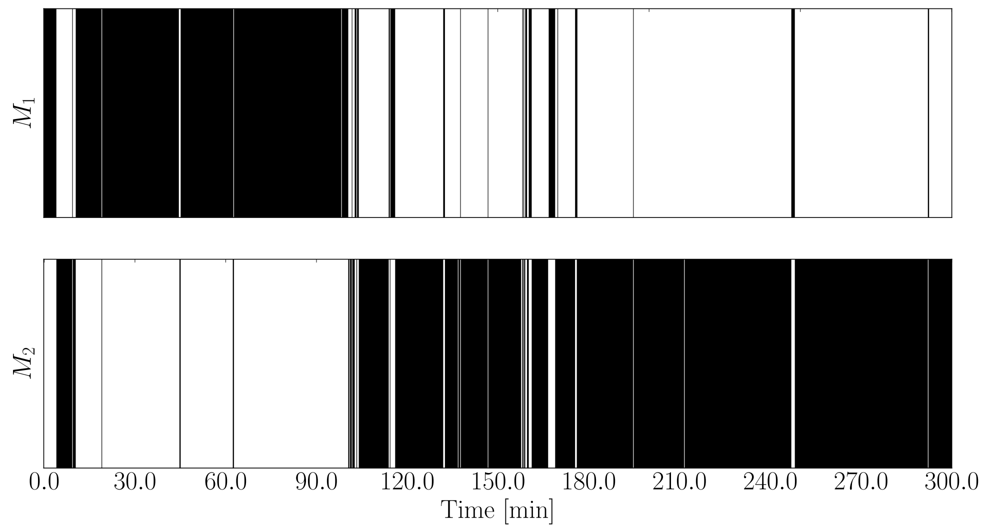

Figure 12.3: Most likely model identified using the particle filter within the context of the switching LQG controller algorithm.

However, like in sections 10.1.1 and 11.5 we see that there is some switching noise. The consequence of this noise is spikes in controller input. This happens because the controller uses the incorrect model to calculate the controller input. In chapter 9 this problem was attenuated by making the switch transition matrix stickier.

In figures 12.4 and 12.5 we use exactly the same algorithm except that we have modified the switch transition matrix: $P_2 = \begin{pmatrix} 0.999 & 0.001 \\ 0.001 & 0.999 \end{pmatrix}$. Based on figure 12.4 it is clear that we have set point tracking even after the catalyst denatures.

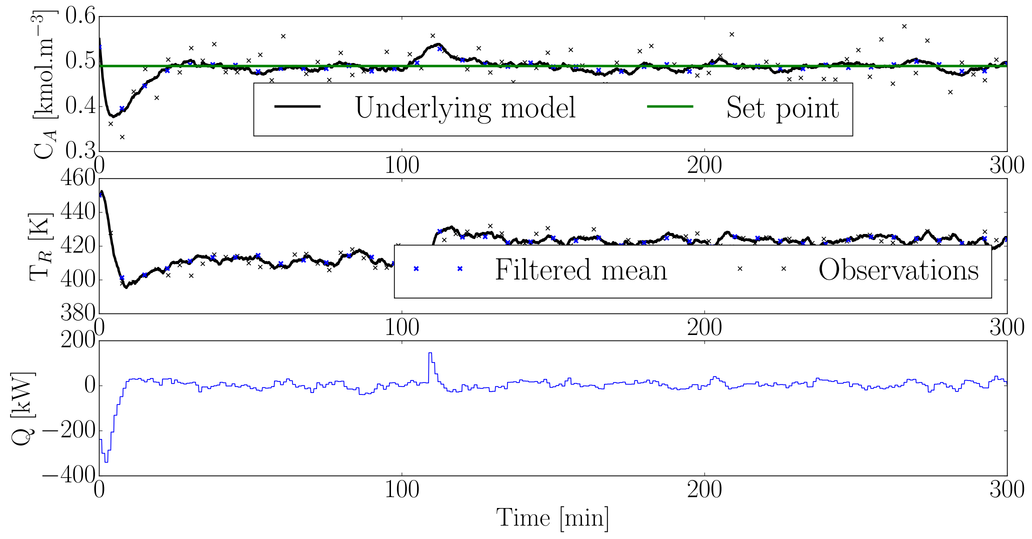

Figure 12.4: Switching LQG controller applied to the CSTR where the catalyst denatures at 100 minutes. Switch transition matrix P_2 was used.

The average concentration error is 2.38% and the average controller input is 19.06 kW. It is not surprising that the controller, using P_2 outperformed the controller using P_1. By inspecting figure 12.5 we see that there is significantly less switcing noise.

