Multiscale Modeling and Model Predictive Control of Processing Systems

by

Orhan I. Karslıgil

B.S., Boğaziçi University (1995)

Submitted to the Department of Chemical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2000

Signature of Author .....
Department of Chemical Engineering
23 May 2000

Certified by .....
George Stephanopoulos
A. D. Little Professor of Chemical Engineering
Thesis Supervisor

Accepted by .....
Robert E. Cohen
St. Laurent Professor of Chemical Engineering
Chairman, Committee for Graduate Students

Multiscale Modeling and Model Predictive Control of Processing Systems

by

Orhan I. Karslıgil

Submitted to the Department of Chemical Engineering
on 23 May 2000, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Traditional approaches in Model Predictive Control (MPC) suffer from several weaknesses such as: (a) Satisfying the output constraints over a finite control horizon and guaranteeing closed loop stability. (b) The need for infinite horizon for robust stability and performance. (c) Limited representation of plant-model mismatch. (d) Inability to shape frequency response characteristics of the outputs through systematically selected weights in the objective function. (e) Significant numerical complexity which depends on the number of input constraints. These weaknesses are addressed by formulating and solving the MPC in a multiscale (time/scale) domain. Based on the *wavelet transformation* of time domain models, the multiscale models are defined on dyadic or higher-order trees, whose nodes are used to index the values of any variable, localized in both time and scale (range of frequencies). The objective function, state equations, output equations and constraints on inputs and outputs are transformed into the multiscale domain. Moreover, feedback information is also generated for all scales using a multiscale constrained state estimator, providing rich depiction of the plant-model mismatch (including modeling errors, external disturbances and measurement noise) than the pure time domain formulation. This problem formulation: (i) incorporates rich depiction of feedback errors and provides an environment to identify plant-model mismatch at multiple scales, (ii) it provides a natural framework for optimum fusion of multirate measurements and control actions. The solution to this problem, (i) satisfies all the constraints on inputs and outputs if the problem is feasible at least over an infinite horizon and (ii) satisfies the frequency response specifications on the controlled outputs. In addition it reduces the computational load through two very effective mechanisms. (1) Sets horizon to the required length at each open loop optimization step. (2) It minimizes the search space for active constraints, because once a constraint is determined to be active at a scale, all the lower scale depictions of the constraint emanating from that node, will also be active, and can be solved using parallelized algorithms thus reducing the complexity further and allowing handling of larger problems with more constraints.

Thesis Supervisor: George Stephanopoulos
Title: A. D. Little Professor of Chemical Engineering

Acknowledgments

Like in the Oscars...

I want to thank the people who made this thesis possible...

Thank you George, many thanks for your patience. He tried hard to change my excessively critical approach to new ideas. He was a great teacher to me, and I am now a proud member of his academic tree...

Hey Matthew, you are the man....

Shahin, Kiko, Jerry, Christine, Alex, Enrique, Ajay, Andreas: Over the years I had the chance to get to know the brightest people around. LISPE was intellectually a very stimulating environment. I was enriched by the discussions on hunting, artificial intelligence, lateral logic questions, history, geography (Brazil, Greece, Turkey, Iraq, Japan, South Africa, Spain, Canada, Korea, India, Egypt, Argentina, Austria)...

I would also like to thank to my parents for their support and love. They still think that I know chemistry, don't tell them that I am a process engineer...

Finally, Gül, thank you, for being the person you are, for your support, help, criticism and for not giving up correcting my errors. I love you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

so easy it seemed
Once found, which yet unfound most
would have thought
Impossible
John Milton

Fast developments in industry, computer technology and mathematics give rise to a need to explore more sophisticated control, identification and estimation methodologies in order to improve process performance in economic, safety, and environmental impact terms.

One of the most important shortcomings of today's methods is handling of multiscale processes, i.e. processes which occur at different scales, at different frequency bands.

In chemical engineering applications multiscale phenomena can appear in many different occasions:

· Sensors sending data at different sampling rate (measurements are done at different scales)

· Actuators operating at different physical rates (control actions are exerted at different scales)

· Different process equipment with different characteristic times

· Chemical reactions with different reaction rates (stiff ODE systems)

· Physical phenomena like mass and heat transport having different characteristic times

· Phenomena like crystallization, turbulence, vortex formation , etc., have multiscale char-

acter.

Modeling and identification of those processes in a consistent way across scales and their control requires a Multiscale Analysis and a Multiscale System Theory Framework . Previous work on Multiscale System Theory by Willsky et al. ([5], [1], [2], [3]) already indicate the advantages of working in multiscale domain such as multiscale data fusion, computational efficiency, statistical optimality, explicit calculation of error statistics, estimates at multiple resolutions and so forth.

The developments during the last 10 years in the wavelet theory give us the opportunity to use wavelets and their multiscale character to establish a Multiscale System Theory Framework. Benveniste at al.([5], [1], [2]) studied multiscale state space models defined on binary tree structures, which are natural extensions of the wavelet theory.

Much of the work on multiscale modeling and identification is done in signal processing field on image analysis and there is not much active research on multiscale control [6].

This knowledge should be extended to create multiscale models describing chemical processes, to identify those models in time-frequency (multiscale) domain, and to control systems described by these multiscale models.

Our work will describe the basic framework of multiscale modeling and we will explore the advantages of using Model Predictive Control (MPC) as the advanced control algorithm for multiscale systems. We will look at first what MPC is and how it works and comment on its shortcomings related with today's' identification and estimation algorithms. We will propose solutions to these problems using the multiscale system theory.

1.1 Control of Large Scale Systems: MPC

Model Predictive Control refers to a class of control algorithms that compute a manipulated variable profile by utilizing a process model to optimize a linear or quadratic open loop performance objective subject to constraints over a future time horizon. The first move of this open loop optimal manipulated variable profile is then implemented. This procedure is repeated at each control interval with the process measurements used to update the optimization problem [8]

Over the years different MPC algorithms are developed. Some of them are listed below:

IDCOM (Identification and Command), DMC (Dynamic Matrix Control), QDMC (Quadratic Programming DMC), IDCOM-M/S (MIMO and SISO), HIECON (Hierarchical Constraint control), PCT (Predictive Control Technology).

In electrical engineering they are called as Receding Horizon Control, Predictive Control, Generalized Predictive Control. Similar methods gave birth to Model Algorithmic Control, Internal Model Predictive Control, and others.

MPC has many advantages for applications in chemical process control:

- Flexibility in modeling: State space models, convolution models, input/output models, linear transfer function models can be used.
- Multivariable systems
- Computational advantages.
- Ability to handle of constraints.
- Ability to handle model uncertainty.
- Incorporation of process nonlinearities.
- Improvement in stability, robustness and performance

For processes with constraints on process input and/or outputs, constrained MPC is currently the most effective methodology for treating constraints in a systematic way.

1.2 Weaknesses of MPC

Although industrial applications prove the advantages of MPC over traditional PID-like feedback controllers, theoretical investigation of issues related to nominal stability and performance, and robust stability and performance still challenge researchers in this field, especially in cases with constraints on inputs and/or states [7].

Robustness of MPC affects its performance and stability through plant-model mismatch and uncertainty. Performance degradation resulting from plant-model mismatch should be reduced and it should be ensured that state constraints are met in case of mismatch/uncertainty [4].

Traditional approaches suffer from several weaknesses, such as the following:

- Satisfying the output constraints over a finite control horizon while guaranteeing closed loop stability.
- The need for infinite (or very large) control horizon to ensure robust stability and performance leads to computationally unrealizable (or very time consuming) algorithms.
- Fairly limited and contrived assumptions for the representation of plant-model mismatch,
- Inability to deliberately shape frequency response characteristics through systematic assignment of weights in the objective function.
- The high he numerical complexity which increases rapidly as to the number of input constraints increases.

