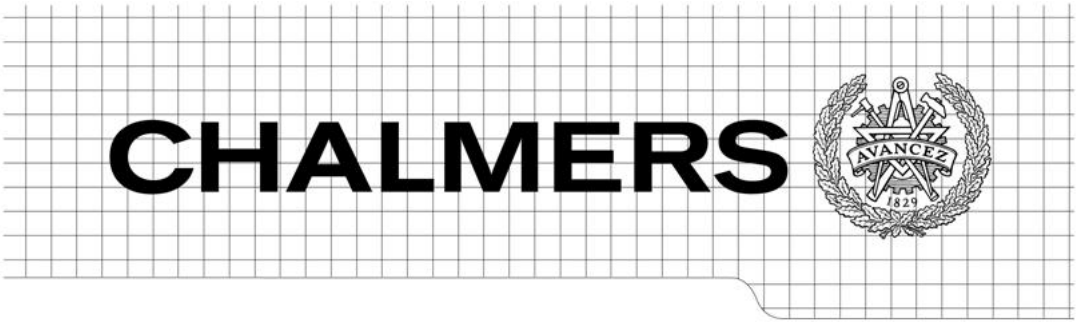

Nonlinear Model Predictive Controller Toolbox

Ehsan Harati

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Science in Systems, Control and Mechatronics,
Reference Number: EX054/2011

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2011

Nonlinear Model Predictive Controller Toolbox

Master's Thesis 2011: May

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
Chalmers University of Technology
SE-41296 Göteborg
Sweden

Tel. +46-(0)31 772 1000

Reproservice / Department of Signals and Systems
Göteborg, Sweden 2011

Nonlinear Model Predictive Controller Toolbox
Master's Thesis in the Master's programme in Systems, Control and Mechatronics
Ehsan Harati
Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
Chalmers University of Technology

Abstract

Model Predictive Control (MPC) is an optimal control method. At each instant of time, a performance index is minimized with respect to a sequence of nominal control inputs and the first optimal control inputs are applied to the plant. At the next time step, the optimization problem is formulated and solved based on new estimates of states. MPC for nonlinear systems can lead to complex optimization problems, which can be computationally demanding and prevents the real-time execution. In this thesis, we describe various low complexity computational schemes for Nonlinear (NL) MPC controller.

Keywords: Model Predictive Control, Quadratic Programming, Optimization

Contents

Abstract iii

Contents vi

Acknowledgements vii

1. Introduction 1
- 1.1. Background and Motivation . . . 1
- 1.2. Thesis Contributions . . . 1
- 1.3. Thesis Review . . . 2

2. Nonlinear MPC 3

3. Linear Time Invariant MPC 7
- 3.1. Plant Model . . . 7
- 3.2. Quadratic Programming Formulation . . . 11

4. Linear Time Variant Model Predictive Control 13
- 4.1. Plant Model . . . 13
- 4.2. Quadratic Programming Formulation . . . 15

5. MPC Additional Considerations 17
- 5.1. Delayed systems formulation . . . 17
  - 5.1.1. Numerical Example . . . 17
- 5.2. MPC Scaling . . . 19
- 5.3. Open Loop Prediction . . . 20
- 5.4. Offset Free MPC . . . 21

6. Results and Conclusion 27
- 6.1. Conclusion . . . 31

Bibliography 33

A. MPC Toolbox 35
- A.1. Software Installation . . . 36
- A.2. MPC Model . . . 37
- A.3. MPC Objective . . . 42
- A.4. MPC constraints . . . 44

A.5. Real-Time Implementation 46
A.6. Advanced Features 49
A.7. Simulink Block 51
A.8. MPC commands 54
A.8.1. General Formulation 54
A.9. Example 56

Acknowledgements

To my family, for their endless love and support

1. Introduction

1.1. Background and Motivation

Model Predictive Control (MPC) is a control strategy that calculates control inputs by solving constrained optimal control problem over a finite time horizon. At each instant of time, a performance index is minimized with respect to a sequence of nominal control inputs and the first optimal control inputs are applied to the plant. A new optimization problem will later be solved when new estimation of states becomes available, this method is also called as Receding Horizon (See figure (1.1)).

MPC arose in industrial and chemical industry in early 1960's, called in its early version as Identification and Command (IDCOM) and Dynamic Matrix Control (DMC). Since then, extensive research efforts have established theories for conditions on feasibility and closed-loop stability of MPC [3], [2]. The emerging computational power paved to apply MPC to new applications. MPC is applied in power electronics [13], aviation [1] or vehicle safety [4], [9].

MPC can be computationally demanding for control of Nonlinear (NL) systems. The optimization problem complexity depends on model, constraints and computational scheme. We consider reducing the complexity of optimization problem by using approximate plant models.

MPC uses a plant model to predict its output trajectories. Nonlinear models arise in various dynamical systems, which give rise to NL-MPC. Nonlinear MPC problems lead to nonlinear and non-convex optimization problems, which can be computationally demanding. Hybrid models can be used approximate nonlinear systems, however MPC schemes based on hybrid models are often too complex. If the plant can be approximated by a Linear Time Invariant (LTI) model, the computational-burden of MPC can be reduced significantly. However, LTI models are often insufficient to accurately model nonlinear systems. Linear Time Variant (LTV) models can approximate nonlinear systems more accurately. MPC based on LTV models (LTV-MPC), can be applied to a broader class of nonlinear systems and they have less complexity than NL-MPC.

1.2. Thesis Contributions

We have developed a MPC Toolbox capable of handling different low complexity MPC schemes. Nonlinear MPC Toolbox can solve MPC problems based on Nonlinear, Linear Time Invariant and Linear Time Variant models. LTI or LTV models can be used to approximate nonlinear sys-

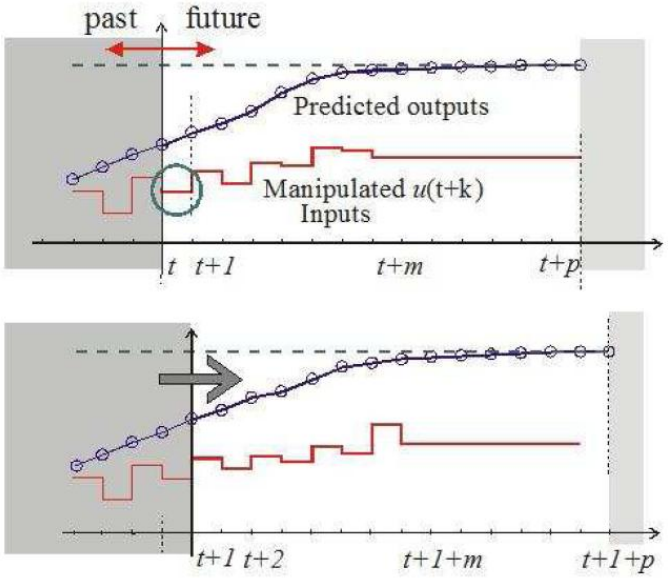

Figure 1.1.: MPC is done in a Receding Horizon scheme

tems, thus reducing the computational burden. Consequently, the most suitable formulation should be chosen based on the computational power available and the accuracy of the approximate models.

1.3. Thesis Review

In the next chapter, we formulate NL-MPC. In chapter three, we present LTI-MPC scheme. In chapter four, we show the LTV-MPC. We will then extend MPC formulation to handle delayed systems and offset-free control. In chapter five, MPC Toolbox is tested on a rich application. The overview of the MPC Toolbox can be found in appendix.

2. Nonlinear MPC

Hereafter, we will use the plant structure as defined in figure (2.1). The inputs of the plant consist of manipulated variables or control inputs, measured and unmeasured disturbances. The plant outputs consist of tracking and constrained outputs

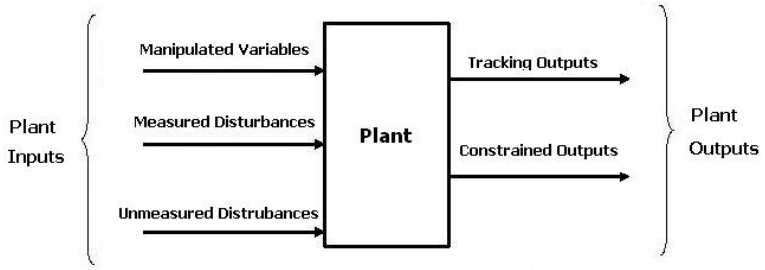

Figure 2.1.: Plant with the inputs and outputs [7].

We assume the plant can be described by a set of first order nonlinear differential equations

$$\dot{x} = f_c(x(t), u(t), d(t), v(t)), \tag{2.1a}$$
$$y = h(x(t), u(t), d(t), v(t)), \tag{2.1b}$$

where $x \in \mathbb{R}^n$ is the state vector, $u \in \mathbb{R}^m$ is manipulated variables of the plant, $d \in \mathbb{R}^{md}$ is the measured disturbances and $v \in \mathbb{R}^{umd}$ is the unmeasured disturbances of the plant. We partition the output vector $y \in \mathbb{R}^p$ as $y = [y_{tr}^T \;\; y_c^T \;\; y_{sc}^T]^T$ where $y_{tr} \in \mathbb{R}^{py}$ are the outputs to be tracked, $y_c \in \mathbb{R}^{pc}$ are the outputs to be hard constrained and $y_{sc} \in \mathbb{R}^{pu}$ are the outputs to be soft constrained.

We assume the state update function f_c, and output function h are continuously differentiable. The plant model (2.1) can be converted to the following discrete time model

$$x(t+h) = f(x(t), u(t), d(t), v(t)) = x(t) + T_s\, f_c(x(t), u(t), d(t), v(t)), \tag{2.2a}$$
$$y = h(x(t), u(t), d(t), v(t)). \tag{2.2b}$$

For simplicity, we will assume the sample time (T_s) to be one. Since we consider the optimal control problem to be solved using a digital processor, we hereafter will use the discrete time

plant description (2.2). The control input rate can be calculated as

$$\triangle u[t+i|t] = u[t+i+1|t] - u[t+i|t]. \quad \forall i \in \{0, ..., N\}. \tag{2.3}$$

We formulate Nonlinear MPC problem as follows

$$J^*(x(t), u(t), \triangle u(t), ..., \triangle u(t+H_c-1)) = \min_{\triangle u(t),...,\triangle u(t+H_c-1)} \sum_{i=0}^{H_p} l(y(t+i|t), u(t+i|t)). \tag{2.4}$$

Subject to

$$x(t+i+1|t) = f(x(t+i|t), u(t+i|t), d(t+i|t)) \quad \forall i \in \{0, ..., H_p-1\}, \tag{2.5a}$$
$$y(t+i|t) = h(x(t+i|t), u(t+i|t), d(t+i|t)) \quad \forall i \in \{0, ..., H_p-1\}, \tag{2.5b}$$
$$x(t|t) = x(t), \tag{2.5c}$$
$$x(t+i|t) \in \mathcal{X}(t+i|t) \quad \forall i \in \{0, ..., H_p-1\}, \tag{2.5d}$$
$$y(t+i|t) \in \mathcal{Y}(t+i|t) \quad \forall i \in \{1, ..., H_p\}, \tag{2.5e}$$
$$u(t+i|t) \in \mathcal{U}(t+i|t) \quad \forall i \in \{1, ..., H_c\}, \tag{2.5f}$$
$$\triangle u(t+i|t) \in \triangle\mathcal{U}(t+i|t) \quad \forall i \in \{0, ..., H_c-1\}, \tag{2.5g}$$

The notation $y(t+i|t)$ denotes that the output predicted at time $t+i$ is based on the information available at time t. The function l is the performance index function. $\mathcal{X}, \mathcal{Y}, \mathcal{U}, \triangle\mathcal{U}$ are the set of admissible states, outputs, control inputs and control input rates, respectively. We assume that the control input remains unchanged after a control horizon (H_c) which is smaller than the prediction horizon (H_p).
The optimal control sequence $[\triangle u^*(t), ..., \triangle u^*(t+H_c-1)]$ is found by minimizing (2.4) subject to (2.5). The optimization problem can be solved using a nonlinear optimization solver [10]. The first optimal control input is then applied to the plant. At the next sampling time the plant states are estimated and a new optimization problem is solved over a shifted horizon. However, the optimization problem (2.4) is in general non-convex and can be computationally demanding for real-time applications.

We use a quadratic function for the performance index, that is to use squared 2 norm to penalize tracking error, control input and input changes. Additionally, we define the feasibility regions with different polytopes. The following Nonlinear MPC formulation is defined in MPC Toolbox

$$J^*(x(t), u(t), \triangle u(t), ..., \triangle u(t+H_c-1)) = \min_{\triangle u(t),...,\triangle u(t+H_c-1)} \sum_{i=1}^{H_p} ||\mathcal{Q}[y_{tr}(t)(t+i) - y_{ref}(t+i)]||_2^2 + ||\mathcal{S}u(t+i)||_2^2 + ||\mathcal{R}\triangle u(t+i-1)||_2^2. \tag{2.6}$$

Subject to

$$x(t+i+1|t) = f(x(t+i|t), u(t+i|t), d(t+i|t), v(t+i|t)) \quad \forall i \in \{0, ..., H_p\}, \tag{2.7a}$$

$$y(t+i|t) = h(x(t+i|t), u(t+i|t), d(t+i|t), v(t+i|t)) \quad \forall i \in \{0, ..., H_p\}, \tag{2.7b}$$

$$x(t|t) = x(t) \tag{2.7c}$$

$$u[t+i+1|t] = u[t+i|t] + \triangle u[t+i|t] \quad \forall i \in \{0, ..., H_c - 1\}, \tag{2.7d}$$

$$\triangle \mathbb{U}_l(t+i) \leq \triangle u[t+i|t] \leq \triangle \mathbb{U}_u(t+i) \quad \forall i \in \{0, ..., H_c\}, \tag{2.7e}$$

$$\mathbb{U}_l(t+i) \leq u[t+i|t] \leq \mathbb{U}_u(t+i) \quad \forall i \in \{1, ..., H_c\}, \tag{2.7f}$$

$$\mathbb{Y}_{c,l}(t+i) \leq y_c[t+i|t] \leq \mathbb{Y}_{c,u}(t+i) \ \ \forall i \in \{0, ..., H_p\}, \tag{2.7g}$$

$$\mathbb{Y}_{sc,l}(t+i) \leq y_{sc}[t+i|t] \leq \mathbb{Y}_{sc,u}(t+i) \ \ \forall i \in \{0, ..., H_p\}, \tag{2.7h}$$

$$\mathbb{E}_l \leq y_{tr}[t+i|t] - y_{ref}[t+i|t] \leq \mathbb{E}_u \quad \forall i \in \{0, ..., H_p\}, \tag{2.7i}$$

$$\mathbb{G}_l \leq E[x(t)...x(t+H_{gc})] + F[u(t)...u(t+H_{gc})] \leq \mathbb{G}_u, \tag{2.7j}$$

where $\mathbb{U}_l, \mathbb{U}_u$ and $\triangle \mathbb{U}_l, \triangle \mathbb{U}_u$ are the lower and upper bound on control inputs and control input rates. $\mathbb{Y}_{c,l}, \mathbb{Y}_{c,u}$ are the lower and upper bound on hard constrained outputs. $\mathbb{Y}_{sc,l}, \mathbb{Y}_{sc,u}$ are the lower and upper bound on soft constrained outputs. $\mathbb{E}_l, \mathbb{E}_u$ are the lower and upper bound on tracking error. $\mathbb{G}_l, \mathbb{G}_u$ are the lower and upper bound on general constraints.

3. Linear Time Invariant MPC

NL-MPC defined in previous chapter can be computationally demanding. Optimization problem complexity, among others, can stem from model nonlinearities. In this chapter, we consider linearizing the plant model around the operation point to obtain Linear Time Invariant (LTI) MPC scheme.

3.1. Plant Model

The plant model (2.2) can be linearized around the operation point to yield the linearized plant formulation

$$x(k+1) = A\,x(k) + B\,u(k) + B_d\,d(k), \tag{3.1a}$$
$$y(k) = C\,x(k) + D\,u(k) + D_d\,d(k), \tag{3.1b}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $B_d \in \mathbb{R}^{n\times md}$, $C \in \mathbb{R}^{p\times n}$, $D \in \mathbb{R}^{p\times n}$, $D_d \in \mathbb{R}^{p\times md}$. We partition the output vector $y \in \mathbb{R}^p$ as $y = [y_{tr}^T \;\; y_c^T \;\; y_{sc}^T]^T$ where $y_{tr} \in \mathbb{R}^{py}$ are the tracking outputs, $y_c \in \mathbb{R}^{pc}$ are the hard constrained outputs and $y_{sc} \in \mathbb{R}^{pu}$ are the soft constrained outputs. The matrices A, B, B_d, C, D, D_d are calculated as

$$A = (\frac{\partial f}{\partial x})_{x=\bar{x},u=\bar{u},d=\bar{d}}, \quad B = (\frac{\partial f}{\partial u})_{x=\bar{x},u=\bar{u},d=\bar{d}}, \tag{3.2a}$$
$$B_d = (\frac{\partial f}{\partial d})_{x=\bar{x},u=\bar{u},d=\bar{d}}, \quad C = (\frac{\partial h}{\partial x})_{x=\bar{x},u=\bar{u},d=\bar{d}}, \tag{3.2b}$$
$$D = (\frac{\partial h}{\partial u})_{x=\bar{x},u=\bar{u},d=\bar{d}}, \quad D_d = (\frac{\partial h}{\partial d})_{x=\bar{x},u=\bar{u},d=\bar{d}}. \tag{3.2c}$$

where $(x = \bar{x}, u = \bar{u}, d = \bar{d})$ denotes the operation point.

