Variable Transition Time Predictive Control

VARIABLE TRANSITION TIME PREDICTIVE CONTROL

BY

KASKA KOWALSKA, M.A.Sc., (Software Engineering)

McMaster University, Hamilton, Ontario, Canada

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTING & SOFTWARE

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

Doctor of Philosophy (2011) McMaster University

(Computing & Software) Hamilton, Ontario, Canada

TITLE:	Variable Transition Time Predictive Control
AUTHOR:	Kaska Kowalska B.Eng., (Computer Engineering) McMaster University, Hamilton, Ontario, Canada M.A.Sc., (Software Engineering) McMaster University, Hamilton, Ontario, Canada
SUPERVISOR:	Dr. Martin von Mohrenschildt
NUMBER OF PAGES:	ix, 100

Abstract

This thesis presents a method for the design of a predictive controller with variable step sizes. Predictive methods such as receding horizon control (or model predictive control) use a a fixed sampling frequency when updating the inputs. In the proposed method, the switching times are incorporated into an optimization problem, thus resulting in an adaptive step-size control process. The controller with variable time steps is shown to require less tuning and to reduce the number of expensive model evaluations. An alternate solution approach had to be developed to accommodate the new problem formulation. The controller's stability is proven in a context that does not require terminal cost or constraints. The thesis presents examples that compare the performance of the variable switching time controller with the receding horizon method with a fixed step size. This research opens many roads for future extension of the theoretical work and practical applications of the controller.

Acknowledgements

I offer my sincerest gratitude to my supervisor, Dr. Martin von Mohrenschildt, who has supported me throughout my Ph.D. studies with his patience, knowledge, and enthusiasm. I appreciate all his contributions of time, funding, and advice that helped me get over the 'bumps'. Without his encouragement and guidance this thesis would not have been completed.

I would also like to express my appreciation to the remaining members of my committee, Dr. Doug Down, Dr. Antoine Deza, and Dr. Ruth Milman, for their valuable comments on this dissertation.

This thesis would not have been possible without the support of my family and friends who encouraged me to complete it by relentlessly asking me "Are you done yet?". Finally, I am especially grateful to Keith for his encouragement and confidence in my work. Thank you!

Contents

List of Figures

Chapter 1

Introduction and Problem Statement

The traditional and popular feedback controllers such as the Proportional, Integral, Derivative (PID) controller use information about the past behaviour of a plant in order to compute a control input for the current state. However, advances in computational hardware and software have made it possible to develop more complex control algorithms that predict the plant's behaviour and pro-actively steer its state. The prediction is based on a mathematical model of the plant. Receding Horizon Control (RHC) (also known as Model Predictive Control (MPC)) is one of the more successful forms of model based control. It uses the plant's model in order to predict its behaviour over a set number of steps (horizon length) and computes an optimal control input for each of those steps. Thus, while PID control can be compared to driving a car forward while looking in the rear-view mirror, RHC allows us to look forward through the windshield of the car to steer it. The RHC algorithm provides us with a *tailor-made* plant control strategy since the plant's model is used explicitly in the formulation of the control problem.

RHC solves an optimal control problem that computes a sequence of control inputs that are meant to optimize the future behaviour of the plant over an interval divided into a finite number of steps. The optimization is defined by a cost function, a function that optimizes the performance of the controller and the response of the plant. Only the first control input in the sequence is applied to the plant, the remaining inputs are discarded and the optimization problem is recomputed with an updated current state as the initial state and a control interval moved forward by one step. The parameters of the control problem, such as the number of steps, interval length, and cost function parameters, can all be used to tune the RHC algorithm. However, tuning is not a trivial task and mostly involves trial-and-error methods.

The thesis proposes a controller that creates a more general control algorithm and eliminates some of the tuning requirements by incorporating the length of the steps within a control interval into the optimization problem. This approach creates step sizes that change depending on the optimization of the behaviour of the plant. Since the transitions from one control input to the next vary in length, we call our proposed approach Variable Transition Control (VTC).

The thesis focuses on developing a sound foundation for the theory of VTC. It develops a stability proof and proposes bounds for the performance of the controller. It also compares the cost of VTC to the cost of the fixed step RHC algorithm. A new approach for solving the control problem was developed as part of the practical implementation of the control algorithm. This was necessitated by the time steps being part of the optimization problem. Examples are included to underline the features of the control algorithm and provide a comparison with RHC. The thesis suggests more advanced features, such as constraints and robustness, as future extensions to the topic.

1.1 Predictive control

Advances in computational hardware and software algorithms have made it possible to develop sophisticated control methods for systems where the desired response is achieved through optimization of a performance function. Optimal control algorithms, for example [1] and [2], use a plant model in order to generate a custom closed-loop controller that minimizes an objective function. The objective function is also referred to as a performance index or a cost function and, in the general case, can be defined to optimize any aspect of the plant and controller behaviour.

A very popular extension of optimal control is receding horizon control that has been developed for both linear and nonlinear systems (see [3, 4, 5, 6, 7, 8] for example). RHC incorporates a moving time interval known as a (receding) prediction horizon into an on-line computation of an open-loop control law using the current state of the system as an initial state for the next step of the computation. The control input computed for current time is applied until a new state measurement is obtained at the next fixed sampling instance. The new control law is computed with the prediction horizon shifted forward by a single step thus creating the receding horizon. The result is a sequence of control inputs for each sampling point that is valid over the horizon length. The horizon length defines the number of decision variables for the optimization problem. Shorter horizon may be desired from a computation effort perspective, but one must be careful not to jeopardize the stability performance of the controller, which can be compromised by a horizon length that is too small.

The thesis presents a control algorithm the solves the optimal control problem on-line over a finite horizon length but instead of computing a control sequence at a fixed sampling instant, it optimizes the time step used to move the control problem forward by including a sequence of switching points in the optimization problem formulation. The result of solving the optimization problem is a sequence of control inputs and a sequence of corresponding switching points. As in the RHC approach, only the first control input is applied to the system at the initial time until the next switching time. The rest of the sequence is discarded and the problem is recomputed after the horizon has been moved forward to that switching point. We call our approach Variable Transition Control (VTC) since the time steps are not equidistant as in RHC but change in size depending on the prediction of the future behaviour of the plant.

Our motivation is to develop an adaptive step-size predictive control method that decreases the dependence on tuning by optimizing the time points at which a new control sequence is computed. The adaptive step idea borrows from methods developed for numerical integration that use large steps in regions where the function behaviour is smooth thus improving efficiency, and smaller steps in regions where it fluctuates in order to improve accuracy. Similarly, VTC apples a control input until a time point at which it becomes optimal, based on a performance measure, to switch to a new control input. As a side effect, our control law becomes more general than RHC as both the switching time and the control input are optimized.

The thesis shows that this approach results in a controller that has a lower or at least equal cost to the RHC controller. The performance proofs are also developed in order to present a complete theoretical framework for the proposed control method.

1.2 Contributions of the thesis

The major contributions of the thesis are in the following domains:

- **Predictive control theory**

  We generalize the existing receding horizon control theory by replacing the fixed sampling time with variable time steps that are incorporated directly into the optimization problem. This results in a predictive controller with an *adaptive* step size that requires less tuning and possibly fewer model evaluations. The proposed controller computes the optimal step size based on the controller's cost function. This is in contrast to numerical integration methods that compute the adaptive step size based on an error tolerance metric. The cost of this approach is an increase in the number of decision variables in the optimization problem. However, as an example in the thesis shows, when the system dynamics are sufficiently smooth, the controller can take larger steps and thus reduce the number of model evaluations.

- **Stability and performance of variable transition control**

  First, we show that the cost of the variable transition controller is in the worst case equal to the cost of RHC and in general $\mathcal{J}_{\text{VTC}} \leq \mathcal{J}_{\text{RHC}}$.

  We prove stability and give a performance bound for our approach by generalizing the existing stability and performance proofs for the fixed step receding horizon controller to a controller with a variable step-size.

  We use Lyapunov's second theorem and formulate the cost function (Lyapunov function candidate) over a number of switching points where the distance between each consecutive pair of points is allowed to vary. We then show that the cost function over the horizon length H is decreasing. The cost function over a single step is not a good candidate in general since that would require it to be monotonically decreasing in each step. This would severely restrict the class of system for which the proof and the controller is applicable. Our approach does not require restrictive terminal cost or state constraints, these are theoretical constraints that cannot be applied in practice.

Similarly, the performance bound is computed by evaluating the variable step size cost function over infinite horizon.

- **Implementation and solution of the optimization problem**

  Addition of a variable step size into the optimization problem of the controller makes it impossible to apply current RHC solution methods. The RHC formulation relies on a solution to the Algebraic Riccati Equation to solve its control problem. Since the control input and the step size variables occurring in the VTC problem result in a nonlinear set of equations, a new solution approach is developed. The method uses Euler's first order approximation to solve the system under the resulting piecewise constant control law for state variables that are then used within the cost function in the optimization problem.

  The resulting framework is a hybrid of symbolic and numeric computing where the control law and approximation function are formulated symbolically and the optimization problem is solved numerically.

  Two optimization approaches are used within the thesis to compute the minimal solution of the cost function based on the above approximation. The first implementation approach uses a symbolic computation engine, Maple, to construct, solve, and implement the entire control problem. The second approach uses Maple to construct the control problem, which is then code generated and solved using an external optimization solver IPOPT.

Using both methods was useful for fully exploring the implementation issues associated with solving receding horizon problems. This analysis is something that is often missing or is only briefly mentioned in the predictive control literature.

1.3 Outline of the thesis

The following chapters will present the background for the theory of predictive control and the VTC problem formulation, explain the computational issues and the approximation approach that was used to overcome those, present arguments for stability and performance properties and show examples of the application of this control method.

The next chapter will provide a concise overview of various control methods that have led or contributed to the development of predictive control techniques. It will focus on developing a solid understanding of the practical and theoretical aspects of RHC since that is crucial to the understanding of the proposed generalization of that theory.

The third chapter outlines how the idea for VTC originated. It provides more background for the research by briefly describing how the limitations of a mode switching control method defined in [9] led to the questions, ideas, and ultimately the solutions presented in this thesis.

The fourth chapter will introduce VTC by presenting the problem statement and then discussing the approach taken to solve that control problem. The theoretical foundation of VTC is built by presenting stability and performance arguments for the proposed controller. The chapter also argues that the cost function for VTC is at least as good as the cost function for RHC, meaning that in the worse case the proposed controller is as good as RHC. This is not surprising since VTC can be converted to the RHC algorithm by simply limiting the switching points to a fixed size.

The fifth chapter discusses implementation of VTC and discusses the methods used to formulate the control problem. It describes the symbolic framework used to construct the solution procedure and then outlines the numerical optimization methods used to solve the control problem. Two approaches were used for this purpose: one involving optimal computation within the symbolic framework and the other involving code generation so that the control problem could be compiled and solved with an external solver. Finally, the chapter will show the examples used to evaluate the effectiveness of VTC and includes comparisons to the results of the RHC algorithm.

The thesis concludes with a discussion of the possible extensions to the theory and application of VTC and summarizes the impact of the results on the field of model based predictive control.

1.4 Basic control terminology

A simple control system consists of a controller that outputs a control signal u to the plant. The plant can then output its updated state x. Open loop control ignores the output of the plant and continues to feed a control input regardless of its effect. Closed loop controllers will instead monitor the output of the plant y, and possibly compare it with a reference signal r. The error, which is the difference between the desired set point and the output of the plant $e = r - y$, is then used to update the control input sent to the plant. This feedback loop is shown in Figure 1.1.

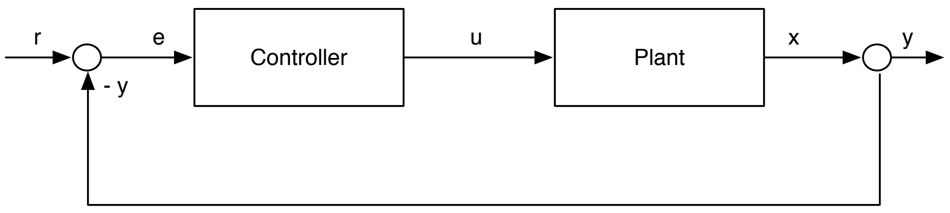

Figure 1.1: Simple feedback control system

Perhaps the best known example of a feedback controller is the Proportional Integral Derivative (PID) controller. The PID control input at any time is a sum of three terms:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t). \tag{1.1}$$

The *proportional* action is based purely on a scaled current error term, the *integral* action represents a response to the sum of the errors, and the *derivative* action is based on the rate of change of the error. The scaling factors K_p, K_i and K_d represent the proportional, integral, and derivative gains respectively. More details on how to compute these gains and PID design in general can be found in [10].

The limiting aspect of PID control is that it provides us with a single control input that is computed for a single measured output variable. Such systems are called Single Input Single Output (SISO) systems. However, many processes require multiple control inputs for the many measured variables, which result in Multiple Input Multiple Output (MIMO) systems. This is where model based control methods excel. They can naturally be extended to handle these multivariate systems. Multivariate systems can be represented by a set of (differential) equations, and if all of the equations are linear (or are linearized) then state-space notation is used.

The state-space representation is the favored notation in multivariate model based control. A linear, multivariate system

$$\begin{aligned} x_1' &= x_1 + 0.5x_2 + u_1 \\ x_2' &= 2x_1 + 0.1u_2 \\ x_3' &= 0.8x_2 + 0.1x_3 \end{aligned} \tag{1.2}$$

can be represented in state space form as

$$x' = Ax + Bu \tag{1.3}$$

where A and B are defined as

$$A = \begin{bmatrix} 1 & 0.5 & 0 \\ 2 & 0 & 0 \\ 0 & 0.8 & 0.1 \end{bmatrix} B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{1.4}$$

and where $x \in \mathbf{R}^{\mathrm{n}}$ is the state of the system and $u \in \mathbf{R}^{\mathrm{m}}$ represents the control input. The output of the above system could also be modeled with $y = Cx + Du$, where $y \in \mathbf{R}^{\mathrm{k}}$ represents the output. The RHC literature often uses a discretized form of the plant thus the state-space model is presented in discrete-time as in (1.5)

$$x_{k+1} = Ax_k + Bu_k,\ k = 0, 1, \ldots \tag{1.5}$$

The state space notation can be used when the model of a system is known. If the physical system model is difficult to develop then simple input/output models such as transfer function models or time series models can be used [11]. However, since model based control methods rely on a mathematical model of the plant, the thesis will use either the set of differential equations or the state space notation.

1.5 Summary of mathematical notation

H	prediction horizon
$\mathcal{J}$	cost function
N	number of switching points
$\mathbf{R}$	real numbers
τ	switching time point
u	control input
x	state
y	output
$\mathbf{U}$	control input sequence
$\mathbf{T}$	switching point sequence
$\mathcal{S}$	a list of pairs of optimal control inputs and their switching times

Chapter 2

Background and Literature Review

This chapter provides background information that will help the reader understand the variable time control approach. The background information also includes a literature review of the major contributions to the theory and applications of model based predictive control. The theoretical work review is especially important to understanding the stability arguments of our control approach.

We will first give a brief overview of Internal Model Control method since this is the approach that introduced the use of a model of the plant in the controller structure. Then, we discuss Optimal Control in more detail since this method introduced the use of a cost function in order to compute an optimal control input for a plant over an infinite horizon. Finally we discuss receding horizon control, which was motivated in part by practical limitations of optimal control. Unlike optimal control, RHC uses a finite horizon to compute a sequence of optimal control inputs over a fixed number of sampling times within each horizon length.

As shown in Figure 2.1, our research is positioned between the Optimal Control and the receding horizon control methods.

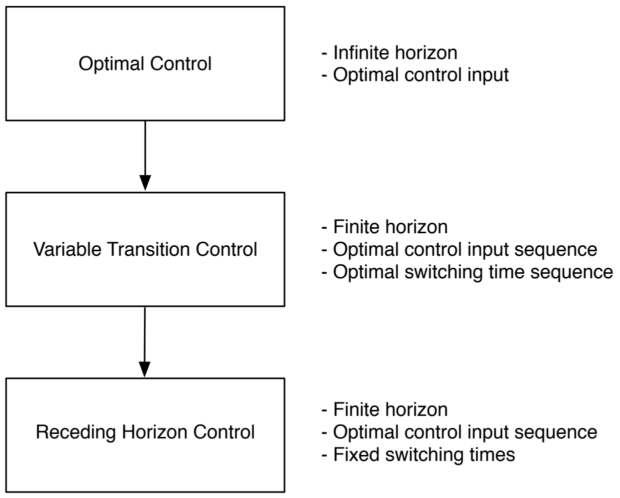

Figure 2.1: Positioning of research area

The variable switching time approach uses a finite horizon length just like RHC, but it does not use a fixed sampling time. Instead, it optimizes the times at which the control inputs are switched, thus reducing the tuning effort.

The characteristics of the above approaches are introduced in more detail in the following sections.

2.1 Internal Model Control

There are many control applications for which a simple form of feedback control is sufficient for achieving the desired plant response. The PID controller in (1.1) provides satisfactory performance for many control problems encountered in industry. However, other applications require high performance controllers to satisfy the economic or safety requirements of plant operations. The knowledge of process dynamics is necessary in order to be able to determine the appropriate control action. For this reason, model based control methods utilize a mathematical model of the plant in order to derive a controller that produces the desired process response.

Internal Model Control (IMC) is an established form of model-based control that links the process model directly with the controller structure. While the roots of model-based control can be traced back to the 1950s, it was the work of [12] that provided major contributions to the advancement of IMC by presenting a sound theoretical framework for the structural representation.

