COMPUTATIONAL SOLUTIONS TO TRADITIONAL PROBLEMS

COMPUTATIONAL APPROACHES TO STATE ESTIMATION OF PERIODIC SIGNALS AND CONTROL OF SWITCHED SYSTEMS

BY

HASSAN ELAGHOURY, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING AND SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2022) McMaster University
(Computing and Software) Hamilton, Ontario, Canada

TITLE:	Computational Approaches to State Estimation of Periodic Signals and Control of Switched Systems
AUTHOR:	Hassan Elaghoury B.Eng. (Mechatronics), McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. Martin von Mohrenschildt
NUMBER OF PAGES:	x, 103

To My Family

Acknowledgements

First and foremost I would like to thank my family for their support and patience throughout my studies. I would not be where I am without them. I would also like to thank my supervisor, Dr. von Mohrenschildt, for his guidance throughout this degree and for his patience and mentorship.

Contents

List of Figures

List of Tables

Thesis Introduction

Computational power nowadays is available in abundance. In the age of cheap sensors and big data, this marked increase in computational power has brought greater interest in solving traditional problems by computational means.

Often these computational solutions are placed under the umbrella term of Artificial Intelligence, and approached with this mindset. The term “Artificial Intelligence” has been around for decades. In fact, its origins date as far back as 1956, when it was first used at the Dartmouth Workshop. Since then, interest in AI has fluctuated, and many ideas and intelligent approaches have been researched. Research from these years is considered “traditional AI”. However, as computational capabilities have boomed, so has a new generation of AI. This generation is aptly referred to as “modern AI”. Modern AI approaches harness big data and immense computational power to solve problems computationally. However, a large downside to the application of modern AI methods is that they are often considered a blackbox, sometimes called “Blackbox AI”. This means that for many of these methods, the reasoning behind their decision-making or precisely how they achieve the results they do is unknown. For example, Neural Networks are quite effective for many applications and consequently are widely used today. However, the true reasoning behind their ability to work so effectively is still unknown. Another reason that blackbox AI methods may be undesirable is that the design parameters of these methods do not necessarily have a physical meaning for systems in question. This makes improvement of physical systems based on these methods quite difficult. Thus, we are interested in utilizing increased computational capacity to explore methods that maintain the physical meaning of the system and that can be analyzed in depth. In this thesis, we examine two traditional problems from this modern lens.

The first problem of interest is real-time parameter estimation from a data stream. Specifically, we are interested in estimating the parameters of a sinusoidal signal in the presence of additive Gaussian noise. For this purpose, an innovative Kalman Filter solution utilizing complex numbers is proposed and evaluated.

Second, a computational method for the approximation of optimal control policies of switched systems is studied. A dynamic programming approach is chosen in order to estimate the "cost-to-go" function of a switched system, leading to the synthesis of a control policy.

Part I

State Estimation of Sinusoidal Signals using the Extended Kalman Filter

Abstract

Sinusoidal signals are quite prevalent in practical applications. For example, any machine driven by a rotary shaft will exhibit periodic behaviour. For this reason, the estimation of sinusoidal parameters is studied extensively in the literature. Often in practical applications, there are unmodeled disturbances to the system, and the incoming measurements are noisy. Thus, estimation of the parameters of a sinusoidal signal in real-time for these conditions is of interest, calling for the use of a filter-based approach such as the Extended Kalman Filter. Considering the sinusoidal signal in its complex form, a novel approach is proposed resulting in a complex-valued filter. The resulting complex Extended Kalman Filter's performance is evaluated in various test environments and is compared to standard approaches to the estimation problem using a Discrete Fourier Transform and standard Extended Kalman Filter. Results show that the complex Extended Kalman Filter outperforms the standard approaches in some cases in both accuracy and convergence rate.

Chapter 1

Introduction

Sinusoidal signals are quite prevalent in the industry today. Such signals are found in many systems such as mechanical (eġṗendulums), electrical (eġȧlternating current), or even biological (eġL̇otka-Volterra equation). Thus, it is common to analyze sinusoidal signals to extract parameters such as amplitude, frequency, phase shift and damping [30].

This problem also arises in industrial applications where the sinusoidal parameters are used to diagnose, tune, and make design decisions regarding machinery with periodic motion. For example, a vibrating screen that separates differently-sized construction aggregate would operate with sinusoidal motion, and we may be interested in identifying the operating conditions of such a machine and tracking these conditions over time. However, practical systems often contain disturbances within the operating environment, as well as noise within the measurement tools used. Oftentimes the noise in the system can be modelled using a Gaussian distribution. Thus, we wish to estimate the frequency, amplitude and phase of sinusoidal signals in the presence of additive Gaussian noise using a real-time filter.

Many approaches exist in the literature to tackle this problem. This includes time-domain approaches as shown in [7], frequency domain approaches [21, 23], smoothing [26], and filtering [18, 20, 28]. A common theme of these methods is that they require the collection of a sufficient number of measurements from the system before an estimate may be found, and if we wish to update the estimate we need to continuously re-apply the methods to newly acquired data. Moreover, their performance under noisy conditions varies. In some applications we may be interested in finding and tracking the parameter in real-time as soon as new data is ready, improving the estimate over time as more data is acquired. A widely used method of real-time estimation that has proven to be quite effective is the Kalman Filter [27]. For example, [25] presents a standard application of the Extended Kalman Filter to estimate the parameters of a noisy sinusoidal signal.

In this thesis, we present what is, to our knowledge, a novel approach to this estimation problem using the Extended Kalman Filter. We use a complex model of the standard sinusoidal signal as the model in the Extended Kalman Filter (EKF). Following this, we apply the novel EKF model to simulated datasets as well as real datasets to evaluate its performance compared to the standard EKF model of sinusoidal signals as well as the Discrete Fourier Transform. We also investigate its performance using real data acquired from industrial partners.

Chapter 2

Background

A vast amount of literature exists on the problem of estimating the parameters of a sinusoidal signal. These methods are typically window-based, meaning they are performed on a window of N data points. In this section, we will discuss some of these methods.

2.1 Problem Setting

In this thesis, we assume that we are estimating a general noisy sinusoidal signal

$$s(t) = e^{-\zeta t} A \sin(\omega t + \phi) + v(t)$$

where ζ is the damping coefficient, A is the signal amplitude, ω (rad/s) is the frequency, ϕ (rad) is the phase angle, and $v(t)$ is Gaussian noise. We assume these parameters are constant over sufficiently small intervals, which allows us to model the signal in discrete time.

Let n be a step for the discrete system, and let Δt be the sampling period. Therefore, the signal is sampled at times $t = n\Delta t$, $n = 1, 2, ...$ Also let

$$\bar{\omega} = \omega \Delta t$$

It then follows that the signal can be described in discrete time as

$$s(n) = e^{\zeta n \Delta t} A \sin(\bar{\omega} n + \phi) + w(n), \tag{2.1.1}$$

where $w(n)$ is noise sampled from a Gaussian distribution at every step n.

2.2 Regression

Direct model fitting methods rely on the use of measurements taken directly from the system to fit a model and come up with an estimate. These measurements are used within a cost function, which is then minimized to produce an estimate.

2.2.1 Time Domain

In the time domain, the goal is to minimize a cost function defined as [21]:

$$J = \sum_{n=0}^{N-1} (y_n - e^{\zeta n \Delta t} A \sin(\bar{\omega} n + \phi))^2$$

where y_n is the n^{th} signal measurement. This is a window-based method that requires N sample points. Since the cost function is nonlinear in the parameters ω, ϕ, there may be local minima present. This causes a significant challenge for the minimization of the function, as methods such as least squares or gradient descent do not guarantee the extraction of a global minimum. These methods may get “stuck” on a local minimum, which could lead to a large error in the estimation of the parameters. Moreover, this estimation technique does not lend itself well to noisy signals, since the noise cannot be modelled anywhere. The presence of noise is the case for most practical applications of estimation methods, so this presents a serious challenge.

2.2.2 Frequency Domain

Similar to the time domain, the goal for direct model-fitting is to minimize a cost function defined in the frequency domain of Discrete Fourier Transform (DFT) coefficients as [21]

$$J = \sum_{n=0}^{N-1} [V_n - V(e^{j\omega_n})]^2$$

V_n is the DFT spectrum of the measured signal, and $V(e^{j\omega_n})$ is the theoretical spectrum of the undamped signal. The damped signal may also be estimated by replacing $V(e^{j\omega_n})$ with $V(\bar{e}^{j\omega_n})$, the theoretical spectrum of the damped signal. Since the amplitude, frequency, and damping coefficient of a sinusoidal signal directly impact the shape and magnitude of the signal’s DFT spectrum, an optimizer can find the optimal parameters to minimize the aforementioned cost function. The phase angle of the signal, however, may not be extracted, since a DFT contains only frequency information. Thus, to extract phase angle information for a sinusoidal signal using this method, an additional step must be done. Least Squares may be used to find the phase angle through the minimization

$$\min_{\phi} \sum_{n=0}^{N-1} \| y_n - e^{\zeta n \Delta t} A \sin(\bar{\omega} n + \phi) \|$$

where ζ, A, $\bar{\omega}$ are known from the estimation procedure. The signals are typically analyzed with a window w before finding the DFT spectrum [30]. The minimization process of the cost function is similar to the time domain approach to direct model-fitting. This also means that the issue of not being able to guarantee a global minimum persists. On the other hand, this approach is better at dealing with noisy signals due to the DFT calculation.

2.3 DFT

For an infinite signal with 2π periodicity, the Discrete-Time Fourier Transform (DTFT) is a frequency domain transform that is greatly beneficial in signal analysis. Mathematically, the DTFT is defined as

$$X(\omega) = \sum_{n=-\infty}^{\infty} x(n) e^{-i\omega n}.$$

There are two important things to note about the DTFT: 1) it transforms an infinite signal, and 2) it is a continuous function of frequency. In general, the signals we wish to analyze are finite in discrete time. Also, when we wish to implement such analysis techniques on embedded systems such as computers, continuous functions cannot be directly represented and must be sampled by the machine. Thus, for finite sequences, the Discrete Fourier Transform (DFT) is used. The DFT uses equally-spaced samples of a signal to find a sequence of equally-spaced samples of the DTFT of the given signal of the same length, and is defined as

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i\frac{2\pi}{N}kn}, \quad k = 0, 1, ..., N-1.$$

As we can see, the DFT is a finite frequency domain representation of a signal, defined only over the length of the original signal. The DFT has numerous practical applications [24] which make it a subject of great interest in literature. Particularly, a vast amount of literature exists on the implementation of the DFT in embedded systems. One well-known method is the Fast Fourier Transform (FFT) and is the method usually employed to calculate the DFT efficiently [3]. FFTW is an example of a computer library that uses the FFT algorithm for incredibly efficient DFT calculation [6]. In [4, 11], other methods for calculating the DFT are presented.

Since the DFT is derived for signals with an integer number of cycles [19], estimating the frequency directly from its coefficients is only accurate for signals containing an integer number of cycles in the measured sequence (with $\zeta = 0$). In this case, the frequency of the signal is equal to the DFT bin with the highest magnitude (magnitude specifically is considered since the DFT is comprised of complex values). If there is no noise present in the signal, there will only be one bin with a non-zero value in the DFT, which is the bin belonging to the frequency of the signal. Further, for $\zeta = 0$

$$X_k = A\frac{N}{2}e^{i\phi}.$$

Thus,

$$A = 2|X_k|/N \text{ and } \phi = \angle X_k.$$

However, in practical applications, we rarely have such an ideal case. First, there is the issue of estimating the damping coefficient ζ for the non-zero case, and then there is the issue of having a non-integer number of cycles of the signal. In the latter case, the DFT suffers from what is called "spectral leakage" [10]. In the cases of spectral leakage and damped signals, interpolation must be used using the DFT bin with the highest magnitude as well as surrounding bins. In [5], various interpolation methods for these cases are discussed. The Bertocco algorithm [2] and Yoshida algorithm [29] are also examples of these interpolation methods. Simpler interpolation methods may also be used such as quadratic interpolation with k surrounding DFT bins.

2.4 Linear Prediction

Another approach to estimating a discrete-time signal such as (2.1.1) is to find a parametric model for the signal that can be used to estimate future samples. One such method is Linear Prediction (LP).

The idea behind LP is to consider a signal $s(n)$ as the output of some system with an unknown input $u(n)$ such that $s(n)$ may be expressed as a linear combination of the system's past inputs and outputs. Mathematically, this is expressed by the relationship

$$s(n) = -\sum_{k=1}^{p} a_k s(n-k) + \sum_{l=0}^{q} b_l u(n-l)$$

where a_k, $1 \leq k \leq p$ and b_k, $0 \leq l \leq q$ are the parameters of the model. If the signal of interest may be correctly expressed using this model, then we say that the signal is predictable from its past inputs and outputs. A thorough background on LP is provided in [15], and readers are encouraged to read this work for more background on LP.

Going back to our signal of interest $s(n)$ (2.1.1), it is shown in [30] that this signal is the impulse response $h(n)$ of the digital filter

$$v(n) = b_1u(n-1) - a_1v(n-1) - a_2v(n-2) \tag{2.4.1}$$

where $u(n)$ is an unknown system input, $v(n)$ is the system output, $b_1 = Ae^{-\zeta}\sin(\omega)$, $a_1 = -2e^{-\zeta}\cos(\omega)$, $a_2 = e^{-2\zeta}$. Following from this,

$$s(n) = -a_1s(n-1) - a_2s(n-2)$$

is a valid model for linear prediction, where a_1 and a_2 are parameters of the model that must be optimized, $s(-1) = 0$ and $s(-2) = -Ae^{-\zeta}\sin(\omega)$. There are plenty of methods discussed in literature to best optimize the parameters of the model. However, we will only discuss the simplest approach which is to simply solve a linear system using collected datapoints. Further approaches to finding these parameters are presented in [1,13,22] to name a few.

Given N noisy measurements of the signal $s(n)$, labelled $y(n), n = 0, ..., N-1$, we wish to find a_1, a_2 such that

$$\begin{bmatrix} y(-1) & y(-2) \\ y(0) & y(-1) \\ \vdots & \vdots \\ y(N-2) & y(N-3) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = - \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix}, \qquad \mathbf{Ya} = -\mathbf{y}.$$

To solve this system for $N \neq 4$, the pseudo inverse of $\mathbf{Y}$ is used. Solving for $\mathbf{a}$, we find that

$$\mathbf{a} = -(\mathbf{Y}^T\mathbf{Y})^{-1}\mathbf{Y}^T\mathbf{y}.$$

Once the LP model is established, it is not immediately clear how the parameters of interest may be derived. To do so, first we note that the transfer function of the filter (2.4.1) is

$$H(z) = \frac{b_1 z^{-1}}{1 + a_1 z^{-1} + a_2 z^{-2}} = \frac{b_1 z^{-1}}{(1 - z_1 z^{-1})(1 - z_1^* z^{-1})}$$

where $z_1 = e^{-\zeta}e^{i\omega}$ and z_1^* is its complex conjugate. From this we see that once the prediction coefficients a_1 and a_2 are found, we find the complex poles of the polynomial $1 + a_1 z^{-1} + a_2 z^{-2}$ and use them to calculate ζ and ω. To be precise,

$$\omega = \angle(z_1), \qquad \zeta = -\ln(|z_1|)$$

or

$$\omega = |Im(\ln(z_1))|, \qquad \zeta = -Re(\ln(z_1)).$$

To find the amplitude A and phase shift ϕ, the signal is first rewritten in its complex form:

$$s(n) = Ae^{-\zeta n}cos(\omega n + \phi) = cE_{n,1} + c^* E_{n,2}$$

where $c = \frac{A}{2}e^{i\phi}$, c^* is its complex conjugate, $E_{n,1} = e^{i\omega n - \zeta n}$ and $E_{n,2} = e^{-i\omega n - \zeta n}$. Thus, using the estimated frequency ω, the damping coefficient ζ and the N measurements of the system, c can be found by solving

$$\begin{bmatrix} E_{0,1} & E_{0,2} \\ E_{1,1} & E_{1,2} \\ \vdots & \vdots \\ E_{N-1,1} & E_{N-1,2} \end{bmatrix} \begin{bmatrix} c \\ c^* \end{bmatrix} = \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix}, \qquad \mathbf{Ec} = \mathbf{y}.$$

From there,

$$A = 2|c|, \quad \phi = \angle c.$$

A more detailed analysis of this method as well as simulations is available in [30].

