University of Pretoria

Department of Chemical Engineering

Masters Dissertation

Stochastic Dynamical Control using Graphical Models

Author:
St. Elmo Wilken

Student Number:
29034133

Co-Supervisor:
Mr. C Sandrock

Department:
Chemical Engineering

Co-Supervisor:
Dr. JP de Villiers

Department:
Electrical, Electronic and
Computer Engineering

June 1, 2015

Contents

1 Introduction

To do.

2 Background Literature

This section is composed of five subsections which introduce the main concepts and results used throughout the rest of the document. Section 2.1 introduces Probability Theory and Section 2.2 very briefly introduces some useful nomenclature from Graph Theory. These two sections serve as an entry point for Section 2.3 which deals with Probabilistic Graphical Models. Section 2.4 deals with Control Theory and Section 2.5 introduces an important result from matrix linear algebra.

It would seem as if Sections 2.1 to 2.3 and Section 2.4 are not related to each other. However, the foundational theory introduced here is expanded upon later and the relationship then becomes clear.

2.1 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.

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

Axiom 2.2. $\forall \alpha \in \Omega,\ P(\alpha) \geq 0$. The probability of any one (or set of) event(s) in Ω occurring is non-negative.

Axiom 2.3. $\forall \alpha, \beta \in \Omega,$ if $\alpha \cap \beta = \emptyset$ 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 we are able to extend the theory to Theorem 2.1.

Theorem 2.1. $\forall \alpha, \beta \in \Omega, P(\alpha \cup \beta) = P(\alpha) + P(\beta) + P(\alpha \cap \beta)$. The probability of two events occurring in Ω is equal to the sum of their probabilities less the probability of both occurring simultaneously.

2.1.1 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 [24].

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 2.2.

Theorem 2.2. 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 2.1 makes precise how the knowledge that event y has occurred alters our view of event x occurring.

Definition 2.1. 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 2.1 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 2.2.

Definition 2.2. 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 2.3.

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 [24].

Definition 2.3. Independence $X \perp\!\!\!\perp Y \equiv P(X|Y) = P(X)$

Generalising the concept of independence we define conditional independence by Definition 2.4. Again this definition is symmetric [24].

Definition 2.4. 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 2.3 which naturally leads us to the formulation of Bayes' Theorem (using Definition 2.1) as shown in Theorem 2.4.

Theorem 2.3. 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 2.4. Bayes' Theorem $P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$

Under the Bayesian interpretation of Theorem 2.4 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.

Bayes' Theorem: Example

This section will attempt to develop some intuition behind Theorem 2.4. We quote an excerpt from an article in the Economist [17] and illustrate the use of Bayes' Rule in a canonical medical example [25].

"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."

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.

2.1.2 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 $\int p(x)dx = 1$, where we have implicitly integrated over the domain of p, then it can be used to generate one. The cumulative probability function $P(X \leq a) = \int_{-\infty}^{a} p(x)dx$ 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 [5]. 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 non-vector valued variables.

Definition 2.5. Gaussian Distribution The univariate Gaussian or Normal distribution of a random variable x is shown in (1). We call μ the mean and σ^2 the variance of the distribution.

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

The multivariate Gaussian distribution is shown in (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 $\Lambda \equiv \Sigma^{-1}$.

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

[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 $P(x \in [a, b]) = \int_a^b p(y)dy$; thus, if we want the probability of x occurring we could just make $[a, b]$ small.

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

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 2.6, the variance in Definition 2.7 and the covariance in Definition 2.8.

Definition 2.6. 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 2.7. 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 formulate $\text{var}[f] = \mathbb{E}[f^2] - \mathbb{E}[f]^2$. Also note that for a univariate Gaussian random variable x we have $\text{var}[x] = \sigma^2$.

Definition 2.8. Covariance For two random variables x, y (which may be 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 2.5.

The identities in Theorem 2.5 will be useful in later sections. We refer the reader to [11] for justification.

Theorem 2.5. 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. $\mathbb{E}[X^T CX] = \text{tr}(C\Sigma) + \mu^T C\mu$

Now we are in a position to perform some manipulations assuming we are using Gaussian random variables. We state without proof Theorem 2.6.

Theorem 2.6. 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 in (3) and the marginal distribution in (4).

$$\begin{aligned} &p(x_a|x_b) = \mathcal{N}(x_a|\mu_{a|b}, \Lambda_{aa}^{-1}) \\ &\text{with } \mu_{a|b} = \mu_a - \Lambda_{aa}^{-1}\Lambda_{ab}(x_b - \mu_b) \end{aligned} \tag{3}$$

$$p(x_a) = \mathcal{N}(x_a|\mu_a, \Sigma_{aa}) \tag{4}$$

Note that for the conditional distribution it is easier to work with the precision matrix.

Next we state and then prove Theorem 2.7 which we will use extensively. The proof for Theorem 2.6 uses the same techniques and can be found in [5].

Theorem 2.7. Bayes' Theorem for Linear Gaussian Models Suppose we have a marginal Gaussian distribution for x and a conditional Gaussian distribution for y in the form shown in (5).

$$\begin{aligned} p(x) &= \mathcal{N}(x|\mu, \Lambda^{-1}) \\ p(y|x) &= \mathcal{N}(y|Ax + b, L^{-1}) \end{aligned} \tag{5}$$

Then the marginal distribution for y is given by (6) and the conditional distribution for x given y is (7).

$$p(y) = \mathcal{N}(y|A\mu + b, L^{-1} + A\Lambda^{-1}A^T) \tag{6}$$

$$\begin{aligned} &p(x|y) = \mathcal{N}(x|\Sigma(A^T L(y-b) + \Lambda\mu), \Sigma) \\ &\text{with } \Sigma = (\Lambda + A^T LA)^{-1} \end{aligned} \tag{7}$$

Note that b is a known vector. Given a deterministic function u we can also write $b = Bu + e$ where B is some conforming matrix and e is some constant vector.

Proof. We begin our proof by noticing that for a general Gaussian $\mathcal{N}(\gamma|\alpha, \beta)$ we can write the exponent as in (8), note *const* is some real number which does not depend on γ.

$$-\frac{1}{2}(\gamma - \alpha)^T \beta^{-1} (\gamma - \alpha) = -\frac{1}{2}\gamma^T \beta^{-1}\gamma + \gamma^T \beta^{-1}\alpha + const \tag{8}$$

Also note 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). To find the joint distribution we let $z = \begin{pmatrix} x \\ y \end{pmatrix}$ and consider the log of the joint in (9).

$$\begin{aligned} \log(z) &= \log(p(x)) + \log(p(y|x)) \\ &= -\frac{1}{2}(x-\mu)^T \Lambda (x-\mu) - \frac{1}{2}(y - Ax - b)^T L(y - Ax - b) + const \end{aligned} \tag{9}$$

Here *const* denotes constant terms which are independent of x and y. Now we make use of (8) to find the mean and covariance of z. Continuing, we consider only the second order terms when (9) is expanded, as shown in (10).

$$\begin{aligned} &-\frac{1}{2}x^T(\Lambda + A^T LA)x - \frac{1}{2}y^T Ly + \frac{1}{2}y^T LAx + \frac{1}{2}x^T A^T Ly \\ &= -\frac{1}{2}\begin{pmatrix} x \\ y \end{pmatrix}^T \begin{pmatrix} \Lambda + A^T LA & -A^T L \\ -LA & L \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\ &= -\frac{1}{2}z^T Rz \end{aligned} \tag{10}$$

From this we immediately have the precision of z: the matrix R; we also use a matrix inversion formula found in [5] to find the covariance. This is shown in (11).

$$\text{cov}[z] = R^{-1} = \begin{pmatrix} \Lambda^{-1} & \Lambda^{-1}A^T \\ A\Lambda^{-1} & L^{-1} + A\Lambda^{-1}A^T \end{pmatrix} \tag{11}$$

We now proceed in exactly the same way to find mean of z. By expanding 9 and only considering the first order terms in x and y we have (12).

$$x^T\Lambda\mu - x^TA^TLb + y^TLb = \begin{pmatrix} x \\ y \end{pmatrix} \begin{pmatrix} \Lambda\mu - A^TLb \\ Lb \end{pmatrix} \tag{12}$$

Again, by making use of (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 [5]. By using Theorem 2.6 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 2.9.

Definition 2.9. 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 [5].

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 to measure the dissimilarity between a known distribution and a sampled approximation thereof. See [5] for the motivation.

Theorem 2.8. 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 2.10.

Definition 2.10. 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 $D_M(x|y, S) = \sqrt{(x-y)^TS^{-1}(x-y)}$.

The Mahalanobis Distance is a 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.

2.2 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. $X_i \to X_j$ 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 2.11. Directed Path We say that the nodes $X_1, X_2, X_3, ..., X_n \in \chi$ form a directed path if $X_i \to X_{i+1}$ for $1 \leq i \leq n-1$.

Definition 2.12. Directed Cycle A directed cycle is a non-singleton directed path which starts and ends at the same node.

Definition 2.13. 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 1 is an example of a DAG.

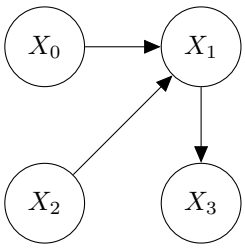

Figure 1: Example of a Directed Acyclic Graph

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

Definition 2.14. 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 2.15. 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 2.16. 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 2.17. 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 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 [13].

2.3 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.

2.3.1 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 2.18. 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 2.19. 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 [24]. 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 2.9.

Theorem 2.9. 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 2 models the system.

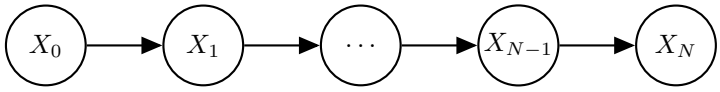

Figure 2: Example simple Bayesian Network

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 2.9: 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 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 2.20. 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 2.21. Blocked Path A path is blocked, given a set of nodes **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 2.22. 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. 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 [25].

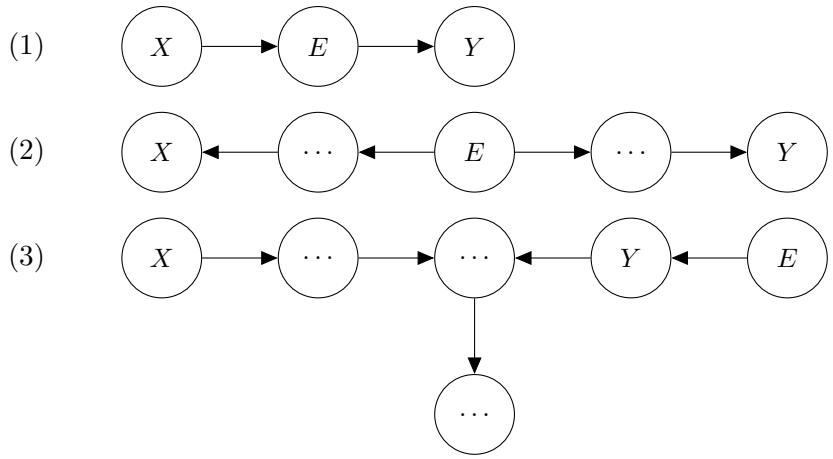

Figure 3: Examples of d-separation

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.

2.3.2 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 [35]. 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 2.23. $\mathbf{N^{th}}$-order Markov assumption A system satisfies the N^{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^{th}-order Markov system into a first order Markov system by modifying the state space [35]. 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. This Bayesian Network graph may be factorised according to the Bayesian Network chain rule such that at each time slice:

$$P(z_t|z_{t-1}) = \Pi_{i=1}^{N} P(z_t^i|\text{Parents}(z_t^i)) \tag{13}$$

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:

$$P(z_{0:T}) = \Pi_{t=1}^{T} \Pi_{i=1}^{N} P(z_t^i|\text{Parents}(z_t^i)) \tag{14}$$

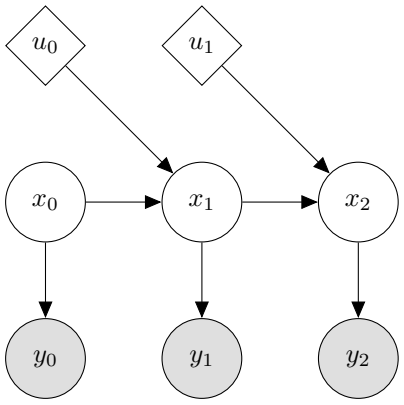

Figure 4: Example of a DBN unrolled for 3 time slices

2.4 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.

2.4.1 Linear Quadratic Regulator Control

We start our analysis by assuming we have an accurate, linear, discrete, time invariant state space representation of a system shown in (15).

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ y_t &= Cx_t \end{aligned} \tag{15}$$

We also assume that the states are measured i.e. $C = I$ and 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 (15).

Definition 2.24. Linear Quadratic Regulator (LQR) Objective Function The controller minimising the quadratic objective function, shown in (16), is called the LQR controller.

$$V(x_0, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N \tag{16}$$

The optimisation of (16) is implicitly subject to the state dynamics. The matrices Q and R 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 2.10 because it will be useful later. The proof may be found in [38] and follows from algebraic manipulations.

Theorem 2.10. 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 in (17) and ana-

lytically solve it using backward Dynamic Programming.

$$\begin{aligned} &\min_{\mathbf{u}} V(x_0, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t \end{aligned} \tag{17}$$

Expanding the objective function to examine its structure we have (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.

$$\begin{aligned} \min_{\mathbf{u}} V(x_0, \mathbf{u}) &= \min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^t R u_k\right) + \frac{1}{2}x_N^T P_f x_N \\ &= \min_{u_0, u_1, \ldots, u_{N-1}} \frac{1}{2}\left(x_0^T Q x_0 + u_0^T R u_0 + x_1^T Q x_1 + u_1^T R u_1 + \ldots + x_N^T P_f x_N\right) \\ &= \min_{u_0, u_1, \ldots, u_{N-2}} \frac{1}{2}\left(x_0^T Q x_0 + u_0^T R u_0 + \ldots + x_{N-2}^T Q x_{N-2} + u_{N-2}^T R u_{N-2}\right) \ldots \\ &\quad + \min_{u_{N-1}} \frac{1}{2}\left(x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + x_N^T P_f x_N\right) \end{aligned} \tag{18}$$

By using Theorem 2.10 and the constraint $x_N = Ax_{N-1} + Bu_{N-1}$ we can simplify the last term in the separated minimisation problem of (16) as shown in (19). This is the first step of backward Dynamic Programming used to solve the problem.

$$\begin{aligned} \min_{u_{N-1}} V_{N-1}(x_N, u_{N-1}) &= \min_{u_{N-1}} \frac{1}{2}\left(x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} + x_N^T P_f x_N\right) \\ &= \min_{u_{N-1}} \frac{1}{2}\left(x_{N-1}^T Q x_{N-1} + (u_{N-1} - v)^T H (u_{N-1} - v)\right) + d \\ &\text{with } H = R + B^T P_f B \\ &\text{and } v = K_{N-1} x_{N-1} \\ &\text{and } d = \frac{1}{2} x_{N-1}^T \left(K_{N-1}^T R K_{N-1} + (A + BK_{N-1})^T P_f (A + BK_{N-1})\right) x_{N-1} \\ &\text{and } K_{N-1} = -(B^T P_f B + R)^{-1} B^T P_f A \end{aligned} \tag{19}$$

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. We summarise the optimal stage cost and controller action, after some simplifications, in (20).

$$\begin{aligned} u_{N-1}^0(x) &= K_{N-1} x \\ x_{N-1}^0(x) &= (A + BK_{N-1})x \\ V_{N-1}^0(x) &= \frac{1}{2} x^T \Pi_{N-1} x \\ K_{N-1} &= -(B^T P_f B + R)^{-1} B^T P_f A \\ \Pi_{N-1} &= Q + A^T P_f A - A^T P_f B (B^T P_F B + R)^{-1} B^T P_f A \end{aligned} \tag{20}$$

The function $V_{N-1}^0(x)$ defines the optimal cost to go from state x for the last stage under the optimal control law $u_{N-1}^0(x)$. Now we proceed with the backward Dynamic Programming

and solve (21).

$$\min_{u_{N-2}} \frac{1}{2}\left(x_{N-2}^T Q x_{N-2} + u_{N-2}^T R u_{N-2}\right) + V_{N-1}^0(x_{N-1}) \tag{21}$$

But now we note the similarity between (19) and (21). Using $x_{N-1} = Ax_{N-2} + Bu_{N-2}$ and the same procedure as before we have (22).

$$\begin{aligned}
&u_{N-2}^0(x) = K_{N-2}x \\
&x_{N-2}^0(x) = (A + BK_{N-2})x \\
&V_{N-2}^0(x) = \frac{1}{2}x^T \Pi_{N-2} x \\
&K_{N-2} = -(B^T \Pi_{N-1} B + R)^{-1} B^T \Pi_{N-1} A \\
&\Pi_{N-2} = Q + A^T \Pi_{N-1} A - A^T \Pi_{N-1} B (B^T \Pi_{N-1} B + R)^{-1} B^T \Pi_{N-1} A
\end{aligned} \tag{22}$$

The recursion to go from Π_{N-1} to Π_{N-2} is known as backward Ricatti iteration and is defined in (23).

$$\Pi_{k-1} = Q + A^T \Pi_k A - A^T \Pi_k B (B^T \Pi_k B + R)^{-1} B^T \Pi_k A \tag{23}$$

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 2.11.

Theorem 2.11. Solution of the Finite Horizon LQR control problem Given a finite horizon N and a discrete linear system as shown in (15) the optimal control policy which minimises the LQR objective function of defintion 2.24 is given by iterating (24) backwards for $k = N-1, N-2, ..., 1$ using backward Ricatti iteration as shown in (23).

$$\begin{aligned}
&u_k^0(x) = K_k x \\
&K_k = -(B^T \Pi_{k+1} B + R)^{-1} B^T \Pi_{k+1} A
\end{aligned} \tag{24}$$

The optimal cost to go from time k to time N is $V_k^0(x) = \frac{1}{2}x^T \Pi_k x$.

After the optimal input $\mathbf{u}$ is found only u_0 is applied. For a treatment of the continuous case see [20].

Unfortunately optimal control in the setting described above does not guarantee stable control [38]. 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 2.25. Infinite Horizon LQR problem Find the optimal control sequence $\mathbf{u}$ which solves (25).

$$\begin{aligned}
&\min_{\mathbf{u}} V(x, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{\infty}\left(x_k^T Q x_k + u_k^T R u_k\right) \\
&\text{subject to } x_{t+1} = Ax_t + Bu_t \\
&\text{and } x_0 = x
\end{aligned} \tag{25}$$

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 2.12 is convergent and stabilising.

Definition 2.26. 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 2.12. Solution of the Infinite Horizon LQR control problem Given the Infinite Horizon LQR problem of definition 2.25 it can be shown that the optimal control is given by (26).

$$\begin{aligned} u_k^0(x) &= Kx \\ K &= -(B^T \Pi B + R)^{-1} B^T \Pi A \\ \Pi &= Q + A^T \Pi A - A^T \Pi B (B^T \Pi B + R)^{-1} B^T \Pi A \end{aligned} \tag{26}$$

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

2.4.2 Reference Tracking

The LQR control problem, as discussed in the previous subsection, 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 as shown in (27). The constants x_{sp} and u_{sp} are the state and corresponding controller set point one would like to drive the system to.

$$\begin{aligned} \tilde{x}_t &= x_t - x_{sp} \\ \tilde{u}_t &= u_t - u_{sp} \end{aligned} \tag{27}$$

The deviation variables are then used in the objective function as opposed to x_t and u_t as shown in (28). Note that the system dynamics remain the same [38].

$$\begin{aligned} &\min_{\tilde{\mathbf{u}}} V(x_0, \tilde{\mathbf{u}}) = \frac{1}{2} \sum_{k=0}^{N-1} \left(\tilde{x}_k^T Q \tilde{x}_k + \tilde{u}_k^T R \tilde{u}_k \right) + \frac{1}{2} \tilde{x}_N^T P_f \tilde{x}_N \\ &\text{subject to } \tilde{x}_{t+1} = A\tilde{x}_t + B\tilde{u}_t \end{aligned} \tag{28}$$

As before only $\tilde{u}_0$ is used. 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 (29). Note that H relates the observed variables to the controlled variables.

$$\begin{pmatrix} I - A & -B \\ HC & 0 \end{pmatrix} \begin{pmatrix} x_{sp} \\ u_{sp} \end{pmatrix} = \begin{pmatrix} 0 \\ r_{sp} \end{pmatrix} \tag{29}$$

If there are more measured outputs than manipulated variables (29) cannot be solved directly. It is possible to cast (29) into an optimisation problem. We refer the reader to [38] for a full treatise on the subject.

2.4.3 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 deals with a performance measure for stochastic systems of the form (30).

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_t \\ y_t &= Cx_t + v_t \end{aligned} \tag{30}$$

With $w_t \sim \mathcal{N}(0, W)$ 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 2.27.

Definition 2.27. Linear Quadratic Gaussian (LQG) Objective Function The controller minimising the quadratic objective function, shown in (31), is called the LQG controller.

$$V(x_0, \mathbf{u}) = \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \tag{31}$$

The restrictions on the tuning parameters are the same as before.

The full LQG control problem is stated in (32). Note that we only observe noisy y_t and that $C \neq I$ in general.

$$\begin{aligned} &\min_{\mathbf{u}} V(x_0, \mathbf{u}) = \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t \\ &\text{and } y_t = Cx_t + v_t \end{aligned} \tag{32}$$

It is indeed possible to solve this controller analytically using stochastic Dynamic Programming but the derivation is tedious. We rather employ the Separation Principle [10]. We do however re-derive the optimal controller in a later section.

Definition 2.28. 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.

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

Definition 2.29. 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 5.

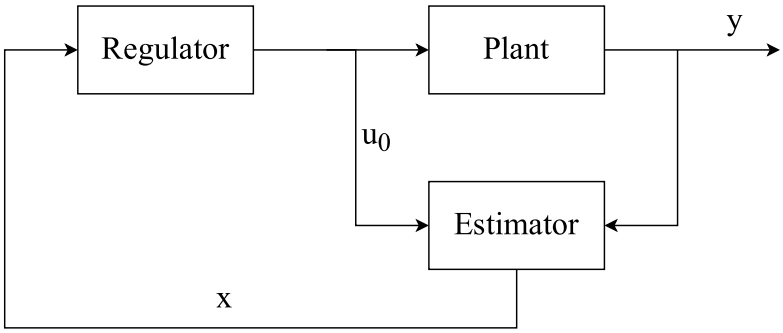

Figure 5: LQG control schematic

The LQR and LQG controllers are two of the most fundamental results in Optimal Control theory [20]. In the next section we discuss Model Predictive Control (MPC) which is a generalisation of the controllers we have discussed so far.

2.4.4 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 in (33).

$$\begin{aligned} &\min_{\mathbf{u}} V(x_0, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t \\ &\text{and } d^T x_t + e \geq 0 \;\forall\; t = 1, 2, ..., N \end{aligned} \tag{33}$$

It is straightforward to incorporate more constraints of the form $d_i^T x_t + e_i \geq 0$ if required. The structure of (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 (QP) techniques which are fast and reliable.

Recent advances in QP algorithms, like the Interior Point algorithm, have made it possible to solve QP problems almost as fast as Linear Programming (LP) problems. These solvers typically exploit the sparseness structure inherent in problems like (33) [31]. Thus, it is desirable to make use of these solvers wherever possible by ensuring that the underlying QP structure is not lost when modifications are made to the algorithm.

Finally, the stability and robustness of deterministic linear MPC is well studied and understood in modern literature. See [38] and [31] 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.

2.5 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 [23] for more details.

Theorem 2.13. 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 in (34) holds with $p = (d - c^T A^{-1} b)$.

$$\begin{pmatrix} A & b \\ c^T & d \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1}(I + p^{-1} b c^T A^{-1}) & -p^{-1} A^{-1} b \\ -p^{-1} c^T A^{-1} & p^{-1} \end{pmatrix} \tag{34}$$

3 Hidden Markov Models

In this section we consider probabilistic graphical models of the form shown in Figure 6. We assume that the n state state random variable X and the m state observation random variable Y are discrete random variables. Models of this form are classically called Hidden Markov Models (HMMs).