By using (2.3), the state space model (3.1) can be rewritten as

$$\tilde{x}(k+1|t) = \tilde{A}\tilde{x}(k|t) + \tilde{\mathcal{B}}\triangle u(k|t) + \tilde{d}(k|t), \tag{3.3a}$$
$$y(k) = \tilde{\mathcal{C}}\tilde{x}(k|t) + \tilde{\mathcal{D}}\triangle u(k|t) + \tilde{e}(k|t), \tag{3.3b}$$

where

$$\tilde{\mathcal{A}} = \begin{pmatrix} \mathcal{A} & \mathcal{B} \\ 0_{m\times n} & I_m \end{pmatrix}, \quad \tilde{\mathcal{B}} = \begin{pmatrix} \mathcal{B} \\ I_m \end{pmatrix} \tag{3.4a}$$

$$,\tilde{\mathcal{C}} = \begin{pmatrix} \mathcal{C} & \mathcal{D} \end{pmatrix}, \quad \tilde{\mathcal{D}} = \mathcal{D}, \tag{3.4b}$$

$$\tilde{x}(k|t) = \begin{pmatrix} x(k|t) \\ u(k-1|t) \end{pmatrix}, \tag{3.4c}$$

$$d(k|t) = \begin{pmatrix} B_d d(k|t) \\ 0_m \end{pmatrix}, \tag{3.4d}$$

$$e_{k,t} = D_d\, d(t|t) \tag{3.4e}$$

$$\tag{3.4f}$$

where 0_n and I_n are zero and identity matrix of size n.

The predicted output at time $t+i$ can be written in terms of the state at time t and the control inputs $u(t), \cdots, u(t+i-1)$ as

$$\begin{aligned} y(t+i|t) = \tilde{\mathcal{C}}\tilde{\mathcal{A}}^i x(t|t) + \tilde{\mathcal{C}} \sum_{k=0}^{i-1} \tilde{\mathcal{A}}^{i-k-1} \tilde{\mathcal{B}} \triangle u(t+k|t) + \\ \tilde{\mathcal{C}} \sum_{k=0}^{i-1} \tilde{\mathcal{A}}^{i-k-1} \tilde{d}(t|t) + \tilde{\mathcal{D}} \triangle u(t+i|t) + \tilde{e}(t|t). \end{aligned} \tag{3.5}$$

The output prediction vector $\mathcal{Y}(t)$ containing output predictions over a time horizon of H_p steps can be calculated as

$$\mathcal{Y}(t) = \Psi x(k|t) + \Theta \triangle \mathcal{U}(t) + \Gamma \Phi(t) + \Lambda(t), \tag{3.6}$$

where $\mathcal{Y}(t) \in \mathbb{R}^{pH_p}$, $\triangle\mathcal{U}(t) \in \mathbb{R}^{mH_c}$, $\Phi(t) \in \mathbb{R}^{nH_p}$ and $\Lambda(t) \in \mathbb{R}^{pH_p}$. Accordingly $\Psi \in \mathbb{R}^{pH_p \times n}$, $\Gamma \in \mathbb{R}^{pH_p \times nH_p}$and $\Theta \in \mathbb{R}^{pH_p \times mH_c}$ are defined as

$$\Psi = \begin{bmatrix} \tilde{\mathcal{C}}\tilde{\mathcal{A}} \\ \tilde{\mathcal{C}}\tilde{\mathcal{A}}\tilde{\mathcal{A}} \\ \tilde{\mathcal{C}}\prod_{i=0}^{2}\tilde{\mathcal{A}} \\ \vdots \\ \tilde{\mathcal{C}}\prod_{i=0}^{H_p-1}\tilde{\mathcal{A}}_{i,t} \end{bmatrix}, \Theta = \begin{bmatrix} \tilde{\mathcal{C}}\tilde{\mathcal{B}} & \tilde{\mathcal{D}} & \cdots & 0 & \\ \tilde{\mathcal{C}}\tilde{\mathcal{A}}\tilde{\mathcal{B}} & \tilde{\mathcal{C}}\tilde{\mathcal{B}} & \tilde{\mathcal{D}} & \cdots & 0 \\ \tilde{\mathcal{C}}\tilde{\mathcal{A}}\tilde{\mathcal{A}}\tilde{\mathcal{B}} & \tilde{\mathcal{C}}\tilde{\mathcal{A}}\tilde{\mathcal{B}} & \tilde{\mathcal{C}}\tilde{\mathcal{B}} & \tilde{\mathcal{D}} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \\ \tilde{\mathcal{C}}\prod_{i=1}^{H_c-1}\tilde{\mathcal{A}}\tilde{\mathcal{B}} & \tilde{\mathcal{C}}\prod_{i=2}^{H_c-1}\tilde{\mathcal{A}}\tilde{\mathcal{B}} & \cdots & \cdots & \tilde{\mathcal{C}}\tilde{\mathcal{B}} \\ \tilde{\mathcal{C}}\prod_{i=1}^{H_c}\tilde{\mathcal{A}}\tilde{\mathcal{B}} & \tilde{\mathcal{C}}\prod_{i=2}^{H_c}\tilde{\mathcal{A}}\tilde{\mathcal{B}} & \cdots & \cdots & \tilde{\mathcal{C}}\tilde{\mathcal{A}}\tilde{\mathcal{B}} \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ \tilde{\mathcal{C}}\prod_{i=1}^{H_p-1}\tilde{\mathcal{A}}\tilde{\mathcal{B}} & \tilde{\mathcal{C}}\prod_{i=2}^{H_p-1}\tilde{\mathcal{A}}\tilde{\mathcal{B}} & \cdots & \cdots & \tilde{\mathcal{C}}\prod_{i=H_c}^{H_p-1}\tilde{\mathcal{A}}\tilde{\mathcal{B}} \end{bmatrix} \tag{3.7}$$

$$\Gamma = \begin{bmatrix} \tilde{\mathcal{C}} & 0_{p\times n} & \cdots & 0_{p\times n} \\ \tilde{\mathcal{C}}\tilde{\mathcal{A}} & \tilde{\mathcal{C}} & \ddots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ \tilde{\mathcal{C}}\prod_{i=1}^{H_p-1}\tilde{\mathcal{A}} & \tilde{\mathcal{C}}\prod_{i=2}^{H_p-1}\tilde{\mathcal{A}} & \cdots & \tilde{\mathcal{C}} \end{bmatrix}$$

$$\Phi = \begin{bmatrix} \tilde{d}(t|t) \\ \tilde{d}(t|t) \\ \vdots \\ \tilde{d}(t|t) \end{bmatrix}, \Lambda = \begin{bmatrix} 0 \\ e(t|t) \\ \vdots \\ e(t|t) \end{bmatrix}$$

Note here that the matrices Ψ,Γ and Θ are time independent and they can be computed offline. Moreover, output predictions for the tracking and constrained outputs can be divided by

$$\mathcal{Y}_{tr}(t) = A_{tr}\,\mathcal{Y}(t), \tag{3.8a}$$
$$\mathcal{Y}_{c}(t) = A_{c}\,\mathcal{Y}(t), \tag{3.8b}$$
$$\mathcal{Y}_{sc}(t) = A_{sc}\,\mathcal{Y}(t), \tag{3.8c}$$

where A_{tr}, A_c and A_{sc} are defined as

$$A_{tr} = \begin{bmatrix} I_{py\times py} & 0_{py\times pc} & 0_{py\times pu} \\ 0_{pc\times py} & 0_{pc\times pc} & 0_{pc\times pu} \\ 0_{pu\times py} & 0_{pu\times pc} & 0_{pu\times pu} \end{bmatrix}_{p\times p} \otimes 1_{H_p\times H_p}, \tag{3.9a}$$

$$A_{c} = \begin{bmatrix} 0_{py\times py} & 0_{py\times pc} & 0_{py\times pu} \\ 0_{pc\times py} & I_{pc\times pc} & 0_{pc\times pu} \\ 0_{pu\times py} & 0_{pu\times pc} & 0_{pu\times pu} \end{bmatrix}_{p\times p} \otimes 1_{H_p\times H_p}, \tag{3.9b}$$

$$A_{sc} = \begin{bmatrix} 0_{py\times py} & 0_{py\times pc} & 0_{py\times pu} \\ 0_{pc\times py} & 0_{pc\times pc} & 0_{pc\times pu} \\ 0_{pu\times py} & 0_{pu\times pc} & I_{pu\times pu} \end{bmatrix}_{p\times p} \otimes 1_{H_p\times H_p} \tag{3.9c}$$

with $\otimes$ denoting Kronecker product and $1_{i\times j}$ is the matrix$(i \times j)$ of ones. Using the equation (3.8), we can derive

$$\underbrace{A_{tr}\mathcal{Y}(t)}_{\mathcal{Y}_{tr}(t)} = \underbrace{A_{tr}\Psi}_{\Psi_{tr}} x(k|t) + \underbrace{A_{tr}\Theta}_{\Theta_{tr}} \triangle\mathcal{U}(t) + \underbrace{A_{tr}\Gamma}_{\Gamma_{tr}} \Phi(t) + \underbrace{A_{tr}\Lambda(t)}_{\Lambda_{tr}(t)}, \tag{3.10a}$$

$$\underbrace{A_{c}\mathcal{Y}(t)}_{\mathcal{Y}_{c}(t)} = \underbrace{A_{c}\Psi}_{\Psi_{c}} x(k|t) + \underbrace{A_{c}\Theta}_{\Theta_{c}} \triangle\mathcal{U}(t) + \underbrace{A_{c}\Gamma}_{\Gamma_{c}} \Phi(t) + \underbrace{A_{c}\Lambda(t)}_{\Lambda_{c}(t)}, \tag{3.10b}$$

$$\underbrace{A_{sc}\mathcal{Y}(t)}_{\mathcal{Y}_{sc}(t)} = \underbrace{A_{sc}\Psi}_{\Psi_{sc}} x(k|t) + \underbrace{A_{sc}\Theta}_{\Theta_{sc}} \triangle\mathcal{U}(t) + \underbrace{A_{sc}\Gamma}_{\Gamma_{sc}} \Phi(t) + \underbrace{A_{sc}\Lambda(t)}_{\Lambda_{sc}(t)}. \tag{3.10c}$$

3.2. Quadratic Programming Formulation

We convert the optimal control formulation defined as (2.6) with the predictions made as (3.7), to a quadratic programming which can be solved by standard solvers. The standard quadratic programming has the form

$$\begin{aligned} J^*(\mathcal{X}^*(t)) &= \min_{\mathcal{X}(t)} X^T A X + B X \\ subjectto &: X_l \leq EX \leq X_u \end{aligned} \tag{3.11}$$

The performance index (2.4) can be calculated as

$$J = ||Q(\mathcal{E}_f + \mathcal{E})||_2^2 + ||S\mathcal{U}(t)||_2^2 + ||R\triangle\mathcal{U}(t)||_2^2 + \rho\epsilon^2 \tag{3.12a}$$

$$\mathcal{E}_f = \mathcal{Y}_{ref}(t) - \Psi_{tr}\tilde{x}(k|t) - \Gamma_{tr}\Phi(t) - \Lambda_{tr}(t) \tag{3.12b}$$

$$\mathcal{E} = \Theta_{tr}\triangle\mathcal{U}(t) \tag{3.12c}$$

We call $\mathcal{E}_f \in \mathbb{R}^{pH_p}$ free response tracking error, and $\mathcal{E} \in \mathbb{R}^{pH_p}$ is control move improvement of the tracking error. We can transform (3.12) to the following quadratic programming

$$\triangle\mathcal{U}^*(t) = \arg\min_{\triangle\mathcal{U}(t)} J = [\triangle\mathcal{U}(t)^T, \epsilon]\mathcal{H}_t[\triangle\mathcal{U}(t)^T, \epsilon]^T + \mathcal{F}_t[\triangle\mathcal{U}(t)^T, \epsilon]^T \tag{3.13a}$$

$$\mathcal{H}_t = \begin{pmatrix} \Theta_{tr}^T Q \Theta_{tr} + R + A_a^T S A_a & 0^{mH_c \times 1} \\ 0^{1 \times mH_c} & \rho \end{pmatrix} \tag{3.13b}$$

$$\mathcal{F}_t = 2\mathcal{E}_f^T Q \Theta_{tr} \tag{3.13c}$$

where $A_a \in \mathbb{R}^{H_c \times H_c}$ is the lower triangular matrix defined as

$$A_a = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}_{H_c \times H_c} \otimes I_m \tag{3.14}$$

The constraints (2.5) can then be converted to

$$\triangle U_l \leq \triangle\mathcal{U} \leq \triangle U_u, \tag{3.15a}$$

$$U_l - U_t \leq \triangle\mathcal{U} \leq \triangle U_u - U_t, \tag{3.15b}$$

$$E_l \leq \mathcal{Y}_{ref}(t) - \Psi_{tr}\tilde{x}(k|t) - \Gamma_{tr}\Phi(t) - \Lambda_{tr}(t) - \Theta_{tr}\triangle\mathcal{U}(t) \leq E_u, \tag{3.15c}$$

$$Y_{c,l} \leq \Psi_c\tilde{x}(k|t) + \Gamma_c\Phi(t) + \Lambda_c(t) + \Theta_c\triangle\mathcal{U}(t) \leq Y_{c,u}, \tag{3.15d}$$

$$Y_{sc,l} - \epsilon\Xi \leq \Psi_{sc}\tilde{x}(k|t) + \Gamma_{sc}\Phi(t) + \Lambda_{sc}(t) + \Theta_{sc}\triangle\mathcal{U}(t) \leq Y_{sc,u} + \epsilon\Xi, \tag{3.15e}$$

where

$$U_t = u(t-1) \otimes 1_H c, \tag{3.16a}$$

$$\Xi = 1_{pu.Hc}. \tag{3.16b}$$

We finally can write the LTI-MPC formulation implemented in MPC Toolbox as the following standard quadratic programming

$$\triangle \mathcal{U}^*(t) = \arg\min_{\triangle \mathcal{U}(t)} J = [\triangle \mathcal{U}(t)^T, \epsilon] \mathcal{H}_t [\triangle \mathcal{U}(t)^T, \epsilon]^T + \mathcal{F}_t [\triangle \mathcal{U}(t)^T, \epsilon]^T \tag{3.17}$$

Subject to

$$\begin{bmatrix} \triangle U_l \\ 0 \end{bmatrix} \leq \begin{bmatrix} \triangle \mathcal{U}(t) \\ \epsilon \end{bmatrix} \leq \begin{bmatrix} \triangle U_u \\ \epsilon_u \end{bmatrix} \tag{3.18a}$$

$$\begin{bmatrix} A & 0_{mH_c \times 1} \\ -A & 0_{mH_c \times 1} \\ -A & 0_{mH_c \times 1} \\ \Theta_{tr} & 1_{py.H_c \times 1} \\ -\Theta_{tr} & 1_{py.H_c \times 1} \\ \Theta_c & 0_{pc.H_c \times 1} \\ -\Theta_c & 0_{pc.H_c \times 1} \\ \Theta_{sc} & 1_{pu.H_c \times 1} \\ -\Theta_{sc} & 1_{pu.H_c \times 1} \end{bmatrix} \begin{bmatrix} \triangle \mathcal{U}(t) \\ \epsilon \end{bmatrix} \leq \begin{bmatrix} U_u - U_t \\ -U_l + U_t \\ \mathcal{Y}_{ref}(t) - \Psi_{tr}\tilde{x}(k|t) - \Gamma_{tr}\Phi(t) - \Lambda_{tr}(t) + E_u \\ -\mathcal{Y}_{ref}(t) + \Psi_{tr}\tilde{x}(k|t) + \Gamma_{tr}\Phi(t) + \Lambda_{tr}(t) - E_l \\ -\Psi_c\tilde{x}(k|t) - \Gamma_c\Phi(t) - \Lambda_c(t) + Y_{c,u} \\ \Psi_c\tilde{x}(k|t) + \Gamma_c\Phi(t) + \Lambda_c(t) - Y_{c,l} \\ -\Psi_{sc}\tilde{x}(k|t) - \Gamma_{sc}\Phi(t) - \Lambda_{sc}(t) + Y_{sc,u} \\ \Psi_{sc}\tilde{x}(k|t) + \Gamma_{sc}\Phi(t) + \Lambda_{sc}(t) - Y_{sc,l} \end{bmatrix} \tag{3.18b}$$

where J is defined as (3.13).

4. Linear Time Variant Model Predictive Control

In the previous chapters, we presented the Nonlinear MPC and LTI-MPC. Although LTI-MPC has reduced computational complexity compared to Nonlinear MPC, LTI models are often insufficient to approximate nonlinear systems over wide operation region of the state and input space. Accuracy of predictions can be improved by using Linear Time Variant(LTV) models. In this chapter, we present MPC scheme based on LTV models, referred to as LTV-MPC. LTV models improves the accuracy of the predictions compared to LTI models while they have reduced computational complexity compared to Nonlinear MPC.