Chapter 3

Kalman Filter

The Kalman Filter was first introduced by R. E. Kalman in his seminal work [12]. In this work, he introduced an optimal estimator that was recursive, allowing new measurements to be incorporated into estimations as they are acquired. Since this paper was published, and in tandem with fast advancements in digital computing, the Kalman Filter has been the subject of a vast amount of research and application. This is largely due to its simplicity in implementation [17]. In this section, we aim to provide a brief introduction to the Kalman Filter and the Extended Kalman Filter (EKF), which is an extension of the Kalman Filter for non-linear stochastic dynamic systems. Many resources exist in the literature that provide more detailed discussions of the Kalman Filter and its variations such as found in [8, 14, 16].

3.1 Kalman Filter

First, the systems to which the Kalman Filter applies must be established. In general, the Kalman Filter estimates the state $x \in \mathbb{R}^n$ of a linear stochastic dynamic system defined by the equations

$$x_{k+1} = A_k x_k + w_k$$

with output $y \in \mathbb{R}^m$ described by equation

$$y_k = C_k x_k + v_k.$$

w_k is the process noise of the system, v_k is the measurement noise of the system and k is a discrete step of the system. The noise variables are assumed to be independent zero-mean stochastic variables belonging to a normal distribution as in

$$w \sim \mathcal{N}(0, Q)$$
$$v \sim \mathcal{N}(0, R).$$

Process noise may be present in the system due to several factors such as uncertainty in modelling, disturbances that cannot be modelled, modelling approximations, etc. Measurement noise is simply noise in the incoming measurements of the system.

The goal of the Kalman Filter is the following: given an uncertain model of a system and incoming noisy measurements of the system, we want to find a *posterior* state estimate $\hat{x}$ that is a linear combination of a *prior estimate* $\hat{x}^+$ of the system and the difference between the measurement of the system z and the predicted measurement of the system $C\hat{x}^+$ weighted by some gain K, often called the *Kalman gain.* Mathematically, this is written as

$$\hat{x} = \hat{x}^+ + K(z - C\hat{x}^+). \tag{3.1.1}$$

To find the gain K, we aim to minimize the posterior estimate error covariance of the system. The prior and posterior estimate error are defined as

$$e^+ = x - \hat{x}^+$$
$$e = x - \hat{x}$$

and the prior and posterior estimate error covariance are

$$P^+ = E[e^+ e^T]$$
$$P = E[ee^T].$$

Thus, by substituting (3.1.1) into the definition for estimate error e, substituting into (3.1.2) and minimizing with respect to K, we find that the gain K that minimizes this error covariance is

$$K = P^+ C^T (CP^+ C^T + R)^{-1}$$

where R is the covariance of the measurement noise. We can see from this that the measurement covariance R plays a pivotal role in finding the estimated state. As R approaches zero, the estimation equation favours the incoming measurement more. On the other hand, as R becomes larger in magnitude relative to the prior estimate error covariance P^+, the prior state estimate is favoured more heavily. To think of this in another way, we can say that the covariance R tells us how uncertain we are of our incoming measurement z.

The full Kalman Filter algorithm is an iterative algorithm with two distinct "steps". First, the *prediction step* is performed to estimate the system state solely based on the system model. The prior estimate error covariance is also calculated at this step. The equations used in the prediction step are

$$\hat{x}_k^+ = A_k \hat{x}_{k-1} \tag{3.1.2}$$

$$P_k^+ = A_k P_{k-1} A_k^T + Q_k. \tag{3.1.3}$$

Second, the *measurement step* is taken once a new measurement is available. In this step, the Kalman gain is calculated, the posterior state estimate is found and the posterior estimate error covariance is calculated. The measurement step is as follows:

$$K_k = P_k^+ C_k^T (C_k P_k^+ C_k^T + R_k)^{-1} \tag{3.1.4}$$

$$\hat{x}_k = \hat{x}_k^+ + K_k(z_k - C_k \hat{x}_k^+) \tag{3.1.5}$$

$$P_k = (I - K_k C_k) P_k^+. \tag{3.1.6}$$

The recursive nature of this algorithm is very appealing, as it is a fairly lightweight approach to the continuous, online estimation of a system's state.

Going back to the discussion regarding the role of the covariance matrix R_k in determining the impact of incoming measurements, we can also see from equation (3.2.4) that the other important factor is the prior estimate error covariance P_k^+. Referring to equation (3.1.3), P_k^+ heavily relies on the process noise covariance Q_k. From this, we can say that the ratio of $Q_k : R_k$ greatly affects the performance of the filter. In theory, calibrating R_k is possible. This can be done using specifications of the physical measurement devices used or by collecting a measurement set of the system and calculating the variance of that set. Due to the nature of uncertainties that Q_k encapsulates, many of which are very difficult to accurately quantify, calibration of

Q_k is much more difficult. Thus, a good estimate of Q_k and especially R_k may be established in the analytical phase of the Kalman Filter design, but tuning of the values in Q_k and R_k is usually required to maximize filter performance for every application.

3.2 Extended Kalman Filter

As mentioned in the previous section, the Kalman Filter is a state estimator for linear stochastic dynamic systems. In reality, many systems that are of interest to us are modelled by non-linear dynamics. For such systems, directly applying the equations of the Kalman Filter is not possible.

One way that is often used to tackle issues of non-linearity is linearization, and it is the same approach used in this case by defining the Extended Kalman Filter (EKF). In the case of non-linear stochastic dynamic systems, the EKF linearizes the estimation around the current state estimate to establish an estimate of a non-linear system.

First, we will revisit the dynamics of the system of interest. In general, a non-linear stochastic dynamic system is defined by the equations

$$\begin{aligned} x_{k+1} &= f(x_k) + w_k \\ y_k &= h(x_k) + v_k \end{aligned}$$

with $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$, w_k is a zero-mean Gaussian process noise and v_k is a zero-mean Gaussian measurement noise as defined for the standard Kalman Filter. The issue of linearization of these dynamics and their specifics is discussed in great detail in the literature. However, for the sake of brevity, these details will be omitted in this work and readers are encouraged to read [27] for details regarding the linearization process. In short, the Jacobian of the respective non-linear processes may be used to linearize them around the current state estimate.

The EKF algorithm contains the same two distinct steps as the linear Kalman Filter. First, the equations belonging to the prediction step are

$$\hat{x}_k^+ = f(\hat{x}_{k-1}) \tag{3.2.1}$$

$$P_k^+ = J_f(\hat{x}_{k-1})P_{k-1}J_f^T(\hat{x}_{k-1}) + Q_k, \tag{3.2.2}$$

where P, P^+ and Q_k are as defined for the linear Kalman Filter, and $J_f(\cdot)$ is the Jacobian of the process equation f with respect to x. The equations in the measurement step are

$$K_k = P_k^+ J_h^T(\hat{x}_k^+)[J_h(\hat{x}_k^+)P_k^+ J_h^T(\hat{x}_k^+) + R_k]^{-1} \tag{3.2.3}$$

$$\hat{x}_k = \hat{x}_k^+ + K_k[z_k - h(\hat{x}_k^+)] \tag{3.2.4}$$

$$P_k = [I - K_k J_h(\hat{x}_k^+)]P_k^+, \tag{3.2.5}$$

where similarly z_k and R_k are as defined for the linear Kalman Filter, and $J_h(\cdot)$ is the Jacobian of the measurement equation with respect to x.

One important thing to note about the EKF is that due to the approximations used to linearize the processes, there is no guarantee that the EKF is the optimal estimator for a non-linear system. On the other hand, the linear Kalman Filter is shown to be the optimal estimator for linear systems.

Chapter 4

Estimation Using the EKF

In this section, the estimation methods used in this thesis will be presented. In general, we wish to estimate a system described by the equation

$$s_n = A \sin(\bar{\omega} n + \phi) \tag{4.0.1}$$

using the EKF. We choose to focus on undamped periodic signals in this case. This is still a reasonable assumption, since often in industry the main interest lies in estimating the state of a machine in steady-state motion and monitoring the deviation of the states from the expected operational state. First, an EKF using real numbers will be presented. Following that, an EKF using complex numbers will be presented.

4.1 Real Numbers Approach

The first approach is to formulate the EKF directly using equation (4.0.1). This approach is outlined in more detail in [25].

First, we define the phase of the signal, θ_n as

$$\theta_n = \bar{\omega} n$$

This helps us in defining the filter. Using θ_n, equation (4.0.1) may be written as

$$s_n = A \sin(\theta_n + \phi) \tag{4.1.1}$$

The parameters of interest are the amplitude, A, frequency, $\bar{\omega}$, and phase shift, ϕ. However, in order to formulate the filter, we will also consider the phase, θ_n, as a state variable. From this, we can establish a state vector, x_n, in order to estimate the parameters as

$$x_n = \begin{bmatrix} \theta_n \\ A \\ \bar{\omega} \\ \phi \end{bmatrix}$$

Next, the update equations of the signal of interest need to be specified. Equation (4.1.1) assumes the parameters A, $\bar{\omega}$, and ϕ are constant. This assumption may be extended to practical systems of interest as long as we assume that these parameters are constant over a sufficiently small interval of time which is greater than or equal to the sampling period T_s. Thus, with this established, the update equations are:

$$\hat{x}_{n+1} = f(\hat{x}_n) = \begin{bmatrix} \hat{\theta}_n + \hat{\bar{\omega}}_n \\ \hat{A}_n \\ \hat{\bar{\omega}}_n \\ \hat{\phi}_n \end{bmatrix}$$

Due to the introduction of the phase θ, the update equations are linear in the state variables, as shown by the Jacobian:

$$\begin{aligned} J_f(\hat{x}_n) &= \frac{\partial f(\hat{x}_n)}{\partial x} \\ &= \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

With the state dynamics of the signal formulated, we must also establish the measurement, or output, equations. The measurement equation, $h(x)$, is the same as equation (4.1.1)

$$h(x_n) = A \sin(\theta_n + \phi).$$

However, unlike the update equations h is non-linear in nature, so the Jacobian J_h linearizes this equation:

$$\begin{aligned} J_h(\hat{x}_n) &= \frac{\partial h(\hat{x}_n)}{\partial x} \\ &= \begin{bmatrix} \hat{A}_n \cos(\hat{\theta}_n + \hat{\phi}_n) & \sin(\hat{\theta}_n + \hat{\phi}_n) & 0 & \hat{A}_n \cos(\hat{\theta}_n + \hat{\phi}_n) \end{bmatrix} \end{aligned}$$

With the state update and measurement equations established, the remaining elements of the EKF, namely the noise covariance matrices Q and R must be defined. Through the testing conducted, which will be discussed in a later section, it was discovered that there is no single ideal way of defining the covariance matrices. As such, they are left as tuning parameters that must be calibrated when applying the filter.

4.2 Complex Numbers Approach

The novel approach introduced in this thesis for formulating the EKF for noisy sinusoidal signals is the complex numbers approach. First, we will return to the original equation of interest, equation (4.0.1). In general, we may use Euler's formula to write sinusoidals as complex exponentials:

$$\begin{aligned} e^{\pm i\theta} &= \cos(\theta) \pm i\sin(\theta) \\ \cos(\theta) &= \frac{1}{2}(e^{i\theta} + e^{-i\theta}) \\ \sin(\theta) &= \frac{1}{2i}(e^{i\theta} - e^{-i\theta}) \end{aligned}$$

Also, note that

$$\cos(\theta) = \sin(\theta + \frac{\pi}{2})$$

Thus, in terms of our parameters of interest A, $\bar{\omega}$, and ϕ, we may consider the signal

$$s_n = A\cos(\bar{\omega}n + \phi) \tag{4.2.1}$$

instead of equation (4.0.1), and only the phase shift ϕ needs to be augmented by adding $\frac{\pi}{2}$ to the cosinusoidal phase shift. Therefore, due to its ease of computation relative to the complex exponential form of the sine equation, we will formulate the EKF using the cosine complex exponential equation (4.2.1), leading to the signal of interest taking the form

$$s_n = A\cos(\bar{\omega}n + \phi) = \frac{A}{2}e^{i\bar{\omega}n}e^{i\phi} + \frac{A}{2}e^{-i\bar{\omega}n}e^{-i\phi} \tag{4.2.2}$$

To aid in setting up the EKF for this system, we introduce two state variables, C_n and $\bar{C}_n$, defined as follows

$$C_n = \frac{A}{2}e^{i\bar{\omega}n} \text{ and } \bar{C}_n = \frac{A}{2}e^{-i\bar{\omega}n}$$

This allows us to rewrite equation (4.2.2) as

$$s_n = C_n e^{i\phi} + \bar{C}_n e^{-i\phi} \tag{4.2.3}$$

Given these new state variables, we can write the state vector for the filter as

$$x_n = \begin{bmatrix} \bar{\omega} \\ \phi \\ C_n \\ \bar{C}_n \end{bmatrix}$$

The amplitude, A, of the signal remains a parameter of interest, but does not need to be present as a state variable in the filter. This is because it can be obtained from C and $\bar{C}$ using

$$A = |C| + |\bar{C}|$$

Since $\bar{C}$ is simply the complex conjugate of C, we may also say that

$$A = 2|C| = 2|\bar{C}|$$

With the state variables of the filter established, we may define the update equations. Similar to the real numbers approach, the frequency $\bar{\omega}$ and phase shift ϕ are constant, so their update equations are unchanged. By definition,

$$\begin{aligned} C_{n+1} &= \frac{A}{2} e^{i\bar{\omega}(n+1)} \\ &= \frac{A}{2} e^{i\bar{\omega}n} e^{i\bar{\omega}} \\ &= C_n e^{i\bar{\omega}} \end{aligned}$$

The update equation for $\bar{C}$ follows the same logic. Therefore, the update equations for the system are established as

$$\hat{x}_{n+1} = f(\hat{x}_n) = \begin{bmatrix} \hat{\bar{\omega}}_n \\ \hat{\phi}_n \\ \hat{C}_n e^{i\hat{\bar{\omega}}_n} \\ \hat{\bar{C}}_n e^{-i\hat{\bar{\omega}}_n} \end{bmatrix}$$

To linearize the update equations for the EKF, their Jacobian is calculated as

$$\begin{aligned} J_f(\hat{x}_n) &= \frac{\partial f(\hat{x}_n)}{\partial x} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ ie^{i\hat{\omega}_n}\hat{C}_n & 0 & e^{i\hat{\omega}_n} & 0 \\ -ie^{-i\hat{\omega}_n}\hat{\bar{C}}_n & 0 & 0 & e^{-i\hat{\omega}_n} \end{bmatrix} \end{aligned}$$

With this, we establish the dynamics matrix of the signal of interest. The next consideration is the measurement equations of the signal. There is one measurement equation for this signal, and it takes advantage of C and $\bar{C}$. In this case, measurement equation h is defined as

$$h(x_n) = C_n e^{i\phi} + \bar{C}_n e^{-i\phi}$$

Once again, the Jacobian J_h is used to linearize a non-linear measurement equation, and is given by

$$\begin{aligned} J_h(\hat{x}_n) &= \frac{\partial h(\hat{x}_n)}{\partial x} \\ &= \begin{bmatrix} 0 & i\hat{C}_n e^{i\hat{\phi}_n} - i\hat{\bar{C}}_n e^{-i\hat{\phi}_n} & e^{i\hat{\phi}_n} & e^{-i\hat{\phi}_n} \end{bmatrix} \end{aligned}$$

Similar to the real numbers approach, the noise covariance matrices are left as tuning parameters.