1.3 Shortcomings and Multiscale Domain

Control relevant modeling enhances the performance of the controller and the controlled system significantly. Existing methodologies in controller design are mainly based on the frequency domain and cannot handle models involving constraints on inputs and outputs. MPC, which is capable of handling such constraints, cannot be formulated in frequency domain and thus cannot utilize the benefits and flexibility of frequency domain controller design techniques, which include loop shaping, noise reduction, and robustness enhancement etc. In addition to those issues the major difficulty with today's modeling and MPC type control strategies is the computational load. Since all of the frequency bands (scales) should be solved simultaneously at every open loop optimization step the size of the optimization problem becomes very large and difficult to solve in the allowed time period. All these weaknesses can be addressed if the MPC is formulated and solved in a multiscale (time/scale) domain. Solution of the optimization problem at selected frequency bands would result in a variable horizon control algorithm, that is computationally much more efficient compared to the classical MPC. Formulating the problem in multiscale (time/scale) domain has the additional benefit of allowing the use of frequency domain type controller design techniques, since the new domain gives access to both scale and time domains simultaneously.

1.4 Multiscale Systems Theory

Multiscale (time/scale or time frequency) formulation of the MPC problem is achieved through the *Multiscale Systems Theory Framework*, which is based on the *wavelet transformation* of time domain models. These models are defined on dyadic or higher-order trees, whose nodes are used to index the values of any variable, localized in both time and scale (range of frequencies).

1.5 Addressing the Issues through Multiscale Systems Theory.

The objective function, state equations, output equations and constraints on inputs and outputs are transformed into the multiscale domain. Moreover, feedback information is also generated for all scales using a multiscale constrained state estimator, providing a richer depiction of the plant-model mismatch (including modeling errors, external disturbances and measurement noise) than the pure time domain formulation.

An initial step with $\log_2(N)$ complexity (N being the length of the control horizon) determines the upper bound on the required length of the control horizon N, thus allowing a variable length horizon algorithm. The constrained quadratic programming problem is solved in the multiscale domain over this horizon, where weighing scales in the objective function differently provides a natural environment to specify frequency response characteristics of the controller explicitly. Large weights at higher scales (low frequencies, longer time scales) cause high frequency controller action and thus fast response, whereas large weights at lower scales (high frequencies, shorter time scales) cause slower but smoother controller action without high frequency components.

This problem formulation offers the following advantages:

- Incorporates rich depiction of feedback errors and provides an environment to identify plant-model mismatch at multiple scales (section 5.4).
- Provides a natural framework for optimum fusion of multirate measurements and control actions.
- Allows for systematic determination of the weights in the objective function, thus achieving the desired behavior of the controlled outputs.

The solution to this problem (section 4.5):

- satisfies all the constraints on inputs and outputs if the problem is feasible at least over an infinite horizon,
- satisfies the frequency response specifications at the same time (section 4.4.3),
- minimizes the computational load by setting the horizon to the required length at each open loop optimization step (section 4.4.1)
- minimizes the search space for active constraints because once a constraint is determined to be active at a scale all the lower scales emanating from that node will also be active, and
- can be obtained using parallel algorithms reducing the complexity further thus allowing handling of larger problems with more constraints (section 3.6.3).

1.6 Thesis Content

This thesis will delve into these issues and elaborate on their solution. Chapter 2 will introduce the Model Predictive Control algorithm and its properties. In Chapter 3 we will develop the multiscale systems theory and show how time domain models can be transformed into the multiscale domain. In Chapter 4 we will formulate the MPC in time/scale domain and we will also discuss the algorithm to solve it. Chapter 5 will illustrate through various case studies how multiscale MPC provides solution to the classical MPC's shortcomings. Finally in Chapter 6 we will present our conclusions and make recommendations for future research steps.

Bibliography

[1] M. Basseville, A. Benveniste, K. Chou, S. Golden, R. Nikoukhah, and A. Willsky. Modeling and estimation of multiresolution stochastic processes. *IEEE Transactions in Information Theory*, 38, 1992.

[2] M. Basseville, A. Benveniste, and A. Willsky. Multiscale-autoregressive processes, part i: Lattice structures for whitening and modeling. *IEEE Transactions on Signal Processing*, 40, 1992.

[3] M. Basseville, A. Benveniste, and A. Willsky. Multiscale-autoregressive processes, part i: Schur-levinson parametrizations. *IEEE Transactions on Signal Processing*, 40, 1992.

[4] M. Bell, D. Limebeer, and R. Sargent. Robust receding horizon optimal control. CPSE Consortium Meeting, 1996.

[5] A. Benveniste, R. Nikoukhah, and A. Willsky. Multiscale system theory. *IEEE Transactions on Circuits and Systems-1: Fundamental Theory and Applications*, 41, 1994.

[6] W. Feng: *The Application of Time-Frequency Techniques to Process Control and Identification.* PhD thesis, Texas AM University, 1996.

[7] J. Lee. Recent advances in model predictive control and other related areas. *CPC-V Conference Proceedings*, 1995.

[8] K. Muske and J. Rawlings. Model predictive control with linear models. *AICHE Journal*, 39(2):262–287, 1993.

Chapter 2

Model Predictive Control

Model Predictive Control (MPC) belongs to the family of optimal controllers. To be able to understand its strength and weaknesses the theory of optimal control should be visited first. A short definition of optimal control will be followed by the first theoretically sound implementation of optimal control: Linear Quadratic Regulator. Constraints will be introduced next and a numerical algorithm to solve such problems will be demonstrated. Finally Model Predictive Control and its properties will be described in great detail.

2.1 Optimal Control

The optimal control problem can be defined simply as: For the continuous-time system, with the obvious extension to the discrete-time system, one wants to determine the control function $u(t)$ from among the admissible set which takes the system from the initial state $x(0)$ at time $t = 0$ to a final condition or goal such that some suitable performance index is minimized.

Assuming that $u^0(k)$ (in discrete-time domain) is the solution of the optimal control problem, the control law, in which the control is some function of the initial state and time, is:

$$u^0(k) = g[x(0), k] \tag{2.1}$$

When such a functionality is obtained, the system is said to be operating in an *open-loop* manner: that is $u^0(k)$ is not an explicit function of $x(k)$. In the same sense, a *closed-loop*

control law could have the form:

$$u^0(k) = g[x(k), k] \tag{2.2}$$

In which the current value of the state variable is included in the functionality. If g is linear in $x(k)$ the control law will be called a *linear control law*, and if g does not depend on k (or time) explicitly the control law is called a *stationary control law.*

One important member of the optimal control family is the Linear Quadratic Regulator, which falls into the category of *open-loop* optimal controllers. The next section will deal with the history, formulation and solution of the LQR.