4.1. Plant Model

We can approximate (2.2) as follows

$$x(k+1) = \mathcal{A}_{k,t}x(k) + \mathcal{B}_{k,t}u(k) + d_{k,t}, \tag{4.1a}$$
$$y(k) = \mathcal{C}_{k,t}x(t+i) + \mathcal{D}_{k,t}u(k) + e_{k,t}. \tag{4.1b}$$

The matrices $\mathcal{A}_{t+i,t}, \mathcal{B}_{t+i,t}, \mathcal{C}_{t+i,t}, \mathcal{D}_{t+i,t}$ are calculated as in (3.2) around the operation point $(x(t+i|t), u(t+i|t), d(t+i|t))$.

Using (2.3), the state space model (4.1) can be transformed to

$$x(k+1|t) = \mathcal{A}_{k,t}x(k|t) + \mathcal{B}_{k,t}\triangle u(k|t) + d_{k,t}, \tag{4.2a}$$
$$y(k) = \mathcal{C}_{k,t}x(k|t) + \mathcal{D}_{k,t}\triangle u(k|t) + e_{k,t}. \tag{4.2b}$$

The matrices are defined as in (3.4), with the additional consideration that the matrices are time variant as well.

The output prediction vector $\mathcal{Y}(t)$ for a horizon of H_p steps is computed as

$$\mathcal{Y}(t) = \Psi_t x(k|t) + \Theta_t \triangle\mathcal{U}(t) + \Gamma_t \Phi(t) + \Lambda(t). \tag{4.3}$$

where $\mathcal{Y}(t) \in \mathbb{R}^{pH_p}$, $\triangle\mathcal{U}(t) \in \mathbb{R}^{mH_c}$ and $\Phi(t) \in \mathbb{R}^{nH_p}$ and $\Lambda(t) \in \mathbb{R}^{pH_p}$. Accordingly $\Psi(t) \in \mathbb{R}^{pH_p \times n}$, $\Gamma_t \in \mathbb{R}^{pH_p \times nH_p}$and $\Theta(t) \in \mathbb{R}^{pH_p \times mH_c}$ and $\Lambda(t) \in \mathbb{R}^{pH_p \times nH_p}$ are defined as in (4.4).

In (4.3) the control input is assumed constant after H_c steps. We observe that the matrices in (4.3) are time variant and they should be calculated at every time step, while in the LTI case the matrices in (3.6) can be calculated offline.

$$\mathcal{Y}(t) = \begin{bmatrix} y(t+1|t) \\ y(t+2|t) \\ \vdots \\ y(t+H_p|t) \end{bmatrix} = \underbrace{\begin{bmatrix} \tilde{\mathcal{C}}_{t+1,t}\tilde{\mathcal{A}}_{t,t} \\ \tilde{\mathcal{C}}_{t+2,t}\tilde{\mathcal{A}}_{t+1,t} \\ \vdots \\ \tilde{\mathcal{C}}_{t+H_p,t}\prod_{i=t}^{t+H_p-1}\tilde{\mathcal{A}}_{i,t} \end{bmatrix}}_{\Psi_t} \times \tilde{x}(t|t) +$$

$$\begin{bmatrix} \tilde{\mathcal{C}}_{t+1,t}\tilde{\mathcal{B}}_{t,t} & \tilde{\mathcal{D}}_{t+1,t} & \cdots & 0 \\ \tilde{\mathcal{C}}_{t+2,t}\tilde{\mathcal{A}}_{t+1,t}\tilde{\mathcal{B}}_{t,t} & \tilde{\mathcal{C}}_{t+2,t}\tilde{\mathcal{B}}_{t+1,t} & \tilde{\mathcal{D}}_{t+2,t} & \cdots & 0 \\ \tilde{\mathcal{C}}_{t+3,t}\tilde{\mathcal{A}}_{t+2,t}\tilde{\mathcal{A}}_{t+1,t}\tilde{\mathcal{B}}_{t,t} & \tilde{\mathcal{C}}_{t+3,t}\tilde{\mathcal{A}}_{t+2,t}\tilde{\mathcal{B}}_{t+1,t} & \tilde{\mathcal{C}}_{t+3,t}\tilde{\mathcal{B}}_{t+2,t} & \tilde{\mathcal{D}}_{t+3,t} & \cdots \\ \vdots & \vdots & \vdots & \ddots & \\ \tilde{\mathcal{C}}_{t+H_c,t}\prod_{i=t+1}^{t+H_c-1}\tilde{\mathcal{A}}_{i,t}\tilde{\mathcal{B}}_{t,t} & \tilde{\mathcal{C}}_{t+H_c,t}\prod_{i=t+2}^{t+H_c-1}\tilde{\mathcal{A}}_{i,t}\tilde{\mathcal{B}}_{t+1,t} & \cdots & \cdots & \tilde{\mathcal{C}}_{t+H_c,t}\tilde{\mathcal{B}}_{t+H_c-1,t} \\ \tilde{\mathcal{C}}_{t+H_c+1,t}\prod_{i=t+1}^{t+H_c}\tilde{\mathcal{A}}_{i,t}\tilde{\mathcal{B}}_{t,t} & \tilde{\mathcal{C}}_{t+H_c+1,t}\prod_{i=t+2}^{t+H_c}\tilde{\mathcal{A}}_{i,t}\tilde{\mathcal{B}}_{t+1,t} & \cdots & \cdots & \tilde{\mathcal{C}}_{t+H_c+1,t}\tilde{\mathcal{A}}_{t+H_c,t}\tilde{\mathcal{B}}_{t+H_c-1,t} \\ \vdots & \cdots & \cdots & \ddots & \vdots \\ \tilde{\mathcal{C}}_{t+H_p,t}\prod_{i=t+1}^{t+H_p-1}\tilde{\mathcal{A}}_{i,t}\tilde{\mathcal{B}}_{t,t} & \tilde{\mathcal{C}}_{t+H_p,t}\prod_{i=t+2}^{t+H_p-1}\tilde{\mathcal{A}}_{i,t}\tilde{\mathcal{B}}_{t+1,t} & \cdots & \cdots & \tilde{\mathcal{C}}_{t+H_p,t}\prod_{i=t+H_c}^{t+H_p-1}\tilde{\mathcal{A}}_{i,t}\tilde{\mathcal{B}}_{t+H_c-1,t} \end{bmatrix} \times$$

$$\underbrace{\begin{bmatrix} \triangle u(t|t) \\ \triangle u(t+1|t) \\ \vdots \\ \triangle u(t+H_c-1|t) \end{bmatrix}}_{\triangle\mathcal{U}(t)} +$$

$$\underbrace{\begin{bmatrix} \tilde{\mathcal{C}}_{t+1,t} & 0_{p\times n} & \cdots & 0_{p\times n} \\ \tilde{\mathcal{C}}_{t+2,t}\tilde{\mathcal{A}}_{t+1,t} & \tilde{\mathcal{C}}_{t+2,t} & \ddots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ \tilde{\mathcal{C}}_{t+H_p,t}\prod_{i=t+1}^{t+H_p-1}\tilde{\mathcal{A}}_{i,t} & \tilde{\mathcal{C}}_{t+H_p,t}\prod_{i=t+2}^{t+H_p-1}\tilde{\mathcal{A}}_{i,t} & \cdots & \tilde{\mathcal{C}}_{t+H_p,t} \end{bmatrix}}_{\Gamma_t} \times \underbrace{\begin{bmatrix} \tilde{d}(t|t) \\ \tilde{d}(t+1|t) \\ \vdots \\ \tilde{d}(t+H_p-1|t) \end{bmatrix}}_{\Phi_t} + \underbrace{\begin{bmatrix} 0 \\ e(t+1|t) \\ \vdots \\ e(t+H_p-1|t) \end{bmatrix}}_{\Lambda_t} \tag{4.4}$$

4.2. Quadratic Programming Formulation

Similar to the derivation of LTI-MPC, we can transform (2.6) the quadratic programming to the following standard quadratic programming formulation ([4])

$$\triangle \mathcal{U}^*(t) = \arg\min_{\triangle \mathcal{U}(t)} J = [\triangle \mathcal{U}(t)^T, \epsilon]\mathcal{H}_t[\triangle \mathcal{U}(t)^T, \epsilon]^T + \mathcal{F}_t[\triangle \mathcal{U}(t)^T, \epsilon]^T \tag{4.5a}$$

$$\mathcal{H}_t = \begin{pmatrix} \Theta_t^T Q \Theta_t + R + A_a^T S A_a & 0^{mH_c \times 1} \\ 0^{1 \times mH_c} & \rho \end{pmatrix}, \tag{4.5b}$$

$$\mathcal{F}_t = 2\mathcal{E}_f^T Q \Theta_t, \tag{4.5c}$$

Subject to

$$\begin{bmatrix} \triangle U_l \\ 0 \end{bmatrix} \leq \begin{bmatrix} \triangle \mathcal{U}(t) \\ \epsilon \end{bmatrix} \leq \begin{bmatrix} \triangle U_u \\ \epsilon_u \end{bmatrix} \tag{4.6a}$$

$$\begin{bmatrix} A & 0_{mH_c \times 1} \\ -A & 0_{mH_c \times 1} \\ -A & 0_{mH_c \times 1} \\ \Theta_{tr,t} & 1_{py.H_c \times 1} \\ -\Theta_{tr,t} & 1_{py.H_c \times 1} \\ \Theta_{c,t} & 0_{pc.H_c \times 1} \\ -\Theta_{c,t} & 0_{pc.H_c \times 1} \\ \Theta_{sc,t} & 1_{pu.H_c \times 1} \\ -\Theta_{sc,t} & 1_{pu.H_c \times 1} \end{bmatrix} \begin{bmatrix} \triangle \mathcal{U}(t) \\ \epsilon \end{bmatrix} \leq \begin{bmatrix} U_u - U_t \\ -U_l + U_t \\ \mathcal{Y}_{ref}(t) - \Psi_{tr,t}\tilde{x}(k|t) - \Gamma_{tr,t}\Phi(t) - \Lambda_{tr,t}(t) + E_u \\ -\mathcal{Y}_{ref}(t) + \Psi_{tr}\tilde{x}(k|t) + \Gamma_{tr,t}\Phi(t) + \Lambda_{tr,t}(t) - E_l \\ -\Psi_{c,t}\tilde{x}(k|t) - \Gamma_{c,t}\Phi(t) - \Lambda_{c,t}(t) + Y_{c,u} \\ \Psi_{c,t}\tilde{x}(k|t) + \Gamma_{c,t}\Phi(t) + \Lambda_{c,t}(t) - Y_{c,l} \\ -\Psi_{sc,t}\tilde{x}(k|t) - \Gamma_{sc,t}\Phi(t) - \Lambda_{sc,t}(t) + Y_{sc,u} \\ \Psi_{sc,t}\tilde{x}(k|t) + \Gamma_{sc,t}\Phi(t) + \Lambda_{sc,t}(t) - Y_{sc,l} \end{bmatrix} \tag{4.6b}$$

5. MPC Additional Considerations

5.1. Delayed systems formulation

The inherent optimization process for solving MPC problems leads to a computational delay. If computational delay is not accounted, controlling the plant with MPC leads to loss of performance or closed-loop instability [12]. The delayed MPC scheme can also be applied to delayed plants.

Consider the optimization problem (2.6), at each time instant the optimal control input is generated as

$$u(\tau) = u^*(t_i, x(t_i)), \tau \in [t_i, t_i + 1]. \tag{5.1}$$

Considering the average delay of $\bar{\delta}$, the open-loop optimal control input applied to the plant is

$$u(\tau) = u^*(\tau, x(t_i)), \tau \in [t_i + \bar{\delta}, t_i + 1 + \bar{\delta}]. \tag{5.2}$$

Note that the control input at time $[t_i, t_i + \bar{\delta})$ is not determined by the optimization problem solved for current time instant. Consequently, the following constraint should be added to optimization formulation ([12])

$$\bar{u}(\tau) = u^*_{t_i - 1}(\tau, x(t_i)), \tau \in [t_i, t_i + \bar{\delta}). \tag{5.3}$$

When all the control input channels have equal time delay the optimization problem formulation, the constraint (5.3) corresponds to letting the system run freely prior to time $t + \bar{\delta}$ with the control inputs already fed to the plant, and the change of problem formulation corresponds to updating the states. The approach is applicable for Nonlinear MPC formulation where stability can be proven for the modified MPC formulation. Assuming there is no model mismatch, the previous optimal control inputs are feasible for the system [12].

5.1.1. Numerical Example

Consider a delayed double integrator system defined as

$$\ddot{x}(t) = u(t - T_d), \tag{5.4a}$$
$$y(t) = x(t). \tag{5.4b}$$

We now simulate the discrete time version of the system (5.4) using a sample time of 10 ms and delay of 25 samples. The closed-loop system controlled with MPC results in performance

loss and permanent oscillations (See Figure (5.1)). Using delayed MPC scheme, we achieve stable closed-loop behavior (See Figure (5.2)).

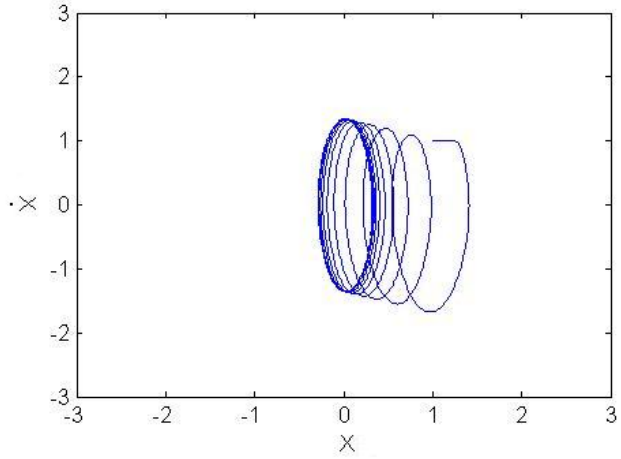

Figure 5.1.: Delayed Double Integrator without delayed MPC scheme

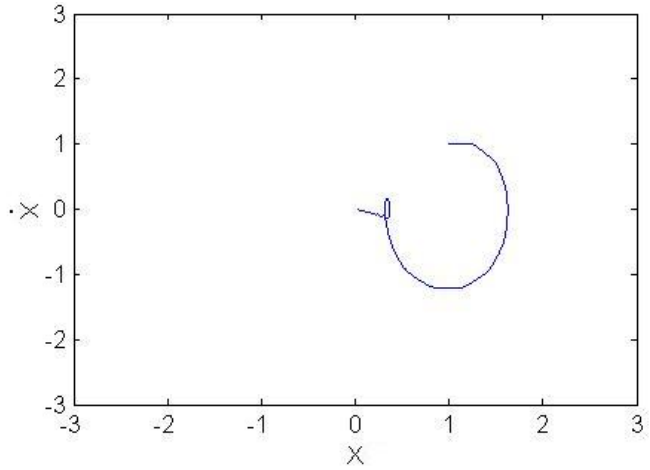

Figure 5.2.: Delayed Double Integrator with delayed MPC scheme

5.2. MPC Scaling

Let us define the following transformation

$$\tilde{x} = T_x x \Leftrightarrow x = T_x^{-1}\tilde{x}, \tag{5.5a}$$
$$\tilde{u} = T_u u \Leftrightarrow u = T_u^{-1}\tilde{u}, \tag{5.5b}$$
$$\tilde{y} = T_x y \Leftrightarrow y = T_y^{-1}\tilde{y}. \tag{5.5c}$$
$$\tag{5.5d}$$

The system (4.2) can then be transformed in the following way

$$T_x^{-1}\tilde{x}(k+1|k) = \mathcal{A}_{k,t}T_x^{-1}\tilde{x}(k) + \mathcal{B}_{k,t}T_u^{-1}\tilde{u}(k) + d_{k,t}, \tag{5.6a}$$
$$T_y^{-1}\tilde{y}(k) = \mathcal{C}_{k,t}T_x^{-1}\tilde{x}(k) + \mathcal{D}_{k,t}T_u^{-1}\tilde{u}(k) + e_{k,t}. \tag{5.6b}$$

Which can be written as

$$\tilde{x}(k+1|k) = T_x\mathcal{A}_{k,t}T_x^{-1}\tilde{x}(k) + T_x\mathcal{B}_{k,t}T_u^{-1}\tilde{u}(k) + T_x d_{k,t}, \tag{5.7a}$$
$$\tilde{y}(k) = T_y\mathcal{C}_{k,t}T_x^{-1}\tilde{x}(k) + T_y\mathcal{D}_{k,t}T_u^{-1}\tilde{u}(k) + T_y e_{k,t}. \tag{5.7b}$$

The state space (4.2) can then be written as

$$x(k+1|k) = \tilde{\mathcal{A}}_{k,t}x(k) + \tilde{\mathcal{B}}_{k,t}u(k) + \tilde{d}_{k,t}, \tag{5.8a}$$
$$y(k) = \tilde{\mathcal{C}}_{k,t}x(k) + \tilde{\mathcal{D}}_{k,t}u(k) + \tilde{e}_{k,t}, \tag{5.8b}$$
$$\tilde{\mathcal{A}}_{k,t} = T_x\mathcal{A}_{k,t}T_x^{-1}, \tag{5.8c}$$
$$\tilde{\mathcal{B}}_{k,t} = T_x\mathcal{B}_{k,t}T_u^{-1}, \tag{5.8d}$$
$$\tilde{\mathcal{C}}_{k,t} = T_y\mathcal{C}_{k,t}T_x^{-1}, \tag{5.8e}$$
$$\tilde{\mathcal{D}}_{k,t} = T_y\mathcal{D}_{k,t}T_u^{-1}. \tag{5.8f}$$

5.3. Open Loop Prediction

At each instant of time, the optimization problem (2.4) yields open loop input trajectory. Respectively, open loop output trajectory can be calculated for the open loop optimal control inputs using (2.1). Open loop predicted trajectories are useful tools to verify MPC strategies.