Chapter 5

Evaluation

The goal of this chapter is to evaluate the performance of the proposed complex EKF. To do so, a few simulation cases are considered to encapsulate various conditions of signal noise content and EKF calibration. The performance is evaluated relative to the real EKF as well as the Fourier Transform, which is the standard method used by industry partners. Finally, the complex EKF is applied to real data to demonstrate its effectiveness.

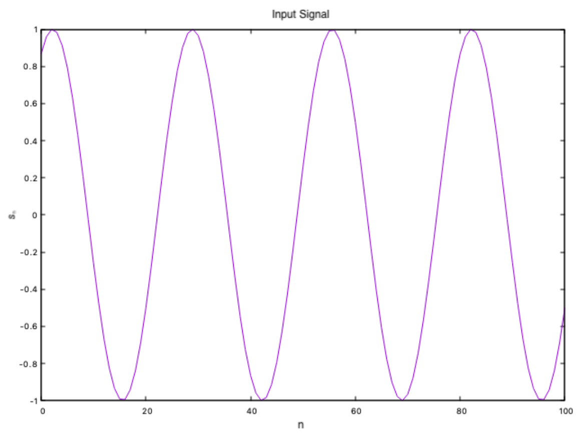
5.1 Experimental Setup

Evaluation of the EKF reduces to two criteria: convergence and estimation error. Both can be assessed by measuring the error in the parameter estimate. The filter converges if it reaches an estimate and remains within a sufficiently small range of that estimate. The estimation error is the final error between the state estimate and the true value of the state. To generate testing scenarios that can exhibit these criteria, data was generated according to the previously introduced model (4.0.1). Real-world data was also used to evaluate the filter's performance in a practical application.

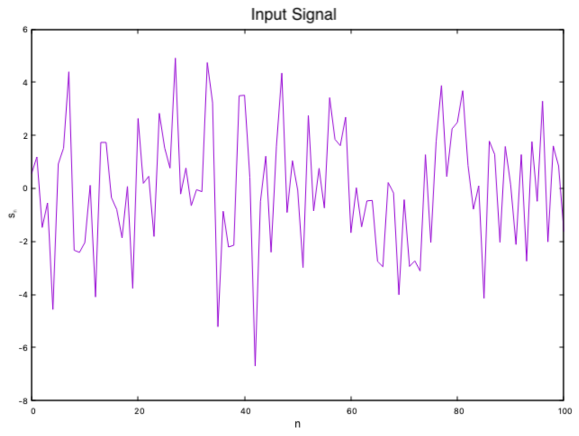
To gain a full picture of how the filter fulfils the above criteria, five main test classes were examined. The incoming measurements of the model may be noisy or noiseless. Another important part of the EKF is the initialization of the filter. Namely, the initial estimate of the state, $\hat{x}_0^+$, and the initial estimate of the error variance, P_0. To capture that, the test cases also take into account a perfect initial guess for each data scenario and an imperfect initial guess for each data scenario. Thus, the five test classes are:

1. Noisy data with perfect initialization
2. Noisy data with imperfect initialization
3. Noiseless data with perfect initialization
4. Noiseless data with imperfect initialization
5. Real data

In the data sets with noise, the noise is Gaussian with a mean of 0 and a variance equal to the amplitude of the signal, $\sigma^2 = A$. Examples of noiseless and noisy signals with the mentioned noise content are shown in Figure 5.1 for reference.



(a) Simulated data with no noise



(b) Simulated data with added noise

Figure 5.1: Example signals

In cases with an imperfect initial guess, the initial guess for frequency, amplitude and phase shift of the signal are altered. For all tests, the sampling frequency is held constant and the frequency of the actual signal is used to manipulate the ratio $\frac{f}{f_s}$. This ratio is significant for two reasons: 1) to be able to distinguish between sine waves and cosine waves, $\frac{f}{f_s} \leq 0.25$ and 2) it dictates how many cycles of the signals we see, in that $cycles = \frac{N \cdot f}{f_s}$, where N is the number of points observed. Not seeing enough cycles of a signal affects convergence and performance.

5.2 Implementation

C++ was used to generate all the data for the experiments, as well as the implementation of the estimation methods. The C++ library Eigen [14] was used to perform the linear algebra manipulations required.

5.3 Convergence

In this section, the convergence of the filter will be presented for the test classes previously mentioned. For all classes, the error is expressed in decibels (dB), where $error_{dB} = 10\log(|error|)$. This is because the filter converges quite quickly and to small magnitudes of error, and using dB makes distinguishing between the convergence of the different $\frac{f}{f_s}$ ratios easier. We only present the plots for the frequency estimate for two reasons: 1) we found that the frequency and amplitude plots followed extremely similar patterns, so for the sake of brevity only the frequency is discussed in this section, and 2) as noted in [30], once the frequency is found, it is straightforward to find the amplitude and phase shift of the signal directly from the data.

5.3.1 Noiseless Data With Perfect Initial Guess

Convergence of the complex EKF is demonstrated for this test class in Figure 5.2.

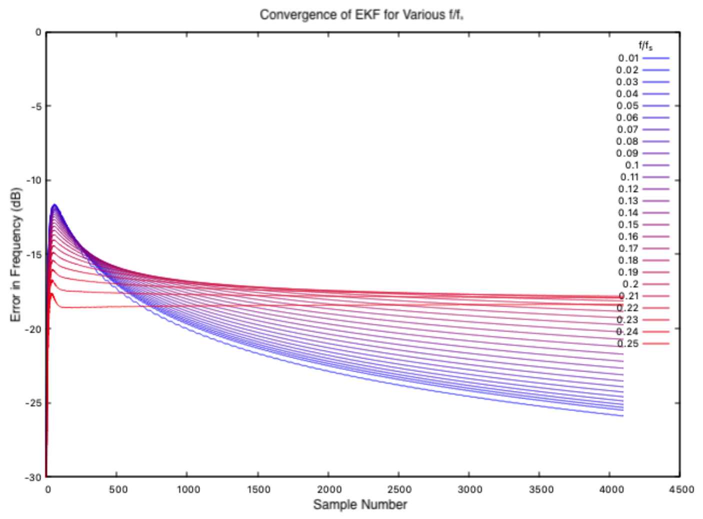


Figure 5.2: Convergence of the complex EKF for noiseless data with perfect initial guess

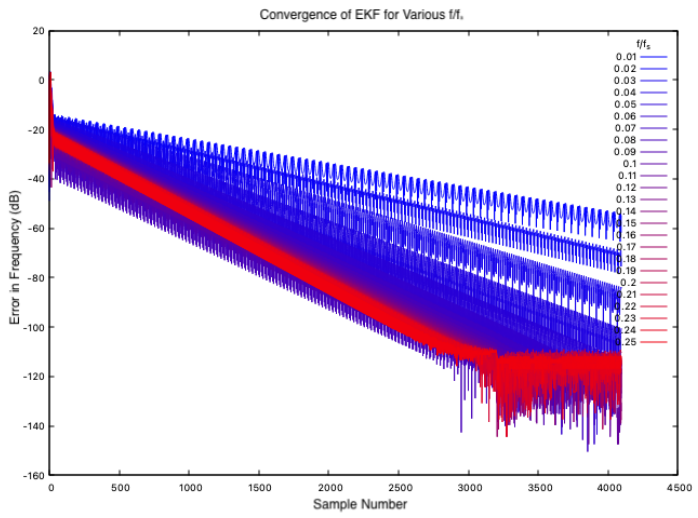


Figure 5.3: Convergence of the real EKF for noiseless data with perfect initial guess

There is a clear pattern that can be seen which is dependent on the ratio $\frac{f}{f_s}$. A colour gradient is used to demonstrate this. The colours of the plot lines range from blue to red, with blue being a signal with $\frac{f}{f_s} = 0.01$, and red being a signal with frequency equal to $\frac{f}{f_s} = 0.25$. Lower ratios of $\frac{f}{f_s}$ lead to a smaller convergence error but have a slower convergence rate. Higher ratios lead to a larger convergence error but converge quicker. The same can be said for the real EKF, which is demonstrated in Figure 5.3. The higher ratio signals still converge quicker. Finally, both the complex and real approaches converge for this test class.

5.3.2 Noiseless Data With Imperfect Initial Guess

Convergence of the complex EKF is demonstrated in Figure 5.4.

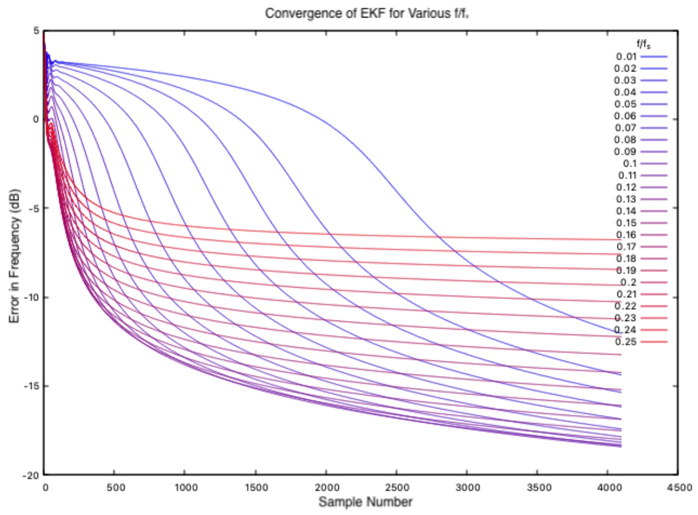


Figure 5.4: Convergence of the complex EKF for noiseless data with imperfect initial guess

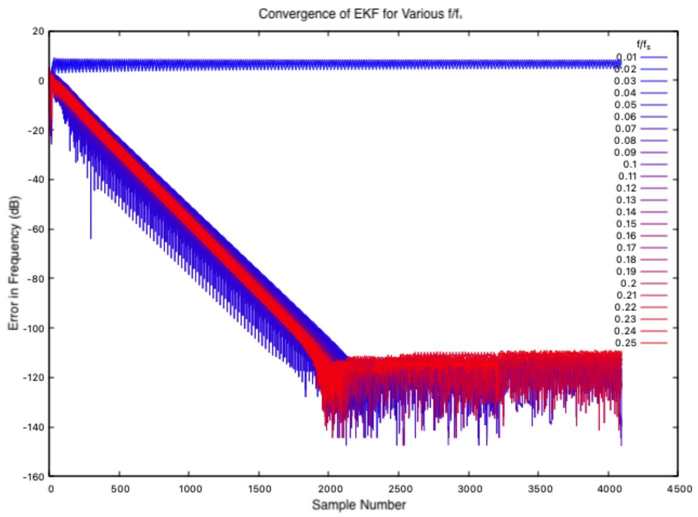


Figure 5.5: Convergence of the real EKF for noiseless data with imperfect initial guess

A similar pattern emerges for this test class compared to the test class with a perfect initial guess, shown in Figures 5.2, 5.3. All the frequency ratios converge, but the convergence rate varies greatly. Also, the smallest error is generated somewhere in the middle of the range $0.01 \leq \frac{f}{f_s} \leq 0.25$, rather than occurring at the lowest frequency ratio. The real EKF also converges as seen in Figure 5.5, and displays a large amount of oscillation. This is due to the choice of uncertainty parameters in the filter, as well as numerical errors.

5.3.3 Noisy Data With Perfect Initial Guess

In this test class Gaussian noise is inserted into the signal as described earlier. The result for the complex EKF is shown in Figure 5.6.

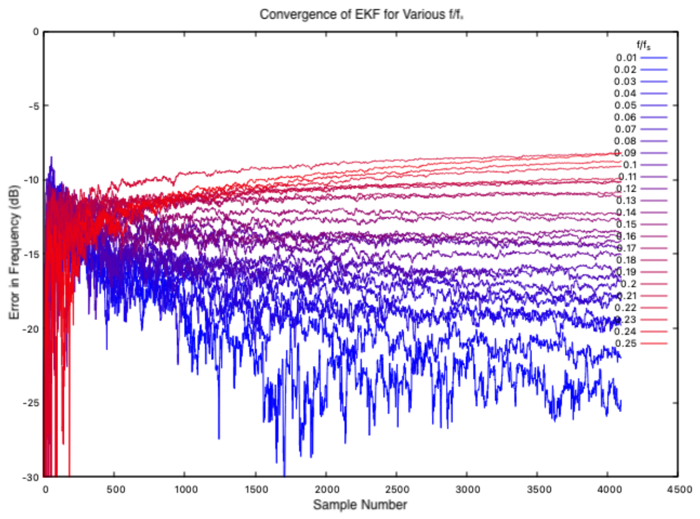


Figure 5.6: Convergence of the complex EKF for noisy data with perfect initial guess

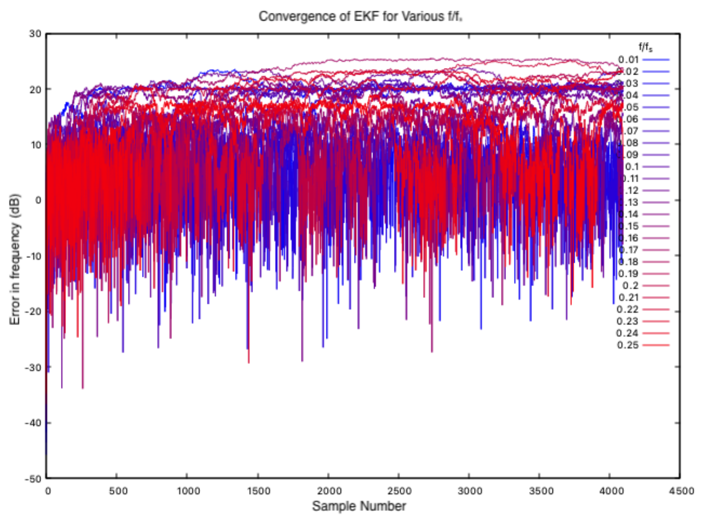


Figure 5.7: Convergence of the real EKF for noisy data with perfect initial guess

The pattern across the various ratios of $\frac{f}{f_s}$ is similar to the test class with noiseless data, seen in Figure 5.2. The difference between the classes is the addition of noise, which causes noise in the estimate of the filter, but does not stop it from converging for all ratios. The same cannot be said for the real EKF, as shown in Figure 5.7. The filter generally does not converge for the various ratios of $\frac{f}{f_s}$, which can be seen by the extreme amounts of variation in the estimated error.

5.3.4 Noisy Data With Imperfect Initial Guess

Once again, a similar pattern occurs for this test class for the complex EKF as its noiseless counterpart, as shown in Figure 5.8.