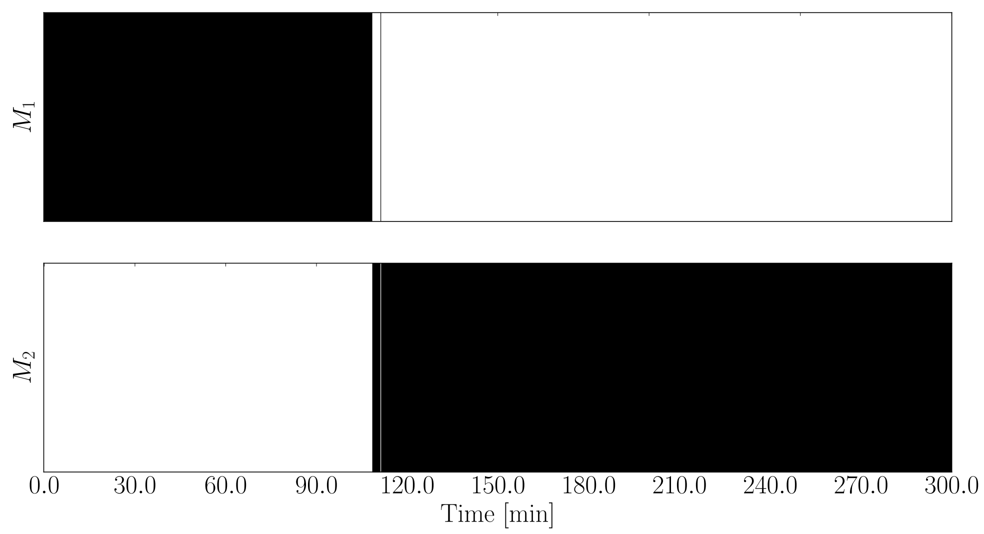

Figure 12.5: Most likely model identified using the particle filter within the context of the switching LQG controller algorithm. Switch transition matrix P_2 was used.

Since there is less switching noise there are less controller spikes in figure 12.4 and thus less controller energy is wasted. Finally, a Monte Carlo simulation was performed (using 50 runs) to illustrate that the switching controller works as desired for not just the realisations shown in this chapter. The Monte Carlo absolute average concentration error from set point was 1.30% taken over the last 10 minutes of each simulation. This indicates that the controller works as desired. We defer further Monte Carlo analysis to the next section.

The results here lend further credibility to our claim that the instability seen in chapter 10 is rooted in the inappropriateness of the model used for prediction rather than the switching noise. Figure 12.3 had significant switching noise yet the controller was not unstable. While reducing the amount of switching noise certainly improved control, as figure 12.4 shows, fundamentally we are not using an inappropriate model for control prediction. This difference is was causes the significantly better stability properties seen here.

Motivated by the success of the switching LQG controller we incorporate constraints in the sequel.

12.2 Constrained switching control

In this section we extend the switching controller algorithm of section 12.1 to the deterministic and stochastic MPCs introduced in section 8.2. We use the same parameters as before. Like in section 8.4 we first illustrate the performance of the stochastic MPC controller with expected value constraints,

$$\begin{aligned}
&\min_{\mathbf{u}} \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^TQx_k + u_k^TRu_k\right) + \frac{1}{2}x_N^TP_fx_N\right] \\
&\text{subject to } x_{t+1} = A_ix_t + B_iu_t + w_t \\
&\text{and } \mathbb{E}[\begin{pmatrix}10\\1\end{pmatrix}^T x_t + 400] \geq 0 \ \forall \ t = 1, ..., N \\
&\text{and } |u_t| \leq 250 \ \forall \ t = 0, ..., N-1,
\end{aligned} \tag{12.2}$$

(for some model M_i) and then incorporate chance constraints later. Using the results of section 8.2 we know that (12.2) can be reformulated as a deterministic problem given the (Gaussian) current state estimate x_0. The state estimate is derived from either the particle filter or switching particle filter using 200 and 500 particles respectively. For the switching particle filter we use the switch transition matrix P_2 due to the results in section 12.1.

In figure 12.6 we see the set point tracking performance of (12.2) using the same particle filter as used in section 12.1. Since the particle filter only uses the healthy plant model we only use M_1.

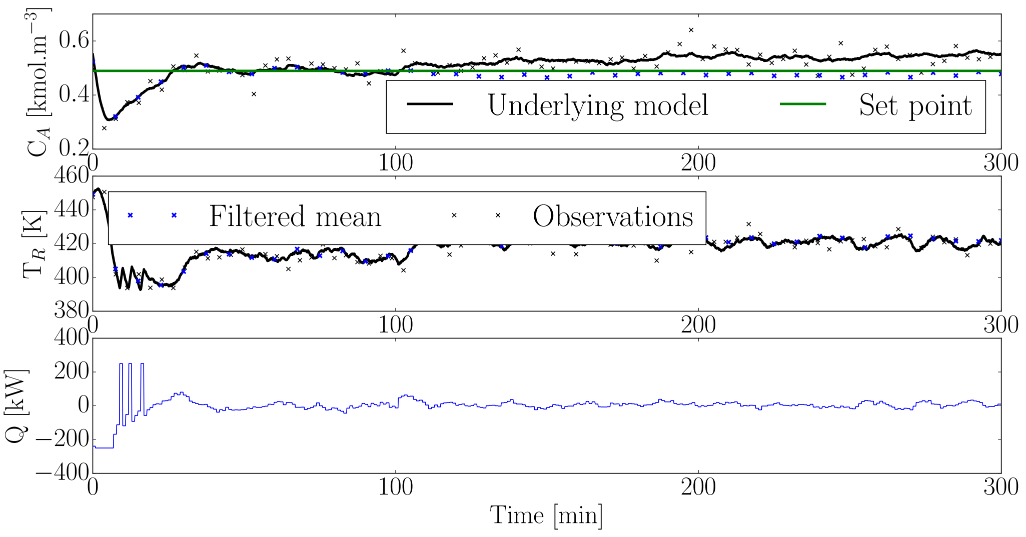

Figure 12.6: Deterministic MPC using a particle filter as the state estimator. The initial point is $(0.55, 450)$. An integrating disturbance model was used to estimate the mismatch between the underlying system and the controller model. The catalyst denatures at 100 minutes.