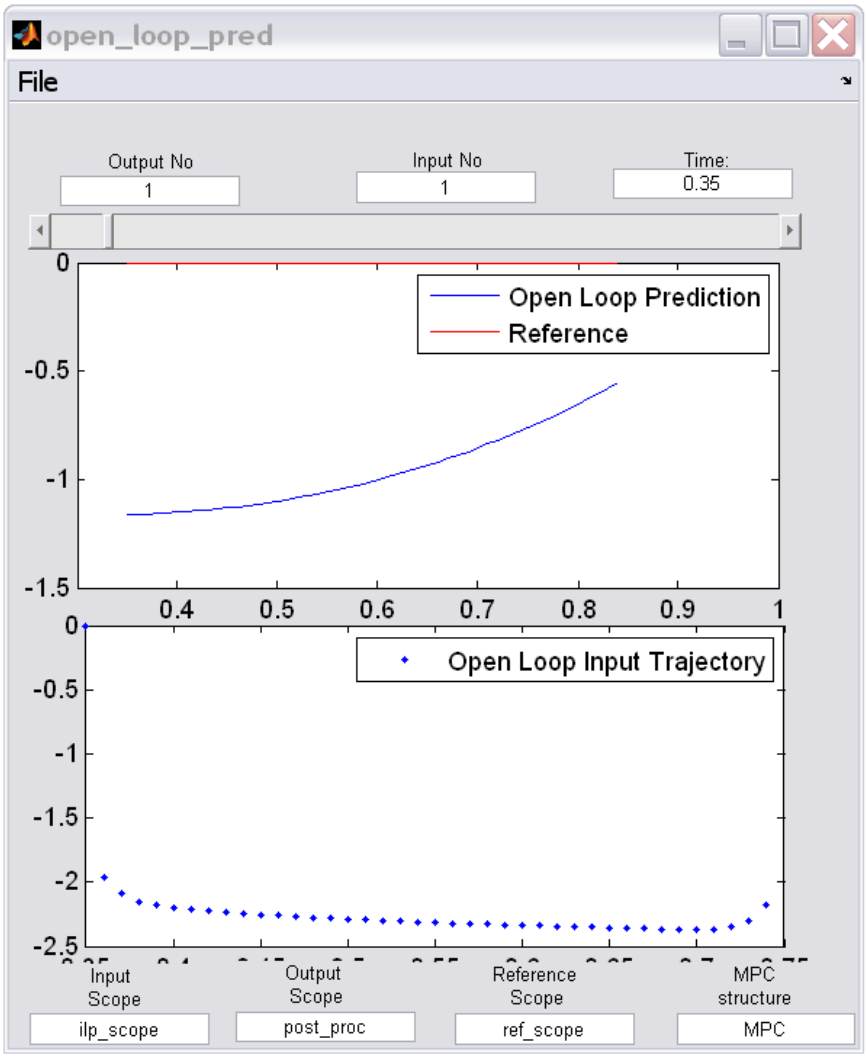

Figure 5.3.: Open Loop Prediction using MPC Toolbox

5.4. Offset Free MPC

Non-zero steady state tracking error occurs if MPC is used to control a plant in the presence of model mismatch. Model mismatch, among others, can happen when approximating nonlinear plants with linear time variant models, it can also arise from unmodeled disturbance dynamics. We will show that by modifying the MPC formulation, we can guarantee zero steady state offset ([14], [5]).

As a motivating example, let us consider a stirred-tank reactor ([8]). The plant can be described by the following state space representation

$$\frac{dc}{dt} = \frac{F_0(c_0 - c)}{\pi r^2 h} - k_0 c exp(-\frac{E}{RT}), \tag{5.9a}$$

$$\frac{dT}{dt} = \frac{F_0(T_0 - T)}{\pi r^2 h} + \frac{-\triangle H}{\rho C_p} k_0 c exp(-\frac{E}{RT}) + \frac{2Uh}{r\rho C_p}(T_c - T), \tag{5.9b}$$

$$\frac{dh}{dt} = \frac{F_0 - F}{\pi r^2}. \tag{5.9c}$$

The output states represents molar concentration of the tank c, tank temperature T and level of the tank h. The manipulated variables are coolant temperature T_c the tank outlet flow F. We consider the inlet flow F_0 as an unmeasured disturbance acting on the plant.

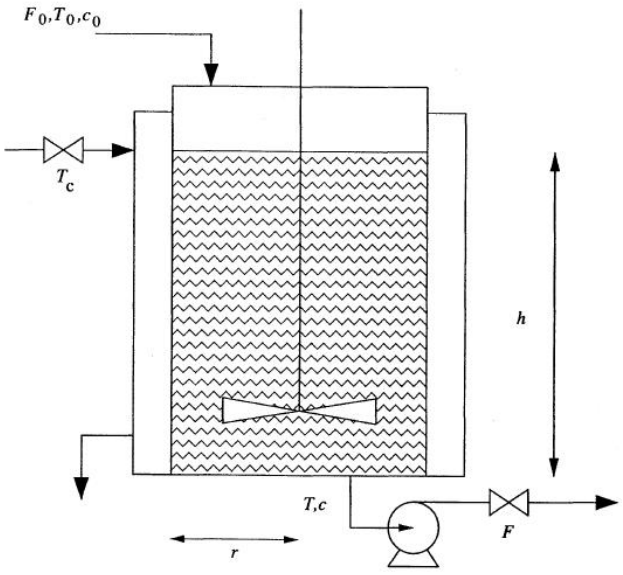

Figure 5.4.: Stirred Tank reactor [5]

Assuming a sampling time of one minute, the plant is linearized around the operation point

to yield the following LTI model

$$x(k+1) = Ax(k) + Bu(k) + B_p p, \tag{5.10a}$$
$$y(k) = Cx(k). \tag{5.10b}$$
$$\tag{5.10c}$$

where the state space is given as

$$A = \begin{bmatrix} 0.2511 & -3.36810^{-3} & -7.05610^{-4} \\ 11.06 & 0.3296 & -2.545 \\ 0 & 0 & 1 \end{bmatrix}, \tag{5.11a}$$
$$B = \begin{bmatrix} -5.42610^{-3} & 1.53010^{-5} \\ 1.297 & 0.1218 \\ 0 & -6.59210^{-2} \end{bmatrix}, \tag{5.11b}$$
$$B_p = \begin{bmatrix} -1.76210^{-5} \\ 7.78410^{-2} \\ 6.59210^{-2} \end{bmatrix}, \tag{5.11c}$$
$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.11d}$$
$$\tag{5.11e}$$

We assume that the inlet flow rate (p) is not measured. Assume that a constant inlet flow enters the tank from time $t = 10$. The objective of control is to regulate the system to zero steady state. We simulate the system (5.10) while it is controlled with MPC. We can see that the output states has nonzero steady state offset.

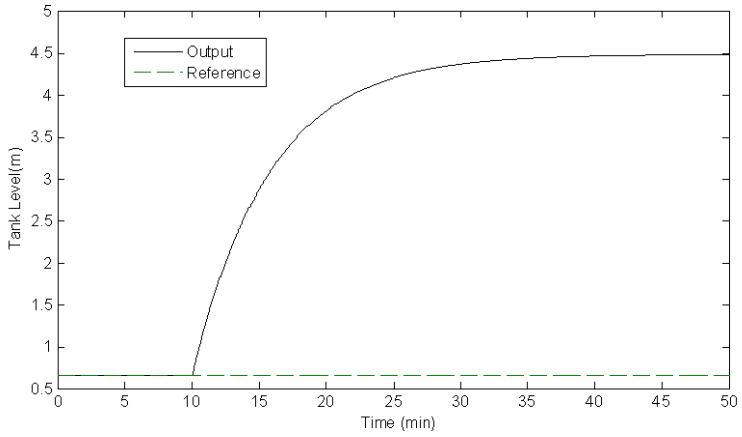

Figure 5.5.: Stirred Tank reactor, unmeasured disturbance causes non-zero offset [5].

Using Open-Loop prediction tool, we find that the plant is not changing as predicted by MPC because of model mismatch.

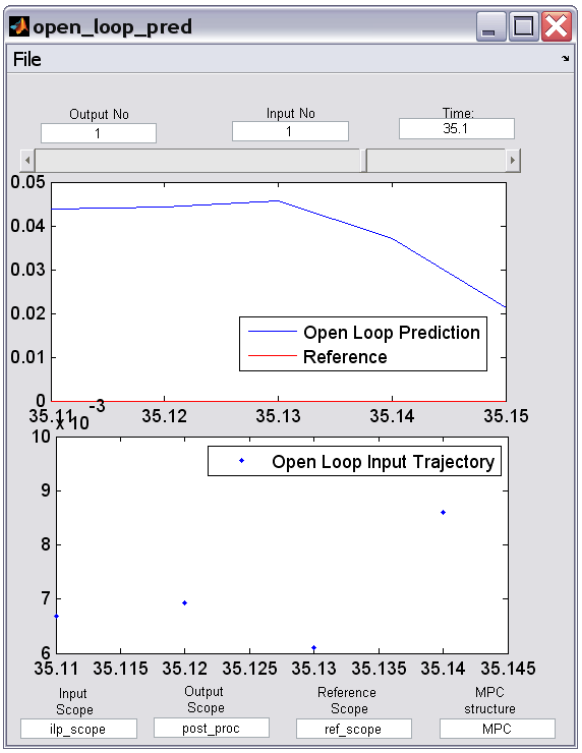

Figure 5.6.: Stirred Tank reactor, Open-Loop Prediction does not conform to plant

We assume that tracking outputs y_{tr} is a subset of measured outputs by defining

$$y_{tr} = Hy(t). \tag{5.12a}$$

Since, we are interested in steady states without loss of generality, we use the LTI model (3.1). The plant is assumed to be observable and controllable. We augment the plant model with additional states equal to the number of measured outputs

$$x(t+1) = A\,x(t) + B\,u(t) + B_{ud}\,ud(t), \tag{5.13a}$$
$$ud(t+1) = ud(t), \tag{5.13b}$$
$$y(t) = C\,x(t) + C_{ud}\,ud(t). \tag{5.13c}$$

The Hautus observability condition for the system (5.13) to be observable, requires the matrix

$$\begin{bmatrix} A - I & B_{ud} \\ C & C_{ud} \end{bmatrix}, \tag{5.14}$$

to have full rank. This assumption in addition to the stability of the observer requires us to have maximum number of p unmeasured disturbance states (5.15). We restrict our attention to

the case where the number of augmented states is equal to the number of measured outputs. Assume our system (5.13) satisfies

$$\begin{bmatrix} A-I & B \\ HC & 0 \end{bmatrix} \begin{bmatrix} \hat{x_\infty} \\ u_\infty \end{bmatrix} = \begin{bmatrix} -B_{ud}\ \hat{ud}_\infty \\ y_{ref,\infty} - HC_{ud}\ \hat{ud}_\infty \end{bmatrix}, \tag{5.15}$$

where u_∞ and $y_{ref,\infty}$ are steady state inputs and references, respectively. It can be shown that the observer based on the system (5.15) satisfies offset-free control [14]. Now we define the change of variables

$$\tilde{u}(t) = u_\infty(t) + u(t), \tag{5.16a}$$
$$\tilde{y}_{ref}(t) = x_\infty(t). \tag{5.16b}$$

For the transformed reference and control input, we modify the MPC formulation (A.2) to

$$\mathcal{U}^*(t) = \underset{\mathcal{U}(t)}{\operatorname{argmin}}\, J = \sum_{i=t}^{t_f} [y_{tr} - \tilde{y}_{ref}]' \mathcal{Q} [y_{tr} - \tilde{y}_{ref}] + [\tilde{u}(t+i)]' \mathcal{S} [\tilde{u}(t+i)] + [\triangle u(t+i)]' \mathcal{R} [\triangle u(t+i)] \tag{5.17a}$$
$$Subjectto: \tag{5.17b}$$
$$x(t+i|t) = A\ x(t+i-1|t) + B\ u(t+i-1|t) + B_d\ d(t+i-1|t) \tag{5.17c}$$
$$y(t+i|t) = C\ x(t+i|t) + D\ u(t+i|t) + D_d\ d(t+i|t) \tag{5.17d}$$
$$x(t+i|t) \in \mathcal{X}(t+i|t) \tag{5.17e}$$
$$\tilde{u}(t+i|t) \in \tilde{\mathcal{U}}(t+i|t) \tag{5.17f}$$
$$y(t+i|t) \in \mathcal{Y}(t+i|t) \tag{5.17g}$$

Assuming MPC to be unconstrained at steady state, the formulation (5.17) guarantees offset-free control of tracking variables.

Using the offset-free MPC scheme, we will have the offset free outputs for the system (5.9)

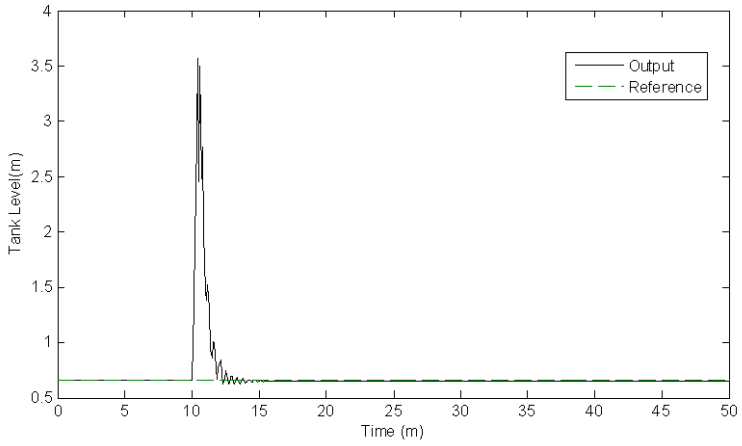

Figure 5.7.: Stirred Tank reactor tank level, offset-free MPC scheme [5]

6. Results and Conclusion

In this chapter, we apply MPC Toolbox on a rich application taken from [9]. The plant to be controlled is a vehicle. A predefined steering maneuver is to be tracked while delivering the requested break forces to the wheels. MPC enables us to maximize the regenerative breaking while preserving the vehicle stability.

We use the plant model as the nonlinear vehicle model taken from [4], it can be written as

$$m\ddot{y} = -m\dot{x}\dot{\psi} + \sum_{*\in\{f,r\}} \sum_{\bullet\in\{l,r\}} F_{y*,\bullet}, \tag{6.1a}$$

$$m\ddot{x} = m\dot{y}\dot{\psi} + \sum_{*\in\{f,r\}} \sum_{\bullet\in\{l,r\}} F_{x*,\bullet}, \tag{6.1b}$$

$$I\ddot{\psi} = a \sum_{\bullet\in\{l,r\}} F_{yf,\bullet} - b \sum_{\bullet\in\{l,r\}} F_{yr,\bullet} + c\Big(\sum_{*\in\{f,r\}} F_{x*,r} - \sum_{*\in\{f,r\}} F_{x*,l}\Big), \tag{6.1c}$$

where $* \in \{f, r\}$ denotes the front and rear axles, $\bullet \in \{l, r\}$ denotes left and right side of vehicle, $\dot{y}$ is lateral velocity, $\dot{x}$ is longitudinal velocity, $\dot{\psi}$ is yaw rate, m is mass of the car, I is inertial moment along the vertical axis, a and b are the distance of center of gravity from front and rear axles, respectively. c is distance of the left and right wheels from longitudinal axis.

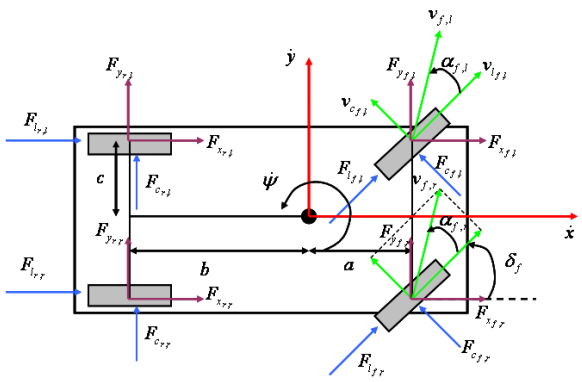

Figure 6.1.: Simplified vehicle dynamical model ([9])

The objective of the control is tracking the specified maneuver, while delivering the total requested breaking forces (F_{rqst}). The outputs of the controller are the breaking forces on each wheel. The constraints are defined as follows, the commanded break forces are constrained to add to the requested force. Yaw rate tracking error and lateral jerk should be bounded. Moreover, we add a constraint that 70 percent of break forces should act on the front wheels.

The constraints can be written as

$$ControlInputLowerBound = \mathbb{U}_l = [0\ 0\ 0\ 0\ 0] \tag{6.2a}$$
$$ControlInputUpperBound = \mathbb{U}_u = [2000\ 2000\ 1000\ 1000\ 2000] \tag{6.2b}$$
$$ControlInputRateLowerBound = \triangle\mathbb{U}_l = [-2e5\ -2e5\ -1e5\ -1e5\ -2e5] \tag{6.2c}$$
$$ControlInputRateUpperBound = \triangle\mathbb{U}_l = [2e5\ 2e5\ 1e5\ 1e5\ 2e5] \tag{6.2d}$$
$$\begin{bmatrix} F_{rqst}.1_{1\times H_c} \\ 0_{1\times H_c} \end{bmatrix} \leq [kron(eye(3),[11111]); kron(eye(3),[11-2.3-2.30])] \begin{bmatrix} u(t+1|t) \\ u(t+2|t) \\ u(t+1|t) \\ \vdots \\ u(t+H_g c|t) \end{bmatrix} \leq \begin{bmatrix} F_{rqst}.1_{1\times H_c} \\ 0_{1\times H_c} \end{bmatrix} \tag{6.2e}$$

where $kron(.)$ is Kronecker product, and $eye(x)$ represents identity matrix of size x.