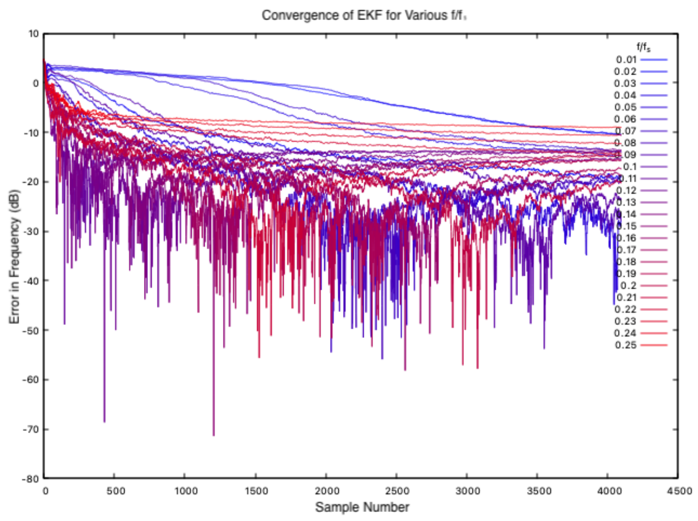


Figure 5.8: Convergence of the complex EKF for noisy data with imperfect initial guess

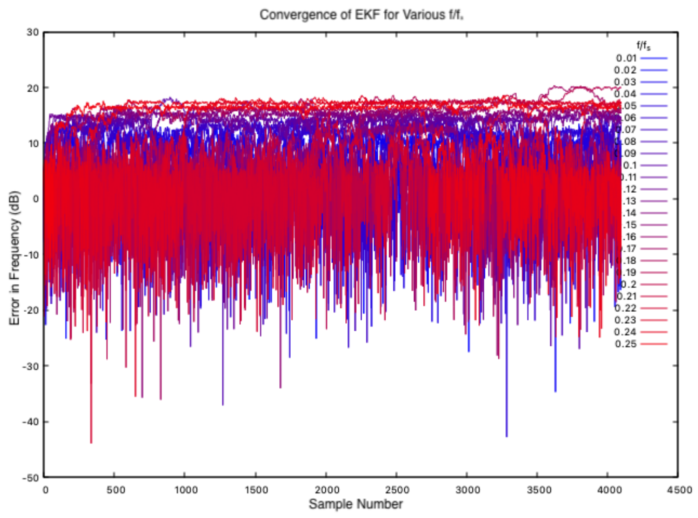


Figure 5.9: Convergence of the real EKF for noisy data with imperfect initial guess

The complex filter still converges for all ratios, but also has the addition of noise affecting the estimation errors. In the case of the real EKF shown in Figure 5.9, the

EKF does not converge. Thus, it is clear that the addition of noise causes the real EKF to diverge.

5.4 Estimation Error

With convergence results established, the estimation error is examined to determine the $\frac{f}{f_s}$ ratios that minimize it. As previously mentioned, the benchmark for this error is the Fourier Transform (FT) method. As noted in [30], amplitude tracking follows a similar error pattern to frequency, so only frequency tracking will be presented. All plots show the final error in estimation after 5000 samples.

5.4.1 Noiseless Data With Perfect Initial Guess

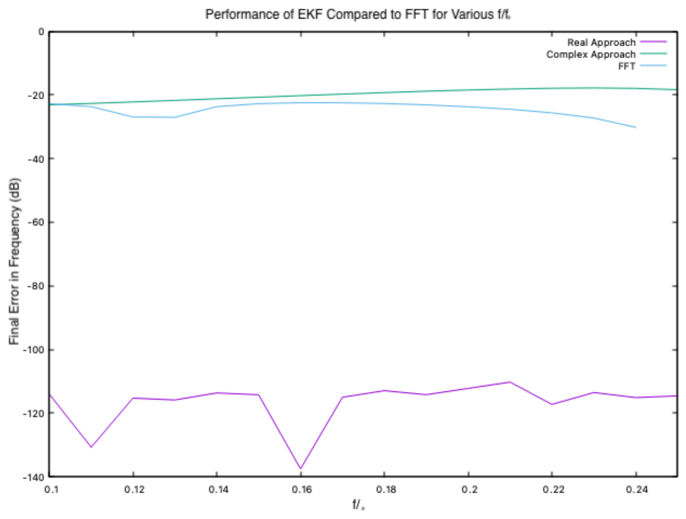


Figure 5.10: Performance of the EKF and FT compared for noiseless data with perfect initial guess

As shown in Figure 5.10, the real EKF outperforms both the complex EKF and the FT. However, it should be noted that at $\frac{f}{f_s} = 0.25$, the FT method yields an error so small that the error is undefined when converted to dB and cannot be represented on the plot. It is also significant to note that the complex EKF is comparable in performance to the FT.

5.4.2 Noiseless Data With Imperfect Initial Guess

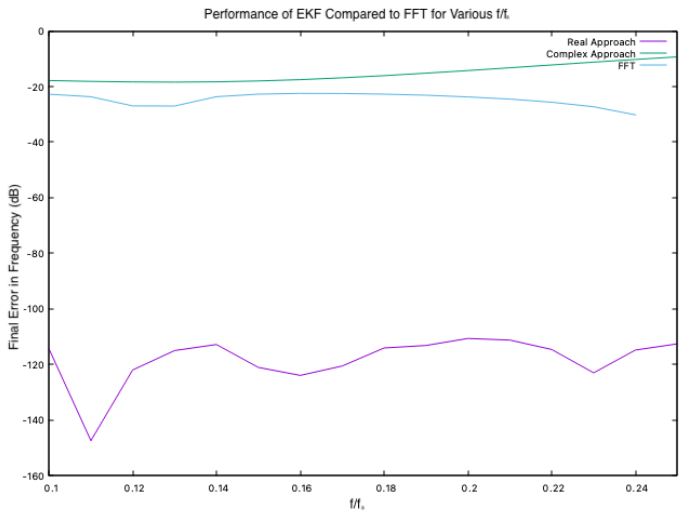


Figure 5.11: Performance of the EKF and FT compared for noiseless data with imperfect initial guess

With an imperfect initialization demonstrated in Figure 5.11, the real EKF significantly outperforms both the complex EKF and the FT, apart from at $\frac{f}{f_s} = 0.25$, where the FT is perfect. Thus, we can say that the systematic error in the real EKF when there is no noise in the signal is the smallest.

5.4.3 Noisy Data With Perfect Initial Guess

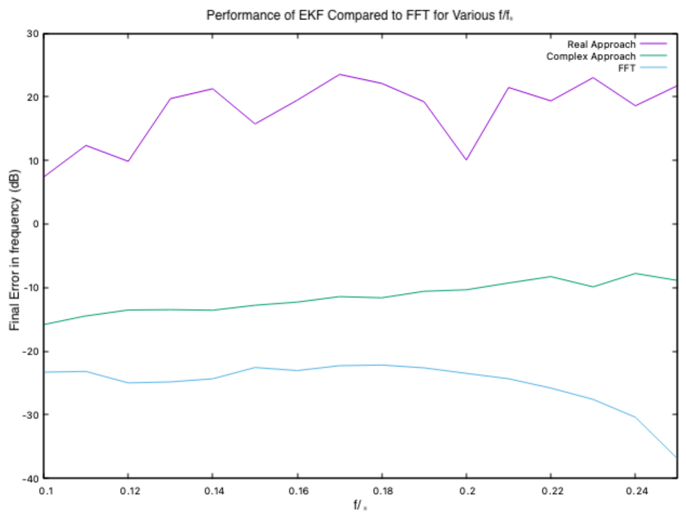


Figure 5.12: Performance of the EKF and FT compared for noisy data with perfect initial guess

With noise added to the signal, the results are different as displayed in Figure 5.12. As pointed out in the convergence section, the real EKF does not converge, so the final errors are not considered. On the other hand, the complex EKF maintains an error comparable to the FT method across most ratios of $\frac{f}{f_s}$. The FT method does outperform the complex EKF more for larger ratios, culminating in a very small error at $\frac{f}{f_s} = 0.25$ as before.

5.4.4 Noisy Data With Imperfect Initial Guess

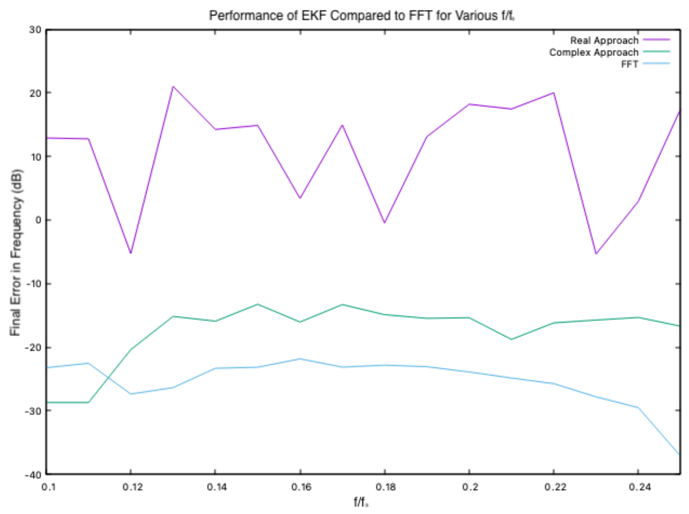


Figure 5.13: Performance of the EKF and FT compared for noisy data with imperfect initial guess

Imperfect initialization produces comparable results to the perfect initial guess case seen in Figure 5.12, especially at lower ratios of $\frac{f}{f_s}$. This can be seen in Figure 5.13. For appropriate ratios, the complex EKF is once again comparable to the FT method, but the FT method outperforms it across all frequencies.

5.5 Real Data

Fortunately, we have access to real data acquired from periodic machinery in the field. Figure 5.14 shows an example of such signal.

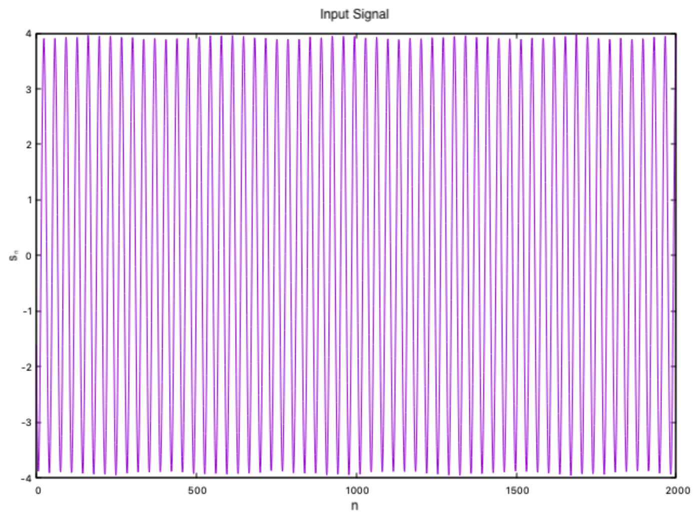
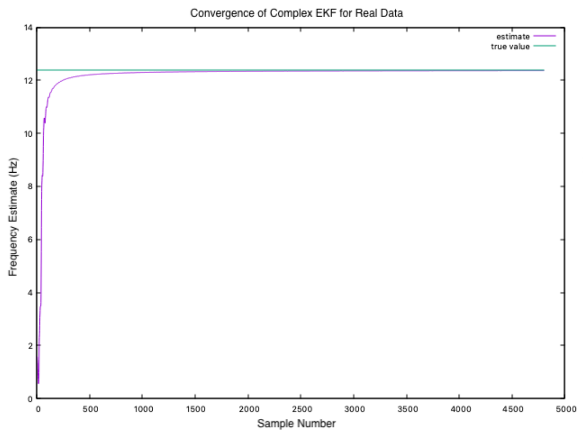
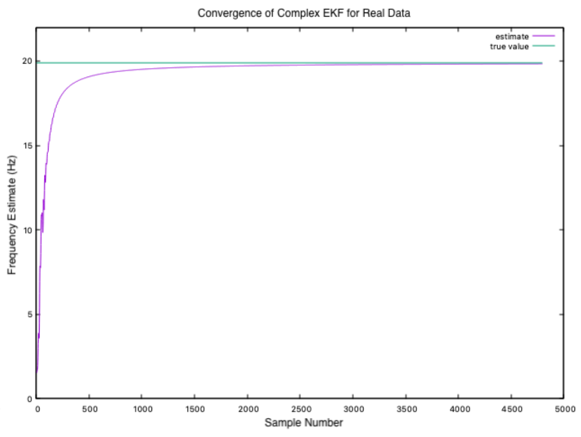


Figure 5.14: Real data acquired from industry partners

Since the performance of the filter has already been established in simulated cases shown in previous sections, only an analysis of the filter's accuracy is necessary at this stage to show that it is also effective in practical applications. Figure 5.15 shows the complex EKF's estimate of frequency for two signals, where their true frequency is already known. These plots made clear the effectiveness of the filter in a practical context.



(a) Complex EKF estimate of a signal with a true frequency of 12.37 Hz



(b) Complex EKF estimate of a signal with a true frequency of 19.89 Hz

Figure 5.15: Complex EKF estimate of real data

5.6 Summary

Summary table 5.1 concisely shows the performance of the three methods evaluated for each case presented above.

Test class	**Real EKF**	**Complex EKF**	**FFT**
Noiseless with perfect initial guess	Converges	Converges	-
	Best estimation error	Good estimation error	Good estimation error
Noiseless with imperfect initial guess	Converges	Converges	-
	Best estimation error	Good estimation error	Good estimation error
Noisy with perfect initial guess	Diverges	Converges	-
		Good estimation error	Best estimation error
Noisy with imperfect initial guess	Diverges	Converges	-
		Good estimation error	Best estimation error

Table 5.1: Summary of Results for EKF Performance.

Chapter 6

Conclusion

6.1 Concluding Remarks

The Extended Kalman Filter provides a proven and efficient way of estimating the parameters of a sinusoidal signal with additive Gaussian noise in real-time. As shown in this thesis, there are various ways to model the same signal for use in the EKF. Specifically, we presented two models of sinusoidal signals for state estimation. The first model uses real variables to represent a sinusoidal signal and has been previously discussed in the literature. The second model presented is, to our knowledge, a novel approach that uses complex variables in the EKF model. These models were evaluated for 5 different test classes, and evaluations were based on 2 factors: convergence and final estimation error. For each case, the ratio of signal frequency to the sampling frequency, or $\frac{f}{f_s}$, proved to be of great importance. Signals ranging from $\frac{f}{f_s} = 0.01$ to $\frac{f}{f_s} = 0.25$ were evaluated. In general, for signals with no measurement noise, the real EKF outperformed the complex EKF, though both showed good performance with very low error. An interesting structure appeared in the convergence of the complex

EKF: lower ratios of $\frac{f}{f_s}$ showed slower convergence but smaller estimation error, while higher ratios of $\frac{f}{f_s}$ converged much quicker with a larger estimation error. When the estimation error of the EKF methods was compared to using the Discrete Fourier Transform for these test classes, the real EKF outperformed both the complex EKF and the DFT, which were comparable over the range of $\frac{f}{f_s}$. A significant amount of noise was present in test classes with noise in the measurement signal. In these test classes, the complex EKF still converged and displayed the same structure present in the noiseless test classes. However, the real EKF could not converge. When compared to the DFT for estimation error, the complex EKF had comparable performance and even outperformed the DFT in some cases. Evaluation of the complex EKF on real data also showed its effectiveness for practical use.

6.2 Future Work

This novel approach to using the EKF for state estimation of sinusoidal signals could benefit from further research. Some recommendations for areas of further research are listed below:

- Analysis of the numerical stability of the proposed model,
- Further investigation of error propagation in complex EKF,
- Possible redundancies or structures in the EKF equations that can improve the computational efficiency of the algorithm.

References

[1] Gopi Krishna Allu. Estimating the parameters of exponentially damped sinusoids in noise. *University of Rhode Island, Kingston, RU, Tech. Rep*, 2003.

[2] Matteo Bertocco, Carlo Offelli, and Dario Petri. Analysis of damped sinusoidal signals via a frequency-domain interpolation algorithm. *IEEE Transactions on Instrumentation and measurement*, 43(2):245–250, 1994.