The average concentration error is 8.33% and the average controller input is 24.21 kW. Unfortunately we do not observe zero set point offset control but rather zero offset state estimates. Clearly the controller input generated by the MPC, which is based on the healthy plant, only drives the particle filter's predictions to the set point. We can see that the classic disturbance model approach [37] to ensure zero set point offset fails here because the underlying (faulty) model is too different from the controller model. Intuitively, we are attempting to control a tricycle (the faulty plant) using a model of a Ferrari.

In figure 12.7 we see the switching controller algorithm applied within the context of (12.2). The model corresponding to the highest weighted switch at each time step was selected for control.

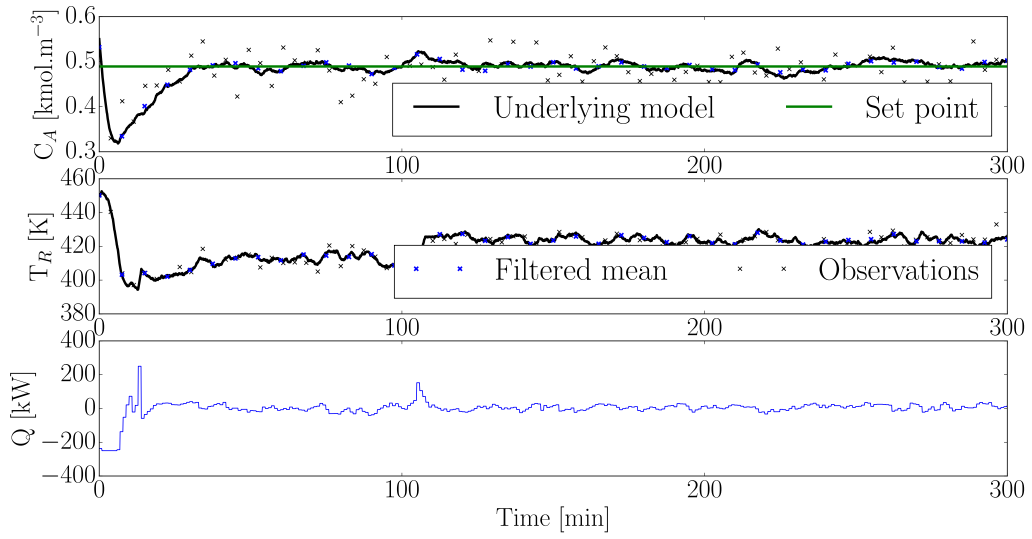

Figure 12.7: The switching MPC controller algorithm applied to the CSTR with catalyst which denatures at 100 minutes.

The average concentration error is 3.19% and the average controller input is 22.61 kW over the simulation time span. The performance of the switching controller is significantly better than the non-switching case. This is not surprising because, as figure 12.8 shows, the filter correctly identifies when the plant breaks. The tell tale controller spike is also evident after 100 minutes indicating that the controller switched.

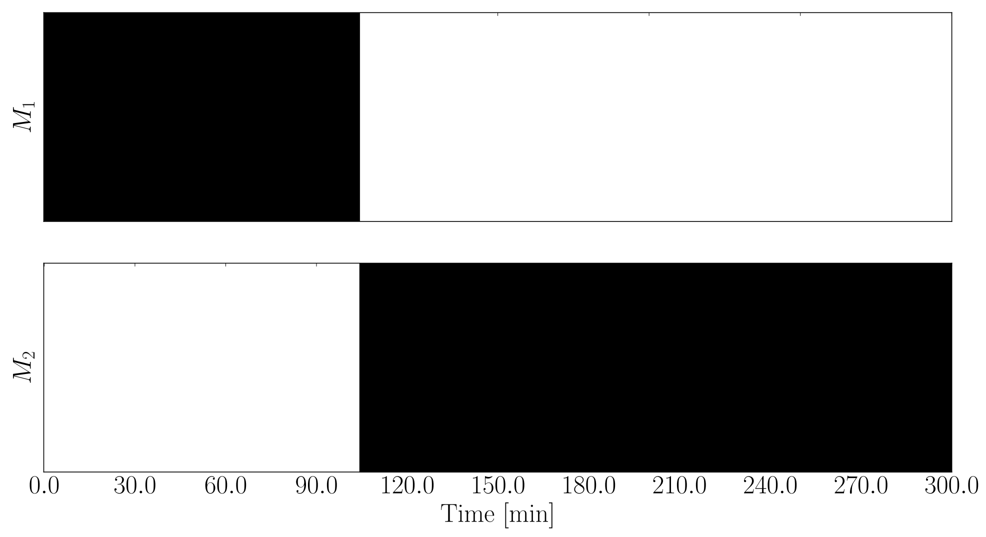

Figure 12.8: Most likely model identified using the particle filter within the context of the switching MPC controller algorithm.

There is almost no switching noise in figure 12.3 due to the static nature of the switch transition matrix P_2. If P_1 were used we would expect more noise. As mentioned in chapter 9 switching noise is not necessarily bad for inference. We do see that the filter does not immediately notice that the underlying model has changed. The stickier the switch transition matrix is the longer it will take for the controller to adjust the model. Depending on the application this delay could be problematic. On the other hand, the switching noise causes controller input spikes which could also be bad for control.

In figure 12.9 we see the state space trajectory of the expected value constraint MPC. It is clear that the operating temperature associated with the set point moves when the catalyst denatures.