2.2 LQR Control

Qin and Badgwell [19] describe the development of modern control concepts by indicating that it can be traced to the work of Kalman in the early 1960's, who sought to determine when a linear control system can be said to be optimal ([9]; [10]). Kalman studied a Linear Quadratic Regulator (LQR) designed to minimize a quadratic objective function. The process to be controlled can be described by a discrete-time, deterministic linear state-space model:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ y_k &= Cx_k \end{aligned} \tag{2.3}$$

The vector u represents process inputs, or manipulated variables; vector y describes process output measurements. The vector x represents process states. The state vector is defined such that, knowing its value at time k and future inputs allows one to predict how the plant will evolve for all future time. Much of the power of Kalman's work relies on the fact that this general process model was used. The objective function to be minimized penalizes squared input and state deviations from the origin and includes separate state and input weight matrices

Q and R to allow for tuning trade-offs:

$$J = \sum_{j=1}^{\infty} \left(\|x_{k+j}\|_Q^2 + \|u_{k+j}\|_R^2 \right) \tag{2.4}$$

where the norm terms in the objective function are defined as follows:

$$\|x\|_Q^2 = x^T Q x \tag{2.5}$$

Implicit in this formulation is the assumption that all variables are written in terms of deviations from a desired steady-state. The solution to the LQR problem was shown to be a proportional controller, with a gain matrix K computed from the solution of a matrix Ricatti equation:

$$u_k = -Kx_k \tag{2.6}$$

A more detailed description of the LQR problem including modeling and measurement errors is given in Appendix A.

The infinite prediction horizon of the LQR algorithm endows the algorithm with powerful stabilizing properties; it was shown to be stabilizing for any reasonable linear plant (stabilizable and detectable), as long as the objective function weight matrices Q and R are positive definite. A dual theory was developed to estimate plant states from noisy input and output measurements, using what is now known as a Kalman Filter. The combined LQR controller and Kalman Filter is called a Linear Quadratic Gaussian (LQG) controller. Constraints on the process inputs, states and outputs were not considered in the development of LQG theory. Although LQG theory provides an elegant and powerful solution to the problem of controlling an unconstrained linear plant, it had little impact on control technology development in the process industries. The most significant of the reasons cited for this failure include ([22], [6]) :

- constraints
- process nonlinearities
- model uncertainty (robustness)
- unique performance criteria

- cultural reasons (people,education, etc.)

It is well known that the economic operating point of a typical process unit often lies at the intersection of constraints [18]. A successful industrial controller must therefore maintain the system as close as possible to constraints without violating them. In addition, process units are typically complex, nonlinear, constrained multivariable systems whose dynamic behavior changes with time due to such effects as changes in operating conditions and catalyst aging. Process units are also quite individual so that development of process models from fundamental physics and chemistry is difficult to justify economically. Indeed the application areas where LQG theory had a more immediate impact, such as the aerospace industry, are characterized by physical systems for which it is technically and economically feasible to develop accurate fundamental models. Process units may also have unique performance criteria that are difficult to express in the LQG framework, requiring time dependent output weights or additional logic to delineate different operating modes. However the most significant reasons that LQG theory failed to have a strong impact may have been related to the culture of the industrial process control community at the time, in which instrument technicians and control engineers either had no exposure to LQG concepts or regarded them as impractical.

In the next section we will introduce techniques to solve the above formulation under input and/or output constraints.

2.2.1 Optimization With Inequality Constraints

To be able to handle the constraints in LQR, one has to introduce a different solution algorithm, which is numerical rather than analytical, as in LQR. The lack of an analytical solution brings additional burden on proving properties like stability or robustness. Although there are quite many work-arounds, which we will mention later, there are still no clear answers how to determine those systems theoretic properties.

2.2.2 The Problem Definition

In order to introduce the constraints one should change the formulation and optimality conditions of the optimization problem slightly as follows:

$$\min J(u) \tag{2.7}$$

$$\text{subject to} \quad f(u) \leq 0 \tag{2.8}$$

The optimal solution can be either on an active constraint or it can be an interior point so that the constraints are all satisfied and passive. Considering small perturbations around the optimum

$$dJ = \left(\frac{\partial J}{\partial u}\right)_{u^0} du \geq 0 \tag{2.9}$$

$$df = \left(\frac{\partial f}{\partial u}\right)_{u^0} du \leq 0 \tag{2.10}$$

above inequalities should be satisfied. The same requirement can be expressed as follows, too:

$$\frac{\partial J}{\partial u} + \lambda^T \frac{\partial f}{\partial u} = 0, \quad \lambda \geq 0 \tag{2.11}$$

Furthermore the original optimization function can be modified to incorporate the constraints

$$H(u, \lambda) = J(u) + \lambda^T f(u) \tag{2.12}$$

Then the necessary condition becomes

$$\frac{\partial H}{\partial u} = 0 \tag{2.13}$$

and

$$f(u) \leq 0, \tag{2.14}$$

where

$$\lambda \begin{cases} \geq 0, & f(u) = 0 \\ = 0, & f(u) < 0 \end{cases} \tag{2.15}$$

Remark 1 *The gradient of J with respect to u at a minimum must be pointed in such a way that decrease of J can only come by violating the constraints.*

2.2.3 The Numerical Method

There are many different approaches to solve the above formulation. The numerical method consists of two major steps:

1. Finding a Feasible Point

Find a y such that

$$f(u) \leq 0 \tag{2.16}$$

Guess a value for y, and the consider the perturbation

$$df = \frac{\partial f}{\partial u} du \tag{2.17}$$

If certain components of f are greater than zero then the matrix F formed by the infeasible rows of $\frac{\partial f}{\partial u}$ should satisfy

$$F du \leq 0 \tag{2.18}$$

and du should improve $f(u + du)$ towards a feasible solution.

2. Finding a Feasible Improvement

Using the feasible y we next search for a better objective function value such that

$$J(u+du) < J(u) \tag{2.19}$$

$$f(u+du) \leq 0 \tag{2.20}$$

and this can be represented as follows

$$\begin{bmatrix} \frac{\partial J}{\partial u} \\ \frac{\partial J}{\partial u} \end{bmatrix} du \equiv H du \leq 0 \tag{2.21}$$

2.2.4 Inequality Constraints on the Control Variables

Suppose that there are constraints of the form

$$C(u,t) \leq 0 \tag{2.22}$$

and if the Hamiltonian is defined as

$$H = \lambda^T f + J + \mu^T C \tag{2.23}$$

then the necessary condition becomes

$$H_u = J_u + \lambda^T f_u + \mu^T C_u = 0 \tag{2.24}$$

with the additional requirements (2.15) and

$$\mu \begin{cases} \geq 0, & C = 0 \\ = 0, & C < 0 \end{cases} \tag{2.25}$$

Detailed information on these subjects can be found in [3, Applied Optimal Control].

2.2.5 Example: Bang Bang Control

The following is a nice treatment of the constrained optimal control (from [4, Intro to Optim. Cont])

Assume parking of a car by manipulating the acceleration

$$\frac{d^2x}{dt^2} = u(t) \tag{2.26}$$

and

$$-\alpha \leq u \leq \beta \tag{2.27}$$

Minimize the time T

$$T = \int_0^T 1dt \tag{2.28}$$

subject to (2.26) and (2.27) and

$$x(0) = 0 \ , \ \dot{x}(0) = 0 \ , \ x(T) = a \ , \ \dot{x}(T) = 0 \tag{2.29}$$

The inequality constraints can be transformed into the constraint equations

$$\nu^2 = (u + \alpha)(\beta - u) \tag{2.30}$$

Since ν is real (2.27) should be satisfied automatically. Convert the problem to state space model by

$$x_1 = x \tag{2.31}$$

$$\dot{x}_1 = x_2 \tag{2.32}$$

$$\dot{x}_2 = u \tag{2.33}$$

$$x_1(0) = x_2(0) = 0 \tag{2.34}$$

$$x_1(T) = a\ ,\ x_2(T) = 0 \tag{2.35}$$

The augmented functional becomes

$$T^* = \int_0^T \left\{1 + p_1\left(x_2 - \dot{x}_1\right) + p_2\left(u - \dot{x}_2\right) + \mu\left[\nu^2 - (u+\alpha)(\beta - u)\right]\right\} dt \tag{2.36}$$

From this the optimality criterion is (Euler Equations for optimal solution)

$$\frac{\partial F}{\partial x_1} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{x}_1}\right) = 0, \quad \text{that is } \dot{p}_1 = 0 \tag{2.37}$$

$$\frac{\partial F}{\partial x_2} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{x}_2}\right) = 0, \quad \text{that is } \dot{p}_2 = -p_1 \tag{2.38}$$

$$\frac{\partial F}{\partial u} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{u}}\right) = 0, \quad \text{that is } p_2 = \mu(\beta - \alpha - 2u) \tag{2.39}$$

$$\frac{\partial F}{\partial \nu} - \frac{d}{dt}\left(\frac{\partial F}{\partial \dot{\nu}}\right) = 0, \quad \text{that is } 2\nu\mu = 0 \tag{2.40}$$

From here there are 2 possibilities, either $\mu = 0$ (this does not generate any solution) or $\nu = 0$ (which generates the Bang Bang control)

2.3 Introduction to MPC

Who invented predictive control?