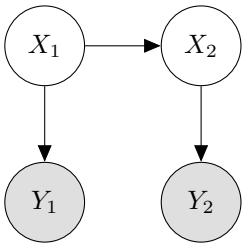

Figure 6: Graphical model of this section

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. Note that at each new time step we create two new random variables X_t, Y_t.

In this section 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 section because the techniques we develop here generalise to Latent Linear Dynamical Systems which is the next section.

3.1 Markov Models

A first order Markov Model (sometimes called a Markov Chain) is shown in Figure 7. Using the chain rule for Bayesian Networks we can immediately write down the joint probability distribution as shown in (35).

$$P(X_{1:T}) = \Pi_{t=1}^{T} P(X_t|X_{t-1}) \text{ with } P(X_1|X_0) = P(X_1) \tag{35}$$

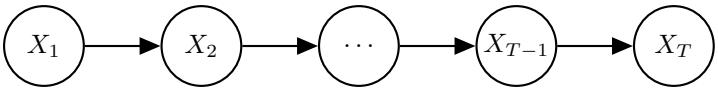

Figure 7: First Order Markov Chain

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 $X_t \perp\!\!\!\perp X_{1:t-2} | X_{t-1}$. Thus, we only have to marginalise out the previous time step to compute the required distribution as shown in (36).

$$P(X_t) = \sum_{x_{t-1}} P(X_t, x_{t-1}) = \sum_{x_{t-1}} P(x_{t-1}) P(X_t | x_{t-1}) \tag{36}$$

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 (36) in vector notation (37). Note that $P(X_t)$ is a discrete random variable and can thus be expressed as a stochastic column vector i.e. $\sum_i P(x_t = i) = 1$.

$$P(X_t) = \mathbf{p}_t = \mathbf{A}\mathbf{p}_{t-1} = \mathbf{M}^{t-1}\mathbf{p}_1 \tag{37}$$

We have implicitly rewritten (37) 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 $\mathbf{p}_1$ we call $\mathbf{p}_\infty$ the equilibrium distribution of the chain.

We define the stationary distribution, in matrix notation, by (38).

$$\mathbf{p}_\infty = \mathbf{A}\mathbf{p}_\infty \tag{38}$$

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 [44]. Intuitively $\mathbf{p}_\infty$ represents the steady state distribution of the random variable X as it is propagated through time by the transition function A. See the work by Streicher, Wilken and Sandrock for an application specifically geared towards Chemical Engineering [41].

3.2 Hidden Markov Models

Hidden Markov Models extend Markov Models by incorporating the observed random variables Y as shown in Figure 6. 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 to be discrete but we do so for the sake of simplicity here. In later sections 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 $P(X_1)$, the transition probability function $P(X_t|X_{t-1})$, the observation or 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 6. **This seems like a clunky definition check again**

We assume that the transition and observation probability functions are stationary. Consequently they may be represented by the row stochastic square matrices $P(x_t = i | x_{t-1} =$

$j) = \mathbf{A}$ and $P(y_t = i|x_t = j) = \mathbf{B}$. 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 the previous section 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 6. In the next section we show that these derivations also hold for the continuous case with some minor modifications.

3.2.1 Filtering

The goal of filtering is to find $P(X_t|y_{1:t})$: the distribution of the current state given all the past observations. We start the derivation by noting that X_{t-1} d-separates X_t from $X_{1:t-2}$. 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 $P(X_t, X_{t-1}|y_{1:t})$.

$$\begin{aligned} P(X_t|y_{1:t-1}) &= \sum_{x_{t-1}} P(X_t, x_{t-1}|y_{1:t-1}) \\ &= \sum_{x_{t-1}} P(x_{t-1}|y_{1:t-1}) P(X_t|y_{1:t-1}, x_{t-1}) \\ &= \sum_{x_{t-1}} P(x_{t-1}|y_{1:t-1}) P(X_t|x_{t-1}) \end{aligned} \tag{39}$$

Where the expansion followed from the chain rule and the cancellation followed from the conditional independence assumption of the transition function. Now we define $\alpha(X_t) \equiv P(X_t|y_{1:t})$. Then (40) follows from (39) and by application of Bayes' Theorem.

$$\begin{aligned} \alpha(X_t) &= \frac{P(y_t|X_t, y_{1:t-1}) P(X_t|y_{1:t-1})}{P(y_t|y_{1:t-1})} \\ &= \frac{P(y_t|X_t) P(X_t|y_{1:t-1})}{P(y_t|y_{1:t-1})} \\ &= \frac{P(y_t|X_t) \sum_{x_{t-1}} P(x_{t-1}|y_{1:t-1}) P(X_t|x_{t-1})}{P(y_t|y_{1:t-1})} \\ &= \frac{P(y_t|X_t) \sum_{x_{t-1}} \alpha(x_{t-1}) P(X_t|x_{t-1})}{P(y_t|y_{1:t-1})} \end{aligned} \tag{40}$$

Note that it is not actually necessary to calculate $p(y_t|y_{1:t-1})$ as it is only a normalisation constant. We thus have a recursion relation for the filtered posterior distribution X_t with initial condition $\alpha(X_1) = P(X_1, y_1) = P(X_1)P(y_1|X_1)$ as shown in (41).

$$\alpha(X_t) \propto P(y_t|X_t) \sum_{x_{t-1}} \alpha(x_{t-1}) P(X_t|x_{t-1}) \tag{41}$$

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 [3].

3.2.2 Smoothing

The goal of smoothing is to find $P(X_t|y_{1:T})$ 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 first is often called the Backwards algorithm in literature [35].

We start by splitting the joint $P(X_t, y_{1:T}) = P(X_t, y_{1:t})P(y_{t+1:T}|X_t, y_{1:t})$ by the chain rule and using d-separation to reduce it further $P(X_t, y_{1:T}) = P(X_t, y_{1:t})P(y_{t+1:T}|X_t)$. Effectively the last step implies that future observations are independent of past observations given the current state. We have defined $\beta(X_t) \equiv P(y_{t+1:T}|X_t)$ and continue the derivation in (42).

$$\begin{aligned} P(y_{t:T}|X_{t-1}) &= \sum_{x_t} P(y_t, y_{t+1:T}, x_t|X_{t-1}) \\ &= \sum_{x_t} P(y_{t+1:T}, x_t|X_{t-1})P(y_t|y_{t+1:T}, x_t, X_{t-1}) \\ &= \sum_{x_t} P(y_{t+1:T}, x_t|X_{t-1})P(y_t|x_t) \\ &= \sum_{x_t} P(x_t|X_{t-1})P(y_{t+1:T}|x_t, X_{t-1})P(y_t|x_t) \\ &= \sum_{x_t} P(x_t|X_{t-1})P(y_{t+1:T}|x_t)P(y_t|x_t) \end{aligned} \tag{42}$$

We have made judicious use of the implied independence assertions (via d-separation) of Figure 6. Making use of the definition of β we have (43).

$$\begin{aligned} &\beta(X_{t-1}) = \sum_{x_t} P(x_t|X_{t-1})\beta(x_t)P(y_t|x_t) \text{ for } 2 \leq t \leq T \\ &\text{with } \beta(X_T) = \mathbf{1} \end{aligned} \tag{43}$$

The recursion initial condition $\beta(X_T) = 1$ stems from Bayes' Theorem and the definition of α as shown in (44). Note that β is not a probability function.

$$\begin{aligned} P(X_T|y_{1:T}) &= \frac{P(X_T, y_{1:T})}{P(y_{1:T})} \\ &= \alpha(X_T)\beta(X_T) \\ &= P(X_T|y_{1:T})\beta(X_T) \\ &\implies \beta(X_T) = \mathbf{1} \end{aligned} \tag{44}$$

The smoothed posterior is given by applying Bayes' Theorem as shown in (45).

$$P(X_t|y_{1:T}) = \frac{\alpha(X_t)\beta(X_t)}{\sum_{x_t} \alpha(x_t)\beta(x_t)} \tag{45}$$

Together the $\alpha-\beta$ recursions are called the Forwards-Backwards algorithm and find extensive use in general purpose exact inference of Dynamic Bayesian Networks [35].

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 [3].

3.2.3 Viterbi Decoding

The goal of Viterbi Decoding is to find $x^*_{1:T} = \arg\max_{x_{1:T}} P(x_{1:T}|y_{1:T})$: finding the most likely sequence of states which best describe the observations by attempting to find the sequence $x_{1:T}$ such that the joint probability function $P(x_{1:T}, y_{1:T})$ is maximised. This is equivalent to finding $\arg\max_{x_{1:T}} P(x_{1:T}|y_{1:T})$ because, if one uses the chain rule on the joint, the observations will just be a constant factor.

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 in (46).

$$\begin{aligned}\max_{x_{1:T}} P(x_{1:T}, y_{1:T}) &= \max_{x_{1:T}} \Pi_{t=1}^{T} P(y_t|x_t)P(x_t|x_{t-1}) \\ &= \left(\max_{x_{1:T-1}} \Pi_{t=1}^{T-1} P(y_t|x_t)P(x_t|x_{t-1})\right) \max_{x_T} P(y_T|x_T)P(x_T|x_{T-1})\end{aligned} \tag{46}$$

Defining $\mu(X_{t-1}) \equiv \max_{x_t} P(y_t|x_t)P(x_t|x_{t-1})$ we can rewrite (46) as (47).

$$\max_{x_{1:T}} P(x_{1:T}, y_{1:T}) = \max_{x_{1:T-1}} \Pi_{t=1}^{T-1} P(y_t|x_t)P(x_t|x_{t-1})\mu(x_{T-1}) \tag{47}$$

Thus we have a recursive expression to find the value of the joint under the most likely sequence of states given the observations as shown in (48).

$$\begin{aligned}&\mu(x_{t-1}) = \max_{x_t} P(y_t|x_t)P(x_t|x_{t-1})\mu(x_t) \text{ for } 2 \leq t \leq T \\ &\text{with } \mu(x_T) = 1\end{aligned} \tag{48}$$

The recursive expression in (48) implies that the effect of maximising over 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_1. We then find the state which maximises this joint and pass this message forward. Continuing in this way, we have (49).

$$\begin{aligned}x^*_1 &= \arg\max_{x_1} P(y_1|x_1)P(x_1)\mu(x_1) \\ x^*_2 &= \arg\max_{x_2} P(y_2|x_2)P(x_2|x^*_1)\mu(x_2) \\ &\vdots \\ x^*_t &= \arg\max_{x_t} P(y_t|x_t)P(x_t|x^*_{t-1})\mu(x_t)\end{aligned} \tag{49}$$

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.

3.2.4 Prediction

The goal of prediction is to find $P(X_{t+1}|y_{1:t})$ and $P(Y_{t+1}|y_{1:t})$: 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. 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 derive (50).

$$\begin{aligned} P(X_{t+1}|y_{1:t}) &= \sum_{x_t} P(X_{t+1}, x_t|y_{1:t}) \\ &= \sum_{x_t} P(x_t|y_{1:t}) P(X_{t+1}|x_t, y_{1:t}) \\ &= \sum_{x_t} P(x_t|y_{1:t}) P(X_{t+1}|x_t) \\ &= \sum_{x_t} \alpha(x_t) P(X_{t+1}|x_t) \end{aligned} \tag{50}$$

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 derive (51).

$$\begin{aligned} P(Y_{t+1}|y_{1:t}) &= \sum_{x_t, x_{t+1}} P(x_{t+1}, x_t, Y_{t+1}|y_{1:t}) \\ &= \sum_{x_t, x_{t+1}} P(x_t|y_{1:t}) P(x_{t+1}, Y_{t+1}|y_{1:t}, x_t) \\ &= \sum_{x_t, x_{t+1}} P(x_t|y_{1:t}) P(x_{t+1}|y_{1:t}, x_t) P(Y_{t+1}|y_{1:t}, x_t, x_{t+1}) \\ &= \sum_{x_t, x_{t+1}} P(x_t|y_{1:t}) P(x_{t+1}|x_t) P(Y_{t+1}|x_{t+1}) \\ &= \sum_{x_t, x_{t+1}} \alpha(x_t) P(x_{t+1}|x_t) P(Y_{t+1}|x_{t+1}) \end{aligned} \tag{51}$$

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.

3.3 Burglar Localisation Problem

While the focus of this dissertation is not on HMM 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* [3]. 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 8: 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.

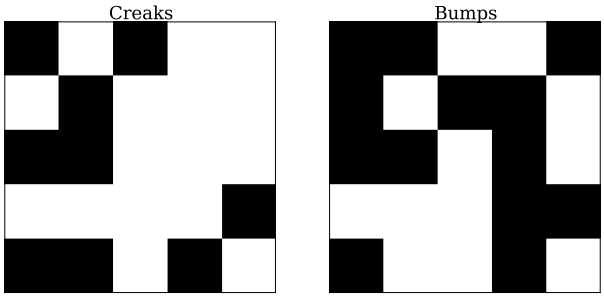

Figure 8: Burglar Problem Observations

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 9.

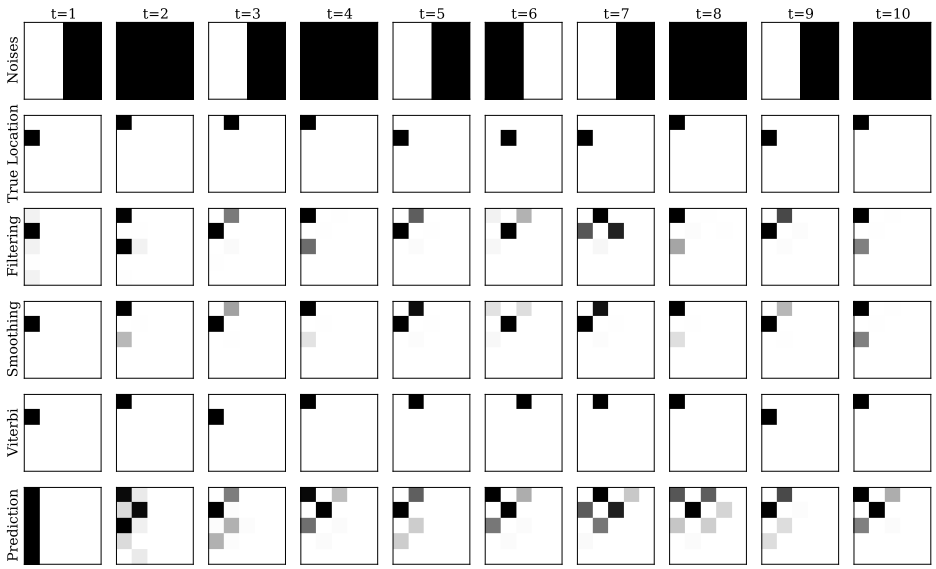

Figure 9: Burglar Problem: Filter, Smoothing, Viterbi Decoding and Prediction

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 [3]. Many extensions of the basic model (see Figure 6) 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 section carry over to the continuous case as we will see next.

4 CSTR Model

In this section 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 10. The model is taken from literature [33].

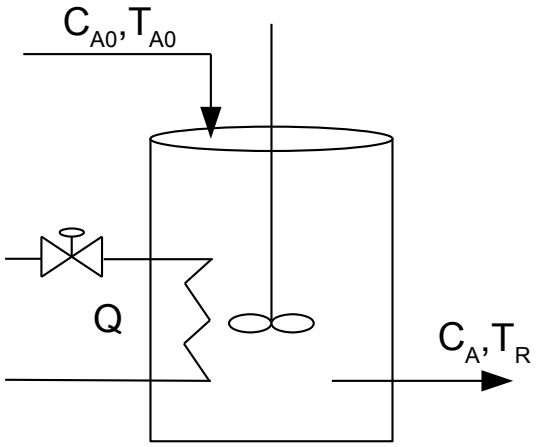

Figure 10: Diagram of a simple CSTR where the heat added to system is the only manipulated variable.

The state space equations describing the reactor are shown in (52) with parameters shown in Table 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.

$$\begin{aligned} \dot{C_A} &= f(C_A, T_R) = \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT_R}} C_A \\ \dot{T_R} &= g(C_A, T_R) = \frac{F}{V}(T_{A0} - T_A) + \frac{-\triangle H}{\rho C_p} k_0 e^{\frac{-E}{RT_R}} C_A + \frac{Q}{\rho C_p V} \end{aligned} \tag{52}$$

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	$100 \times 10^{-3}\ \frac{m^3}{min}$

Table 1: CSTR parameters

The CSTR model is a familiar control example. Similar models may be found in [16][8][37][47]. We use this model because it is low dimensional yet complex enough to illustrate the principles

we investigate. 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.

4.1 Qualitative Analysis

In this section we use standard mathematical tools, as found in [18], to analyse the qualitative behaviour of the CSTR. By inspecting (52) we see that the model is coupled and non-linear. By solving (53) we see that for nominal operating conditions ($Q = 0$) there exist 3 operating points (critical points) as shown in Table 2.

$$\begin{aligned} 0 &= \frac{F}{V}(C_{A0} - C_A) - k_0 e^{\frac{-E}{RT_R}} C_A \\ 0 &= \frac{F}{V}(T_{A0} - T_A) + \frac{-\triangle H}{\rho C_p} k_0 e^{\frac{-E}{RT_R}} C_A + \frac{Q}{\rho C_p V} \end{aligned} \tag{53}$$

Critical Point	C_A	T_R	Stability
(C_A^1, T_R^1)	0.0097	508.0562	Stable Improper Node
(C_A^2, T_R^2)	0.4893	412.1302	Unstable Saddle Point
(C_A^3, T_R^3)	0.9996	310.0709	Stable Improper Node

Table 2: Nominal operating points for the CSTR

The stability of the operating points were found by linearising (52) and computing the eigenvalues of the Jacobian, shown in (54), at each critical point.

$$J(C_A, T_R) = \begin{pmatrix} -\frac{F}{V} - k_0 e^{\frac{-E}{RT_R}} & -k_0 e^{\frac{-E}{RT_R}} C_A \left(\frac{E}{RT_R^2}\right) \\ \frac{-\triangle H}{\rho C_p} k_0 e^{\frac{-E}{RT_R}} & -\frac{F}{V} + \frac{-\triangle H}{\rho C_p} k_0 e^{\frac{-E}{RT_R}} C_A \left(\frac{E}{RT_R^2}\right) \end{pmatrix} \tag{54}$$

In Figure 11 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 [30].

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 3.

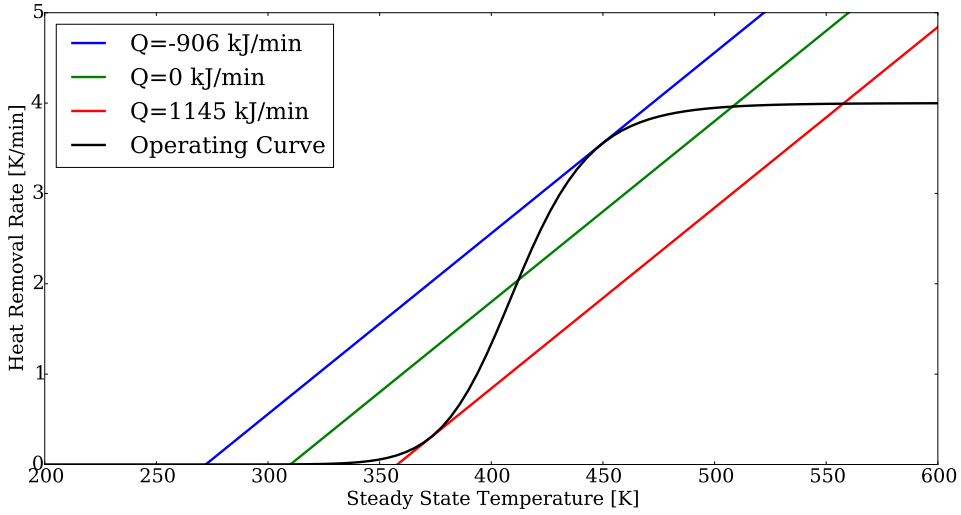

Figure 11: CSTR operating curve with different input curves. Nominal operating conditions are $Q = 0$ kJ/min.

Heat Input	C_A	T_R	Stability
$Q = -906$ kJ/min	0.1089	450.3531	Stable Improper Node
$Q = -906$ kJ/min	0.9999	272.1346	Stable Improper Node
$Q < -906$ kJ/min	$\backsim$	$\backsim$	Stable Improper Node
$Q = 1145$ kJ/min	0.0017	557.5243	Stable Improper Node
$Q = 1145$ kJ/min	0.9263	372.5959	Stable Improper Node
$Q > 1145$ kJ/min	$\backsim$	$\backsim$	Stable Improper Node

Table 3: Bifurcation analysis of the CSTR at different heat input values.

The multiple stable critical points for $Q \in [-906, 1145]$ kJ/min makes control of this system challenging. For example consider a situation where one starts at some point on the black line, below the green line in Figure 11. If one wishes to move to the high temperature low concentration stable operating point large, non-smooth, controller action will be required. By slowly heating up the CSTR the green line will gradually move to the right and this will push the system, somewhat counter-intuitively, towards the low temperature high concentration critical point. It is necessary to quickly heat up the CSTR so that the green line is below the current operating point on the black line. The self-regulatory nature of the CSTR will then move the system to the desired operating point.

4.2 Nonlinear Model

In this section we evaluate the transient response of the CSTR. The nonlinear differential equation shown in (52) is intractably difficult to solve analytically. For this reason we will use a numerical method, specifically the Runge-Kutta method [18], 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. The explicit nature of the method will also be useful later when it is necessary to discretise the system in the standard linear state space form.

For completeness we show the method here. Suppose we have an autonomous ordinary differential equation as shown in (55) 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.

$$\begin{aligned} &\dot{x}(t) = f(x(t)) \\ &\text{with } x(t) = x_a \text{ for } t = t_a \end{aligned} \tag{55}$$

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 shown in (56) is called the Runge-Kutta method. We assume that for sufficiently small time steps, h, the method is stable and convergent.

$$\begin{aligned} x_{t+1} &= x_t + \frac{h}{6}\left(k_1 + 2k_2 + 2k_3 + k4\right) \\ k_1 &= f(x_t) \\ k_2 &= f(x_t + \frac{h}{2}k_1) \\ k_3 &= f(x_t + \frac{h}{2}k_2) \\ k_4 &= f(x_t + hk_3) \end{aligned} \tag{56}$$

By applying the Runge-Kutta method to the CSTR we have Figures 12 and 13. 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.

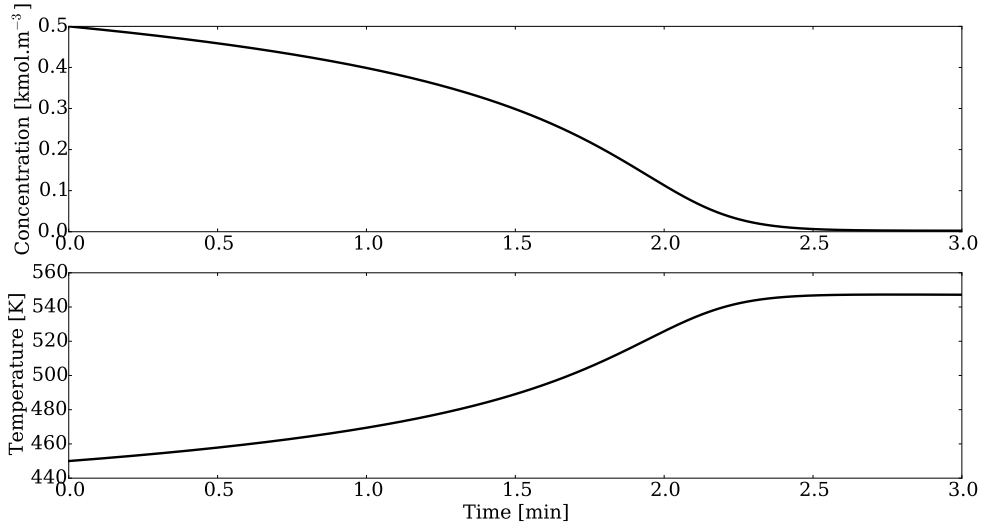

Figure 12: Transient response of the CSTR under nominal operating conditions with initial condition $(0.5, 450)$ and $h = 0.001$.