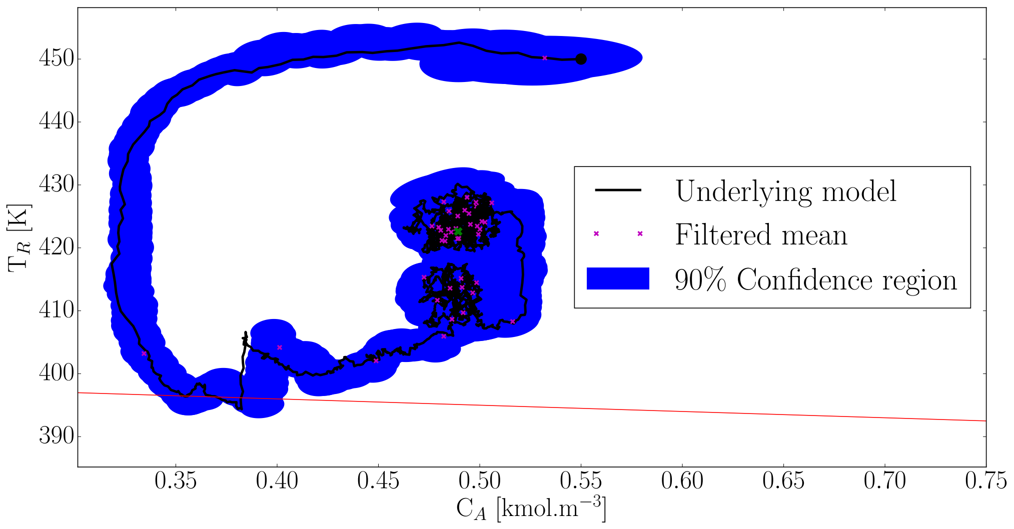

Figure 12.9: State space trajectory of the expected value constrained stochastic MPC using the switching controller algorithm.

Clearly there is a constraint violation - similar to that found in sections 8.4 and 8.5. By extending the MPC problem of (12.2) to the chance constrained problem

$$\begin{aligned}
&\min_{\mathbf{u}} \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\
&\text{subject to } x_{t+1} = A_i x_t + B_i u_t + w_t \\
&\text{and } \mathbb{E}[\begin{pmatrix}10\\1\end{pmatrix}^T x_t + 400] \geq 0\ \forall\ t = 1, ..., N \\
&\text{and } \Pr(\begin{pmatrix}10\\1\end{pmatrix}^T x_t + 400 \geq 0) \geq 0.99\ \forall\ t = 1, ..., N \\
&\text{and } |u_t| \leq 250\ \forall\ t = 0, ..., N-1
\end{aligned} \tag{12.3}$$

we attempt to ensure that the constraint is not violated in this way. The same switching controller algorithm, as discussed previously, is implemented. We have used the 99% chance constraint to highlight the effectiveness of the method compared to the expected value version.

In figure 12.10 we see that the switching chance constrained MPC successfully tracks the set point.

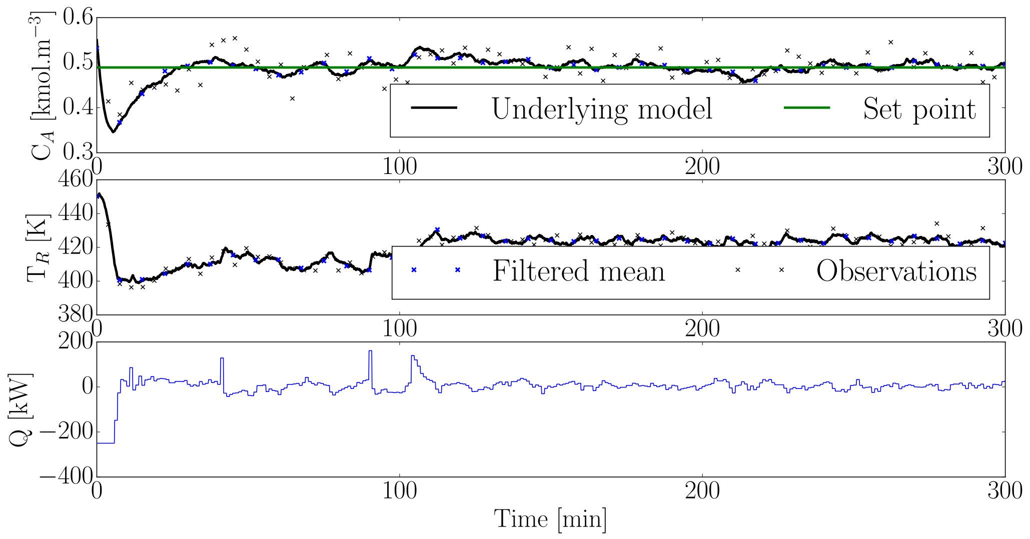

Figure 12.10: The switching MPC controller algorithm applied to the CSTR with catalyst which denatures at 100 minutes. The chance constrained MPC was used.

The average concentration error is 3.01% and the average controller input is 22.03 kW. In figure 12.11 we see the familiar model switching diagram. Clearly the controller isolates when the fault occurs and tracks the set point successfully.