[3] JW Cooley, P Lewis, and P Welch. The finite fourier transform. *IEEE Transactions on audio and electroacoustics*, 17(2):77–85, 1969.

[4] Krzysztof Duda. Accurate, guaranteed stable, sliding discrete fourier transform [dsp tips & tricks]. *IEEE Signal Processing Magazine*, 27(6):124–127, 2010.

[5] Krzysztof Duda, Leszek B Magalas, Mariusz Majewski, and Tomasz P Zielinski. Dft-based estimation of damped oscillation parameters in low-frequency mechanical spectroscopy. *IEEE Transactions on Instrumentation and Measurement*, 60(11):3608–3618, 2011.

[6] Matteo Frigo and Steven G Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216–231, 2005.

[7] Hua Fu and Pooi-Yuen Kam. Phase-based, time-domain estimation of the frequency and phase of a single sinusoid in awgn—the role and applications of the additive observation phase noise model. *IEEE transactions on information theory*, 59(5):3175–3188, 2013.

[8] Arthur Gelb et al. *Applied optimal estimation.* MIT press, 1974.

[9] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[10] Fredric J Harris. On the use of windows for harmonic analysis with the discrete fourier transform. *Proceedings of the IEEE*, 66(1):51–83, 1978.

[11] Eric Jacobsen and Richard Lyons. The sliding DFT. *IEEE Signal Processing Magazine*, 20(2):74–80, 2003.

[12] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[13] Norman Levinson. The Wiener (root mean square) error criterion in filter design and prediction. *Journal of Mathematics and Physics*, 25(1-4):261–278, 1946.

[14] Frank L Lewis. *Optimal estimation: with an introduction to stochastic control theory.* Wiley New York, 1986.

[15] John Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, 1975.

[16] Peter S Maybeck. *Stochastic models, estimation, and control.* Academic press, 1982.

[17] Richard J Meinhold and Nozer D Singpurwalla. Understanding the Kalman filter. *The American Statistician*, 37(2):123–127, 1983.

[18] Mohsen Mojiri, Masoud Karimi-Ghartemani, and Alireza Bakhshai. Time-domain signal analysis using adaptive notch filter. *IEEE Transactions on Signal Processing*, 55(1):85–93, 2006.

[19] Alan V Oppenheim. *Discrete-time signal processing.* Pearson Education India, 1999.

[20] Soo-Chang Pei and Chien-Cheng Tseng. Real time cascade adaptive notch filter scheme for sinusoidal parameter estimation. *Signal processing*, 39(1-2):117–130, 1994.

[21] R Pintelon and J Schoukens. System identification: A frequency domain approach ieee press. *Piscataway, NJ*, 2001.

[22] John G Proakis. *Digital signal processing: principles algorithms and applications.* Pearson Education India, 2001.

[23] Fengyong Qian, Shuhung Leung, Yuesheng Zhu, Waiki Wong, Derek Pao, and Winghong Lau. Damped sinusoidal signals parameter estimation in frequency domain. *Signal Processing*, 92(2):381–391, 2012.

[24] Julius Orion Smith. *Mathematics of the discrete Fourier transform (DFT): with audio applications.* Julius Smith, 2007.

[25] Muhittin Uener. Frequency, amplitude, and phase tracking of nonsinusoidal signal in noise with extended Kalman filter. Technical report, Naval Postgraduate School Monterey CA, 1991.

[26] Paul F Velleman. Definition and comparison of robust nonlinear data smoothing algorithms. *Journal of the American Statistical Association*, 75(371):609–615, 1980.

[27] Greg Welch, Gary Bishop, et al. An introduction to the Kalman filter. 1995.

[28] Yegui Xiao and Yoshiaki Tadokoro. LMS-based notch filter for the estimation of sinusoidal signals in noise. *Signal Processing*, 46(2):223–231, 1995.

[29] I Yoshida, T Sugai, S Tani, M Motegi, K Minamida, and H Hayakawa. Automation of internal friction measurement apparatus of inverted torsion pendulum type. *Journal of Physics E: Scientific Instruments*, 14(10):1201, 1981.

[30] Tomasz Piotr Zieliński and Krzysztof Duda. Frequency and damping estimation methods - an overview. *Metrology and Measurement Systems*, 18(4):505–528, 2011.

Part II

Synthesis of Switched System Controllers using Dynamic Programming

Abstract

Research on hybrid systems has seen a large growth in interest in recent years. This is largely due to the increase of natural systems where discrete mode dynamics interact with continuous state dynamics. Switched systems are a subclass of hybrid systems that restrict their definition to continuous dynamic systems that interact with discrete switching events. Controller synthesis for such systems is no trivial task. Given the current trend in Artificial Intelligence and Machine Learning approaches, Dynamic Programming is explored as a means to approximate optimal control policies for switched systems. Discussions of discretization of the system's state space are presented, followed by a high-level overview of an algorithm that leverages Dynamic Programming to find the approximated optimal control policies. Finally, the algorithm is applied to several examples to demonstrate its effectiveness.

Chapter 1

Introduction

Research on hybrid control systems has had considerable growth in recent times. This is largely because control systems have become much more complex, often consisting of many smaller subsystems, modes, and computational devices [21]. Thus, in many cases, it is natural to represent a control system using a hybrid model where interactions between continuous state dynamics and discrete mode dynamics can be accurately captured. For example, it is shown in [3] that classical models of a four-stroke gasoline engine often rely on approximations to represent the system as an entirely continuous one, and that a hybrid model is a much more natural representation of the engine. Due to their general nature, hybrid models can also be used to break down complex systems into smaller subsystems, making the overall control design problem simpler for each subsystem.

Owing to this increased interest in hybrid systems, establishing control schemes is an important topic discussed today in literature. Realizing control schemes for hybrid systems is much more complicated than traditional control approaches where we only consider the state dynamics or the mode dynamics. This is because frameworks that describe hybrid systems are quite general and finding a control strategy that applies to any system that can be described by a hybrid model is quite difficult. For this reason, we choose to focus on specific subclasses of hybrid systems called switched systems, where discrete events that occur within the system cause transitions in the system mode. Specifically, we are interested in the optimal control problem for such systems. Despite the relative novelty of switched systems within the area of control theory, there exists plenty of literature regarding optimal control of switched systems. For example, [22–24, 26] discuss the extension of the well-known Maximum Principle to switched systems to achieve optimal control.

Given the growing interest in Artificial Intelligence and machine learning, we are particularly interested in using Dynamic Programming approaches to achieve optimal control of switched systems. Specifically, we are interested in producing an algorithm to approximate an optimal control policy for a switched system. Hence, the main contributions of this work are two-fold: 1) experiments into an approximation of optimal control policies for switched systems using dynamic programming approaches, and 2) validation of the aforementioned experiments through simulation using classical examples from the literature. An inline literature review is incorporated throughout.

Chapter 2

Hybrid Systems

Hybrid systems involve interaction between both continuous-valued and discrete variables. The evolution of such systems typically depends on the interaction between all variables. This means that the equations of motion that govern the evolution contain a mixture of discrete-variable or *mode* dynamics, and continuous-variable or *state* dynamics. The mode and state dynamics interact at "event" times, often when the continuous state reaches a prescribed set in the continuous state space [7].

2.1 Background

Studies in hybrid systems are generally broken down into four categories:

- **Modeling:** How to precisely model the behaviour of hybrid systems.
- **Analysis:** Development of tools to simulate, analyze, and verify hybrid systems. Issues such as well-posedness, simulation, and correctness are considered [11]. HyTech is a well-known software for checking such properties of

hybrid systems [1].

- **Control:** Synthesis of hybrid controllers to achieve given performance goals for a hybrid system.
- **Design:** Researching new high-level frameworks to improve modelling, analysis and control of hybrid systems.

[7, 27] give a comprehensive overview of these categories, and we would recommend reading them to gain a deeper background. In this thesis, we are only interested in the control of a subclass of hybrid systems. The goal is to synthesize controllers that use both discrete and continuous control inputs to control a given hybrid system. Approaches to such control established in the literature will be presented later in this section.

There are many examples of hybrid systems in the real world that are studied extensively in the literature. The following are a few:

- computer disk drives [13],
- robotic systems [18],
- transmissions and stepper motors [9],
- modern flight control systems [20],
- systems with hysteresis, relays or switches [29],
- automotive control applications [10]

These systems are quite widespread and diverse. Due to the diversity of settings relating to hybrid systems, there are generally four different approaches used in the literature to study them, summarized in [7]:

- **Aggregation** focuses on the mode dynamics of the system while ignoring the state dynamics [2],
- **Continuation** ignores the mode dynamics by augmenting the state dynamics to simulate the behaviour of the mode dynamics [8],
- **Automata approach** views the hybrid system as a network of interfacing automata. This approach was fully established by Nerode and Kohn in [17],
- **Systems approach** views the hybrid system as a set of interacting dynamical systems [7].

The different approaches listed above allow researchers to study different aspects, properties, and behaviours of hybrid systems. The choice of approach depends heavily on the overall goal of the research or application. In our research, we take the systems approach of study.

To establish a general framework for describing hybrid systems, a few concepts must be discussed. Namely, the state dynamics and mode dynamics must be formally defined. This is examined in the next sections.

2.2 State Dynamics

To provide the setup for hybrid systems, we must first define the state dynamics. A continuous state-space system is comprised of a set of state variables $x \in \mathcal{X} \subseteq \mathbb{R}^n$, and a set of external inputs $u \in \mathcal{U} \subseteq \mathbb{R}^m$. These are related to each other by a set of differential equations taking the form

$$\dot{x} = f(x, u) \tag{2.2.1}$$

$$y = h(x) \tag{2.2.2}$$

where y is the system output.

The state dynamics of the system may also be approximated by a discrete-time set of difference equations using a variety of methods, including exact discretization and Runge-Kutta Methods. The most basic method for solving ordinary differential equations with a given initial value which belongs to the Runge-Kutta family is the Euler Method [4]. In this thesis, we will not deal with complex differential equations, so the Euler Method is sufficient for our purposes. Given an ODE

$$y'(t) = f(t, y(t)), \qquad y(0) = y_0$$

the one-step solution to the ODE is

$$y_{n+1} = y_n + hf(t_n, y_n)$$

where h is a sufficiently small step size and

$$t_n = t_0 + nh.$$

Once discretized, the state dynamics of the system take the form

$$x(n+1) = f(x(n), u(n)) \tag{2.2.3}$$

$$y(n) = h(x(n)). \tag{2.2.4}$$

2.3 Mode Dynamics

Mode dynamics are often described using finite automaton. Finite automaton are described by the triple

$$(Q, A, \delta)$$

where $Q = \{q_0, q_1, ..., q_K\}$ is a finite set of K *modes*, A is a finite set of symbols, and $\delta : (Q \times A) \to Q$ is a transition function describing the next state given a symbol. FSMs are often visualized using graphs, where the discrete states described by Q are the vertices, and the edges given by δ. For example, this FSM

$$\begin{tikzcd}
q_0 \arrow[r, bend left, "a"] & q_1 \arrow[l, bend left, "b"] \arrow[d, bend right, "c"'] \\
& q_2 \arrow[u, bend right, "a"'] \arrow[ul, bend left, "c"]
\end{tikzcd}$$

describes a triple

$$(Q, A, \delta) \tag{2.3.1}$$

where $Q = \{q_0, q_1, q_2\}$, $A = \{a, b, c\}$ and, for example, $\delta(q_0, a) = q_1$.

2.4 General Hybrid Systems

Given the generality of the dynamics defined by (2.2.4) and (2.3.1), as well as the relative novelty of hybrid systems in the literature, there are many different ways a hybrid system may be defined. However, one definition is widely used in literature such as [7, 27, 28]. This definition is also one used by well-known hybrid systems analysis tools such as HyTech [1].

A hybrid system may be described by a tuple

$$H = (Q, \mathcal{X}, Y, F, \delta, I, G, A)$$

where $Q = \{q_1, ..., q_K\}$ is the set of system modes, $\mathcal{X}$ is the set of allowable states that satisfy the system constraints, Y is the set of continuous system outputs, $F = \{f_{q_1}, ..., f_{q_K}\}$ is the set of differential equations that describe the state dynamics of the system, δ is a transition function describing the discrete dynamics of the system, $I = \{i_{q_1}, ..., i_{q_K}\}$ is a set of invariants that must remain true while the system is in a given mode, G is a guard function that must be true for a discrete transition to occur and A is a transition action that describes the continuous state of a hybrid system after a mode switch.

Chapter 3

Switched Systems

By construction, hybrid control systems are very general. Therefore, obtaining practical algorithms to synthesize controllers for them, or proving the existence of solutions for any system that can be described using the general hybrid system framework in the first place has proven to be difficult [27]. Therefore, we must restrict the definition to a meaningful but computationally tractable subclass.

3.1 Switched Systems

Switched systems are a class of hybrid systems with dynamic systems representing the state dynamics and discrete switching events representing the mode dynamics [16]. Switching events are represented using a switching signal σ which specifies the mode of the system at a given time or state. Switched systems are classified in many ways, but the most important classifications that pertain to this thesis are as follows: **controlled switched systems** choose the mode q at any given time based on a "higher process", such as a supervisor or human operator, while switching events in **autonomous switched systems** may be a function of time or state [7]. The definition of the switching signal for autonomous systems depends on the type of autonomous switched system in question, as it may be a function of state or time. This is discussed below.

In **state-dependent autonomous switched systems**, switching signals are a function of the system state and occur when the system state reaches a *switching surface*, which is a surface that exists in $\mathcal{R}^n$. The switching signal $\sigma_s(x)$ for such systems maps the system state to a mode of the system, and is defined as

$$\sigma_s : \mathbb{R}^n \to Q.$$

Switching signals of **time-dependent autonomous switched systems** are a function of time for systems with continuous state dynamics and step for systems with discrete state dynamics. The switching signal $\sigma_n(n)$ for time-dependent switched systems in discrete-time defines the mode of the system at all steps n in the system evolution. Thus, it is defined as

$$\sigma_n : \mathbb{N} \to Q.$$

Assumption: In this thesis, we will focus on autonomous switched systems with time-dependent switching, henceforth referred to as TDSS. State dynamics for TDSS are modelled by

$$x(n+1) = f_q(x(n)) \tag{3.1.1}$$

$$y(n) = h_q(x(n)). \tag{3.1.2}$$

The triple

$$H = (Q, \mathcal{X}, F) \tag{3.1.3}$$

fully defines the TDSS, with parameters previously established in Section 2.4. As we can see, switched system models are much simpler and focus mainly on the system's state dynamics.

3.2 Controllers

In order to discuss controllers for switched systems of interest, we must define the notion of a solution for a TDSS. Given an initial state x_0, a solution of a TDSS with discrete state equations is a time-dependent switching signal $\sigma(n)$. For a solution to be proper, it must also satisfy the following:

- $x(0) = x_0$,
- $x(n+1) = f_{\sigma(n)}(x(n))$,
- System constraints must be satisfied.

Using a proper solution to the system means the system state evolves as follows:

$$x(0) \xrightarrow[f_{\sigma(0)}]{} x(1) \xrightarrow[f_{\sigma(1)}]{} x(2) \xrightarrow[f_{\sigma(2)}]{} ...$$

Given the above definition of a proper solution, controllers are defined as strategies that generate proper solutions to a system to achieve prescribed control objectives. Given any allowable initial state x_0, a controller $C : \mathcal{X} \to Q$ will produce a unique proper solution to the system. In the case of a TDSS, this is a switching signal $\sigma(n)$.

