CHAPTER 3

MODEL PREDICTIVE CONTROL

This chapter describes MPC and especially RNMPC that is applied to the plant outlined in Chapter 2. The chapter starts by explaining MPC and its history, followed by a description of robust MPC and the reason for its development, and finally focuses on the controller theory used for the simulation study in Chapter 5. The description of MPC and the development of stability theory, including robust stability, in Sections (3.1)-(3.4) are summaries from the survey done by Mayne *et al.* (2000) and provided here for background.

3.1 INTRODUCTION

MPC, also known as receding horizon control (RHC), takes a measurement of the plant, uses a mathematical model of the system to predict its future behaviour in order to calculate a sequence of control moves (N steps) into the future that will optimise (usually minimise) an objective or penalty function, which describes a measure of performance of the system. The first control move of the calculated sequence is applied to the system and a new measurement is taken. The process is then repeated for the next time step. MPC calculates the control sequence on-line at each time step, compared to conventional control theory where the control law is pre-calculated and valid for all possible states of the system. MPC has the distinct advantage of controlling multi-variable systems well and can explicitly take into consideration constraints on the inputs (such as actuators, valves, etc.) as well as states or outputs (Camacho and Bordons, 2003). MPC is especially useful in situations where an explicit controller cannot be calculated offline.

The basic ideas present in the MPC family, according to Camacho and Bordons (2003), are that

- outputs at future time instances are predicted by the explicit use of a *mathematical model,*

- an *objective function* is minimised by calculating the appropriate control sequence, and
- at each time instant, the horizon is displaced towards the future, which involves applying the first control signal calculated at each time instance to the system; called the *receding horizon* strategy.

The MPC theory described in this chapter is in discrete time and the system takes the following form (Mayne *et al.*, 2000):

$$x(k+1) = f(x(k),u(k)), \tag{3.1}$$
$$y(k) = h(x(k)). \tag{3.2}$$

To simplify the notation, the value of x at time k is given by $x_k = x(k)$. The value of x at time $k+1$, therefore, becomes $x_{k+1} = x(k+1)$, the value of y at time k becomes $y(k) = y_k$ and the value of u at time k becomes u_k. The equations (3.1)-(3.2), therefore, becomes

$$x_{k+1} = f(x_k,u_k), \tag{3.3}$$
$$y_k = h(x_k). \tag{3.4}$$

The control and state sequences must satisfy

$$x_k \in \mathbb{X}, \tag{3.5}$$
$$u_k \in \mathbb{U}, \tag{3.6}$$

where $\mathbb{X} \subset \mathbb{R}^{n_x}$ and $\mathbb{U} \subset \mathbb{R}^{n_u}$.

Given a control vector sequence of length N, $\mathbf{u}^N = \{u^N(k), u^N(k+1), \ldots, u^N(k+N-1)\}$, where $u^N(k+1)$ specifies the control vector at time $k+1$, the predicted state vector sequence based on the control sequence $\mathbf{u}^N$ is given by

$$\mathbf{x}^{\mathbf{u}}(x_k,k) = \{x^{\mathbf{u}}(k,x_k,k), x^{\mathbf{u}}(k+1,x_k,k), \ldots, x^{\mathbf{u}}(k+N-1,x_k,k), x^{\mathbf{u}}(k+N,x_k,k)\}, \tag{3.7}$$

where $x^{\mathbf{u}}(k+1,x_k,k)$ is the calculated state vector at time $k+1$ from the initial state x_k, initial time k and control vector $u^N(k)$ using $f(\cdot,\cdot)$. The notation for the state sequence is simplified by stating the current state x_k and time k as subscripts to give $\mathbf{x}^N_{x_k,k} = \mathbf{x}^N(x_k,k) = \left\{x^{\mathbf{u}}_{x_k,k}(k), x^{\mathbf{u}}_{x_k,k}(k+1), \ldots, x^{\mathbf{u}}_{x_k,k}(k+N-1), x^{\mathbf{u}}_{x_k,k}(k+N)\right\}$.

The objective function that is used in the optimisation process has the following form:

$$\phi(x_k,k,\mathbf{u}^N) = \sum_{i=k}^{k+N-1} L(x^{\mathbf{u}}_{x_k,k}(i), u^N(i)) + E(x^{\mathbf{u}}_{x_k,k}(k+N)), \tag{3.8}$$

where $L(x^{\mathbf{u}}_{x_k,k}(i), u^N(i))$ is the cost at each time step into the future with regard to the states and inputs, while $E(x^{\mathbf{u}}_{x_k,k}(k+N))$ is the cost at the final state, reached after the whole control sequence has been applied. At each time k, the final time is $k+N$, which increases as k increases and is called a *receding horizon*. In certain MPC formulations, a terminal constraint set is defined

$$x^{\mathbf{u}}_{x_k,k}(k+N) \quad \in \quad X_f \subset \mathbb{X}. \tag{3.9}$$

The optimal control problem $P(x_k,k)$ of minimising the objective function is performed subject to the constraints on the control and state sequences, and in certain cases the terminal constraint to yield the optimised control sequence

$$\mathbf{u}^{\text{opt}}(x_k,k) \quad = \quad \left\{u^{\text{opt}}_{x_k,k}(k), u^{\text{opt}}_{x_k,k}(k+1), ..., u^{\text{opt}}_{x_k,k}(k+N-1)\right\}, \tag{3.10}$$

and optimised value for the objective function

$$\phi^{\text{opt}}(x_k,k) \quad = \quad \phi(x_k,k,\mathbf{u}^{\text{opt}}_{x_k,k}). \tag{3.11}$$

The first control move at time k of the sequence $\mathbf{u}^{\text{opt}}(x_k,k)$ is implemented to form an implicit control law for time k

$$\kappa(x_k,k) \quad = \quad u^{\text{opt}}_{x_k,k}(k). \tag{3.12}$$

The objective function is time invariant, because neither $L(x^{\mathbf{u}}_{x_k,k}(i), u^N(i))$ nor $E(x^{\mathbf{u}}_{x_k,k}(k+N))$ has terms that depend on time. The optimal control problem $P(x_k,k)$ can be defined as starting at time 0 to give $P_N(x_k) = P(x_k,0)$. N represents the finite prediction horizon over which the optimisation takes place, and the optimisation problem can be redefined as

$$P_N(x_k) : \phi^{\text{opt}}_N(x_k) \quad = \quad \min_{\mathbf{u}^N} \left\{\phi_N(x_k,\mathbf{u}^N) | \mathbf{u}^N \in \mathbb{U}_N\right\}, \tag{3.13}$$

where the objective function is now

$$\phi_N(x_k,\mathbf{u}^N) \quad = \quad \sum_{i=0}^{N-1} L(x^{\mathbf{u}}_{x_k}(i), u^N(i)) + E(x^{\mathbf{u}}_{x_k}(N)), \tag{3.14}$$

with $\mathbb{U}_N$ the set of feasible control sequences that satisfy the control, state and terminal constraints. If problem $P_N(x_k)$ is solved, the optimal control sequence is obtained

$$\mathbf{u}^{\text{opt}}(x_k) \quad = \quad \left\{u^{\text{opt}}_{x_k}(0), u^{\text{opt}}_{x_k}(1), ..., u^{\text{opt}}_{x_k}(N-1)\right\}, \tag{3.15}$$

and the optimal state trajectory, if the control actions are implemented, is given by

$$\mathbf{x}^{\text{opt}}(x_k) \quad = \quad \left\{x^{\text{opt}}_{x_k}(0), x^{\text{opt}}_{x_k}(1), ..., x^{\text{opt}}_{x_k}(N-1), x^{\text{opt}}_{x_k}(N)\right\}. \tag{3.16}$$

The optimal objective value is

$$\phi_N^{\text{opt}}(x_k) \quad = \quad \phi_N(x_k, \mathbf{u}_{x_k}^{\text{opt}}). \tag{3.17}$$

The first control action is implemented, leading to the implicit time invariant control law

$$\kappa_N(x_k) \quad = \quad u_{x_k}^{\text{opt}}(0). \tag{3.18}$$

Dynamic programming can be used to determine a sequence of objective functions $\phi_j(\cdot)$ deterministically in order to calculate the sequence of control laws $\kappa_j(\cdot)$ offline, where j is the time-to-go until the prediction horizon. This is possible because of the deterministic nature of the open-loop optimisation. This would be preferable, but is usually not possible. The difference between MPC and dynamic programming is purely a matter of implementation. MPC differs from conventional optimal control theory in that MPC uses a receding horizon control law $\kappa_N(\cdot)$ rather than an infinite horizon control law.

3.2 HISTORICAL BACKGROUND

MPC builds on optimal control theory, the theory (necessary and sufficient conditions) of optimality, Lyapunov stability of the optimal controlled system, and algorithms for calculating the optimal feedback controller (if possible) (Mayne *et al.*, 2000). There are a few important ideas in optimal control that underlie MPC. The first links together two principles of the control theory developed in the 1960s: the Hamilton-Jacobi-Bellman theory (Dynamic Programming) and the maximum principle, which provides necessary conditions for optimality. Dynamic programming provides sufficient conditions for optimality, as well as a procedure to synthesise an optimal feedback controller $u = \kappa(x_k)$. The maximum principle provides necessary conditions of optimality as well as computational algorithms for determining the optimal open-loop control $u_{x_k}^{\text{opt}}(\cdot)$ for a given initial state x_k. These two principles are linked together as

$$\kappa(x_k) \quad = \quad u_{x_k}^{\text{opt}}(0), \tag{3.19}$$

in order for the optimal feedback controller to be obtained by calculating the open-loop control problem for each x (Mayne *et al.*, 2000). From the commencement of optimal control theory it is stated by Lee and Markus (1967, p. 423): "*One technique for obtaining a feedback controller synthesis from knowledge of open-loop controllers is to measure the current control process state and then compute very rapidly for the open-loop control function. The first portion of this function is then used during a short time interval, after which a new measurement of the process state is made and a new open-loop control function is computed for this new measurement. The procedure is then repeated.*"

Kalman, as discussed in Mayne *et al.* (2000), observed that *optimality* does not guarantee *stability*. There are conditions under which optimality results in stability: *infinite horizon* controllers are stabilising, if the system is stabilisable and detectable. Calculating infinite horizon optimal solutions is not always practical on-line and an alternate solution was needed to stabilise the receding horizon controller. The first results for stabilising receding horizon controllers were given by Kleinman (1970), who developed a minimum energy controller for linear systems. He showed that the feedback controller is linear, time invariant and stable if a Lyapunov function $\phi(x) = x^T Px$ is used as the objective function. Another approach is to define a *stability constraint* as part of the optimal control problem. The stability constraint is defined as an equality constraint $x(T) = 0$ that forces the solution to converge to the origin. Thomas, as discussed in Mayne *et al.* (2000), suggested this technique as part of a linear quadratic control problem and implemented it by using $M \triangleq P^{-1}$ in place of P as the Riccati variable and solving the Riccati-like differential equation with terminal condition $M(T) = 0$.

MPC was really driven by industry as part of process control theory. Richalet *et al.* (1978) was the first to propose MPC for process control applications, but MPC was proposed earlier by Propoi and Lee and Markus (as discussed in Mayne *et al.* (2000)). The MPC method, called identification and command (IDCOM), was proposed by Richalet *et al.* (1978). It uses a linear model in the form of a finite horizon impulse response, quadratic cost and constraints on the inputs and outputs. The method makes provision for linear estimation using least squares, and the algorithm for solving the open-loop optimal control problem is the "dual" of the identification algorithm.

DMC is a later method proposed by Cutler and Ramaker (1980) and Prett and Gillette (as discussed in Mayne *et al.* (2000)). DMC uses a step response model, but as in IDCOM, handles constraints in an ad hoc fashion. This limitation was addressed by García and Morshedi (as discussed in Mayne *et al.* (2000)) by using quadratic programming to solve the constrained open-loop optimisation problem. This method also allows certain violations of the constraints in order to enlarge the set of feasible states. This method is called Quadratic Dynamic Matrix Control (QDMC).

The third generation of MPC technology, introduced about a decade ago, *"distinguishes between several levels of constraints (hard, soft and ranked). This technology provides some mechanism to recover from an infeasible solution, and addresses the issues resulting from a control structure that changes in real time, and allows for a wider range of process dynamics and controller specifications*" (Qin and Badgwell, 2003). The Shell multi-variable optimising control (SMOC) uses state-space models, incorporates general disturbance models and allows for state estimation using Kalman filters (as discussed in Mayne *et al.* (2000)).

An independent but similar approach was developed from the adaptive control theory and is called generalised predictive control (GPC). The method uses models in the backward shift operator q^{-1} which is more general than the impulse and step response models of DMC. GPC started as minimum variance control (Mayne *et al.*, 2000) that only allowed for a horizon of

length 1. Minimum variance control was extended to allow for longer prediction horizons by Peterka (1984) as well as Clarke *et al.* (1987*a*, *b*). GPC, and early versions of DMC, did not explicitly incorporate stability in the method and had to rely on the tuning of the prediction horizon as well as the weights of the states and inputs to achieve stability.

3.3 STABILITY OF MPC

The inability of both GPC and DMC to guarantee stability caused researchers to focus more on modifying $P_N(x)$ to ensure stability, owing to increased criticism (Bitmead *et al.*, 1990) of the makeshift approach of using tuning to attain stability.

With *terminal equality constraints*, the system is forced to the origin by the controller that takes the form $E(x) = 0$, as there is no terminal cost and the terminal set is $X_f = \{0\}$. Keerthi and Gilbert, as discussed in Mayne *et al.* (2000), proposed this stabilising strategy for constrained, nonlinear, discrete time systems and showed a stability analysis of this version (terminal equality constraints) of discrete-time receding horizon control. MPC, with a terminal equality constraint, can be used to stabilise a system that cannot be stabilised by continuous feedback controllers, according to Meadows *et al.* (as discussed in Mayne *et al.* (2000)).

Using a *terminal cost function* is an alternative approach to ensure stability. Here the terminal cost is $E(\cdot)$, but there is no terminal constraint and the terminal set is thus $X_f = \mathbb{R}^{n_x}$. For unconstrained linear systems the terminal cost of $E(x) = \frac{1}{2}x^T P_f x$ is proposed by Bitmead *et al.* (1990).

Terminal constraint sets differ from terminal equality constraints, in that subsets of $\mathbb{R}^{n_x}$ that include a neighbourhood of the origin are used to stabilise the control, not just the origin. The terminal constraint set, like the terminal equality constraint, does not employ a terminal cost, thus $E(x) = 0$. The MPC controller should steer the system to X_f within a finite time, after which a local stabilising controller $\kappa_f(\cdot)$ is employed. This methodology is usually referred to as dual mode control and was proposed by Michalska and Mayne (1993) in the context of constrained, nonlinear, continuous systems using a variable horizon N.