A simple way to define IMC mathematically is to replace the plant in Figure 1.1 with a function G (transfer function model of the plant) and add a collective disturbance measure d so that the process dynamics are represented by

$$y = Gu + d. \tag{2.1}$$

Figure 2.2 shows a diagram of this simple feedback control system. If the objective

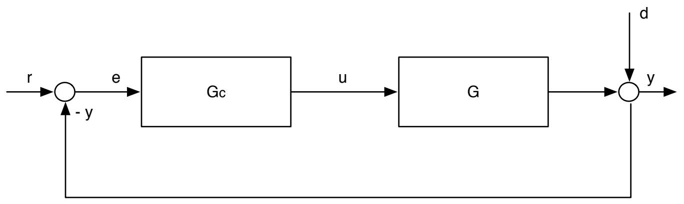

Figure 2.2: Control system with feedback

is to have "perfect" control where the plant follows the desired set point r then substituting that signal for the output in (2.1) results in

$$r = Gu + d \tag{2.2}$$

which can be used to solve for the control input

$$u = \frac{1}{G}(r - d). \tag{2.3}$$

Thus, the control input is dependent on the inverse of the plant model.

Note that in practice it is difficult to match the model to the physical plant or to measure the disturbances, thus approximations must be used instead. Also, making the control input an inverse of the model might not be possible if the model is not invertible. Finally, the pre-specified trajectory r may be unattainable or require excessive control action. It is more practical to follow the "best" possible trajectory instead. This is accomplished by an optimization approach.

The optimization approach differs in that instead of defining a reference signal, it requires a definition of an *objective function* of the state and control input. If the system dynamics are defined by the differential equation:

$$\frac{dx}{dt} = f(x, u) \tag{2.4}$$

the general form of an objective function can be defined as

$$\mathcal{J} = \int_0^{t_f} l(x, u) \tag{2.5}$$

where the objective is to compute a control input that will minimize (or maximize) the cost subject to additional constraints on the control input or state. The advantage of the optimization approach is that it automatically considers operating constraints and, unlike IMC, can easily be applied to nonlinear multivariate systems of arbitrary order.

The optimal control approach is at the core of the model predictive control methods. Thus, the following section will discuss optimal control in more detail as it is essential to understanding how RHC works.

Note that the methods used to obtain a model of the plant are outside the scope of the thesis. Computing the models through the process of *system identification* is a very large research area especially within the field of control theory, where high fidelity models are of foremost importance to the performance of a model-based controller. A comprehensive overview of this topic can be found in [13].

2.2 Optimal Control

The mathematical foundations of RHC are closely based on those found in optimal control theory [1, 2]. More specifically, the receding horizon problem formulation is similar to the Linear Quadratic Regulator (LQR) problem. The LQR problem also uses a model of a system in order to determine a control input that minimizes an associated quadratic performance index. The solution to this problem requires a formulation of a Hamiltonian [2], which is then solved by taking into account any boundary conditions provided for the system. The following discussion will aim to explain the general problem setting for LQR and it will also relate this to the RHC scheme.

The most crucial part of the LQR and the RHC problem is the knowledge of the model of the system since the accuracy of the model will have a direct effect on the precision of the control moves. The general state-space formulation of a linear system model is given in (2.6)

$$x' = Ax + Bu \tag{2.6}$$

where $x \in \mathbf{R}^{\mathrm{n}}$ is the state of the system, and $u \in \mathbf{R}^{\mathrm{m}}$ represents the control input. The system definition could also include a reference trajectory value r. Another important requirement for the RHC problem formulation is the definition of the *cost function*, or what is called the *performance index* in LQR. While there are many variants of the cost function that appear in the literature, the basic ingredients must always be present. This function must calculate the cost of future controls and a low value should imply good control performance. Thus, this function is considered as the performance index and can be expressed in quadratic form over a finite horizon such as

$$J(t_0) = \frac{1}{2}x^{\mathrm{T}}(T)S(T)x(T) + \frac{1}{2}\int_{t_0}^{T}(x^{\mathrm{T}}Qx + u^{\mathrm{T}}Ru)\,dt \tag{2.7}$$

where S, Q are symmetric and positive semi-definite weighting matrices and R is a symmetric, positive definite weighting matrix. These can be tuned to achieve the desired performance.

The LQR problem proceeds by solving the optimal control problem. In the general formulation of the optimization problem, the system is defined by $x'(t) = f(x(t), u(t))$ and the performance index by $J(t_0) = \phi(x(T), T) + \int_{t_0}^{T} L(x(t), u(t))\,dt$, where $\phi(x(T), T)$ is the final weighting function and $[t_0, T]$ is the interval of interest. The optimal control input is computed by first constructing a Hamiltonian function of the system

$$H(x, u, t) = L(x, u, t) + \lambda^{\mathrm{T}} f(x, u, t) \tag{2.8}$$

(where λ is a function analogous to Lagrange multiplier [2]) followed by finding its corresponding state, costate, and stationary condition equation given by equations (2.9), (2.10), and (2.11) respectively

$$x' = \frac{\partial H}{\partial \lambda} = f \tag{2.9}$$

$$-\lambda' = \frac{\partial H}{\partial x} = \frac{\partial L}{\partial x} + \frac{\partial f^{\mathrm{T}}}{\partial x}\lambda \tag{2.10}$$

$$0 = \frac{\partial H}{\partial u} = \frac{\partial L}{\partial u} + \frac{\partial f^{\mathrm{T}}}{\partial u}\lambda\ . \tag{2.11}$$

Thus, for the system defined by equation (2.6) and a cost function given by equation (2.7), the Hamiltonian is defined as:

$$H = \frac{1}{2}(x^{\mathrm{T}}Qx + u^{\mathrm{T}}Ru) + \lambda^{\mathrm{T}}(Ax + Bu) \tag{2.12}$$

and the state and costate equations become:

$$\begin{aligned} x' &= Ax - BR^{-1}B^{\mathrm{T}}\lambda && (2.13)\\ -\lambda' &= Qx + A^{\mathrm{T}}\lambda. && (2.14) \end{aligned}$$

These equations were obtained by substituting the stationary condition $u = -R^{-1}B^{\mathrm{T}}\lambda$ into (2.9). To compute feedback control, given the initial state $x(t_0)$, a two-point boundary-value problem is solved under the assumption that x and λ satisfy a linear relation

$$\lambda(t) = S(t)x(t) \tag{2.15}$$

for the yet unknown matrix function $S(t)$. This approach, described in more detail in [2] is called free-final-state since the final state is not fixed, rather it is only required to minimize the performance index. To find this function, the costate equation is differentiated, such that:

$$\lambda' = S'x + Sx' = S'x + S(Ax - BR^{-1}B^{T}Sx). \tag{2.16}$$

Finally, using (2.14), a matrix Riccati equation [2, 11], is obtained:

$$S' = SBR^{-1}B^{T}S - A^{T}S - SA - Q. \tag{2.17}$$

The optimal control input is defined, using the Riccati equation solution and (2.15), as

$$\begin{aligned} u^* &= -R^{-1}B^{T}Sx(t) \qquad (2.18)\\ &= -K(t)x(t) \end{aligned}$$

where $K(t)$ represents Kalman gain given by

$$K(t) = R^{-1}B^{T}S(t). \tag{2.19}$$

Should the performance be undesirable, the weighting matrices can be tuned to find a more appropriate control input for the plant.

Computing a solution for the feedback control problem that provides optimal control for all states is a very hard, if not an impossible problem. This is avoided by the receding horizon approach, which solves an open loop control problem on-line.

It computes an open loop solution over a number of steps, each time updating the initial state of the system with the current state. This approach results in an optimal feedback control (when no uncertainties are present) which is far less computationally expensive [14]. This is why RHC is sometimes called open loop optimal control.

The decreased computational effort and the ease with which RHC can be implemented and applied to multivariate, nonlinear, and constrained systems, have made it a popular control method for plants requiring advanced controllers.

2.3 Receding Horizon Control

The RHC strategy has been presented in detail in [3, 15, 16], just to name a few. The advantage of RHC over other popular forms of control such as PID is that it uses a model of the plant in order to predict its future behaviour. This control strategy is similar to the one employed by chess players. They do not decide their next move based solely on their opponent's previous move. Rather, they try to anticipate their opponent's future moves and then base their decisions on that prediction.

Similar strategy is employed in RHC. The next control move is computed by the model-based prediction of the future behaviour of the system. The advantage of this method is easy to conceptualize. Imagine driving a car down a road. If we used a PID form of control it would be like trying to control the car by looking only in the rear-view mirror. However, with the RHC scheme we can look through the car's front windshield to drive.

2.3.1 Overview of RHC

The receding horizon control is very similar to the optimal control problem in that we still require a model of a system and a cost function (objective function) in order to compute a sequence of minimizing control inputs. However, there are some additional aspects of RHC that make it a desirable form of control for industrial applications. For example, is the only known technique that incorporates constraints naturally and systematically [11].

RHC also evaluates the effect of the control inputs on the system over some predefined prediction horizon, H_p. Figure (2.3), which looks at RHC from a discrete time perspective, will be used in the explanation of the RHC algorithm.

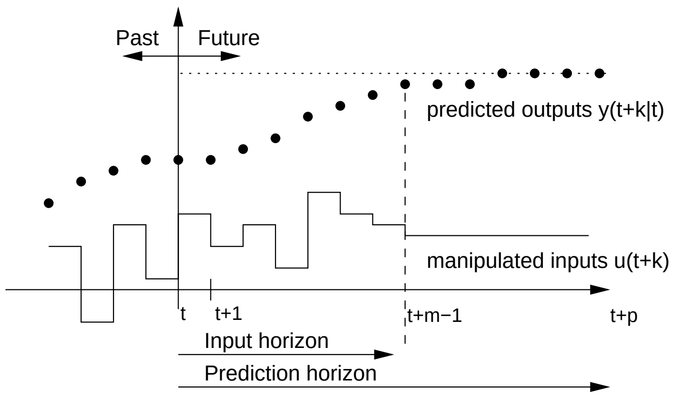

Figure 2.3: Model predictive control scheme

RHC often uses a discrete-time notation such that the model given in (2.6) can be transformed into a difference equation of the form $x_{k+1} = Ax_k + Bu_k$. In Figure (2.3), H_p is the prediction horizon and H_u is the control input horizon and the notation $y(k+i|k)$ means that the value was predicted i steps into the future based on the information available at step k. The cost function is solved at each time t by applying

$u_t, \ldots, u_{t+k-1}$ to the model and using the predictions of the new state vector $x_{t+k|t}$, which denotes the state predicted k steps into the future based upon the information available at time t.

The optimization problem is then implemented as a minimization of the quadratic cost function, over the prediction horizon H_p, such that the constraints hold. Thus, the standard RHC optimization is a quadratic programming (QP) problem. The constraints are usually defined in a form of linear inequalities encompassing the state variables and the control inputs such that:

$$x_{\min} \leq x \leq x_{\max} \tag{2.20}$$

$$u_{\min} \leq u \leq u_{\max} \tag{2.21}$$

where $x_{\min}$, $x_{\max}$, $u_{\min}$, and $u_{\max}$ are constant vectors. Furthermore, the constraints can also be placed on the output ($y_{\min} \leq y \leq y_{\max}$) or even on the input rates ($\triangle u_{\min} \leq \triangle u \leq \triangle u_{\max}$). In the nonlinear case, the constraints may be defined as belonging to a set of allowable values such as $x \in \mathcal{X}$ and $u \in \mathcal{U}$. In the absence of constraints and when the input and prediction horizons both approach infinity, the linear RHC problem becomes the LQR problem presented earlier.

In the proceeding sections, unless it is stated otherwise, it will be assumed that $H_p = H_u = H$.

2.3.2 Development of RHC

There are a number of excellent references that summarize the development of linear and nonlinear RHC algorithms. While we will provide a concise background for the evolution of RHC methods and theory, we recommend a review of these works for further study. For example, in [16], Morari and Lee give an overview of the origins of predictive control and present some current and future research directions. In [8], Findeisen and Allgöwer give an introduction specifically to nonlinear predictive control. Also, in [17], Qin and Badgwell provide an overview of commercial predictive controllers for both linear and nonlinear systems.

Some of the early work in the area of predictive control started in the late 1970s and early 1980s with the development of the IDCOM, DMC, and GPC algorithms. In 1976, Richalet et al. presented the first report of model predictive control applications at a conference and later summarized their model predictive heuristic control algorithm in [5]. They presented corresponding software called IDCOM (for IDentification-COMmand) that, based on an impulse response model and a reference trajectory, would compute a vector of future inputs. The dynamic matrix control (DMC) algorithm was presented at a conference in 1979 by Cutler and Ramaker and later defined in [18]. Developed at Shell, DMC used step or impulse response models to compute optimal control input for multivariate systems with constraints. Another related algorithm called Generalized Predictive Control (GPC) developed in 1980s by Clarke et al. was presented in [19, 20]. They described an adaptive control method that used transfer function models. It included more features than IDCOM or DMC but the impulse and step response models were easier to use for the chemical industry, [16], thus GPC did not gain as much industrial acceptance.

Both IDCOM and DMC algorithms had great impact on industrial process control and initiated the development of model predictive control theory [17]. While industrial application algorithms continued to improve and add new features, the academic research community was stimulated by the success of RHC and started to develop a theoretical foundation that would address the stability issues. Currently, most of the RHC literature is presented in the state-space form and is divided into linear and nonlinear research areas. The theoretical RHC research for linear models is very well advanced. The nonlinear models (and constraints) pose another set of challenging research problems, although much progress has been made.

The following sections summarize the main results of that research and underline the aspects that are used in the development of the theoretical framework for our VTC algorithm.

2.4 Theoretical background

The predictive control algorithm has been applied in many industrial projects with great success. However, the theoretical aspects of this approach, such as stability and performance, are difficult to prove especially for complex systems. The initial formulations of stability theory relied mostly on linear algebra thus the proofs could only be applied to simple linear systems with no noise or constraints. With the use of Lyapunov theory it became possible to prove stability for a wider range of systems. Lyapunov stability theorem states that if there exists a positive-definite Lyapunov function V such that $V \to \infty$ as $\|x\| \to \infty$, then the system is asymptotically stable in the region where $\dot{V}(x) < 0$ for all $x \neq 0$ [1]. This theory, called Lyapunov's second method in the work of Kalman and Bertram [21, 22] who showed its application, is currently being used in various attempts to prove stability of nonlinear systems with constraints. However, nonlinear RHC theory is still an open area of research.

One of the critical objectives of the thesis is to provide a proof for the stability of the variable transition controller. The proof method will closely follow the approaches presented in the RHC literature. Thus, it is important to review the theoretical advancements in this area. While there exists an extensive body of work on the stability and performance of linear RHC systems, the theoretical aspects of nonlinear RHC are still largely under investigation. Often the nonlinear methods employ linearization of the system's model, which is then used for optimal control input computation as in the linear case. This control input is then used in the nonlinear plant and the model is updated.

The following section summarizes the major advances in the stability theory of predictive control for linear and nonlinear systems. More detailed surveys on the topic of predictive control stability are presented in [14, 16]. Here, we repeat some notation for convenience. A linear system is defined as

$$x' = Ax + Bu \tag{2.22}$$

where A and B are matrices of the state x and control input u respectively. (The x and u quantities could be vectors but we will use scalars for ease of notation.) For a general, possibly nonlinear, system we will use the notation

$$x' = f(x, u) \tag{2.23}$$

and for the general form of a cost function we use

$$\mathcal{J} = \int_{t_0}^{H} l(x, u)\, d\tau \tag{2.24}$$

where H is the horizon length and $l(x, u)$ is usually a quadratic function of state and control input, as in

$$l(x, u) = x^T Q x + u^T R u. \tag{2.25}$$

2.4.1 Stability of predictive control

There are two main approaches used to show that RHC cost function is a Lyapunov function. The first one involves constraining the original problem formulation, while the second follows an argument for monotonicity of a sequence of cost functions.

The constrained approach, also called the *direct method* in [14], uses the cost function in (2.24) as a Lyapunov function and adds to the problem formulation one or a combination of the following constraints:

- **terminal equality constraint**: $x(t + H) = 0$

  This term enforces stability within a finite horizon by forcing the state of the system to the origin in finite time. The disadvantage of this method is that this constraint might be difficult to satisfy for systems with short horizon lengths. Also, since the terminal constraint is not met during operation, it is somewhat artificial and can lead to aggressive controller response.

- **terminal region/set constraint**: $x(t + H) \in \Omega \subseteq \mathcal{X}$

  This constraint term is set so that the controller steers the state to the terminal region Ω within finite time. This term cannot be chosen freely, rather it must be chosen so that closed-loop stability is achievable. An example of the off-line computation procedure for this term is discussed in [8].

- **terminal cost term**: $F(x(t + H))$

This penalty term is added to the cost function in (2.24) such that

$$\mathcal{J} = \int_{t_0}^{H} l(x,u)\, d\tau + F(x(t_0 + H)). \tag{2.26}$$

Again, just like the terminal region, this penalty term must be determined such that stability is ensured. This term usually gives an upper approximation of the infinite horizon cost function. For example, it can approximate the *cost-to-go*: $\int_{t_0+H}^{\infty} l(x,u)\, d\tau$.

The use of the terminal equality constraint was presented in the work of Keerthi and Gilbert in [23]. There the authors investigated nonlinear, discrete-time, and constrained systems and used a cost function as a Lyapunov function candidate for their stability argument. The cost function used in this proof was not quadratic, rather a general definition was used, $J_i = \sum_{k=i}^{\infty} l_k(y_k, u_k)$, where y is the output and u is the control input. It was shown that as the horizon is extended to infinity ($H \to \infty$) then the infinite cost for the receding horizon controller approaches the optimal infinite horizon cost ($\mathcal{J}_{H\to\infty} \to \mathcal{J}_{\text{opt}}$).

Similar argument was used by Mayne and Michalska in [7], but for the continuous system representation. Again, the authors made use of the Lyapunov stability theorem and added a terminal state constraint to the optimization problem. The follow-up work presented in [24] extended the work to nonlinear systems with constraints and relaxed the terminal constraint. The variable horizon presented in that work used a variable prediction horizon and a terminal region constraint ($x(T) \in \Omega$) instead of the terminal equality constraint. The resulting dual-mode controller used a local linear feedback control when the system was inside the terminal region and a receding horizon controller when outside that region.