The weighting parameters are defined as

$$TrackingErrorWeight = Q = 1 \tag{6.3a}$$
$$ControlInputRateWeight = R = [10000\ 10000\ 10000\ 10000\ 1e-5] \tag{6.3b}$$
$$ControlInputWeight = S = [5000\ 5000\ 4000\ 4000\ -1] \tag{6.3c}$$

The scaling parameters are

$$T_u = [1/1200\ 1/1200\ 1/1200\ 1/1200\ 1/1200] \tag{6.4a}$$
$$T_y = [1\ 1/4] \tag{6.4b}$$
$$T_x = [1/2\ 1/20\ 1/0.14] \tag{6.4c}$$
$$S = [5000\ 5000\ 4000\ 4000\ -1] \tag{6.4d}$$

The MPC will yield the following break forces, see figure (6.2)). The resulting yaw rate given in figures ((6.3) and (6.4)) shows consistency with previous controllers([9]) on the same problem formulation.

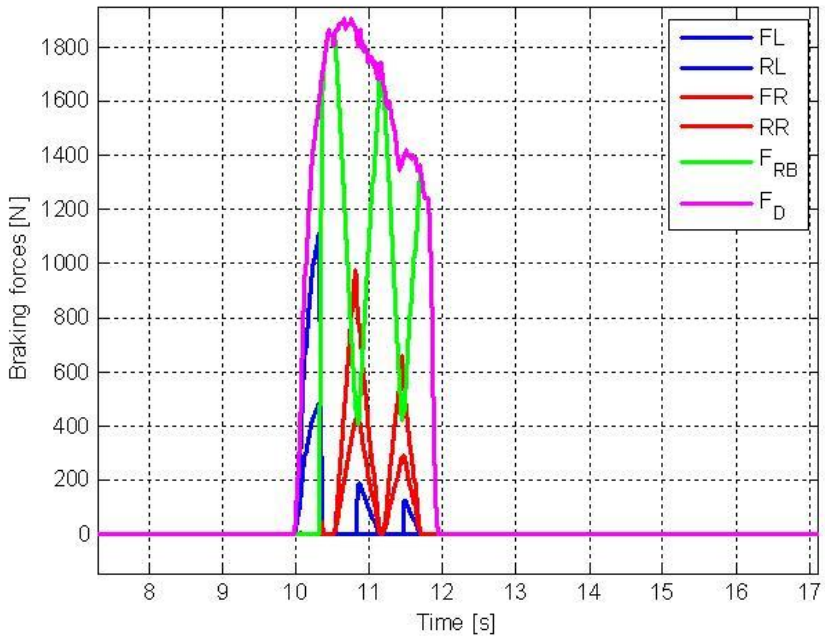

Figure 6.2.: MPC yields the commanded forces

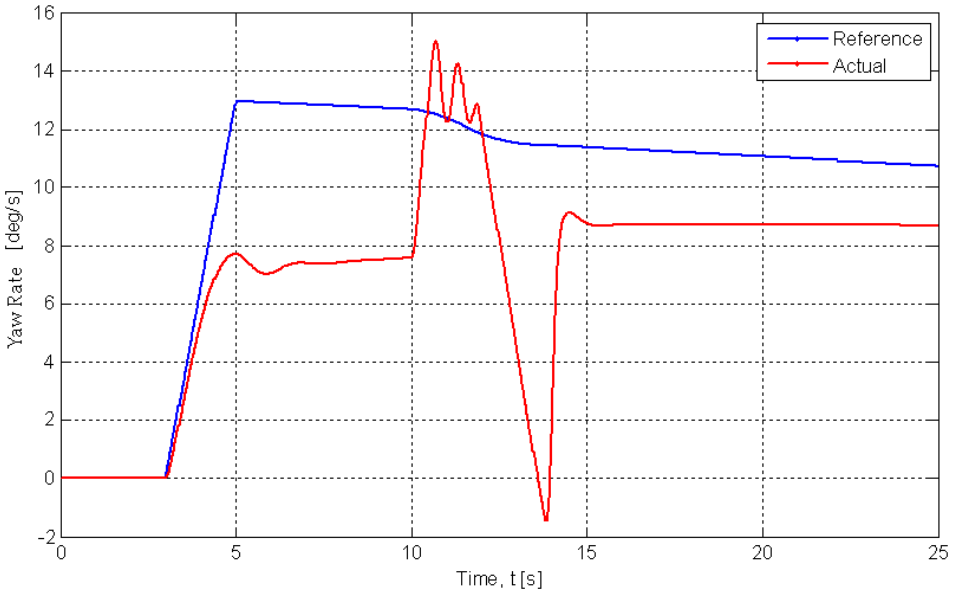

Figure 6.3.: Output yaw rate and Reference

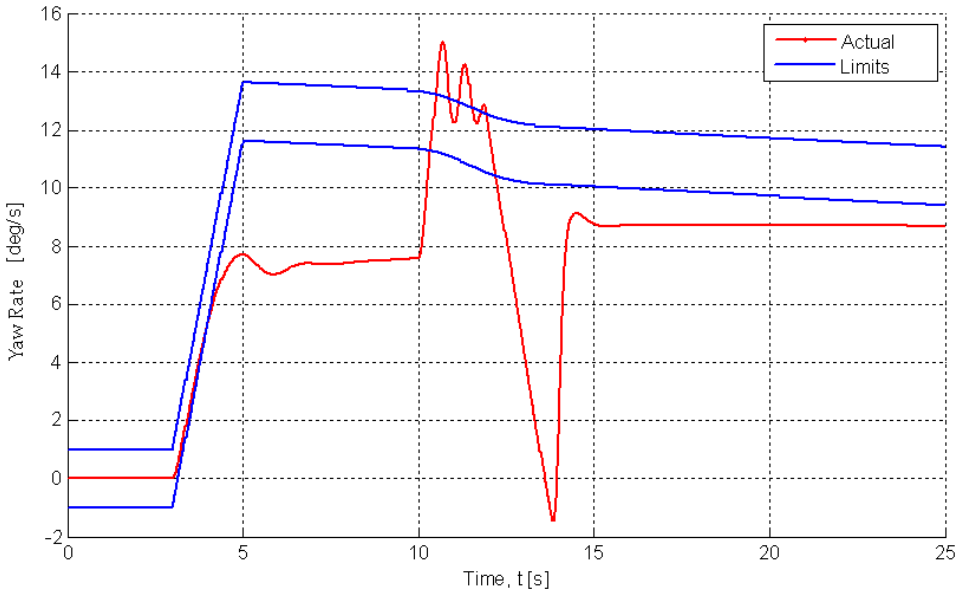

Figure 6.4.: Output yaw rate and limits

6.1. Conclusion

The results in the previous section shows consistency of MPC Toolbox with previous MPC controllers [9]. MPC Toolbox is also compared with the MATLAB's official version of MPC Toolbox [7] on a double integrator, (See appendix), which verifies that we have similar results. We thereby can assert that our Toolbox gives consistent results under different MPC schemes. We hope that the MPC Toolbox is soon released and tested with more users. This can help us to find possible needs or resolutions for future versions of the Toolbox.

Bibliography

[1] Arban Alaniz. Model predictive control with application to real-time hardware and a guided parafoil. *Massachusetts Institute of Technology*, 2004.

[2] Manfred Morari Alberto Bemporad, Francesco Borrelli. Model predictive control based on linear programming the explicit solution. *Transacations on Autonatic Control, Vol 47 No 12, IEEE*, 2002.

[3] C. V. Rao P. O. M. Scokaert D. Q. Mayne, J. B. Rawlings. Constrained model predictive control: Stability and optimality. *Automatica 36*, 2000.

[4] Paolo Falcone. Nonlinear model predictive control for autonomous vehicles. *Universita del Sannio, Dipartimento di Ingegneria, Piazza Roma 21, 82100, Benevento, Italy*, 2007.

[5] James B. Rawlings Gabriele Pannocchia. Disturbance models for offset-free model predictive control. *AIChE Journal*, 2003.

[6] Holger Diedam Boris Houska Aude Perrin Thomas Wiese Hans Joachim Ferreau, Eckhard Arnold. qpoases users manual. 2009.

[7] N. Lawrence Ricker Manfred Morari. Model predictive control toolbox, for use with matlab. 1998.

[8] Dale E. Seborg Micheal A.Henson. Nonlinear process control. 1997.

[9] S. Solyom P. Falcone, S. Khoshfetrat Pakazad. Predictive approaches to rear axle regenerative braking control in hybrid vehicles. *48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, 2009.

[10] Michael A.. Saunders Margaret H. Wright Philip E. Gill, Walter MurraY. User's guide for npsol (version 5.o): A fortran package for nonlinear programing. 1998.

[11] Walter MurraY Michael A.. Saunders Margaret H. Wright Philip E. Gill, p Sven J. Hammarlingt. User's guide for lssol (version 1.o): A fortran package for constrained linear. least squares and convex quadratic programing. 1986.

[12] Frank Allgower Rolf Findeisen. Computational delay in nonlinear model predictive control. *In Proc. Int. Symp. Adv. Control of Chemical Processes, Hong Kong*, 2004.

[13] Georgios Papafotiou Tobias Geyer and Manfred Morari. Model predictive control in power electronics: A hybrid systems approach. *Decision and Control, European Control Conference*, 2005.

[14] Manfred Morari Urban Maeder, Francesco Borrelli. Linear offset-free model predictive control. *Automatica 45(10)*, 2009.

A. MPC Toolbox

In This chapter, we provide an overview of the MPC Toolbox. We describe how to setup the model, specify the horizons, weights, constraints, scaling and estimator.

For a continuous time system described by the following state space formulation

$$\begin{aligned} \dot{x} &= f(x(t), u(t)) \\ y &= h(x(t), u(t)) \end{aligned} \tag{A.1}$$

The output vector y is partitioned into $[y_{tr}, y_c, y_{sc}]^T$, Where y_{tr} is the tracking output, y_c is the hard constrained output and y_{sc} is the soft constrained output.

The task is to control the system to follow special trajectory, and conforms to some constraints on outputs or control inputs, the MPC formulation can be written as

$$\begin{aligned} J^*(x(t), u^*(t), ..., u^*(t+H_c)) = \min_{\triangle u(t),...,\triangle u(t+H_c)} \\ \sum_{i=0}^{H_p-1} ||\mathcal{Q}[y_{tr}(t+i|t) - y_{ref}(t+i|t)]||_2^2 + ||\mathcal{S}u(t+i|t)||_2^2 + ||\mathcal{R}\triangle u(t+i|t)||_2^2 \end{aligned} \tag{A.2}$$

Subject to

$$x(t|t) = x(t) \tag{A.3a}$$
$$u[t+i+1|t] = u[t+i|t] + \triangle u[t+i|t] \quad \forall i \in \{0, ..., H_c - 1\} \tag{A.3b}$$
$$\triangle \mathbb{U}_l(t+i) \leq \triangle u[t+i|t] \leq \triangle \mathbb{U}_u(t+i) \quad \forall i \in \{0, ..., H_c - 1\} \tag{A.3c}$$
$$\mathbb{U}_l(t+i) \leq u[t+i|t] \leq \mathbb{U}_u(t+i) \quad \forall i \in \{1, ..., H_c\} \tag{A.3d}$$
$$\mathbb{Y}_{c,l}(t+i) \leq y_c[t+i|t] \leq \mathbb{Y}_{c,u}(t+i) \ \ \forall i \in \{1, ..., H_p\} \tag{A.3e}$$
$$\mathbb{Y}_{sc,l}(t+i) \leq y_{sc}[t+i|t] \leq \mathbb{Y}_{sc,u}(t+i) \ \ \forall i \in \{1, ..., H_p\} \tag{A.3f}$$
$$\mathbb{E}_l \leq y_{tr}[t+i|t] - y_{ref}[t+i|t] \leq \mathbb{E}_u \ \ \forall i \in \{1, ..., H_p\} \tag{A.3g}$$
$$G_l \leq E[x(t)...x(t+H_{gc})] + F[u(t)...u(t+H_{gc})] \leq G_u \tag{A.3h}$$

In the following sections, we will describe how this formulation can be embedded in the MPC Toolbox.

A.1. Software Installation

MPC toolbox requires MATLAB and a compatible C compiler. It is required that MATLAB includes Simulink and Real Time Workshop. The toolbox is tested with Visual Studio 2005, and MATLAB R2008a up to R2010a.

Download and extract the compressed archive *MPC Toolbox_v1.zip*, then use MATLAB addpath command to add the toolbox and its subfolder to MATLAB's default search path,i.e. run

$>> addpath(genpath('MPCToolbox\_v1'))$
$>> savepath$

Alternatively, you can use File $>$ Set Path to open the Set Path dialog box, and add the toolbox and its subfolders to MATLAB's search path and save it. Make sure the compiler is set correctly in MATLAB using

$>> mex - setup$

You can now start with one of the examples of the graphical user interface using

$>> mpcdesign$

When running the toolbox for the first time you will need to provide the directory where toolbox is located,e.g. where the file *MPC.C* is located in. It is recommended to explore the GUI and examples before starting your own project.

A.2. MPC Model

We here present a double integrator example for illustration on how to use the correct syntax. For the Nonlinear MPC formulation given as

$$\begin{aligned} \dot{x} &= f(x(t), u(t)) \\ y &= h(x(t), u(t)) \end{aligned} \tag{A.4}$$

The model could be written in the form of Nonlinear MPC model formulation as

no_of_states = 2;
no_of_inputs = 1;
no_of_disturbance = 1;
no_of_outputs = 2;
no_hard_constrained_outputs = 0;%optional
no_soft_constrained_outputs = 0;%optional
sample_time = 0.01;%optional
time_delay = [0];
no_of_params = 1;
%%StateSpaceModel
dx1 = x2;
dx2 = u1;
out1 = x1;
out2 = x2;
%%LinearizedStateSpaceModel

The reserved words for the formulation are 'x_N,u_N,d_N,p_N,dx_N,out_N', which corresponds to states, control input, measured disturbance, parameters, derivative function(f), output function h.

The model also could be described by Linear Time Invariant system as

$$\begin{aligned} x(t+1) &= A\,x(t) + B\,u(t) + B_d\,d(t) \\ y(t) &= C\,x(t) + D\,u(t) + D_d\,d(t) \end{aligned} \tag{A.5}$$

which can be fully described for MPC design tool by the appropriate definition of matrices

$$\begin{aligned} A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, B_d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\ C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, D_d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \end{aligned} \tag{A.6}$$

The Linear Time Variant form described as

$$\begin{aligned} x(t+i+1) &= \mathcal{A}_{t+i,t}x(t+i) + \mathcal{B}_{t+i,t}u(t+i) + d_{t+i,t} \\ y(t+i) &= \mathcal{C}_{t+i,t}x(t+i) + \mathcal{D}_{t+i,t}u(t+i) + e_{t+i,t} \end{aligned} \tag{A.7}$$

the corresponding model compatible for MPC Toolbox is defined as
no_of_states = 2;
no_of_inputs = 1;
no_of_disturbance = 1;
no_of_outputs = 2;
no_hard_constrained_outputs = 0;%optional
no_soft_constrained_outputs = 0;%optional
sample_time = 0.01;%optional
time_delay = [0];
no_of_params = 1;
%%StateSpaceModel
dx1 = 0;
dx2 = 0;
out1 = x1;
out2 = x2;
%%LinearizedStateSpaceModel sp_A1 = 0;
sp_A2 = 1;
sp_A3 = 0;
sp_A4 = 0;
sp_B1 = 0;
sp_B2 = 1;
sp_F1 = 0;
sp_F2 = 0;
sp_C1 = 1;
sp_C2 = 0;
sp_C3 = 0;
sp_C4 = 1;
sp_D1 = 0;
sp_D2 = 1;

To specify the corresponding model for MPC Toolbox, you should specify the MPC formulation and appropriate model. Using design tool you can choose MPC formulation (See figure (A.1)) and choose the model (See figure (A.2)).