A *terminal cost and constraint set* is employed in most modern model predictive controller *theory* (Mayne *et al.*, 2000). If an infinite horizon objective function can be used, on-line optimisation is not necessary and stability and robustness can be guaranteed. In practical systems, constraints and other nonlinearities make the use of infinite horizons impossible, but it is possible to approximate an infinite horizon objective function if the system is suitably close to the origin. By choosing the terminal set X_f as a suitable subset of $\mathbb{R}^{n_x}$, the terminal cost $E(\cdot)$ can be chosen to approximate an infinite horizon objective function. A terminal cost and constraint set controller therefore needs a terminal constraint set $\mathbb{X}_f$ in which the terminal cost $E(\cdot)$ and infinite horizon feedback controller K_f are employed. To synthesise these, Sznaier and Damborg (as discussed in Mayne *et al.* (2000)) proposed that the terminal cost

$E(\cdot)$ and feedback controller K_f of a standard linear-quadratic (LQ) problem be used, which is an unconstrained infinite horizon problem, when the system is linear ($f(x,u) = Ax + Bu$) and the state and input constraint sets, $\mathbb{X}$ and $\mathbb{U}$, are polytopes. The terminal constraint set X_f is chosen to be the *maximal output admissible set* (Gilbert and Tan, 1991) of the system $f(x,u) = (A + BK_f)x$.

Most industrial or commercial MPC controllers do not use terminal costs or constraints, because they do not even provide nominal stability that these terminal costs and constraints are designed to provide and can be attributed to their DMC and IDCOM heritage (Qin and Badgwell, 2003). Most industrial MPC controllers, therefore, require brute-force simulation to evaluate the effects of model mismatch on closed-loop stability (Qin and Badgwell, 2003). Time spent tuning and testing of industrial controllers can, however, be significantly reduced if the controllers implement nominal and potentially robust stability measures, even though closed-loop stability of industrial MPC itself is not perceived to be a serious problem by industry practitioners (Qin and Badgwell, 2003).

3.3.1 Stability conditions for model predictive controllers

From the above discussion, it is clear that the addition of a terminal constraint set X_f, terminal cost $E(\cdot)$ and local feedback controller κ_f in the terminal constraint set forms the basis of stabilising MPC. Some conditions, in the form of axioms, are formulated (Mayne *et al.*, 2000) for the terminal constraint set, terminal cost and local feedback controller, which ensures that the controller is stabilising.

Two related methods are available for establishing stability. Both methods use a Lyapunov function as the objective function. The first method ensures that the objective function $\phi_N^{\text{opt}}(x_k)$ evolves with the state from x_k to $x_{k+1} = f(x_k, \kappa_N(x_k))$ so that

$$\phi_N^{\text{opt}}(x_{k+1}) - \phi_N^{\text{opt}}(x_k) + L(x_k, \kappa_N(x_k)) \quad \leq \quad 0, \tag{3.20}$$

while the alternative method uses the fact that

$$\phi_N^{\text{opt}}(x_{k+1}) - \phi_N^{\text{opt}}(x_k) + L(x_k, \kappa_N(x)) \quad = \quad \phi_N^{\text{opt}}(x_{k+1}) - \phi_{N-1}(x_{k+1}), \tag{3.21}$$

and shows that the right-hand side is negative, either directly or by showing that $\phi_1^{\text{opt}}(\cdot) \leq \phi_0^{\text{opt}}(\cdot)$ and exploiting monotonicity, which implies that if $\phi_1^{\text{opt}}(\cdot) \leq \phi_0^{\text{opt}}(\cdot)$ then $\phi_{i+1}^{\text{opt}}(\cdot) \leq \phi_i^{\text{opt}}(\cdot)$ for all $i \geq 0$.

Assume a model predictive controller that can steer the system state x to the terminal constraint set X_f within the prediction horizon N or fewer steps. The control sequence that accomplishes this is called an admissible or feasible control sequence $\mathbf{u} = \{u(0), u(1), ..., u(N-1)\}$. This control sequence should satisfy the control constraints $u(i) \in \mathbb{U}$ for $i = 0, 1, ..., N-1$ and ensure that the controlled states satisfy the state constraints $x_{x_k}^{\mathbf{u}}(i) \in \mathbb{X}$ for $i = 0, 1, .., N$

and the final state satisfies the terminal constraint set $x_{x_k}^{\mathbf{u}}(N) \in \mathbb{X}_f$. If the control problem $P_N(x_k)$ is solved, the control sequence $\mathbf{u}^{\text{opt}}(x_k)$ is obtained that will steer the system within the set of states that is possible with an MPC with horizon N, $x \in \mathbb{X}_N$. The optimal control sequence $\mathbf{u}^{\text{opt}}(x_k) = \left\{u_{x_k}^{opt}(0), u_{x_k}^{opt}(1), ..., u_{x_k}^{opt}(N-1)\right\}$ will result in the optimal state sequence $\mathbf{x}^{\text{opt}}(x_k) = \left\{x_{x_k}^{\text{opt}}(0), x_{x_k}^{\text{opt}}(1), ..., x_{x_k}^{\text{opt}}(N-1), x_{x_k}^{\text{opt}}(N)\right\}$. The first control action of $\mathbf{u}^{\text{opt}}(x_k)$, that is $u_k = \kappa_N(x_k) = u_{x_k}^{\text{opt}}(0)$ is implemented to get to the next state $x_{k+1} = f(x_k, \kappa_N(x_k)) = x_{x_k}^{\text{opt}}(1)$. A feasible control sequence $\tilde{\mathbf{x}}(x_{k+1})$ for the state x_{k+1}, will result in an upper bound for the optimal objective function $\phi_N^{\text{opt}}(x_{k+1})$, because a feasible control sequence should give a larger value for the objective function than an optimal control sequence. The abbreviated control sequence $\left\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1)\right\}$ derived from $\mathbf{u}^{\text{opt}}(x_k)$ should be a feasible control sequence to steer state x_{k+1} to $x_{x_k}^{\text{opt}}(N) \in \mathbb{X}_f$. If an extra term is added to the control sequence $\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1), v\}$, the control sequence will be feasible for $P_N(x_{k+1})$ if $v \in \mathbb{U}$ and v steers $x_{x_k}^{\text{opt}}(N) \in \mathbb{X}_f$ to $f(x_{x_k}^{\text{opt}}(N), v) \in \mathbb{X}_f$. This will be true if $v = \kappa_f(x_{x_k}^{\text{opt}}(N))$, with the terminal state constraint X_f and local controller $\kappa_f(\cdot)$ having the properties:

$$\mathbb{X}_f \subset \mathbb{X},\ \kappa_f(x_k) \in \mathbb{U} \quad \text{and} \quad f(x_k, \kappa_f(x_k)) \in \mathbb{X}_f\ \forall x_k \in \mathbb{X}_f, \tag{3.22}$$

implying that the terminal set $\mathbb{X}_f$ is invariant when the controller is $\kappa_f(\cdot)$. The feasible control sequence for $P_N(x_{k+1})$ is

$$\tilde{\mathbf{u}}(x_k) \quad = \quad \left\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1), \kappa_f(x_{x_k}^{\text{opt}}(N))\right\}, \tag{3.23}$$

with the associated cost

$$\begin{aligned} \phi_N(x_{k+1}, \tilde{\mathbf{u}}(x_k)) \quad = \quad & \phi_N^{\text{opt}}(x_k) - L(x_k, \kappa_N(x_k)) - E(x_{x_k}^{\text{opt}}(N)) \\ & + L(x_{x_k}^{\text{opt}}(N), \kappa_f(x_{x_k}^{\text{opt}}(N))) \\ & + E(f(x_{x_k}^{\text{opt}}(N), \kappa_f(x_{x_k}^{\text{opt}}(N)))). \end{aligned} \tag{3.24}$$

This cost $\phi_N(x_{k+1}, \tilde{\mathbf{u}}(x_k))$ is the upper bound on $\phi_N^{\text{opt}}(x_{k+1})$ and satisfies

$$\phi_N(x_{k+1}, \tilde{\mathbf{u}}(x_k)) \quad \leq \quad \phi_N^{\text{opt}}(x_k) - L(x_k, \kappa_N(x_k)), \tag{3.25}$$

if

$$E(f(x_k, \kappa_f(x_k))) - E(x_k) + L(x_k, \kappa_f(x_k)) \leq 0,\ \forall x_k \in \mathbb{X}_f. \tag{3.26}$$

The condition (3.26) will hold if $E(\cdot)$ is a control Lyapunov function in the neighbourhood of the origin and the controller κ_f and the terminal constraint set $\mathbb{X}_f$ are chosen appropriately. If the condition (3.26) is satisfied, then (3.20) will hold for all $x_k \in \mathbb{X}_N$, which is sufficient for the closed-loop system $x_{k+1} = f(x_k, \kappa_N(x_k))$ to converge to zero as time k tends to infinity, provided that the initial state is within $\mathbb{X}_N$. The stability conditions can be summarised in the following axioms (Mayne *et al.*, 2000):

A1: $\mathbb{X}_f \subset \mathbb{X}$, $\mathbb{X}_f$ is a closed set and $0 \in \mathbb{X}_f$. This condition implies that the state constraints should be satisfied in the terminal constraint set.

A2: $\kappa_f(x) \in \mathbb{U}$, $\forall x \in \mathbb{X}_f$. This condition implies that the constraints on the controls should be satisfied by the local controller in the terminal constraint set $\mathbb{X}_f$.

A3: $f(x, \kappa_f(x)) \in \mathbb{X}_f$, $\forall x \in \mathbb{X}_f$. This implies that the terminal constraint set $\mathbb{X}_f$ is positively invariant under the local controller $\kappa_f(\cdot)$.

A4: $E(f(x, \kappa_f(x))) - E(x) + L(x, \kappa_f(x)) \leq 0 \; \forall x \in \mathbb{X}_f$. The terminal cost function $E(\cdot)$ is a local Lyapunov function in the terminal constraint set $\mathbb{X}_f$.

The conditions, as summarised in A1 to A4, are merely sufficient conditions to ensure stability in model predictive controllers. These conditions can be shown to hold for the monotonicity approach as well as the continuous case (Mayne *et al.*, 2000). The following paragraphs will show how the stabilising methods of Section 3.3 satisfy the stability conditions A1 to A4.

3.3.2 Terminal state MPC

The *terminal state* variant of model predictive controllers (Mayne *et al.*, 2000) uses the terminal state $\mathbb{X}_f = \{0\}$ with no terminal cost $E(\cdot) = 0$. The local controller in the terminal constraint set is $\kappa_f(x) = 0$ that will ensure that the state remains at the origin if this controller is applied. The functions $E(\cdot)$ and $\kappa_f(\cdot)$ are only valid in $\mathbb{X}_f$ which is at the origin. The satisfaction of the stability conditions A1 to A4 are as follows:

A1: $\mathbb{X}_f = \{0\} \in \mathbb{X}$ - Satisfied.

A2: $\kappa_f(0) = 0 \in \mathbb{U}$ - Satisfied.

A3: $f(0, \kappa_f(0)) = f(0,0) = 0 \in \mathbb{X}_f$ - Satisfied.

A4: $E(f(0, \kappa_f(0))) - E(0) + L(0, \kappa_f(0)) = 0$ - Satisfied.

The controller ensures that the closed-loop system is asymptotically (exponentially) stable with region of attraction $\mathbb{X}_N$.

3.3.3 Terminal cost MPC

Terminal cost model predictive controllers are only valid in linear unconstrained (Bitmead *et al.*, 1990) and linear, stable, constrained (Rawlings and Muske, 1993) cases. In order to ensure stability, a terminal constraint is necessary if the system is nonlinear or linear, constrained and unstable. *Linear, unconstrained systems* are defined as $f(x,u) = Ax + Bu$,

and $L(x,u) = \frac{1}{2}(|x|_Q^2 + |u|_R^2)$ where $Q > 0$ and $R > 0$. The first three conditions A1 to A3 are trivially satisfied in the unconstrained case, because $\mathbb{X} = \mathbb{R}^{n_x}$ and $\mathbb{U} = \mathbb{R}^{n_u}$. In the case where A and B are stabilisable, the local controller is defined as $\kappa_f \triangleq K_f x$, and $P_f > 0$ should satisfy the Lyapunov equation

$$A_f^T P_f A_f + Q_f = 0, \quad A_f \triangleq A + BK_f, \quad Q_f \triangleq Q + K_f R K_f, \tag{3.27}$$

then the terminal cost function $E(x) \triangleq \frac{1}{2}x^T P_f x$ satisfies A4 and the closed-loop system is asymptotically (exponentially) stable with a region of attraction $\mathbb{R}^{n_x}$. *Linear, constrained, stable* systems have control constraints $u \in \mathbb{U}$, but no constraints on the states, thus $\mathbb{X} = \mathbb{X}_f = \mathbb{R}^{n_x}$. In order to satisfy A2, the controller function, if linear, should be $\kappa_f(x) = 0$ (Rawlings and Muske, 1993), that leads to the first three conditions (A1 to A3) being satisfied. The final condition A4 is satisfied if the terminal cost function is $E(x) \triangleq \frac{1}{2}x^T P_f x$, where P_f satisfies the Lyapunov equation $A^T P_f A + Q = 0$, that results in a controller with asymptotic (exponential) stability with region of attraction $\mathbb{R}^{n_x}$.

3.3.4 Terminal constraint set MPC

Terminal constraint set model predictive controllers employ a terminal constraint set $x_{x_k}^{\mathbf{u}}(N) \in \mathbb{X}_f$ without a terminal cost $E(x) = 0$ for nonlinear, constrained systems. Michalska and Mayne (1993) introduced the idea of a variable prediction horizon N for continuous-time, constrained, nonlinear systems. Scokaert *et al.* (1999) proposed a fixed horizon version for nonlinear, constrained, discrete-time systems. The controller steers the state of the system x to within the terminal constraint set $\mathbb{X}_f$, after which a local stabilising controller $\kappa_f(x) = K_f x$ is employed. This type of MPC is sometimes referred to as *dual-mode MPC*. This method is similar to the terminal equality constraint method, except that the equality $\{0\}$ is replaced by a set $\mathbb{X}_f$. The local controller $\kappa_f(\cdot)$ and the terminal constraint set $\mathbb{X}_f$ are chosen to satisfy the first three conditions A1 to A3. The local controller $\kappa_f(\cdot)$ is chosen to steer the system exponentially fast to the origin for all states in the terminal constraint set ($\forall x \in \mathbb{X}_f$). The stage cost of the objective function $L(x, \kappa_f(x))$ should be 0 when the system state is within the terminal constraint set $\mathbb{X}_f$ in order to satisfy A4. A suitable choice for the stage cost is

$$L(x,u) \quad \triangleq \quad \alpha(x)\overline{L}(x,u), \tag{3.28}$$

where $\alpha(x) = 1$, $\forall x \notin \mathbb{X}_f$, else $\alpha(x) = 0$ and $\overline{L}(x,u) = \frac{1}{2}(x^T Q x + u^T R u)$, where $Q > 0$ and $R > 0$. The closed-loop system is exponentially stable with domain of attraction $\mathbb{X}_N$, because the MPC controller steers the system with initial state $x \in \mathbb{X}_N$ within finite time to $\mathbb{X}_f$ with the controller value $\kappa_N(\cdot)$.