Another approach presented by Jadbabaie, Yu, and Hauser in [25], proved stability for unconstrained, finite horizon RHC by adding a terminal penalty term to the cost function. The terminal cost, $F(x(t + H))$, was used as an approximation of the cost-to-go of the infinite cost function. This approach did not require a terminal stability constraint as the previous two approaches. The rest of the proof followed similar Lyapunov theory arguments in showing the improvement property of the cost function.

An interesting quasi-infinite horizon approach was developed by Chen and Allgöwer in [26]. There, the authors used a terminal region constraint and a cost function that consisted of a finite horizon cost and a terminal cost. The terminal cost was bounded by the infinite-horizon cost, thus the name *quasi-infinite*, and the bound was computed with the use of a local linear feedback law ($u = Kx$). This feedback control was never implemented, it was only used to compute the terminal penalty term and the terminal region bounds.

The constrained approaches discussed so far introduced constraints into the RHC problem that were not motivated by physical restrictions or performance requirements. Rather their exclusive purpose was to enforce stability in the closed-loop for the finite horizon cost. The terminal equality constraint was deemed as an artificial constraint [14, 26] with possible implementation drawbacks. While the terminal region and penalty cost introduced additional computational burden since they could not be chosen freely, they must be computed to ensure stability.

The second approach to proving stability of RHC with a finite horizon uses an argument based on the monotonicity property for a sequence of cost functions. In this approach, it is shown that a cost function of horizon length H at step i is greater than or equal to the cost function of horizon length H at step $i+1$ when the horizon is large enough. The main idea of this approach was used by Chen and Shaw in [27] and later by Shamma and Xiong in [28] and by Nevistic and Primbs in [29].

In Nevistic and Primbs [29] and Primbs and Nevistic [30], the authors showed a stability proof for linear, constrained systems over a finite horizon that did not rely on terminal constraints or penalty terms like the direct approaches. Here, the optimal cost function was used as the Lyapunov function candidate and the objective was to show that

$$J_H(x(k)) - J_H(x(k+1)) > 0 \text{ for } x \neq 0. \tag{2.27}$$

The authors then proceeded by arguing that there exists a finite horizon length, H^*, such that equation (2.27) holds for any $H > H^*$.

The proof is straightforward to follow and is not as restrictive as other methods. Thus, the proof method in the thesis will follow the arguments outlined in [28, 29, 30] and extend them to prove the stability of variable transition models.

2.4.2 Performance

The receding horizon approach proposes a different implementation for optimal control. It solves a finite horizon optimal control problem on-line for the current state of the system rather than computing (off-line) a feedback control that is optimal for all states. Finding a feedback solution is not feasible in general since it requires computing a solution to the Hamilton-Jacobi-Bellman differential equation (or difference equation in the discrete case) [2].

However, normally it is not true that repeating a minimization over a finite horizon will result in the same optimal solution as the infinite horizon computation.

An infinite horizon formulation would be ideal but it is not practical since it cannot be solved sufficiently fast. Shorter horizons are preferred from a computational standpoint but they might result in the finite and infinite horizon solutions being considerably different.

The performance of the receding horizon controller over infinite horizon is often addressed in the same setting as the stability proof. For example, in the nonlinear constrained case, Chen and Allgöwer [26] chose their terminal penalty cost term, $F(x(t+H))$, so that the local control law was a good approximation of the infinite horizon cost. They further argued that using this approach results in a finite horizon controller that can recover the performance of the infinite horizon cost even for short horizons. These problems are subject to research that attempts to quantify the performance of the finite horizon controller over the infinite horizon.

For the linear case, the basis of which is used in our performance analysis, Nevistic and Primbs [29] provided a bound calculation that was used to ensure that the cost of their receding horizon controller will perform within a specified tolerance of the optimal infinite horizon performance.

A comprehensive survey of receding horizon stability and performance methods can be found in [14].

2.5 Other related controllers

The receding horizon approach is not the only sophisticated alternative to simple feedback control. Extensions to RHC theory have been made to accommodate systems where manipulated variables are controlled at different rates such as in Multirate MPC. Also, some research has been done on switched control where the switching times are optimized, although there is no prediction of the system behaviour. Finally, a lot of research is currently being done in the area of hybrid system control with and without the receding horizon approach.

- **Multirate MPC**:
  This extension to the receding horizon algorithm is discussed, for example, in [31]. Here the authors outline an approach that can be used to accommodate systems that require, usually due to technological constraints, that the system measurements and the control input computations be done at different sampling frequencies. This is not the same as the variable switching time approach presented in the thesis. In the thesis, the switching times will not be dictated by the system rather, we will compute their optimal values.

- **Variable time control**:
  An example of this form of control has been presented in the work of Lee et al. in [32]. Here the system is parameterized over a finite set of control inputs $\{u_1, u_2, \ldots, u_n\}$ and an optimization problem, based on a performance objective, is formulated in order to determine a sequence of optimal switching times that will be used to switch between the pre-defined set of control inputs. This approach is not predictive and the control inputs are not computed with the switching times.

- **Switched systems**:
  An excellent introduction to switched system control is given by Liberzon in [33]. Examples of switched system include variable structure control systems (VSCS) which use sliding mode control (SMC). SMC determines a switching

surface that the state trajectory should remain on and uses a switching control law, for example

$$u = \begin{cases} u^{+} \text{ when } s(x) > 0, \\ u^{-} \text{ when } s(x) < 0 \end{cases} \tag{2.28}$$

(where $s(x)$ is a switching function based on the state) to drive the plant on to that surface. A good introduction to this control strategy was presented early on by Utkin in [34], but a more detailed overview was given in a book by Edwards and Spurgeon [35].

This control scheme is attractive in nonlinear systems since the problem statement can include a term for uncertainties or nonlinearities in the model. However, an undesirable motion called *chattering* can result if the states frequently cross the switching surface rather than remaining on it.

- **Hybrid MPC**:

  A lot of recent research in the field of RHC is focusing on extensions to hybrid systems. Hybrid systems are a mix of continuous and discrete time systems, for example, when a continuous plant uses a digital time controller. An example of a hybrid system is shown in Figure 2.4. The mode transitions can be based on timed events or can be triggered by state or input. The issues of stability and performance of hybrid model predictive controllers have been addressed in, for example, [36]. A very detailed investigation of stability and robustness has been presented in the Ph.D. thesis of Lazar [37].

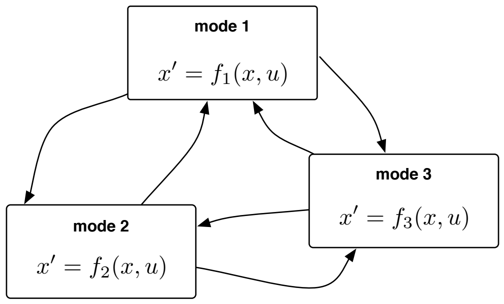

Figure 2.4: Hybrid system with three modes

2.6 Overview of commercial process control tools

The following is a brief summary of some of the commercial tools developed for RHC synthesis. All of them are based on one of the RHC schemes mentioned in earlier sections.

2.6.1 DMCplus

DMCplus is an extension of the DMC (Dynamic Matrix Control) tool. DMC was originally built for Shell. Its developers then formed the DMC Corporation that was bought out by Aspentech, a company that now owns DMCplus. According to [15], DMCplus is the most widely known and used tool out of the commercially available predictive control products. It incorporates a GUI system into its development software. The software can interface directly with a distributed control system or indirectly through a process information management system. The controller design is focused on running the processes close to their constraint boundaries in order to maximize the profits. The process models in DMC are represented by multivariable finite step response models. The step response matrix , called the Dynamic Matrix, is formed to model the plant. The control input is determined by first computing a set of steady-state points for the controller variables and then computing the future control trajectory over a finite prediction horizon. The cost function and the linear inequality constraints are evaluated in order to find the set points. This computation involves solving the linear programming (LP) problem for the target levels. The future control trajectory is computed over the control horizon by solving the least squares problem. It also entails a quadratic penalization of the control moves.

This tool is available from Aspentech [38].

2.6.2 Connoisseur

This software is supplied by invensys [39] (originally Predictive Control Ltd. of England). The algorithm uses an LP technique, a steady-state model, and a cost function to derive the optimal operating point while it aims to minimize energy usage and maximize throughput. According to the description in [15], it offers three control modes, 'LR' for Long Range, 'LR-QP', and 'QP'. This choice is available so that the computational effort can be reduced. The LR mode optimizes a quadratic cost function while initially ignoring any constraints. If it predicts constraint violations then the weights in the cost function are modified until such violations are avoided. LR mode uses a linear model and solves the resulting Linear Quadratic problem that has a closed form solution. This is done by solving the Riccati equation. The 'LR-QP' mode optimizes the quadratic cost function while taking into consideration the hard constraints. The output constraints are disregarded. Again, the weights of the cost function are adjusted to avoid any constraint violations. Finally, the 'QP' mode includes all the input and output constraints while it solves the optimization problem but all the constraints are soft. The most recent brochure mentions the use of neural network technology to provide nonlinear modeling from process data. The dynamic model of the process is constructed from the process data by recursively applying least squares techniques. The Connoisseur package also contains statistical tools and should be compatible with most major computer and control equipment suppliers.

2.6.3 Process Perfecter

This software package is produced by Pavilion Technologies. Its application areas include mostly nonlinear and multivariable process industries. It uses artificial neural networks as the modeling foundation for the tool's control algorithm. This is based on the assumption that any nonlinearities can be detected as changes in the steady-state gain. Perfecter uses steady-state optimization and dynamic control models to control the plant toward its optimal set of operating conditions. For a mathematical introduction to how neural networks are used within Perfecter see [15]. The software package allows the user to include all the information about process constraints, economic objectives, and other relevant model parameters. The extensive set of plotting options is complemented by the intuitive GUI. An additional Perfecter Builder package is used to interface with a distributed control system. All of these tools are available from Pavilion Technologies [40].

2.6.4 PFC, HIECON

PFC and HIECON (or IDCOM-HIECON) are both maintained by a French company Adersa. These two tools use Predictive Functional Control (PFC) to formulate their

MPC problem, see [15]. PFC is the original CAD software tool that is used for mono-variable systems. The model is obtained through GLIDE software from a recorded plant test. Both PFC and GLIDE run in the MATLAB environment.

HIECON uses similar methodology but it is to be used for multivariable systems. The model representation used is the step response or the impulse response that fits the target process. It consists of two parts. There is CAD software that includes the controller design and testing tools and then there is also the controller written in Fortran 77, which allows for the easy transfer of the application from the PC to the target process computer. HIECON also includes more algorithms for solving the control problem, including fully constrained optimization algorithm.

Chapter 3

Towards Variable Step Size

The receding horizon control methods described in the previous chapter use a fixed sampling time at which they update the optimal control input sequence. This sampling frequency combined with the horizon length determine the number of steps taken within a single horizon length and thus the number of optimal control inputs that must be computed within a single iteration of the control loop. The sampling must be tuned carefully since at every step, we must solve the cost function optimization problem and update the controller. The variable switching time controller presented in this thesis is, in part, a step toward eliminating the tuning and thus generalizing the RHC method. However, this work does not represent the initial step in improving the existing receding horizon algorithms. The variable time step idea evolved from research presented in [41] into another model predictive algorithm that would use a combination of pre-determined control inputs, instead of computing an *optimal* one, and would switch between these inputs as required based on the constraint evaluation and the cost function computation.

We present a brief description of this method and discuss the limitation that led to the development of the predictive control algorithm with a variable step size that is the focus of this thesis work.

3.1 Mode switching controller

The control algorithm, as summarized in [9], uses a closed-form solution (existence assumed) of a general model $x' = f(x, u)$ where $x \in \mathbf{R}^n$, $u \in \mathbf{R}^m$, $f : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^n$. The important difference is that here, u is a set of possible control functions, Cf_i, such that $u = \{\mathsf{Cf}_1, \mathsf{Cf}_2, ..., \mathsf{Cf}_n\}$ where $\mathsf{Cf} : \mathbf{R}^n \to \mathbf{R}^m$. The parameterized closed-form solution, $\Phi(\mathsf{Cf}_i, x_{\text{init}}, t_{\text{init}}, t_{\text{end}})$ is a function of the initial state, x_{init}, initial time t_{init}, end time t_{end}, and the control function for mode i, Cf_i.

The objective of the controller, in general terms, is to steer the plant toward a given set point s, while ensuring that none of the constraints are violated and the cost function is minimized. The constraints used in this algorithm are limited to linear constraints, these can restrict the control function, $u_{\text{min}} \leq u \leq u_{\text{max}}$, or the state variables such that $x_{\text{min}} \leq x \leq x_{\text{max}}$. It also allows for the constraints to be placed on the rates of change of u and x. The cost function most often used is a quadratic function of the state and the control input.

This control algorithm explores all the possible combinations of control functions $\{\mathsf{Cf}_1, \mathsf{Cf}_2, ..., \mathsf{Cf}_n\}$ over a number of steps N, by creating a *search tree* for all of the configurations of the control modes and then selects the sequence that results in the lowest cost at the end of the horizon length without violating any constraints. Thus, at the end of N steps, it must compare the cost functions for all the 2^N combinations of the control inputs $\mathcal{J}_{[u_1,u_1,u_1]}, \mathcal{J}_{[u_1,u_1,u_2]}, \ldots, \mathcal{J}_{[u_2,u_2,u_2]}$. This idea is shown in Figure 3.1 for three steps with two control inputs $u = \{u_1, u_2\}$.

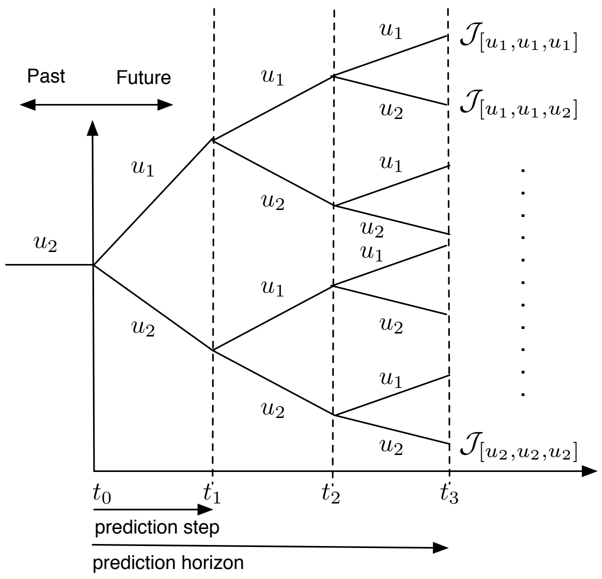

Figure 3.1: Predictive mode switching

The first input in the sequence, Cf_i, is then applied to the plant and the state is updated. Since the set of all possible control inputs is limited to n, it is possible to remain within a single mode for a number of steps without switching the control input. The input is switched only if it violates the constraints or a lower cost function can be obtained by using a different input. Otherwise, the system remains in mode Cf_i.

Note that this approach differs from the variable time control presented in [32] and discussed in the previous chapter, which computes an optimal time for the controller to switch between the members of a set of predefined constant control inputs. That method is not predictive, it does not use the receding horizon strategy. It also finds $n-1$ switching times for the n control inputs and all of the control inputs must be used once within the time interval. The same control input cannot be used multiple times.

The example below is a demonstration of the above control algorithm followed by a discussion of the limitations of the mode switching controller that led our research toward the variable step size and influenced the development of VTC.

3.2 Example: Two Tanks

The two tank system has been used in the hybrid systems literature and more detailed description can be found in [42]. This system is shown in Figure 3.2. The two state

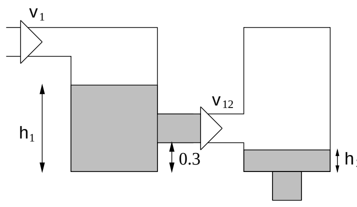

Figure 3.2: Two tank system diagram

variables h_1 and h_2 represent the fluid height in tanks 1 and 2 respectively. Tank 1 is being filled with a fluid that is being pumped at a constant rate and the flow is controlled by a valve, v_1, that can be turned on ($v_1 = 1$) or off ($v_1 = 0$). Tank 2 is connected to tank 1 through a unidirectional valve $v_{12} \in \{0, 1\}$, which controls the fluid flow from tank 1 to tank 2. The fluid is drained out of tank 2 at a constant rate a_2. The linearized dynamics of the system are described by

$$\begin{aligned} \dot{h}_1 &= a_1 v_1 - b v_{12}(h_1 - 0.3) \\ \dot{h}_2 &= b v_{12}(h_1 - 0.3) - a_2. \end{aligned} \tag{3.1}$$

Here, a_1 and a_2 are constants that depend on the flow rates into tank 1 and from tank 2 respectively. Constant b depends on the flow rate from tank 1 to tank 2.

Figure 3.3 shows the simulation results of this system within our framework for the flow rates set at $a_1 = 1$, $b = 1$, and $a_2 = 0.2$. These results correspond to the results discussed in [42]. As shown in Figure 3.3, the height of fluid in tank 1 changes more rapidly than in tank 2 since the fluid flow rate into tank 1 is greater.