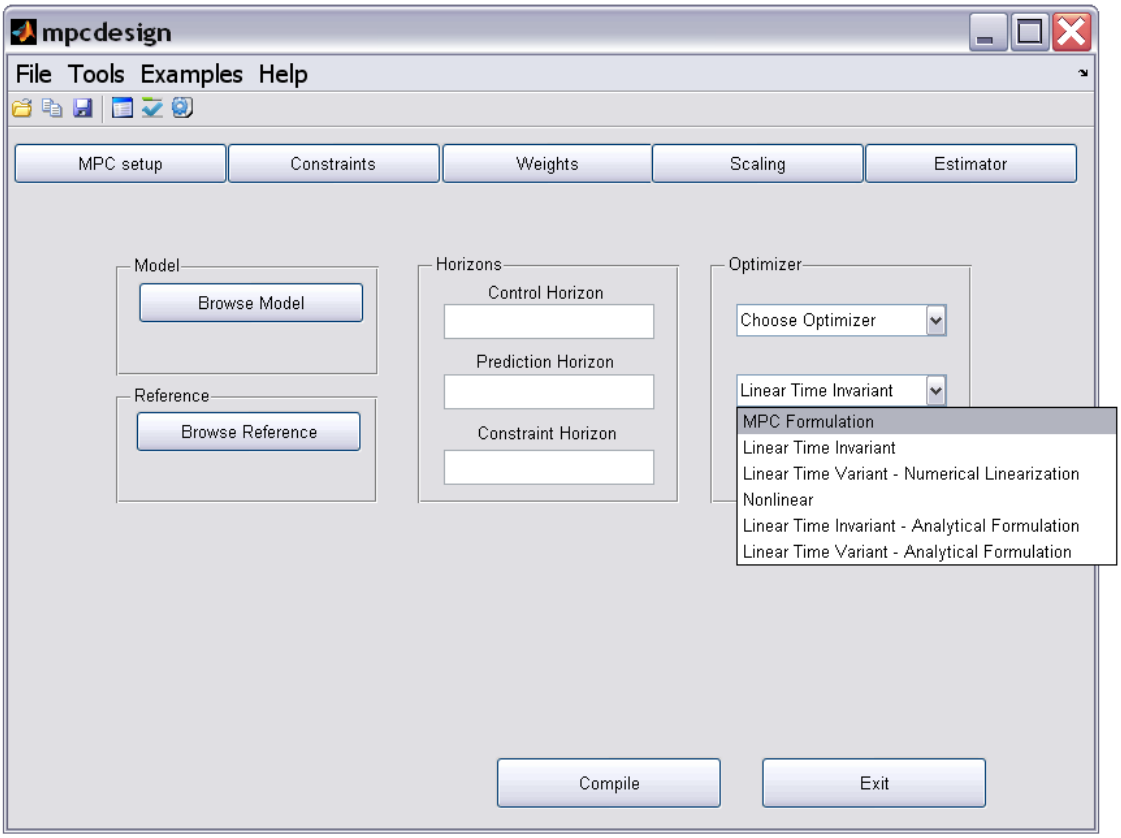

Figure A.1.: MPC formulation can be specified in design tool

Note: For a nonlinear model you can specify

- *LinearTimeVariant – NumericalLinearization*
- *Nonlinear*

where for Linear Time Invariant model you can specify

- *LinearTimeInvariant*
- *LinearTimeInvariant – AnalyticalFormulation*

and for Linear Time Variant model you can specify

- *LinearTimeVariant – AnalyticalFormulation*

The distinction between the analytical LTI and LTV formulation is due to the assumption that LTV formulation is linearized around a nonlinear trajectory.

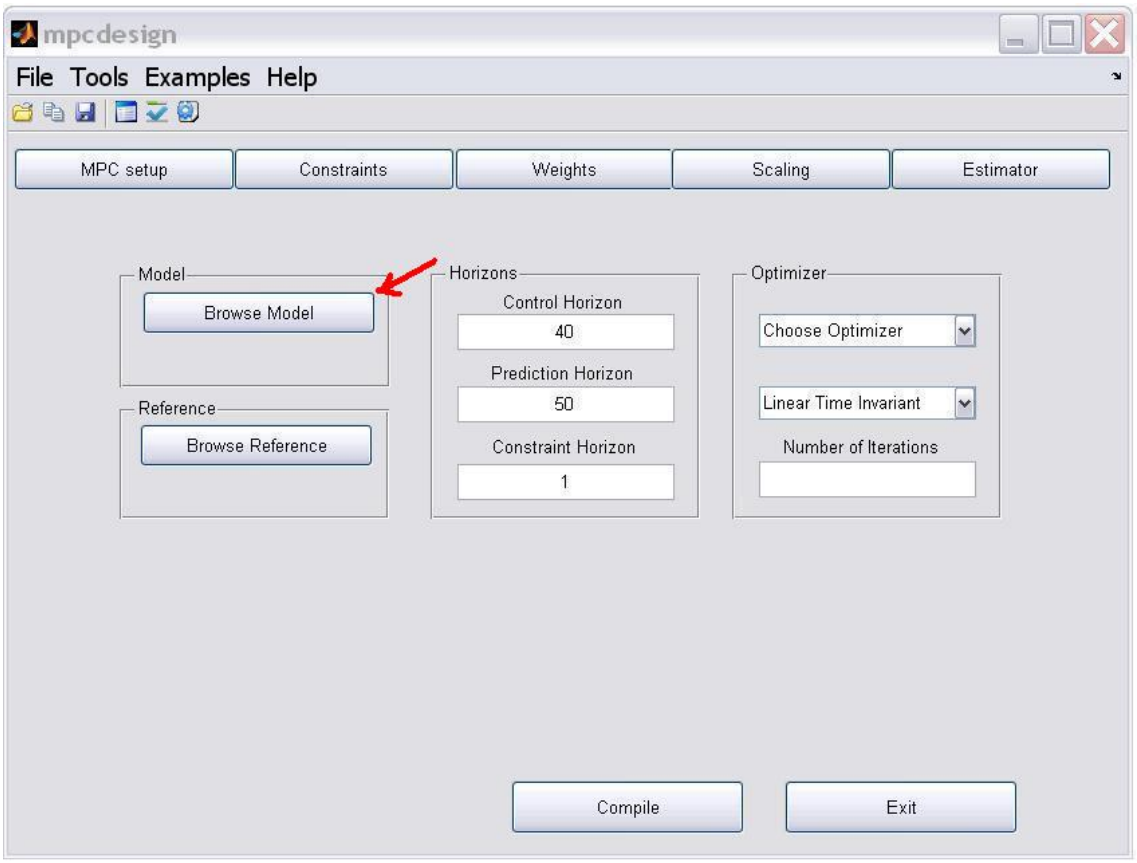

Figure A.2.: Using design tool you can specify you model

A.3. MPC Objective

We then specify the MPC objective defined as

$$\mathcal{U}^*(t) = \underset{\mathcal{U}(t)}{\operatorname{argmin}} J = \sum_{i=t}^{t+Hp} [y_{tr} - y_{ref}]' \mathcal{Q} [y_{tr} - y_{ref}] + [\mathcal{U}(t)]' \mathcal{S} [\mathcal{U}(t)] + [\triangle \mathcal{U}(t)]' \mathcal{R} [\triangle \mathcal{U}(t)] \quad \text{(A.8)}$$

MPC objective is specified by indication of MPC horizons and weights. Using design tool you can specify the horizons in *MPCSetup* tab. You then should specify the prediction horizon, control horizon and constraint horizon.

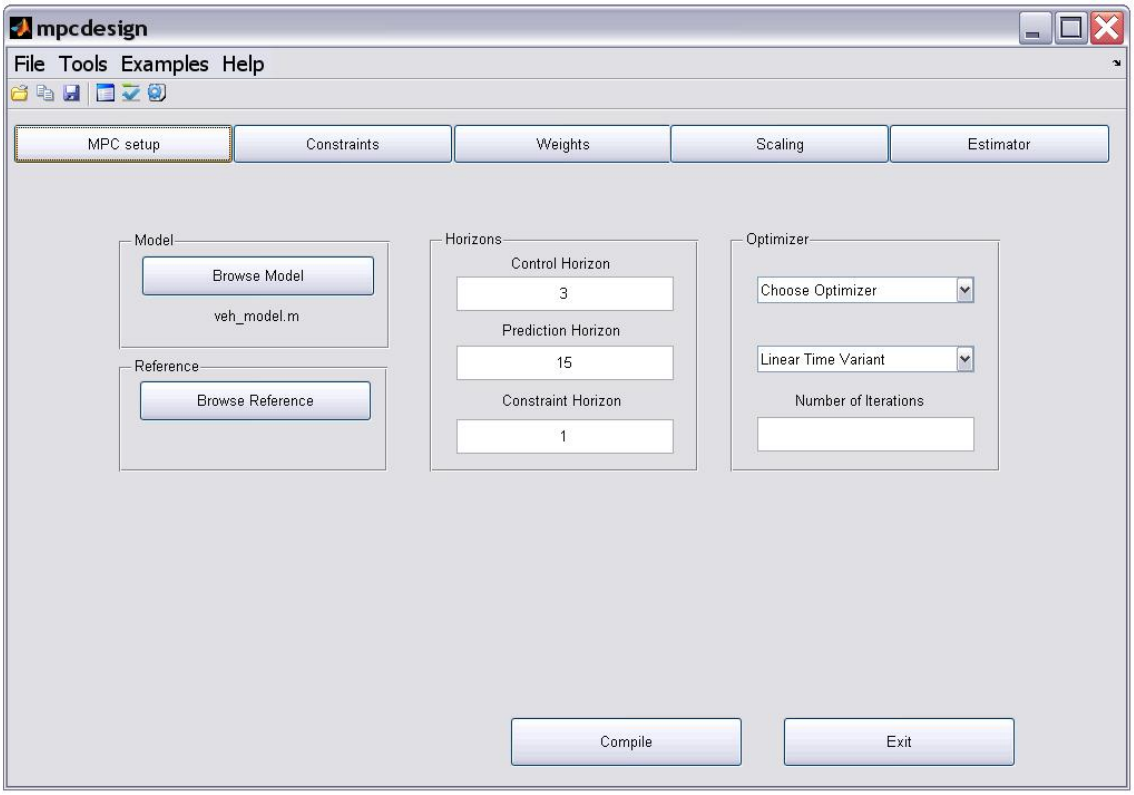

Figure A.3.: You can specify the MPC horizons in design tool

The next step to define the MPC objective is to define the MPC weights. You can specify the MPC weights inside the design tool under MPC weights tab.Moreover, you can specify the weight penalty on control inputs, command changes(A.4), tracking error, and slack variable,i.e. feasibility violation for soft constrained variables.

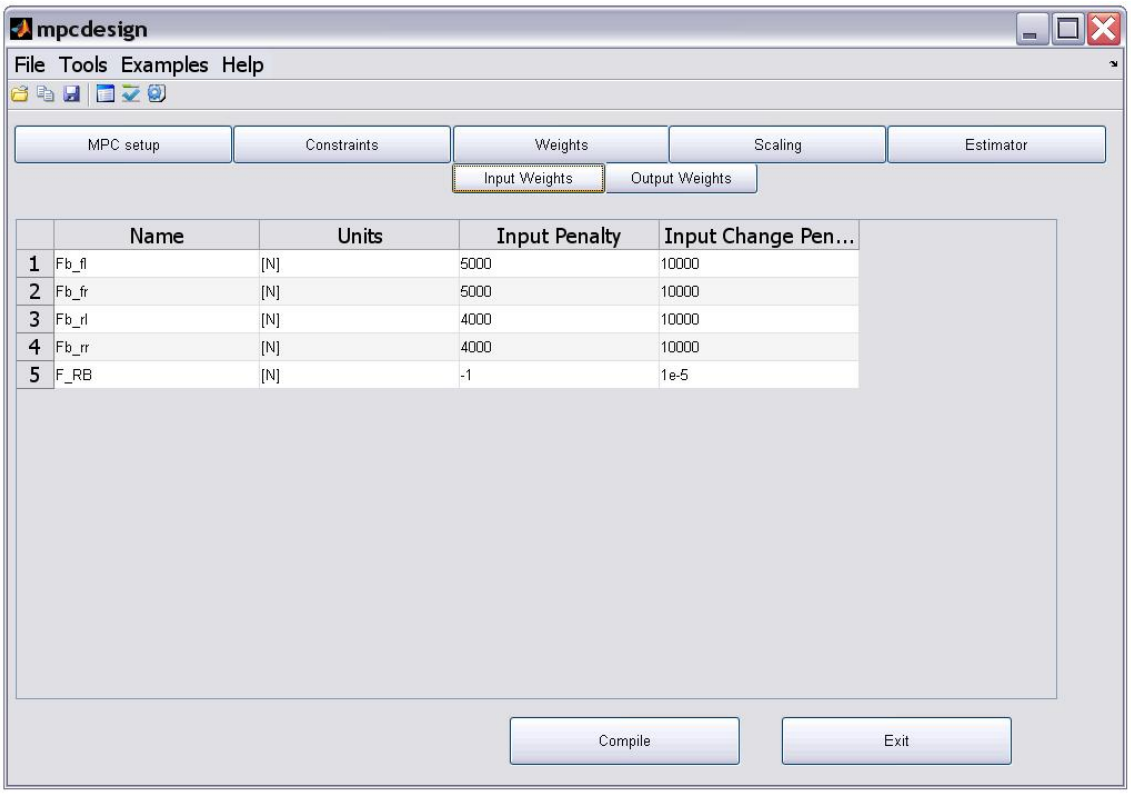

Figure A.4.: Input Penalties

A.4. MPC constraints

Next we define the MPC constraints given in the form

$$\triangle \mathbb{U}_l \leq \mathcal{U} \leq \triangle \mathbb{U}_u \;\; for \;\; \triangle \mathcal{U}(t) \cdots \mathcal{U}(t+Hu) \tag{A.9a}$$

$$\mathbb{U}_l \leq \mathcal{U} \leq \mathbb{U}_u \;\; for \;\; \mathcal{U}(t) \cdots \mathcal{U}(t+Hu) \tag{A.9b}$$

$$\mathbb{Y}_{c,l} \leq y_c \leq \mathbb{Y}_{c,u} \;\; for \;\; t \in [t, t+Hc] \tag{A.9c}$$

$$\mathbb{Y}_{sc,l} \leq y_{sc} \leq \mathbb{Y}_{sc,u} \;\; for \;\; t \in [t, t+Hc] \tag{A.9d}$$

$$\mathbb{E}_l \leq |y_{tr} - y_{ref}| \leq \mathbb{E}_u \;\; for \;\; t \in [t, t+Hc] \tag{A.9e}$$

$$\tag{A.9f}$$

Under constraints tab, you can specify control input upper and lower bound and bounds on the command changes(A.5). You can also specify the constraints on output tracking,hard constrained, and soft constrained variables. Note the constraints on tracking variable is assumed on tracking error.

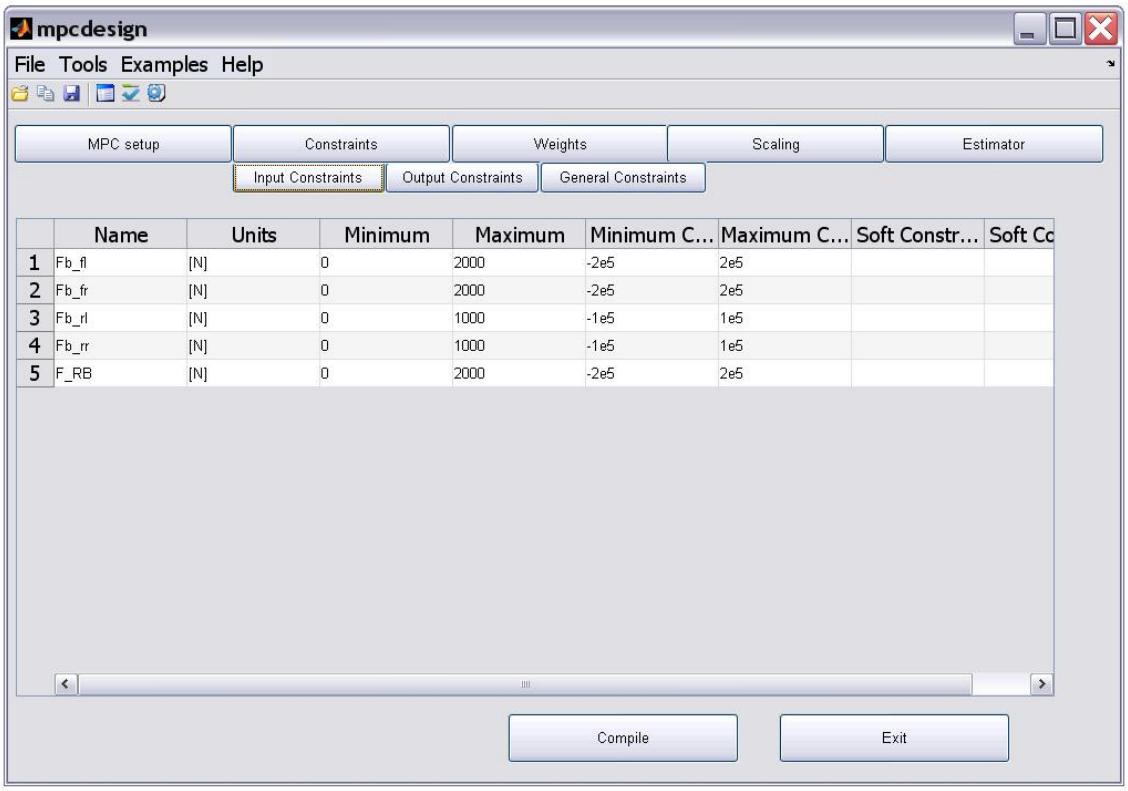

Figure A.5.: Input Constraints

General constraints has the form

$$GL - Ds \leq E \begin{bmatrix} x(t+1|t) \\ x(t+2|t) \\ x(t+1|t) \\ \vdots \\ x(t+H_g c|t) \end{bmatrix} + F \begin{bmatrix} u(t+1|t) \\ u(t+2|t) \\ u(t+1|t) \\ \vdots \\ u(t+H_g c|t) \end{bmatrix} \leq GU + Ds \tag{A.10}$$