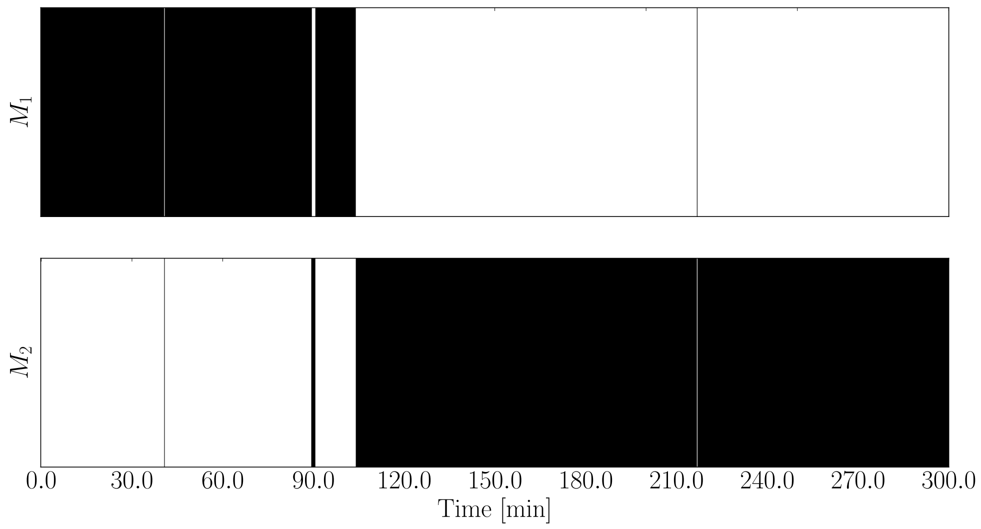

Figure 12.11: Most likely model identified using the particle filter within the context of the chance constrained switching MPC controller algorithm.

Finally, in figure 12.12 we see that the constraint is not violated.

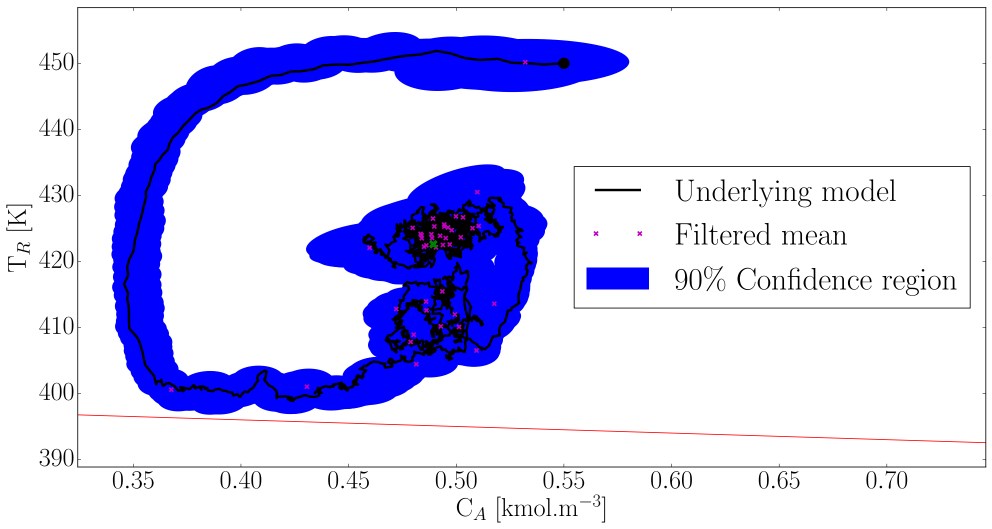

Figure 12.12: State space trajectory of the chance constrained stochastic MPC using the switching controller algorithm.

Since figure 12.12 only illustrates that the constraint is not violated for a single run we again use a Monte Carlo technique to justify the assertion that, for this example, the stochastic controller can successfully reduce the constraint violation probability. In figure 12.13 we have used the data of over 500 simulations to illustrate the benefit, in terms of constraint violation characteristics, of using the switching controller algorithm with the stochastic MPC controllers developed in chapter 8.

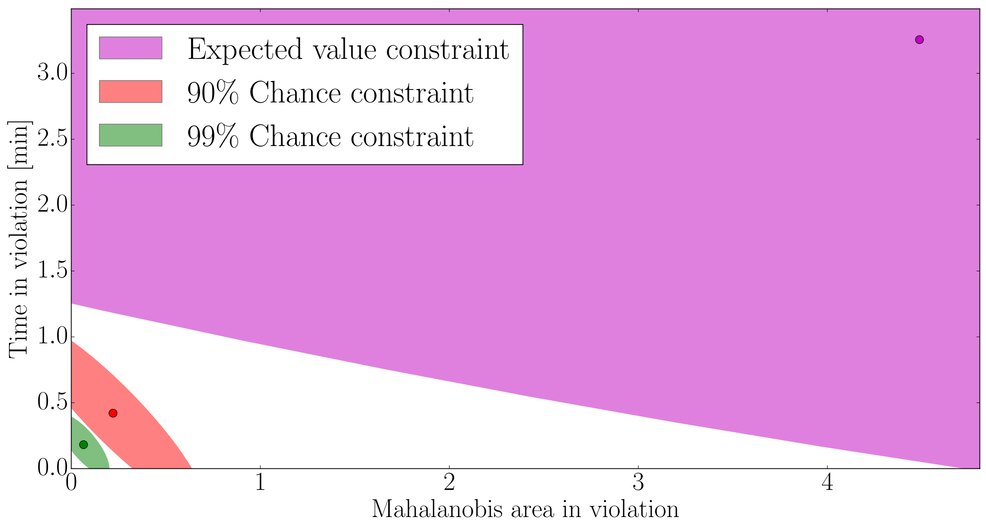

Figure 12.13: Monte Carlo analysis of the switching controller algorithm applied within the context of the expected value and chance constrained MPCs developed in chapter 8. Each shaded region is where 90% of the simulations scored.