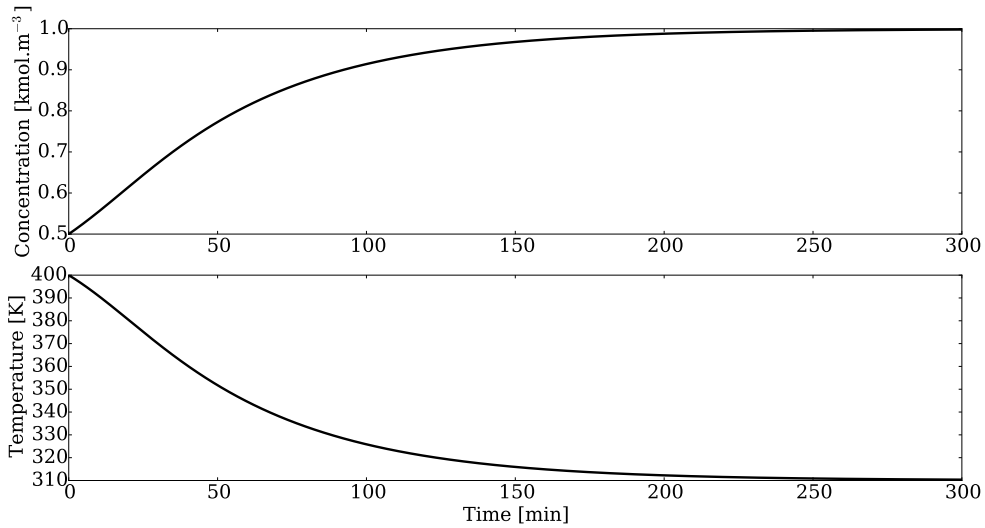

Figure 13: Transient response of the CSTR under nominal operating conditions with initial condition $(0.5, 400)$ and $h = 0.001$.

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 many control techniques have been designed specifically for linear systems. 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 12, 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 13, 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.

4.3 Linearised Models

The approach of using piecewise affine (linear) functions for control, based on linearisation around critical points, has been investigated in literature [16][26]. Typically the state domain is discretised into regimes and the linear approximation of the model in each regime is used for control. The benefit of this approach is that the non-linear problem can then be handled by linear methods for which efficient algorithms exist. Drawbacks of this approach are computational complexity [16] and poor control performance because the models are inaccurate.

We will also attempt to use linear models for the purposes of control. First we present a general linearisation technique. Consider an arbitrary point in the state space (C_A^*, T_R^*).

Then (57) is the general linearised model around (C_A^*, T_R^*).

$$\begin{pmatrix} \dot{C}_A \\ \dot{T}_R \end{pmatrix} = \begin{pmatrix} f(C_A^*, T_R^*) \\ g(C_A^*, T_R^*) \end{pmatrix} + J(C_A^*, T_R^*) \left(\begin{pmatrix} C_A \\ T_R \end{pmatrix} - \begin{pmatrix} C_A^* \\ T_R^* \end{pmatrix} \right) \tag{57}$$

It is often desirable to change the variables such that (57) has no constant terms. This change of variables, which holds even if the linearisation point is not a critical point of the model, is shown in (58).

$$\begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix} = \begin{pmatrix} C_A \\ T_R \end{pmatrix} - J(C_A^*, T_R^*)^{-1} \left(J(C_A^*, T_R^*) \begin{pmatrix} C_A^* \\ T_R^* \end{pmatrix} - \begin{pmatrix} f(C_A^*, T_R^*) \\ g(C_A^*, T_R^*) \end{pmatrix}_{Q=0} \right) \tag{58}$$

We then have (59). Note that the input term B originates from $\begin{pmatrix} f(C_A^*, T_R^*) \\ g(C_A^*, T_R^*) \end{pmatrix}$ and in (58) we specifically set it to zero so that it is not removed.

$$\frac{d}{dx} \begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix} = J(C_A^*, T_R^*) \begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix} + B(C_A^*, T_R^*) Q \tag{59}$$

We now use the Bilinear Transform (also known as Tustin's Transform) to convert (59) into the discrete equation (60). Note that (60) implicitly depends on the sampling time.

$$\begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix}_{t+1} = A(C_A^*, T_R^*) \begin{pmatrix} \tilde{C}_A \\ \tilde{T}_R \end{pmatrix}_t + B(C_A^*, T_R^*) Q \tag{60}$$

Where 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 (58) when we want to inspect the results of applying (60).

Figure 14 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 2.

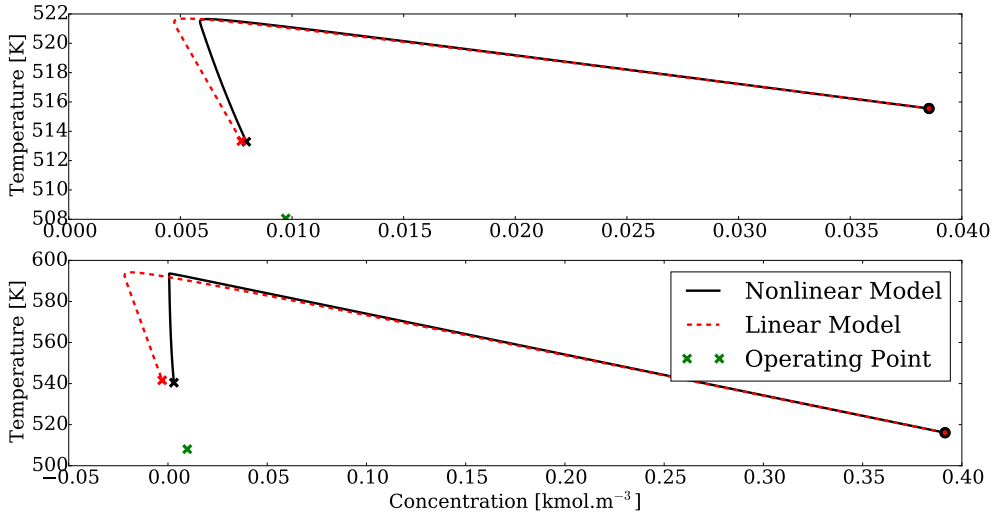

Figure 14: State space response of the CSTR under nominal operating conditions linearised around (C_A^1, T_R^1) with different initial conditions. The dot indicates where the simulation started and the cross where it finished.

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 15 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.

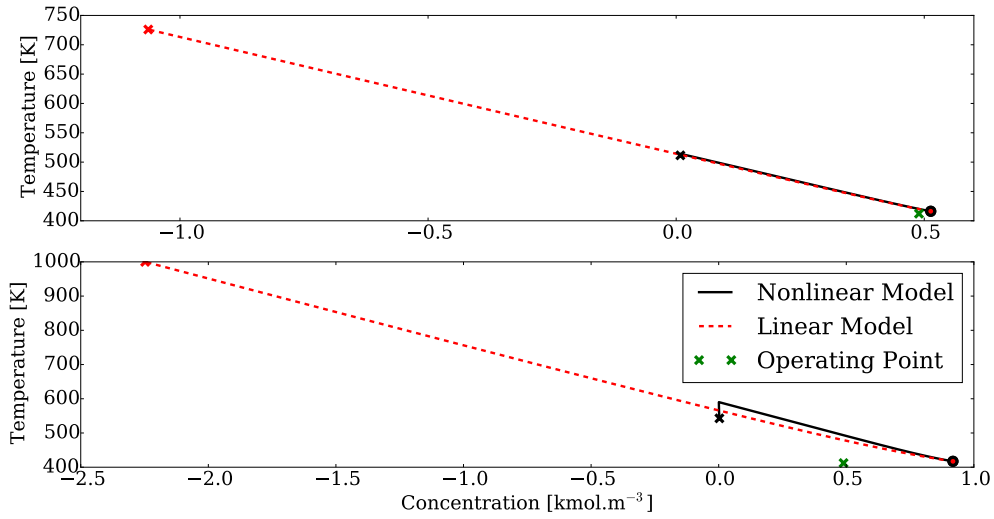

Figure 15: State space response of the CSTR under nominal operating conditions linearised around (C_A^2, T_R^2) with different initial conditions. The dot indicates where the simulation started and the cross where it finished.

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 16 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.

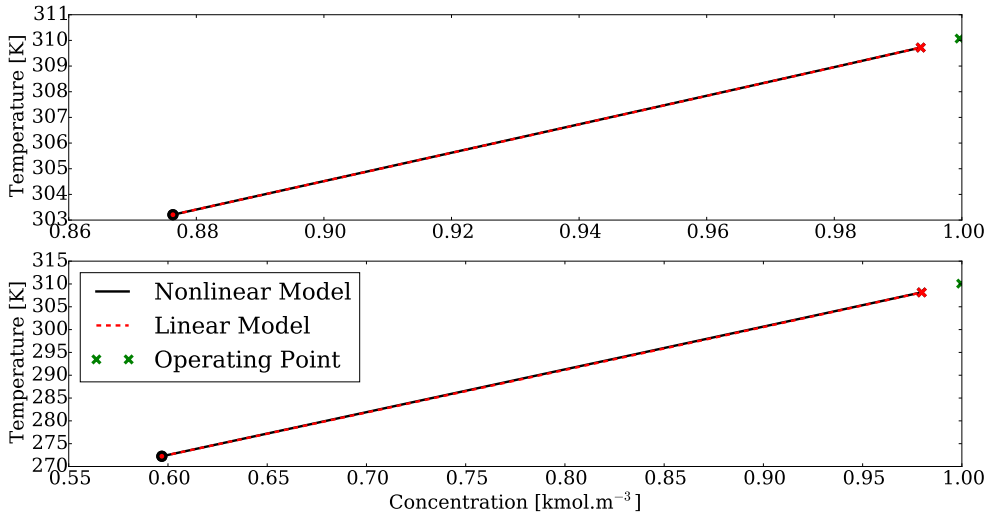

Figure 16: State space response of the CSTR under nominal operating conditions linearised around (C_A^3, T_R^3) with different initial conditions. The dot indicates where the simulation started and the cross where it finished.

Out of the 3 linear models we have considered so far, this one seems to be the most accurate for initial conditions near it. This could possibly be because the system dynamics are quite slow and thus more linear. However, we do not expect this linear model to be accurate in regions of the state space where the dynamics are faster e.g. near the high temperature operating point. This is shown in Figure 17.

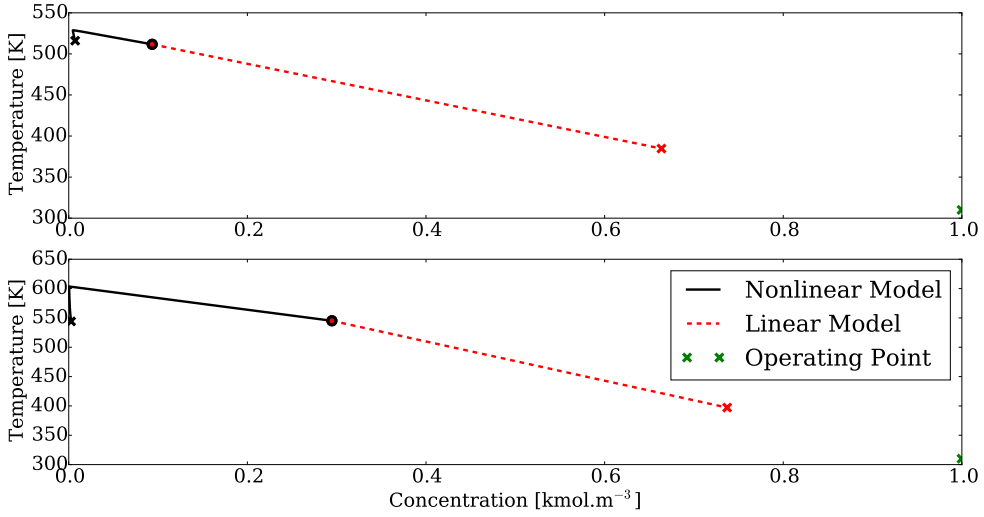

Figure 17: State space response of the CSTR under nominal operating conditions linearised around (C_A^3, T_R^3) with initial condition far away from the linearisation point.

Based on the general linearisation formula in (57) 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. For

example, suppose one has 3 linear models. Each model will be more accurate in different regions of the state space. By selecting a model to use based on some metric taking into account the current location in the state space, it is reasonable to suppose that one will be able to more accurately model the system.

In later sections we make precise which metric we will use in this dissertation. For now we merely illustrate this idea as shown in Figure 18. Here we have 10 different linear models with linearisation points chosen at random in the state space. We also include the 3 linear models with linearisation points at the nominal critical points of the system.

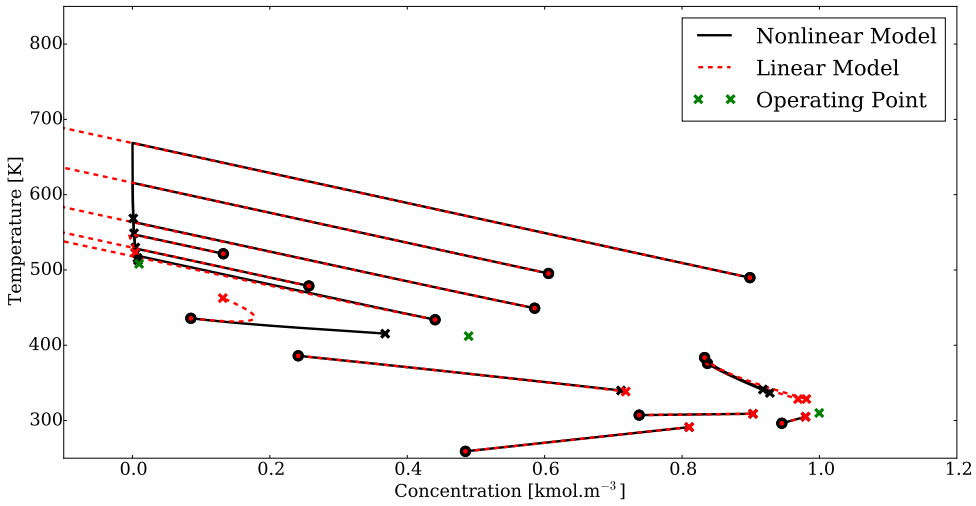

Figure 18: State space response of the CSTR under nominal operating conditions with 13 different linear models. Each linear model has an different initial condition located near the point of linearisation.

Based on Figure 18 we can see that, at least initially, the linear models describe the nonlinear evolution well (the red and black lines overlap). Inevitably they diverge but we suppose that it is possible to switch linear models when this happens. Ideally we would then switch to a linear model which better describes the dynamics.

However, Figure 18 is misleading in the sense that we do not show the rate at which the linear models diverge from the nonlinear model. Overlap of the black and red lines do not indicate a one to one correspondence with respect to the time evolution of the two systems. The linear models in the upper half of the state space (the high temperature region)"move" much faster than the nonlinear system. In Figure 18 we see that these models rapidly become unbounded. We expect this to be problematic if we are to use linear models for prediction (i.e. control) of the system.

To illustrate this divergence we plot the maximum absolute relative difference between the linear models and the nonlinear model at $t = 0.5$ min in Figure 19, at $t = 5$ min in Figure 20 and $t = 50$ min in Figure 21 .

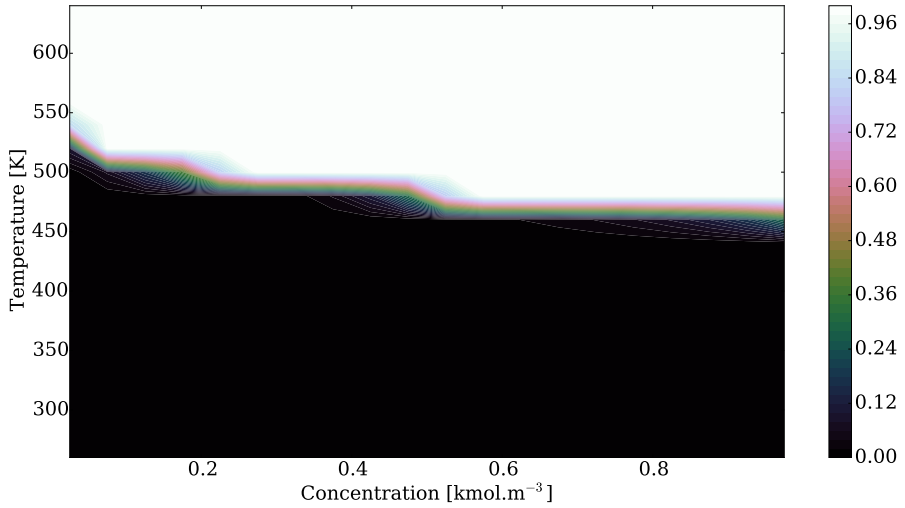

Figure 19: Maximum absolute relative difference between the linear models and the nonlinear model at $t = 0.5$ min. Each value in the state space serves as the linearisation point for the respective linear model. The linearisation points was also used as the respective initial conditions.

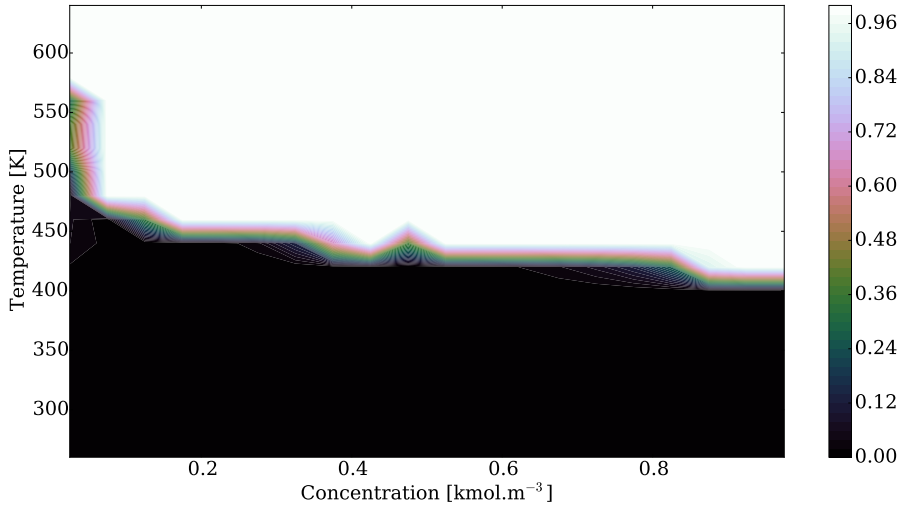

Figure 20: Maximum absolute relative difference between the linear models and the nonlinear model at $t = 5$ min.

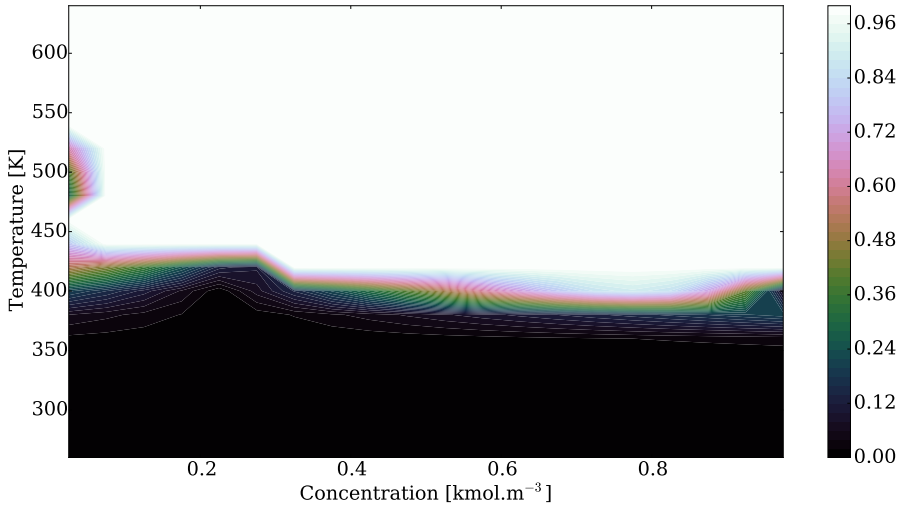

Figure 21: Maximum absolute relative difference between the linear models and the nonlinear model at $t = 50$ min.

We see that in both figures the high temperature region of the state space has the worst linear model accuracy. This could be attributed to the exponential term in (52) which is more pronounced in this area and consequently the model is more nonlinear. We also see that the high temperature linear models quickly become much less accurate with time when compared to the low temperature models.

5 Inference using Linear Models

In this section we consider probabilistic graphical models of the form shown in Figure 22. This model is a generalisation of the graphical model seen in the Hidden Markov Model section. We now assume that the states $(x_1, x_2, ...)$ and observations $(y_1, y_2, ...)$ are continuous random variables but the inputs $(u_1, u_2, ...)$ are deterministic. Models of this form are called Latent Linear Dynamical Systems (the famous Kalman Filter model falls into this category).

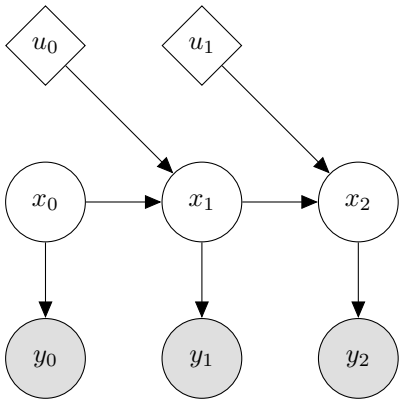

Figure 22: Graphical model of this section

In the previous section 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 data. 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.

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 section. We also assume that the transition and emission functions are time invariant and that the state space model is of the form (61).

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_{t+1} \text{ with } \mathcal{N}(w_{t+1}|0, W) \\ y_{t+1} &= Cx_{t+1} + v_{t+1} \text{ with } \mathcal{N}(v_{t+1}|0, V) \end{aligned} \tag{61}$$

Rewriting the state space model we see that the transition and emission probability density functions are given by (62). Note that we also assume that the system is first order Markov.

$$\begin{aligned} p(x_{t+1}|x_t, u_t) &= \mathcal{N}(x_{t+1}|Ax_t + Bu_t, W) \\ p(y_{t+1}|x_t) &= \mathcal{N}(y_{t+1}|Cx_{t+1}, V) \end{aligned} \tag{62}$$

We have implicitly assumed that the noise is Gaussian and white[3]. Intuitively one can think of V as the noise associated with state measurements and W being a form of the

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

uncertainty associated with the linear model of the plant. Additionally, W can also model any unmeasured disturbances which may influence the system[4]. Thus, larger V and W indicate more uncertainty in the system.

To fully specify the system we require the transition and emission probability density functions (these implicitly depend on the internal structure of the graphical model in Figure 22) as well as the prior (initial) distribution of x_0.