3.3 The TDSS Optimal Control Problem

In order to describe the optimal control problem for a TDSS, a performance measure J must be defined. J, also called the cost function, is described as a sum representing the cumulative cost over a trajectory, and a term representing the terminal penalty. Given a switching signal $\sigma(n)$, the cost function over a a prediction horizon N is

$$J_N = \sum_{n=0}^{N-1} l_{\sigma(n)}(x(n) - x_f) + \phi(x(N) - x_f) \tag{3.3.1}$$

where x_f is the desired final state of the system, and constraints may be present on the system state. Thus, the TDSS Optimal Control Problem is defined as follows:
Given a TDSS H, initial state x_0 and a prediction horizon N, find a controller $C(x)$ such that

1. *The state constraints are satisfied*
2. *The performance measure J_N is minimized over switching signal $\sigma_{opt}(\cdot)$*

If these conditions are satisfied, $C(x(n)) = \sigma_{opt}(n)$ is the optimal controller for this system.
Mathematically,

$$\underset{\sigma_{opt}(\cdot)}{argmin}\, J_N \text{ s.t. TDSS model H and initial state } x_0$$

3.4 Switched System Control Schemes

While switched systems are a relatively new field of study in control theory, various control schemes have already been established for them in literature. In [16], the author presents two common control schemes for switched systems: sliding mode control and hysteresis switching. In a sliding mode control scheme, the goal is to control the switched system by "sliding" along a switching surface S which is chosen to meet control objectives. However, as discussed by the author, this control scheme often leads to *chattering*, a phenomenon where the system successively switches between modes infinitely. Such behaviour is often undesirable in real-world systems, as it leads to excessive physical device wear. Hysteresis switching aims to solve this problem by adding a hysteresis band around the switching surface, such that the controller simply keeps the system within the hysteresis band, greatly decreasing instances of chattering. Other well-known control schemes that may be applied to switched systems such as relay control (sometimes known as on-off control), gain scheduling and fuzzy control, are discussed in more detail in [27].

More modern approaches to switched control systems have also been used in literature. The authors of [15] examine the use of Model Predictive Control (MPC) to solve the switched system. In general, MPC is considered an optimization problem, where the optimal control sequence is found over a sliding horizon. This makes it a good candidate for controlling switched systems since the mode switching is included in the optimization problem in addition to the optimization of the continuous control input. For background on MPC, readers are encouraged to review [12].

Another approach is discussed in [30] which adapts the Linear Quadratic Regulator for systems with discrete state dynamics to switched systems, named the

"Discrete-time Switched Linear Quadratic Regulator" (DSLQR). As in the standard LQR controller, the DSLQR only applies to Linear Time-Invariant (LTI) systems with quadratic cost functions. The key concept introduced in the DSLQR is the Switched Riccati Set (SRS), which is a set of matrices describing the Discrete Riccati Equations of the switched system for all possible trajectories of the mode dynamics over a finite time horizon. As noted in the paper, the SRS allows us to use Dynamic Programming principles to find the optimal controller. Issues of stability and implementation are also discussed.

In [21], the author proposes an optimal controller for switched systems. The proposed approach is not limited to LTI systems or specific cost functions. Multiple possible cost functions are proposed. The proposed approach is also based on Dynamic Programming principles, where the "Hybrid Bellman Equation" is the recursive relation introduced in this work to numerically approximate the solution of the optimal control problem for switched systems. This relation is directly based on the Bellman Equation introduced in [5], but adapted to switched systems. Through optimization of the Hybrid Bellman Equation, as well as a discretization of parts of the continuous state space where switches occur, a controller is found.

Chapter 4

Dynamic Programming

Dynamic Programming (DP) has long been established as a key concept for solving optimal control problems. While we will present DP to derive the solution for the TDSS Optimal Control Problem, it has wide-ranging applications for solving multi-stage decision problems [19], where sequential decisions must be made to determine a solution for the overall problem.

4.1 Principle of Optimality

The key concept behind DP is the Principle of Optimality, presented by Bellman in his work [5] as follows:

An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regards to the state resulting from the first decision.

Disregarding the exact definition of a policy and optimal policy, for now, this concept is quite intuitive, and can be demonstrated through a simple example of a multi-stage process.

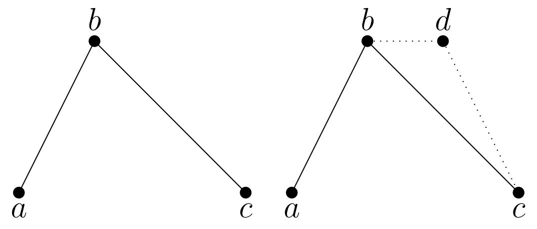


Figure 4.16: (a): Optimal path from a to c (b) Two possible paths from b to c

Suppose that the first decision made in the system described by Figure 4.16 results in the segment $a - b$, with cost J_{ab}. Now assume that the rest of the decision making results in the segment $b - c$, with cost J_{bc}. The optimal path from a to c is then $a - b - c$, with overall cost $J^*_{ac} = J_{ab} + J_{bc}$. This means that the optimal path from b to c is the segment $b - c$. If there exists a segment, for example, $b - d - c$ with $J_{bdc} < J_{bc}$, then $J_{ab} + J_{bdc} < J_{ab} + J_{bc} = J^*_{ac}$. This cannot be true, since we know that the optimal path from a to c is the segment $a - b - c$.

4.2 The cost-to-go function

Drawing on the principle of optimality, we derive a recurrence relation to solve the TDSS Optimal Control problem. First, we introduce the *cost-to-go* function V. The cost-to-go function describes the cost of going from one state $x(n)$ to the next state $x(n + 1)$ while satisfying the system dynamics and the state constraints. To be more precise, the cost-to-go function $V^\pi(x)$ is given by

$$V^\pi(x) = l_{\pi(x)}(x) + V^\pi(f_{\pi(x)}(x)). \tag{4.2.1}$$

The policy $\pi : \mathbb{R}^n \to Q$ is a function that maps any system state x to a mode q. This recurrence relation is derived from the *Bellman Equation* [5], and allows us to numerically solve the optimal control problem.

We define Π as the set of all possible policies for a given system. The performance of different policies $\pi \in \Pi$ relative to each other may be evaluated based on the resultant cost-to-go function. For example, given two policies π_1, $\pi_2 \in \Pi$, π_1 is better for some state x if

$$V^{\pi_1}(x) < V^{\pi_2}(x).$$

The optimal policy $\pi^* \in \Pi$ is one where

$$V^{\pi^*}(x) \le V^{\pi}(x) \quad \forall x \in \mathcal{X}, \pi \in \Pi.$$

There are several methods established in the literature refine the cost-to-go function (4.2.1). Here, we choose to use the Value Iteration algorithm to do the refinement, which is well-established as an approach to DP in the context of Optimal Control [6]. First, V is initialized using the terminal cost defined in cost function J as

$$V^0(x) = \phi(x) \quad \forall x \in \mathcal{X}.$$

Following this,

$$V^{n+1}(x) = \min_q (l_q(x) + V^n(f_q(x))) \quad \forall x \in \mathcal{X},$$

where $q \in Q$, and $n \in \{0, 1, ..., N-1\}$. Once a point is reached where the cost-to-go function cannot be improved for any state, it has converged. We can then use a "greedy scheme" to find the optimal policy π^*. A greedy scheme is one where we only minimize the cost-to-go value of the immediate state evolution according to the given mode. This scheme is applied by

$$\pi^*(x) = \underset{q}{argmin}\, V(f_q(x)).$$

Relating the optimal policy back to controllers defined in Section 3, the optimal controller will generate a switching signal $\sigma_{opt}(n)$ such that

$$\sigma_{opt}(n) = \pi^*(x(n)).$$

Chapter 5

Computational Approximation of the Optimal Policy

This section aims to tackle the issue of computing an approximation of the optimal control policy using dynamic programming. First, issues relating to the direct application of dynamic programming will be discussed. Next, we will address the problem of discretizing the state space. Finally, we will present the implementation of the proposed method.

5.1 Problems with Practical Implementation of DP for Control

In the previous section, we introduced equation (4.2.1) and stated that it can be used to define an optimal policy π^*. Switched systems such as the ones used in this thesis lend themselves nicely to dynamic programming. The discrete controls inputs can be seen as discrete actions, and finding the optimal policy simply involves finding the best mode, or action, for the system at any given state. In practice, this is not straightforward. For example, in [25] dynamic programming is used to solve the optimal control problem, and numerical issues relating to the boundary lines created by system constraints are discussed. The following are two problems that make the implementation challenging:

- **State explosion:** The implementation becomes computationally intractable with increasing states and state spaces, as the problem grows exponentially.
- **State space discretization:** The Bellman Equation is based on discrete processes, so the state space for continuous systems must be segmented such that discrete approaches can be applied.

5.2 Discretization of the State Space

To discretize the state space for the purpose of approximating the cost-to-go function V, we segment the space into a map. By doing so, every state is mapped to a specific cell. Many system states may be mapped to the same cell in this setup, but the same state may not be mapped to multiple different cells. As an example, assume x_1 is a state variable constrained to the interval $[0, 10)$. If we chose to segment the state space for x_1 into 2 even segments, all states $x_1 \in [0, 5)$ would map to the first segment, and $x_1 \in [5, 10)$ would map to the second segment. Notice that the number of segments each state variable's state space is split into may not be uniform, which can be of great use. For example, different coordinate systems such as Cartesian, radial, and polar coordinates may each be used for different state variables. This means that an independent analysis of each state variables's limits and the appropriate number of segments can be performed, which is greatly beneficial.

A key aspect of this approach that must be examined is the state variable constraints. Since the state space must be constrained to a finite number of segments to use dynamic programming approaches, the state variables' state space is limited to a bounded box.

The limits of the bounded box heavily depend on the problem and directly affect the number of segments the state space is split into. Having too few segments results in too coarse of a discretization of the system dynamics, as jumps between segments happen less frequently. On the other hand, making the segments too fine means computations become computationally intractable.

Chapter 6

Implementation

In this section, the computation of the approximated optimal policy π^* will be discussed. Only an approximation of π^* can be found due to the discretization of the state space. To find the exact optimal policy, the cost-to-go function $V(x)$ would have to converge for every possible state x in the state space, which is computationally intractable. By creating a map of the state space with a finite number of segments, and choosing a *representative state* for each segment, we approximate the function $V(x)$ which in turn leads to an approximated optimal policy π^*. The representative state of a cell is used in computations to represent all states contained in the cell.

First, the implementation modules will be discussed. Following that, the most important parts of the implementation algorithm will be presented. The implementation of the controller synthesis algorithm was programmed in C++, with the linear algebra operations handled by the Eigen library [14].

6.1 Modules

There are several modules necessary for the implementation of the proposed controller synthesis algorithm:

- **Model:** This module contains the system model, which consists of the set F, cost functions l and ϕ, and the set Q as defined in the switched system tuple in Section 3. In some examples, such as the holonomic robot example provided later, this module also contains information about the system's environment, such as walls that must be avoided or other obstacles.

- **Model Solver:** This module performs the prediction of the future system state by solving the system's dynamic equations contained in the Model module, which aids in the exploration of the state space.

- **Approximated Map:** This module contains the discretized system state space and has helper functions that convert the representative states of the system to discrete cell indices and vice versa.

- **DP:** This is the key module in the implementation. It contains functions to apply equation (4.2.1) and to create the final policy, as well as the cost-to-go function V and the policy π.

6.2 Algorithm

While all modules are important to the implementation of the algorithm, only the DP module will be expanded on in this section. Going forward, the discretized state space will be referred to as the *map*.

The first step of the proposed algorithm is to initialize the map V. The variable s is a discrete index in the discretized state space, and s_{rep} is the representative state for the cell s in the map. Procedure $\Phi(s_{rep})$ is external to the module, and calculates the terminal cost Φ of s_{rep} according to the cost function J as defined in Section 3. MAX_COST is a constant value global to all modules and is a value large enough relative to the largest cost-to-go in the system that it may be considered an infinite value.

```
Inputs: $V$
Outputs: $V$
FUNCTION initialize_map
  FOR each $s$ in $V$
    IF $s$ satisfies constraint THEN
      $V[s] = \Phi(s_{rep})$
    ELSE
      $V[s]$ = MAX_COST
    END IF
  END FOR
END FUNCTION
```

Figure 6.17: The procedure to initialize the map

There are a few things to note about this procedure. The initial cost-to-go of any given cell depends on whether the cell satisfies the constraints or not. As shown above, if the cell does not satisfy the state constraints of the system, the cost-to-go of the cell is set to MAX_COST. Analytically, these cells would have infinite cost. By doing so, the system constraints are encoded directly in the cost-to-go function.

If the given cell does satisfy the system constraints, its value is set to the terminal cost of the cell according to the function J. Since the state space is discretized, the representative state of the cell is used to calculate the cost of the cell.

The next step is to use value iteration to refine the cost-to-go function. The procedures step_model(s_{rep}, q) and cost(s_{rep}, q) are imported from the Model Solver module and the Model module, respectively. step_model(s_{rep}, q) solves a single step of the dynamic model with the given mode using the set F and returns the new representative state the system is in, while cost(s_{rep}, q) returns the cost of performing the step according to l in the cost function J. The refine_map procedure takes the current V function as input and returns a new function V' as output which is the updated cost-to-go function. Variable s is a representative state of a cell as described earlier, and the variable q is a mode in the set Q.

```
Inputs: $V$
Outputs: $V'$
FUNCTION refine_map
  FOR all $s$ in $V$
    IF $V[s] \geq$ MAX_COST THEN
      $V'[s]$ = MAX_COST
    ELSE
      $q$ = some mode in $Q$
      $s'_{rep}$ = step_model($s_{rep}$, $q$)
      minimum_value = $V[s']$
      FOR all remaining $q$ in $Q$
        $s'_{rep}$ = step_model($s_{rep}$, $q$)
        value = cost($s_{rep}$, $q$) + $V[s']$
        IF value < minimum_value THEN
          minimum_value = value
        END IF
      END FOR
      $V'[s]$ = minimum_value
    END IF
  END FOR
END FUNCTION
```

Figure 6.18: The procedure to refine the map V

In this function, the single-step update using equation (4.2.1) is performed. First, we must check whether the current cell already has a value that is greater than or equal to MAX_COST. A cell that has such a value breaks the system's constraints and should not be refined. The rest of the function is straightforward: the future cost is evaluated for every possible mode the controller can choose, and the lowest future cost the system may incur is assigned as the value of the current cell. The procedure must choose one of the modes as the most viable. This is why the minimum_value is initialized to the value of some arbitrary mode in the set Q, as all modes are ultimately evaluated. This procedure is iterated for N iterations, where N is the prediction horizon for the system. If an infinite prediction horizon is used, the procedure is repeated until convergence of V.

Finally, once the cost-to-go function has been repeated for N iterations a control policy may be synthesized by computing the optimal policy based on the refined function as follows.

```
FUNCTION make_policy
  FOR all $s$ in $V$
    IF $V[s] \geq$ MAX_COST
      $\pi[s] = -1$
    ELSE
      $q$ = some mode in $Q$
      best_mode = $q$
      $s'_{rep}$ = step_model($s_{rep}$, $q$)
      minimum_value = $V[s']$
      FOR all remaining $q$ in $Q$
        $s'_{rep}$ = step_model($s_{rep}$, $q$)
        current_value = $V[s']$
        IF current_value < minimum_value
          minimum_value = current_value
          best_mode = $q$
        END IF
      END FOR
      IF minimum_value $\geq$ MAX_COST THEN
        $\pi[s] = -1$
      ELSE
        $\pi[s]$ = best_mode
      END IF
    END IF
  END FOR
END FUNCTION
```

Figure 6.19: The procedure to synthesize the approximated optimal control policy

Assigning a value of -1 to the policy is a simple way to mark cells that break the system constraints directly, as well as cells where the system has no choice but to take an action that leads to breaking the constraints.