3.3.5 Terminal cost and constraint set MPC

In *linear, constrained systems* the terminal cost function can be chosen as $E(x) = \phi_{uc}^0(x) = \frac{1}{2}x^T P_f x$, that is the same as the unconstrained infinite horizon optimal control problem. The local controller $\kappa_f(x_k) = K_f x_k$ is the optimal infinite horizon controller and the terminal constraint set $\mathbb{X}_f$ is the maximal admissible set for the system $x_{k+1} = A_f x_k$, $A_f \triangleq A + BK_f$, thus satisfying A1-A4. This results in an exponentially stable controller with domain of attraction $\mathbb{X}_f$. The ideal choice for the terminal cost would be to choose $E(x) = \phi_\infty^{\text{opt}}(x)$, the objective function of an infinite horizon optimal controller, that would result in the objective function for the model predictive controller being $\phi_N^{\text{opt}}(x) = \phi_\infty^{\text{opt}}(x)$, and on-line optimisation would not be necessary. The resulting model predictive controller will have all the advantages of infinite horizon control. This is usually not practical, and the use of the terminal constraint set $\mathbb{X}_f$ and $E(x) = \phi_{uc}^0(x) = \frac{1}{2}x^T P_f x$ approximates the advantages of using $E(x) = \phi_\infty^{\text{opt}}(x)$. The nonlinear case is also given in Mayne *et al.* (2000).

From this discussion, it is clear that the use of a terminal constraint set $\mathbb{X}_f$, terminal cost function $E(\cdot)$ and local stabilising controller $\kappa_f(\cdot)$ is necessary to ensure stability in MPC. The first two requirements, terminal constraint set $\mathbb{X}_f$ and terminal cost function $E(\cdot)$, are explicitly incorporated into the controller, while the feedback controller $\kappa_f(\cdot)$ is only implicitly needed to prove stability. If the cost function $E(\cdot)$ is as close to the objective function $\phi_\infty^{\text{opt}}(\cdot)$ as possible, the closed-loop trajectory is exactly the same as that predicted by the solution of the optimal control problem $P_N(x)$.

3.4 ROBUST MPC - STABILITY OF UNCERTAIN SYSTEMS

Robust MPC is concerned with the stability and performance of the closed-loop system in the presence of uncertainty in the plant model. Early studies in the robustness of model predictive controllers considered unconstrained systems and found that if the Lyapunov function retains its descent property in the presence of disturbances (uncertainty), it will remain stable. In the constrained case, the problem becomes more complex, because the uncertainty or disturbances should not cause the closed-loop system to violate its state or control constraints.

Richalet *et al.* (1978) performed one of the earliest studies in robustness on systems with impulse response models, by investigating the effect of gain mismatches on the closed-loop system. Later work on systems modelled by impulse responses approached the optimal control problem as a min-max problem, that caused the problem to grow exponentially with the size of the prediction horizon.

There are several approaches to robust MPC, the first being a study of the robustness of MPC designed with a nominal model (that does not take uncertainty into account). The second

approach considers all the possible realisations of the uncertain system when calculating the open-loop optimal controller (min-max open-loop MPC). The open-loop nature of MPC is a problem when model uncertainty is present and the third approach addresses this by introducing feedback in the optimal control problem that is solved on-line.

For the discussion of robust MPC, the uncertain system is described as

$$x_{k+1} = f(x_k, u_k, w_k), \tag{3.29}$$
$$y_k = h(x_k), \tag{3.30}$$

where the state x_k and control u_k satisfy the same constraints

$$x_k \in \mathbb{X}, \tag{3.31}$$
$$u_k \in \mathbb{U}, \tag{3.32}$$

and the disturbance or uncertainty w_k satisfies $w_k \in W(x_k, u_k)$ for all k where, for each (x_k, u_k), $W(x_k, u_k)$ is closed and contains the origin in its interior. The disturbance sequence $\mathbf{w}^N \triangleq \{w^N(0), w^N(1), ..., w^N(N-1)\}$, together with the control sequence $\mathbf{u}^N$ and initial state x_k, will produce the resulting state sequence

$$\mathbf{x}^{\mathbf{u},\mathbf{w}}(x_k) = \left\{x_{x_k}^{\mathbf{u},\mathbf{w}}(0), x_{x_k}^{\mathbf{u},\mathbf{w}}(1), \ldots, x_{x_k}^{\mathbf{u},\mathbf{w}}(N-1), x_{x_k}^{\mathbf{u},\mathbf{w}}(N)\right\}. \tag{3.33}$$

Let $\mathscr{F}(x_k, u_k) \triangleq f(x_k, u_k, W(x_k, u_k))$, which will map values in $\mathbb{X}$ and $\mathbb{U}$ to subsets of $\mathbb{R}^{n_x}$, resulting in $x_{k+1} \in \mathscr{F}(x_k, u_k)$.

De Nicolao *et al.* (1996) and Magni and Sepulchre (1997) studied the inherent robustness of model predictive controllers that were designed without taking uncertainty into account. A more recent study by Grimm *et al.* (2004) found that there are examples showing that when MPC is applied to a nonlinear system, it is asymptotically stable without any robustness to measurement error or additive disturbances. This only happens in a nonlinear system where both the MPC feedback control law and the objective function are discontinuous at some point(s) in the interior of the feasibility region.

3.4.1 Stability conditions for robust MPC

Most versions of robust MPC take all the realisations of the uncertainty or disturbance w into consideration that requires strengthened assumptions to be satisfied, which are summarised as *robust* versions of axioms A1-A4 (Mayne *et al.*, 2000):

A1: $\mathbb{X}_f \subset \mathbb{X}$, $\mathbb{X}_f$ closed, $0 \in \mathbb{X}_f$.

A2: $\kappa_f(x) \in \mathbb{U}$, $\forall x \in \mathbb{X}_f$.

A3a: $f(x, \kappa_f(x), w) \in \mathbb{X}_f$, $\forall x \in \mathbb{X}_f$, $\forall w \in W(x, \kappa_f(x))$.

A4a: $E(f(x,\kappa_f(x),w))-E(x)+L(x,\kappa_f(x),w)\leq 0,\ \forall x\in\mathbb{X}_f,\ \forall w\in W(x,\kappa_f(x))$.

If $E(\cdot)$ is a robust Lyapunov function in the neighbourhood of the origin, there exists a triple $(E(\cdot),\mathbb{X}_f,\kappa_f(\cdot))$, which ensures that A4a is satisfied and results in an asymptotically or exponentially stable controller.

3.4.2 Open-loop min-max MPC

Open-loop min-max MPC considers all the possible realisations of the uncertain system in order to ensure that the state, control and terminal constraints are met for all the possible realisations (Michalska and Mayne, 1993). The objective function value in this case is determined for each realisation

$$J(x_k,\mathbf{u}^N,\mathbf{w}^N) \triangleq \sum_{i=0}^{N-1} L(x_{x_k}^{\mathbf{u},\mathbf{w}}(i),u^N(i))+E(x_{x_k}^{\mathbf{u},\mathbf{w}}(N)), \tag{3.34}$$

and the final objective value is the worst case for all the realisations

$$\phi_N(x_k,\mathbf{u}^N) \triangleq \max_{\mathbf{w}^N}\left\{J(x_k,\mathbf{u}^N,\mathbf{w}^N)|\mathbf{w}^N\in W_N(x_k,\mathbf{u}^N)\right\}, \tag{3.35}$$

where $W_N(x_k,\mathbf{u}^N)$ is the set of admissible disturbance sequences. Other choices are to take the objective value as the nominal objective value by using $\mathbf{w}^N=\mathbf{0}$. Badgwell (as discussed in Mayne *et al.* (2000)) used an interesting approach, where the controller should reduce the objective function value for every realisation, which is assumed finite, for a linear system. This is stronger than only reducing the worst-case objective value.

The set of admissible control sequences $\mathscr{U}_N^{ol}(x_k)$ is the one that satisfies the control, state and terminal constraints for all possible realisation of the disturbance sequence $\mathbf{w}^N$ when the initial state is x_k. Suppose the set $\mathbb{X}_i^{ol}$, for all $i\geq 0$, is the set of states that can be robustly steered to the terminal state constraint $\mathbb{X}_f$ in i steps or fewer by an admissible control sequence $\mathbf{u}^N\in\mathscr{U}_N^{ol}(x_k)$. The open-loop optimal control problem is

$$P_N^{ol}(x_k):\ \phi_N^{\text{opt}}(x_k) = \min_{\mathbf{u}^N}\left\{\phi_N(x_k,\mathbf{u}^N)|\mathbf{u}^N\in\mathscr{U}_N^{ol}(x_k)\right\}. \tag{3.36}$$

The solution to $P_N^{ol}(x_k)$ yields the optimal control sequence $\mathbf{u}^{\text{opt}}(x_k)$, where the implicit min-max control law is

$$\kappa_N^{ol}(x_k) \triangleq u_{x_k}^{\text{opt}}(0), \tag{3.37}$$

as in the nominal case. The control sequence will result in multiple optimal state sequences $\{\mathbf{x}^{\text{opt}}(x_k,\mathbf{w}^N)\}$ as a result of the disturbance sequence $\mathbf{w}^N$, so that

$$\mathbf{x}^{\text{opt}}(x_k,\mathbf{u}^{\text{opt}}) = \left\{x_{x_k,\mathbf{w}}^{\text{opt}}(0),x_{x_k,\mathbf{w}}^{\text{opt}}(1),\ldots,x_{x_k,\mathbf{w}}^{\text{opt}}(N-1),x_{x_k,\mathbf{w}}^{\text{opt}}(N)\right\}. \tag{3.38}$$

The triple $(E(\cdot), \mathbb{X}_f, \kappa_f(\cdot))$ is assumed to satisfy the stability conditions A1-A4a. Assume the process is started with an initial state $x_k \in \mathscr{X}_N^{ol}$ and has an optimal (and by implication a feasible) control sequence $\{u_{x_k}^{\text{opt}}(0), u_{x_k}^{\text{opt}}(1), ..., u_{x_k}^{\text{opt}}(N-1)\}$ for the optimal control problem $P_N^{ol}(x_k)$ that steers the state to within the terminal constraint set $\mathbb{X}_f$ within N steps or fewer, so that $x_{x_k,\mathbf{w}}^{\text{opt}}(N) \in \mathbb{X}_f$, $\forall w \in \mathscr{W}(x_k, \mathbf{u}^{\text{opt}}(x_k))$. As a result the abbreviated control sequence $\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1)\}$ should steer the state $x_{k+1} \in \mathscr{F}(x_k, \kappa_N(x_k))$ to the terminal constraint set $\mathbb{X}_f$ within $N-1$ steps or fewer, where $x_{k+1} \in \mathscr{X}_{N-1}^{ol}$. A problem arises when a feasible control sequence needs to be generated by adding a term to the abbreviated control sequence

$$\tilde{\mathbf{u}}(x_k) \quad = \quad \left\{u_{x_k}^{\text{opt}}(1), u_{x_k}^{\text{opt}}(2), ..., u_{x_k}^{\text{opt}}(N-1), v\right\}, \tag{3.39}$$

for the optimal control problem $P_N^{ol}(x_{k+1})$, where the control action $v \in \mathbb{U}$ is required to satisfy $f(x_{x_k,\mathbf{w}}^{\text{opt}}(N), v, w_N) \in \mathbb{X}_N$ for all $\mathbf{w}^N \in \mathscr{W}(x_k, \mathbf{u}^{\text{opt}}(x_k))$. The stability condition A3a does not ensure that such a control action v can be obtained, which prevents the upper bound of the objective function $\phi_N^{\text{opt}}(x_{k+1})$ from being calculated. Michalska and Mayne (1993) circumvent this problem by using a variable horizon optimal control problem $P(x_k)$ with decision variables $(\mathbf{u}^N, N)$. The optimal solution $(\mathbf{u}^{\text{opt}}(x_k); N^{\text{opt}}(x_k))$ is obtained by solving the optimal control problem $P(x_k)$, where

$$\mathbf{u}^{\text{opt}}(x_k) \quad = \quad \left\{u_{x_k}^{\text{opt}}(0), u_{x_k}^{\text{opt}}(1;x), ..., u_{x_k}^{\text{opt}}(N(x_k)-1)\right\}.$$

For the optimal control problem $P(x_{k+1})$ the solution $(\overline{\mathbf{u}}(x_k), N^{\text{opt}}(x_k) - 1)$ is a feasible solution for any $x_{k+1} \in \mathscr{X}(x_k, \kappa_N(x_k))$. The variable horizon objective function $\phi^{\text{opt}}(\cdot)$ and implicit controller $\kappa^{ol}(\cdot)$ will ensure that stability condition A4a holds for all $x_k \in \mathbb{X}_N^{ol} \subset \mathbb{X}_f$, $\forall w^N \in W(x_k, \kappa^{ol}(x_k))$. Inside the terminal constraint set $\mathbb{X}_f$, a suitable local controller $\kappa_f(\cdot)$ is used subject to stability conditions A1-A4a. This will result in an asymptotic (exponential) stable controller with domain of attraction $\mathbb{X}_N^{ol}$, subject to further modest assumptions (Michalska and Mayne, 1993).