5.1 Filtering

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

We start with the prediction expression in (63) and assume, due to the closure of linear conditional Gaussian distributions, that $\alpha(x_{t-1}) = p(x_{t-1}|y_{1:t-1}, u_{1:t-2}) = \mathcal{N}(x_{t-1}|\mu_{t-1}, \Sigma_{t-1})$.

$$\begin{aligned} p(x_t|y_{1:t-1}, u_{1:t-1}) &= \int_{x_{t-1}} p(x_t, x_{t-1}|y_{1:t-1}, u_{1:t-1}) \\ &= \int_{x_{t-1}} p(x_{t-1}|y_{1:t-1}, u_{1:t-1}) p(x_t|x_{t-1}, y_{1:t-1}, u_{1:t-1}) \\ &= \int_{x_{t-1}} p(x_{t-1}|y_{1:t-1}, u_{1:t-2}) p(x_t|x_{t-1}, u_{1:t-1}) \\ &= \int_{x_{t-1}} \alpha(x_{t-1}) \mathcal{N}(x_t|Ax_{t-1} + Bu_{t-1}, W) \\ &= \int_{x_{t-1}} \mathcal{N}(x_{t-1}|\mu_{t-1}, \Sigma_{t-1}) \mathcal{N}(x_t|Ax_{t-1} + Bu_{t-1}, W) \end{aligned} \tag{63}$$

Now we use Bayes' Theorem for Linear Gaussian Models to evaluate the marginal expression as shown in (64).

$$\begin{aligned} \int_{x_{t-1}} \mathcal{N}(x_{t-1}|\mu_{t-1}, \Sigma_{t-1}) \mathcal{N}(x_t|Ax_{t-1} + Bu_{t-1}, W) &= \mathcal{N}(x_t|A\mu_{t-1} + Bu_{t-1}, W + A^T \Sigma_{t-1} A) \\ &= \mathcal{N}(x_t|\mu_{t|t-1}, \Sigma_{t|t-1}) \end{aligned} \tag{64}$$

Intuitively, (64) 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 Bayes' Theorem to

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

update our view of x_t given the current observation as shown in (65).

$$\begin{aligned} p(x_t|y_{1:t}, u_{1:t-1}) &= p(x_t|y_t, y_{1:t-1}, u_{1:t-1}) \\ &= \frac{p(y_t|x_t, y_{1:t-1}, u_{1:t-1})p(x_t|y_{1:t-1}, u_{1:t-2}, u_{t-1})}{p(y_t|y_{1:t-1}, u_{1:t-1})} \\ &= \frac{p(y_t|x_t)p(x_t|y_{1:t-1}, u_{1:t-1})}{p(y_t|y_{1:t-1}, u_{1:t-1})} \\ &\propto p(y_t|x_t)p(x_t|y_{1:t-1}, u_{1:t-1}) \\ &= p(y_t|x_t)\mathcal{N}(x_t|A\mu_{t-1} + Bu_{t-1}, W + A^T\Sigma_{t-1}A) \\ &= \mathcal{N}(y_t|Cx_t, V)\mathcal{N}(x_t|\mu_{t|t-1}, \Sigma_{t|t-1}) \end{aligned} \tag{65}$$

Now we again make use of Bayes' Theorem for Linear Gaussian Models to evaluate the conditional expression as shown in (66).

$$\begin{aligned} p(x_t|y_{1:t}, u_{1:t-1}) &= \mathcal{N}(y_t|Cx_t, V)\mathcal{N}(x_t|\mu_{t|t-1}, \Sigma_{t|t-1}) \\ &= \mathcal{N}(x_t|\Gamma(C^TV^{-1}y + \Sigma_{t|t-1}^{-1}\mu_{t|t-1}), \Gamma) \\ &\text{with } \Gamma = (\Sigma_{t|t-1}^{-1} + C^TV^{-1}C)^{-1} \end{aligned} \tag{66}$$

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 shown in (67). 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 as shown in (68). Together (67) and (68) are known as the Kalman Filter equations [36].

$$\Sigma_t = (I - K_tC)\Sigma_{t|t-1} \tag{67}$$

$$\mu_t = \mu_{t|t-1} + K_t(y_t - C\mu_{t|t-1}) \tag{68}$$

Note that 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 [5].

5.2 Prediction

The goal of prediction is to find an expression for the distributions $p(x_{t+h}|y_{1:t}, u_{1:t+h-1})$ and $p(y_{t+h}|y_{1:t}, u_{1:t+h-1})$ with $h \geq 1$. Note that these derivations follow in exactly the same way as the prediction derivations did for the Hidden Markov Models. 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).

We start the derivation by considering the one step ahead state prediction in (69).

$$\begin{aligned}
p(x_{t+1}|y_{1:t}, u_{1:t}) &= \int_{x_t} p(x_{t+1}, x_t|y_{1:t}, u_{1:t}) \\
&= \int_{x_t} p(x_t|y_{1:t}, u_{1:t-1}) p(x_{t+1}|x_t, y_{1:t}, u_{1:t}) \\
&= \int_{x_t} p(x_t|y_{1:t}, u_{1:t-1}) p(x_{t+1}|x_t, u_t) \\
&= \int_{x_t} \alpha(x_t) p(x_{t+1}|x_t, u_t) \\
&= \int_{x_t} \mathcal{N}(x_t|\mu_t, \Sigma_t) \mathcal{N}(x_{t+1}|Ax_t + Bu_t, W) \\
&= \mathcal{N}(x_{t+1}|Ax_t + Bu_t, W + A\Sigma_t A^T) \\
&= \mathcal{N}(x_{t+1}|\mu_{t+1|t}, \Sigma_{t+1|t})
\end{aligned} \tag{69}$$

Note that μ_t and Σ_t is the filtered mean and covariance. We have again relied upon Bayes' Theorem for Linear Gaussian Models to evaluate the marginal integral. We now consider the two step ahead state prediction in (70).

$$\begin{aligned}
p(x_{t+2}|y_{1:t}, u_{1:t+1}) &= \int_{x_{t+1}} p(x_{t+2}, x_{t+1}|y_{1:t}, u_{1:t+1}) \\
&= \int_{x_{t+1}} p(x_{t+1}|y_{1:t}, u_{1:t}) p(x_{t+2}|x_{t+1}, y_{1:t}, u_{1:t+1}) \\
&= \int_{x_{t+1}} p(x_{t+1}|y_{1:t}, u_{1:t}) p(x_{t+2}|x_{t+1}, u_{t+1}) \\
&= \int_{x_t} \mathcal{N}(x_{t+1}|\mu_{t+1|t}, \Sigma_{t+1|t}) \mathcal{N}(x_{t+2}|Ax_{t+1} + Bu_{t+1}, W) \\
&= \mathcal{N}(x_{t+2}|A\mu_{t+1|t} + Bu_{t+1}, W + A\Sigma_{t+1|t} A^T) \\
&= \mathcal{N}(x_{t+2}|\mu_{t+2|t}, \Sigma_{t+2|t})
\end{aligned} \tag{70}$$

It is clear that we have derived a recursive algorithm to estimate the h^{th}-step ahead state prediction as shown in (71).

$$\begin{aligned}
&p(x_{t+h}|y_{1:t}, u_{1:t+h}) = \mathcal{N}(x_{t+h}|\mu_{t+h|t}, \Sigma_{t+h|t}) \\
&\text{with } \mu_{t+h|t} = A\mu_{t+h-1|t} + Bu_{t+h-1} \\
&\text{and } \Sigma_{t+h|t} = W + A\Sigma_{t+h-1|t} A^T \\
&\text{and } \mu_{t+1|t} = A\mu_t + Bu_t \\
&\text{and } \Sigma_{t+1|t} = W + A\Sigma_t A^T
\end{aligned} \tag{71}$$

Inspecting (71) 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 for increasing h (obviously bounded by Q below). For, if the eigenvalues of A are less than one 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, $p(y_{t+h}|y_{1:t}, u_{1:t+h-1})$. Again consider the one

step ahead prediction as shown in (72).

$$\begin{aligned}
p(y_{t+1}|y_{1:t}, u_{1:t}) &= \int_{x_t, x_{t+1}} p(y_{t+1}, x_{t+1}, x_t|y_{1:t}, u_{1:t}) \\
&= \int_{x_t, x_{t+1}} p(x_t|y_{1:t}, u_{1:t-1}) p(y_{t+1}, x_{t+1}|x_t, y_{1:t}, u_{1:t}) \\
&= \int_{x_t, x_{t+1}} p(x_t|y_{1:t}, u_{1:t-1}) p(x_{t+1}|x_t, y_{1:t}, u_{1:t}) p(y_{t+1}|x_{t+1}, x_t, y_{1:t}, u_{1:t}) \\
&= \int_{x_t, x_{t+1}} \alpha(x_t) p(x_{t+1}|x_t, u_t) p(y_{t+1}|x_{t+1}) \\
&= \int_{x_t, x_{t+1}} \mathcal{N}(x_t|\mu_t, \Sigma_t) \mathcal{N}(x_{t+1}|Ax_t + Bu_t, W) \mathcal{N}(y_{t+1}|Cx_{t+1}, V) \\
&= \mathcal{N}(y_{t+1}|C\mu_{t+1|t}, V + C\Sigma_{t+1|t}C^T)
\end{aligned} \tag{72}$$

We have again used Bayes' Theorem for Linear Gaussian Models 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 given by (73).

$$p(y_{t+h}|y_{1:t}, u_{1:t+h-1}) = \mathcal{N}(y_{t+h}|C\mu_{t+h|t}, R + C\Sigma_{t+h|t}C^T) \tag{73}$$

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

5.3 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 except that we again make use of Bayes' Theorem for Linear Gaussian Models. 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 $p(x_t|y_{1:T}, u_{1:T-1})$ 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 shown in (74).

$$\begin{aligned}
&\hat{\mu}_t = \mu_t + J_t\left(\hat{\mu}_{t+1} - (A\mu_t + Bu_{t-1})\right) \\
&\hat{\Sigma}_t = \Sigma_t + J_t(\hat{\Sigma}_{t+1} - P_t)J_t^T \\
&\text{with } P_t = A\Sigma_t A^T + W \\
&\text{and } J_t = \Sigma_t A^T (P_t)^{-1} \\
&\text{and } \hat{\mu}_T = \mu_T \\
&\text{and } \hat{\Sigma}_T = \Sigma_T
\end{aligned} \tag{74}$$

Finally, we know from the Chain Rule for Bayesian Networks and Figure 22 that the joint distribution for $p(x_{1:T}, y_{1:T}, u_{1:T-1}) = p(x_1)p(y_1|x_1)\Pi_{t=2}^{T} p(y_t|x_t)p(x_t|x_{t-1}, u_{t-1})$. 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 [3].

5.4 Filtering the CSTR

In this section we apply the Kalman Filter to the CSTR introduced earlier. We use the linear model around the unstable operating point (C_A^2, T_R^2) as shown in (75). 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 we only measure temperature for now.

$$\begin{aligned} A &= \begin{pmatrix} 0.9959 & -6.0308 \times 10^{-5} \\ 0.4186 & 1.0100 \end{pmatrix} \\ B &= \begin{pmatrix} 0 \\ 8.4102 \times 10^{-5} \end{pmatrix} \\ C &= \begin{pmatrix} 0 & 1 \end{pmatrix} \\ W &= \begin{pmatrix} 1 \times 10^{-6} & 0 \\ 0 & 0.1 \end{pmatrix} \\ V &= \begin{pmatrix} 10 \end{pmatrix} \end{aligned} \tag{75}$$

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.

In Figure 23 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. During the initial part 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 the system can be described by a linear model. 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.

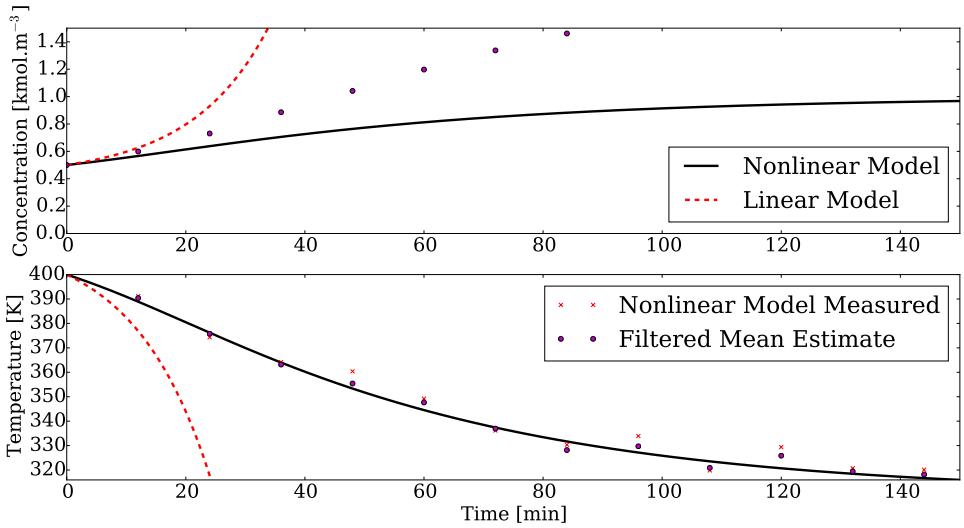

Figure 23: Kalman Filter superimposed on the time series evolution of the CSTR with initial condition $(0.50, 400)$ and measuring only temperature.

In Figure 24 we see another interesting property of Kalman Filters. The posterior covariance quickly converges to a constant value (the error ellipses quickly stop changing shape) which is independent of the observations. This is a general property of linear Gaussian systems [3] 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 error ellipses. Notice how they are 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 24 we see that while the temperature estimate is still trustworthy (the black crosses line up horizontally with the red crosses) the concentration estimate diverges.

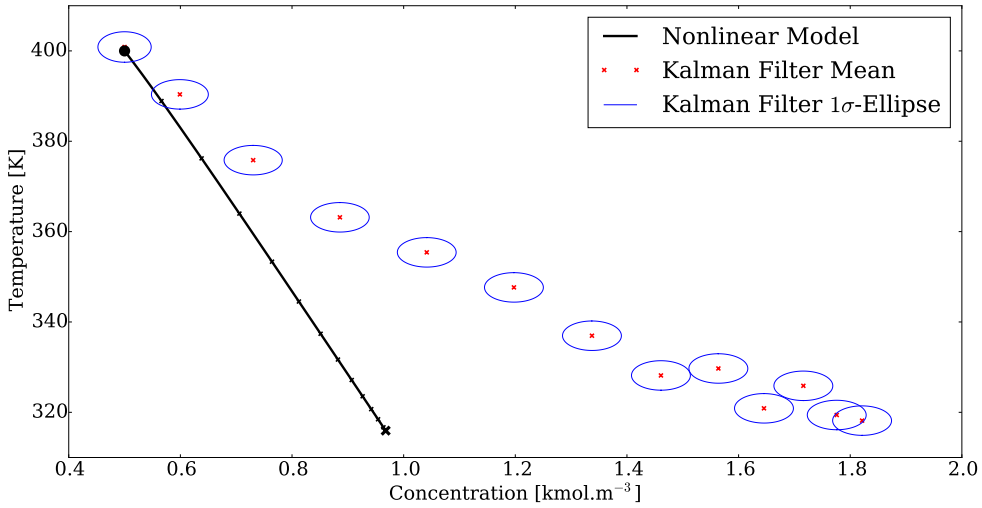

Figure 24: Phase plane of the hidden states of the CSTR with mean and one-sigma ellipses superimposed thereupon. Only temperature is measured.

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 [1].

Based on our discussion where the CSTR example was introduced we know that the linear models will not always be very accurate. We therefore modify (75) to also incorporate concentration measurements. 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 25 and the state space representation is shown in Figure 26.

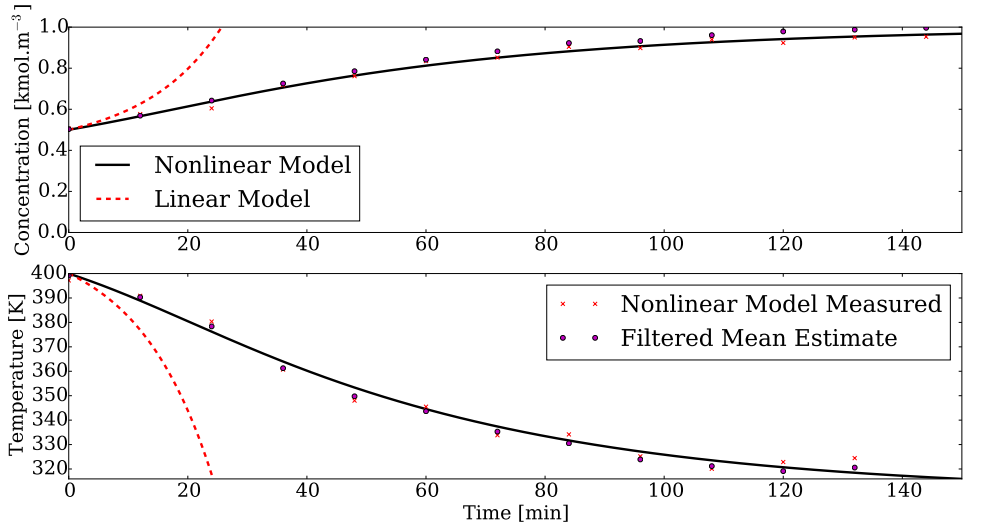

Figure 25: Kalman Filter superimposed on the time series evolution of the CSTR with initial condition $(0.50, 400)$ and measuring both temperature and concentration.

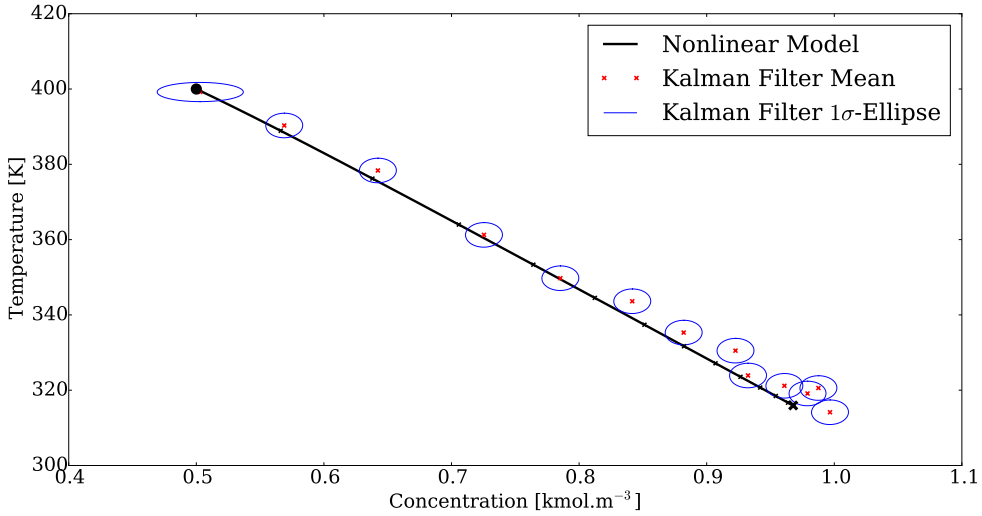

Figure 26: State space diagram of the CSTR with mean and one-sigma ellipses superimposed thereupon. Both concentration and temperature are measured.

Comparing Figures 24 and 26 we see that by incorporating the state measurement the state estimation is much more accurate. 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.

6 Inference using Nonlinear Models

In this section we still consider probabilistic graphical models of the form shown in Figure 27. 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.

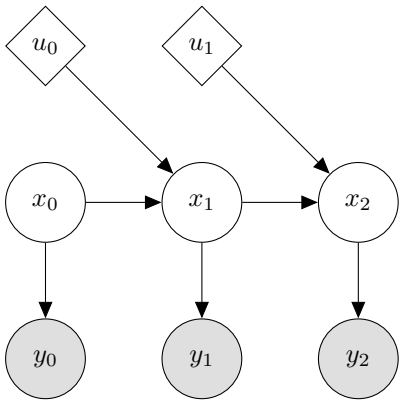

Figure 27: Graphical model of this section

We again assume that the transition and emission functions are time invariant. The state space model is now of the form (76).

$$\begin{aligned} x_{t+1} &= f(x_t, u_t, w_{t+1}) \\ y_{t+1} &= g(x_{t+1}, v_{t+1}) \end{aligned} \tag{76}$$

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.

6.1 Sequential Monte Carlo Methods

Many approximate inference techniques exist in literature, the most notable ones include Gaussian Sum Filters [22] and Particle based methods. We shall focus only on Sequential Monte Carlo (SMC) 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.

SMC methods are a general class of Monte Carlo methods which sample sequentially from the growing target distribution $\pi_t(x_{0:t})$. By only requiring that γ_t be known point-wise we have the framework of SMC methods as shown in (77). Note that Z_t is some normalisation

constant [14].

$$\begin{aligned} \pi_t(x_{0:t}) &= \frac{\gamma_t(x_{0:t})}{Z_t} \\ Z_t &= \int_{x_{0:t}} \gamma_t(x_{0:t}) \end{aligned} \tag{77}$$

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 $\hat{\pi}_t(x_{0:t})$ as shown in (78).

$$\hat{\pi}_t(x_{0:t}) = \frac{1}{N} \sum_{i=1}^{N} \delta(X^i_{0:t}, x_{0:t}) \tag{78}$$

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 shown in (79).

$$\hat{\pi}_t(x_t) = \frac{1}{N} \sum_{i=1}^{N} \delta(X^i_t, x_t) \tag{79}$$

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 SMC framework (77) we have (80).

$$\begin{aligned} \pi_t(x_{0:t}) &= \frac{w_t(x_{0:t}) q_t(x_{0:t})}{Z_t} \\ Z_t &= \int_{x_{0:t}} w_t(x_{0:t}) q_t(x_{0:t}) \end{aligned} \tag{80}$$

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 (80) we have (81).

$$\begin{aligned} \hat{\pi}_t(x_{0:t}) &= \frac{1}{N} \sum_{i=1}^{N} W^i_t \delta(X^i_{0:t}, x_{0:t}) \\ \hat{Z}_t &= \frac{1}{N} \sum_{i=1}^{N} w_t(X^i_{0:t}) \\ W^i_t &= \frac{w_t(X^i_{0:t})}{\sum_{i=1}^{N} w_t(X^i_{0:t})} \end{aligned} \tag{81}$$

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 $q_t(x_{0:t}) = q_{t-1}(x_{0:t-1})q_t(x_t|x_{0:t-1}) = q_0(x_0)\Pi_{k=1}^t q_k(x_k|x_{0:k-1})$. In this way we only need to sample sequentially at each time step: at time $t = 0$ we sample $X_0^i \backsim q_0(x_0)$, at time $t = 1$ we sample $X_1^i \backsim q_1(x_1|x_0)$ and so we build up $X_{0:t}^i \backsim q_t(x_{0:t})$ factor by factor.

The weights can be written in the form (82).

$$\begin{aligned} w_t(x_{0:t}) &= \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t})} \\ &= \frac{\gamma_{t-1}(x_{0:t-1})}{q_{t-1}(x_{0:t-1})} \frac{\gamma_t(x_{0:t})}{\gamma_{t-1}(x_{0:t-1})q_t(x_t|x_{0:t-1})} \\ &= w_{t-1}(x_{0:t-1})\alpha_t(x_{0:t-1}) \\ &= w_0(x_0)\Pi_{k=1}^t \alpha_k(x_{0:k}) \end{aligned} \tag{82}$$

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 [14].

We overcome this problem by resampling and thus introduce the Sequential Importance Resampling (SIR) 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 $W_{0:t}^i$ to draw samples from $\hat{\pi}_t(x_{0:t})$ [14].

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 SIR algorithm:

SIR 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. 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 $\pi(x_{0:t})$ as shown in (83).

$$\begin{aligned}\hat{\pi}(x_{0:t}) &= \sum_{i=1}^{N} W_t^i \delta(X_{0:t}^i, x_{0:t}) \\ \bar{\pi}(x_{0:t}) &= \frac{1}{N}\sum_{i=1}^{N} \delta(\bar{X}_{0:t}^i, x_{0:t})\end{aligned} \tag{83}$$

The latter approximation represents the resampled estimate and the former represents the sampled estimate [14]. 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 [35].

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 (84).

$$\text{ESS} = \frac{1}{\sum_{i=1}^{N}(W_n^i)^2} \tag{84}$$

If the ESS 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 SMC algorithm as shown below.

Adaptive SIR 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 SMC 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 [14]. 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.

6.2 Particle Filter

We now apply the adaptive SIR 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 as shown in (85).

$$\begin{aligned} w_t(x_{0:t}) &= \frac{\gamma_t(x_{0:t})}{q_t(x_{0:t}|y_{0:t})} \\ &= \frac{p(x_{0:t}, y_{0:t})}{q_t(x_{0:t}|y_{0:t})} \\ &\propto \frac{p(x_{0:t}|y_{0:t})}{q_t(x_{0:t}|y_{0:t})} \\ &\propto \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q_t(x_t|x_{0:t-1}, y_{0:t})} \frac{p(x_{0:t-1}|y_{0:t-1})}{q_{t-1}(x_{0:t-1}|y_{0:t-1})} \\ &= \alpha_t(x_{0:t})w_{t-1}(x_{0:t-1}) \end{aligned} \tag{85}$$

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 (86).

$$\alpha_t(x_{0:t}) = \frac{p(y_t|x_t)p(x_t|x_{t-1})}{q_t(x_t|x_{t-1}, y_t)} \tag{86}$$

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 [21] in the setting of the non-adaptive SIR 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 [35]. 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. 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 [21]. We thus make a final modification to the adaptive SIR 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 $w_0(X_0^i) = p(Y_0^*|X_0^i) = \mathcal{N}(Y_0^*|g(X_0^i), \text{covar}[v_0])$ 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 $X_t^i = f(\bar{X}_{t-1}^i, w_t) \backsim p(x_t|\bar{X}_{t-1}^i)$.