-God

-Predictive control is a discovery not an invention but God needs prophets.

Richalet

2.3.1 History

The Model Predictive Control (MPC) methodology has been developed in industry and thoroughly implemented to the point where it is now considered a basic tool of the trade of the process control engineer [6].

The industry addressed the weaknesses of LQR by introducing a more general model based

control methodology in which the dynamic optimization problem is solved on-line at each control execution. The differences from LQR being the generality of the objective, the process model and the application of the computed process input profile, where only the first element of the profile is fed to the system, the new methodology provided a better and more flexible framework to handle practical issues.

Although the development and application of MPC technology was driven by the industry, the idea of using sequential open-loop dynamic optimization to control a system was not new. Moving or receding horizon controllers were proposed in early 60's, but since the quality of the model based controller depends heavily on the speed of the solution of the open loop dynamic optimization problem at each execution step, the practical implementation did not occur until mid 70's, when digital computers started finding a place in process control industries.

The first description of MPC application was presented by Richalet et al. [22] at a 1976 conference. They called their approach as Model Predictive Heuristic Control (MPHC) and the algorithm to solve it was called IDCOM (Identification and Command). MPHC formulation had hard constraints on inputs and/or outputs, a quadratic performance objective function with a finite prediction horizon, a reference path to follow, and a linear impulse response model for the plant.

Around the same time Shell Engineers developed their own MPC technology and named it as DMC, Dynamic Matrix Control [5]. DMC formulation used linear step response model for the plant, a setpoint to follow rather than a reference path, a quadratic objective with finite prediction horizon, and a least square type solution for the open loop optimization problem. The objective function for DMC consists of two parts, one penalizing the deviations of the states and the other penalizing the deviations of the inputs from their setpoint. This formulation results in a less aggressive input profile and it provides a degree of robustness to modeling error.

Incorporation of constraints into DMC required a new algorithm to solve the optimization problem. Shell engineers decided to use Quadratic Programming (QP), which gave the name QDMC to the new methodology.

Adding constraints to the open loop optimization step begged other issues, though. Feasibility of the QP problem was not guaranteed, and there was no clear way to handle an infeasible solution. Converting hard constraints into soft ones was not completely satisfactory. Differ-

ent solutions were proposed to address this problem, and the major idea was prioritization of the constraints. At the same time translation of control requirements into relative weights for a single objective function became increasingly difficult. Some researchers proposed the use of multiple objective functions. These motivations resulted in new algorithms IDCOM-M (Identification and Command- Multivariable systems) and HIECOM (Hierarchical Constraint Control).

There are many other variations in the market, but the underlying methodology is the same, as will be shown more in detail in the following sections.

2.3.2 Properties and Advantages of the Formulation

As its history shows MPC technology has been developed to address the issues encountered in the industry. As a result its properties reflect the flexibility required for applications in chemical process control.

A quick glance at the properties would generate a list as below. Some of the properties are there by design, and some are natural outcomes.

- Flexibility in modeling: State space models, convolution models (finite step response, finite impulse response models), input/output models, and linear transfer function models can all be used in the formulation of MPC. Each model has its advantages and disadvantages. In the last decade state space models are used extensively, because of their generality and easy representation.

- Multivariable systems: One important reason why MPC was so successful is its ability to handle multivariable systems exactly as it handles single variable systems. There is absolutely no need to make any changes in formulation or theory. Multivariable process control is a natural extension to MPC. State space models representing multivariable processes can be used without any modification.

- Computational advantages: The optimization of the open-loop performance objective is performed by either linear or quadratic programming algorithms. These algorithms are efficient and robust, which is essential for on-line applications. Additionally more than one

performance objective can be constructed in MPC paradigm, which provides flexibility in handling the constraints and increases performance.

- Handling of constraints: For processes with constraints on process input and/or outputs, constrained MPC is currently the most effective methodology for treating constraints in a systematic way. Constraints on inputs, rate of change of inputs, states and outputs can be incorporated into the MPC formulation as part of the linear or quadratic program. This allows operation close to process constraints, which is necessary for economically optimal control of chemical processes.

- Model uncertainty: Model uncertainties and measurement errors can be incorporated into the model or into the on-line identification algorithms. This property provides a better control on the robustness characteristics of the controller.

- Process nonlinearities: The MPC algorithm can handle many different process models. So nonlinearities can easily be added and handled. The formulation remains the same but the optimization problem becomes much more difficult, depending on the type of non-linear models used. A global minimum can not be guaranteed for each open loop step and theoretical treatment of nonlinear MPC is not easy at all, but still the existing formulation can be used without any further changes to incorporate process nonlinearities.

- Stability, robustness and performance: One of the most important successes of MPC is its stabilizing effect. MPC performs well even with non-minimum phase plants where there is no easy way to design a feedback controller ([15], [14]). Although industrial applications prove the advantages of MPC over traditional PID-like feedback controllers, theoretical investigation of issues related to nominal stability and performance, and robust stability and performance still challenge researchers in this field, especially in cases with constraints on inputs and/or states [11]. Robustness of MPC affects its performance and stability through plant-model mismatch and uncertainty. Performance degradation resulting from plant-model mismatch should be reduced and it should be ensured that state constraints are met in case of mismatch/uncertainty [1]. Rawlings and Muske [15] discuss the nominal stability and performance of MPC in three sections:

– Stable plants: If the plant is described through the model,

$$x_{k+1} = Ax_k + Bu_k$$

in which $x_k \in \Re^n, u_k \in \Re^m$ and x_0 is assumed measured and the performance objective is written as follows:

$$\min_{u^N} = \sum_{k=0}^{\infty} x_k^T Q x_k + u_k^T R u_k$$

where Q and R are positive definite, symmetric weighing matrices. Then for stable A and $N \geq 1$, the receding horizon controller with the above objective function is stabilizing.

– Unstable plants: For stabilizable $\{A, B\}$ with r unstable modes and $N \geq r$, the receding horizon controller with the above objective function is stabilizing.

– Constraints: We consider input and state constraint:

$$\begin{aligned} Du_k &\leq d \\ Hx_k &\leq h \end{aligned}$$

in which $d \in \Re^p$ and $h \in \Re^q$ and $d_i, h_i > 0$.

* Stable plants: For stable A and $N \geq 1$, $x_k = 0$ is an asymptotically stable solution of the closed-loop receding horizon controller with the above objective function and feasible constraints for every $x_0 \in \Re^n$.