Chapter 7

Examples

In this chapter, several examples are presented to show the effectiveness of the synthesized controller. First, a pendulum is used as an example of an exponentially stable system to demonstrate the effectiveness of the controller and how its behaviour is directly affected by the control objectives. The second example is a single-valve tank. This control system is a more complex system with physical constraints, so the point of the example is to show the behaviour of the synthesized controller for such systems. Finally, an example is presented to show a possible practical use for such a controller in the context of autonomous motion planning.

All the simulations in this section were done using the framework presented in the previous chapter and implemented using C++ and the Eigen library.

7.1 Pendulum

In this example, a simple pendulum with piecewise-constant discrete-valued control inputs is considered. The continuous state dynamics are defined by the following equations:

$$f_q(x_1, x_2, t) = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} x_2(t) \\ -x_1(t) - 1.2x_2(t) + u_q(t) \end{bmatrix}$$

Using the Euler approximation, the discretized state dynamics are:

$$f_q(x_1, x_2, n) = \begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} = \begin{bmatrix} x_1(n) + hx_2(n) \\ -hx_1(n) - (1 - 1.2h)x_2(n) + hu_q(n) \end{bmatrix}$$

where $h = 0.01$ is the discrete step size. To define the complete TDSS model H_p of the system, Q, F, and $\mathcal{X}$ must be established. This control system has three modes, thus the set of system modes is

$$Q = \{1, 2, 3\}.$$

The set of allowable control input values corresponding to the system modes are $\mathcal{U} = \{0, 20, -20\}$. F is the set

$$F = \{f_1(x_1, x_2, n), f_2(x_1, x_2, n), f_3(x_1, x_2, n)\}$$

where

$$f_1(x_1, x_2, n) = \begin{bmatrix} x_1(n) + 0.01x_2(n) \\ -0.01x_1(n) - 0.012x_2(n) \end{bmatrix}$$

$$f_2(x_1, x_2, n) = \begin{bmatrix} x_1(n) + 0.01x_2(n) \\ -0.01x_1(n) - 0.012x_2(n) + 0.2 \end{bmatrix}$$

$$f_3(x_1, x_2, n) = \begin{bmatrix} x_1(n) + 0.01x_2(n) \\ -0.01x_1(n) - 0.012x_2(n) - 0.2 \end{bmatrix}$$

The allowable state space for the system is defined by the set $\mathcal{X}$, where

$$\mathcal{X} = \{x \in \mathbb{R}^2 \mid -10 \leq x_1 \leq 10,\ -10 \leq x_2 \leq 10\}.$$

The control objective is to reach the desired final state $x_f = [0.0 \quad 0.0]^T$. Thus, the cost function J_N is

$$J_N = \sum_{n=0}^{N-1} x_1(n)^2 + 0.6x_2(n)^2 \ \ \forall q \in Q.$$

Choosing an arbitrary allowable initial state $x_0 = [5.0 \quad 0.0]^T$, the system was simulated without any control input to establish a baseline reference for its state dynamics. Figure 7.20 shows how the system behaves naturally. Since it is a stable system, it converges without the need for control input.

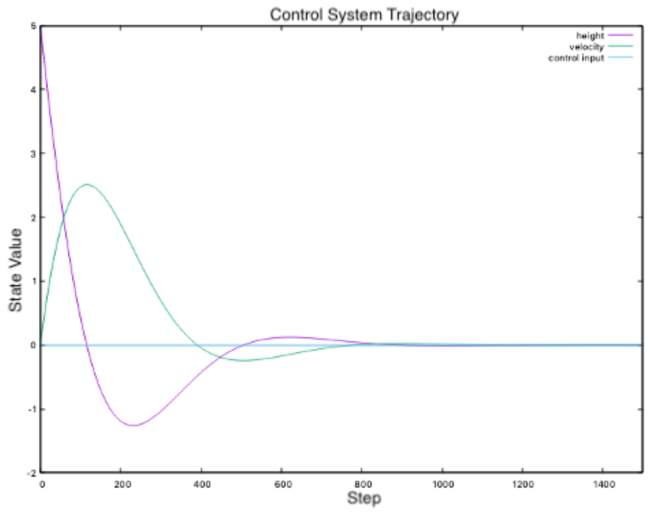


Figure 7.20: Uncontrolled behaviour of the pendulum control system

As mentioned in Section 5, since the state space must also be partitioned into discrete cells, a choice must be made regarding the size of the cells. In this case, the state space was partitioned into a 1024×1024 grid.

Having established all the definitions needed to solve the optimal control problem for the system, it was simulated using the synthesized controller. The fact that this system is a 2-dimensional system allows us to visualize the cost-to-go function and the resultant synthesized controller in a plot. First, the Bellman module previously discussed was used to learn the cost-to-go function V. Figure 7.21 shows the function V over the state space of the system.

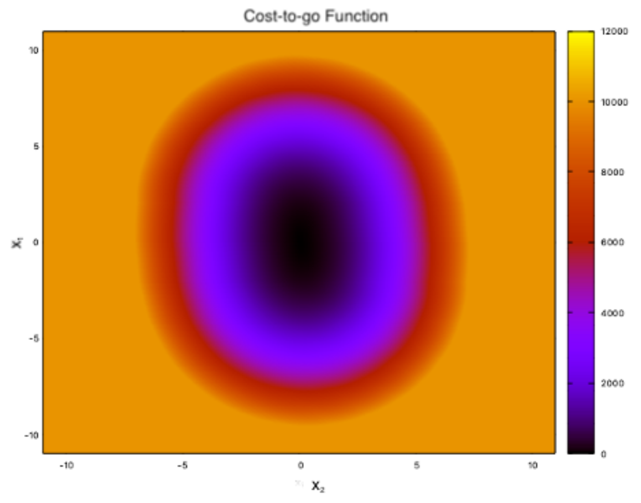


Figure 7.21: Cost-to-go function for the pendulum control system

This clearly shows the effect of the cost function used on the cost-to-go function for the system. In this case, the equal weighting in the cost function means there is a circle pattern in the cost-to-go function, where the closer the system gets to the desired final state, the lower the cost is. The resultant policy can also be visualized this way, as seen in Figure 7.22.

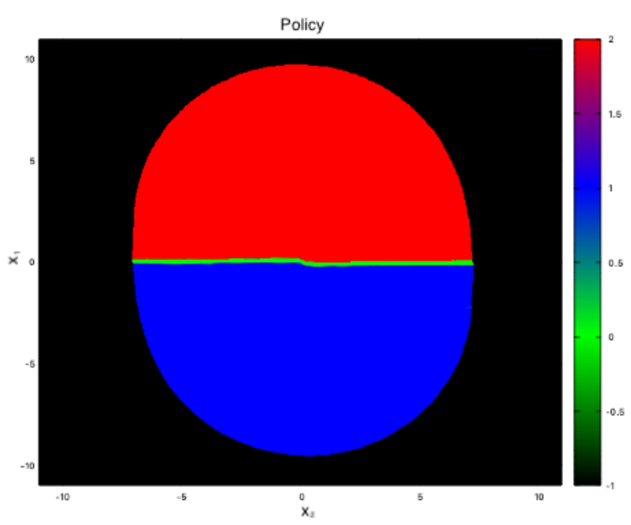
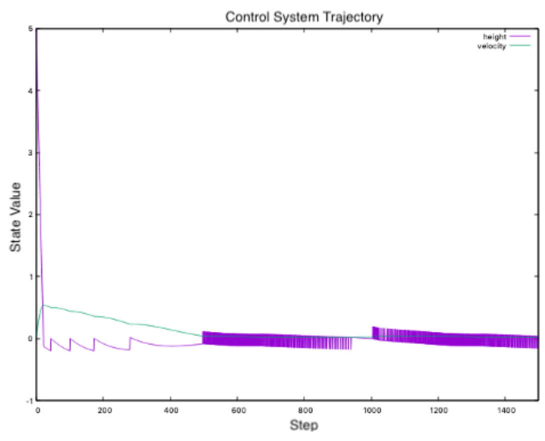
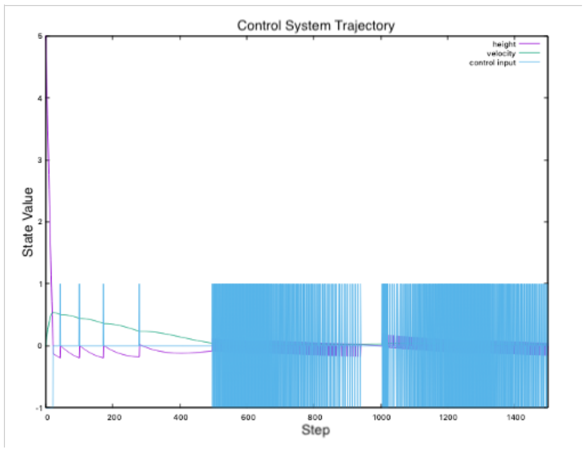


Figure 7.22: Policy for the pendulum control system

The black cells in the policy represent “bad states”. These are states that either directly break the system constraints or where it is impossible for the system to take any action which does not eventually lead to breaking the system constraints. Figure 7.22 also shows the phase portrait of the system overlaid on the policy. This shows that the controller does take actions that take the system to the desired cell. Finally, Figure 7.23 shows the system’s trajectory and control history.



(a) State trajectory



(b) State trajectory and control history

Figure 7.23: Trajectory of the pendulum control system with the synthesized controller

For clarity, there are two subfigures in Figure 7.23. Figure 7.23a only shows the trajectory of the control system, whereas Figure 7.23b shows the trajectory of the control system as well as the control history. These plots show that the system does converge with the synthesized controller, but they also show that there is significant chattering due to the discretization of the system's state space.

7.2 Single Valve Tank

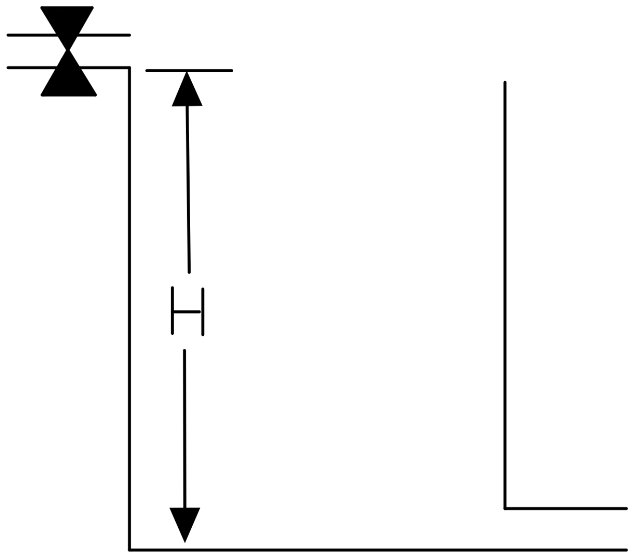


Figure 7.24: Diagram of the single valve tank control system

In the next example, a single-valve tank control system is simulated. The system, shown in Figure 7.24, consists of a tank with an inflow, which is controlled by a continuous valve, and an outflow. Continuous state dynamics for such a system are described by the following equations:

$$f_q(x_1, x_2, t) = \begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} -ax_1(t) + x_2(t) \\ u_q(t) \end{bmatrix}$$

Where x_1 is the height of the liquid in the tank, and x_2 is the rate of inflow into the tank. Once again, we are interested in the discrete state dynamics, so the Euler approximation is used to discretize the continuous dynamics as shown below:

$$f_q(x_1, x_2, n) = \begin{bmatrix} x_1(n+1) \\ x_2(n+1) \end{bmatrix} = \begin{bmatrix} (1 - ha)x_1(n) + hx_2(n) \\ x_2(n) + hu_q(n) \end{bmatrix}$$

We choose $h = 0.01$, and $a = 0.95$. To define the TDSS model H_t for the system, Q, F and $\mathcal{X}$ are defined. This control system has three modes, making the set of modes as follows:

$$Q = \{1, 2, 3\}$$

These modes correspond to the set of allowable inputs into the system $\mathcal{U} = \{0, 12, -12\}$. This means that the set F is described as:

$$F = \{f_1(x_1, x_2, n), f_2(x_1, x_2, n), f_3(x_1, x_2, n)\}$$

where

$$f_1(x_1, x_2, n) = \begin{bmatrix} 0.9905x_1(n) + 0.01x_2(n) \\ x_2(n) \end{bmatrix}$$

$$f_2(x_1, x_2, n) = \begin{bmatrix} 0.9905x_1(n) + 0.01x_2(n) \\ x_2(n) + 0.12 \end{bmatrix}$$

$$f_3(x_1, x_2, n) = \begin{bmatrix} 0.9905x_1(n) + 0.01x_2(n) \\ x_2(n) - 0.12 \end{bmatrix}$$

There are physical constraints for the system that must be satisfied, governed by the tank's characteristics and the valve. The tank may not overflow or underflow, and the size of the inflow valve affects the maximum amount of inflow there may be. Thus, the constrained allowable state space is the set

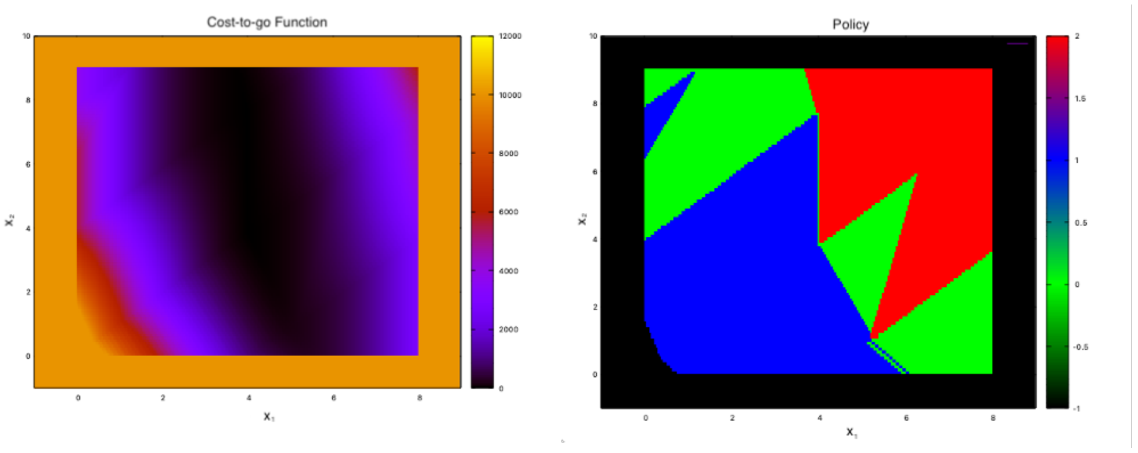
$$\mathcal{X} = \{x \in \mathbb{R}^2 \,|0 < x_1 < H,\ 0 \leq x_2 < 9\},$$

where $H = 8$ is the height of the tank.

The objective of the synthesized controller for this control system was to reach and maintain a desired liquid height r in the tank. In this case the desired height $r = 4$. The cost function J_N to represent this objective is

$$J_N = \sum_{n=0}^{N-1} (x_1(n) - r)^2 \ \ \forall q \in Q.$$

Choosing an allowable initial state $x_0 = [7.0 \quad 0.5]^T$, a controller was synthesized to solve the optimal control problem for the system. The system's state space was partitioned into a 256×128 grid.Similar to the pendulum example, this control system is a 2-dimensional one. This means that the cost-to-go function and the policy may be visualized with a 2D plot, as shown in Figure 7.25. This Figure shows the final values of V and π when the algorithm was run until the convergence of V.