2. Compute the weights $\alpha_t(X_t^i) = p(Y_t^*|X_t^i) = \mathcal{N}(Y_t^*|g(X_t^i), \text{covar}[v_t])$ 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. [21].

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. 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.

6.3 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 methods. Recalling the prediction derivations of the Hidden Markov Model section and the Linear Models section 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 emission function.

Inspecting the bootstrap Particle Filter algorithm presented in the previous subsection 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 understand what we mean.

Particle Prediction Algorithm

1. Sample $X_{t+1}^i = f(\bar{X}_t^i, w_{t+1}) \backsim p(x_{t+1}^i | y_t, \bar{X}_t^i)$
2. Project $X_{t+2}^i = f(X_{t+1}^i, w_{t+2}) \backsim p(x_{t+2}^i | y_t, X_{t:t+1}^i)$
3. Project $Y_{t+2}^i = g(X_{t+2}^i, v_{t+2}) \backsim p(y_{t+2}^i | y_t, X_{t:t+1}^i)$

6.4 Smoothing and Viterbi Decoding

In the context of nonlinear transition and emission 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 [14][22][35][36][3].

6.5 Filtering the CSTR

In this section we apply the Particle Filter to the nonlinear CSTR problem. 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 apply the Particle Filter to the CSTR using the linear model and compare it to the Kalman Filter.

Although it is not necessary we assume that the process and measurement noise is Gaussian with the same distributions as in the previous section. Unless otherwise noted we use the same parameters (e.g. $h = 0.1$) as before. We have used 100 particles to represent the state posterior. In Figure 28 we see the state estimates as a function of time.

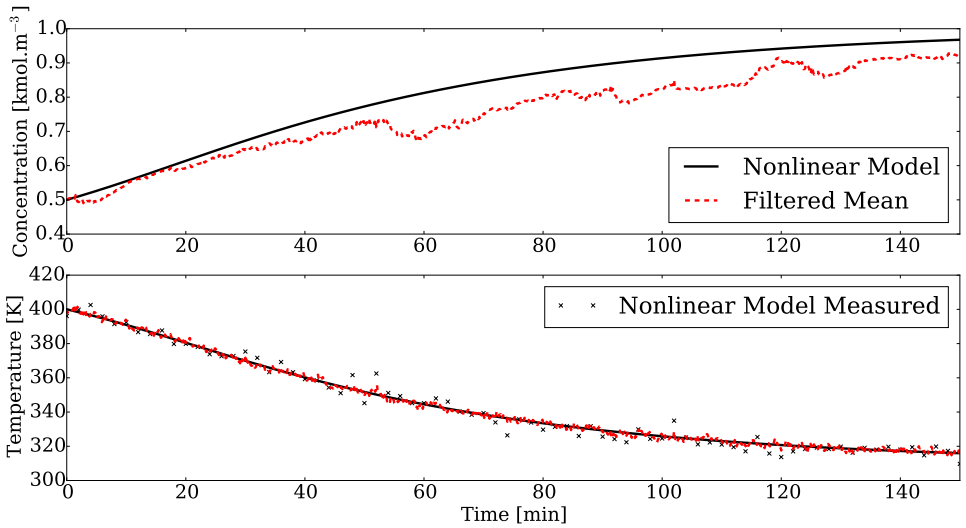

Figure 28: Time series state estimates using the Particle Filter on the nonlinear CSTR model with initial condition $(0.5, 400)$ and measuring only temperature.

The filter 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. This is also reflected in the state space evolution curve in Figure 29.

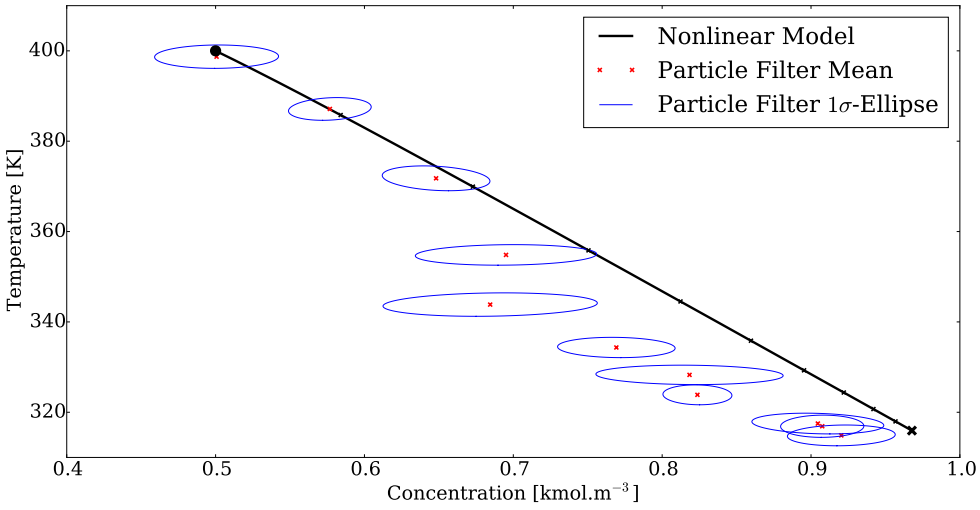

Figure 29: State space evolution of the Particle Filter on the nonlinear CSTR model with initial condition $(0.5, 450)$ and measuring only temperature.

We also see in Figure 29 that the variance of the estimates is quite high (the ellipses are quite big). We expect that by also measuring concentration this will decrease. In Figures 30 and 31 we incorporate a concentration measurement to aid inference.

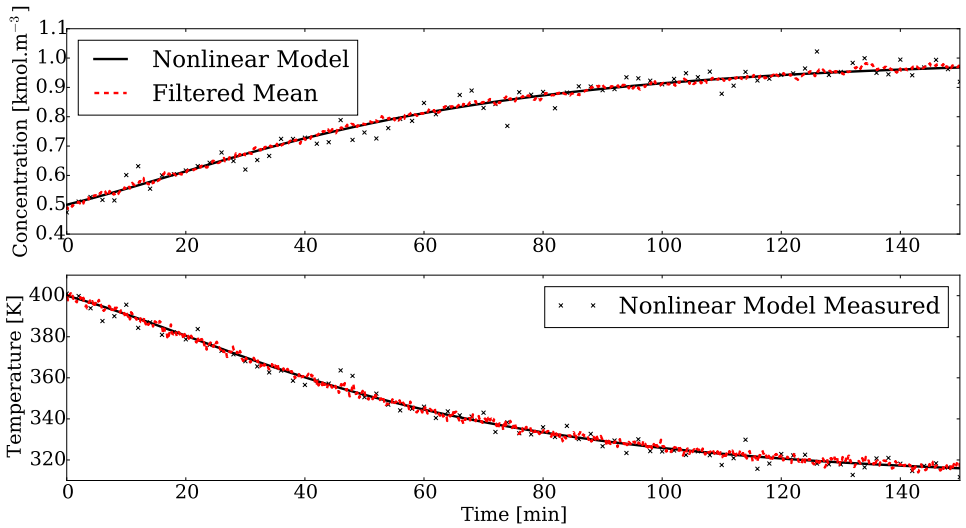

Figure 30: Time series state estimates using the Particle Filter on the nonlinear CSTR model with initial condition $(0.5, 450)$ and measuring both states.

It is clear that that the Particle Filter reliably tracks the state evolution in the presence of plant and measurement noise. We see that by also measuring the concentration the size of the error ellipses decrease in Figure 31.

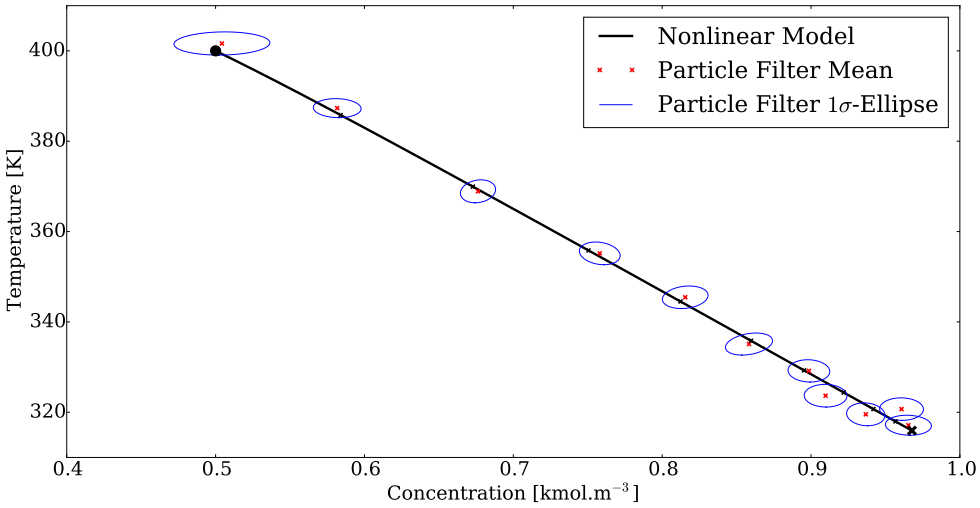

Figure 31: State space evolution of the Particle Filter on the nonlinear CSTR model with initial condition $(0.5, 450)$ and measuring both states.

Finally we compare the Particle Filter to the Kalman filter, using both temperature and concentration measurements and the linear model linearised around the unstable operating point. In Figure 32 we see that both the Particle Filter and the Kalman Filter are able to accurately estimate the posterior state distribution. Note that we have used 500 particles to meaningfully compare the distribution estimates (the Particle Filter should converge to the

Kalman Filter as the number of particles get big).

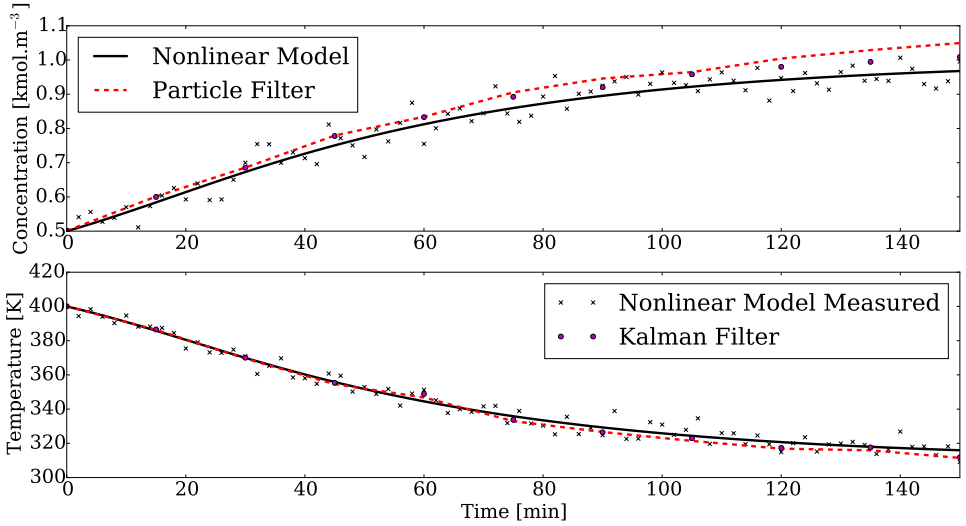

Figure 32: Time series state estimates using the Particle Filter and the Kalman Filter on the linear CSTR model (around the unstable operating point) with initial condition $(0.5, 400)$ and measuring both temperature and concentration.

In Figure 33 we see empirical proof of the assertion that the Kalman Filter is the optimum state estimator for conditionally linear Gaussian systems [1]: the 1σ ellipses for the Kalman Filter are smaller than the corresponding ellipses for the Particle Filter and they track the true states better than the Particle Filter.

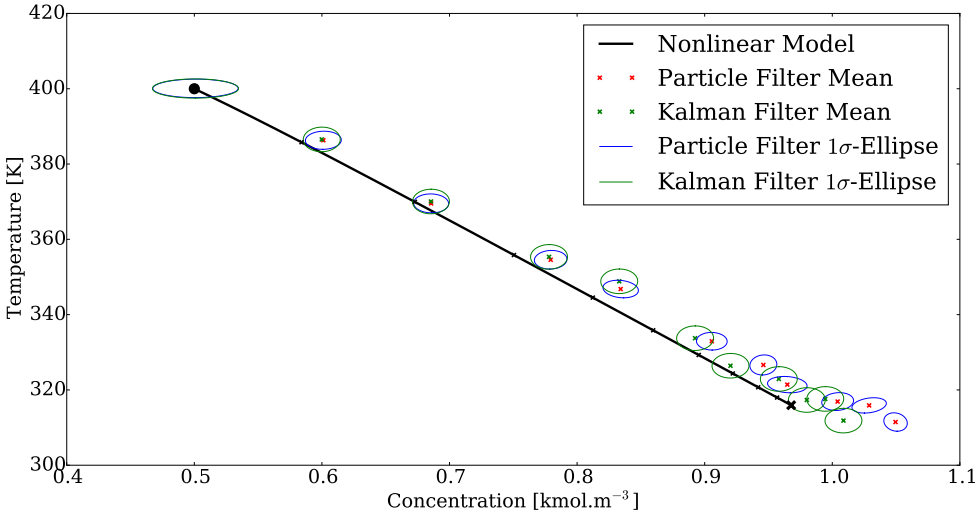

Figure 33: State space evolution of the Particle Filter and the Kalman Filter on the linear CSTR model (around the unstable operating point) with initial condition $(0.5, 400)$ and measuring both temperature and concentration.

It is clear that if one only has access to linear models it makes sense from both a computational

and accuracy point of view to use the Kalman Filter. However, if one needs to perform inference on a system better descibed by a nonlinear model the Particle Filter becomes more useful.

7 Stochastic Linear Control

In this section we consider the stochastic reference tracking problem. It is required to move the states and manipulated variables of the system, shown in (87), to the set point (x_{sp}, u_{sp}) by manipulating the input variables u.

$$\begin{aligned} x_{t+1} &= f(x_t, u_t) + w_{t+1} \\ y_{t+1} &= g(x_{t+1}) + v_{t+1} \end{aligned} \tag{87}$$

We assume uncorrelated zero mean additive Gaussian noise 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 7.1.

Definition 7.1. 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.

In this section 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 shown in (88) and is subject to the same noise as (87).

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t + w_{t+1} \\ y_{t+1} &= Cx_{t+1} + v_{t+1} \end{aligned} \tag{88}$$

We will endeavour to develop predictive controllers using the Graphical Models of Section 5 and 6.

7.1 Current Literature

Linear unconstrained stochastic control subject to white additive Gaussian noise is well studied in literature. The Linear Quadratic Gaussian (LQG) controller introduced in Section 2.4 is one of the most fundamental results in Optimal Control Theory [10]. A significant drawback of the LQG controller, and by extension the LQR controller, is that it is inherently unconstrained.

Conventional deterministic MPC is very well studied in literature [38] and can be seen as the constrained generalisation of the LQR controller. 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 somewhat more tractable from a 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 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 distributions are approximated by particles (much like the Particle Filter of Section 6) 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.

Indeed, this is the approach taken by [6]. They attempt to solve the stochastic MPC problem with stochastic (chance) constraints and variables by approximating the current and predicted distributions with particles. Referring back to Section 6.3 we see that this is indeed possible. By using the particle approach to model distributions it is possible to convert the resultant 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. Their algorithm is appealing because it is not necessary to assume Gaussian distributions. However, it is possible that the algorithm could become computationally intractable due to the integer constraints which are used to approximate the chance constraints: it is necessary to include an integer variable for each particle at each time step in the prediction horizon. For large problems with long prediction horizons this can be problematic.

The approach taken by [29] 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. The approach taken by [4] uses a randomized optimisation algorithm in concert with the empirical mean of the variables. A penalty method is used to enforce constraints. The latter paper takes a step in the direction of simplifying the problem without sampling.

In the paper [28] 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 [32] the feasibility of stochastically constrained predictive control is considered. An algorithm enforcing joint chance constraints and recursive feasibility is discussed using a risk allocation approach.

While [40] 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 [43] and [42] an ellipsoidal approximation is used to ensure constraint satisfaction. Using the ellipsoidal approximations the stochastic optimisation problem can transformed into a second order conic optimisation problem. The approach of using confidence ellipsoids is refined in [7].

Although [45] and [46] primarily deal with a univariate problem they show that if the under-

lying system is linear and Gaussian it is possible to manipulate the constrained stochastic problem into a deterministic problem without any approximations. Their analysis allows the stochastic objective function and constraints to be rewritten as deterministic expressions using the properties of conditionally linear Gaussian distributions.

In this work we show that by starting the MPC design from the graphical model shown in Figure 27 it is possible to easily and intuitively re-derive the LQG controller. During that derivation we naturally arrive at the conclusions reached by [45] and [46]. Next we generalise our method to incorporate stochastic constraints. We use a method related to the ellipsoidal approximation technique to handle the stochastic constraints. We show that our method replaces the chance constraints with linear constraints. This is highly desirable because it allows one to rewrite the full stochastic MPC problem as a conventional deterministic Quadratic Programming (QP) MPC problem. All of these results are underpinned by the realisation that Graphical Models, as used in the field of Machine Learning, are closely related to predictive control.

7.2 Unconstrained Stochastic Control

Our goal is to solve the problem in (89) given the current state estimate x_0. If the system is controllable and linear the Stochastic Reference Tracking Goal will be satisfied.

$$\begin{aligned} &\min_{\mathbf{u}} J_{LQG}(x_0, \mathbf{u}) = \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t \\ &\text{and } y_t = Cx_t + v_t \end{aligned} \tag{89}$$

Note that the future inputs $\mathbf{u} = (u_0, u_1, ..., u_{T-1})$ are denoted in boldface to emphasise that it could be a vector of vectors. Inspecting (89) we see that this is none other than the LQG control problem of Section 2.4.3. Therefore we know what the optimal solution should look like.

We start out analysis using the results of Section 5. We immediately realise that the optimal linear state estimator is the Kalman Filter. We assume that at every sequential time step we have the current state estimate, supplied by the Kalman Filter, and denote this by x_0. Since we are using the Kalman Filter the mean and covariance of the state estimate is well defined.

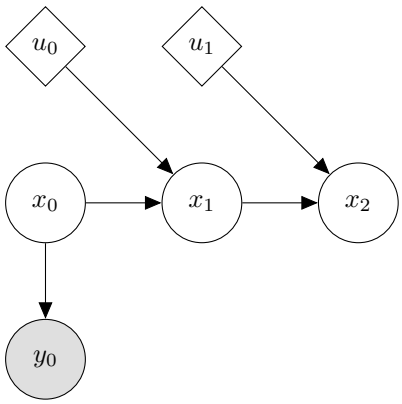

Figure 34: Graphical model of this section

Inspecting Figure 34 we note that the state prediction equations derived in Subsection 5.2 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 7.1. We will use this to link the predictive controller derived using the results of Section 5 to the LQR controller derived in Section 2.4.1. We provide two proofs, the second of which is more general than the first.

Theorem 7.1. Optimisation Equivalence Suppose we have two real valued 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 $k \geq 0$ 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 $g(x_0, \mathbf{u})$ also minimises $f(x_0, \mathbf{u})$.

Proof. This proof only holds over functions which are at least twice differentiable and convex. 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 [19] 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. □

Proof. This proof hold over differentiable and non-differentiable objective functions which are not necessarily convex. Suppose not i.e. there exists $\mathbf{u}_g \in \mathcal{U}$ such that $g(x_0, \mathbf{u}_g) < g(x_0, \mathbf{u})$ $\forall \mathbf{u} \in \mathcal{U}$ but $f(x_0, \mathbf{u}_g) \not< f(x_0, \mathbf{u})$ $\forall \mathbf{u} \in \mathcal{U}$. This implies that for $\mathbf{u}_f \in \mathcal{U}$ the global minimiser of f we have $f(x_0, \mathbf{u}_f) \leq f(x_0, \mathbf{u}_g)$.

Consider the case where $f(x_0, \mathbf{u}_f) = f(x_0, \mathbf{u}_g)$. This implies that both $\mathbf{u}_f$ and $\mathbf{u}_g$ are global minimisers of f and contradicts our assumption that the global minimiser is unique.

Consider the case where $f(x_0, \mathbf{u}_f) < f(x_0, \mathbf{u}_g)$. Since $g(x_0, \mathbf{u}) + k = f(x_0, \mathbf{u})$ $\forall \mathbf{u} \in \mathcal{U}$ this

implies that $g(x_0, \mathbf{u}_f) < g(x_0, \mathbf{u}_g)$. But this contradicts our assumption that $\mathbf{u}_g$ is the global minimiser of g.

It must then hold that $f(x_0, \mathbf{u}_g) < f(x_0, \mathbf{u})\ \forall \mathbf{u} \in \mathcal{U}$. Therefore the global minimiser $\mathbf{u}_g$ of $g(x_0, \mathbf{u})$ also minimises $f(x_0, \mathbf{u})$. □

Now we are in a position to show the equivalence between the LQR control problem and the LQG control problem using the results of Section 5. Theorem 7.2 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 [45] and [46] in an intuitively simple manner.

Theorem 7.2. LQR and LQG Objective Function Difference Consider the LQR and LQG Objective Functions in (90) and (91) respectively.

$$J_{LQR}(x_0, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N \tag{90}$$

$$\text{with } x_{t+1} = Ax_t + Bu_t$$

$$J_{LQG}(x_0, \mathbf{u}) = \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \tag{91}$$

$$\text{with } x_{t+1} = Ax_t + Bu_t + w_{t+1}$$

Suppose x_0 is the state estimate supplied by the Kalman Filter 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 $J_{LQR}(x_0, \mathbf{u}) + \frac{1}{2}\sum_{k=0}^{N} \text{tr}(Q\Sigma_k) = J_{LQG}(x_0, \mathbf{u})$ where $\Sigma_{t+1} = W + A\Sigma_t A^T$ and Σ_0 is the covariance matrix of the current state given by the Kalman Filter.

Proof. Expanding the LQG objective function and noting that $\mathbf{u}$ is deterministic we have (92). Note that the conditional expectations in the expansion originate from the graphical model in Figure 22 (due to the first order Markov assumption).

$$\begin{aligned} J_{LQG}(x_0, \mathbf{u}) &= \frac{1}{2}\mathbb{E}\left[x_0^T Q x_0 + u_0^T R u_0\right] + \frac{1}{2}\mathbb{E}\left[x_1^T Q x_1 + u_1^T R u_1 | x_0\right] + ... \\ &+ \frac{1}{2}\mathbb{E}\left[x_{N-1}^T Q x_{N-1} + u_{N-1}^T R u_{N-1} | x_{N-2}\right] + \frac{1}{2}\mathbb{E}\left[x_N^T P_f x_N | x_{N-1}\right] \\ &= \frac{1}{2}\mathbb{E}\left[x_0^T Q x_0\right] + \frac{1}{2}u_0^T R u_0 + \frac{1}{2}\mathbb{E}\left[x_1^T Q x_1 | x_0\right] + \frac{1}{2}u_1^T R u_1 + ... \\ &+ \frac{1}{2}\mathbb{E}\left[x_{N-1}^T Q x_{N-1} | x_{N-2}\right] + \frac{1}{2}u_{N-1}^T R u_{N-1} + \frac{1}{2}\mathbb{E}\left[x_N^T P_f x_N | x_{N-1}\right] \end{aligned} \tag{92}$$

We know that $x_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ because the current state estimate comes from the Kalman Filter. This means that we can evaluate the first expected value in (92) using Theorem 2.5 as shown in (93).

$$\mathbb{E}\left[x_0^T Q x_0\right] = \text{tr}(Q\Sigma_0) + \mu_0^T Q \mu_0 \tag{93}$$

Now we turn out attention to the second expected value in (92). First note that because we have x_0 and $\mathbf{u}$ we can use the result from Section 5.2 to predict (optimally) 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 2.5 as before we have (94).

$$\mathbb{E}\left[x_1^T Q x_1 | x_0\right] = \text{tr}(Q\Sigma_1) + \mu_1^T Q \mu_1 \tag{94}$$

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 (95).

$$\begin{aligned} &J_{LQG}(x_0, \mathbf{u}) = \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q \mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N + \frac{1}{2}\sum_{k=0}^{N}\text{tr}(Q\Sigma_k) \\ &\text{with } \mu_{t+1} = A\mu_t + Bu_t \\ &\text{and } \Sigma_{t+1} = W + A\Sigma_t A^T \end{aligned} \tag{95}$$

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 $\mathbf{u}$. □

Finally we combine Theorem 7.1 and 7.2 to produce Theorem 7.3 which is the main result of this subsection.