* Unstable plants: For stabilizable $\{A, B\}$ with r unstable modes and $N \geq r$, $x_k = 0$ is an asymptotically stable solution of the closed-loop receding horizon controller with the above objective function and feasible constraints for every $x_0 \in X_N$ (where X_N demote the set of x_0 for which there exists $\{u_k\}^{N-1} \in U$ and $u_k = 0$, $k \geq N$ such that $\lim_{k\to\infty} x_k = 0$).

For systems with constraints which give rise to non-linear control there are no explicit controller design methodologies, which can guarantee stability and performance robustness.

However, there are methods that account for the asymptotic nominal stability of MPC algorithms with infinite prediction horizon (Rawlings, Muske 1993). The basic parameters of the algorithm affecting the stability, and performance,both nominal and robust, are the length of prediction and control horizons, and the weights of various components in the performance objective. Of course, like in all other control methods a good model and clean measurements from the process improve the performance of the MPC.

Better identification, a general framework to deal with multirate, multivariable, ([11] [17] and [2]) and multiscale processes is required to satisfy the need of industrial applications.

2.3.3 Problem Formulation

Model Predictive Control formulation represents a collection of knowledge from various fields such as optimization, control theory and estimation. This section will describe how those fields are brought together to represent the MPC formulation and how to implement the solution.

The notation for the following representation can be found at the end of this section.

The state and output equations for a system with m inputs, m outputs and n states can be represented in state space as follows:

$$x_{k+1} = A_{n\times n}x_k + B_{n\times m}u_k + d_k \tag{2.41}$$

$$y_k = C_{m\times n}x_k \tag{2.42}$$

The objective function for the state tracking problem over an horizon of length N becomes:

$$\Phi\left(x_0, u\right) = \sum_{N} \left(R_k - x_k\right)^T \left(R_k - x_k\right) \tag{2.43}$$

This objective function can also be written as:

$$\Phi\left(x_0, u\right) = \left(\mathbf{R} - \mathbf{x}\right)^T \left(\mathbf{R} - \mathbf{x}\right) \tag{2.44}$$

where

$$\mathbf{R} = \begin{pmatrix} R_1 \\ \vdots \\ R_N \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}$$

and additionally state equations over the horizon with length N can compactly be described as

$$\begin{aligned} \mathbf{x} = & \begin{pmatrix} A \\ \vdots \\ A^N \end{pmatrix} x_0 + \begin{pmatrix} B & \cdots & 0 \\ \vdots & \vdots & \vdots \\ A^{N-1}B & \cdots & B \end{pmatrix} \begin{pmatrix} u_0 \\ \vdots \\ u_{N-1} \end{pmatrix} + \\ & \begin{pmatrix} I & \cdots & 0 \\ \vdots & \vdots & \vdots \\ A^{N-1} & \cdots & I \end{pmatrix} \begin{pmatrix} d_0 \\ \vdots \\ d_{N-1} \end{pmatrix} \end{aligned} \tag{2.45}$$

The input constraints are:

$$\underline{U} \leq u_k \leq \overline{U}, \ \forall \ k = 0..N-1 \tag{2.46}$$

The output constraints are:

$$\underline{Y} \leq y_k \leq \overline{Y}, \ \forall \ k = 1..N \tag{2.47}$$

Additionally one can have bounds on the rate of change of inputs, end of horizon state or output constraints or penalty terms (soft constraints) in the objective function. Different formulations will be demonstrated in the next section along with their theoretical and practical properties.

The mathematical dual of the prediction problem is the estimation problem. In most applications the states are not directly measured. If the state space model comes from a discrete transfer function, then the states will usually have no physical meaning and not be measurable. Even if the states are physically meaningful, sensors may not be available to measure each state. In these cases, output feedback must be performed using an observer that recon-

structs the states from the output measurements. Since the controller presented above is in the state-space linear quadratic framework, it can take direct advantage of the results from linear quadratic filtering theory.

The standard linear observer is constructed for the system below in which w_k and v_k are zero-mean, uncorelated, normally distributed, stochastic variables.

$$x_{k+1} = A_{n\times n}x_k + B_{n\times m}u_k + G_{n\times m}w_k \tag{2.48}$$

$$y_k = C_{m\times n}x_k + v_k \tag{2.49}$$

The optimal linear observer for this system in which $\hat{x}_{k+1|k}$ is the estimate of the state vector at time $k+1$ given output measurements up to time k is :

$$\hat{x}_{k+1|k} = A\hat{x}_{k|k-1} + Bu_k + L\left(y_k - C\hat{x}_{k|k-1}\right) \tag{2.50}$$

The discrete Kalman filter gain, L, minimizes the mean square error of the state estimate $\hat{x}_{k+1|k}$. It is computed from the solution of the following discrete filtering steady-state Riccati equation with Q_w and R_v the covariance matrices of w_k and v_k respectively.

$$P = A\left[P - PC^T\left(CPC^T + R_v\right)^{-1}CP\right]A^T + GQ_wG^T \tag{2.51}$$

$$L = APC^T\left(CPC^T + R_v\right)^{-1} \tag{2.52}$$

This observer optimally reconstructs the states from the output measurements given the noise assumptions above. The steady-state Riccati formulation guarantees nominal stability of the filter in (2.50) provided $R_v > 0$, $[C, A]$ detectable, and $[A, GQ^{\frac{1}{2}}]$ stabilizable.

With this observer the term d_k in equation (2.41) can be expressed as:

$$d_k = L\left(y_k - C\hat{x}_{k|k-1}\right) \tag{2.53}$$

and this value can be assumed to be constant over the horizon of length N when solving the minimization problem. d_k represents the combination of measurement and modeling errors at

a given time, where the measurement noise (and modeling noise) are assumed to have a known statistical structure (i.e. known covariances)

One problem with the observer described above is that it can generate physically unrealistic state estimations. To prevent this one can modify the problem and formulate the estimation problem with constraints as follows:

$$\min_{\hat{w}} \Phi_k = \hat{w}_{k-N-1}^T P^{-1} \hat{w}_{k-N-1} + \sum_{j=k-N}^{k-1} \left(\hat{w}_j^T Q_w^{-1} \hat{w}_j\right) + \sum_{j=k-N}^{k} \left(\hat{v}_j^T R_v^{-1} \hat{v}_j\right) \tag{2.54}$$

subject to:

$$\begin{aligned} \hat{x}_{j+1|k} &= A\hat{x}_{j|k} + Bu_j + \hat{w}_j \\ y_i &= C\hat{x}_{j|k} + \hat{v}_j \end{aligned}$$

with initial estimate

$$\hat{x}_{k-N|k} = \hat{x}_{k-N|k-N-1} + \hat{w}_{k-N-1} \tag{2.55}$$

and the constraints

$$\begin{aligned} \hat{w}_{\min} &\leq \hat{w}_j \leq \hat{w}_{\max} \\ \hat{x}_{\min} &\leq \hat{x}_{j|k} \leq \hat{x}_{\max} \end{aligned}$$

One can also add output constraints to this formulation but inputs and outputs are already fixed and bounds on noise terms can guarantee the feasibility of the solution in terms of outputs.