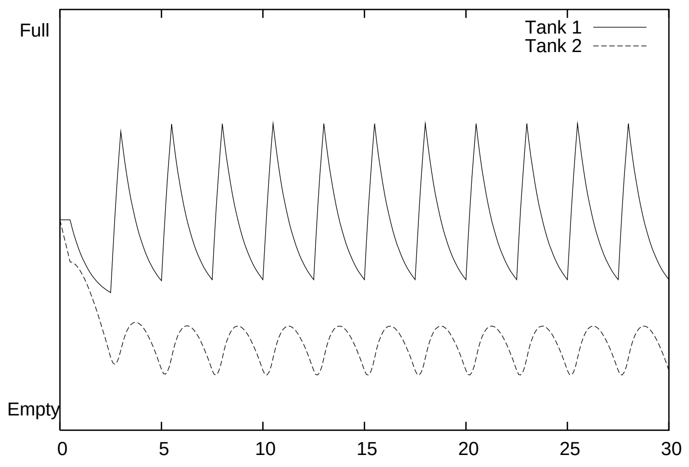

Figure 3.3: Water level of tank 1, h_1, and tank 2, h_2, versus t

Figure 3.4 shows the switching of the control inputs that change the state of the valves. Again, valve v_1 is opened for a much shorter period of time than valve v_{12} due to a higher flow rate.

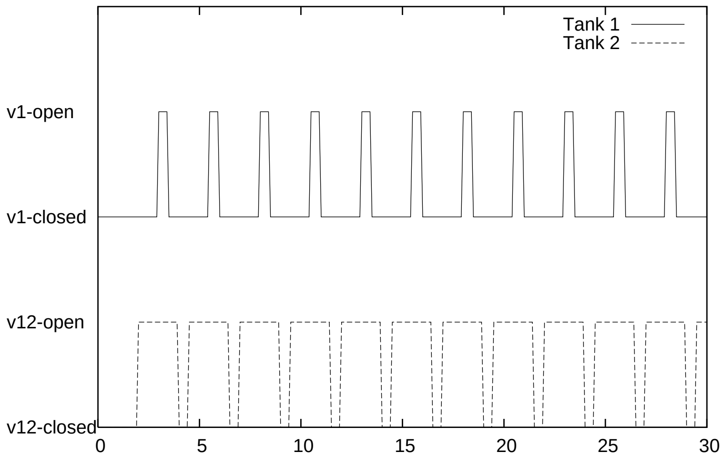

Figure 3.4: Control of valve v_1 and v_{12}, versus t

3.3 Evaluation

The mode switching controller presented a novel model-based control approach that demonstrated many computational advantages. It was interpreted as a method with variable time switching since the controller could remain in a particular mode as long as it was cost effective to do so and no constraints were violated. One of the advantages of this approach was that the control function set was not limited to constant control inputs only, the control functions Cf_i could define any form of control input, for example, even a PID controller. As such, no optimization problem had to be solved to compute an optimal control input. In fact, the entire problem was simplified to expression evaluations since closed-form solutions were used to update the state of the system. It also naturally incorporated constraint evaluation into the control problem.

However, one of the evident disadvantages of this method was the potentially large search space that had to be created to account for all the possible mode configurations. This combinatorial effect imposed a practical limitation on the number of steps used in the implementation of mode switching, since for every N steps, with n control inputs, we had to compute cost functions for n^N possible combinations. It was also up to the engineer to propose the set of control functions, which might be trivial in cases where only on/off control is desired but not for more complex models. As a result, it became apparent that computing an optimal control value presents an advantage of having the *best* possible control input for the model as opposed to picking a member out of a limited set.

The closed-form solution used in the control loop computation was initially perceived as a computational advantage since it reduced the run time problem to expression evaluations. As powerful as this concept was, it also limited the application of this method to systems with closed-form solutions. This manifested a need to use numerical solution methods for all other models. The research was split into two branches. One branch of research continued to investigate and improve upon the mode switching controllers [43]. Another branch of research, which is a source of this thesis, began to investigate numerical approaches to mode switching time computation where the modes are based on optimal control inputs.

The idea for variable step size computation is based on a concept used in adaptive numerical integration. An adaptive integration method must determine how large a step size should be to achieve a desired accuracy. The method may take very small steps in fast changing regions and larger steps when the changes are slow, as shown in Figure 3.5.

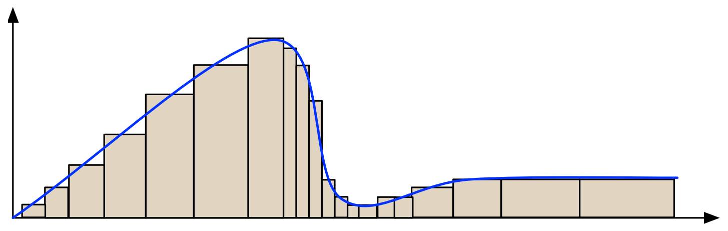

Figure 3.5: Adaptive integration step

This integration approach allows the solver to be more accurate when the derivative is large by taking smaller steps in that region, and be very efficient by taking large steps when there is not much change in the system.

The research, motivated by the adaptive integration approach, was directed toward a method that would compute an optimal control input sequence but would not update that sequence at a fixed sampling frequency. Instead, it would compute a time point at which it became optimal to do the update. The time step computation would not be based on a measure of accuracy, rather a performance measure (the cost function) would be used to find the optimal step size. This required adding a sampling time to the control problem and resulted in Variable Transition Control method defined in the following chapter.

Chapter 4

Variable Transition Control

The receding horizon control method defined in the second chapter computes a sequence of control inputs that are updated using fixed sampling frequency. At every step, only the first control input is applied to the plant, the rest of the sequence is discarded and a new one is computed. The computation results are tuned, as necessary, by manipulating the parameters of the RHC problem, such as the horizon length H, the weighting matrices of the cost function Q and R, and the sampling time (the number of steps, N, within a single horizon length H). This tuning is largely a manual trial-and-error effort.

Our motivation is to generalize the RHC algorithm and to develop a method that decreases the dependence on tuning by optimizing the time points at which a new control sequence is computed. We present a Variable Transition Control (VTC) algorithm that, in addition to computing a sequence of optimal control inputs over a horizon length H, also computes a sequence of optimal switching times at which a new control sequence is computed. The optimization is based on a definition of a cost function, as in RHC, where the time points are part of the optimization problem.

This idea is loosely based on adaptive step-size methods developed for numerical integration that use large steps in regions where the function behavior is smooth thus improving efficiency, and smaller steps in regions where it fluctuates. The VTC method presented in the subsequent sections results in an approach that eliminates the need to tune sampling time, potentially decreases the number of expensive model evaluations, and presents a generalization of the RHC law.

4.1 Problem Definition

Consider a system defined by a set of differential equations

$$\mathbf{x}(t)' = f(\mathbf{x}(t), \mathbf{u}(t)) \tag{4.1}$$

where the function $f : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^n$ locally satisfies the Lipschitz condition [44] and the initial condition for (4.1) is

$$\mathbf{x}(t_{\text{init}}) = \mathbf{x}_0 \tag{4.2}$$

with $\mathbf{x} \in \mathbf{R}^n$ representing the state vector and $\mathbf{u}$ representing a piecewise constant function such that $\mathbf{u} : [t_{\text{init}}, t_{\text{end}}) \to U$. The set U is a finite set in $\mathbf{R}^m$. The switching points are defined as a sequence of variably spaced points $\tau_1, \tau_2, \ldots, \tau_{\text{N}}$ such that $t_{\text{init}} = \tau_0 < \tau_1 < \tau_2 < \ldots < \tau_{\text{N}} < \tau_{\text{N}+1} = t_{\text{end}}$ (N represents a fixed number of switchings).

The solution $\mathbf{x}(t)$ for (4.1) is obtained in a piecewise manner by integrating over each interval $[\tau_{\text{i}}, \tau_{\text{i}+1}], i = 0, 1, 2, \ldots, N$. The system solution, $\mathbf{x}(t)$, is continuous and piecewise differentiable on $(t_{\text{init}}, t_{\text{end}})$. The interval length $t_{\text{end}} - t_{\text{init}}$ will be referred to as the horizon length, H.

The optimal control problem may now be formally stated as: Given the dynamical system (4.1)–(4.2), find the sequence of control inputs $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{N+1}] \in U$ and their corresponding switching points, $\mathbf{T} = [\tau_1, \tau_2, \ldots, \tau_N]$, such that the cost functional

$$\mathcal{J}_H = \sum_{i=0}^{N} \int_{\tau_i}^{\tau_{i+1}} l(\mathbf{x}, \mathbf{u})\, dt \tag{4.3}$$

is minimized. The function $l(\mathbf{x}, \mathbf{u})$ will in general be defined by the quadratic function:

$$l(\mathbf{x}, \mathbf{u}) = \mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T R \mathbf{u} \tag{4.4}$$

where Q and R are positive definite weighting matrices. Let $\mathbf{U}$ and $\mathbf{T}$ be as defined above, then the control problem $P_{\mathbf{U},\mathbf{T}}$ can be simply restated as the following optimization problem:

$$\min_{\mathbf{U},\, \mathbf{T}} \mathcal{J}_H(\mathbf{x}, \mathbf{u}). \tag{4.5}$$

In order to solve the proposed optimization problem $P_{\mathbf{U},\mathbf{T}}$, we need to compute the sequence of $N+1$ control inputs and N switching times

$$\mathcal{S}_N = [(\mathbf{u}_1^*, \tau_0), (\mathbf{u}_2^*, \tau_1^*), \ldots, (\mathbf{u}_N^*, \tau_{N-1}^*), (\mathbf{u}_{N+1}^*, \tau_N^*)]. \tag{4.6}$$

Note that τ_0 and τ_{N+1} are known since they are the initial and final time points respectively.

Any $\mathbf{u} \in U$ can be written as a piecewise constant function:

$$\mathbf{u}(t) = \mathbf{u}_{\text{i}},\ t \in [\tau_{\text{i-1}}, \tau_{\text{i}}]. \tag{4.7}$$

We use the notation $\mathbf{u}^*_{\text{i}}$ and τ^*_{i} to denote the optimal values (with respect to the cost function (4.3)). The entire sequence of optimal control inputs can be derived by representing the control function as a piecewise continuous function as defined in (4.8),

$$\mathbf{u}(t) := \begin{cases} \mathbf{u}^*_1, & t \in [t_{\text{init}} = \tau_0, \tau^*_1) \\ \mathbf{u}^*_2, & t \in [\tau^*_1, \tau^*_2) \\ ... & ... \\ \mathbf{u}^*_{\text{N}}, & t \in [\tau^*_{\text{N-1}}, \tau^*_{\text{N}}) \\ \mathbf{u}^*_{\text{N}+1}, & t \in [\tau^*_{\text{N}}, \tau_{\text{N}+1} = t_{\text{end}}). \end{cases} \tag{4.8}$$

Only the first control input in (4.6) is applied to the system at the corresponding time point. The remainder of the sequence is discarded and the problem is recomputed starting at τ^*_1. The piecewise control input strategy is shown for one horizon length in Figure 4.1.

The overall structure of the VTC control loop is shown in Figure 4.2. Note that the state estimator (observer) is not implemented as part of our control strategy since we assumed no disturbances on the model. We can thus update the model as necessary.

The main steps of the VTC scheme are as follows:

1. obtain current (initial) state of the system

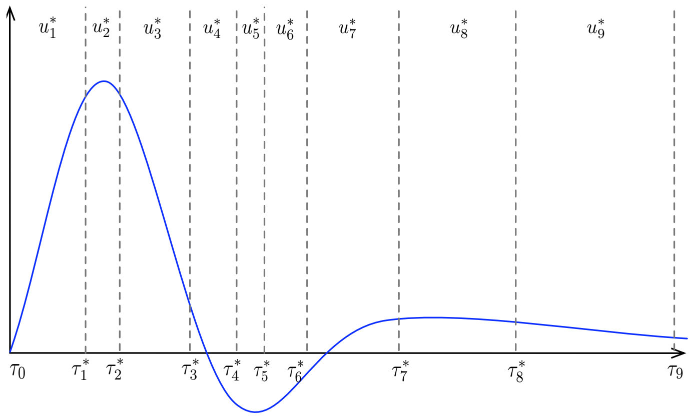

Figure 4.1: Variable switching time strategy with $N = 8$

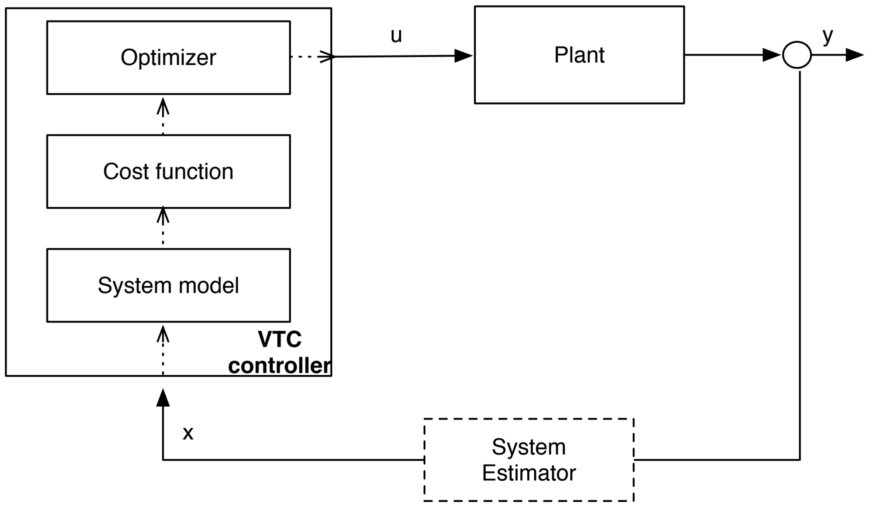

Figure 4.2: Variable time control loop

2. update the cost function and compute optimal switching time and control input by minimizing the cost function over a specified horizon length using the model of the system

3. implement the first control input in the sequence and obtain a new estimate

(measurement) of the state at the first time point in the sequence

4. continue with 1. until the switching time $\tau_i^* \geq t_{\text{end}}$.

The problem in step 2, $P_{\mathbf{U},\mathbf{T}}$ as defined in (4.5), is hard to solve in general since, as we increase N, the number of decision variables increases, thus increasing the complexity of the optimization problem. We face an additional level of complexity over RHC since we have almost doubled the number of decision variables $(2N - 1)$. However, as a side-effect of our approach, we may need fewer model evaluations in cases where larger steps are sufficient.

Since we have added a switching time sequence to the optimization problem, we cannot use the RHC standard dynamic programming approach that uses a closed form solution to the Algebraic Riccati Equation (ARE). Early attempts to formulate and solve the models in (4.1) symbolically with the control strategy represented as a piecewise function (4.8) have resulted in very long simulations even for a small number of switching points. An approximation method was developed for the system (4.1) - (4.8) in order to simplify the computational complexity of the problem.

4.2 Problem Solution

We can approximate a solution for the system defined in (4.1) using first order Euler approximation [45]. The general forward Euler formula is given by

$$\mathbf{x}(t_{\text{init}} + h) = \mathbf{x}(t_{\text{init}}) + hf(\mathbf{x}(t_{\text{init}}), \mathbf{u}(t_{\text{init}})) \tag{4.9}$$

where t_{init} is the initial time point and h is the fixed step size. Using (4.1) we infer that $f(\mathbf{x}(t_{\text{init}}), \mathbf{u}(t_{\text{init}})) = \mathbf{x}'(t_{\text{init}})$, thus (4.9) can be expressed as

$$\mathbf{x}(t_{\text{init}} + h) = \mathbf{x}(t_{\text{init}}) + h\mathbf{x}'(t_{\text{init}}). \tag{4.10}$$

Or, similarly, using Taylor series expansion defined by

$$f(x) = f(a) + f'(a)(x - a) + \ldots, \tag{4.11}$$

the expansion of x in h about t_{init} is

$$\begin{aligned} \mathbf{x}(t_{\text{init}} + h) &= \mathbf{x}(t_{\text{init}}) + (t_{\text{init}} + h - t_{\text{init}})\mathbf{x}'(t_{\text{init}}) \\ &= \mathbf{x}(t_{\text{init}}) + h\mathbf{x}'(t_{\text{init}}). \end{aligned} \tag{4.12}$$

Either approach can be applied to models defined by (4.1) but instead of a fixed step size, we incorporate variable steps.

We can include variable step sizes using:

$$\begin{aligned} \mathbf{x}(\tau_{\text{i+1}}) &= \mathbf{x}(\tau_{\text{i}}) + (\tau_{\text{i+1}} - \tau_{\text{i}})\mathbf{x}'(\tau_{\text{i}}) \\ &= \mathbf{x}(\tau_{\text{i}}) + (\tau_{\text{i+1}} - \tau_{\text{i}})f(\mathbf{x}(\tau_{\text{i}}), \mathbf{u}(\tau_{\text{i}})) \end{aligned} \tag{4.13}$$

for $i = 0, 1, 2, \ldots, N$. Using (4.7), the above equation can be re-written:

$$\mathbf{x}(\tau_{\text{i+1}}) = \mathbf{x}(\tau_{\text{i}}) + (\tau_{\text{i+1}} - \tau_{\text{i}})f(\mathbf{x}(\tau_{\text{i}}), \mathbf{u}_{\text{i+1}})) \tag{4.14}$$

where $\tau_0 = t_{\text{init}}$, $\tau_{\text{N+1}} = t_{\text{end}}$, and $\mathbf{x}(\tau_0) = \mathbf{x}(t_{\text{init}}) = \mathbf{x}_0$ are known. We can now construct an entire chain of approximations for the entire horizon length H ($H = \tau_{\text{N+1}} - \tau_0$):

$$\begin{aligned}
\mathbf{x}(\tau_1) &= \mathbf{x}(\tau_0) + (\tau_1 - \tau_0) f(\mathbf{x}(\tau_0), \mathbf{u}_1) \\
\mathbf{x}(\tau_2) &= \mathbf{x}(\tau_1) + (\tau_2 - \tau_1) f(\mathbf{x}(\tau_1), \mathbf{u}_2) \\
&\vdots \\
\mathbf{x}(\tau_{\text{N}}) &= \mathbf{x}(\tau_{\text{N-1}}) + (\tau_{\text{N}} - \tau_{\text{N-1}}) f(\mathbf{x}(\tau_{\text{N-1}}), \mathbf{u}_{\text{N}}) \\
\mathbf{x}(\tau_{\text{N+1}}) &= \mathbf{x}(\tau_{\text{N}}) + (\tau_{\text{N+1}} - \tau_{\text{N}}) f(\mathbf{x}(\tau_{\text{N}}), \mathbf{u}_{\text{N+1}}). \qquad (4.15)
\end{aligned}$$

The solution approximation given in (4.15) can be used within the cost function defined in (4.3). The resulting cost function is then minimized using an optimization solver and the optimal values are computed for the trace (4.6).