Like in chapter 8 we see a clear benefit in increasing the chance constraint threshold: the higher the threshold the less the constraint is violated both in time and in severity. Figure 12.14 illustrates that we also have set point tracking. A Monte Carlo simulation using 50 runs each is plotted for each switching controller to illustrate their effectiveness.

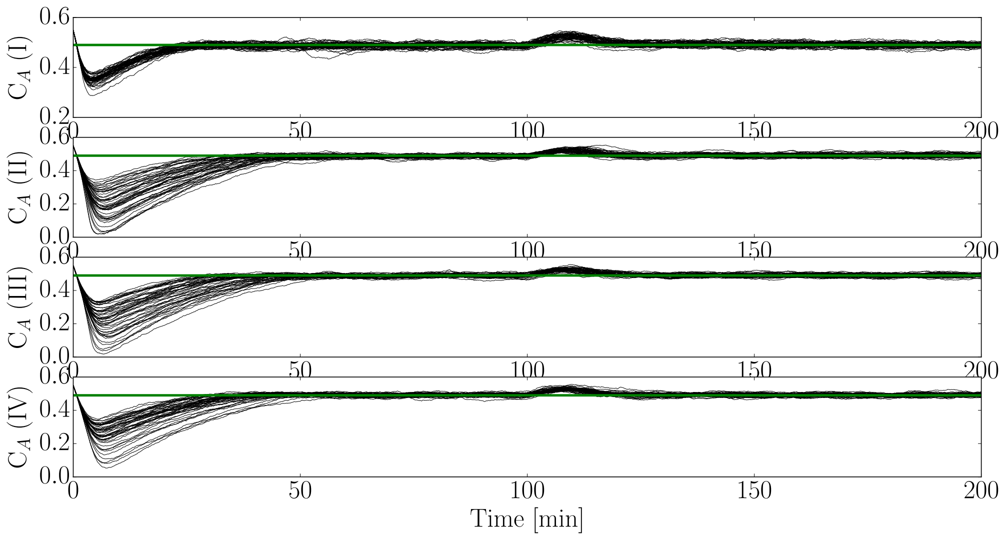

Figure 12.14: Monte Carlo set point tracking for the switching controllers. Subplot (I) shows the set point tracking of the switching LQG controller. Suplots (II), (III) and (IV) show the set point tracking of the expected value, 90% chance and 99% chance constrained switching MPC controllers. The green line is the set point.

In figure 12.14 we clearly see a concentration tracking bump after about 100 minutes - when the controller breaks. After that it is clear that the controller has switched because we again have set point tracking. The Monte Carlo absolute set point error over the last 10 minutes of each simulation is 1.30%, 1.64%, 1.47% and 1.42% for the LQG controller, the expected value-, 90% chance- and 99% chance constrained MPC controllers.

It is interesting that the same trends observed in chapter 8 are evident here. The chance constrained MPC controllers show better tracking error than the expected value controller. Again the aggressiveness of the expected value controller causes the higher error. In all cases we have set point tracking (a small δ as per definition 8.1) despite the plant fault.

12.3 Conclusion

In this chapter we have developed a switching controller algorithm which detects when a process fault has occurred by using noisy measurement data in conjunction with a switching particle filter. The controller uses this knowledge to adapt the model used for control. The switching particle filter successfully identified when the modelled fault occurred in all the simulations.

The LQG controller, expected value- and chance constrained MPC controllers were all implemented within the switching controller algorithm framework. It was found that they were able to stabilise and control the system to set point despite the plant fault. This is was confirmed using Monte Carlo analysis. Furthermore, the scheme is computationally efficient because the controllers can be formulated as deterministic, linearly constrained quadratic programming optimisation problems.

The bottleneck in the switching controller algorithm is inference. Since it is essentially a particle filter it is reasonable to suppose that there might be problems in higher dimensional spaces. However, recent work [16] on particle filters looks extremely promising in reducing the computational problems associated with filters of this kind.

Chapter 13

Future work and conclusion

In this chapter we briefly comment on extensions and future research possibilities within the context of stochastic dynamical control using probabilistic graphical models. Finally we conclude the dissertation with a brief summary of the major results.

13.1 Parameter optimisation

In chapters 8, 10 and 12 the control tuning parameters were not changed. It could be argued that the control comparisons (LQG vs. expected value MPC vs. chance constrained MPC) were unfair because the set of tuning parameters might have favoured one of them disproportionately. It should be investigated what effect parameter optimisation has on the results. It could also be interesting to view the tuning parameters as so-called hyper-parameters within the context of Bayesian optimisation.

13.2 Generalised graphical models

The motivation for this dissertation was the work by [18] and [14]. In the former paper the author pioneers work on using dynamic Bayesian networks within the context of particle predictive MPC. The authors use graphical models similar to those of chapters 6 and 7 but do not assume that the inputs are deterministic. It would be interesting to extend the current approach to this case. In the latter paper the authors use contextual variables in their graphical models to identify anomalies within the context of maritime piracy. The graphical model employed by them is significantly more complex but allows for much greater expressivity. Within the context of fault detecting and controller model modification this could be quite interesting as well.