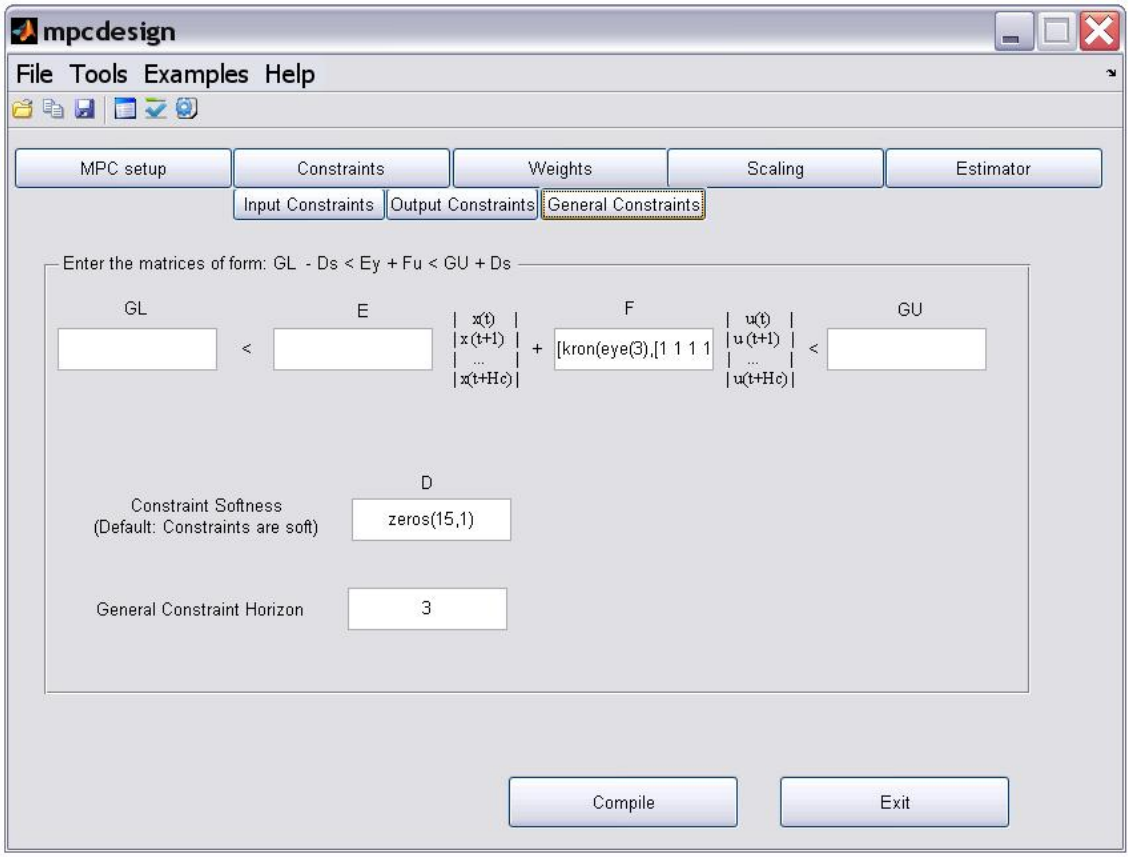

Figure A.6.: General Constraints

A.5. Real-Time Implementation

MPC Simulink Block can be used for Real-Time Hardware. The MPC Toolbox is tested successfully for dSPACE - MicroAutoBox-1401 Power Processors and for MATLAB - xPC target. We here include the necessary steps you should take when using MPC Simulink Block for Real-Time Hardware.

First, Make sure the correct Building Target is selected for Simulink. Using Simulation→ Configuration Parameters→ Real Time Workshop→ System Target file. The corresponding target for xPC target is 'xpctarget.tlc'. The corresponding target for MicroAutobox Power PC is 'rti1401.tlc'.

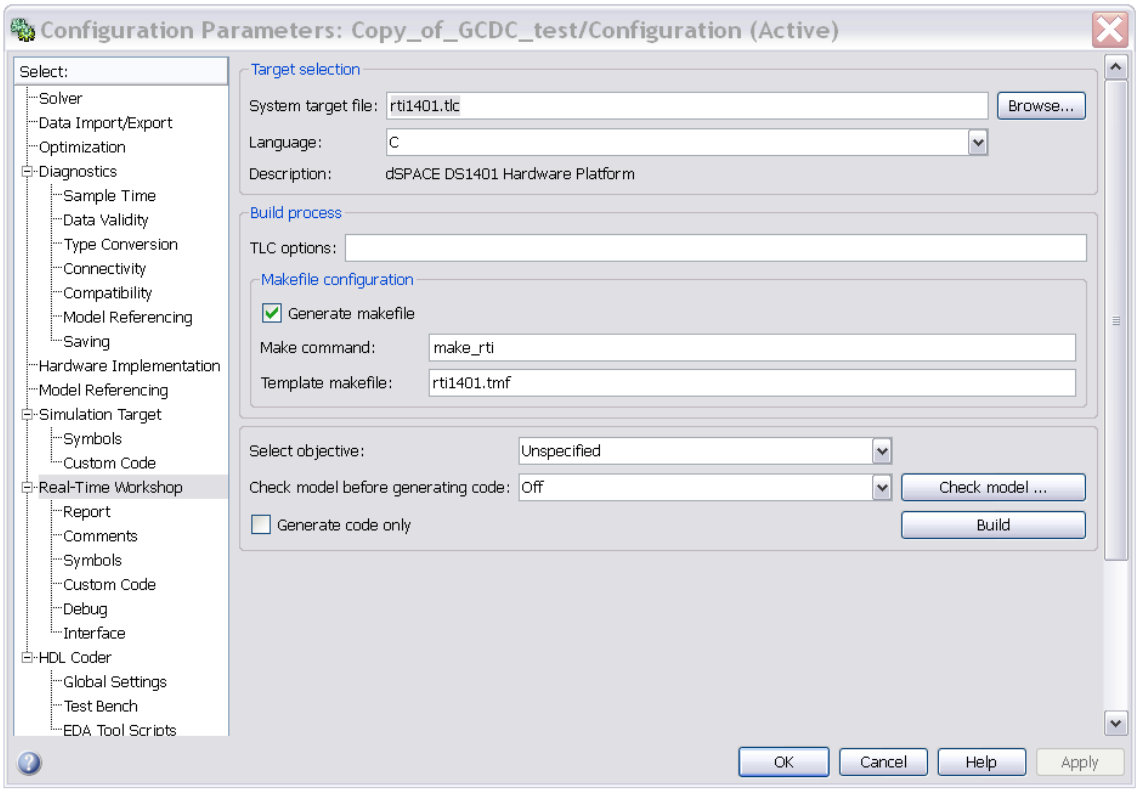

Figure A.7.: To setup the Real-Time Workshop you should choose the target correctly

Second, Make sure the MPC libraries are included inside MATLAB Simulink window: Simulation→ Configuration Parameters→Real Time Workshop → Custom Code → Libraries

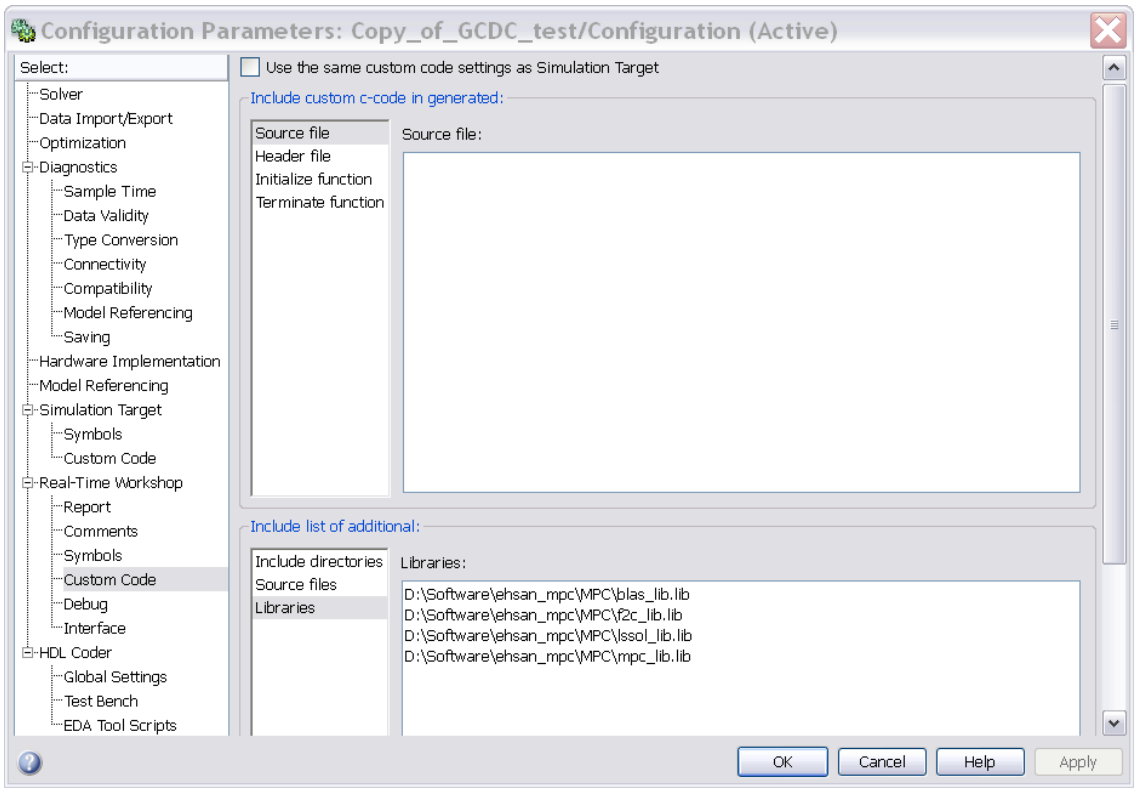

Figure A.8.: The MPC libraries should be added to build the system into the corresponding Real-Time target

Additionally, It might be necessary to use the MPC Toolbox exactly inside where your Simulink block is located.

To build for xPC Target on MATLAB version higher than 2008a the following code should be changed inside the file:
$'(MATLAB\_ROOT) \setminus toolbox \setminus rtw \setminus targets \setminus xpc \setminus xpc \setminus xpc\_vc.tmf'$
$LDFLAGS =' ...'$
Should be changed to
$LDFLAGS = .../NODEFAULTLIB : MSVCRT.lib/NODEFAULTLIB : LIBC.lib/NODEFAULTLIB : LIBCD.lib$

Having done the previously mentioned steps, you can use the MPC Simulink block inside your model and download it using Simulink Build feature to your desired target. The tests taken so far, have shown consistency with the MPC Simulations.

A.6. Advanced Features

You can specify the state, output, input scaling defined as

$$\tilde{x} = T_x x \tag{A.11a}$$
$$\tilde{u} = T_u u \tag{A.11b}$$
$$\tilde{y} = T_y y \tag{A.11c}$$
$$\tag{A.11d}$$

Using the design tool, you can specify under the scaling tab as follows

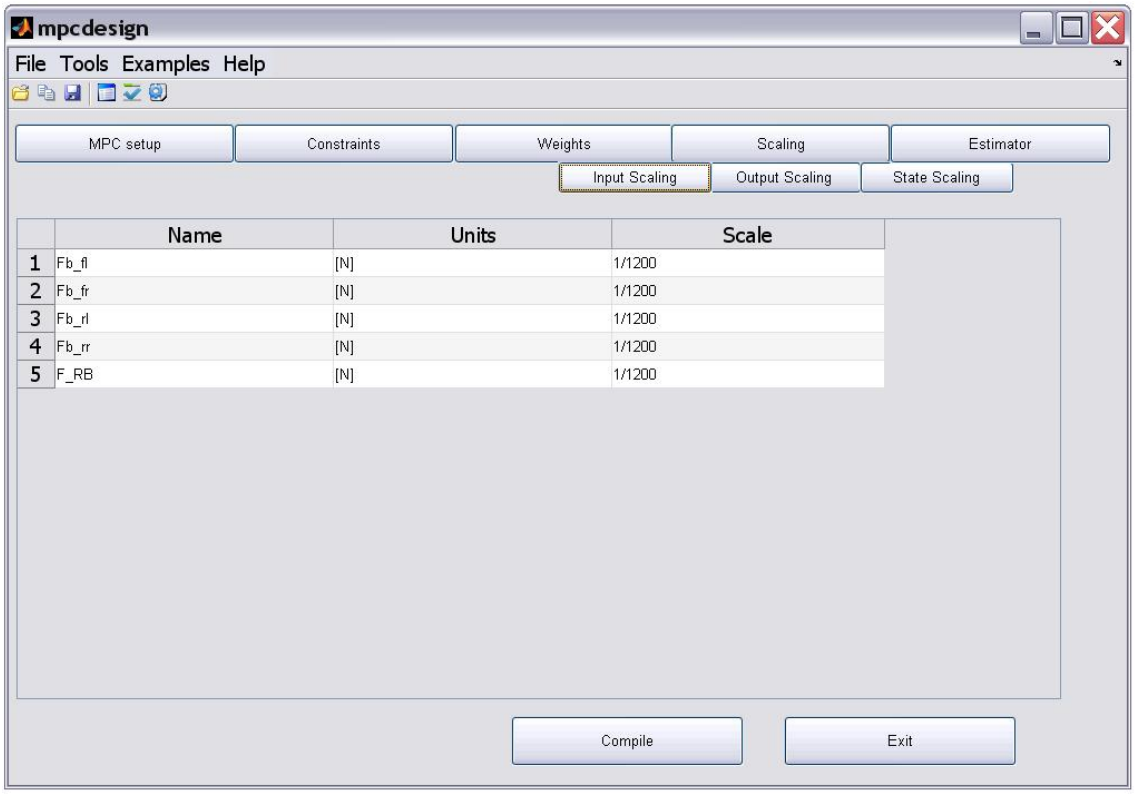

Figure A.9.: Input Scaling

Finally, the estimator can be determined inside design tool

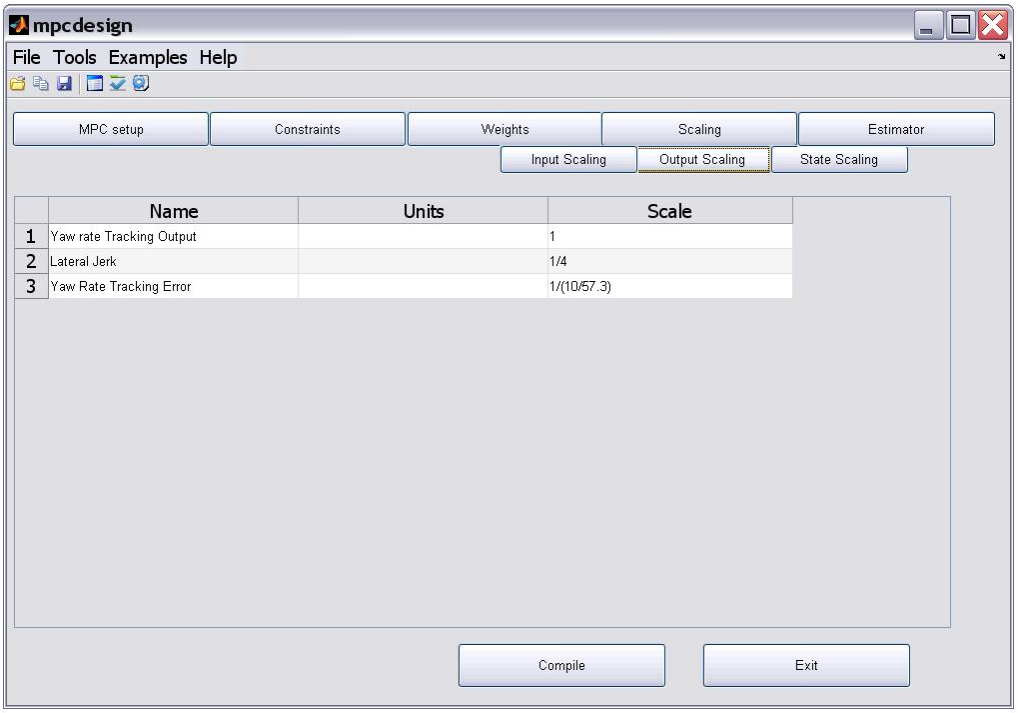

Figure A.10.: Output Scaling

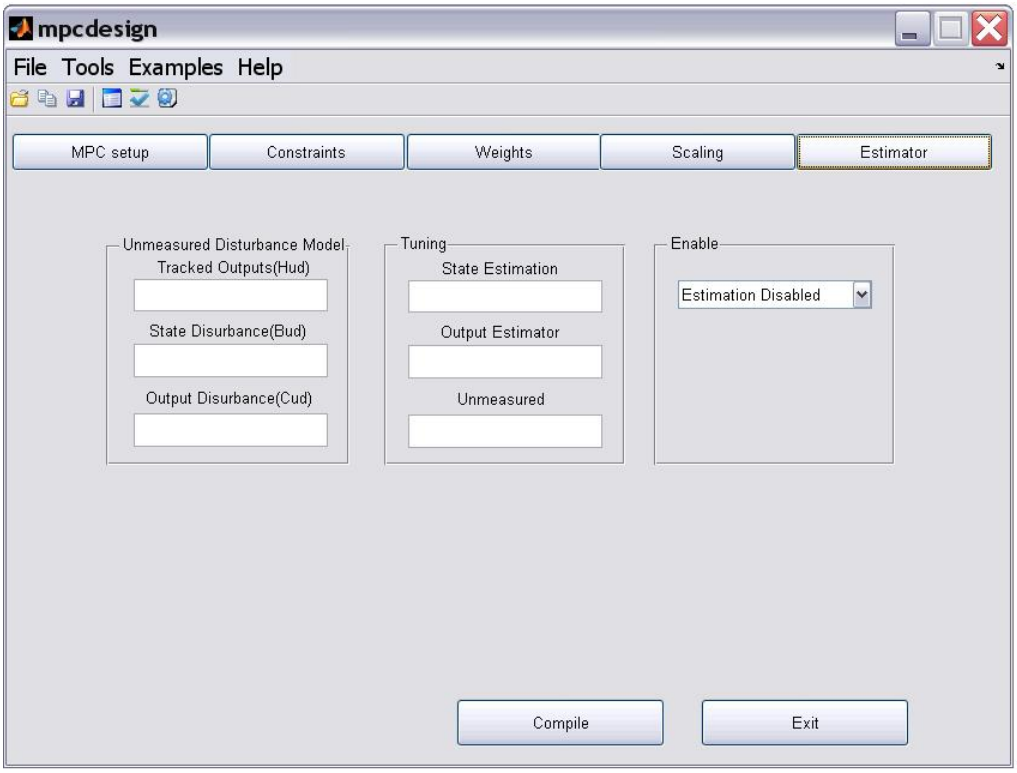

Figure A.11.: Estimator setup

A.7. Simulink Block

The MPC can be simulated using MPC simulink block. The simulink blocks can be found by entering
>> MPC_model

Note that the compilation should be done to be able to use the MPC simulink block.