3.4.3 Feedback robust MPC

Feedback robust MPC is better suited for uncertain systems than open-loop min-max controllers, because open-loop controllers assume that the trajectories of the system may diverge, which may cause $\mathbb{X}_N^{ol}$ to be very small, or even empty for a modest-sized prediction horizon N, which is very conservative. This happens because the open-loop min-max controllers do not take the effect of feedback into consideration, which would prevent the trajectories from diverging too much. To address the shortcomings of open-loop min-max control, feedback MPC was proposed by Lee and Yu (1997), Scokaert and Mayne (1998), Magni *et al.* (2001) and Kothare *et al.* (1996). In feedback MPC, the control sequence $\mathbf{u}$ is replaced by a control

policy π which is a sequence of control laws:

$$\pi \triangleq \{u(o), \kappa_1(\cdot), ..., \kappa_{N-1}(\cdot)\}, \tag{3.40}$$

where $\kappa_i(\cdot) : \mathbb{X} \to \mathbb{U}$ is a control law for each i, while $u(0)$ is a control action, because there is only one initial state. The resulting state sequence when applying the sequence of control laws π and starting at the initial state x_k subject to the disturbance sequence $\mathbf{w}$ is given by $\mathbf{x}^{\pi,\mathbf{w}}(x_k) = \{x_{x_k}^{\pi,\mathbf{w}}(0), x_{x_k}^{\pi,\mathbf{w}}(1), \ldots, x_{x_k}^{\pi,\mathbf{w}}(N-1), x_{x_k}^{\pi,\mathbf{w}}(N)\}$. The objective function for the feedback model predictive controller is

$$\phi_N(x_k, \pi) \triangleq \max_{\mathbf{w}^N} \{J(x_k, \pi, \mathbf{w}^N) | \mathbf{w}^N \in \mathscr{W}_N(x_k, \pi)\} \tag{3.41}$$

and the objective function for each realisation

$$J(x_k, \pi, \mathbf{w}^N) \triangleq \sum_{i=0}^{N-1} L(x_{x_k}^{\pi,\mathbf{w}}(i), u^{\pi}(i)) + E(x_{x_k}^{\pi,\mathbf{w}}(N)), \tag{3.42}$$

where $u^{\pi}(i) \triangleq \kappa_i(x_{x_k}^{\pi,\mathbf{w}}(i))$, $i = 1, \ldots, N-1$ and $u^{\pi}(0) = u(0)$. The admissible set of disturbances, given the control policy π is implemented, is $\mathscr{W}_N(x_k, \pi)$. The set of admissible control policies that will satisfy the control, state and terminal constraints for all the admissible disturbances with initial state x_k, is $\Pi_N(x_k)$. The set of initial states that can be steered to the terminal constraint set $\mathbb{X}_f$ by an admissible control policy π in i steps or fewer, is $\mathbb{X}_i^{fb}$, $\forall i \geq 0$. The feedback optimal control problem becomes

$$P_N^{fb}(x_k) : \phi_N^{\text{opt}}(x_k) = \min_{\pi} \{\phi_N(x_k, \pi) | \pi \in \Pi_N(x_k)\}. \tag{3.43}$$

If a solution to $P_N^{fb}(x_k)$ exists, the optimal control policy is

$$\pi^{\text{opt}}(x_k) = \left\{u_{x_k}^{\text{opt}}(0), \kappa_{1,x_k}^{\text{opt}}(\cdot), \kappa_{2,x_k}^{\text{opt}}(\cdot), ..., \kappa_{N-1,x_k}^{\text{opt}}(\cdot)\right\}, \tag{3.44}$$

where the implicit feedback MPC law is

$$\kappa_N^{fb}(x_k) \triangleq u_{x_k}^{\text{opt}}(0). \tag{3.45}$$

If the stability conditions A1-A4a are satisfied for $P_N^{fb}(x_k)$, a feasible control policy for $P_N^{fb}(x_{k+1})$ for all $x_{k+1} \in \mathscr{F}(x_k, \kappa_N^{fb}(x_k))$ and $x_k \in \mathbb{X}_N^{fb}$ is

$$\tilde{\pi}(x_k, x_{k+1}) \triangleq \left\{\kappa_{1,x_k}^{\text{opt}}(x_{k+1}), \kappa_{2,x_k}^{\text{opt}}(\cdot), ..., \kappa_{N-1,x_k}(\cdot), \kappa_f(\cdot)\right\}. \tag{3.46}$$

With this feasible control policy, and with $\mathbb{X}_N^{fb}$ an invariant set for $x_{k+1} \in \mathscr{F}(x_k, \kappa_N^{fb}(x_k))$, assumption A4a will be satisfied for all $x_k \in \mathbb{X}_N^{fb}$ and $w \in W(x_k, \kappa_N^{fb}(x_k))$. The resulting robust model predictive controller is asymptotically (exponentially) stable with domain of

attraction $\mathbb{X}_N^{fb}$ under further modest assumptions. The results are very similar to open-loop min-max control, except that the domain of attraction $\mathbb{X}_N^{fb}$ includes $\mathbb{X}_N^{ol}$ and could possibly be much larger. Feedback MPC is encouraging, but suffers from much higher complexity than open-loop min-max control.

Other formulations for robust MPC are also provided in Mayne *et al.* (2000).

3.5 ROBUST NONLINEAR MPC FORMULATIONS

3.5.1 Lyapunov-based robust model predictive control

Mhaskar and Kennedy (2008) propose a robust model predictive controller for nonlinear systems with constraints on the magnitude of the inputs and uncertainties as well as *rate constraints on the inputs*. Rate constraints can easily be handled by MPC formulations as soft constraints (Zheng and Morari, 1995), but the effect of rate constraints on stability and guaranteed satisfaction of rate constraints as hard constraints have not been addressed. The proposed formulation can handle rate constraints as soft constraints or as hard constraints when possible with fall-back to soft constraints to maintain feasibility and robust stability. The controller synthesis is Lyapunov-based to allow for explicit characterisation of the initial conditions that guarantee stabilisation, as well as state and control constraint satisfaction (Mhaskar *et al.*, 2005, 2006). The explicit characterisation of the initial conditions is important to ensure that stability is guaranteed, because the stability guarantees rely on stability constraints embedded in the optimisation problem. Guaranteeing stability requires feasibility of the stability constraints, which is assumed by most formulations, not guaranteed (Mhaskar and Kennedy, 2008).

Mhaskar *et al.* (2005) proposes a robust hybrid predictive control structure that acts as a safety net for any other MPC formulation. The other MPC formulations can explicitly take uncertainties into account, but it is not required. The control structure includes a Lyapunov-based robust nonlinear controller developed by El-Farra and Christofides (2003) that has well-defined stability characteristics and constraint-handling properties, but cannot be designed to be optimal to an arbitrary performance function as with MPC. The structure in addition contains switching laws that try to use the performance of MPC as much as possible, but falls back to the Lyapunov robust controller when the MPC fails to deliver a control move, which may be due to computational difficulties or infeasibility, or to instability of the MPC, which may be caused by inappropriate penalties or horizon length in the performance function (Mhaskar *et al.*, 2005).

Mhaskar (2006) proposes a robust model predictive controller for nonlinear systems that are subject to constraints and uncertainty and that is robust to *faults* in the control actuator. Lyapunov-based techniques (El-Farra and Christofides, 2003) are employed to design the robust model predictive controller by formulating constraints that explicitly account for the

uncertainties in the predictive control law and allow the set of initial conditions to be explicitly characterised that will guarantee constraint satisfaction initially and successive feasibility necessary for robust stability.

3.5.2 Reachable set methods

Feedback robust MPC is concerned with bounding the system trajectories to increase the possible domain of attraction compared to open-loop min-max robust MPC. Bravo *et al.* (2006) present an RNMPC for constrained nonlinear systems with uncertainties. The evolution of the system under uncertainties is predicted by using reachable sets of the system. The reachable sets are approximated by outer bounds on the reachable set. Bravo *et al.* (2006) use a method based on zonotopes (Alamo *et al.*, 2005) to describe the approximate reachable sets, which improves on a previous formulation by Limon *et al.* (2005) that used a conservative approximation of the reachable set based on interval arithmetic. The controller drives the system to a robust invariant set and forces the closed-loop system to be bounded by using contractive constraints.

3.5.3 Closed-loop min-max predictive control

Closed-loop min-max MPC reduces the conservatism of open-loop min-max MPC by incorporating the effect of feedback at each time-step into the optimisation problem. This limits the spread of the trajectories into the future and increases the feasible region of the controller. This was first proposed by Lee and Yu (1997) and further developed by Lazar *et al.* (2008), Limon *et al.* (2006), Magni *et al.* (2006, 2003). The stability of the closed-loop min-max MPC was studied in the input-to-state stability (ISS) framework (Jiang and Wang, 2001, Sontag, 1989, 1990, 1996) by Limon *et al.* (2006), Magni *et al.* (2006), where Limon *et al.* (2006) showed that in general only input-to-state *practical* stability (ISpS) (Jiang *et al.*, 1996, Sontag, 1996) can be ensured for min-max nonlinear MPC. It is desirable to have ISS compared to ISpS, because ISpS does not imply *asymptotic stability* when there are no disturbances on the inputs, while ISS ensures that *asymptotic stability* will be recovered when the input disturbances vanish. Lazar *et al.* (2008) proposed a new approach that guarantees ISS for min-max MPC of nonlinear systems with bounded disturbances.

3.5.4 Open-loop min-max predictive control

Open-loop min-max MPC of a continuous-time nonlinear system with constraints and uncertainties uses robust nonlinear optimal control at its core. The open-loop optimal control problem is solved for the current state measurement and the first control move implemented. The next state measurement is taken and the nonlinear optimal control problem resolved in a receding horizon fashion that leads to RNMPC.

In order to solve the continuous-time nonlinear optimal control problem, it should be discretized by casting it into a nonlinear parameter optimisation problem. Continuous-time nonlinear optimal control can be cast into a nonlinear parameter optimisation problem using the direct-multiple shooting formulation (Bock and Plitt, 1984).

Robust control can be obtained by performing robust nonlinear parameter optimisation, which is quite difficult to solve in general. Diehl *et al.* (2006) describes a method of approximating robust nonlinear optimisation through an approximate worst-case formulation. The approximation linearises the uncertainty set, described by norm bounded parameter uncertainty, to obtain penalty functions for the objective function and constraint functions. The penalty functions are further approximated to maintain the sparsity of the large-scale problem as well as the smoothness of the objective and constraints functions, enabling efficient implementation of the resulting approximate robust nonlinear parameter optimisation problem. This formulation can be solved quite efficiently by using a real-time iteration scheme for nonlinear optimisation proposed by Diehl *et al.* (2005).

3.5.5 Linear embedding of nonlinear models

Embedding of the trajectories of the original nonlinear system into a family of linear plants is exploited by some authors to implement robust MPC of even highly nonlinear systems by using linear methods (Casavola *et al.*, 2003). Linear matrix inequality (LMI) based controllers produce feedback policies, which are implemented at each time interval. The problem with these controllers is that they use an ellipsoid invariant set for their domain of attraction, which makes them conservative. This is because the sets must be symmetric, and in systems where the constraints are non-symmetric, the ellipsoid sets will be a small subset of the maximum admissible set. The feedback robust MPC technique was introduced by Kothare *et al.* (1996). The technique was improved by Cuzzola *et al.* (2002) by describing the uncertain system as a polytope and applying different Lyapunov functions to each vertex of the uncertain polytope to reduce the conservatism of the method. The method uses semidefinite programming (SDP) to solve the minimisation problem on-line, which is computationally very expensive compared to quadratic programming (QP) used in nominal MPC. Further improvements made by Casavola *et al.* (2004), Ding *et al.* (2004), Wan and Kothare (2003) resulted in an attempt to move as much as possible of the calculation offline.

Nonlinear systems with uncertainties and constraints can be approximated by radial basis function – auto-regressive with exogenous input (RBF-ARX) models, which are hybrid pseudo-linear time-varying models that contain elements of Gaussian RBF neural network and linear ARX model structures. Peng *et al.* (2007) used this RBF-ARX model in conjunction with a min-max MPC to do RNMPC. The min-max MPC is based on the LMI-based method proposed in Cuzzola *et al.* (2002) that falls in the feedback robust MPC framework.

An approach to feedback robust MPC is proposed by Langson *et al.* (2004) who use tubes to

encapsulate all the possible states that can result from the controller. If the uncertainties can be sufficiently described, the optimisation problem only has to be calculated once, and the control policy will steer the system to the terminal constraint set $\mathbb{X}_f$, where a local stabilising controller will keep the uncertain system in the terminal constraint set.

The robust model predictive controllers do not always provide off-set free tracking and this problem is addressed by Wang and Rawlings (2004*a*, *b*), who use a robust predictor that updates itself each time measurements are available to ensure that the off-set is eliminated. Pannocchia and co-workers (Pannocchia, 2004, Pannocchia and Kerrigan, 2003, 2005) approached the problem by designing a robust linear feedback controller and an appropriate invariant set where the controller will satisfy the constraints. The controller uses the "pre-stabilisation" approach suggested by Rossiter *et al.* (1998) and later implemented by Kouvaritakis *et al.* (2000), Schuurmans and Rossiter (2000) and Lee and Kouvaritakis (2000), where the feedback law $u_i(\cdot)$ in the policy π is restricted to have the form $u_i(x) = v_i + Kx$, $i = 0,1,2,...,N-1$, that changes the optimisation problem to calculating the free control moves $\{v_0, v_1, v_2, ..., v_{N-1}\}$ rather than the policy. The "pre-stabilisation" of the controller in Pannocchia and Kerrigan (2005), however, uses a dynamic state feedback controller, rather than a static state feedback gain to obtain the offset free tracking property.

3.6 NONLINEAR MODEL PREDICTIVE CONTROL

NMPC takes a measurement of the current state of the plant and then uses a nonlinear model to predict the future behaviour of the plant in order to calculate the optimal control moves or control laws with regard to a specified objective function. NMPC is derived from nonlinear optimal control over a constant or varying time interval into the future $[t_k, t_k + T]$. Only the first control move or control law is implemented and a new state measurement is taken. The nonlinear optimal control problem is then recalculated for the new time interval $[t_{k+1}, t_{k+1} + T]$, which leads to receding horizon control (Mayne *et al.*, 2000).

The nonlinear optimal control problem is to find a control profile $u(\cdot)$ such that it minimises a particular scalar performance index

$$\min_{x,u} \quad \phi_c(x,u) \tag{3.47}$$

$$\text{s.t.} \quad \dot{x}(t) = f_c(x(t), u(t), \tilde{p}) \tag{3.48}$$

$$\theta_c(x,u) \leq 0 \tag{3.49}$$

$$t_1 \triangleq 0,\ x_1 \triangleq x(t_1) \tag{3.50}$$

$$t_f \triangleq T,\ x_f \triangleq x(t_f),\ \psi(x_f) = 0 \tag{3.51}$$

where $x : \mathbb{R} \to \mathbb{R}^{n_x}$ is the state trajectory, $u : \mathbb{R} \to \mathbb{R}^{n_u}$ is the control trajectory, $x(t) \in \mathbb{R}^{n_x}$ is the state vector, $\dot{x}(t) \in \mathbb{R}^{n_x}$ is the state sensitivities to time, $u(t) \in \mathbb{R}^{n_u}$ is the control vector, $x_f \in \mathbb{R}^{n_x}$ is the terminal state vector, $(x,u) \mapsto \phi_c(x,u)$ is the scalar performance function,

$(x,u) \mapsto \theta_c(x,u)$ is the inequality constraints function, $\psi : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal constraint function, $\tilde{p} \in \mathbb{R}^{n_p}$ is the nominal parameter vector and $f_c : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ is the ordinary differential equation describing the dynamics of the plant. The plant dynamics are time invariant and each optimal control problem can therefore be solved from time $t_1 = 0$ without affecting the result. The initial state value $x_1 \in \mathbb{R}^{n_x}$ is the currently measured state of the system. The final state $x_f \in \mathbb{R}^{n_x}$ will be a fixed value based on the setpoint for the current iteration of the optimisation problem, because the terminal constraints are defined as equality constraints (3.55). The final state value x_f may vary from iteration to iteration if setpoint changes are made. The final time t_f of the optimisation problem is fixed for this implementation.