There is yet another approach to solve the estimation problem which is similar to the MPC algorithm and actually can be called the dual of the MPC problem (as Kalman filter can be called the dual of LQR) This method also allows system parameter identification at the same

time:

$$x_{k+1} = \begin{bmatrix} A_k \\ B_k \end{bmatrix}^T \begin{bmatrix} x_k \\ u_k \end{bmatrix} + w_k^x \tag{2.56}$$

$$\begin{bmatrix} A_{k+1} \\ B_{k+1} \end{bmatrix} = \begin{bmatrix} A_k \\ B_k \end{bmatrix} + w_k^p \tag{2.57}$$

$$y_k = Cx_k + v_k \tag{2.58}$$

$$z_k = \left[x_k \vdots A_k \vdots B_k\right]^T \tag{2.59}$$

$$w_k = \begin{bmatrix} w_k^x \\ w_k^p \end{bmatrix} \tag{2.60}$$

where w_k^x, w_k^p, v_k are zero mean white noise. The minimization problem can be defined then as

$$\min_{z_{k-m+1}, \mathbf{v}, \mathbf{w}} \left(z_{k-m+1} - z_{k-m+1|k-m}\right)^T P_{k-m+1|k-m}^{-1} \left(z_{k-m+1} - z_{k-m+1|k-m}\right) + \sum_{i=k-m+1}^{k} v_i^T R^{-1} v_i + \sum_{i=k-m+1}^{k} w_i^T Q^{-1} w_i \tag{2.61}$$

subject to the constraints

$$v_{\min} \leq v_i \leq v_{\max} \; \forall i \in [k-m+1, k] \tag{2.62}$$

$$w_{\min} \leq w_i \leq w_{\max} \; \forall i \in [k-m+1, k-1] \tag{2.63}$$

$$x_{\min} \leq x_i \leq x_{\max} \; \forall i \in [k-m+1, k] \tag{2.64}$$

$$p_{\min} \leq p_i \leq p_{\max} \; \forall i \in [k-m+1, k] \tag{2.65}$$

2.3.4 Notation:

x: States

u: Inputs

y: Outputs

R: Reference path

d: Disturbance

v: Measurement noise

w: Modeling noise

$\underline{U}$: Lower input constraint

$\overline{U}$: Upper input constraint

$\underline{Y}$: Lower output constraint

$\overline{Y}$: Upper output constraint

2.4 Issues with Classic MPC Formulation

MPC has been first introduced by industry and thus some theoretical properties were developed later by academicians. Theorems involving the closed loop stability are happen to be the most difficult ones to prove. There are also some other practical issues for which there are workarounds but no definitive solutions, yet. Next sections will go more into detail in those subjects.

2.5 Theoretical

Unlike regular feedback systems where nominal and robust stability and performance properties can be derived from analytical expressions, constraints in MPC turn the problem into a non-linear one and thus there exists no closed form solution. The lack of analytical expressions results in indirect methods outside linear control theory to prove nominal, robust stability and performance of MPC.

2.5.1 Nominal Stability of MPC

Stability in the sense of MPC requires little bit more explanation than its definition in linear control theory. First of all since we are dealing with, at least theoretically, infinite horizon controllers the type of stability we are looking for is asymptotic stability, where simply the gap between the target and the controlled variable becomes infinitely small as time progresses.

A more deliberate definition will require first the definition of stability in general:

The issues of stability can be grouped in two main classes:

1. stability with respect to initial conditions

2. Input output stability

These two classes are complementary to each other and can also be combined. For linear systems, the two classes are, in general, equivalent. However, they are differentfor nonlinear dynamic systems [16].

1. **Stability with respect to initial conditions.**

For the following definitions consider a discrete dynamic system of the form

$$x_{k+1} = f(k, x_k), \quad k \geq 0 \tag{2.66}$$

and assume that $x = 0$ is the equilibrium point of this dynamic system.

Definition 1 *Stability: The equilibrium point* 0 *at time* k_0 *is said to be stable at time* k_0*, if for any* $\varepsilon > 0$*, there exists a* $\delta(k_0, \varepsilon) > 0$ *such that*

$$||x_{k_0}|| < \delta(k_0, \varepsilon) \Longrightarrow ||x_k|| < \varepsilon \ \forall \ k \geq k_0 \tag{2.67}$$

Definition 2 *Uniform Stability: The equilibrium point* 0 *at time* k_0 *is said to be uniformly stable over* $[k_0, \infty)$*, if for any* $\varepsilon > 0$*, there exists a* $\delta(\varepsilon) > 0$ *such that*

$$\left.\begin{array}{c} ||x_l|| < \delta(\varepsilon \\ l \geq k_0 \end{array}\right\} \Longrightarrow ||x_k|| < \varepsilon \ \ \forall \ k > l \tag{2.68}$$

Definition 3 *Asymptotic Stability: The equilibrium point* 0 *at time* k_0 *is said to be asymptotically stable at time* k_0*, if*

1. *it is stable at time* k_0 *and*

2. *there exists a* $\delta(k_0) > 0$ *such that*

$$||x_{k_0}|| < \delta(k_0) \Longrightarrow \lim_{k \to \infty} ||x_k|| = 0 \tag{2.69}$$

Definition 4 *Uniform Asymptotic Stability: The equilibrium point* 0 *at time* k_0 *is said to be uniformly asymptotically stable at time* k_0*, if*

$$\lim_{k\to\infty} ||x_k|| = 0 \text{ for any } x_{k_0} \tag{2.70}$$

2. **Input Output Stability**

I/O stability refers to the effect of inputs to systems outputs, i.e. how much amplification the system exerts on the inputs.

Definition 5 *Bounded input - bounded output stability: A system* S*, mapping an input signal* u *to an output signal* x *with* $S(0) = 0$ *is stable if bounded inputs produce bounded outputs, i.e.,*

$$||u||_p < \infty \Longrightarrow ||x||_q < \infty \tag{2.71}$$

where

$$||x||_p = \left(\sum_{k=1}^{\infty} ||x_k||^p\right)^{1/p} \tag{2.72}$$

Although this definition seems to be logical in some circumstances where infinite norm is bounded but finite norms are not, questions arise whether the system is stable or not. A complimentary definition for the amplification of the system fills this hole.

Definition 6 *Finite Gain Stability: A system* $S : l_p^m \to l_q^n$, $u \to x \triangleq Su$ *is finite gain stable if the gain (induced norm) of* S *is finite i.e.,*

$$||S||_{i,pq} \triangleq \sup_{u\in l_p^m \to \{0\}} \frac{||x||_q}{||u||_p} < \infty \tag{2.73}$$

This definition can be further improved if the search of the supremum is performed over the set of inputs u for initial conditions S_0 in the set S. This covers everything required but the computations involved are not trivial.

From these definitions we will definitely use the asymptotic stability and finite gain stability, though in a different way.

Stability is very important and it is fundamental property of a dynamic system. In the MPC formulation there are two types of stability:

1. the stability of the open loop

2. the stability of the closed loop

The first type is relatively easy to show. If the problem is feasible i.e. there exists a feasible set of inputs to drive the system from given initial conditions to the desired set points then ultimately the optimization algorithm will result in an asymptotically stable solution.

The easiest case is the infinite horizon MPC with an objective function which minimizes the deviations from the desired target point. If this problem is feasible then the objective function should have a finite value indicating that the magnitude of the deviations decreases towards to zero as time progresses indicating asymptotically stability.

$$\Phi = \sum_{i=0}^{\infty} (y_i^s - y_i)^T (y_i^s - y_i) \tag{2.74}$$

$$\Phi < \infty$$

$$\Phi_j = (y_i^s - y_i)^T (y_i^s - y_i) = ||(y_i^s - y_i)|| \quad \text{and} \quad \Phi_j \geq 0$$

$$\Longrightarrow \lim_{j \to \infty} \Phi_j = \lim_{j \to \infty} ||(y_i^s - y_i)|| = 0 \tag{2.75}$$

Infeasibilities can give rise to instabilities, or a fully constrained solution can leave an offset which again is not asymptotically infeasible and results in an infinitely large objective function value.

Using an infinite horizon simplifies the theoretical approach but at the same time makes the problem computationally not realizable.