Theorem 7.3. 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 shown in (96).

$$\begin{aligned} &\min_{\mathbf{u}} V(x_0, \mathbf{u}) = \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\ &\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t \\ &\text{and } y_t = Cx_t + v_t \end{aligned} \tag{96}$$

We assume that we have the Kalman Filter state estimate for x_0. We use Theorem 7.2 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 7.1 to prove that we only need to solve for the optimal controller input $\mathbf{u}^0$ using the LQR objective function. Thus we can use Theorem 2.11 to find $\mathbf{u}$.

As we have mentioned before, the Separation Theorem 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 7.3 is confirmed by this result. 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 7.3 to the infinite horizon case as shown in Theorem 7.4.

Theorem 7.4. Solution of the Infinite Horizon LQG control problem If the linear model of (88) is stable then, using, with some minor adjustments, Theorems 7.1 and 7.2 it is possible to show that the infinite horizon LQG problem is solved in a similar manner: the Kalman Filter 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 $\frac{1}{2}\sum_{k=0}^{N}\mathrm{tr}(Q\Sigma_k)$. If it is unbounded the optimisation minimum will tend to 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). Note that $||\cdot||$ is some matrix norm. It can be shown that if the eigenvalues of A are less than unity i.e. the linear model is stable, then $||A\Sigma_{t+1}A^T|| \leq ||A\Sigma_{t+1}A^T||$ which implies that $\frac{1}{2}\sum_{k=0}^{N}\mathrm{tr}(Q\Sigma_k)$ is bounded and the optimisation is reasonable.

7.3 Constrained Stochastic Control

The goal of this section is to solve the stochastic constrained optimisation problem shown in (97). We assume that the underlying system is linear and the probability distributions are Gaussian[5]. We also restrict our analysis to affine constraints.

$$\begin{aligned}
&\min_{\mathbf{u}} \mathbb{E}\left[\frac{1}{2}\sum_{k=0}^{N-1}\left(x_k^T Q x_k + u_k^T R u_k\right) + \frac{1}{2}x_N^T P_f x_N\right] \\
&\text{subject to } x_{t+1} = Ax_t + Bu_t + w_t \\
&\text{and } y_t = Cx_t + v_t \\
&\text{and } \mathbb{E}[d^T x_t + e] \geq 0 \;\forall\; t = 1, ..., N \\
&\text{and } \Pr(d^T x_t + e \geq 0) \geq p \;\forall\; t = 1, ..., N
\end{aligned} \tag{97}$$

It might seem that the last constraint is a duplicate of the preceding one. Closer inspection reveals their different character. The first inequality constraint requires that the predicted states satisfy the constraint "on average" while the second inequality constraint requires that the predicted states jointly satisfy the constraint with at least some probability p. The reason both constraints are required will become clear later.

Theorem 7.5 succinctly shows that it is simple to convert the first stochastic constraint in (97) to a linear deterministic constraint.

Theorem 7.5. Affine Expected Value Constraints Suppose we have a stochastic variable x with a known Gaussian distribution. 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.

[5]From the results of Section 5 the probability distributions will be Gaussian if the system dynamics are linear. However, it is well known [31] 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.

Proof. We know that x is a Gaussian stochastic variable. By the results of Section 2.5 we know that $\mathbb{E}[d^T x + e] = d^T \mu + e$. This immediately implies the Theorem. □

Theorem 7.6 is a necessary step before we can convert the last stochastic inequality constraint of (97) into a nonlinear deterministic constraint.

Theorem 7.6. 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 $b^T x + c = 0$ is given by $\frac{(b^T y + c)^2}{b^T S b}$.

Proof. It is natural to formulate Theorem 7.6 as an optimisation problem as shown in (98).

$$\begin{aligned} &\min_x \ (x-y)^T S^{-1}(x-y) \\ &\text{subject to } b^T x + c = 0 \end{aligned} \tag{98}$$

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 [19] we have the system of equations (99)

$$\begin{aligned} 2S^{-1}x - 2S^{-1}y + \lambda b &= 0 \\ b^T x + c &= 0 \end{aligned} \tag{99}$$

This can be rewritten in block matrix form as shown in (100).

$$\begin{pmatrix} 2S^{-1} & b \\ b^T & 0 \end{pmatrix} \begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} 2S^{-1}y \\ -c \end{pmatrix} \tag{100}$$

The special structure of the left hand side matrix in (100) allows us to analytically compute the inverse (see Theorem 2.13 in Section 2.5) as shown in (101).

$$\begin{pmatrix} 2S^{-1} & b \\ b^T & 0 \end{pmatrix}^{-1} = \begin{pmatrix} \frac{1}{2}S(I - \frac{bb^T S}{b^T S b}) & \frac{Sb}{b^T S b} \\ \frac{b^T S}{b^T S b} & -\frac{2}{b^T S b} \end{pmatrix} \tag{101}$$

To find the arguments which satisfy (99) we solve (102) which is equivalent to solving the system of linear equations in (100).

$$\begin{pmatrix} \frac{1}{2}S(I - \frac{bb^T S}{b^T S b}) & \frac{Sb}{b^T S b} \\ \frac{b^T S}{b^T S b} & -\frac{2}{b^T S b} \end{pmatrix} \begin{pmatrix} 2S^{-1}y \\ -c \end{pmatrix} = \begin{pmatrix} S(I - \frac{bb^T S}{b^T S b})S^{-1}y - c\frac{Sb}{b^T S b} \\ 2(\frac{b^T S}{b^T S b} + \frac{c}{b^T S b}) \end{pmatrix} \tag{102}$$

Therefore, the arguments which minimise (98) are $x^* = S(I - \frac{bb^T S}{b^T S b})S^{-1}y - c\frac{Sb}{b^T S b}$. Substituting

this into the objective function we have (103).

$$
\begin{aligned}
&(x^* - y)^T S^{-1} (x^* - y) \\
&= \left(S \left(I - \frac{bb^T S}{b^T S b} \right) S^{-1} y - c \frac{Sb}{b^T S b} - y \right)^T S^{-1} \left(S \left(I - \frac{bb^T S}{b^T S b} \right) S^{-1} y - c \frac{Sb}{b^T S b} - y \right) \\
&= \left(\frac{Sbb^T y}{b^T S b} + c \frac{Sb}{b^T S b} \right)^T S^{-1} \left(\frac{Sbb^T y}{b^T S b} + c \frac{Sb}{b^T S b} \right) \\
&= \frac{(Sb)^T}{b^T S b} \left(b^T y + c \right)^T S^{-1} \frac{Sb}{b^T S b} \left(b^T y + c \right) \\
&= \frac{b^T S}{b^T S b} \left(b^T y + c \right)^T \frac{b}{b^T S b} \left(b^T y + c \right) \\
&= \left(b^T y + c \right)^T \frac{b^T S b}{(b^T S b)^2} \left(b^T y + c \right) \\
&= \frac{(b^T y + c)^T (b^T y + c)}{b^T S b} \\
&= \frac{(b^T y + c)^2}{b^T S b}
\end{aligned}
\tag{103}
$$

We can conclude that the shortest squared Mahalanobis Distance between a point y and the constraint of (98) is $\frac{(b^T y + c)^2}{b^T S b}$. □

In Theorem 7.7 we apply Theorem 7.6 to convert the stochastic constraints into nonlinear deterministic constraints.

Theorem 7.7. Gaussian Affine Chance Constraints If the underlying distribution of a random variable x is Gaussian then the chance constraint $\Pr(d^T x + e \geq 0) \geq p$ can be rewritten as the deterministic constraint $\frac{(d^T \mu + e)^2}{d^T S d} \geq k^2$ 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 $\Pr(\mathcal{X} \leq k^2) = p$.

Proof. Intuitively Theorem 7.7 posits that if the shortest squared Mahalanobis distance is further away than some threshold k^2 the chance constraint $\Pr(dx + e \geq 0) \geq p$ will be satisfied. 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 [39] 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 $d^T z + e = 0$ 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. This implies, with at least a probability of p, that the chance constraint will not be violated because the "confidence ellipse" does not intersect the constraint. See Figure for a diagram of the principle behind Theorem 7.7 [12]. **insert picture** □

It is interesting to note the striking similarity between the work in [43] and [42] and Theorem 7.7 given that we started analysing the problem within the framework of Graphical Models.

In Theorem 7.8 we combine and elaborate on the work of [45], [43] to yield a QP MPC which exactly satisfies the stochastic MPC in (97) given the assumptions. Note that the dimensionality of the problem is arbitrary.

Theorem 7.8. 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 (97) as a standard deterministic QP MPC problem shown in (104).

$$\begin{aligned}
&\min_{\mathbf{u}} \frac{1}{2}\sum_{k=0}^{N-1}\left(\mu_k^T Q\mu_k + u_k^T R u_k\right) + \frac{1}{2}\mu_N^T P_f \mu_N + \frac{1}{2}\sum_{k=0}^{N}\mathrm{tr}(Q\Sigma_k)\\
&\text{subject to } \mu_{t+1} = A\mu_t + Bu_t\\
&\text{and } \Sigma_{t+1} = W + A\Sigma_t A^T\\
&\text{and } d^T\mu_t + e \geq 0 \;\forall\; t = 1, ..., N\\
&\text{and } d^T\mu_t + e \geq k\sqrt{d^T\Sigma_t d} \;\forall\; t = 1, ..., N
\end{aligned} \tag{104}$$

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.

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 7.2 the objective function and equality constraints follow. By Theorem 7.5 the first inequality constraint follows. The last inequality constraint follows from Theorem 7.7 as shown in (105).

$$\begin{aligned}
\frac{(d^T\mu + e)^2}{d^T S d} \geq k^2 &\implies (d^T\mu + e)^2 \geq k^2 d^T S d\\
&\implies d^T\mu + e \geq k\sqrt{d^T S d}
\end{aligned} \tag{105}$$

The first line of (105) follows because S is positive semi-definite and $d \neq 0$ (otherwise it would not be a constraint), therefore it can be multiplied over the inequality sign. The second line follows because of the first inequality constraint in (104): we know that $d^T\mu + e \geq 0$ and therefore we can square root both sides of the inequality constraint. □

The beauty of Theorem 7.8 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 last inequality constraint to your existing MPC formulation. Most practical MPCs will have some form of state estimation and thus no new parameters are introduced either. Finally, 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.

7.4 Reference Tracking

So far we have only dealt with controllers which would drive the system to the origin. The more general situation we are interested in is arbitrary reference point tracking. Fortunately, Section 2.4.2 applies without modification because we managed to cast the Stochastic MPC problem into a deterministic MPC problem.

7.5 Linear System

7.6 Nonlinear System

8 Inference using Linear Hybrid Models

In this section we generalise the graphical models of the previous sections as shown in Figure 35. We include the discrete random variables, $s_1, s_2, ...$ where each variable has N 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 N 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 [35].

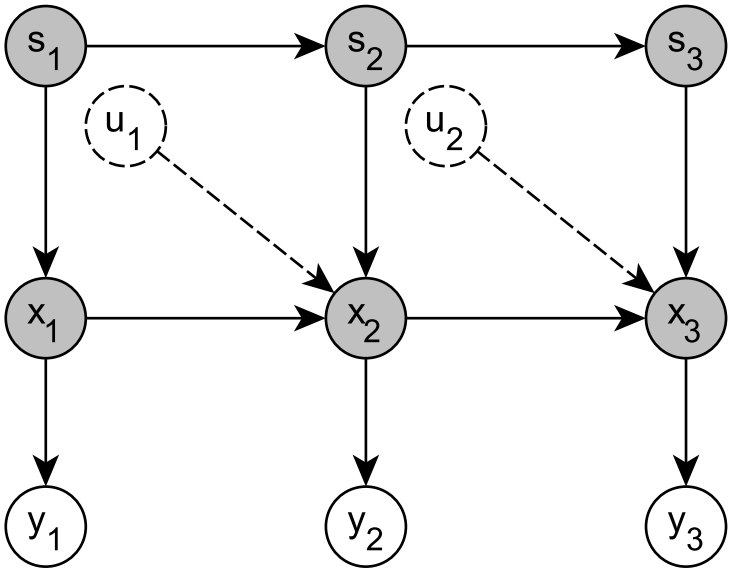

Figure 35: Graphical model of this section

One of the benefits of combining discrete switching variables with linear dynamical models is that it allows us to model nonlinear processes with linear models. Intuitively, we can glue together linear models which each describe a nonlinear model in some region and use the switch to determine which one 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 $s_t \in (1, 2, ..., N)$ denote a discrete, time homogeneous N state first order Markov chain with transition matrix P as discussed in the Hidden Markov Model section. Let each state $s_t = i$ be associated with a parameter set $(A_i, B_i, b_i, Q_i, C_i, R_i)$ used to evaluate the dynamical model shown in (106). If s_t were observed then (106) would simplify to a set of linear latent dynamical systems we could perform inference on using the methods investigated in the Linear Models section. However, we assume that s_t is a hidden random variable.

$$\begin{aligned} x_{t+1} &= A_i x_t + B_i u_t + w_{t+1} \text{ with } \mathcal{N}(w_{t+1}|0, W_i) \\ y_{t+1} &= C_i x_{t+1} + v_{t+1} \text{ with } \mathcal{N}(v_{t+1}|0, V_i) \end{aligned} \tag{106}$$

To fully specify the system we also require the prior distributions $p(s_1)$ and $p(x_1|s_1)$ as well as the transition matrix P. For the purposes of this dissertation we calculate the transition matrix based on the Euclidean distance between the linearisation points of the linear models. Suppose there are N linear models. The transition from the model to itself is $\frac{N}{\sum_{i=1}^{N} i}$. The transition from the model to the next closest model is $\frac{N-1}{\sum_{i=1}^{N} i}$. Continuing in this way we can construct the stochastic matrix P.

8.1 Exact Filtering

The switching variables $(s_1, s_2, ...)$ are discrete random variables exactly like the ones seen in the Hidden Markov Model section. There we derived recursive analytic expressions for inference which were computationally inexpensive. The stochastic linear dynamical system ($x_1, x_2, ...$ and $y_1, y_2, ...$) is exactly the same as the section on Linear Models. 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 (106) can be formulated in a computationally feasible manner.

Unfortunately, it can be shown that this is not possible in general [27][34] 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 N Kalman Filter models (due to the linear assumption and the N switching indices). At time step two the system is described by a weighted set of N^2 Kalman Filter models. Continuing in this manner we see that at time step t the memory requirement is N^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 [2] and Particle Filtering based methods (specifically the Rao-Blackwellisation approach, see [9][15]). 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 the Particle based methods because it can be extended to nonlinear systems with ease.

8.2 Rao-Blackwellised Particle Filter

It is our objective to find the joint posterior distribution $p(s_{1:t}, x_{1:t}|y_{1:t})$. This joint posterior admits filtering of Figure 35 if we discard the trajectory and focus only in s_t, x_t. By the chain rule we immediately have that $p(s_{1:t}, x_{1:t}|y_{1:t}) = p(s_{1:t}|y_{1:t})p(x_{1:t}|y_{1:t}, s_{1:t})$. Given $s_{1:t}$ we see that $p(x_{1:t}|y_{1:t}, s_{1:t})$ can be evaluated using the Kalman Filter equations and thus we are only concerned with finding some approximation for $p(s_{1:t}|y_{1:t})$. This is the essence of the Rao-Blackwellised Particle Filter - taking advantage of the Gaussian nature of the system to analytically evaluate a part of the posterior distribution [15].

Using the formulation of the adaptive Sequential Importance Sampling algorithm discussed in the previous section we can apply it to find an approximation of $p(s_{1:t}|y_{1:t})$. We set $\gamma_t(s_{1:t}) = p(s_{1:t}, y_{1:t})$ and $Z_t = p(y_{1:t})$ and then have that $\frac{\gamma_t(s_{1:t})}{Z_t} = p(s_{1:t}|y_{1:t})$ as desired. We then choose our proposal distribution $q_t(s_{1:t}|y_{1:t})$ to be recursive and follow the same procedure as before shown in (107).

$$\begin{aligned} w_t(s_{1:t}) &= \frac{\gamma_t(s_{1:t}, y_{1:t})}{q_t(s_{1:t}|y_{1:t})} \\ &= \frac{p(s_{1:t}, y_{1:t})}{q_t(s_{1:t}|y_{1:t})} \\ &\propto \frac{p(s_{1:t}|y_{1:t})}{q_t(s_{1:t}|y_{1:t})} \\ &\propto \frac{p(y_t|s_t)p(s_t|s_{t-1})}{q_t(s_t|s_{1:t-1}, y_{1:t})} \frac{p(s_{1:t-1}|y_{1:t-1})}{q_t(s_{1:t-1}|y_{1:t-1})} \\ &= \alpha_t(s_{1:t}) w_{t-1}(s_{1:t-1}) \end{aligned} \tag{107}$$

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. $q_t(s_t|s_{1:t-1}, y_{1:t}) = p(s_t|s_{t-1})$. This is suboptimal but easy to sample from [15]. The incremental weight then simplifies to $\alpha_t(s_{1:t}) = p(y_t|s_t)$. We can evaluate this distribution by marginalising out x_t and using the properties of the Gaussian distributions as before (108).

$$\begin{aligned} \alpha_t(s_{1:t}) &= p(y_t|s_t) \\ &= \int_{x_t} p(y_t|x_t, s_t) p(x_t|s_{1:t}, y_{1:t-1}) \\ &= p(y_t|y_{1:t-1}, s_{1:t}) \\ &= \mathcal{N}\left(y_t|C_{s_t} A_{s_t} \mu_{t-1}, C_{s_t}\left(A_{s_t} \Sigma_{t-1} A_{s_t}^T + Q_{s_t}\right) C_{s_t} + R_{s_t}\right) \end{aligned} \tag{108}$$

Where s_t is denotes a specific state of the switching variable [35]. Upon inspection we see that (108) is just the one step ahead prediction likelihood as discussed in the Linear Models section on prediction [35]. Note that we will still need to resample s_t 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 $p(s_t, x_t|y_{1:t}) = \sum_i w_t^i(S_t^i)\delta(S_t^i, s_t)p(x_t|y_{1:t}, S_t^i)$ where S_t^i is the i^{th} particle. 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 $t = 1$:

1. Sample $S_1^i \backsim p(s_1)$ and $\mu_{1|0}^i \backsim p(x_1|s_1)$.
2. Compute the weights $w_1(S_1^i) = p(Y_1^*|S_1^i)$ where Y_1^* is the observation. Normalise $W_1^i \propto w_1(S_1^i)$.

3. Apply the update step of the Kalman Filter to each particle i and associated parameters to find μ_1^i and Σ_1^i.

4. If the number of effective particles is below some threshold apply resampling with roughening $(W_1^i, S_1^i, \mu_1^i, \Sigma_1^i)$ to obtain N equally weighted particles $(\frac{1}{N}, \bar{S}_1^i, \bar{\mu}_1^i, \bar{\Sigma}_1^i)$ and set $(\bar{W}_1^i, \bar{S}_1^i, \bar{\mu}_1^i, \bar{\Sigma}_1^i) \leftarrow (\frac{1}{N}, \bar{S}_1^i, \bar{\mu}_1^i, \bar{\Sigma}_1^i)$ otherwise set $(\bar{W}_1^i, \bar{S}_1^i, \bar{\mu}_1^i, \bar{\Sigma}_1^i) \leftarrow (W_1^i, S_1^i, \mu_1^i, \Sigma_1^i)$

For $t \geq 2$:

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 $(\bar{W}_t^i, \bar{S}_t^i, \bar{\mu}_t^i, \bar{\Sigma}_t^i) \leftarrow (W_t^i, S_t^i, \mu_t^i, \Sigma_1^t)$

8.3 Rao-Blackwellised Particle Prediction

Like the Particle Predictor studied in the previous section, 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.

8.4 Smoothing and Viterbi Decoding

It is also possible to take advantage of the Gaussian structure in Figure 35 to derive a so-called Rao-Blackwellised Smoothing Algorithm. We do not include it here because it is not necessary for the aims of this dissertation. We refer the reader to the relevant literature [9][15].

Viterbi decoding is likewise not within the scope of this dissertation and as such we refer the reader to [35] for more information. Suffice to say, by increasing the complexity of Figure 35 we increase the difficulty of inference in general.

8.5 Filtering the CSTR

We now apply the Rao-Blackwellised Particle Filter to the CSTR introduced earlier. We first perform filtering on the CSTR by only measuring temperature. Since we are using linear models we compare the performance of the Switching Kalman Filter to that of the standard Kalman Filter which only uses one linear model. We then extend our system to measure both temperature and concentration and perform the same comparisons.

We use the same parameters as before and 500 particles unless otherwise stated. We also assume that the measurement and plant noise covariance matrices (V, W) are the same across all the models. To illustrate how the Switching Kalman Filter works we initially only use 3 linear models to perform inference. Figures 36 to 38 show how the filter performs using three linear models corresponding to the nominal operating points for the CSTR. We only measure temperature for the moment.

Figure 36 shows where the linearisation points are with respect to the state space evolution of the system. We expect the Switching Kalman Filter to switch to models in the vicinity of its current position in the state space.

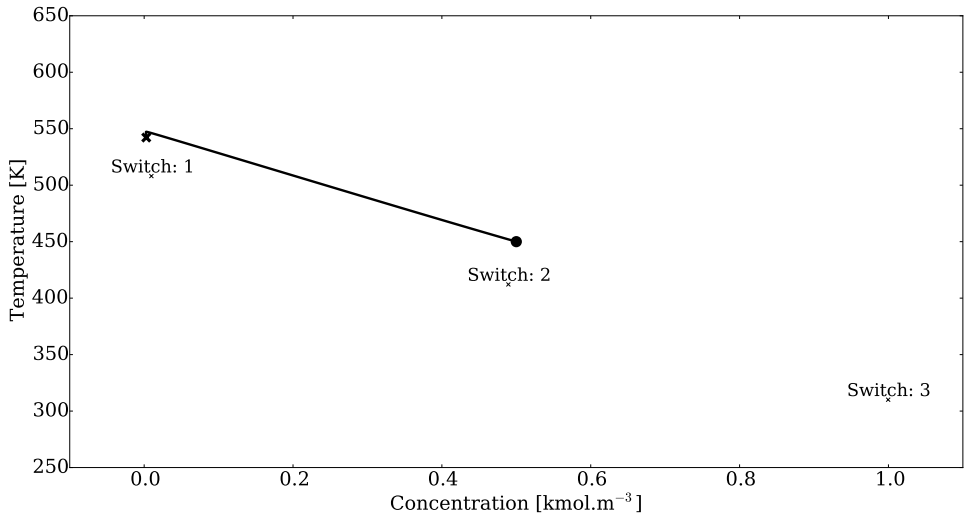

Figure 36: State space evolution and linearisation points of the three linear models used. The system starts at the black circle and ends at the black cross.

Figure 37 shows the weight of each model as time progresses. We see that the second model (S::2 aka switch 2) has the highest weight initially but after about 2 minutes the system switches models so that the first model (S::1 aka switch 1) has the highest weight. Intuitively the system has moved away from the region where the second model describes the system best into a region where the first model is a better descriptor.