The reference generator creates the reference corresponding to the reference model. When no reference model is specified, it is assumed the reference is not changing over prediction horizon.

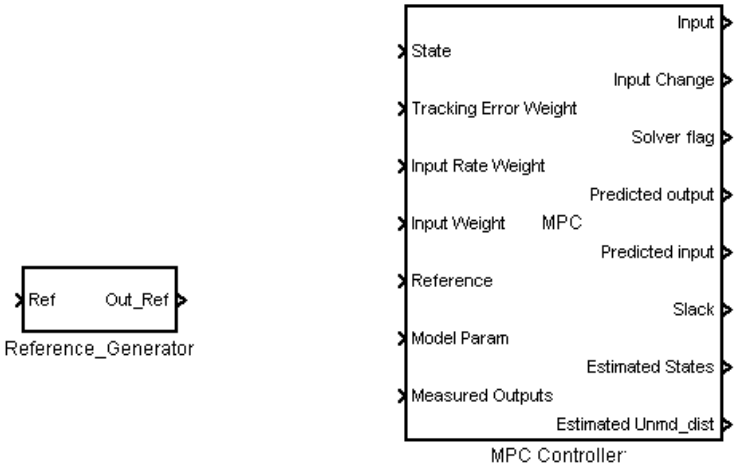

Figure A.12.: Simulink Block

You should specify the MPC Simulink Block's input parameters. In main block, you should give the MPC structure variable's name. MPC structure should be available in the workspace. Moreover, you should specify which input ports for MPC Simulink block is used. Default parameters are assumed for the unused ports.

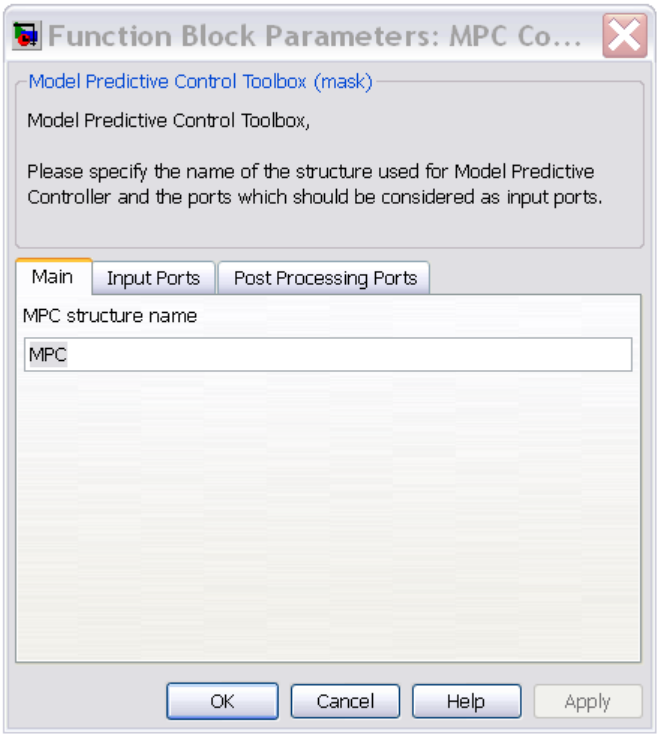

Figure A.13.: Simulink MPC main, you should give the MPC structure variable name

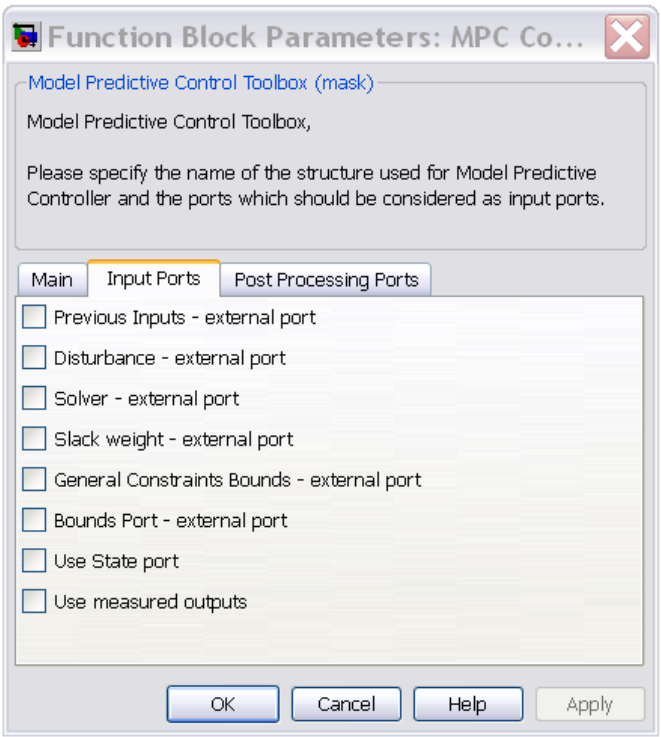

Figure A.14.: Simulink MPC input port, you should define which inputs are given to the MPC simulink's block

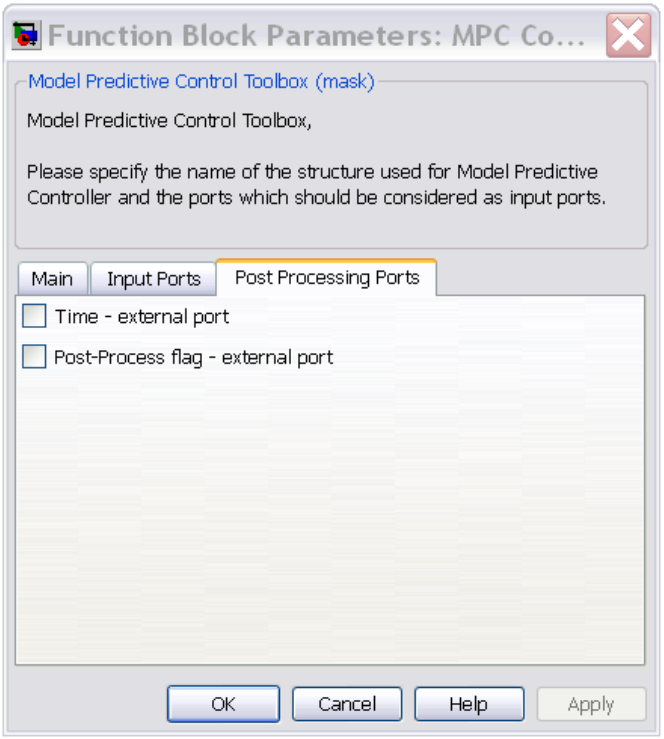

Figure A.15.: Simulink MPC Post Processing ports, you should define which inputs are given to the MPC simulink's block

A.8. MPC commands

MPC Toolbox can be used inside your own MATLAB functions. In this section, we describes the commands to embed MPC Toolbox in your functions.

A.8.1. General Formulation

To setup MPC you should specify a model for the plant, define constraints and weights on tracking error, control input and command changes. Reference model can also be included which enables MPC to anticipate the reference changes (look-ahead feature).

Horizons

You can define the following variables to specify the MPC horizons.
MPC.Hp, Hp should contain the prediction horizon, where the outputs are estimated for Hp steps ahead, and the optimality and feasibility of the problem will be ensured for the Hp steps ahead.
MPC.Hu, Hu should contain the control horizon. The optimal control inputs are obtained by considering the optimality for Hu steps command changes. It is assumed the prediction is bigger than control horizon, where the control inputs remain unchanged after Hu steps.
MPC.Hc, Hc should contain the constraint horizon. Where the constrained outputs and soft constrained inputs are ensured to remain feasible in Hc steps.

Defining Constraints

For handling the constrained MPC, you need to specify the bound on input and output variables. The constraints can be of type hard and soft. The soft constraint is assumed to have a tolerance on feasibility violations, in contrast hard constrained variables can not get violated. The violation of the soft constraints will be penalized to ensure least violation of constraints.
MPC.constraints.u_ubnd, u_ubnd should contain the upper bounds on the control inputs.
MPC.constraints.u_lbnd, u_lbnd should contain the lower bounds on the control inputs.
MPC.constraints.u_sc_upper, u_sc_upper should contain the upper bounds on the control inputs.
MPC.constraints.u_sc_lbnd, u_sc_lbnd should contain the lower bounds on the control inputs.
MPC.constraints.du_ubnd, du_ubnd should contain the upper bounds on the control input changes.
MPC.constraints.du_lbnd, du_lbnd should contain the lower bounds on the control input changes.
MPC.constraints.y_c_ubnd, y_c_ubnd should contain the upper bounds on the hard constrained outputs.
MPC.constraints.y_c_lbnd, y_c_lbnd should contain the lower bounds on the hard constrained outputs.
MPC.constraints.y_sc_ubnd, y_sc_ubnd should contain the upper bounds on the soft constrained outputs.

MPC.constraints.y_sc_lbnd, y_sc_lower should contain the lower bounds on the soft constrained outputs.
MPC.constraints.e_sc_ubnd, e_sc_ubnd should contain the upper bounds on the tracking errors.
MPC.constraints.e_sc_lbnd, e_sc_lbnd should contain the lower bounds on the tracking errors.

Defining Weights

In the current version of the toolbox, enabling the user to have time varying weighting. Weighting penalties can be matrices of the appropriate dimension.

Q, Weight on the tracking errors, where the dimension should agree with the number of tracking outputs. R, Weight on the input rates, where the dimension should agree with the number of control inputs. S, Weight on the inputs, where the dimension should agree with the number of control inputs.
rho, Weight on the slack variable which is the maxiumum amount where the constraints are violated.

Additional parameters

You can define the following parameters to specify the MPC controller solver.
MPC.solver, By determining this parameter you can choose from the solvers $LSSOL$ ([11]),$QPDANTZ$, $NPSOL$ ([10]), $qpOASES$ ([6]) .

MPC.buffer_variables, This parameter forces the MPC to compute all the matrices once, which improves the controllers speed for time invariant plants.

Saving the structure

$>> MPC = savempcparams(MPC, model\_file, ref\_file)$;
The command *savempcparams* generates the model and generates the MPC structure for the MPC Simulink Block. The inputs of the subroutine are MPC structure including the MPC options, model and reference files.

Compilation

$>> compilempc(compile\_type)$;
The command *compilempc* generates the MATLAB executable files to be used inside the Simulink.
compile_type could take the following types
all (Default) Where the MPC and reference, MATLAB executable files are generated.
all+ (Default) Where the MPC,reference and plant, MATLAB executable files are generated. The plant will start initially with the initial state specified.

MPC (Default) Where the MPC, MATLAB executable files are generated.
ref (Default) Where the reference, MATLAB executable files are generated.

For the double integrator example using the following code one can run the toolbox
$model\_file = 'funcs2Ic.m'$;
$ref\_file = ''$;
$MPC.u\_ubnd = [5]$;
$MPC.u\_lbnd = [-5]$;
$MPC.du\_ubnd = [1]$;
$MPC.du\_lbnd = [-1]$;
$MPC.du\_ubnd = [1]$;
$MPC.Hu = 40$;
$MPC.Hp = 50$;
$MPC = savempcparams(MPC, model\_file, ref\_file)$;
$compilempc('all+')$;

A.9. Example

We define our state space formulation by

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, B_d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\ C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, D_d = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{A.12}$$

We introduce how this simple state space model can be embedded in Model Predictive Control Toolbox. We do it in the following steps. First, we define the Model Predictive Control formulation using design tool as Linear Time Invariant formulation, and determine the state space model, We then define tuning parameters. We define any positive definite matrix for tracking error weight, input weight and output weight. Finally, we can use MPC Simulink block to simulate the system(Figure (A.16)) , which yields the optimal output trajectory shown in Figure(A.17)

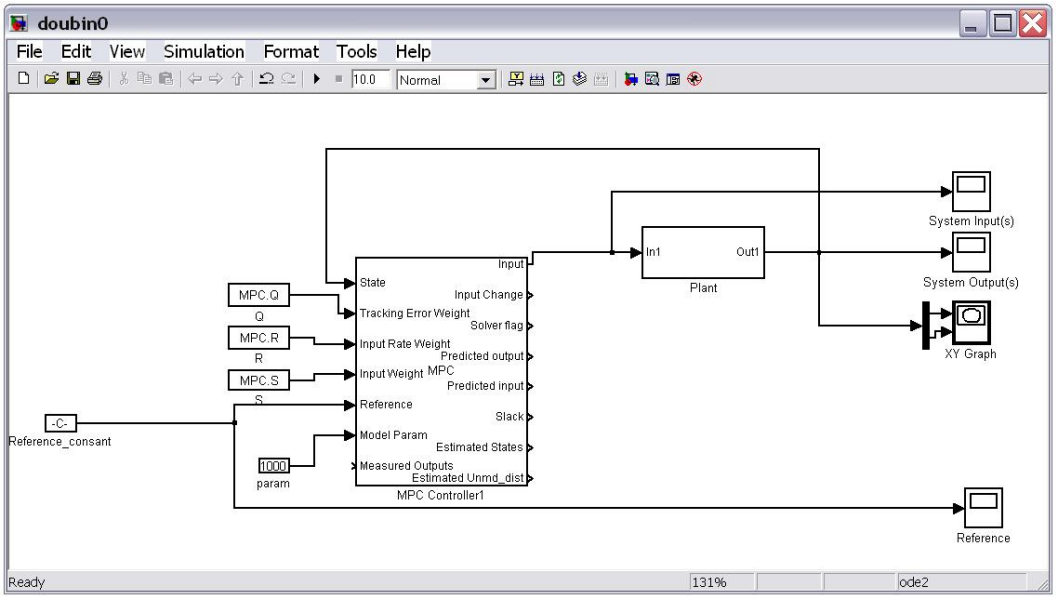

Figure A.16.: The MPC Toolbox can be embedded in Simulink software

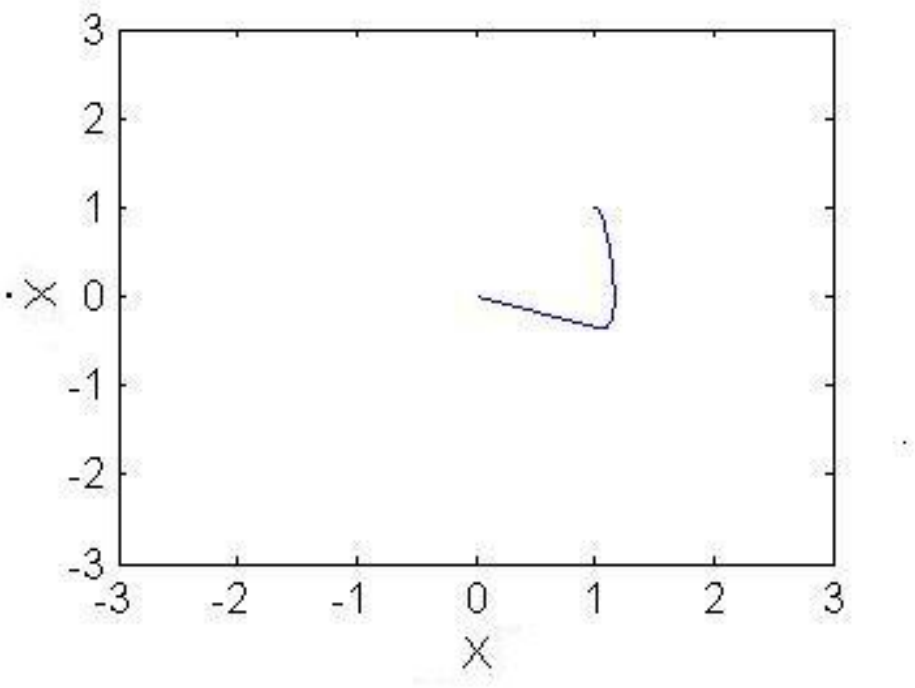

Figure A.17.: The resulting optimal trajectory