Furthermore, while chapter 12 only dealt with a single modelled process fault there is no fundamental reason why more process faults cannot be incorporated. However, the model overlap problem might become more pronounced in such a setting. It is not physically realistic to use the augmented switching filter model (see chapter 10) because the current state would not reliably inform the filter of a process fault. The aforementioned contextual variables could become useful in such a setting.

13.3 Filtering techniques

The particle filters used throughout this dissertation can become problematic when applied to high dimensional problems. Recent work in [16] pioneers using a particle flow technique which, by all accounts, greatly reduces the computational burden particle methods usually introduce. The degree to which the inference techniques bottleneck the control systems should be investigated and an effort should be made to incorporate particle flow techniques.

Furthermore, there is no fundamental reason why the switching particle filter could not be replaced by the Rao-Blackwellised particle filter in chapter 12. This possibility should be investigated because the latter filter, due to its partly analytic nature, is computationally more efficient than the former. The obvious drawback of such an approach is that the linear model used to detect the process fault might not be as sensitive as the nonlinear model currently used.

13.4 Conclusion

The primary goal of this dissertation was to illustrate the relationship between dynamic Bayesian networks and model based predictive control. The two most important results were:

1. Assuming linearity and normality it is possible to reformulate the stochastic, chance constrained MPC problem as a linearly constrained deterministic MPC problem. The holistic approach adopted here, due to the probabilistic graphical model framework, makes the proof and derivation both simple and intuitive.

2. By extending the graphical model we were able to formulate a novel switching controller algorithm by combining the computationally efficient deterministic controllers developed earlier with a switching particle filter. Using this approach the controller was able to identify and adapt to a modelled process fault.

Lastly, it is important to realise that graphical models have tacitly been used within control schemes since state observers were introduced. Therefore, making the connection between them overt was not a purely academic endeavour but an attempt at highlighting the potential benefit of understanding stochastic model predictive controllers within the context of graphical models.

Bibliography

[1] MOSEK ApS. *MOSEK solver API using the Julia language package JuMP.*, 2015.

[2] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation.* John Wiley and Sons, 2001.

[3] D. Barber. Expectation correction for smoothed inference in switching linear dynamical systems. *Journal of Machine Learning*, 7:2515–2540, 2006.

[4] D. Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.

[5] I. Batina, A.A. Stoorvogel, and S. Weiland. Optimal control of linear, stochastic systems with state and input constraints. In *Proceedings of the 41st IEEE Conference on Decision and Control*, 2002.

[6] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatics*, 35:407–427, 1999.

[7] Jeff Bezanson, Stefan Karpinski, Viral B. Shah, and Alan Edelman. Julia: A fast dynamic language for technical computing. September 2012.

[8] C.M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[9] L. Blackmore, Hui Li, and B. Williams. A probabilistic approach to optimal robust path planning with obstacles. In *American Control Conference*, June 2006.

[10] L. Blackmore, O. Masahiro, A. Bektassov, and B.C. Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE Transactions on Robotics*, 26, 2010.

[11] M. Cannon, B. Kouvaritakis, and X. Wu. Probabilistic constrained mpc for multiplicative and additive stochastic uncertainty. *IEEE Transactions on Automatic Control*, 54(7), 2009.

[12] A.L. Cervantes, O.E. Agamennoni, and J.L Figueroa. A nonlinear model predictive control system based on weiner piecewise linear models. *Journal of Process Control*, 13:655–666, 2003.

[13] R. Chen and J.S. Liu. Mixture kalman filters. *Journal of Royal Statistical Society*, 62(3):493–508, 2000.

[14] J.J. Dabrowski and J.P. de Villiers. A method for classification and context based behavioural modelling of dynamical systems applied to maritime piracy. *Expert Systems with Applications*, 2014.

[15] B.N. Datta. *Numerical Methods for Linear Control Systems - Design and Analysis*. Elsevier, 2004.

[16] F. Daum and J. Huang. Particle flow for nonlinear filters. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5920–5923, May 2011.

[17] M. Davidian. Applied longitudinal data analysis. North Carolina State University, 2005.

[18] J.P. de Villiers, S.J. Godsill, and S.S. Singh. Particle predictive control. *Journal of Statistical Planning and Inference*, 141:1753–1763, 2001.

[19] N. Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, 1974.

[20] M. Diehl, H.J. Ferreau, and N. Haverbeke. Efficient numerical methods for nonlinear mpc and moving horizon estimation. *Control and Information Sciences*, 384:391–417, 2009.

[21] A. Doucet and A.M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. Technical report, The Institute of Statistical Mathematics, 2008.

[22] A.D. Doucet, N.J. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624, March 2001.

[23] J. Du, C. Song, and P. Li. Modeling and control of a continuous stirred tank reactor based on a mixed logical dynamical model. *Chinese Journal of Chemical Engineering*, 15(4):533–538, 2007.

[24] The Economist. In praise of bayes. Article in Magazine, September 2000.

[25] C. Edwards, S.K. Spurgeon, and R.J. Patton. Sliding mode observers for fault detection and isolation. *Automatica*, 36:541–553, 200.

[26] H.C. Edwards and D.E. Penny. *Elementary Differential Equations*. Pearson, 6th edition edition, 2009.

[27] W. Forst and D. Hoffmann. *Optimisation - Theory and Practice*. Springer, 2010.

[28] O.R. Gonzalez and A.G. Kelkar. *Electrical Engineering Handbook*. Academic Press, 2005.