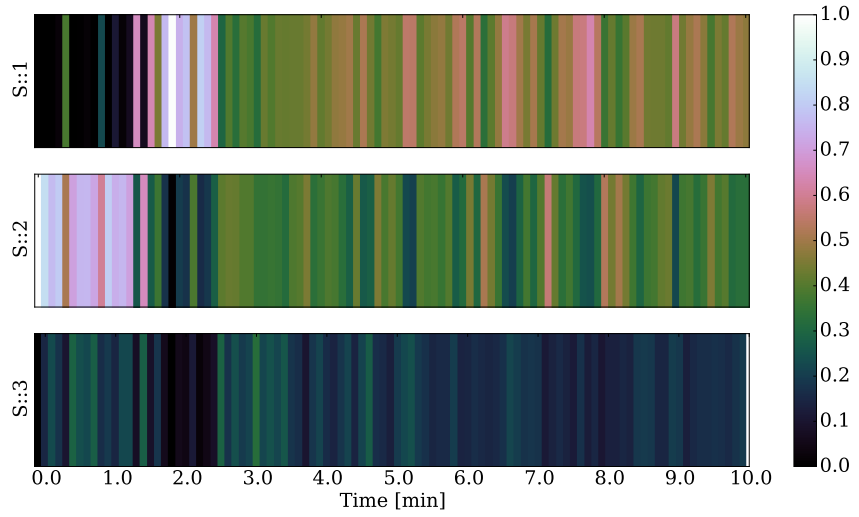

Figure 37: Weight of each switching index as time progresses.

In Figure 38 we see the time series response of the system. Note that only temperature is measured but the system manages to track both states reasonably well. This is a vast improvement over the situation where only one linear model was used as in the section on Linear Models.

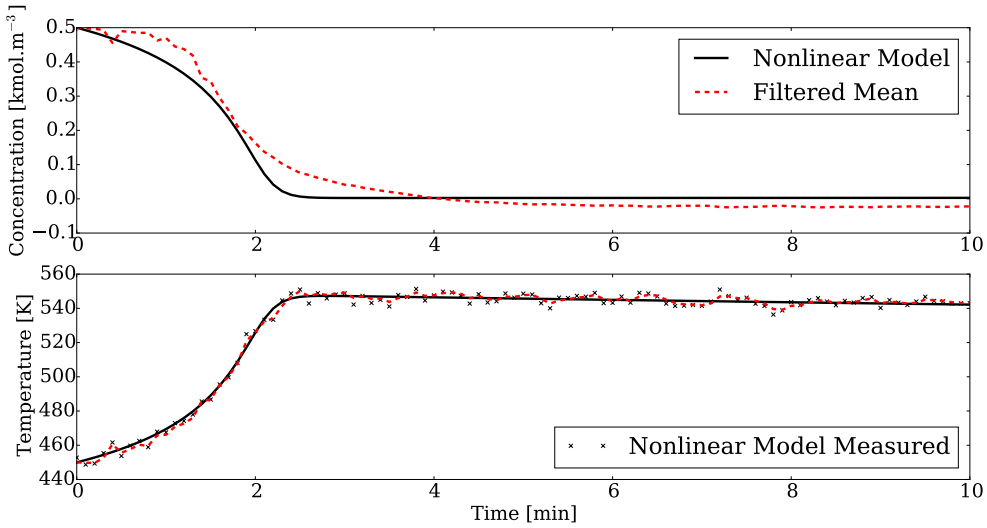

Figure 38: Time series evolution of the states with initial condition $(0.5, 450)$.

Unfortunately, using only these three linear models is not adequate for this system. Since the models are far apart in the state space it sometimes happens that most of the particles are propagated forward using the wrong model. By not measuring a state this problem is exacerbated. Some of these incorrect models may predict the right temperature but the wrong concentration; consequently they are not weeded out by the weighting procedure because the concentration is not observed. The filter then fails to switch and this causes the filter to diverge from the true states.

We illustrate this in Figures 39 to 41. In Figure 39 we see the regions the CSTR traverses in the state space i.e. which switches we expect to be active. We expect switch 2 and 3 to be the most active.

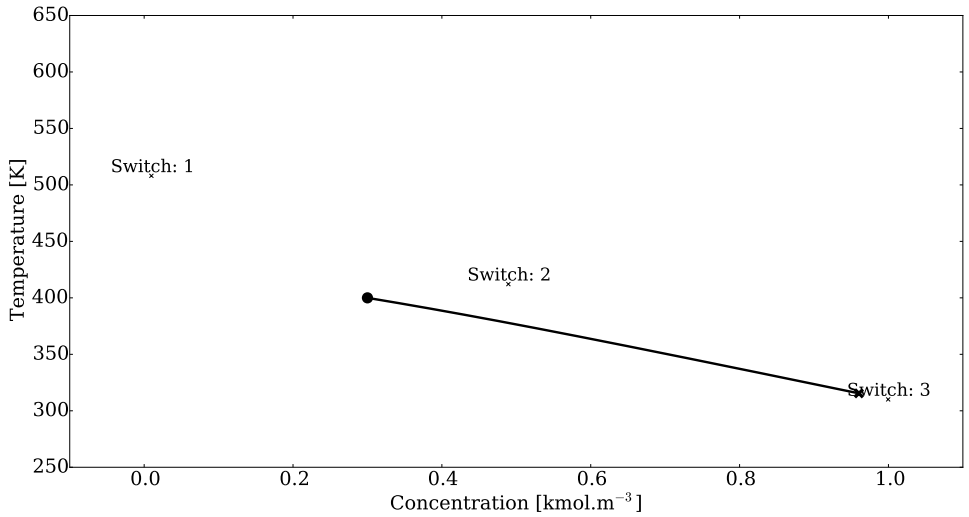

Figure 39: State space evolution and linearisation points of the three linear models used. The system starts at the black circle and ends at the black cross.

Contrary to Figure 39 we see in Figure 40 that switch 1 and 2 are the most active. This causes filter divergence as seen in Figure 41.

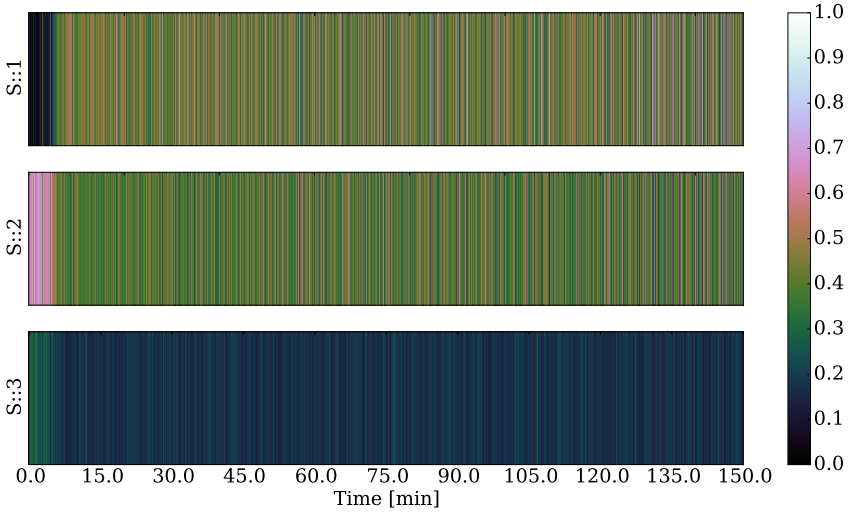

Figure 40: Weight of each switching index as time progresses.

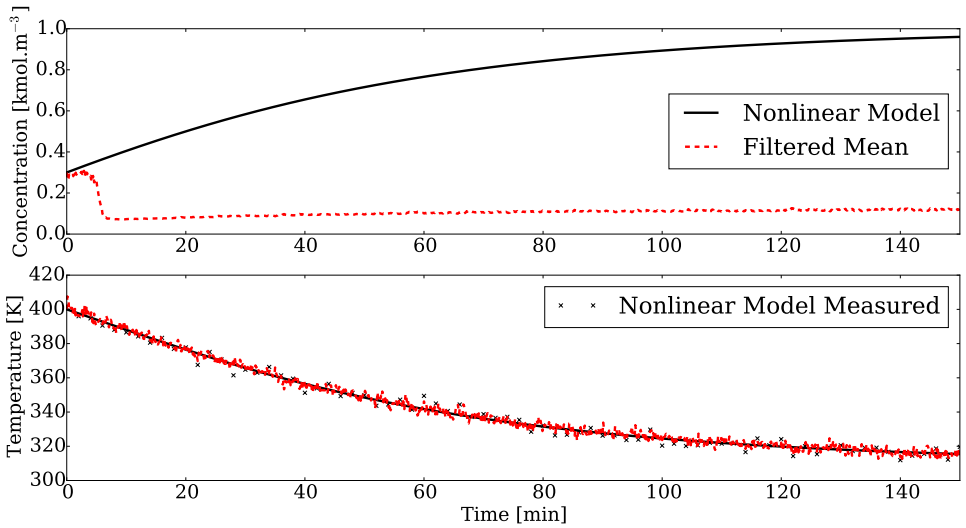

Figure 41: Time series evolution of the states with initial condition $(0.3, 400)$.

Problems like this can be attenuated by using more particles or more linear models (or both). By trial and error we found that 7 linear models and 500 particles describes the system well while only measuring one state.

We now use 7 linear models to perform inference using the same initial conditions as Figure 41 to illustrate how increasing the number of linear models is beneficial to filtering accuracy. Figure 42 indicates that the lower temperature switches (models 1,3,6,7) should be most prominent during the run.

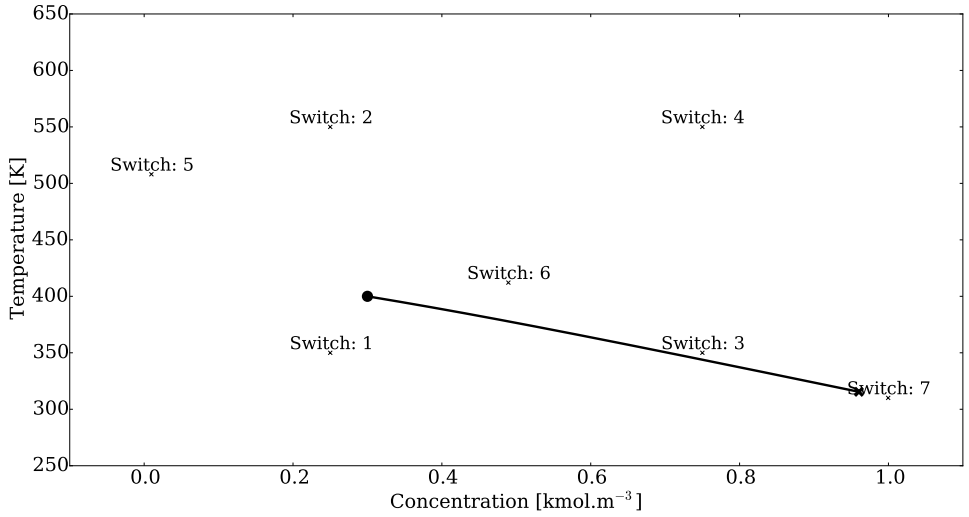

Figure 42: State space evolution and linearisation points of the seven linear models used. The system starts at the black circle and ends at the black cross.

In Figure 43 we see that models 1,3,6 and 7 are indeed the most prominent models. This is reassuring because it indicates the filter correctly "pick" the models which should best

describe the system at each time step.

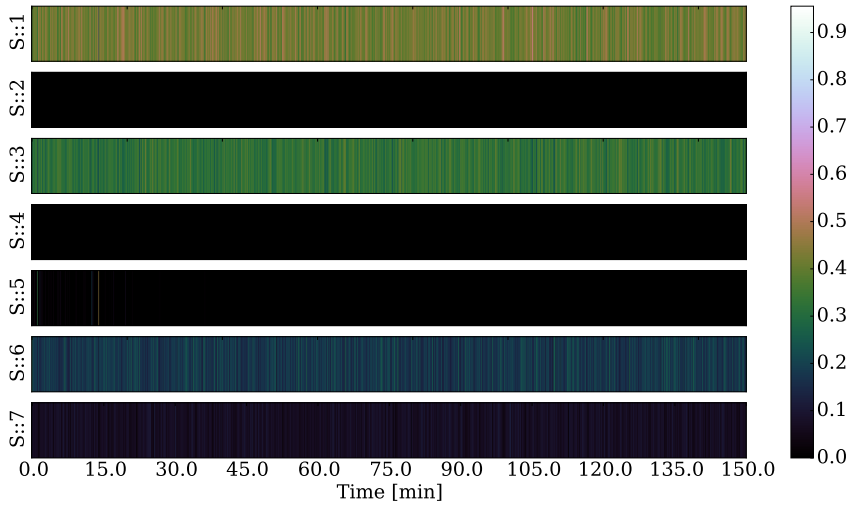

Figure 43: Weight of each switching index as time progresses.

In Figure 44 we see the filtered states as a function of time. We note that the temperature is accurately tracked but that there is almost a 20% error in the concentration estimate. This is due to our single state measurement. Models 1 and 3 both predict the temperature of the system well but only model 3 predicts the concentration well. Since we do not measure concentration the filter has no way of knowing this and therefore includes a significant contribution of model 1 into the posterior state estimate. This causes the error we see in Figure 44.

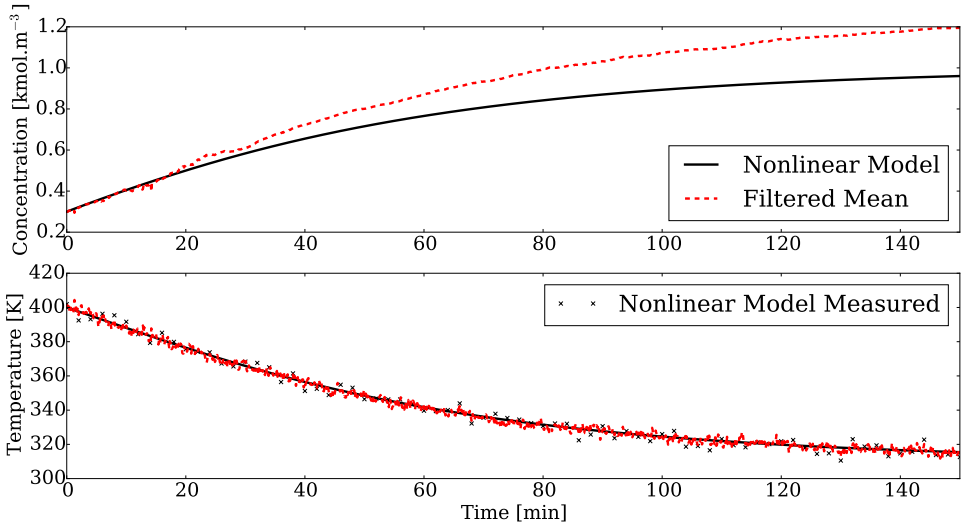

Figure 44: Time series evolution of the states with initial condition $(0.3, 400)$.

The only way to eliminate the type of error we see in Figure 44 is to measure some facet of the concentration. In Figure 45 we see the benefit of using the Switching Kalman Filter even

when only measuring one state. The standard Kalman Filter used here was linearised about the unsteady operating point (model 6).

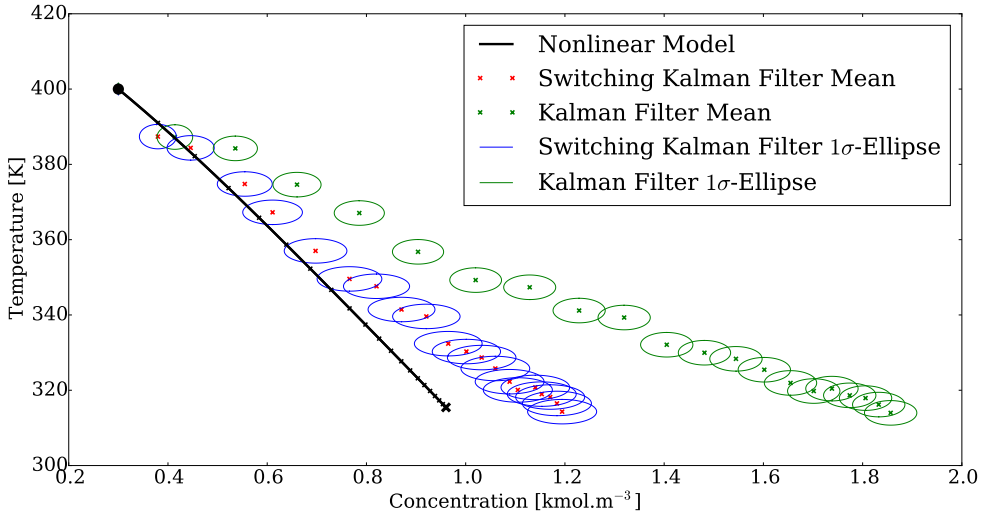

Figure 45: State space evolution of the Switching Kalman Filter and the standard Kalman filter which only uses one linear model.

As discussed before, the standard Kalman Filter poorly estimates the unmeasured state because the linear model is uses is not accurate throughout the state space. However, the Switching Kalman Filter can switch models so that poor models do not form part of its state estimate. Thus it is better able to track the unmeasured state estimate.

As mentioned in the preceding discussion, measuring only one state can cause inaccurate state estimates. It is always better (perhaps not in an economic sense) to measure as much as possible for the purposes of inference. In Figures 46 to 49 we incorporate a concentration measurement to improve inference.

In Figure 46 we see the trajectory of the state evolution in state space. From this we expect models 1,3,6 and 7 to be active.

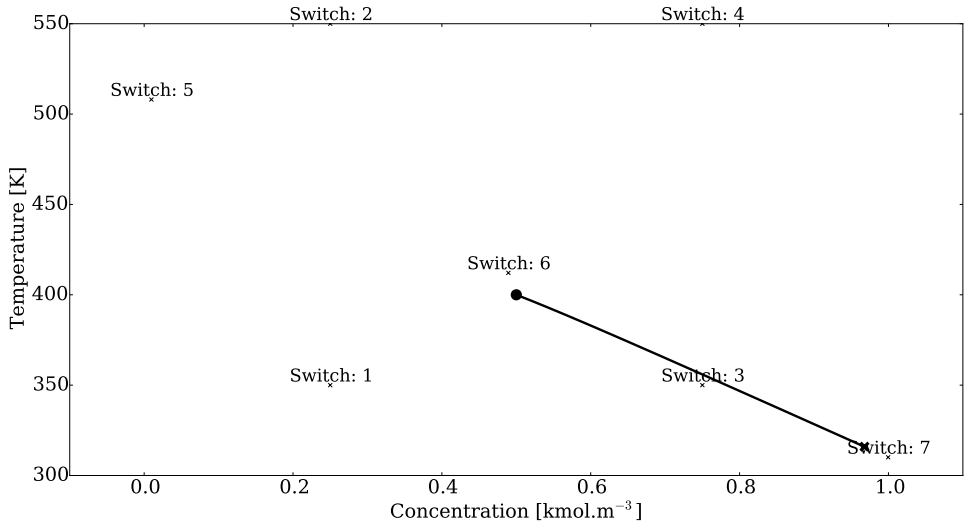

Figure 46: State space evolution and linearisation points of the three linear models used. The system starts at the black circle and ends at the black cross.

In Figure 47 we see that indeed models 1,3,6 and 7 are active. This indicates that the filter is likely to track the states well because the more accurate models are used for inference.

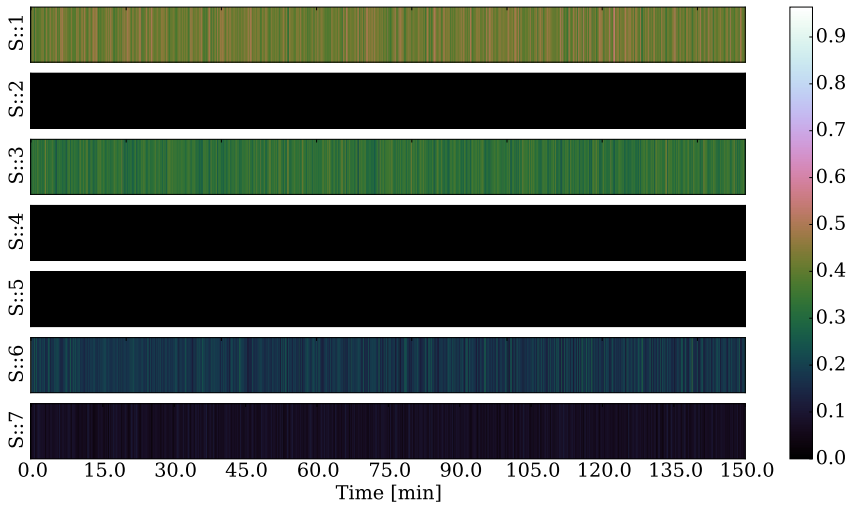

Figure 47: Weight of each switching index as time progresses.

In Figure 48 we see that indeed the filter accurately tracks both concentration and temperature. At this stage it becomes important to question if the Switching Kalman Filter increased the accuracy of the state estimate. After all, the standard Kalman Filter also accurately tracked both states given both measurements and was computationally much less intensive than the Switching Kalman Filter.

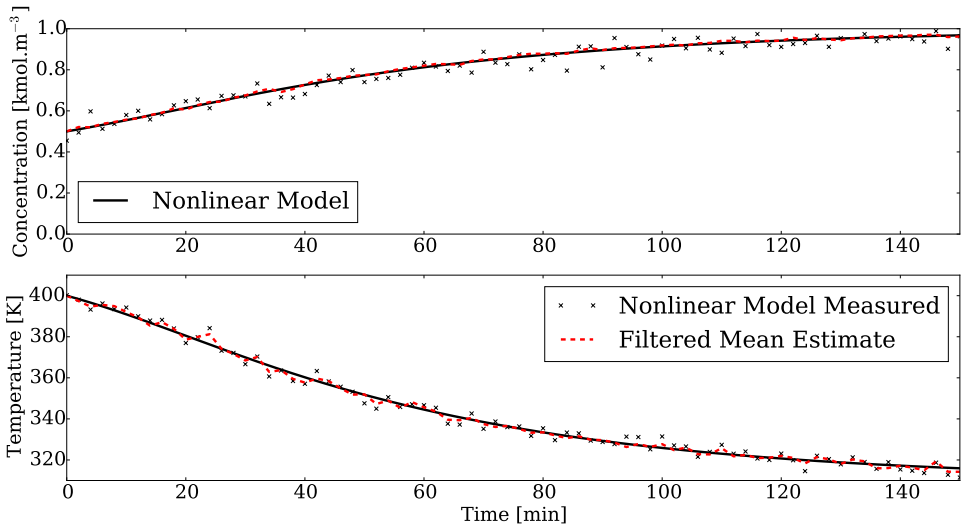

Figure 48: Time series evolution of the states with initial condition $(0.5, 400)$.

In Figure 49 we compare the Switching Kalman Filter to the standard Kalman Filter which uses only model 6.

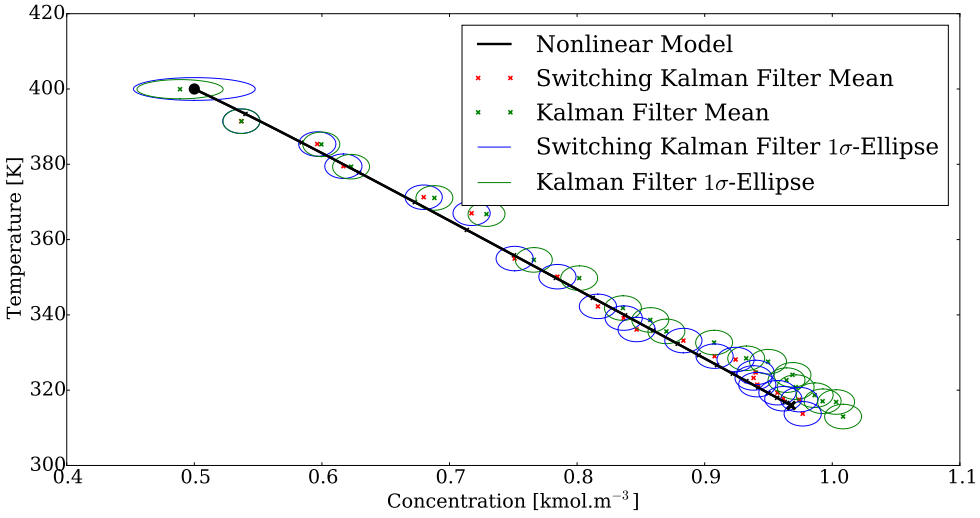

Figure 49: State space evolution of the Switching Kalman Filter and the standard Kalman filter which only uses one linear model.