Research on finite horizon MPC generated a wealth in new techniques to tackle some of the issues.

One major technique is to use a finite control horizon and an infinite prediction horizon.

For the sake of simplicity assume that the objective function is the deviation from a reference

path:

$$\begin{aligned} \Phi &= \sum_{i=1}^{\infty} (R_i - x_i)^T (R_i - x_i) \\ x_{i+1} &= Ax_i + Bu_i + d_i \\ |u_i| &\leq \alpha \; \forall \; 0 \leq i \leq p \\ u_j &= u_p \; \forall \; j > p \end{aligned} \tag{2.76}$$

This formulation is computationally realizable since the degrees of freedom is finite. The same objective function can be written as

$$\Phi = \Phi_1 + \Phi_2 \tag{2.77}$$

$$\Phi = \sum_{i=1}^{p+1} (R_i - x_i)^T (R_i - x_i) + \sum_{i=p+2}^{\infty} (R_i - x_i)^T (R_i - x_i) \tag{2.78}$$

it is easy to see that

$$x_{p+1} = A^p x_0 + \sum_{i=0}^{p} A^{p-i-1} B u_i \tag{2.79}$$

and starting from x_{p+2} all the points depend on the first $p+1$ inputs.

$$x_{p+1+k} = A^{p+k} x_0 + \sum_{i=0}^{p} A^{p+k-i-1} B u_i + \left[\sum_{l=0}^{k-1} A^{k-l-1} \right] B u_p \tag{2.80}$$

For asymptotic stability Φ_2 should have a finite value and thus

$$||R_j - x_j|| \rightarrow 0 \text{ as } p+1 < j \rightarrow \infty \tag{2.81}$$

and notice that

$$\begin{aligned} ||R_j - x_j|| &= \left\| R_j - A^k x_p - \left[\sum_{l=0}^{k-1} A^l \right] B u_p \right\| \\ k &= j - p - 1 \end{aligned} \tag{2.82}$$

Assuming that the reference path satisfies

$$R_j = x_{final} + \delta_j \tag{2.83}$$

$$\delta_j \rightarrow 0 \text{ as } j \rightarrow \infty \tag{2.84}$$

where x_{final} should satisfy the criteria

$$\begin{aligned} x_{final} &= Ax_{final} + Bu_{final} \\ x_{final} &= (I - A)^{-1} Bu_{final} \end{aligned} \tag{2.85}$$

and we should also notice that the last input variable u_p should be equal to u_{final} in order to reach the steady state target value ultimately. If we again concentrate on the norm of the deviation from the reference path after the control horizon, $\left\| R_j - A^k x_p - \left[\sum_{i=0}^{k-1} A^i\right] Bu_p \right\|$, we want to show that this norm decays to zero if the dynamic equation satisfies certain criteria. Showing this property will also prove the asymptotic stability.

$$\left[\sum_{i=0}^{k-1} A^i\right] = (I - A)^{-1} \left(I - A^k\right) \tag{2.86}$$

Using the above identity and the following inequality we can establish the proof as follows:

$$\begin{aligned} 0 \leq & \left\| R_j - A^k x_p - \left[\sum_{i=0}^{k-1} A^i\right] Bu_p \right\| \\ = & \left\| R_j - A^k x_p - (I - A)^{-1} \left(I - A^k\right) Bu_p \right\| \\ \leq & \left\| R_j - (I - A)^{-1} Bu_p \right\| + \\ & \left\| -A^k x_p + (I - A)^{-1} A^k Bu_p \right\| \end{aligned}$$

If we substitute equation (2.83) in this inequality, we get

$$\begin{aligned} 0 &\leq \left|\left| R_j - A^k x_p - \left[\sum_{i=0}^{k-1} A^i\right] B u_p \right|\right| \\ &\leq |\delta_j| + \left|\left| -A^k x_p + (I-A)^{-1} A^k B u_p \right|\right| \\ &= |\delta_j| + \left|\left| -A^k x_p + A^k x_{final} \right|\right| \\ &= |\delta_j| + \left|\left| A^k \right|\right| |x_{final} - x_p| \end{aligned}$$

If the dynamic system is stable, i.e. all the eigenvalues of A are positive and less than one, then

$$\begin{aligned} |\delta_j| &\rightarrow 0 \\ \left|\left| A^k \right|\right| &\rightarrow 0 \text{ as } k, j \rightarrow \infty \end{aligned}$$

so the norm becomes

$$\begin{aligned} & \left|\left| R_j - A^k x_p - \left[\sum_{i=0}^{k-1} A^i\right] B u_p \right|\right| \\ \leq\ & |\delta_j| + \left|\left| A^k \right|\right| |x_{final} - x_p| \\ \rightarrow\ & 0 \text{ as } k \rightarrow \infty \end{aligned} \tag{2.87}$$

This concludes the proof showing that the norm approaches zero as we progress in time indicating the asymptotic stability.

Another way to work around the infinite horizon is to put a terminal state constraint. Although it has no physical meaning and can create easily open-loop infeasible problems, this technique results in smaller optimization problems with built-in stability guarantees.

$$\begin{aligned} & \min_{u_i} \sum_{i=1}^{k} (R_i - x_i)^T (R_i - x_i) \\ s.t \quad x_{i+1} &= A x_i + B u_i + d_i \\ |u_i| &< \alpha \quad i \leq k \\ x_k &= x_{\text{Terminal}} \end{aligned} \tag{2.88}$$

To ensure that feasibility is not affected the following system should have a solution:

$$\begin{aligned} x_{\text{Terminal}} &= A^k x_0 + \sum_{j=0}^{k-1} A^{k-1-j} B u_j \\ |u_j| &< \alpha \end{aligned}$$

A less aggressive approach would be relaxing the terminal condition and just putting an additional inequality constraint $|x_{\text{Terminal}} - x_k| < \sigma$ and choosing σ such that the problem remains feasible, but still asymptotically stable.

Adding such new constraints increases the number of tuning parameters and thus makes the use of MPC more difficult.

The second type of stability in MPC theory is the stability of the closed-loop. It is essential to prove that a proposed MPC algorithm is closed-loop stable.

The proof is not trivial even for linear and stable plants. The constraints convert the linear problem into a nonlinear one and there exists no close form solution on which stability analysis can be performed. Instead indirect methods are used:

Consider the following state space model

$$x_{k+1} = Ax_k + Bu_k$$

We assume perfect knowledge of the matrices A and B, full state information x_k and absence of disturbances. The objective function used in the MPC formulation is

$$\min_{\mathbf{u}} \Phi_k = \sum_{i=1}^{p} x_{k+i|k}^T Q x_{k+i|k} + \sum_{i=1}^{p} u_{k+i-1|k}^T R u_{k+i-1|k}$$

where Q is positive definite and R is positive semi-definite matrix. The constraints on inputs and outputs are:

$$\begin{aligned} Gu &\leq g \\ Hx &\leq h \end{aligned}$$

These constraints are assumed to define a feasible solution space containing the origin point.