4.3 Theoretical foundation

The development of RHC theory is unconventional in its inception since it was stimulated by the industrial success of the practical implementations of the algorithm. The developers of the early predictive control applications were aware of the importance of stability; thus, they made the controllers stabilizing by choosing a large enough horizon to account for the settling time of the plant. Academic research began to investigate stability within the RHC framework and the comprehensive review of this extensive body of work can be found in [14]. Currently, any academic extensions to the predictive control algorithms are accompanied by proofs for the stability and also the performance of the proposed method.

Below we present a complete theoretical framework for the variable transition control method. We begin by presenting a comparison of the cost of VTC versus the cost of RHC and show that the variable step size guarantees that the resulting VTC cost is lower than, or in the worst case equal to, the cost of RHC. Then we present a proof for the stability of the controller using a Lyapunov based method that employs a sequence of cost functions for the horizon length H to show that this sequence is decreasing. Finally, we present bounds for the performance of our controller over the infinite horizon.

4.3.1 Cost function comparison

The objective is to show that the cost function for the proposed variable transition control approach, $\mathcal{J}_{\text{VTC}}$, is at least as good as the cost function resulting from a fixed step receding horizon control approach, $\mathcal{J}_{\text{RHC}}$. We show that

$$\mathcal{J}_{\text{VTC}} \leq \mathcal{J}_{\text{RHC}} \tag{4.16}$$

over the same horizon length, H, with the same number of steps, N.

First, we will show that the relationship in (4.16) holds for linear systems defined by

$$x' = ax + bu, \tag{4.17}$$

to simplify the understanding of the proof, and then we will extend the proof to a more general system

$$x' = f(x, u). \tag{4.18}$$

Linear case

Let the control input be defined as a sequence of piecewise constant inputs $\mathbf{U} = [K_1, K_2, \ldots, K_{N+1}]$. We want to compare the effects of the RHC and the VTC approach on the cost function associated with solving the control problem. We first define the cost of the RHC control algorithm and then compare it with the VTC cost for the same system.

For the continuous system in (4.17), the RHC solution at time point t_{next} is approximated (discretized) using first-order Taylor series approximation as given in (4.9), which is suggested in [2]

$$x(t_{\text{next}}) = x(t_{\text{prev}}) + h(ax(t_{\text{prev}}) + bK_i) \tag{4.19}$$

where h defines a fixed step-size $t_{\text{next}} - t_{\text{prev}}$ for the forward step.

This can be applied to construct a chain of computations that approximate the solution to (4.17) using a piecewise constant control input from $\mathbf{U}$ over N steps

$$\begin{aligned}
x(t_1) &= x(t_0) + h(ax(t_0) + bK_1) \qquad (4.20)\\
x(t_2) &= x(t_1) + h(ax(t_1) + bK_2)\\
&\vdots\\
x(t_N) &= x(t_{N-1}) + h(ax(t_{N-1}) + bK_N)\\
x(t_{N+1}) &= x(t_N) + h(ax(t_N) + bK_{N+1})
\end{aligned}$$

where $t_{N+1} = t_{\text{end}}$ and $h = t_{i+1} - t_i$ for $i = 1, \ldots, N$. In practice the above chain can be easily constructed using a tail-recursive algorithm. RHC can then use this approximation (when the model is continuous instead of discrete) to construct the cost function used in the minimization problem

$$\begin{aligned}\mathcal{J}_{\text{RHC}} &= \int_{t_0}^{t_{N+1}} x^2 q + u^2 r \, d\tau \\ &= \sum_{i=1}^{N} \int_{t_i}^{t_{i+1}} x^2 q + u^2 r \, d\tau\end{aligned} \tag{4.21}$$

where q and r are scaling factors that can be used to tune the cost. For simplification, we will assume $q = r = 1$ for the remainder of this argument. Also, the optimal solution for variable x will be denoted with x^*. This cost function is then minimized over $\mathbf{U} = [K_1, K_2, \ldots, K_{N+1}]$ in order to find the optimal sequence of control inputs, $\mathbf{U}^* = [K_1^*, K_2^*, \ldots, K_{N+1}^*]$, for the current step, as defined by the optimization problem

$$\min_{\mathbf{U}} \mathcal{J}_{\text{RHC}}. \tag{4.22}$$

Only the first control input, K_1^*, from the resulting optimal sequence $[K_1^*, K_2^*, \ldots, K_{N+1}^*]$ is applied to the system, the rest are discarded and the optimal control problem is solved again after a single forward step is taken in the computation.

Since the cost function over the horizon length H is a Lyapunov function (from the stability proof) then it follows that as x increases, so does $\mathcal{J}_H$ (i.e. $x \to \infty \Rightarrow \mathcal{J}_H \to \infty$). This is also evident from the cost function definitions in (4.22) and (4.27) since they are both quadratic and dependent on x^2, thus the cost function over the horizon length H will grow with every step up to N. We will ignore the explicit control term $u^2 r$ in the cost functions since the control input is a constant term and $r = 1$. Thus, it is sufficient for us to examine the change in state with the optimal sequences computed from (4.22) and (4.27).

The state resulting from an RHC problem solution, computed using the chain of first order approximations for the entire horizon length H over N steps, results in a function g defined as

$$\begin{aligned} g_{\mathrm{RHC}}(t_0, t_1, \ldots, t_N, t_{N+1}, K_1^*, K_2^*, \ldots, K_{N+1}^*) &= x(t_{N+1}) \\ &= x(t_N) + (t_{N+1} - t_N)(ax(t_N) + bK_{N+1}^*) \end{aligned} \tag{4.23}$$

where $t_0, t_1, \ldots, t_{N+1}$, and $x(t_0)$ are known quantities, thus (4.25) can be updated to

$$\begin{aligned} g_{\mathrm{RHC}}(K_1^*, K_2^*, \ldots, K_{N+1}^*) &= x(t_{N+1}) \\ &= x(t_N) + (t_{N+1} - t_N)(ax(t_N) + bK_{N+1}^*). \end{aligned} \tag{4.24}$$

Now we apply the same approach to compute the cost function for the VTC algorithm. VTC does not use a fixed step-size, rather it attempts to compute the optimal switching sequence $\mathbf{T}^* = [\tau_1^*, \tau_2^*, \ldots, \tau_N^*]$ for the control sequence $\mathbf{U}^* = [K_1^*, K_2^*, \ldots, K_{N+1}^*]$. The approximation in (4.21) is defined to accommodate the step size change:

$$\begin{aligned} x(\tau_1) &= x(\tau_0) + (\tau_1 - \tau_0)(ax(\tau_0) + bK_1) \\ x(\tau_2) &= x(\tau_1) + (\tau_2 - \tau_1)(ax(\tau_1) + bK_2) \\ &\vdots \\ x(\tau_N) &= x(\tau_{N-1}) + (\tau_N - \tau_{N-1})(ax(\tau_{N-1}) + bK_N) \\ x(\tau_{N+1}) &= x(\tau_N) + (\tau_{N+1} - \tau_N)(ax(\tau_N) + bK_{N+1}). \end{aligned} \tag{4.25}$$

This approximation can now be used in the cost function for the VTC problem

$$\begin{aligned}\mathcal{J}_{\text{VTC}} &= \int_{\tau_0}^{\tau_{N+1}} x^2q + u^2r \; dt \\ &= \sum_{i=1}^{N} \int_{\tau_i}^{\tau_{i+1}} x^2q + u^2r \; dt.\end{aligned} \tag{4.26}$$

Again, we minimize the cost function in order to find the optimal sequence of control inputs but in this case, we also compute the optimal switching points

$$\min_{\mathbf{U},\mathbf{T}} \mathcal{J}_{\text{VTC}} \tag{4.27}$$

and then we apply the first control input, K_1^*, until the first optimal switching time τ_1^* to the system (4.17).

The state resulting from the VTC optimal control problem at the end of the horizon length can be defined by function g, similar to (4.25), as follows

$$\begin{aligned}&g_{\text{VTC}}(\tau_0, \tau_1^*, \ldots, \tau_N^*, \tau_{N+1}, K_1^*A, K_2^*, \ldots, K_{N+1}^*) \\ &= \; x(\tau_{N+1}) \\ &= \; x(\tau_N^*) + (\tau_{N+1} - \tau_N^*)(ax(\tau_N^*) + bK_{N+1}^*)\end{aligned} \tag{4.28}$$

where τ_0, τ_{N+1}, and $x(\tau_0)$ are known quantities representing the initial time, the horizon length ($\tau_{N+1} = H$), and the initial state respectively.

The only difference between g_{RHC} and g_{VTC} is that the latter employs optimal step times $\tau_1^*, \tau_2^*, \ldots, \tau_N^*$ whereas the former has predefined values for the time sequence

$t_1, t_2, \ldots, t_N$. If $[\tau_1^*, \tau_2^*, \ldots, \tau_N^*] = [t_1, t_2, \ldots, t_N]$ then

$$g_{\text{VTC}}(\tau_0, \tau_1^*, \ldots, \tau_N^*, \tau_{N+1}, K_1^*, K_2^*, \ldots, K_{N+1}^*) = g_{\text{RHC}}(K_1^*, K_2^*, \ldots, K_{N+1}^*). \quad (4.29)$$

Thus the general relationship

$$g_{\text{VTC}} \leq g_{\text{RHC}} \quad (4.30)$$

holds and only when the RHC sampling time points are optimal, i.e. $[t_1, t_2, \ldots, t_N] = [\tau_1^*, \tau_2^*, \ldots, \tau_N^*]$, will $g_{\text{VTC}} = g_{\text{RHC}}$. From the relationship in (4.30) it directly follows that

$$\mathcal{J}_{\text{VTC}} \leq \mathcal{J}_{\text{RHC}}. \quad (4.31)$$

General case

Now we use the same approach to prove (4.31) for the system given in (4.18).

Let the control input be defined as a sequence of piecewise constant inputs $\mathbf{U} = [K_1, K_2, \ldots, K_{N+1}]$. Then the system in (4.18) can be defined as

$$x' = f(x, \mathbf{U}) \quad (4.32)$$

for which the solution at time point t_{next} can be approximated using first-order approximation as in [2]

$$x(t_{\text{next}}) = x(t_{\text{prev}}) + hf(x(t_{\text{prev}}), K_i) \quad (4.33)$$

where h defines a fixed step-size $t_{\text{next}} - t_{\text{prev}}$ for the forward step.

This can be applied to construct a chain of approximations that define the solution to (4.18) using the piecewise constant control input over N steps

$$\begin{aligned} x(t_1) &= x(t_0) + hf(x(t_0), K_1) \\ x(t_2) &= x(t_1) + hf(x(t_1), K_2) \\ &\vdots \\ x(t_N) &= x(t_{N-1}) + hf(x(t_{N-1}), K_N) \\ x(t_{N+1}) &= x(t_N) + hf(x(t_N), K_{N+1}) \end{aligned} \tag{4.34}$$

where $t_{N+1} = t_{\text{end}}$ and $h = t_{i+1} - t_i$ for $i = 1, \ldots, N$.

Again, we concentrate on x in the quadratic cost functions defined in (4.22) and (4.27), and construct a function g over the entire horizon length for RHC

$$\begin{aligned} g_{\text{RHC}}(K_1^*, K_2^*, \ldots, K_{N+1}^*) &= x(t_{N+1}) \\ &= x(t_N) + (t_{N+1} - t_N) f(x(t_N), K_{N+1}^*) \end{aligned} \tag{4.35}$$

and VTC

$$\begin{aligned} g_{\text{VTC}}(\tau_0, \tau_1^*, \ldots, \tau_N^*, \tau_{N+1}, K_1^*, K_2^*, \ldots, K_{N+1}^*) &= x(\tau_{N+1}) \\ &= x(\tau_N^*) + (\tau_{N+1} - \tau_N^*) f(x(\tau_N^*), K_{N+1}^*). \end{aligned} \tag{4.36}$$

Under the assumption of convexity for existence of solution of g_{RHC} and g_{VTC}, the two functions are equivalent when $[\tau_1^*, \tau_2^*, \ldots, \tau_N^*] = [t_1, t_2, \ldots, t_N]$ and the relationship $g_{\text{VTC}} \leq g_{\text{RHC}}$ still holds. Only when the RHC sampling time points are optimal, i.e. $[t_1, t_2, \ldots, t_N] = [\tau_1^*, \tau_2^*, \ldots, \tau_N^*]$, will $g_{\text{VTC}} = g_{\text{RHC}}$. From this, it directly follows that $\mathcal{J}_{\text{VTC}} \leq \mathcal{J}_{\text{RHC}}$.

4.3.2 Stability

The standard tool for establishing stability of receding horizon controllers is the second theory of Lyapunov, which is especially well suited to nonlinear systems. In the RHC stability theory, the cost function over the horizon length H is used as a Lyapunov function candidate [14]. There are two ways that the cost function has been shown to satisfy the Lyapunov function requirements. One is to show the monotonicity of $\mathcal{J}_H$ directly, which has been discussed in [23, 46] and for nonlinear systems in [7, 24] and [26], amongst others. The other approach is to prove the monotonicity property for a *sequence* of cost functions $[\mathcal{J}_H(x_{\mathrm{k}}), \mathcal{J}_H(x_{\mathrm{k+1}}), \mathcal{J}_H(x_{\mathrm{k+2}}), \ldots]$. This approach has been used in [29, 30] and [28]. We will use the second approach in our stability argument.

The RHC stability proof using Lyapunov's second method requires us to show that the cost function J_H is a Lyapunov function, thus the following must hold:

$$\mathcal{J}_H(\mathbf{x}_k, \mathbf{u}_k) - \mathcal{J}_H(\mathbf{x}_{k+1}, \mathbf{u}_{k+1}) > 0 \text{ for } \mathbf{x} \neq 0 \tag{4.37}$$

and $\mathbf{x}_k, \mathbf{u}_k$ are the state and control trajectory, respectively, resulting from the RHC control policy $\mathbf{u}_k = \hat{\mathbf{u}}_H(\mathbf{x}_k)$.

It is important to note that the cost function may not decrease at each time instant. Thus the local cost functions between each switching point,$\int_{\tau_i^*}^{\tau_{i+1}^*} l(\cdot,\cdot)\, dt$, do not qualify as Lyapunov function candidates as the cost may not be monotonic, it may fluctuate between switching points. However, the total cost over the horizon H will be shown here to decrease with each shift of $\triangle\tau_i$ as required by Lyapunov's method.

While our stability proof will follow along the lines of the second method of Lyapunov, the main difference between our stability proof and the ones presented in RHC literature is the fact that RHC method results in a periodic sampled system whereas the variable control results in an optimally sampled system. The outline of the proof begins with some definitions and the necessary assumptions. We will use scalar variables to simplify the notation.

We want to show that for some horizon length $H \geq H^*$ the cost function over that horizon decreases from one switching point to the next. This can be expressed as

$$\mathcal{J}_H(x(\tau_i^*), \mathcal{S}_N) - \mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N) > 0 \tag{4.38}$$

where $\triangle\tau_i^*$ is the difference between the variable time switches ($\triangle\tau_i^* = \tau_{i+1}^* - \tau_i^*$) and $\triangle\tau_i^* \geq \tau_{\text{dwell}} > 0$ (minimum dwell time). Figure 4.3 shows the different cost functions required for the proof. The cost $\mathcal{J}_H(x(\tau_i^*), \mathcal{S}_N)$ specifies the cost over the horizon length H starting at the time point τ_i^* using the optimal trace $\mathcal{S}_N$ computed from the optimization problem (4.5). The cost $\mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N)$ is similarly defined but this time, the cost has been shifted forward by one time step $\triangle\tau_i^*$. Finally, $\mathcal{J}_{H-\triangle\tau_i^*}$ defines a cost function computed over a horizon length shortened by one time step (i.e. $H - \triangle\tau_i^*$) and the integral $\int_{\tau_i^*}^{\tau_i^*+\triangle\tau_i^*} l(x(\tau_i^*), u_i^*)\, dt$ measures the cost for one step only.

The first cost function term in (4.38) can be rewritten as a sum of the cost for the first step up to $\tau_i^* + \triangle\tau_i^*$ and the remainder of the cost for the horizon length minus

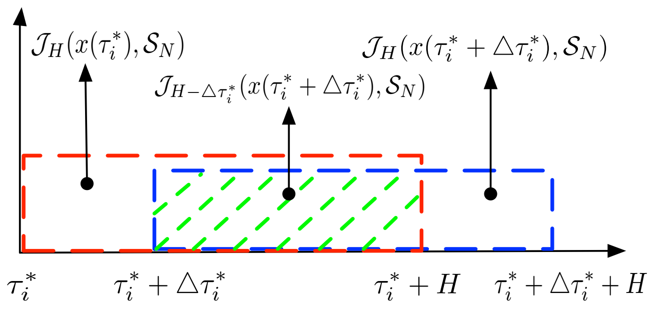
$\mathcal{J}_H(x(\tau_i^*), \mathcal{S}_N)$
$\mathcal{J}_{H-\triangle\tau_i^*}(x(\tau_i^*+\triangle\tau_i^*), \mathcal{S}_N)$
$\mathcal{J}_H(x(\tau_i^*+\triangle\tau_i^*), \mathcal{S}_N)$
τ_i^*
$\tau_i^*+\triangle\tau_i^*$
τ_i^*+H
$\tau_i^*+\triangle\tau_i^*+H$

Figure 4.3: Cost functions over H shifted by $\triangle\tau_i^*$

that first step $(H-\triangle\tau_i^*)$

$$\mathcal{J}_H(x(\tau_i^*), \mathcal{S}_N) = \int_{\tau_i^*}^{\tau_i^*+\triangle\tau_i^*} l(x(\tau_i^*), u_i^*)\, dt + \mathcal{J}_{H-\triangle\tau_i^*}(x(\tau_i^*+\triangle\tau_i^*), \mathcal{S}_N) \tag{4.39}$$

with $\mathcal{J}_{H-\triangle\tau_i^*}(x(\tau_i^*+\triangle\tau_i^*), \mathcal{S}_N) = \sum_{i=1}^{N}\int_{\tau_i^*}^{\tau_{i+1}^*} l(x(\tau_i^*, u_i^*))\, dt$ and the second cost function in (4.38) is as (4.3) but shifted by the length of one time step $\triangle\tau_i^*$.