For the sequel the ordered pair $(a,b) \triangleq \begin{bmatrix} a^T & b^T \end{bmatrix}^T$ is defined as a column vector.

The nonlinear optimal control problem, consisting of a system with continuous dynamics, needs to be discretized in order to be cast in terms of a nonlinear parameter optimisation problem. This is accomplished by dividing the prediction horizon $[0,T]$ into N discrete time intervals called nodes (Hull, 1997) $t_0 \triangleq 0 < t_1 < t_2 < \cdots < t_k < \cdots < t_{N-1} < t_N \triangleq T$ where the sampling time is defined as $\tau_s \triangleq t_{k+1} - t_k$. The functions of time $x(\cdot)$ and $u(\cdot)$ are replaced by their values at the nodes $x_k \in \mathbb{R}^{n_x}$ and $u_k \in \mathbb{R}^{n_u}$ for $k = 0,\ldots,N$ and some form of interpolation between the nodes.

The resulting nonlinear controlled discrete-time system is $x_{k+1} \triangleq f_k(x_k,u_k,\tilde{p})$, $k = 0,1,\ldots,N-1$. The nonlinear optimal control problem can now be cast into the following nonlinear parameter optimisation problem

$$\min_{\mathbf{s},\mathbf{q}} \quad \phi(\mathbf{s},\mathbf{q}) \tag{3.52}$$

$$\text{s.t.} \quad g(\mathbf{s},\mathbf{q},\tilde{\mathbf{p}}) = 0 \tag{3.53}$$

$$\theta_{i,j}(s_j,q_j) \leq 0, \quad \begin{matrix} i = 1,\ldots,n_c, \\ j = 1,\ldots,N-1, \end{matrix} \tag{3.54}$$

$$\theta_N(s_N) \leq 0, \tag{3.55}$$

where $\phi : \mathbb{R}^{N\cdot n_x} \times \mathbb{R}^{(N-1)\cdot n_u} \to \mathbb{R}$ is the performance function to be optimised, $g : \mathbb{R}^{N\cdot n_x} \times \mathbb{R}^{(N-1)\cdot n_u} \times \mathbb{R}^{N\cdot n_p} \to \mathbb{R}^{N\cdot n_x}$ is the equality constraint function that describes the discrete time system dynamics, $\theta_{i,j} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$, $i = 1,\ldots,n_c$, $j = 1,\ldots,N-1$ are the inequality constraint functions, $\theta_N : \mathbb{R}^{n_x} \to \mathbb{R}$ is the terminal constraint function, $s_i \in \mathbb{R}^{n_x}$, $i = 1,\ldots,N$ are the estimated state parameters, $q_i \in \mathbb{R}^{n_u}$, $i = 1,\ldots,N-1$ are the control parameters, $\tilde{\mathbf{p}} \triangleq (\tilde{p},\ldots,\tilde{p}) \in \mathbb{R}^{(N-1)n_p}$ is the sequence of nominal model parameters, $\mathbf{s} \triangleq (s_1,\ldots,s_N)$ is the state sequence and $\mathbf{q} \triangleq (q_1,\ldots,q_{N-1})$ is the control sequence to be optimised in the nonlinear optimisation problem (Diehl *et al.*, 2005).

Using only the control moves in the parameter optimisation problem leads to a single integration of the state equations over the time interval $[0,T]$. If the time interval is long, the accuracy of the numerical integration is affected. The accuracy of the numerical derivatives

for a given perturbation size is affected even more. To compensate for this problem, an estimate of the state value x_k at each time node is made (represented by s_k) and the system dynamics are integrated between nodes. This method is called direct multiple shooting (Bock and Plitt, 1984, Hull, 1997). The nonlinear optimiser then removes any error between the estimate (s_k) and actual dynamics (x_k) through the equality constraints (3.53).

3.7 ROBUST NONLINEAR MODEL PREDICTIVE CONTROL

Plant models always differ from the real system owing to incomplete modelling, parameter uncertainty and unmodelled disturbances. In this section, the NMPC developed earlier in (Coetzee *et al.*, 2008) are robustified to parameter uncertainty, by approximating the worst-case realisations of the objective function and the constraint functions to the parameter variations (Coetzee *et al.*, 2009). The worst-case objective function is then minimised subject to the worst-case constraints. The approximated min-max optimisation problem is called an approximate robust counterpart formulation (Ben-Tal and Nemirovskii, 2001, 2002, Diehl *et al.*, 2006).

3.7.1 Parameter uncertainty description

Consider an uncertain parameter vector $p \in \mathbb{R}^{n_p}$ and nominal parameter vector $\tilde{p} \in \mathbb{R}^{n_p}$, which are assumed to be restricted to a generalised ball

$$\mathbb{P} = \{p \in \mathbb{R}^{n_p} \,|\, \|p - \tilde{p}\| \leq 1\} \tag{3.56}$$

defined by using a suitable norm $\|\cdot\|$ in $\mathbb{R}^{n_p}$ (Diehl *et al.*, 2006). A suitable norm may be the scaled Hölder q-norm ($1 \leq q \leq \infty$), $\|p\| = \left\|A^{-1}p\right\|_q$, with an invertible $A \in \mathbb{R}^{n_p \times n_p}$ matrix.

The Hölder q-norm is also suitable for describing box uncertainty where the upper p_u and lower p_l bounds on the parameters p are known:

$$\begin{aligned} \mathbb{P}_{\text{box}} &\triangleq \{p \in \mathbb{R}^{n_p} | p_l \leq p \leq p_u\}, \\ &= \left\{ p \in \mathbb{R}^{n_p} \left| \left\| \text{diag}\left(\frac{p_u - p_l}{2}\right)^{-1} \left(p - \frac{p_l + p_u}{2}\right) \right\|_\infty \leq 1 \right. \right\} \end{aligned} \tag{3.57}$$

where the centre of the box is defined as $\bar{p} \triangleq \frac{p_l + p_u}{2} \in \mathbb{R}^{n_p}$. In general the centre of the box $\bar{p}$ and the nominal parameter vector $\tilde{p}$ do not have to be the same point ($\tilde{p} \neq \bar{p}$).

The dual norm $\|\cdot\|_{\#}$ for the norm $\|\cdot\|$ is the mapping

$$\begin{aligned} \|\cdot\|_{\#} : \mathbb{R}^{(N-1)\cdot n_p} &\rightarrow \mathbb{R} \\ \mathbf{a} &\mapsto \|\mathbf{a}\|_{\#} \triangleq \max_{\mathbf{p}\in\mathbb{R}^{(N-1)\cdot n_p}} \mathbf{a}^T\mathbf{p} \text{ s.t. } \|\mathbf{p}\| \leq 1, \end{aligned} \tag{3.58}$$

and the dual norm $\|\cdot\|_{*}$ for the norm $\|\cdot\|$ is the mapping

$$\begin{aligned} \|\cdot\|_{*} : \mathbb{R}^{n_p} &\rightarrow \mathbb{R} \\ a &\mapsto \|a\|_{*} \triangleq \max_{p\in\mathbb{R}^{n_p}} a^T p \text{ s.t. } \|p\| \leq 1, \end{aligned} \tag{3.59}$$

It is well known that for any scaled Hölder q-norm $\|p\| = \|Ap\|_q$ (A an invertible matrix, $1 \leq q \leq \infty$), the dual norm is $\|a\|_{*} = \left\|A^{-1}a\right\|_{\frac{q}{q-1}}$ (for $q = 1$, define $\frac{q}{q-1} \triangleq \infty$ and for $q = \infty$, define $\frac{q}{q-1} \triangleq 1$) as observed in the context of the worst-case analysis by Ma and Braatz (2001) and independently by Bock and Kostina (2001).

The q-norm $\|\cdot\|_q : \mathbb{R}^{n_p} \longrightarrow \mathbb{R}$ for $1 \leq q \leq \infty$ is defined as

$$\|\mathbf{x}\|_q = \left\{ \sum_{i=1}^{n_p} |x_i|^q \right\}^{\frac{1}{q}} \tag{3.60}$$

and the case where $q = \infty$ becomes

$$\|\mathbf{x}\|_\infty = \max_{1\leq i\leq n_p} |x_i| . \tag{3.61}$$

3.7.2 Direct approximate robust counterpart formulation

To add uncertainty into an optimisation problem, a min-max optimisation can be done (Diehl *et al.*, 2006, Ma and Braatz, 2001). The worst-case value for the cost $\phi(\mathbf{s},\mathbf{q})$ is defined as

$$\psi(\mathbf{q}) \triangleq \max_{\mathbf{s},\mathbf{p}} \quad \phi(\mathbf{s},\mathbf{q}) \tag{3.62}$$

$$\text{s.t.} \quad g(\mathbf{s},\mathbf{q},\mathbf{p}) = 0 \tag{3.63}$$

and the worst-case values for the constraint functions $\theta_{i,j}(s_j,q_j)$ are defined as

$$\omega_{i,j}(q_j) \triangleq \max_{s_j,p_j} \quad \theta_{i,j}(s_j,q_j) \tag{3.64}$$

$$\text{s.t.} \quad g_{\tau_s}(s_j,q_j,p_j) = 0, \tag{3.65}$$

where $g_{\tau_s} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \rightarrow \mathbb{R}^{n_x}$ is the system dynamic for one time step τ_s, $p_i \in \mathbb{P}_{box}$, $i = 1,\ldots N-1$ are defined as the unknown time varying model parameters and $\mathbf{p} \triangleq (p_1,\ldots,p_{N-1}) \in \mathbb{P}_{box}^{(N-1)}$ is defined as the sequence of time varying model parameters. The worst-case cost and constraint functions are calculated by maximising the cost function and constraint func-

tions with regard to the model parameter sequence $\mathbf{p} \in \mathbb{P}_{box}^{(N-1)}$ and state values $\mathbf{s} \in \mathbb{R}^{(N \cdot n_x)}$. The worst-case cost function $\psi(\mathbf{q})$ is then minimised by choosing the control moves $\mathbf{q} \triangleq (q_1, q_1, \ldots, q_{N-1}) \in \mathbb{R}^{((N-1)\cdot n_u)}$ subject to the worst-case constraints $\omega_{i,j}(q_j)$

$$\min_{\mathbf{q}} \quad \psi(\mathbf{q}) \tag{3.66}$$

$$\text{s.t.} \quad \omega_{i,j}(q_j) \leq 0, \quad \begin{matrix} i = 1, \ldots, n_c, \\ j = 1, \ldots, N-1. \end{matrix} \tag{3.67}$$

This min-max optimisation problem is difficult to solve for general nonlinear systems. The optimisation problem can, however, be simplified by approximating the worst-case calculations for the cost $\tilde{\psi}(\mathbf{q})$ and the constraints $\tilde{\omega}_{i,j}(q_j)$. The approximate robust counterpart formulation is given by

$$\min_{\mathbf{q}} \quad \tilde{\psi}(\mathbf{q}) \tag{3.68}$$

$$\text{s.t.} \quad \widetilde{\omega}_{i,j}(q_j) \leq 0, \quad \begin{matrix} i = 1, \ldots, n_c, \\ j = 1, \ldots, N-1, \end{matrix} \tag{3.69}$$

where the approximation of the worst-case cost $\tilde{\psi}(\mathbf{q})$ and constraints $\tilde{\omega}_{i,j}(\mathbf{q})$ can be done through linearisation of the system dynamics $g(\mathbf{s}, \mathbf{q}, \mathbf{p}) = 0$ and the cost $\phi(\mathbf{s}, \mathbf{q})$ and constraint $\theta_{i,j}(s_j, q_j)$ functions. The approximation of the worst-case cost, $\psi(\mathbf{q})$ by $\tilde{\psi}(\mathbf{s}, \mathbf{q})$, is defined by a convex optimisation problem

$$\max_{\Delta\mathbf{s}, \Delta\mathbf{p}} \quad \phi(\mathbf{s}, \mathbf{q}) + \frac{\partial \phi}{\partial \mathbf{s}}(\mathbf{s}, \mathbf{q}) \Delta\mathbf{s} \tag{3.70}$$

$$\text{s.t.} \quad \frac{\partial g}{\partial \mathbf{s}}(\mathbf{s}, \mathbf{q}, \bar{\mathbf{p}}) \Delta\mathbf{s} + \frac{\partial g}{\partial \mathbf{p}}(\mathbf{s}, \mathbf{q}, \bar{\mathbf{p}}) \Delta\mathbf{p} = 0, \tag{3.71}$$

$$\|\Delta p_i\| \leq 1, \, i = 1, \ldots, N-1, \tag{3.72}$$

and the approximation of the worst-case constraints $\omega_{i,j}(q_j)$ by $\tilde{\omega}_{i,j}(\mathbf{s}, \mathbf{q})$ are defined as

$$\max_{\Delta\mathbf{s}, \Delta\mathbf{p}} \quad \theta_{i,j}(s_j, q_j) + \frac{\partial \theta_{i,j}}{\partial \mathbf{s}}(s_j, q_j) \Delta\mathbf{s} \tag{3.73}$$

$$\text{s.t.} \quad \frac{\partial g}{\partial \mathbf{s}}(\mathbf{s}, \mathbf{q}, \bar{\mathbf{p}}) \Delta\mathbf{s} + \frac{\partial g}{\partial \mathbf{p}}(\mathbf{s}, \mathbf{q}, \bar{\mathbf{p}}) \Delta\mathbf{p} = 0, \tag{3.74}$$

$$\left\|\Delta p_j\right\| \leq 1, \; j = 1, \ldots, N-1 \tag{3.75}$$

where $\Delta\mathbf{s} \in \mathbb{R}^{(N \cdot n_x)}$, $\bar{\mathbf{p}} \triangleq (\bar{p}, \ldots, \bar{p}) \in \mathbb{R}^{(N-1)\cdot n_p}$ is a sequence of parameters at the centre of the box, $\Delta p_j \triangleq p_j - \bar{p} \in \mathbb{R}^{n_p}$ is the deviation of the model parameters from the centre of the box and $\Delta\mathbf{p} \triangleq (\Delta p_1, \ldots, \Delta p_{N-1}) \in \mathbb{R}^{(N-1)\cdot n_p}$ is defined as the sequence of model parameter deviations.