Based on Figure 49 we see that there is some benefit in using the Switching Kalman Filter. However, this is less pronounced than when only one state is measured (see Figure 45). The reason is: the standard Kalman Filter eventually only uses the measurements to infer the posterior state estimates because the model is bad. Thus the accuracy of the standard Kalman Filter will depend strongly the magnitude of the measurement noise. However, the Switching Kalman Filter can depend on the models to a much greater extent and we thus expect large measurement noise to not affect the quality of the state estimate as much.

To illustrate this effect we increase the measurement noise significantly: $R = \begin{pmatrix} 0.1 & 0 \\ 0 & 100 \end{pmatrix}$. The results are shown in Figures 50 to 51. We see in Figure 50 that the measurement noise is significantly worse but the filter still tracks the states reliably.

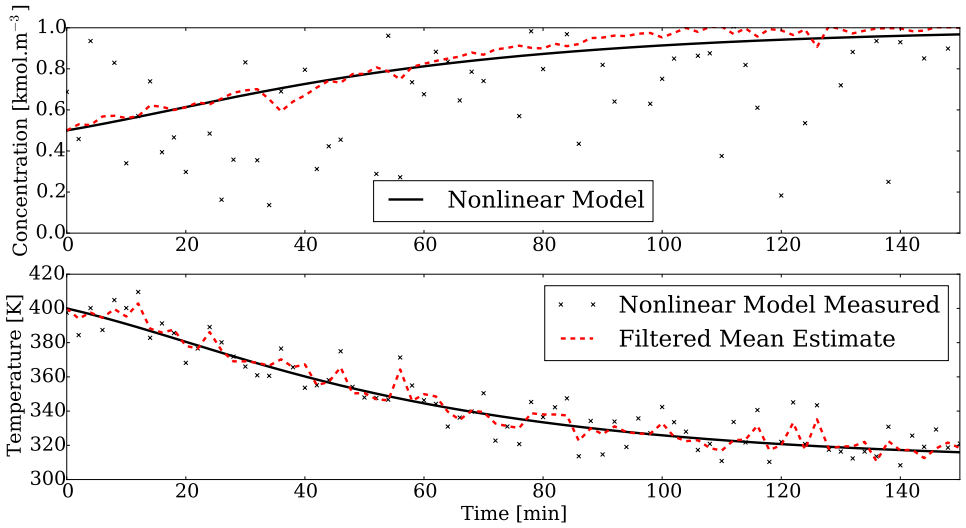

Figure 50: Time series evolution of the states with initial condition $(0.5, 400)$ and using a much larger noise covariance.

However, in Figure 51 we see that the standard Kalman Filter does not track the states well. Since the model is bad the standard Kalman Filter has to rely on measurements which are also bad, hence the poor performance. The Switching Kalman Filter does not have this problem and is significantly more reliable.

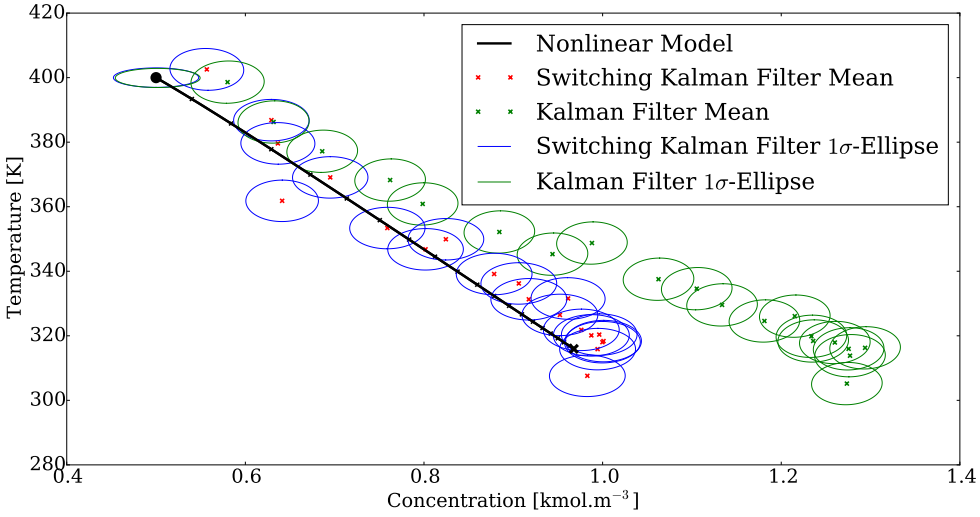

Figure 51: State space evolution of the Switching Kalman Filter and the standard Kalman filter which only uses one linear model.

The primary benefit of the Switching Kalman Filter is its ability to choose which models to use based on the observations. Although the technique is significantly more computationally intensive it is beneficial in situations where a single linear model does not adequately describe the system.

9 Inference using Nonlinear Hybrid Models

In this section we generalise the graphical model (shown in Figure 52 for convenience) of the previous section by dropping the assumption that the dynamic models are linear. The variables retain their meaning as before.

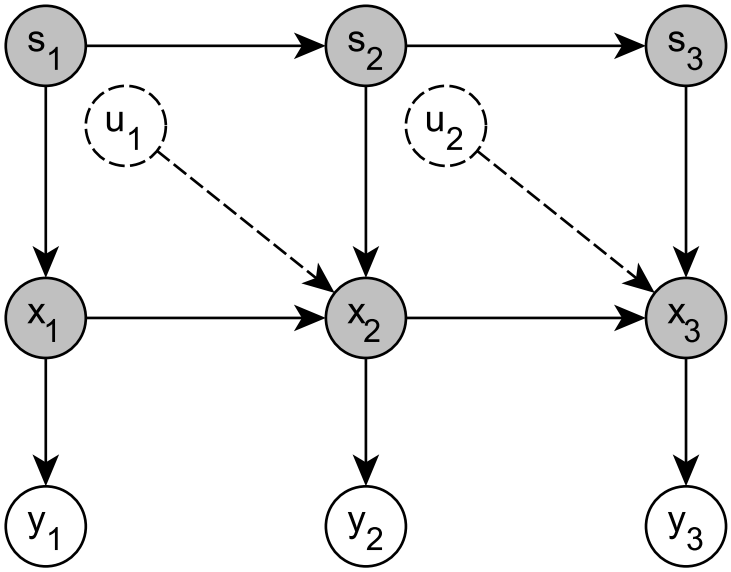

Figure 52: Graphical model of this section

Intuitively we are now using the switching variables to decide which nonlinear model better describes the observed system behaviour. At each time point 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 $s_t = 1, 2, ..., N$ denote a discrete, time homogeneous N state first order Markov chain with transition matrix P as discussed in the previous section. 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 shown in (109).

$$\begin{aligned} x_{t+1} &= f_i(x_t, u_t, w_{t+1}) \text{ with } w_{t+1} \backsim \mathcal{N}(0, W_i) \\ y_{t+1} &= g_i(x_{t+1}, v_{t+1}) \text{ with } v_{t+1} \backsim \mathcal{N}(0, V_i) \end{aligned} \tag{109}$$

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 $p(s_1)$ and $p(x_1|s_1)$ as well as the stochastic matrix P. In this section we manually specify the matrix P.

9.1 Exact Inference

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. We again turn to approximate inference algorithms.

Note that we cannot apply Rao-Blackwellisation (i.e. analytically evaluate the stochastic dynamical system) as before because the dynamic models are no longer linear. We use the adaptive Sequential Importance Resampling (i.e. the bootstrap) Particle Filter algorithm as discussed in the Nonlinear Models section.

9.2 Approximate Inference

We cannot analytically evaluate any part of the desired posterior distribution $p(s_{1:t}, x_{1:t}|y_{1:t})$ in a computationally feasible manner, so we must apply the adaptive Sequential Importance Resampling algorithm to the entire state space of Figure 52. The algorithm follows straightforwardly from our previous discussion [35]. We merely state the incremental weight function and proposal distribution we sample from in (110).

$$\begin{aligned} q_t(s_t, x_t|s_{1:t-1}, x_{1:t-1}, y_{1:t}) &= p(s_t|s_{t-1})p(x_t|s_t, x_{t-1}) \\ \alpha_t(s_{1:t}, x_{1:t}) &= p(y_t|x_t, s_t) \end{aligned} \tag{110}$$

Applying the algorithm is a straightforward extension of the bootstrap filter shown in the Nonlinear Models section given the weighting function and proposal distribution as shown below.

Switching Particle Filter Algorithm
For $t = 1$:

1. Sample $S_1^i \backsim p(s_1)$ and $X_1^i \backsim p(x_1|s_1)$.

2. Compute the weights $w_1(S_1^i, X_1^i) = p(Y_1^*|S_1^i, X_1^i)$ where Y_1^* is the observation. Normalise $W_1^i \propto w_1(S_1^i, X_1^i)$.

3. If the number of effective particles is below some threshold apply resampling with roughening (W_1^i, S_1^i, X_1^i) to obtain N equally weighted particles $(\frac{1}{N}, \bar{S}_1^i, \bar{X}_1^i)$ and set $(\bar{W}_1^i, \bar{S}_1^i, \bar{X}_1^i) \leftarrow (\frac{1}{N}, \bar{S}_1^i, \bar{X}_1^i)$ otherwise set $(\bar{W}_1^i, \bar{S}_1^i, \bar{X}_1^i) \leftarrow (W_1^i, S_1^i, X_1^i)$

For $t \geq 2$:

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_1^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 $(\bar{W}_t^i, \bar{S}_t^i, \bar{X}_t^i) \leftarrow (W_t^i, S_t^i, X_t^i)$

9.3 Particle Prediction

The prediction of the hybrid nonlinear states follows in an analogous manner to the prediction of the Nonlinear Models seen in the previous sections. 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).

9.4 Filtering the CSTR

In this section we illustrate the use of the Switching Particle Filter using nonlinear dynamical models. We assume a scenario where the rate constant of the CSTR decreases by an order of magnitude. 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.

We use 500 particles during all runs and use the stochastic matrix $P = \begin{pmatrix} 0.9 & 0.1 \\ 0.1 & 0.9 \end{pmatrix}$. The form of the matrix is motivated by physical considerations: once the catalyst denatures it is unlikely to fix itself. Thus, the matrix reflects a situation where a state is not likely to jump to another state. In all the simulations the catalyst denatures at 40 minutes into the run.

First we investigate a situation where only the temperature is measured. Second we include a concentration measurement and third we investigate the benefit of adding the second state measurement. Consider Figures 53 and 54 which show the Switching Particle Filter at work using only one measurement.

Based on Figure 53 it is evident that the first model (with the catalyst at full effectiveness) is the dominant model until about 40 minutes into the simulation. There is an abrupt change as the filter realises that first model no longer describes the system well and it switches to the second model. Up to about 120 minutes into the simulation the second model dominates. We then see a gradual decrease in the prominence of the second model.

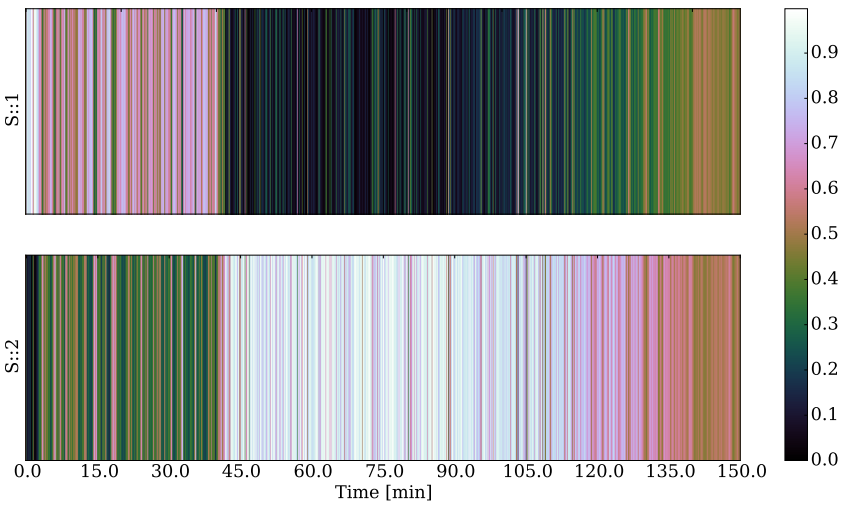

Figure 53: Weight of each switching index as time progresses. The catalyst denatures at 40 min.

This is exactly the type of behaviour we expected except, perhaps, the gradual decrease near the end of the simulation. The reason why this happens is because at high concentrations and low temperatures both models drive the system to the same steady state concentration. Therefore we see that both models explain the data.

In Figure 54 we see the transient response of the system. The green dashed line indicates the trajectory of the system if the catalyst did not denature.

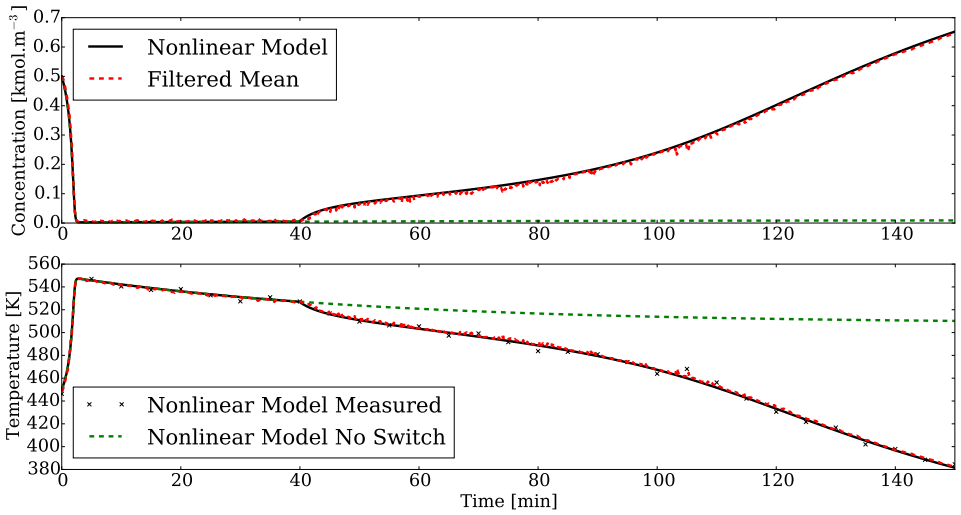

Figure 54: Time series evolution of the states with initial condition $(0.5, 450)$. The catalyst denatures at 40 min.

We see that the Switching Particle Filter accuracy tracks both temperature and concentration. The high accuracy of the model causes the concentration, although unmeasured, to be inferred

accurately. We expect that if the models were not as accurate the inference would suffer.

Next we consider Figures 55 and 56 which shows the filter at work using both temperature and concentration state measurements. In Figure 55 we see much the same behaviour as Figure 53.

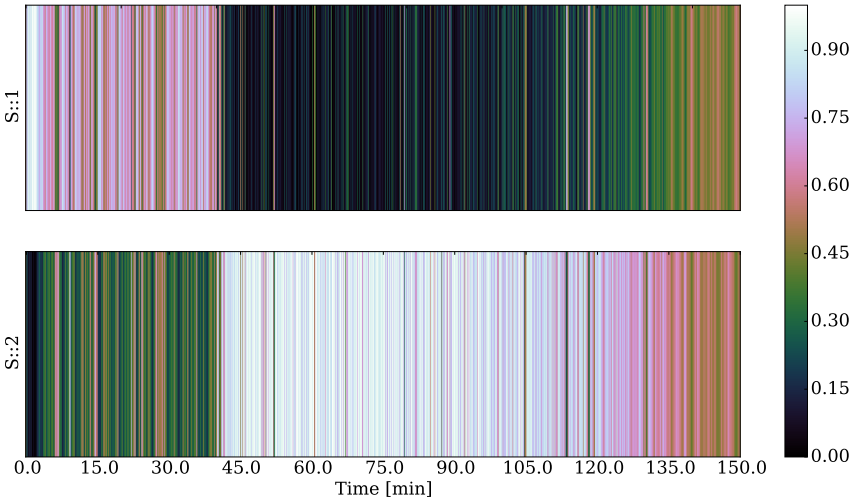

Figure 55: Weight of each switching index as time progresses. The catalyst denatures at 40 min.

In Figure 56 we see the transient response of the system. Again the filter is able to accurately track the system states.

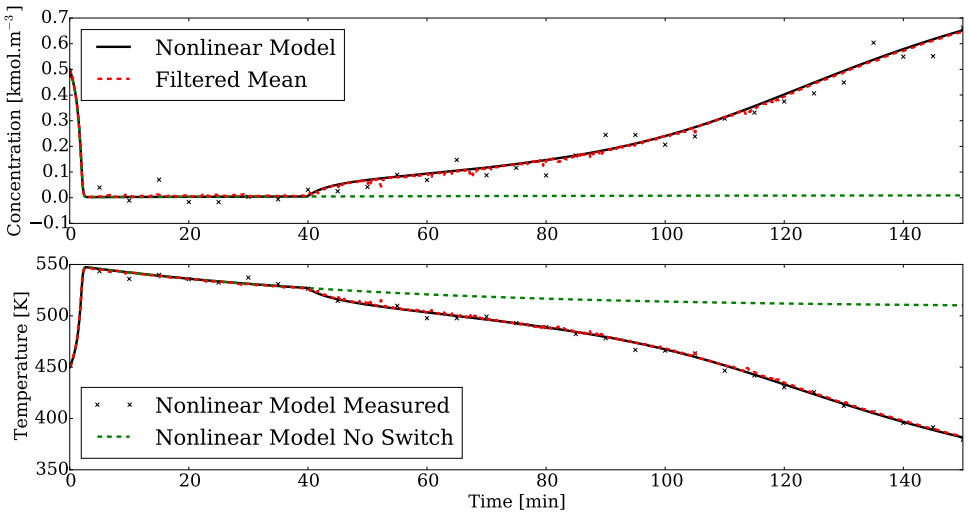

Figure 56: Time series evolution of the states with initial condition $(0.5, 450)$. The catalyst denatures at 40 min.

Generally speaking the second measurement will increase the filter accuracy if the models are not accurate. It is not clear if there is a tangible benefit to using the second measurement in

this case. To this end we introduce Figure 57 which compares the posterior state estimates using one and two measurements.

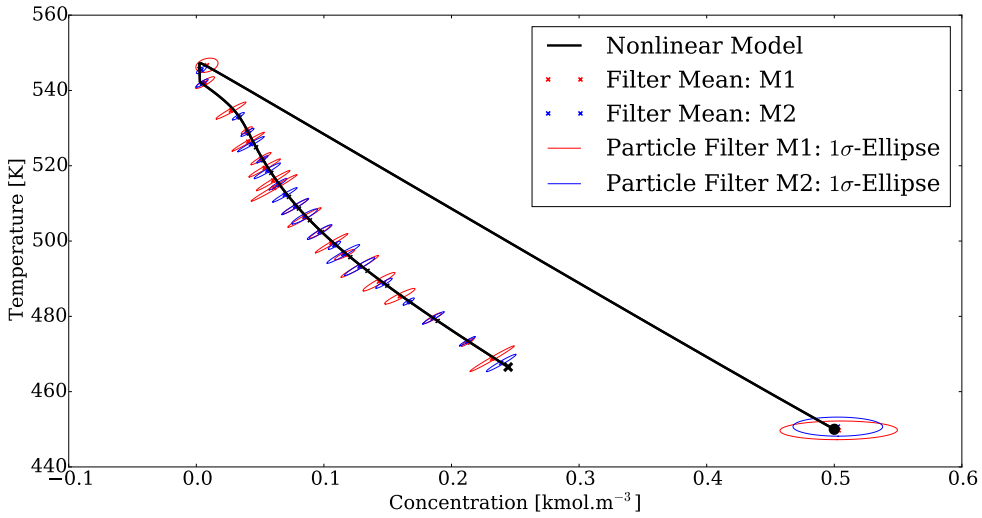

Figure 57: State space trajectory of the states with posterior state estimates and confidence regions superimposed thereupon. The blue curves correspond to a two measurement filter and the red curves to a single measurement filter.

Based on Figure 57 it is clear that the second measurement does indeed increase the accuracy of the posterior state estimate. This is not unexpected because the second measurement allows the filter to weed out even more particles which do not support the observation i.e. are representative of the true underlying state.

10 Stochastic Switching Linear Control

10.1 Current Literature

10.2 Unconstrained Control

10.3 Constrained Control

11 Conclusion

To do.

References

[1] Y. Bar-Shalom, X.R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation.* John Wiley and Sons, 2001.

[2] D. Barber. Expectation correction for smoothed inference in switching linear dynamical systems. *Journal of Machine Learning*, 7:2515–2540, 2006.

[3] D. Barber. *Bayesian Reasoning and Machine Learning.* Cambridge University Press, 2012.

[4] 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.

[5] C.M. Bishop. *Pattern Recognition and Machine Learning.* Springer, 2006.

[6] 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.

[7] M. Cannon, B. Kouvaritakis, and X. Wu. Probabilistic constrained mpc for multiplicative and additive stochastic uncertainty. *IEEE Transactions on Automatic Control*, 54(7), 2009.

[8] 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.

[9] R. Chen and J.S. Liu. Mixture kalman filters. *Journal of Royal Statistical Society*, 62(3):493–508, 2000.

[10] B.N. Datta. *Numerical Methods for Linear Control Systems - Design and Analysis.* Elsevier, 2004.

[11] M. Davidian. Applied longitudinal data analysis. North Carolina State University, 2005.

[12] R. De Maesschalck, D. Jouan-Rimbaus, and D.L. Massart. Tutorial: The mahalanobis distance. *Chemometrics and Intelligent Laboratory Systems*, 50:1–18, 2000.

[13] N. Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice-Hall, 1974.

[14] A. Doucet and A.M. Johansen. A tutorial on particle filtering and smoothing: fifteen years later. Technical report, The Institute of Statistical Mathematics, 2008.

[15] 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.

[16] 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.

[17] The Economist. In praise of bayes. Article in Magazine, September 2000.

[18] H.C. Edwards and D.E. Penny. *Elementary Differential Equations*. Pearson, 6th edition edition, 2009.

[19] W. Forst and D. Hoffmann. *Optimisation - Theory and Practice*. Springer, 2010.

[20] O.R. Gonzalez and A.G. Kelkar. *Electrical Engineering Handbook*. Academic Press, 2005.

[21] 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.

[22] K. Ito and K. Xiong. Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 45(5):910–928, 2000.

[23] R. J. Jang and C.T. Sun. *Neuro-fuzzy and soft computing: a computational approach to learning and machine intelligence*. Prentice-Hall, 1996.

[24] D. Koller and N. Friedman. *Probabilistic Graphical Models*. MIT Press, 2009.

[25] K. B. Korb and A. E. Nicholson. *Bayesian Artificial Intelligence*. Series in Computer Science and Data Analysis. Chapman & Hall, first edition edition, 2004.

[26] 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.

[27] U.N. Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems*. PhD thesis, Stanford Univesity, 2002.

[28] 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.

[29] P. Li, M. Wendt, and G. Wozny. A probabilistically constrained model predictive controller. *Automatica*, 38:1171–1176, 2002.

[30] W.L. Luyben. *Process Modeling, Simulation and Control for Chemical Engineers*. McGraw-Hill, 2nd edition edition, 1990.

[31] J.M. Maciejowski. *Predictive Control with constraints*. Prentice-Hall, 2002.

[32] O. Masahiro. Joint chance-constrained model predictive control with probabilistic resolvability. *American Control Conference*, 2012.

[33] 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.

[34] K.P. Murphy. Switching kalman filters. Technical report, Compaq Cambridge Research Lab, 1998.

[35] K.P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, University of California, Berkeley, 2002.

[36] K.P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[37] 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.

[38] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control*. Nob Hill Publishing, 2009.

[39] B. Reiser. Confidence intervals for the mahalanobis distance. *Communications in Statistics: Simulation and Computation*, 30(1):37–45, 2001.

[40] A.T. Schwarm and Nikolaou. Chance constrained model predictive control. Technical report, University of Houston and Texas A&M University, 1999.

[41] 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.

[42] 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.

[43] 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.

[44] R.S. Wills. Google's pagerank: the math behind the search engine. Technical report, North Carolina State University, 2006.

[45] J. Yan and R.R. Bitmead. Model predictive control and state estimation: a network example. In *15th Triennial World Conference of IFAC*, 2002.

[46] 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.

[47] 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.