Thus, indirectly we must show that the following holds

$$\int_{\tau_i^*}^{\tau_i^*+\triangle\tau_i^*} l(x(\tau_i^*), u_i^*)\, dt > \mathcal{J}_H(x(\tau_i^*+\triangle\tau_i^*), \mathcal{S}_N) - \mathcal{J}_{H-\triangle\tau_i^*}(x(\tau_i^*+\triangle\tau_i^*), \mathcal{S}_N) \tag{4.40}$$

for the inequality in (4.38) to hold.

In order to proceed with the proof we must first define two ratios that will help us quantify the behaviour of the cost function when the prediction horizon is increased and also the cost of the first step with respect to the cost over the remaining horizon.

Definition 4.3.1. By definition the Lyapunov function is positive definite; therefore, the cost function will grow if we increase the horizon length by an additional time step $\triangle\tau_i^*$. Let us define this increase in cost by α_H, where

$$\alpha_H = \max(\frac{\mathcal{J}_{H+\triangle\tau_i^*}}{\mathcal{J}_H}).$$

Then α_H represents the maximum ratio by which the cost function will increase if we increase the prediction horizon by $\triangle\tau_i^* > 0$. If $\alpha_H < 1$ then the cost function is monotonic and the stability argument follows automatically. Otherwise α_H serves as a measure of the cost function increase. We also propose the immediate property

$$\lim_{H\to\infty} \alpha_H = 1.$$

Definition 4.3.2. Let β_H represent the *lower* bound of the ratio of the cost function over the first switching point compared to the cost over the entire prediction horizon, then

$$\int_{\tau_i^*}^{\tau_i^*+\triangle\tau_i^*} l(x(\tau_i^*), u_i^*)\, dt \geq \beta_H \mathcal{J}_H.$$

Depending on the weighting parameters R and Q in $l(\cdot,\cdot)$, $0 < \beta_H \leq 1$.

Finally, let us formally propose the finite horizon cost function as a candidate Lyapunov function.

Definition 4.3.3. Given the control sequence $\mathcal{S}_N$ where the time sequence is strictly increasing ($t_{\text{init}} < \tau_1^* < \tau_2^* < \ldots < t_{\text{end}} = H$), the candidate Lyapunov function is the cost function $\mathcal{J}_H(x(\tau_i^*), \mathcal{S}_N)$.

We can now proceed with the final theorem.

Theorem 4.3.4. *Let the horizon length H be such that*

$$\gamma_H = \alpha_{H-\triangle\tau_i^*}(1-\beta_H) < 1$$

then the control sequence $\mathcal{S}_N$ is stabilizing for (4.1) *and $\mathcal{J}_H$ is a Lyapunov-like function for the variable transition system with*

$$\mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N) \leq \gamma_H \mathcal{J}_H(x(\tau_i^*), \mathcal{S}_N).$$

Proof of 4.3.4. Using the notation in (4.40), (4.38) can be rewritten such that

$$\begin{aligned}
&\mathcal{J}_H(x(\tau_i^*), \mathcal{S}_N) - \mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N) \qquad (4.41)\\
&\quad = \int_{\tau_i^*}^{\tau_i^*+\triangle\tau_i^*} l(x(\tau_i^*), u_i^*)\, dt \\
&\quad +\mathcal{J}_{H-\triangle\tau_i^*}(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N) - \mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N) \\
&\geq \int_{\tau_i^*}^{\tau_i^*+\triangle\tau_i^*} l(x(\tau_i^*), u_i^*)\, dt + \frac{1}{\alpha_{H-\triangle\tau_i^*}} \mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N) \\
&\quad -\mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N) \\
&\geq \beta_H \mathcal{J}_H(x(\tau_i^*), \mathcal{S}_N) + \frac{1}{\alpha_{H-\triangle\tau_i^*}} \mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N) \\
&\quad -\mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N)
\end{aligned}$$

Rearranging the terms in the above equation, we get:

$$\alpha_{H-\triangle\tau_i^*}(1-\beta_H)\mathcal{J}_H(x(\tau_i^*), \mathcal{S}_N) \geq \mathcal{J}_H(x(\tau_i^* + \triangle\tau_i^*), \mathcal{S}_N) \qquad (4.42)$$

which finalizes the proof. □

4.3.3 Performance

We can construct the performance bounds for our controller, $\mathcal{J}_\infty(x, \mathcal{S}_N) \leq P_H \mathcal{J}_H(x)$ where $\mathcal{J}_\infty(x, \mathcal{S}_N)$ is defined as the cost of running our controller over the infinite horizon and $\mathcal{J}_H(x)$ is the finite horizon cost of the optimal controller. We continue to extend the arguments presented in [28] and [29] to our variable switching time predictive controller.

Theorem 4.3.5. *Let H, γ_H, α_H be defined as in 4.3.4. The control sequence S_N results in a stabilizing control law $u_H^*(x)$ with a corresponding infinite horizon cost function defined by*

$$\mathcal{J}_\infty(x, \mathcal{S}_N) = \int_0^\infty l(x(t), u_H^*(x(t)))dT. \tag{4.43}$$

Then the bound for infinite-horizon performance is given by

$$\mathcal{J}_\infty(x, \mathcal{S}_N) \leq P_H \mathcal{J}_H(x) \tag{4.44}$$

where

$$P_H = (1 + (\frac{\max(0, \alpha_{H-\triangle\tau_1^*} - 1)}{\alpha_{H-\triangle\tau_1^*}})\frac{\gamma_H}{1-\gamma_H}) \tag{4.45}$$

where γ_H is defined Theorem 4.3.4.

Proof of 4.3.5. Let the control policy $u(t) = u_H^*(x(t))$ and $\mathcal{J}_H(x) = \mathcal{J}_H(x, \mathcal{S}_N)$. We continue to use the argument presented in [28] and [29] to provide bounds on the performance of the controller u_H^*.

Using equation (4.40), the above relationship is shown step-by-step:

$$\begin{aligned}\int_{\tau_0^*}^{\tau_1^*} l(x(0),u)\;dt &= \mathcal{J}_H(x(0)) - \mathcal{J}_{H-\triangle\tau_i^*}(x(\tau_1^*)) \\ &= \mathcal{J}_H(x(0)) - \mathcal{J}_H(x(\tau_1^*)) \\ &\quad +\mathcal{J}_H(x(\tau_1^*)) - \mathcal{J}_{H-\triangle\tau_1^*}(x(\tau_1^*)) \\ &\leq \mathcal{J}_H(x(0)) - \mathcal{J}_H(x(\tau_1^*)) \\ &\quad +\mathcal{J}_H(x(\tau_1^*)) - (\frac{1}{\alpha_{H-\triangle\tau_1^*}})\mathcal{J}_H(x(\tau_1^*)) \\ &\leq \mathcal{J}_H(x(0)) - \mathcal{J}_H(x(\tau_1^*)) \\ &\quad +(\frac{\alpha_{H-\triangle\tau_1^*}-1}{\alpha_{H-\triangle\tau_1^*}})\mathcal{J}_H(x(\tau_1^*))\end{aligned} \tag{4.46}$$

similarly:

$$\begin{aligned}&\int_{\tau_1^*}^{\tau_2^*} l(x(\tau_1^*),u)\;dt \leq \\ &\quad \mathcal{J}_H(x(\tau_1^*)) - \mathcal{J}_H(x(\tau_2^*)) + (\frac{\alpha_{H-\triangle\tau_2^*}-1}{\alpha_{H-\triangle\tau_2^*}})\mathcal{J}_H(x(\tau_2^*))\end{aligned} \tag{4.47}$$

then, by summation:

$$\begin{aligned}\int_0^\infty l(x,u)\;dt &\leq (\frac{\alpha_{H-\triangle\tau_1^*}-1}{\alpha_{H-\triangle\tau_1^*}})\sum_i^\infty \mathcal{J}_H(x(\tau_i^*)) \\ &\leq \mathcal{J}_H(x(0)) + (\frac{\max(0,\alpha_{H-\triangle\tau_1^*}-1)}{\alpha_{H-\triangle\tau_1^*}})\sum_i^\infty \mathcal{J}_H(x(\tau_i^*)) \\ &\leq (1+(\frac{\max(0,\alpha_{H-\triangle\tau_1^*}-1)}{\alpha_{H-\triangle\tau_1^*}})\sum_k^\infty \gamma_H^k)\mathcal{J}_H(x(0)) \\ &\quad (by\ Theorem\ 4.3.4) \\ &\leq (1+(\frac{\max(0,\alpha_{H-\triangle\tau_1^*}-1)}{\alpha_{H-\triangle\tau_1^*}})(\frac{\gamma_H}{1-\gamma_H}))\mathcal{J}_H(x(0))\end{aligned} \tag{4.48}$$

which finalizes the proof. $\square$

Chapter 5

Implementation and Evaluation

The previous chapter defined the variable transition control problem and presented a theoretical basis for this control algorithm. However, the variable switching times pose an interesting implementation challenge since the solution methods used in predictive control theory can no longer be applied directly. The computation of the approximation chain developed in (4.15) has now become an initial step of the control loop. The control algorithm is the main module of implementation, shown in Figure 5.1.

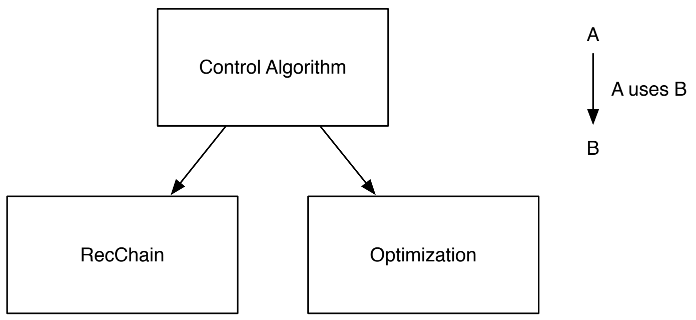

Figure 5.1: Main algorithms

It serves as a sequencing module that calls the procedure that builds the recursive chain and then evaluates that expression for the current solution step and passes the result to the algorithm that performs the optimization. The result obtained from the optimization procedure is then used to update the control loop with the size of step and the control input to be used during that step. The main loop runs until the step size reaches the simulation end time.

The next section describes the main steps of the implementation process followed by a set of examples that emphasize the main features of the VTC algorithm and provide a comparison with the fixed step predictive control algorithm.

5.1 Implementation

The implementation of the variable transition controller framework is a hybrid of symbolic formulation and numeric computation tools. The recursive model approximation formula is constructed symbolically using a computer algebra system, for example Maple [47]. This allows us to construct an expression for the solution of the state with the switching times present as symbolic parameters. The recursive procedure for symbolically computing an n-step approximation is outlined below.

```
1: procedure RecChain($n$)
2:     if $n > 0$ then
3:         $x_{n+1} = \textsc{RecChain}(n-1) + (\tau_{n+1} - \tau_n)\, f(\textsc{RecChain}(n-1), u_n)$
4:     else if $n = 0$ then
5:         $x_{n+1} = x_0 + (\tau_{n+1} - \tau_n) f(x_0, u_1)$
6:     else
7:         error $n$ must be a nonnegative integer
```

```
 8:         end if
 9:         return $x_{n+1}$
10: end procedure
```

The resulting symbolic approximation expression with symbolic parameters for $\tau_1, \tau_2, \ldots, \tau_n$ is then used to construct the cost function. The solution to the cost function minimization problem can be approached in two ways. The simplest approach is to remain within a computer algebra framework and use its numerical optimization methods to solve the control problem $P_{\mathbf{U},\mathbf{T}}$. Here we used a sequential quadratic programming (SQP) method to obtain the solution within a computer algebra system Maple. Maple's SQP solver is an interface to NAG's solver `E04UCA` [48]. The SQP problem is defined as follows:

$$\min_{x \in R^n} f(x) \text{ s.t. } g_{\mathrm{L}} \leq \begin{pmatrix} x \\ A_{\mathrm{L}} x \\ c(x) \end{pmatrix} \leq g_{\mathrm{U}} \tag{5.1}$$

where $f(x)$ represents a nonlinear objective function, A_{L} is an n_{L} by n constant matrix, and $c(x)$ is an n_{N} element vector of nonlinear constraint functions. The g_{L} and g_{U} are the lower and upper bounds, respectively. The advantage of using this approach was that gradient of the objective function, $\bigtriangledown f(x)$, the Jacobian of the constraints $\bigtriangledown g(x)$, and any other information required by the SQP solver, was computed automatically by the symbolic tool.

Finally, the control loop was then implemented and simulated in the computer algebra system, which also provided us with an environment to visualize the simulation results. The main steps of the control loop are outlined in algorithm below.

```
1: procedure VTControl($N, x_{\text{init}}, t_i, H, t_{\text{end}}$)
2:     $J = \int_{t_i}^{t_i+H} l(\text{RecChain}(N))$
3:     while $t_i < t_{\text{end}}$ do
4:         evaluate $J$ at $\{\tau_0 = t_i, \tau_{N+1} = t_i + H, x_0 = x_{\text{init}}\}$
5:         $\mathbf{U}, \mathbf{T} = \texttt{Minimize}(J)$
6:         $x_{\text{new}} = x_{\text{init}} + \mathbf{T}[1] f(x_{\text{init}}, \mathbf{U}[1])$
7:         $x_{\text{init}} = x_{\text{new}}$
8:         $t_i = \mathbf{T}[1]$
9:     end while
10: end procedure
```

The disadvantages of the above approach concern the issues of portability and scalability of the computation. The control loop is executed within a symbolic tool, which uses an interpreted language. This not only makes the computation slower but also prevents the control algorithm from being executed on platforms without a computer algebra tool. Furthermore, although the focus of the current implementation is as a *proof-of-concept* for the control algorithm, in the future the computational slowdown might prevent the method from being scalable to larger models. We thus present an alternate implementation approach that overcomes these limitations.

The second approach uses optimization solvers external to the symbolic tool. Using a symbolic cost function expression computed through the recursive approximation chain, we generate the C code for the cost function and all the other necessary information into the target language of an independent optimization solver. This approach was motivated in part as a way to scale up the size of the simulation models in the future but it was also used for comparison with the SQP method. We have used the interior point optimization package called IPOPT [49] as our external solver. The use of interior point methods in predictive control is discussed by the developers of IPOPT in [50].

The IPOPT problem is defined as follows:

$$\begin{aligned} &\min_{x \in R^n} f(x) \\ \text{s.t. } & g_{\mathrm{L}} \leq g(x) \leq g_{\mathrm{U}} \\ & x_{\mathrm{L}} \leq x \leq x_{\mathrm{U}} \end{aligned} \tag{5.2}$$

where $f(x) : R^n \rightarrow R$ is the objective function, and $g(x) : R^n \rightarrow R^m$ are the constraint functions. The vectors g_{L} and g_{U} denote the lower and upper bounds on the constraints, and the vectors x_{L} and x_{U} are the bounds on the variables x. The functions $f(x)$ and $g(x)$ can be nonlinear and non-convex.

The IPOPT solver requires some supplementary information in order to be able to solve the optimization problem. The user must provide the details about the number of variables and constraints, variable bounds, and a starting point in addition to:

- the objective function, $f(x)$
- the gradient of the objective, $\bigtriangledown f(x)$
- the constraint function values, $g(x)$
- the Jacobian of the constraints, $\bigtriangledown g(x)^T$

and other information concerning the structure of the problem such as the number of non-zeroes in the constraint Jacobian. The four items in the above list were all code generated from the symbolic tool into the C language.

The advantage of code generating the problem into C is that we can then execute the control loop in compiled code and deploy the controller on many platforms. This makes our algorithm more portable and scalable with less concern for computational efficiency. We were also able to verify the VTC solution obtained using the SQP solver with the solution obtained using IPOPT.

5.1.1 Real-time predictive control

Another concern is the real-time implementation of the controller. Real-time RHC is an open and complex area of research. The cause for the complexity is the fact that a quadratic programming problem has to be solved on-line at each step. This can be a slow process especially for large problems with many optimization variables (due to a large horizon). The current RHC research suggests two implementation approaches for real-time control: embedding a numerical solver in the real-time control code or moving parts of the computation off-line. For example, in [51], the authors propose an off-line computation of a piecewise affine approximation of the optimal solution that can be used to warm-start an on-line computation, although the problems are limited to linear systems. A good summary of additional approaches to real-time RHC is given in [52].

We have moved our algorithm a step closer to real-time implementation by generating the problem definition into C, which in turn allows us to execute our algorithm in compiled code. Furthermore, we also constrain the adaptive step size with minimum dwell time, τ_{dwell}, which can ensure that we do not have to solve the optimization problem too often. Taking larger steps when possible gives us the advantage of reducing the number of model evaluations within a control loop. However, more research will be necessary in the future in order to investigate how VTC can meet the real-time requirements.