The optimisation problems (3.70)-(3.75) have analytical solutions. The approximation for the worst-case cost can be expressed as

$$\tilde{\psi}(\mathbf{s},\mathbf{q}) = \phi(\mathbf{s},\mathbf{q}) + \left\| -\left(\frac{\partial g}{\partial \mathbf{p}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^T \left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^{-T} \left(\frac{\partial \phi}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q})\right)^T \right\|_{\#} \tag{3.76}$$

and the approximation of the worst-case constraints can be expressed as

$$\tilde{\omega}_{i,j}(\mathbf{s},\mathbf{q}) = \theta_{i,j}(s_j,q_j) + \left\| -\left(\frac{\partial g}{\partial \mathbf{p}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^T \left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^{-T} \left(\frac{\partial \theta_{i,j}}{\partial \mathbf{s}}(s_j,q_j)\right)^T \right\|_{\#} \tag{3.77}$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$ as described in (3.59). For notational convenience the uncertainty term of (3.76) is defined as

$$\Delta\phi(\mathbf{s},\mathbf{q}) \triangleq \left\| -\left(\frac{\partial g}{\partial \mathbf{p}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^T \left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^{-T} \left(\frac{\partial \phi}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q})\right)^T \right\|_{\#} \tag{3.78}$$

and the uncertainty term for (3.77) is defined as

$$\Delta\theta_{i,j}(\mathbf{s},\mathbf{q}) \triangleq \left\| -\left(\frac{\partial g}{\partial \mathbf{p}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^T \left(\frac{\partial g}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})\right)^{-T} \left(\frac{\partial \theta_{i,j}}{\partial \mathbf{s}}(s_j,q_j)\right)^T \right\|_{\#} \tag{3.79}$$

The approximate robust counterpart of (3.68)-(3.69) can now be expressed as

$$\min_{\mathbf{s},\mathbf{q}} \quad \phi(\mathbf{s},\mathbf{q}) + \Delta\phi(\mathbf{s},\mathbf{q}) \tag{3.80}$$

$$\text{s.t.} \quad \theta_{i,j}(s_j,q_j) + \Delta\theta_{i,j}(\mathbf{s},\mathbf{q}) \leq 0, \quad \begin{matrix} i = 1,\ldots,n_c, \\ j = 1,\ldots,N-1, \end{matrix} \tag{3.81}$$

$$g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}}) = 0 \tag{3.82}$$

In order to solve this approximate min-max optimisation problem (3.68)-(3.69) efficiently, new optimisation variables $\mathbf{D} \in \mathbb{R}^{(N\cdot n_x)\times n_p}$ are defined (Diehl *et al.*, 2006) in the form of a sensitivity matrix $\mathbf{D} \triangleq (D_1, D_2, \ldots, D_N) = \left(-\left(\frac{\partial g}{\partial \mathbf{s}}(\bar{\mathbf{s}},\mathbf{q},\bar{\mathbf{p}})\right)^{-1} \frac{\partial g}{\partial \mathbf{p}}(\bar{\mathbf{s}},\mathbf{q},\bar{\mathbf{p}})\right)$ to prevent the explicit calculation of the matrix inverse in (3.78) and (3.79), which would reduce the sparsity of the problem. The direct approximate robust counterpart formulation becomes

$$\min_{\mathbf{s},\mathbf{q},\mathbf{D}} \quad \phi(\mathbf{s},\mathbf{q}) + \left\| \mathbf{D}^T \left(\frac{\partial \phi}{\partial \mathbf{s}}(\mathbf{s},\mathbf{q})\right)^T \right\|_{\#} \tag{3.83}$$

$$\text{s.t.} \quad g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}}) = 0, \tag{3.84}$$

$$\frac{\partial g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})}{\partial \mathbf{s}}\mathbf{D} + \frac{\partial g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})}{\partial \mathbf{p}} = 0, \tag{3.85}$$

$$\theta_{i,j}(s_j,q_j) + \left\| D_j^T \left(\frac{\partial \theta_{i,j}}{\partial s_j}(s_j,q_j)\right)^T \right\|_* \leq 0, \quad \begin{matrix} i = 1,\ldots,n_c, \\ j = 1,\ldots,N-1, \end{matrix} \tag{3.86}$$

$$\theta_N(s_N) \leq 0. \tag{3.87}$$

In order to maintain smooth objective and constraint functions in the optimisation problem, slack variables $\delta \triangleq (\delta_{0,1}, \dots \delta_{n_c,1}, \dots, \delta_{n_c,N-1})$, $\delta_{i,j} \in \mathbb{R}$, $i = 0, \dots, n_c$, $j = 1, \dots, N-1$ are introduced to replace the dual norms in the optimisation problem (3.83)-(3.87) (Diehl *et al.*, 2006).

The nonlinear parameter optimisation problem that is solved at each time step is given by

$$\min_{\mathbf{s},\mathbf{q},\mathbf{D},\delta} \quad \phi(\mathbf{s},\mathbf{q}) + e^T\delta_{0,1} + \cdots + e^T\delta_{0,N-1} \tag{3.88}$$

$$\text{s.t.} \quad g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}}) = 0, \tag{3.89}$$

$$\frac{\partial g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})}{\partial \mathbf{s}}\mathbf{D} + \frac{\partial g(\mathbf{s},\mathbf{q},\bar{\mathbf{p}})}{\partial \mathbf{p}} = 0, \tag{3.90}$$

$$-\delta_{0,j} \le D_j^T \left(\frac{\partial \phi}{\partial s_j}(\mathbf{s},\mathbf{q})\right)^T \le \delta_{0,j}, \quad j = 1, \dots, N, \tag{3.91}$$

$$\theta_{i,j}(s_j,q_j) + e^T\delta_{i,j} \le 0, \quad \begin{matrix} i = 1,\dots,n_c, \\ j = 1,\dots,N-1, \end{matrix} \tag{3.92}$$

$$-\delta_{i,j} \le D_j^T \left(\frac{\partial \theta_{i,j}}{\partial s_j}(s_j,q_j)\right)^T \le \delta_{i,j}, \quad \begin{matrix} i = 1,\dots,n_c, \\ j = 1,\dots,N-1, \end{matrix} \tag{3.93}$$

$$\theta_N(s_N) \le 0. \tag{3.94}$$

where $e \triangleq (1, \dots, 1) \in \mathbb{R}^{n_p}$ (Diehl *et al.*, 2006, 2005).

3.7.3 RNMPC implementation

For implementation of the RNMPC, the scalar performance function and the discrete time system dynamics function need to be defined. The scalar performance function is defined as

$$\phi(\mathbf{s},\mathbf{q}) \triangleq \sum_{i=1}^{N-1} L_i(s_i,q_i) + E(s_N) \tag{3.95}$$

where the scalar interval performance indexes and the terminal performance index are defined as quadratic functions

$$L_i(s_i,q_i) \triangleq h(s_i,q_i)^T Q h(s_i,q_i) + \Delta q_i^T R \Delta q_i \tag{3.96}$$

$$E(s_N) \triangleq s_N^T P s_N, \tag{3.97}$$

where Q and R represent the weighting matrices on the outputs and controls respectively, $h : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_y}$ is the function that maps the current state and control vector to the output vector using nominal model parameters, $\Delta q_i \triangleq q_i - q_{i-1}$ and P is the terminal cost weighting matrix.

The equality constraint function describing the discrete time system dynamics is defined as

$$g(\mathbf{s},\mathbf{q},\mathbf{p}) \triangleq \begin{cases} x_k - s_1, \\ f_1(s_1,q_1,p_1) - s_2, \\ \vdots \\ f_{N-1}(s_{N-1},q_{N-1},p_{N-1}) - s_N. \end{cases} \tag{3.98}$$

where $f_i(s_i,q_i,p_i)$, $i = 1,\ldots,N-1$ are defined as the function values at the nodes for the continuous time dynamics $\dot{x} = f_c(x(t),u(t),p(t))$. The interpolation between nodes is defined as integrating the state equations for one sample time τ_s

$$\begin{aligned} x_{k+1} &\triangleq f_k(x_k,u_k,p_k) \\ &\triangleq \int_{t_k}^{t_k+\tau_s=t_{k+1}} f_c(x,u_k,p_k)\,dt \end{aligned} \tag{3.99}$$

with initial conditions $x(t_k) \triangleq x_k = s_k$ and the controls $u_k = q_k$ constant over the interval $[t_k,t_{k+1}]$. The integration of the state equations can be done with a software package called CPPAD (Lougee-Heimer, 2003). CPPAD is also capable of calculating sensitivities of the integral. This is useful for calculating the sensitivities of the system dynamics $f_k(x_k,u_k,p_k)$ with regard to states s_k, inputs q_k and parameters p_k. It is also capable of calculating second order derivatives of the integral. CPPAD does automatic differentiation of specially modified C++ code. It works by recording all the mathematical operations being done in the desired function and then uses the chain rule of differentiation to calculate the derivatives. The advantages of automatic differentiation are that it is fast to calculate and does not suffer from truncation errors present in other numerical methods. CPPAD also contains a module to solve ordinary differential equations (ODEs) with the Runge-Kutta method that is required to integrate the state equations. The derivatives of the integral are then calculated by automatic differentiation.

Nonlinear optimisation software is required to solve the nonlinear parameter optimisation problem stated in (3.88)-(3.94). The software used is a package called IPOPT (Kawajir *et al.*, 2006), which is useful when solving large-scale sparse nonlinear optimisation problems.

IPOPT requires the following functions to be provided:

- Scalar performance function value $\varphi(X)$
- Gradient of the performance $\nabla\varphi(X)$
- Constraint functions values $\vartheta(X)$
- Jacobian of the constraints $\nabla\vartheta(X)$

where X is the vector of all decision variables.

For the problem specified in (3.88)-(3.94), the vector of decision variables is defined as

$$X \triangleq (s_1, q_1, D_1, \delta_{0,1}, \ldots, \delta_{n_c,1}, \ldots, s_{N-1}, q_{N-1}, D_{N-1}, \delta_{0,N-1}, \ldots, \delta_{n_c,N-1}, s_N, D_N) \quad (3.100)$$

where $X \in \mathbb{R}^{((n_x+n_u+n_D+n_{slack})\cdot(N)+n_x+n_D)}$, $n_D \triangleq n_x \times N$ and $n_{slack} \triangleq (n_c+1) \times n_p$.

All matrix variables in X are unrolled into vectors to simplify implementation. The matrix is unrolled by placing the rows of the matrix sequentially to form a vector.

The scalar performance function is defined in terms of the decision variables X as

$$\varphi(X) \triangleq \sum_{i=0}^{N-1} \left(L_i(s_i, q_i) + e^T \delta_{0,i}\right) + E(s_N) \quad (3.101)$$

where the scalar interval performance indexes and the terminal performance index are the same as in (3.96) and (3.97).

The gradient of the performance function $\nabla\varphi(X) \in \mathbb{R}^{((n_x+n_u+n_D+n_{slack})\cdot N+n_x+n_D)}$ then becomes

$$\nabla\varphi(X) = \begin{pmatrix} 2Qs_1 \\ 2Rq_1 \\ 0_{n_D} \\ 1_{n_p} \\ 0_{n_c \cdot n_p} \\ \vdots \\ 2Qs_{N-1} \\ 2Rq_{N-1} \\ 0_{n_D} \\ 1_{n_p} \\ 0_{n_c \cdot n_p} \\ 2Ps_N \\ 0_{n_D} \end{pmatrix}. \quad (3.102)$$

where $0_n \triangleq (0, \ldots, 0) \in \mathbb{R}^{n_x}$ and $1_n \triangleq (1, \ldots, 1) \in \mathbb{R}^{n_x}$.

The values of the constraint functions $\vartheta(X)$ are interleaved equality and inequality constraint functions with the same decision variables that will result in a Jacobian of the constraints $\nabla\vartheta(X)$ that is sparse and banded. For RNMPC, the following structure for the constraints

function $\vartheta(X) \in \mathbb{R}^{((n_x+n_D+n_c+2\cdot n_{slack})\cdot N+n_x+n_D)}$ is defined

$$\vartheta(X) \triangleq \begin{pmatrix} \vartheta_1(X) \\ \vartheta_2(X) \\ \vdots \\ \vartheta_{N-1}(X) \\ \vartheta_N(X) \end{pmatrix} \tag{3.103}$$

where

$$\vartheta_1(X) \triangleq \begin{pmatrix} x_k - s_1 \\ -I_{n_x} D_2 + 0 \\ \theta_{1,1}(s_1,q_1) + e^T \delta_{1,1} \\ \vdots \\ \theta_{n_c,1}(s_1,q_1) + e^T \delta_{n_c,1} \\ D_2^T \left(\frac{\partial L_1}{\partial s_1}(s_1,q_1)\right)^T - \delta_{0,1} \\ D_2^T \left(\frac{\partial L_1}{\partial s_1}(s_1,q_1)\right)^T + \delta_{0,1} \\ D_2^T \left(\frac{\partial \theta_{1,1}}{\partial s_1}(s_1,q_1)\right)^T - \delta_{1,1} \\ D_2^T \left(\frac{\partial \theta_{1,1}}{\partial s_1}(s_1,q_1)\right)^T + \delta_{1,1} \\ \vdots \\ D_2^T \left(\frac{\partial \theta_{n_c,1}}{\partial s_1}(s_1,q_1)\right)^T - \delta_{n_c,1} \\ D_2^T \left(\frac{\partial \theta_{n_c,1}}{\partial s_1}(s_1,q_1)\right)^T + \delta_{n_c,1} \end{pmatrix}, \tag{3.104}$$

$$\vartheta_i(X) \triangleq \begin{pmatrix} f_{i-1}(s_{i-1},q_{i-1},\bar{p}) - s_i \\ \left(\begin{array}{c} \frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial s_{i-1}} \cdot D_{i-1} - \\ I_{n_x} \cdot D_i + \frac{\partial f_{i-1}(s_{i-1},q_{i-1},\bar{p})}{\partial p} \end{array}\right) \\ \theta_{1,i}(s_i,q_i) + e^T \delta_{1,i} \\ \vdots \\ \theta_{n_c,i}(s_i,q_i) + e^T \delta_{n_c,i} \\ D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T - \delta_{0,i} \\ D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T - \delta_{1,i} \\ \vdots \\ D_{i+1}^T \left(\frac{\partial \theta_{n_c,i}}{\partial s_i}(s_i,q_i)\right)^T - \delta_{n_c,i} \\ D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T + \delta_{0,i} \\ D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T + \delta_{1,i} \\ \vdots \\ D_{i+1}^T \left(\frac{\partial \theta_{n_c,i}}{\partial s_i}(s_i,q_i)\right)^T + \delta_{n_c,i} \end{pmatrix}, \forall i = 2,\ldots,N-1, \tag{3.105}$$

$$\vartheta_N(X) \triangleq \begin{pmatrix} f_{N-1}(s_{N-1},q_{N-1},\bar{p}) - s_N \\ \left(\begin{array}{c} \frac{\partial f_{N-1}(s_{N-1},q_{N-1},\bar{p})}{\partial s_{N-1}} \cdot D_{N-1} - \\ I_{n_x} \cdot D_N + \frac{\partial f_{N-1}(s_{N-1},q_{N-1},\bar{p})}{\partial p} \end{array}\right) \end{pmatrix}. \tag{3.106}$$

which results in a Jacobian of the constraints function $\nabla\vartheta(X) \in \mathbb{R}^{n_{\text{jac\_row}} \times n_{\text{jac\_col}}}$ where $n_{\text{jac\_row}} \triangleq ((n_x + n_D + n_c + 2 \cdot n_{slack}) \cdot N + n_x + n_D)$ and $n_{\text{jac\_col}} \triangleq ((n_x + n_u + n_D + n_{slack}) \cdot N + n_x + n_D)$, with a block structure defined in Figure 3.1. The blocks lie on the diagonal of the Jacobian matrix

$$\nabla\vartheta(X) = \begin{bmatrix} \nabla\vartheta_1(X) & 0 & 0 & 0 \\ 0 & \nabla\vartheta_2(X) & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \nabla\vartheta_{N-1}(X) \end{bmatrix} \tag{3.107}$$

where the first 2×3 entries of each block are the same entries as the last 2×3 entries of the previous block.