(a) Cost-to-go function (b) Policy

Figure 7.25: Visualization of the cost-to-go function and the policy for the tank control system

These figures demonstrate two important features of the synthesized controller. First, it clearly shows the effect of the chosen cost function on the cost-to-go function. Due to the chosen desired height, the lowest costs are associated with the cells around that target. This in turn affects the policy and the resultant controller. Second, it shows the controller's predictive nature. Rather than constantly changing the amount of inflow into the tank in the direction of the target height, the controller finds the path to a state where the inflow matches the outflow at the target height.

This is also made clear by the simulation of the system with the synthesized controller, shown in Figure 7.26. This plot exhibits how the system behaves from the initial conditions previously mentioned. The control input is normalized to make the plot more readable.

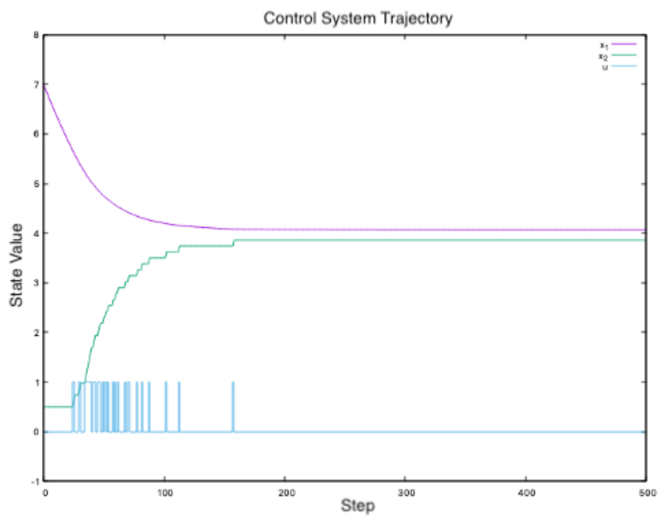


Figure 7.26: Trajectory of the tank control system with the synthesized controller

7.3 Holonomic Robot

This example demonstrates a more complex application of the proposed algorithm. The system considered is a constant-velocity holonomic mobile robot with non-linear dynamics. Put simply, the term holonomic means that the number of controllable degrees of freedom in the system is equal to the total degrees of freedom. For example, this means that the robot can rotate without requiring translational movement. This robot is defined by the following state variables:

- p_x: the x coordinate of the robot
- p_y: the y coordinate of the robot
- θ: the bearing of the robot

These state variables may be relative to the robot, the *robot frame*, or they may be considered relative to the environment in which the robot operates, the *world frame*. The discretized state dynamics for this system are considered in the context of the world frame, allowing the state updates to be directly mapped to the discrete cells that partition the robot's state space. These dynamics are defined by the equations below:

$$f_q(p_x, p_y, n) = \begin{bmatrix} p_x(n+1) \\ p_y(n+1) \\ \theta(n+1) \end{bmatrix} = \begin{bmatrix} p_x(n) + hv\cos(\theta(n) + hu_q(n)) \\ p_y(n) + hv\sin(\theta(n) + hu_q(n)) \\ \theta(n) + hu_q(n) \end{bmatrix}$$

In the model above, $h = 0.01$ and $v = 0.1$. This is a slightly simplified model in that the robot can move forward at a constant speed v but cannot stop or reverse. The only input in the system, u, controls the turning angle of the robot at every step. Once again, the TDSS model for this system, H_r is defined by defining the sets Q, F, and $\mathcal{X}$. The set Q, as well as the associated set of control inputs, is as follows:

$$Q = \{1, 2, 3\} \qquad \mathcal{U} = \{0, \frac{\pi}{0.04}, \frac{\pi}{0.04}\}$$

The set F can then be defined as

$$F = \{f_1(p_x, p_y, \theta, n), f_2(p_x, p_y, \theta, n), f_3(p_x, p_y, \theta, n)\}$$

where

$$f_1(p_x, p_y, \theta, n) = \begin{bmatrix} p_x(n) + 0.001\cos(\theta(n)) \\ p_y(n) + 0.001\sin(\theta(n)) \\ \theta(n) \end{bmatrix}$$

$$f_2(p_x, p_y, \theta, n) = \begin{bmatrix} p_x(n) + 0.001\cos(\theta(n) + \pi/4) \\ p_y(n) + 0.001\sin(\theta(n) + \pi/4) \\ \theta(n) + \pi/4 \end{bmatrix}$$

$$f_3(p_x, p_y, \theta, n) = \begin{bmatrix} p_x(n) + 0.001\cos(\theta(n) - \pi/4) \\ p_y(n) + 0.001\sin(\theta(n) - \pi/4) \\ \theta(n) - \pi/4 \end{bmatrix}$$

The robot's state space is the environment in which it operates. Physically, it can be thought of as an environment where the robot can move freely, with obstacles being state constraints on the system. Figure 7.27 shows the environment considered in this example. The lines represent the walls of the room that the robot must avoid, and the markers represent the start position and goal position. The state space for the state variable θ is more difficult to visually represent but is simply the direction in which the robot is allowed to point. Thus, the allowable state space $\mathcal{X}$ is the set

$$\mathcal{X} = \{x \in \mathbb{R}^3 \,|1 < p_x < 9,\ 1 < p_y < 9,\ (3 \le p_x \le 7) \to 3 < p_y < 9,\ -\pi \le \theta < \pi\}.$$

With H_r clearly defined, the final pieces of information necessary to for the optimal control problem are the control objectibe described by J_N and the initial state x_0. In this case, as the objective is to find the shortest allowable path from an initial state, chosen as $x_0 = [2.0 \quad 2.0 \quad \pi/2]^T$, and a desired final position, chosen as $r_x = 8.0$ and

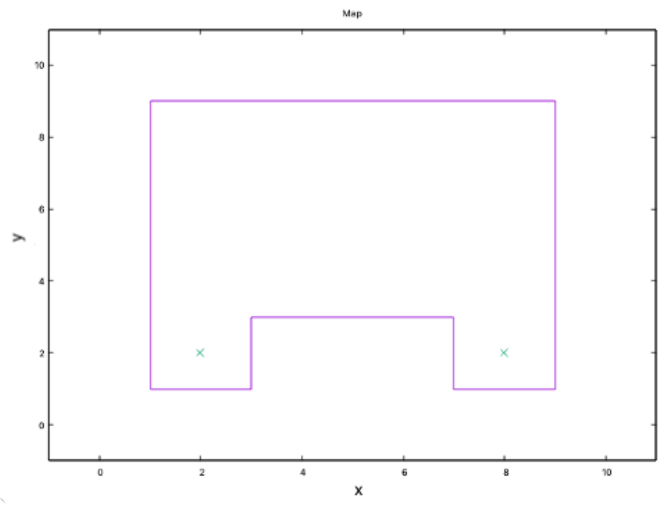


Figure 7.27: Robot environment

$r_y = 2.0$. J_N is then defined as

$$J_N = \sum_{n=0}^{N-1} (p_x(n) - r_x)^2 + (p_y(n) - r_y)^2 \;\; \forall q \in Q.$$

To discretize the state space to synthesize a controller, the p_x and p_y spaces were partitioned into 256 cells each. The θ state space could not be split into an equally fine grid to keep the algorithm computationally tractable. However, when considering the robot's bearing, one can see that the cells do not need to be as fine. It is enough to know the general direction that the robot points. Thus, the θ state space is split into 8 distinct cells, each representing a $\pi/4$ portion of the space. This also improves the computational resources required to synthesize the controller compared to having a uniformly-sized grid for all state variables. Therefore, the map is partitioned into $256 \times 256 \times 8$ cells.

The controller was synthesized using this control objective, and the system trajectory was simulated. The results of the simulation are included in Figure 7.28 As

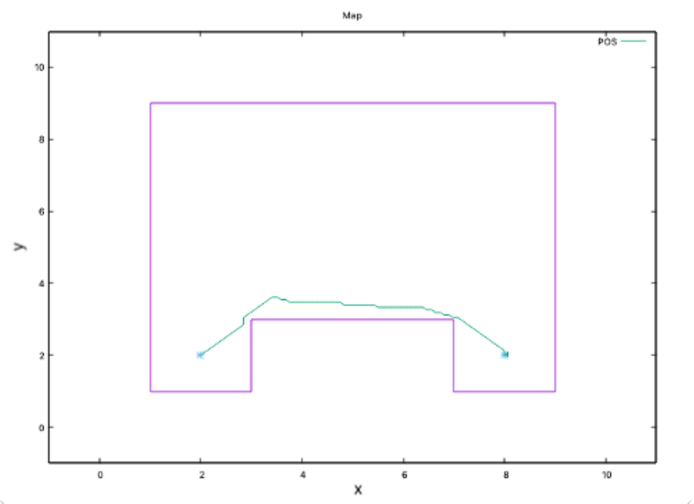


Figure 7.28: Trajectory of the Robot Control System with the Synthesized Controller

we can see, the controller can find a path to the desired final target while avoiding obstacles. It is important to note that this simulation considers the robot as a point mass, without additional constraints for its physical body.

Chapter 8

Conclusion

8.1 Concluding Remarks

This portion of the thesis examined the use of Dynamic Programming as a computational tool to approximate optimal control policies for switched systems. It was shown that due to the reliance of Dynamic Programming on discrete transitions, the continuous state space of switched systems of interest must be discretized to use a Dynamic Programming approach. Discretization of the state space means that the synthesized control policies are only approximations of the optimal control policies, but it was also discussed that the state space for different state variables may be discretized into a unique number of segments. This provides opportunities for improvement of the discretization for various systems, thus improving the performance of synthesized controllers. A high-level overview of the controller synthesis implementation was also discussed, which provided a framework for synthesizing the approximate controllers computationally. Using this implementation, three classic examples were studied to evaluate the effectiveness of the proposed algorithm. First, the pendulum example demonstrated the effectiveness of the approximated controller for a stable system, allowing for a thorough evaluation of the effect of the control objectives chosen on the resultant controller. The single-valve tank example was used to study the synthesis process for a more complex system with hard physical constraints. It was shown that the algorithm produced a predictive controller that also satisfied the control objectives. Finally, a holonomic mobile robot control system showed off more subtle, but equally attractive, features of the algorithm. For one, it showed the synthesis process' compatibility with non-linear dynamic systems. This example also demonstrated how non-uniform discretization of the state variables' state space would be beneficial for some systems.

8.2 Future Work

There are various parts of this work that call for further investigation. Proposed areas for future work are as follows:

- The use of continuous controllers to correspond to the modes of the switched system, rather than constant control inputs,
- Improvements to the implementation of the controller synthesis algorithm to improve numerical performance and computational effort,
- Studies into the stability of the resultant controllers,
- Studies into the robustness of the resultant controllers.

References

[1] R. Alur, T.A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22(3):181–201, 1996.

[2] Rajeev Alur, Costas Courcoubetis, Thomas A Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid systems*, pages 209–229. Springer, 1992.

[3] Panos J Antsaklis, Xenofon D Koutsoukos, and N Dame. Hybrid systems control. *Encyclopedia of Physical Science and Technology*, 7:445–458, 2002.

[4] Kendall E Atkinson. An introduction to numerical analysis. John Wiley and sons, 1989.

[5] Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716, 1952.

[6] Dimitri P Bertsekas. Value and policy iterations in optimal control and adaptive dynamic programming. *IEEE transactions on neural networks and learning systems*, 28(3):500–509, 2015.

[7] Michael S Branicky. *Studies in hybrid systems: Modeling, analysis, and control.* PhD thesis, Massachusetts Inst. of Tech. Cambridge Lab for Information and Decision Systems, 1995.

[8] Roger W Brockett. Smooth dynamical systems which realize arithmetical and logical operations. In *Three decades of mathematical system theory*, pages 19–30. Springer, 1989.

[9] Roger W Brockett. Hybrid models for motion control systems. In *Essays on Control*, pages 29–53. Springer, 1993.

[10] Ken Butts, Ilya Kolmanovsky, N Sivashankar, and Jing Sun. Hybrid systems in automotive control applications. In *Control using logic-based switching*, pages 173–189. Springer, 1997.

[11] Elena De Santis, Maria Domenica Di Benedetto, et al. Observability of hybrid dynamical systems. *Foundations and Trends in Systems and Control*, 3(4):363–540, 2016.

[12] Carlos E Garcia, David M Prett, and Manfred Morari. Model predictive control: Theory and practice - a survey. *Automatica*, 25(3):335–348, 1989.

[13] Aleks Gollu and Pravin Varaiya. Hybrid dynamical systems. In *Proceedings of the 28th IEEE Conference on Decision and Control*, pages 2708–2712. IEEE, 1989.

[14] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[15] WPMH Heemels, B De Schutter, J Lunze, and M Lazar. Stability analysis and controller synthesis for hybrid dynamical systems. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1930):4937–4960, 2010.

[16] Daniel Liberzon. *Switching in systems and control*, volume 190. Springer, 2003.

[17] Anil Nerode and Wolf Kohn. Models for hybrid systems: Automata, topologies, controllability, observability. In *Hybrid systems*, pages 317–356. Springer, 1992.

[18] Michael Neunert, Abbas Abdolmaleki, Markus Wulfmeier, Thomas Lampe, Tobias Springenberg, Roland Hafner, Francesco Romano, Jonas Buchli, Nicolas Heess, and Martin Riedmiller. Continuous-discrete reinforcement learning for hybrid control in robotics. In *Conference on Robot Learning*, pages 735–751. PMLR, 2020.

[19] Singiresu S Rao. *Engineering optimization: theory and practice*. John Wiley & Sons, 2019.

[20] Shankar Sastry, George Meyer, Claire Tomlin, John Lygeros, Datta Godbole, and George Pappas. Hybrid control in air traffic management systems. In *Proceedings of 1995 34th IEEE Conference on Decision and Control*, volume 2, pages 1478–1483. IEEE, 1995.

[21] Angela Schöllig. *Optimal control of hybrid systems with regional dynamics*. PhD thesis, Georgia Institute of Technology, 2007.

[22] M Shahid Shaikh and Peter E Caines. On trajectory optimization for hybrid systems: Theory and algorithms for fixed schedules. In *Proceedings of the 41st*

IEEE Conference on Decision and Control, 2002., volume 2, pages 1997–1998. IEEE, 2002.

[23] M Shahid Shaikh and Peter E Caines. On the optimal control of hybrid systems: Optimization of switching times and combinatoric location schedules. In *Proceedings of the 2003 American Control Conference, 2003.*, volume 4, pages 2773–2778. IEEE, 2003.

[24] M Shahid Shaikh and Peter E Caines. Optimality zone algorithms for hybrid systems computation and control: From exponential to linear complexity. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 1403–1408. IEEE, 2005.

[25] Olle Sundström, Daniel Ambühl, and Lino Guzzella. On implementation of dynamic programming for optimal control problems with final state constraints. *Oil & Gas Science and Technology–Revue de l'Institut Français du Pétrole*, 65(1):91–102, 2010.

[26] Héctor J Sussmann. Set-valued differentials and the hybrid maximum principle. In *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, volume 1, pages 558–563. IEEE, 2000.

[27] Arjan J Van Der Schaft and Johannes Maria Schumacher. *An introduction to hybrid dynamical systems*, volume 251. Springer, 2000.

[28] Martin von Mohrenschildt. A hybrid controller for a nonholonomic car-like robot. In *ICINCO*, pages 282–288, 2005.

[29] Hans Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Transactions on Automatic Control*, 11(2):161–167, 1966.

[30] Wei Zhang, Jianghai Hu, and Alessandro Abate. A study of the discrete-time switched LQR problem. 2009.