5.2 Evaluation Examples

We present a number of examples that we use to show specific aspects of the VTC algorithm. The first example of a pendulum is used to show the general effect of the algorithm on the control system and discuss the basic features of VTC and how they affect the control problem. The second example shows the two optimization implementation approaches used in the thesis: one within the symbolic framework that uses the SQP algorithm for optimization and the other which uses code generation into C and then links the C code with IPOPT solver for optimization. Finally, we present two examples that show some of the possible advantages of using the variable step size controller over the standard RHC algorithm. A chemical reactor model, also employed in [53], is used to show that variable step size can help us reduce the number of model evaluations. The second example of a damped oscillator is used to show that VTC can decrease the amount of manual tuning required to control a system by automatically reducing the step size, when necessary, to achieve the desired response. For both of these examples, we first executed our control algorithm with the variable step, the VTC mode, and then with a fixed step-size to emulate the RHC algorithm.

5.2.1 Inverted Pendulum

The inverted pendulum model shown in Figure 5.2 can be defined by simplified equations as follows:

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= -9.78x_1 - 0.1K. \end{aligned} \tag{5.3}$$

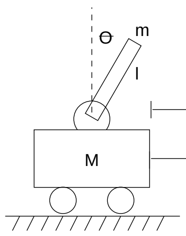

Figure 5.2: Inverted pendulum on a cart

Using Euler approximation formula as given in (4.13), the model in (5.3) is expressed as:

$$\begin{aligned} x_1(k+1) &= x_1(k) + \tau x_1(k) \\ x_2(k+1) &= x_2(k) + \tau(-9.78x_1(k) - 0.1K) \end{aligned} \tag{5.4}$$

The cost function is given by

$$J = \int_0^H x_1^2 \tag{5.5}$$

since we are only interested in the angle of the pendulum. We could consider the rate of change of the angle, in which case the cost function would be

$$J = \int_0^H x_2^2. \tag{5.6}$$

We simulated the inverted pendulum model in the computer algebra software Maple. The simulation used $x_1(0) = 0.02$, $x_2(0) = 0.0$ as initial conditions and the horizon length $H = 0.05$ over the simulation interval of $t_{\text{end}} = 2.0$. In Maple, we constructed a parameterized approximation chain for solution of the model in (5.3) and used a sequential quadratic programming method to solve for the decision variables in the optimization problem.

Figure 5.3 shows the simulation result for the state and the control input of a VTC algorithm with three switching points. The state variable has been scaled up for improved viewing of the combined plot. This plot demonstrates the predictive nature of the controller. The controller switches the gain well before the state crosses over and goes below the set-point at $x_1(t) = 0$ and then it switches again before the state trajectory goes above that point.

In Figure 5.4 we show the effect of increasing the number of switching points on the state of the system. We are able to steer the system toward the steady-state faster as we increase the number of switching points of the controller.

However, increasing the number of switching points also increases the number of decision variables for the optimization problem. Thus, both VTC and RHC algorithms favor a longer sampling frequency with fewer decision variables, if possible, to avoid the increased computational cost.

Finally, Figure 5.5 shows the cost function for a controller with three switching

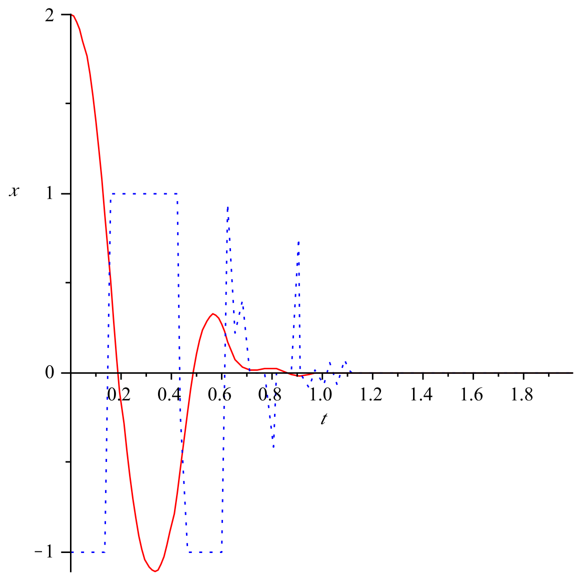

Figure 5.3: Pendulum state and control input

points. We can see that the cost function is not monotonically decreasing; however, the cost function over the set horizon length ($\mathcal{J}_H$) is shown to decrease over time.

This shows why it was important that the stability proof does not argue that the cost function at each step is a Lyapunov function candidate. Rather the proof uses the cost function over the entire horizon length to argue monotonicity of a sequence of cost functions.

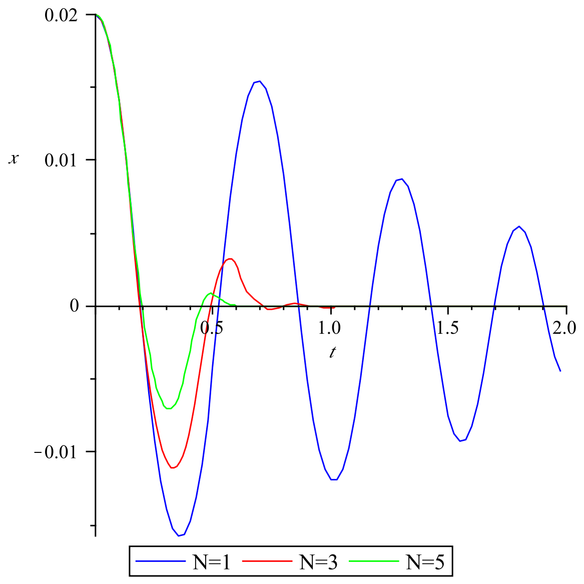

Figure 5.4: Pendulum state with increasing N

5.2.2 Double Integrator

In this example we will apply the proposed control strategy to a double integrator system defined by

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= u \end{aligned} \tag{5.7}$$

in order to discuss the two solution methods used in the thesis: the SQP method within the symbolic tool and the IPOPT method with a control loop implemented in the C language. We do not impose any constraints on the system and we use a controller with one switching point in order to simplify the comparison.

Both implementations begin with generation of the approximation chain built by

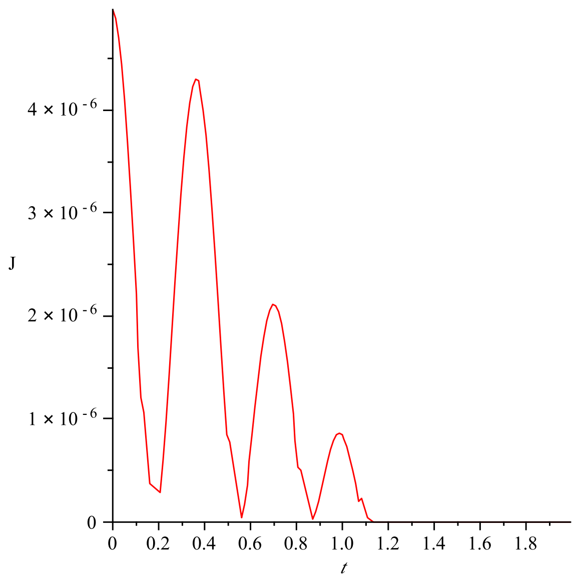

Figure 5.5: Cost function for N=3

a recursive routine. Since we are interested in computing one switching point and thus two control inputs $u = \{K_1, K_2\}$, as shown in Figure 5.6, we need a two step

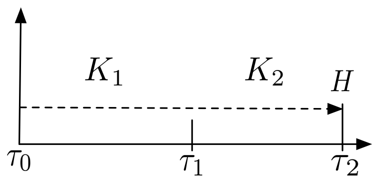

Figure 5.6: Problem with one switching point τ_1

approximation

$$\begin{aligned}
x_1(\tau_1) &= x_1(\tau_0) + (\tau_1 - \tau_0)x_2(\tau_0) \\
x_2(\tau_1) &= x_2(\tau_0) + (\tau_1 - \tau_0)K_1 \\
x_1(\tau_2) &= x_1(\tau_1) + (\tau_2 - \tau_1)x_2(\tau_1) \\
x_2(\tau_2) &= x_2(\tau_1) + (\tau_2 - \tau_1)K_2.
\end{aligned} \tag{5.8}$$

The approximation in (5.8) is then used to construct an expression for the cost function where the decision variables for the control inputs and switching points have been left as symbolic parameters of the expression. The optimization problem

$$\min_{\tau_1, K_1, K_2} \mathcal{J}_H \tag{5.9}$$

is then solved for τ_1, K_1, and K_2 and the results are used in the control loop to update the control problem and obtain new measurements for the state of the system in (5.7).

The simulation result for the control loop using the SQP solver is shown in Figure 5.7. The simulation was performed with the initial conditions set to $\mathbf{x}_0 = [0.2, 0.2]^{\mathrm{T}}$. We set the weights in the cost function (4.3) to $Q = I, R = I$.

The use of the IPOPT solver required us to generate the C code for functions defining the cost function and the gradient of the cost function. (Note that we would also have to generate procedures for the constrains and the constraint Jacobian if this problem had constraint equations.) The cost function C code is defined within the IPOPT function call:

```
Bool eval_f(Index n, Number* x, Bool new_x, Number* obj_value,
   UserDataPtr user_data)
```

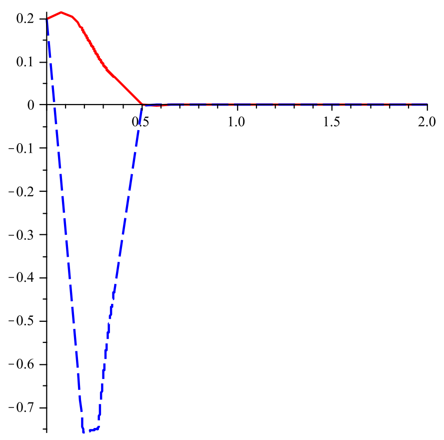

Figure 5.7: Double integrator solved with SQP

and the gradient of the cost function is specified within the function call:

```
Bool eval_grad_f(Index n, Number* x, Bool new_x, Number* grad_f,
   UserDataPtr user_data)
```

which, for this particular model, contains code generated expressions for the three quantities defining $\frac{\partial \mathcal{J}_H}{\partial \tau_1}$, $\frac{\partial \mathcal{J}_H}{\partial K_1}$, and $\frac{\partial \mathcal{J}_H}{\partial K_2}$. The control loop was then implemented in C and the resulting data was plotted back in the tool. Figure 5.8 shows the result of double integrator simulation using the IPOPT solver.

The simulation results for both solvers look consistent since similar settings were used in both. There was a small difference in the minimum horizon length that was required to compute the controller parameters. The minimum horizon length required to solve the problem using SQP was $H = 0.15$ while IPOPT produced similar result with $H = 0.22$. It would not be legitimate to compare the run times of both implementations since we had to generate the gradients and Jacobians for the IPOPT problem setup, whereas the symbolic tool took care of all the necessary

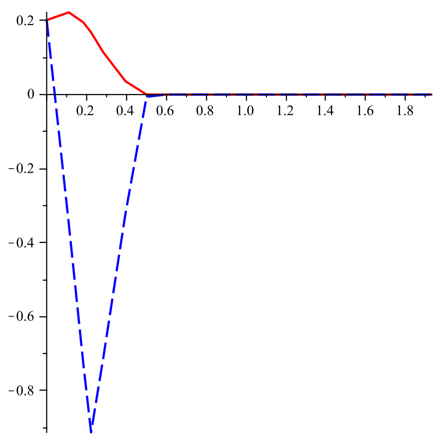

Figure 5.8: Double integrator solved with IPOPT

vector computations. However, it is reasonable to believe that, especially for a large model, the compiled control loop code would be faster than the interpreted code of the symbolic tool.

5.2.3 Chemical Reactor

The petrochemical industry is one of the main application areas of predictive control, [15], thus variations of the chemical reactor example are popular in the RHC literature. The example below was used in [53] to show the advantages of constrained LQR algorithm over the traditional RHC algorithm. Here we demonstrate the results of a simulation for the unconstrained problem and compare the results for the fixed and variable time step approach.

The Van de Vusse reactor model is defined as follows:

$$\begin{aligned} x_1' &= -k_1 x_1 - k_3 x_1^2 - x_1 u \\ x_2' &= k_1 x_1 - k_2 x_2 - x_2 u \end{aligned} \tag{5.10}$$

and we let $u = [K_1, K_2]^T$ be the vector of control inputs. As described in [53], this reactor has constant volume and the system (5.10) models a series of reactions

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C,\ 2A \xrightarrow{k_3} D \tag{5.11}$$

where x_1 and x_2 in (5.10) represent the concentrations of A and B and we set $k_1 = 50, k_2 = 100, k_3 = 10$.

The controllers were implemented using the cost function (4.3) with $Q = I$ and $R = I$ and the initial conditions were set to $\mathbf{x}_0 = [0.5, 0.1]^{\mathrm{T}}$. The horizon length was set to $H = 0.005$ and the number of steps to four (three switching points). The simulation was run to $T = 0.1$, which means that RHC has to take 80 steps (and thus 80 model evaluations) to reach the end time.

The result of simulation using the VTC approach with three switching points is shown in Figure 5.9. The control inputs were scaled up so that they can be viewed easily on the same plot as the state.

Since the system is smooth, the VTC algorithm took a larger first step τ_1^* thus resulting in fewer simulation steps overall. The total number of steps taken by VTC algorithm was 22.

For comparison, the smaller steps of the RHC algorithm are evidenced by the control inputs as shown in Figure 5.10. Again, the control inputs were scaled up to

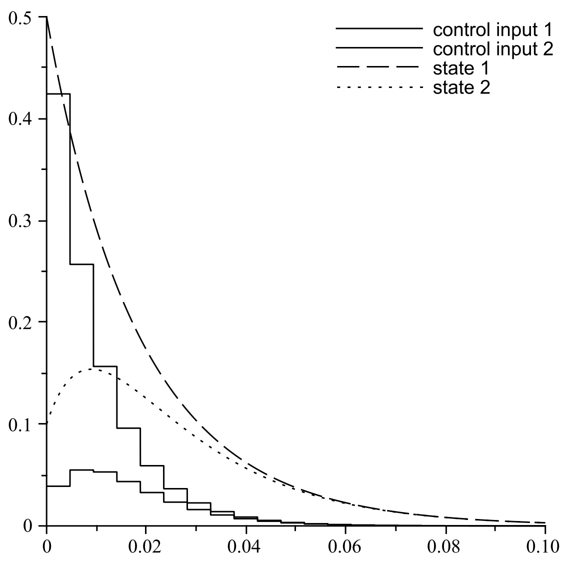

Figure 5.9: Reactor control with variable time switching (VTC)

help visualize the system response.

As the plots show, the responses for both algorithms are almost identical however VTC decreased the number of model evaluations by 72.5%. The number of iterations taken by the SQP solver for the fixed step (RHC) version of the algorithm was 125 and the time spent in the solver was 0.738 seconds of CPU time (on a 2.53GHz Intel Core 2 Duo processor). The VTC algorithm required a total of 36 solver iterations (due to fewer model evaluations being required) and took 0.497 seconds of total solver time.

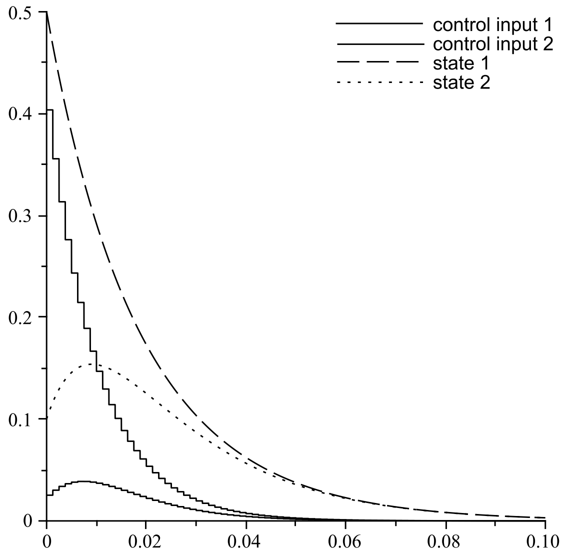

Figure 5.10: Reactor control with fixed time switching (RHC)

5.2.4 Damped Oscillator

The damped harmonic oscillator model is defined by the equation

$$x'' + 2\gamma x' + \omega^2 x = 0 \tag{5.12}$$

where γ and ω are the damping factor and the frequency of the oscillator, respectively. Here we apply the VTC and RHC algorithms to a first order model of the under-damped oscillator:

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= -\frac{1}{2}x_2 - x_1 + u \end{aligned} \tag{5.13}$$

with the initial conditions $\mathbf{x}_0 = [0.2, 0]^{\mathrm{T}}$. If the oscillator receives no control input ($u = 0$) then it will naturally steer down to $x_1 = 0$. We will attempt to control it to $x_1 = 1$ by setting the cost function to

$$\mathcal{J} = \int_{t_0}^{T_{\text{end}}} (x_1 - 1)^2 \, d\tau. \tag{5.14}$$

We purposefully chose a large horizon, $H = 3\pi/2$, with four steps in each horizon length to force the fixed step RHC algorithm to take large steps. As one might expect, this has prevented the controller from being able to steer the oscillator to $x_1 = 1$, as show in Figure 5.11.

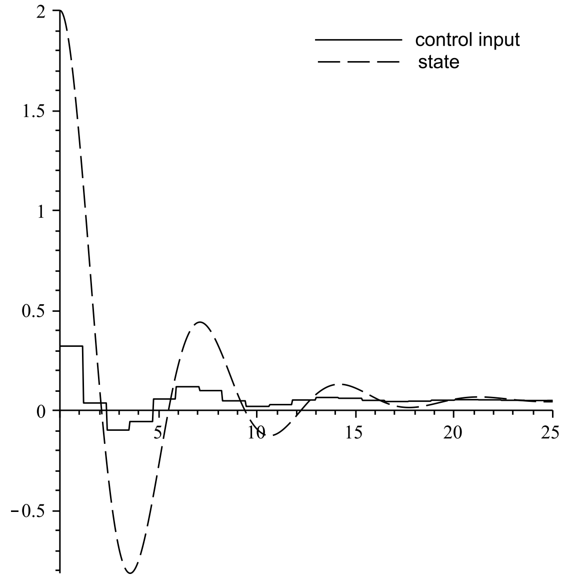

Figure 5.11: Oscillator with fixed time switching (RHC)

However, since VTC computes the optimal switching times, it is able to *adapt* the step size to suit the model and control the oscillator to the desired set point, as shown in Figure 5.12.