IPOPT only requires the non-zero entries to be populated for this sparse Jacobian matrix. The number of non-zero entries for this structure is defined as $n_{nz} \triangleq N(n_x + n_x(n_x + n_u)) + N(n_D(2n_x + n_u + 1)) + N(n_c(n_x + 2n_u + N)) + 2N(n_{slack}(2n_x + 2n_u + 1)) + n_x + n_D - n_c \cdot n_u - 2 \cdot n_{slack} \cdot n_u$

All the sensitivities and second-order derivatives as shown in the constraints (3.104)-(3.106) and Jacobian of the constraints (Figure 3.1) are also calculated by CPPAD (Lougee-Heimer, 2003).

$$\nabla \vartheta_i(X) = \begin{bmatrix}
-\mathbb{I}_{n_x} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & -\mathbb{I}_{n_D} & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
\frac{\partial \theta_{1,i}(s_i,q_i)}{\partial s_i} & 0 & \frac{\partial \theta_{1,i}(s_i,q_i)}{\partial q_i} & 0_{n_p}^T & 0_{n_p}^T & \cdots & 0_{n_p}^T & 0 & 0 & 0 & 0 \\
\frac{\partial \theta_{2,i}(s_i,q_i)}{\partial s_i} & 0 & \frac{\partial \theta_{2,i}(s_i,q_i)}{\partial q_i} & 0_{n_p}^T & 1_{n_p}^T & \cdots & 0_{n_p}^T & 0 & 0 & 0 & 0 \\
\vdots & 0 & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial \theta_{nc,i}(s_i,q_i)}{\partial s_i} & 0 & \frac{\partial \theta_{nc,0}(s_i,q_i)}{\partial q_i} & 0_{n_p}^T & 0_{n_p}^T & \cdots & 1_{n_p}^T & 0 & 0 & 0 & 0 \\
\frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & -\mathbb{I}_{n_p} & 0 & 0 & 0 & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & 0 & -\mathbb{I}_{n_p} & 0 & \cdots & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & 0 & 0 & 0 & -\mathbb{I}_{n_p} & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & \mathbb{I}_{n_p} & 0 & 0 & 0 & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial L_i}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & 0 & \mathbb{I}_{n_p} & 0 & 0 & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{1,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial s_i} & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial q_i} & 0 & 0 & 0 & \mathbb{I}_{n_p} & 0 & 0 & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} & \frac{\partial D_{i+1}^T \left(\frac{\partial \theta_{nc,i}}{\partial s_i}(s_i,q_i)\right)^T}{\partial D_{i+1}} \\
\frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} & 0 & \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial q_i} & 0 & 0 & 0 & 0 & -\mathbb{I}_{n_x} & 0 & 0 & 0 \\
\left(\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial s_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial s_i}\right) & 0 & \left(\frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial q_i} + \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial p}}{\partial q_i}\right) & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\mathbb{I}_{n_D} \\
 & & & & & & \frac{\partial \frac{\partial f_i(s_i,q_i,\bar{p})}{\partial s_i} \cdot D_i}{\partial D_i} & & & &
\end{bmatrix} \tag{3.108}$$

Figure 3.1: Jacobian block.

3.8 STATE OBSERVERS

Some mathematical models of plants use state-space descriptions. State-space description of plant make practical control difficult, because it is not usually possible to measure all the states in the model and some sort of reconstruction of the states from the available measurements is needed. An *observer* can be constructed from a mathematical model of the plant to estimate the states from the available measurements (Marquez and Riaz, 2005). The milling circuit is described mathematically by nonlinear state-space models in Section 2.3.4. Observers were not used in this thesis and only a quick overview is provided here for completeness.

State observers for nonlinear system are more challenging than for linear systems, because the separation principle does not apply in general for nonlinear system (Freeman, 1995), where the separation principle states that a stable controller and a stable observer can be designed independently for a plant and the combined closed-loop system will be stable. Atassi and Khalil (1999, 2000) showed that high-gain observers can be constructed for certain classes of nonlinear systems that maintain the separation principle.

Some possible observers that can be used for nonlinear systems with robustness to model mismatches are input-output observers (Marquez and Riaz, 2005), robust H_∞ observers based on LMI for nonlinear systems with time-varying uncertainties (Abbaszadeh and Marquez, 2009), sliding mode observers (Bartolini *et al.*, 2003, Davila *et al.*, 2005, Spurgeon, 2008, Xiong and Saif, 2001), high gain observers (Atassi and Khalil, 2000) and moving horizon estimators or moving horizon state observers (Chu *et al.*, 2007, Michalska and Mayne, 1995, Rao *et al.*, 2003).

Sliding mode observers have received much attention recently, because: "*Sliding mode observers have unique properties, in that the ability to generate a sliding motion on the error between the measured plant output and the output of the observer ensures that a sliding mode observer produces a set of state estimates that are precisely commensurate with the actual output of the plant. It is also the case that analysis of the average value of the applied observer injection signal, the so-called equivalent injection signal, contains useful information about the mismatch between the model used to define the observer and the actual plant. These unique properties, coupled with the fact that the discontinuous injection signals which were perceived as problematic for many control applications have no disadvantages for software-based observer frameworks, have generated a ground swell of interest in sliding mode observer methods in recent years.*" (Spurgeon, 2008) and "*For both relatively general non-linear system representations and also application specific models with significant non-linearity sliding mode observers are seen to be at the forefront of robust techniques for state and parameter estimation.*" (Spurgeon, 2008)

Moving horizon estimators (MHE) or moving horizon state observers (MHSO) provide a very close parallel to MPC controllers (Chu *et al.*, 2007). It uses a moving window of

previous measurements to obtain an estimate of the current state of the system (Rao *et al.*, 2003). It can handle nonlinear systems and inequality constraints on the decision variables explicitly (Rao *et al.*, 2003). Recently, Chu *et al.* (2007) proposed a refinement to MHSOs that allow MHSOs to handle model uncertainty in addition to external disturbances in a robust manner to produce a robust moving horizon state observer (RMHSO). The RMHSO can possibly be combined with the RNMPC of Section 3.7 to obtain an output-feedback RNMPC (Michalska and Mayne, 1995).

3.9 CONCLUSION

This chapter outlines the development of MPC and stability theory for MPC. The stability theory focuses on the requirements for exponential stability, while some control formulations are based on other stability formulations such as Lyapunov, asymptotic, ISS and ISpS. It outlines the RNMPC theory (Section 3.7) that will be applied to the nonlinear model presented in Section 2.3.4. Simulation results of the RNMPC applied to the nonlinear milling circuit model are provided in Chapter 5 for different operational conditions.

CHAPTER 4

PID CONTROL

The PI controllers presented in this chapter serve only as an example implementation to provide a baseline for comparison with RNMPC and NMPC. Single-loop PI controllers without a MIMO compensator or a centralised design were used, because more than 60% of all respondents still use PI controllers, according to a recent survey by Wei and Craig (2009), usually single-loop PI controllers. A decentralised PID controller design that takes interaction into account was attempted in Addendum C.

The PI controllers presented here are not intended to serve as the best PI controller design for the presented milling circuit based on an exhaustive study, because that was not the main focus of this thesis. The comparison of the RNMPC and NMPC controllers to the PI controllers should, therefore, not be seen as a definitive, but rather serve as an example of possible benefits that RNMPC can provide over PI control typically employed in industry (Wei and Craig, 2009), when large feed disturbances are common in the milling circuit.

4.1 INTRODUCTION

PID control is a fundamental feedback control method employed broadly in process control. PID control usually forms the lowest level of control with multi-variable controllers such as MPC providing the setpoints for the PID controllers. PID control can be described in the time domain by the following algorithm

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(t)d\tau + T_d\frac{de(t)}{dt}\right) \tag{4.1}$$

where the error is defined as the difference between the plant output and the setpoint ($e(t) := y(t) - r(t)$). The transfer function form of the PID controller is given by

$$\frac{U(s)}{Y(s)} = C_{parallel}(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right) \tag{4.2}$$

where K is the gain, T_i the integral time and T_d the derivative time. The PID form in (4.2) is known as the parallel ("ideal", "non-interacting") form. The tuning rules based on the IMC principle developed by Skogestad (2003) uses the series (cascading, "interacting") form given by

$$\frac{U(s)}{Y(s)} = C_{series}(s) = K_c \left(\frac{\tau_I s + 1}{\tau_I s} \right) \cdot (\tau_D s + 1) \tag{4.3}$$

where K_c is the proportional gain, τ_I is the integral time and τ_D the derivative time for the series form of the controller. The parameters from the series form can be converted to the parallel form by

$$\begin{aligned} K &= K_c \left(1 + \frac{\tau_D}{\tau_I} \right) \\ T_i &= \tau_I \left(1 + \frac{\tau_D}{\tau_I} \right) \\ T_d &= \frac{\tau_D}{1 + \frac{\tau_D}{\tau_I}} \end{aligned} \tag{4.4}$$

The two transfer functions (equation (4.2) and (4.3)) describing the PID controller are improper. If the closed-loop transfer function is also improper, a suitable order filtering term should be added to the denominator to produce a proper closed-loop transfer function.

4.2 PI CONTROL WITH ANTI-WINDUP

The derivative term in PID control allows the controller to react quickly to sudden changes in the process. The derivative term of the PID controller acts on process noise as if the process is changing rapidly, which causes undesirable closed-loop behaviour. Industry therefore usually only employs PI control rather than full PID control, because the derivative term is so sensitive to noise. The rest of this chapter will therefore focus only on PI control.

Windup in PI/PID control is when there is saturation of the control action that prevents the control error from reaching zero. This will cause the integrator value to keep on increasing/decreasing in an effort to eliminate the control error. If the plant output passes the setpoint value, the sign of the error will change, but the integrator value has to wind down before normal operation can resume. Anti-windup therefore forces the integrator input to zero when saturation occurs to prevent it from winding up (Åström, 2002).

In Figure 4.4 the input to the integrator is the control error multiplied by the integrator gain

$$\frac{K}{T_i} e$$

where e is the control error, but if the error cannot vanish due to saturation on the control, the integrator value will keep increasing. To prevent windup, a second control loop is added

to the integrator input (Åström, 2002)

$$\frac{K}{T_i}e + \frac{1}{Tt}e_s$$

where e_s is the error between the desired control action v and the saturated control action u and T_t is the tracking time constant that controls how fast the controller resets after saturation. When the control action u saturates, the error e_S equals the control error e

$$e_s = -\frac{KT_t}{T_i}e \tag{4.5}$$

that results in the desired control action value that settles at

$$v = u_{\text{lim}} + \frac{KT_t}{T_i}e \tag{4.6}$$

which is slightly higher than the saturation value and prevents the integrator from winding up. The smaller T_t is, the faster the integrator resets and the quicker the controller can react to a change in error. The tracking time constraint should ideally be chosen to be larger than T_d and smaller than T_i and as a rule of thumb can be chosen to be $T_t = \sqrt{T_i T_d}$ (Åström, 2002).

Implementing anti-windup for PID is defined for the parallel form and the controller structure is shown in Figure 4.4.

4.3 LINEARISED MODELS FOR SIMC TUNING METHOD

Linearised models are necessary to design the PI controllers using the SIMC method. Linearised models can be created by performing step tests on the nonlinear model and performing system identification (SID) on the step responses.

The output-input pairings for single-loop controllers on multivariable systems are very important, because the output should be paired with the input that has the most influence on that output and the input should have the least interaction with other outputs. The traditional output-input pairings used on milling circuits are LOAD-MFS, PSE-SFW and SLEV-CFF (Chen *et al.*, 2007*b*, Conradie and Aldrich, 2001, Lynch, 1979, Napier-Munn and Wills, 2006). This pairing is not without its problems when used on industrial plants, as described by Chen *et al.* (2007*b*): *"Decoupled PID control had been frequently interrupted by changes in mineral ore hardness, feed rate, feed particle size, etc., ..."* This statement was supported by preliminary simulations using the above-mentioned pairing where the sump would either overflow or underflow as soon as ore hardness and composition disturbances were introduced. Craig and MacLeod (1996) also found SLEV control to be the most problematic aspect of controlling the milling circuit.

An alternative output-input pairing was then investigated that paired LOAD-MFS, PSE-CFF and SLEV-SFW (Smith *et al.*, 2001). The pairing of SLEV-SFW, rather than PSE-SFW and SLEV-CFF, was traditionally used only when a fixed speed sump discharge pump was available (Lynch, 1979). The pairing LOAD-MFS, PSE-CFF and SLEV-SFW, however, shows much better robustness to the feed disturbances subject to actuator limitations, as shown later in Section 5.3.2 and Addendum B.

The single-loop PI controllers are designed with the above-mentioned output-input pairings. The interactions between loops are ignored, because they cannot be included in the PI controller design using the SIMC tuning method, unlike other methods (Pomerleau *et al.*, 2000). The PI controllers are SISO controllers and the three controlled variables (PSE, LOAD and SLEV) will be independently controlled by three independent PI loops. Neither a multivariable compensator (Vázquez and Morilla, 2002) nor a centralised design (Morilla *et al.*, 2008) will be used for the PI controllers, because most plants that use PI controllers employ only single-loop PI controllers (Wei and Craig, 2009). An attempt at decentralised PID tuning that takes interactions into account was made in Addendum C.