[29] N.J. Gordon, D.J. Salmond, and A.F.M. Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings-F*, 140(2):107–113, 1993.

[30] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[31] R. Isermann and P. Balle. Trends in the application of model based fault detection and diagnosis of technical processes. *Control Engineering Practice*, 5(5):709–719, 1997.

[32] K. Ito and K. Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 45(5):910–928, 2000.

[33] R. J. Jang and C.T. Sun. *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. Prentice-Hall, 1996.

[34] D. Koller and N. Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

[35] K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence*. Series in Computer Science and Data Analysis. Chapman & Hall, first edition edition, 2004.

[36] M. Kvasnica, M. Herceg, L. Cirka, and M. Fikar. Model predictive control of a cstr: a hybrid modeling approach. *Chemical Papers*, 64(3):301–309, 2010.

[37] J.H. Lee, M. Morari, and C.E. Garcia. *Model Predictive Control*. Prentice Hall, 2004.

[38] U.N. Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems*. PhD thesis, Stanford Univesity, 2002.

[39] P. Li, M. Wendt, H. Arellano-Garcia, and G. Wozny. Optimal operation of distrillation processes under uncertain inflows accumulated in a feed tank. *American Institute of Chemical Engineers*, 2002.

[40] P. Li, M. Wendt, and G. Wozny. A probabilistically constrained model predictive controller. *Automatica*, 38:1171–1176, 2002.

[41] Miles Lubin and Iain Dunning. Computing in operations research using julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

[42] W.L. Luyben. *Process Modeling, Simulation and Control for Chemical Engineers*. McGraw-Hill, 2nd edition edition, 1990.

[43] J.M. Maciejowski. *Predictive Control with constraints*. Prentice-Hall, 2002.

[44] O. Masahiro. Joint chance-constrained model predictive control with probabilistic resolvability. *American Control Conference*, 2012.

[45] P. Mhaskar, N.H. El-Farra, and P.D. Christofides. Stabilization of nonlinear systems with state and control constraints using lyapunov-based predictive control. *Systems and Control Letters*, 55:650–659, 2006.

[46] K.P. Murphy. Switching kalman filters. Technical report, Compaq Cambridge Research Lab, 1998.

[47] K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* PhD thesis, University of California, Berkeley, 2002.

[48] K.P. Murphy. *Machine Learning: A Probabilistic Perspective.* MIT Press, 2012.

[49] N. Nandola and S. Bhartiya. A multiple model approach for predictive control of nonlinear hybrid systems. *Journal of Process Control*, 18(2):131–148, 2008.

[50] L. Ozkan, M. V. Kothare, and C. Georgakis. Model predictive control of nonlinear systems using piecewise linear models. *Computers and Chemical Engineering*, 24:793–799, 2000.

[51] T. Pan, S. Li, and W.J. Cai. Lazy learning based online identification and adaptive pid control: a case study for cstr process. *Industrial Engineering Chemical Research*, 46:472–480, 2007.

[52] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control.* Nob Hill Publishing, 2009.

[53] B. Reiser. Confidence intervals for the mahalanobis distance. *Communications in Statistics: Simulation and Computation*, 30(1):37–45, 2001.

[54] Y. Sakakura, M. Noda, H. Nishitani, Y. Yamashita, M. Yoshida, and S. Matsumoto. Application of a hybrid control approach to highly nonlinear chemical processes. *Computer Aided Chemical Engineering*, 21:1515–1520, 2006.

[55] A.T. Schwarm and Nikolaou. Chance constrained model predictive control. Technical report, University of Houston and Texas A&M University, 1999.

[56] C. Snyder, T. Bengtsson, P. Bickel, and J. Anderson. Obstacles to high-dimensional particle filtering. *Mathematical Advances in Data Assimilation*, 2008.

[57] S.J. Streicher, S.E. Wilken, and C. Sandrock. Eigenvector analysis for the ranking of control loop importance. *Computer Aided Chemical Engineering*, 33:835–840, 2014.

[58] D.H. van Hessem and O.H. Bosgra. Closed-loop stochastic dynamic process optimisation under input and state constraints. In *Proceedings of the American Control Conference*, 2002.

[59] D.H. van Hessem, C.W. Scherer, and O.H. Bosgra. Lmi-based closed-loop economic optimisation of stochastic process operation under state and input constraints. In *Proceedings of the 40th IEEE Conference on Decision and Control*, 2001.

[60] H. Veeraraghavan, P. Schrater, and N. Papanikolopoulos. Switching kalman filter based approach for tracking and event detection at traffic intersections. *Intelligent Control*, 2005.

[61] A. Wachter and L.T. Biegler. On the implementation of a prima-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–27, 2006.

[62] D. Wang, W. Wang, and P. Shi. Robust fault detection for switched linear systems with state delays. *Systems, Man and Cybernetics*, 39(3):800–805, 2009.

[63] R.S. Wills. Google's pagerank: the math behind the search engine. Technical report, North Carolina State University, 2006.

[64] J. Yan and R.R. Bitmead. Model predictive control and state estimation: a network example. In *15th Triennial World Conference of IFAC*, 2002.

[65] J. Yan and R.R. Bitmead. Incorporating state estimation into model predictive control and its application to network traffic control. *Automatica*, 41:595–604, 2005.

[66] M.B. Yazdi and M.R. Jahed-Motlagh. Stabilization of a cstr with two arbitrarily switching modes using model state feedback linearisation. *Chemical Engineering Journal*, 155(3):838–843, 2009.