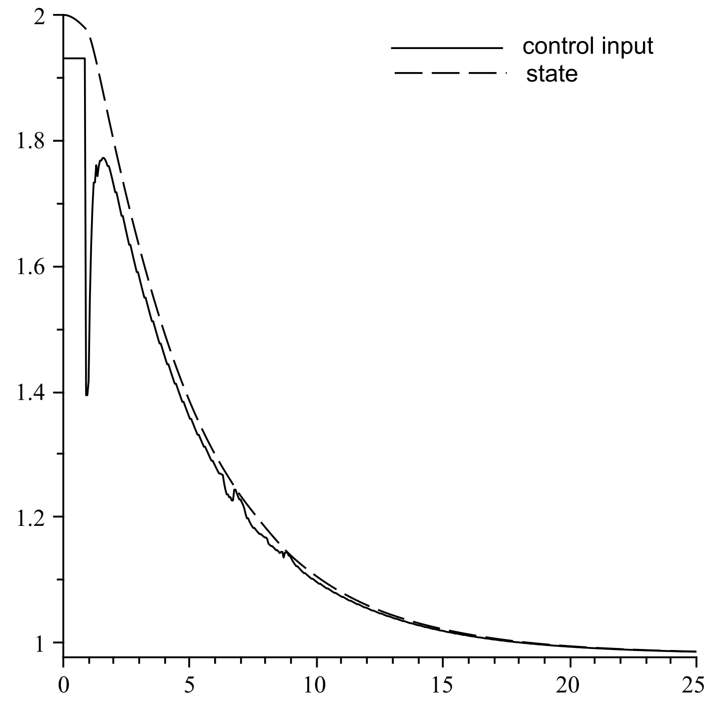

Figure 5.12: Oscillator with variable time switching (VTC)

The above response differences result in a very large difference in the value of the cost function with $\mathcal{J}_{\text{VTC}} = 3.0658$ and $\mathcal{J}_{\text{RHC}} = 25.6568$. The number of solver iterations and simulation times are not reported since the fixed step algorithm was not able to control the system to the desired set point. While RHC could give us the required response by adding more decision variables to the optimization problem, we have avoided this additional tuning by introducing the switching points into the problem formulation.

Chapter 6

Conclusions and Future Work

A summary of the contributions, the main results, and future directions conclude this thesis. Below we present a brief discussion of how the research presented in the thesis contributed to the expansion of predictive control theory and methods. We also make concluding remarks on the results obtained from the implementation of the variable transition control algorithm. We complete the thesis with some suggestions for what the future direction of the research should be in order to develop the VTC algorithm into a mature control method.

6.1 Contributions

The main domains of the contributions of the thesis include:

- **Predictive control theory**

  We presented a form of a predictive controller that expanded the receding horizon control approach to include an optimal sampling time. The traditional RHC methods presented in literature, for example [3, 15, 16], use a fixed sampling time at which a new sequence of optimal control inputs is computed. In the VTC approach, we included the time parameter into the optimization problem. Thus, we compute an optimal control input sequence and a sequence of time points at which the control algorithm should switch between these inputs. The benefit of this design is that it requires less tuning since the controller adapts the step size to the response of the plant, taking smaller steps when there is frequent, large changes and greater steps otherwise.

- **Stability and performance of VTC**

  We have built our control algorithm on a sound theoretical foundation by presenting a stability proof for VTC. We used an approach developed for proving the stability of the RHC algorithm that did not require additional constraints, such as the terminal state or cost constraints, and expanded it to allow for a variable sampling time. We then also provided a bound on the performance of our controller over the infinite horizon. Finally, we showed that in the worst case, our controller performs as well as a traditional model predictive controller.

- **Implementation and the solution of the optimization problem**

  We developed and discussed two implementation methods for the variable transition controller. We first had to develop a different solution method than what is used for traditional predictive controllers with fixed sampling. We had to find a way to add the sequence of time points into the optimization problem. We showed that through first order approximations, we can construct a chain formula that includes both the control inputs and time points. This formula was then used to form an objective function for the optimization problem. We then presented two approaches to solving the resulting optimization problem.

The first approach used the SQP solver within the symbolic framework used to construct the objective function and the second approach used code generation in order to generate functions that could be used by the IPOPT solver. The advantage of the second implementation is that it is more scalable and portable than the symbolic approach.

6.2 Results

The thesis presented examples used in RHC literature for evaluation of the performance of the VTC algorithm. The examples helped us show the following advantages of the VTC approach:

- **Fewer model evaluations**

  At every sampling instance, the simulation model has to be evaluated so that the optimization problem can be updated and a new sequence of control inputs is computed. We showed an example where the model behaviour was smooth over the simulation horizon. Since the RHC algorithm uses a fixed sampling time, we had to solve the optimization problem a predetermined number of times. However, since our control algorithm optimizes the sampling time, we were able to take larger steps and thus significantly decrease the number of times we had to update the model with the new control input sequence. This resulted in a considerable reduction of our computational effort.

- **Finer steps when necessary**

  We used another example to illustrate an opposite advantage of our control algorithm. Again, since we are optimizing the switching times, we showed that the algorithm can take sufficiently small steps where the traditional predictive algorithm fails to control the system to the desired set point due to its fixed step size. Thus, the variable step size structure can benefit us in both smooth and fast changing systems by adapting the sampling frequency.

- **Less tuning**

  The fixed sampling time of the RHC algorithm is often tuned to suit the model. This is not the only parameter used for tuning predictive algorithms. A lot of trail-and-error effort is put toward determining the appropriate horizon length and weighting matrices in a predictive control problem as well. By incorporating the sampling time into the optimization problem, we have eliminated one of the tuning parameters and thus simplified the tuning process of our algorithm.

We presented other examples to highlight or discuss specific aspects of our algorithm or its implementation. For those examples, our algorithm performed on par with the RHC algorithm. Since our approach requires additional variables in the optimization problem, in a case like this, one has to evaluate whether it is more important to reduce the complexity of the optimization problem or the tuning.

6.3 Future Work

Since the research presented here forms a basis of a new control algorithm, there are a number of additional research directions that can be pursued next. We briefly outline a few interesting topics to explore in the future.

6.3.1 Models with constraints

Originally one of the main advantages of the predictive control algorithm was thought to be the ease with which it handled multivariate systems. Later, it became clear that the applications also benefited from the integration of constraints into the optimization problem formulation. The constraints can be placed on the state and control input and can be defined as

$$\mathbf{x}(t) \in \mathcal{X},\ \mathbf{u}(t) \in \mathcal{U} \tag{6.1}$$

where $\mathbf{x}(t) \in \mathcal{X} \subseteq \mathbf{R}^n$ and $\mathbf{u}(t) \in \mathcal{U} \subseteq \mathbf{R}^m$. denote the vector of states and control inputs respectively. $\mathcal{U}$ defines a set of feasible input values and $\mathcal{X}$ defines a set of feasible states. Usually the feasible sets are defined by simple box constraints of the form

$$\begin{aligned}\mathcal{X} &= \{\mathbf{x} \in \mathbf{R}^n \mid \mathbf{x}_{\min} \le \mathbf{x} \le \mathbf{x}_{\max}\} \\ \mathcal{U} &= \{\mathbf{u} \in \mathbf{R}^m \mid \mathbf{u}_{\min} \le \mathbf{u} \le \mathbf{u}_{\max}\}\end{aligned} \tag{6.2}$$

where $\mathbf{x}_{\min}, \mathbf{x}_{\max}, \mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$ are constant vectors specified by the user. However, they can also be defined by a more general region, which is common if the problem is nonlinear. It is also common to add constraints on the derivatives of state and control input.

Our VTC algorithm formulation does not currently include constraints. They could easily be added to the problem formulation and used as input to the optimization solver; however, their inclusion mandates further study of their effect on the stability and performance of the controller. We decided to limit the scope of the theoretical framework in the initial development of the algorithm and instead concentrate on the actual implementation. Thus, the next logical step in furthering the VTC framework would be to add constraints and extend the existing proofs to provide a sound theoretical basis for the constrained problem.

6.3.2 Robustness

Robustness of predictive controllers is a wide research area that studies how well the model predictive control algorithms handle model uncertainty and noise. Our formulation of VTC algorithm assumes that the physical plant and its model used during computation are the same and that there is no unmeasured disturbance acting on the system. This is very common in standard RHC literature. However, there is a large and mostly open area of research that investigates if model predictive control is robust. A control system is robust when the stability of the system is maintained and the performance specifications are met for a specific range of a plant model and uncertainty (noise signal). The unmeasured noise can be added to the system by specifying the model as

$$x' = f(x, u, w) \tag{6.3}$$

where $w(t) \in \mathcal{W}$ and $\mathcal{W}$ is a given set.

A future area of study might be to prove that the stability and performance arguments for VTC still hold in presence of the model and noise uncertainties. However, this is a very large and open area of RHC research. Most studies of robustness involve unconstrained systems only since constraints add another level of complexity [14]. Extending the VTC framework to include uncertainties would be a substantial research project since a wide variety of approaches and noise models could be considered.

6.3.3 Scalability

The thesis concentrated on proving the concept of VTC and thus the examples presented presented in the thesis as *proof of concept* originated in RHC literature that compared RHC to other control methods (such as LQR for example). These examples are small but are good for highlighting a specific feature of an algorithm. In the future, it would be essential to investigate how well the VTC algorithm handles large scale models with many variables. Such models often require constraints so this study should most likely follow incorporation of constraints into the VTC framework.

We have started to create a foundation for large scale models already in the thesis since we discussed an implementation of our algorithm that includes code generation of the model to an external optimization software called IPOPT. In the future, we could code generate the necessary cost functions and gradient vectors and link them with an appropriate solver for faster synthesis.

Bibliography

[1] Vincent TL, Grantham WJ. *Nonlinear and Optimal Control Systems*. John Wiley & Sons: New York, 1997.

[2] Lewis FL. *Optimal Control*. John Wiley & Sons: New York, 1986.

[3] Camacho EF, Bordons C. *Model Predictive Control*. Springer-Verlang: London, 1999.

[4] Garcia CE, Prett DM, Morari M. Model predictive control: Theory and practice - a survey. *Automatica* 1989; **25**(3):335–348.

[5] Richalet J, Rault A, Testud J, Papon J. Model predictive heuristic control: Applications to industrial processes. *Automatica*, vol. 14, 1978; 413–428.

[6] Muske KR, Rawlings JB. Model predictive control with linear models. *AIChE Journal* 1993; **39**(2):262–287.

[7] Mayne DQ, Michalska H. Receding horizon control of nonlinear systems. *IEEE Trans. Auto. Control* 1990; **35**(7):814–824.

[8] Findeisen R, Allgower F. An introduction to nonlinear model predictive control. *Proc. 21st Benelux Meeting on Systems and Control*, Veldhoven, 2002; 1–23.

[9] Kowalska K, v Mohrenschildt M. A framework for the development of variable transition controllers. *The 8th World Multi-Conference on Systematics, Cybernetics and Informatics*, vol. 8, 2004; 235–240.

[10] Astrom KJ, Hagglund T. *PID Controllers: Theory, Design, and Tuning*. ISA - Instrument Society of America: Research Triangle Park, NC 27709, 1995.

[11] Levine W ((ed.)). *The Control Handbook*. CRC Press: Boca Raton, 1996.

[12] Garcia C, Morari M. Internal model control. 1. A unifying review and some new results. *Ind. Eng. Chem. Process Des. Dev.*, vol. 21, 1982; 308–323.

[13] Ljung L ((ed.)). *System Identification: Theory for the User*. Prentice Hall: Englewood Cliffs, NJ, 1987.

[14] Mayne DQ, Rawlings JB, Rao CV, Scokaert POM. Constrained model predictive control: Stability and optimality. *Automatica*, vol. 36, 2000; 789–814.

[15] Maciejowski JM. *Predictive Control with Constraints*. Prentice Hall: England, 2002.

[16] Morari M, Lee J. Model predictive control: past, present and future. *Computer and Chemical Engineering*, vol. 23, 1999; 667–682.

[17] Qin SJ, Badgwell TA. A survey of industrial model predictive control technology. *Control Engineering Practice*, vol. 11, 2003; 733–764.

[18] Cutler C, Ramaker B. Dynamic matrix control - a computer control algorithm. *Automatic Control Conference, San Francisco*, WP5-B, 1980.

[19] Clarke D, Mohtadi C, Tuffs P. Generalized predictive control - Part I. The basic algorithm. *Automatica* 1987; **23**(2):137–148.

[20] Clarke D, Mohtadi C, Tuffs P. Generalized predictive control - Part II. Extensions and interpretations. *Automatica* 1987; **23**(2):149–160.

[21] Kalman RE, Bertram JE. Control system analysis and design via the second method of Lyapunov: I Continuous-time systems. *Trans. ASME* 1960; **82**(2):371–393.

[22] Kalman RE, Bertram JE. Control system analysis and design via the second method of Lyapunov: II Discrete-time systems. *Trans. ASME* 1960; **82**(2):394–399.

[23] Keerthi SS, Gilbert EG. Optimal infinite-horizon feedback laws for general class of constrained discrete-time systems: Stability and moving-horizon approximations. *Journal of Optimization Theory and Applications* 1988; **57**(2):265–293.

[24] Michalska H, Mayne DQ. Robust receding horizon control of constrained nonlinear systems. *IEEE Trans. Auto. Control* 1993; **38**(11):1623–1633.

[25] Jadbabaie A, Yu J, Hauser J. Unconstrained receding horizon control of nonlinear systems. *IEEE Trans. Auto. Control* 2001; **46**(5):776–783.

[26] Chen H, Allgower F. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica* 1998; **34**(10):1205–1217.

[27] Chen C, Shaw L. On receding horizon feedback control. *Automatica* 1982; **18**:349–352.

[28] Shamma J, Xiong D. Linear non-quadratic optimal control. *IEEE Trans. on Auto. Control* 1997; **42**(6):875–879.

[29] Nevistic V, Primbs J. Finite receding horizon linear quadratic control: A unifying theory for stability and performance analysis. *Technical Report CIT-CDS 97-001*, California Institute of Technology, Pasadena, California Jan 1997.

[30] Primbs JA, Nevistic V. Feasibility and stability of constrained finite receding horizon control. *Automatica*, vol. 36, 2000; 965–971.

[31] Scattolini R, Schiavoni N. A multirate model-based predictive control. *IEEE Trans. on Auto. Control* 1995; **40**(6):1093–1097.

[32] Lee H, Teo K, Rehbock V, Jennings L. Control parametrization enhancing technique for optimal discrete-valued control problems. *Automatica* 1999; **35**(8):1401–1407.

[33] Liberzon D. *Switching in Systems and Control.* Birkhäuser: Boston, MA, 2003.

[34] Utkin VI. Variable structure systems with sliding modes. *IEEE Trans. on Auto. Control* 1977; **22**(2):212–222.

[35] Edwards C, Spurgeon S. *Sliding Mode Control: Theory and Applications.* Taylor & Francis: Padstow, 1998.

[36] Lazar M, Heemels W, Weiland S, Bemporad A. Stabilizing model predictive control of hybrid systems. *IEEE Trans. on Auto. Control* 2006; **51**(11):1813–1818.

[37] Lazar M. Model predictive control of hybrid systems: stability and robustness. PhD Thesis, Technische Universiteit Eindhoven, Eindhoven 2006.

[38] Aspentech. DMCplus ; URL `http://www.aspentech.com`.

[39] invensys. Connoisseur ; URL `http://iom.invensys.com`.

[40] Pavilion Technologies. Process Perfecter ; URL `http://www.pavtech.com`.

[41] Kowalska K. Framework for the development of variable transition controllers. Master's Thesis, McMaster University, Hamilton 2003.

[42] Abdelwahed S, Karsai G, Biswas G. Online safety control of a class of hybrid systems. *Proceedings of the 41st IEEE Conference on Decision and Control*, vol. 2, 2002; 1988 – 1990.

[43] v Mohrenschildt M. Model predictive traces. *Journal of Software Engineering and Knowledge Engineering, Special Issue: Selected Papers from the 2005 International Conference on Embedded and Hybrid Systems*, vol. 15, 2005; 289–298.

[44] Zabczyk J. *Mathematical Control Theory: An Introduction*. Birkhäuser: Boston, 1995.

[45] Heath M. *Scientific Computing: An Introductory Survey*. McGraw-Hill: London, 1999.

[46] Rawlings J, Muske K. The stability of constrained receding horizon control. *IEEE Trans. on Auto. Control* 1993; **38**(10):1512–1516.

[47] Maplesoft. Maple computer algebra system ; URL `http://www.maplesoft.com`.

[48] NAG. Nag fortran library routine document: E04ucf/e04uca ; URL `http://www.nag.co.uk/numeric/fl/manual20/pdf/E04/e04ucf.pdf`.

[49] Wächter A, Biegler LT. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming* 2006; **106**(1):25–57.

[50] Bartlett RA, Wächter A, Biegler LT. Active set vs. interior point methods for nonlinear model predictive control. *Proc. American Control Conference*, vol. 6, 2000; 4229–4233.

[51] Zeilinger MN, Jones CN, Morari M. Real-time suboptimal model predictive control using a combination of explicit mpc and online optimization. *IEEE Trans. on Auto. Control* 2011; **56**(7):1524–1534.

[52] Bemporad A. Model predictive control design: New trends and tools. *Proceedings of the 45th IEEE Conference on Decision and Control*, December 13-15, 2006; 6678–6683.

[53] Scokaert P, Rawlings J. Constrained linear quadratic regulation. *IEEE Trans. on Auto. Control* 1998; **43**(8):1163–1169.