4.3.1 PSE – CFF model

PSE exhibits a non-minimum phase first order response with time-delay to a change in CFF. A first order transfer function model was fitted to the step response data that has the following form:

$$G_{\text{PSE-CFF}}(s) = K_{PC}\frac{(1+Z_{PC}s)}{(1+P_{PC}s)}e^{(-\theta_{PC}s)} \tag{4.7}$$

$$G_{\text{PSE-CFF}}(s) = -0.00035\frac{(1-0.63s)}{(1+0.54s)}e^{(-0.011s)} \tag{4.8}$$

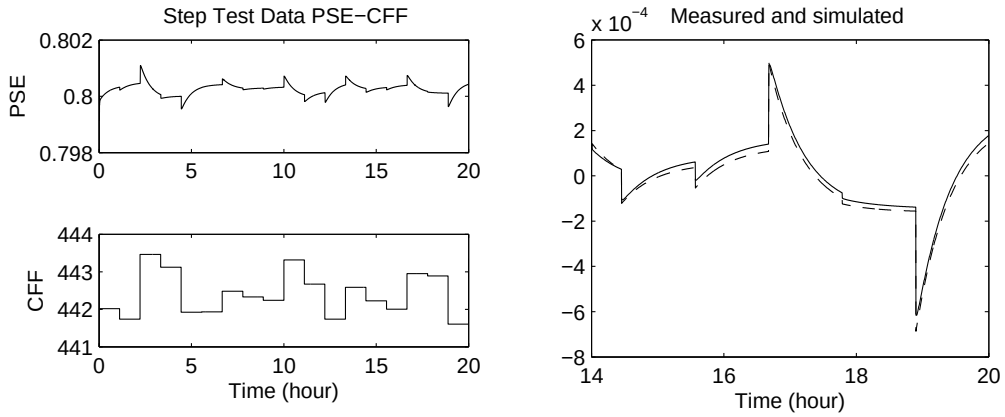
The model form was changed to include a zero in order to improve the fit of the linear model to the step response data of the nonlinear model. The step response data for the model fitting as well as the comparison between the linear and nonlinear models are shown in Figure 4.1. The linear model for PSE-CFF shows good agreement with the nonlinear model response.

4.3.2 LOAD – MFS model

The mill load volume exhibits an integrating response to the feed-rate of ore. An integrator transfer function model is fitted to the step test data of the nonlinear model with the following form:

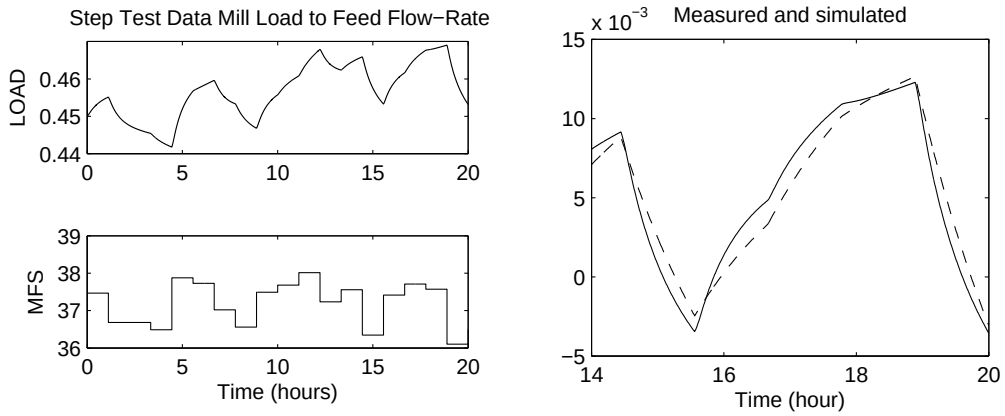
$$G_{\text{LOAD-MFS}}(s) = \frac{K_{LF}}{s} \tag{4.9}$$

$$G_{\text{LOAD-MFS}}(s) = \frac{0.01}{s} \tag{4.10}$$



(a) PSE vs CFF (b) Nonlinear vs Linear model

Figure 4.1: The change in PSE with a step change in CFF. The nonlinear (solid line) model is compared to the linear model (dashed line).



(a) LOAD vs MFS (b) Nonlinear vs Linear Model

Figure 4.2: The change in LOAD with a step change in MFS. The nonlinear (solid line) model is compared to the linear model (dashed line).

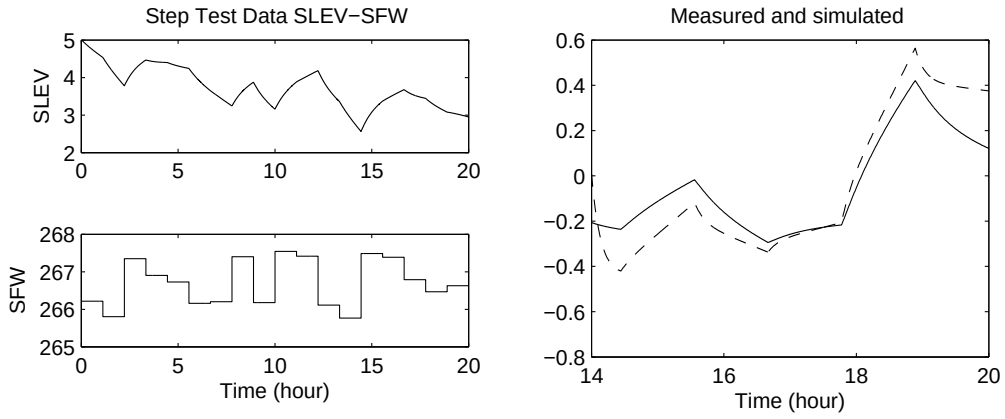
The step response data for the model fitting, as well as the comparison between the linear and nonlinear models, are shown in Figure 4.2.The linear model for PSE-CFF shows good agreement with the nonlinear model response.

4.3.3 SLEV – SFW model

The sump level exhibits an integrating response to the flow-rate of water added to the sump. An integrator transfer function model is fitted to the step test data of the nonlinear model with the following form:

$$G_{\text{SLEV-SFW}}(s) = \frac{K_{SW}}{s} \tag{4.11}$$

$$G_{\text{SLEV-SFW}}(s) = \frac{0.42}{s} \tag{4.12}$$



(a) SLEV vs SFW (b) Nonlinear vs Linear Model

Figure 4.3: The change in SLEV with a step change in SFW. The nonlinear (solid line) model is compared to the linear model (dashed line).

The step response data for the model fitting, as well as the comparison between the linear and nonlinear models, are shown in Figure 4.3. The linear model for PSE-CFF shows good agreement with the nonlinear model response.

4.4 SIMC TUNING METHOD

SIMC is a model-based tuning method with only a single tuning parameter for the closed-loop response. It is based on the IMC method. The PID parameter settings for the SIMC-PID method (Skogestad, 2003) are given in Table 4.1. The method tries to obtain a first-order closed loop response with time delay of the form

$$\left(\frac{y}{r}\right)_{\text{desired}} = \frac{1}{\tau_c s + 1} e^{-\theta s} \tag{4.13}$$

where τ_c is the desired closed-loop time constant, which is the only tuning parameter. If the process models are not in the forms given in Table 4.1, then they should be manipulated to conform to the basic forms given in Table 4.1 using the rules given below. Only the rules from Skogestad (2003) that apply to the linear models of Section 4.3 are given below.

4.4.1 Simplifying first-order transfer function models

Skogestad (2003) developed simplification rules to get almost any arbitrary transfer function model into either a first-order transfer function model with time delay or a second order model with time delay. Only the rules that apply to the linear models obtained in Section 4.3 will be given here.

The first-order linear model for the PSE-CFF loop contains a negative numerator time constant relating to a non-minimum phase zero. This is cast into the first-order response of

Table 4.1: SIMC-PID settings with τ_c as a tuning parameter for the serial form of the PID controller (from Skogestad (2003)).

Process	$g(s)$	K_c	τ_i	τ_D
First-order	$k\frac{e^{-\theta s}}{(\tau_1 s+1)}$	$\frac{1}{k}\frac{\tau_i}{\tau_c+\theta}$	$\min\{\tau_1, 4(\tau_c+\theta)\}$	—
Integrating	$k\frac{e^{-\theta s}}{s}$	$\frac{1}{k}\frac{1}{\tau_c+\theta}$	$4(\tau_c+\theta)$	—

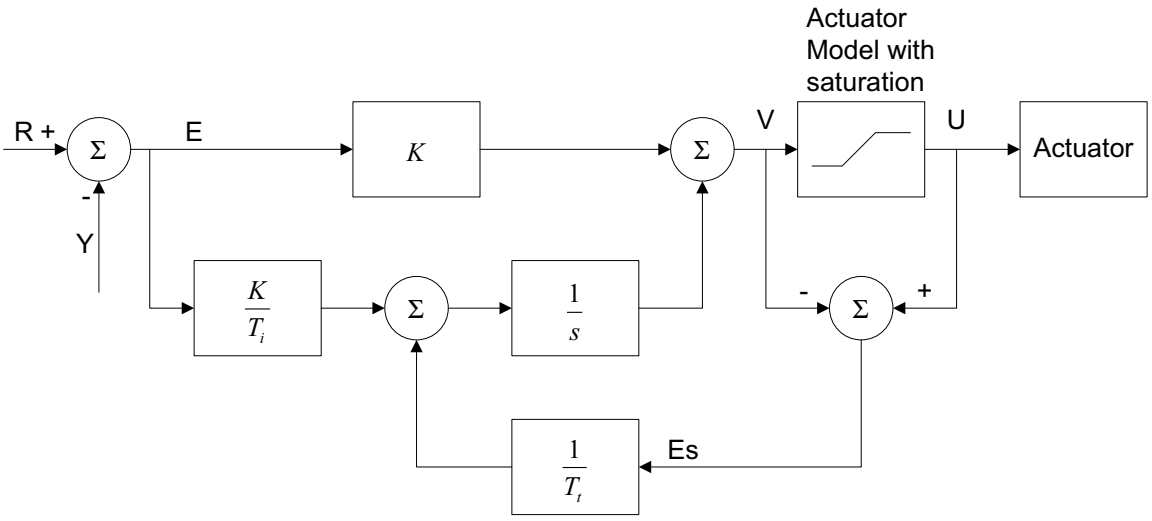


Figure 4.4: PI Controller with Anti-Windup (from Åström (2002)).

Table 4.1 by subtracting the value of the negative numerator time constant from the time delay to obtain the effective first-order time delay (Skogestad, 2003).

To illustrate the simplification, start with a first-order transfer function model with time delay of the following form

$$G_{\text{fo}} = K \cdot \frac{(1+Zs)}{(1+Ps)} e^{-\theta s}, \tag{4.14}$$

and define the effective time delay as

$$\theta_{\text{effective}} := \theta - Z. \tag{4.15}$$

Applying the effective time delay to the transfer function model in (4.14) gives the equivalent first-order transfer function model

$$G_{\text{fo-eq}} = K \cdot \frac{1}{(1+Ps)} e^{-\theta_{\text{effective}} s}. \tag{4.16}$$

4.5 IMPLEMENTATION

The PI parameter values are obtained from Table 4.1 using the simplified models obtained by following the rules outlined above. The PI parameter values are for the serial form of the

controller as well as the parallel form, because there is no differential term. The structure of the anti-windup PI controller is shown in Figure 4.4.

4.5.1 PI controller for the PSE-CFF loop

The PSE-CFF loop is characterised by a first-order transfer function model with time delay. The model is derived in Section 4.3.1 and given by

$$G_{\text{PSE-CFF}}(s) = K_{PC}\frac{(1+Z_{PC}s)}{(1+P_{PC}s)}e^{(-\theta_{PC}s)} \tag{4.17}$$

$$G_{\text{PSE-CFF}}(s) = -0.00035\frac{(1-0.63s)}{(1+0.54s)}e^{(-0.011s)} \tag{4.18}$$

with the effective time delay given by

$$\theta_{\text{PC-EFF}} = \theta_{PC} - Z_{PC} \tag{4.19}$$

$$= 0.01 + 0.63 \tag{4.20}$$

$$= 0.64 \tag{4.21}$$

to give the equivalent first-order model

$$G_{\text{PSE-CFF-EQ}}(s) = K_{PC}\frac{1}{(1+P_{PC}s)}e^{(-\theta_{\text{PC-EFF}}s)} \tag{4.22}$$

$$= -0.00035\frac{1}{(1+0.54s)}e^{(-0.64s)} \tag{4.23}$$

that gives the following PI parameter values by using the rules of Table 4.1

$$K_c = -1187,\ \tau_i = 2.6,\ \tau_d = 0. \tag{4.24}$$

4.5.2 PI controller for the LOAD-MFS loop

The LOAD-MFS loop is characterised by an integrating transfer function model. The model is derived in Section 4.3.2 and given by

$$G_{\text{LOAD-MFS}}(s) = \frac{K_{LF}}{s} \tag{4.25}$$

$$G_{\text{LOAD-MFS}}(s) = \frac{0.01}{s} \tag{4.26}$$

that gives the following PI parameter values by using the rules of Table 4.1

$$K_c = 10000,\ \tau_i = 0.04,\ \tau_d = 0. \tag{4.27}$$

Table 4.2: Parameters of the three PI Controllers

Process	g(s)	k	τ_1	θ	K_c	τ_i	τ_D
PSE-CFF	$k\frac{e^{-\theta s}}{(\tau_1 s+1)}$	-0.00035	0.54	0.64	-1187	2.6	—
SLEV-SFW	$\frac{k}{s}$	0.42	—	—	238	0.04	—
LOAD-MFS	$\frac{k}{s}$	0.01	—	—	10000	0.04	—

4.5.3 PI controller for the SLEV-SFW loop

The SLEV-SFW loop is characterised by an integrating transfer function model. The model is derived in Section 4.3.3 and given by

$$G_{\text{SLEV-SFW}}(s) = \frac{K_{SW}}{s} \tag{4.28}$$

$$G_{\text{SLEV-SFW}}(s) = \frac{0.42}{s} \tag{4.29}$$

that gives the following PI parameter values by using the rules of Table 4.1

$$K_c = 238,\ \tau_i = 0.04,\ \tau_d = 0. \tag{4.30}$$

4.6 SUMMARY

A tuning method for PI control with anti-windup is provided in this chapter.

Linearised models are derived from the nonlinear model of Mintek by conducting step tests on the nonlinear model and fitting it to models with relevant forms.

PI controllers are designed for the linear models derived in Section 4.3 and some models are further simplified in Section 4.5 before obtaining the PI controller parameters from Table 4.1.

The PI controllers are applied to the nonlinear model presented in Section 2.3.4 and the results of the simulations are given in Chapter 5.1 for comparison to the simulations of the robust nonlinear model predictive controller presented in Section 3.7.

The three loops with their model and controller parameters are summarised in Table 4.2.