Closed-loop MPC stability can be established using the following Lyapunov argument [21]. Consider a solution

$$Uopt_{k|k}^{k+p-1|k} = \{uopt_{k|k}, \ldots, uopt_{k+p-1|k}\}$$

to the objective function at time k, and assume that the horizon length p is large enough such that $x_p = 0$. Now guess a solution at time point k+1

$$Uopt_{k+1|k+1}^{k+p|k+1} = \{uopt_{k+1|k}, \ldots, uopt_{k+p-1|k}, \mathbf{0}\}$$

This solution is just a shifted version of the previous optimal solution, where the last value is set to be zero. It is a feasible solution, since zero satisfies the input constraints. The objective function value at point k+1 then becomes

$$\begin{aligned}
\Phi_{k+1} &= \sum_{i=1}^{p} x_{k+i+1|k+1}^T Q x_{k+i+1|k+1} + \sum_{i=1}^{p} uopt_{k+i|k}^T R uopt_{k+i|k} \\
&= \sum_{i=1}^{p} x_{k+i|k}^T Q x_{k+i|k} + \sum_{i=1}^{p} uopt_{k+i-1|k}^T R uopt_{k+i-1|k} + \\
&\quad + x_{k|k}^T Q x_{k|k} + u_{k|k}^T R u_{k|k} - x_{k|k}^T Q x_{k|k} + u_{k|k}^T R u_{k|k} \\
&= \Phi_k^{opt} - x_{k|k}^T Q x_{k|k} + u_{k|k}^T R u_{k|k}
\end{aligned}$$

The real optimal objective function value would be less than or equal to this value, i.e.:

$$\begin{aligned}
\Phi_{k+1}^{opt} &\leq \Phi_{k+1} \\
&= \Phi_k^{opt} - x_{k|k}^T Q x_{k|k} + u_{k|k}^T R u_{k|k} \\
&\leq \Phi_k^{opt}
\end{aligned}$$

Therefore the sequence $\Phi_{k=k_0 \to \infty}^{opt}$ is non-increasing, and consequently it will converge to a constant value, i.e.

$$\lim_{k \to \infty} \Phi_k^{opt} = c$$

In addition, as shown below,

$$c = 0$$

$$\begin{aligned} x_{k|k}^T Q x_{k|k} + u_{k|k}^T R u_{k|k} &\leq \Phi_k^{opt} - \Phi_{k+1}^{opt} \\ \lim_{k\to\infty} \left(x_{k|k}^T Q x_{k|k} + u_{k|k}^T R u_{k|k} \right) &\leq \lim_{k\to\infty} \left(\Phi_k^{opt} - \Phi_{k+1}^{opt} \right) \\ &= 0 \end{aligned}$$

The implication of the above is that

$$\lim_{k\to\infty} x_k = 0 \text{ and } \lim_{k\to\infty} u_k = 0$$

and the closed-loop stability is proven.

The proof itself is not difficult, but there are certain assumptions it hinges on:

1. The open loop optimization problem is feasible.
2. In finite time the desired set-points can be reached.
3. There is no model-plant mismatch.
4. There are no disturbances.
5. States are perfectly observable.
6. Constraints on inputs and states or outputs are time independent.
7. The open loop problems solution is globally optimum.

For stable processes selecting a proper control and prediction horizon ensures the assumptions 1 and 2. Again practically infinite horizon for the open loop problem is not the only solution, techniques like terminal point constraints, constraining the final state point to belong to a small neighborhood of the set point or state contraction arguments help to ensure stability.

For unstable processes one needs to show that the stabilization is possible with the current state values. The main idea is doing a transformation and steering the controllable variables to

their set points and ensuring that uncontrollable modes are stabilizable. If it is not the case, then there is no way to bring the states to their set points.

2.5.2 Robust Stability

Whenever there is a discrepancy between the computed or assumed and real values, the robustness issue arise, because the process behaves differently than assumed by the controller. Even very small differences can be amplified to destabilize an otherwise nominally stable controller.

In MPC to guarantee robustness of stability the following inequality constraint should be satisfied

$$0 \leq L\left(x_k, u_k\right) \leq \Phi_k^{opt} - \Phi_{k+1}^{opt} \tag{2.89}$$

where

$$L\left(x_k, u_k\right) = \left(R_k - x_k\right)^T \left(R_k - x_k\right)$$

The same inequality constraint was used to prove closed-loop stability. Even if there is model uncertainty the above constraint would force the controller to remain stable. There two main approaches to satisfy this constraint,

1. Tune the objective function in such a way that the consecutive objective function values at each open-loop step have a decreasing value. This would prevent major modification of the MPC structure.

2. Add the constraint below explicitly to the problem formulation. Although the optimization problem becomes more difficult to solve, robustness is preserved without any tuning, which could make the controller more conservative than required.

$$0 \leq \Phi_{k-1}^{opt} - \Phi_k \tag{2.90}$$

There are many different MPC formulations based on these two main approaches to guarantee the robust stability. Interested readers are suggested to look at Nikolau [16], Badgwell [19],

and others.

Of course like in any other control scheme a better knowledge about the process and the possible uncertainty sources within their corresponding statistics can vastly improve the controller. The main sources of uncertainty are the external disturbances, model-plant mismatch, uncertainty in actuators (calibration errors, equipment degradation or faults, human interception) and measurement errors.

Having perfect knowledge about the range of fluctuations in those sources one can in theory develop a controller which would remain robust under any condition, but such a controller would suffer from the trade-off of being very conservative in return.

In the recent years with the increasing number of control variables that MPC should handle, a different idea started gaining more ground:

Claim 2.1 *If the optimization is very large and complicated to solve, sometimes just a feasible solution is enough.*

This idea supported the development of techniques to simplify the large optimization problems by reducing the number of free variables. This is achieved by blocking the input variables, i.e. forcing a block of input variables to have the same value. A similar technique called condensing can be used to decrease the number of output constraints. More information about these techniques can be found in the section 4.4.5.

One additional benefit of blocking is that it enhances robustness of the MPC. The reduction in the number of control variables put a bound on the rate of change of the output variables. Thus a disturbance cannot move the outputs as much as it could do without blocking. Of course in return performance suffers because the decrease of the number of degrees of freedom also slows down the tracking of desired reference path (or rejecting disturbances).

2.5.3 Performance

The classical measures of system performance such as steady state error, gain margin and phase margin are essential criteria of optimality, and control system compensators are designed to meet these requirements.

In MPC the systems measure of performance, or performance index, is not fixed beforehand. Instead compensation is chosen so that the performance index is maximized or minimized. The value of performance index is unknown until the completion of the optimization process. In unconstrained MPC since the solution can be expressed in a closed form, the performance can be expressed in terms of system parameters and weights in the objective function. So the nominal performance can be tuned as well. Changing the values of the weights one can specify any performance characteristics.

A similar specification cannot be maid for the constrained MPC, though. Rigorous results for the performance of constrained MPC are lacking.

There are several propositions to improve the robust performance of MPC. Those propositions are mainly based on either modifying the structure of the MPC formulation, or tuning specifically for robust performance or improving the optimization algorithms such that more complex and realistic formulations can be solved efficiently on-line.

Since robust performance requires that MPC performs within the desired bounds under any expected uncertainty, all of the propositions above in some way cause a more conservative controller.

There are certain intuitive parameters which effect the performance of MPC:

1. Horizon length: The longer the horizon the more the open-loop optimization step knows about the system and can respond better, but if there are some additional constraints to satisfy, robust stability increasing the horizon length can have the reverse effect and the controller can become more conservative causing inferior performance.

2. Weights in the objective function: Weights in the objective function are far from being arbitrary. They reflect the frequency domain characteristics in a very convoluted manner. A correct choice of weights can generate a response which can reject uncertainties (like noise at certain frequencies) in the system and generate superior performance. Unfortunately classical constrained MPC does not have the advantage of using the frequency domain ideas as the unconstrained MPC can. Frequency shaped cost functionals ([12], [8]) provide a nice framework to shape the loop characteristics.

3. Design of the reference path: The reference path defines the nominal performance. The

best the unconstrained MPC can do is to perfectly track the reference path. Constrained MPC minimizes the deviation from the reference path. The shape of the reference path is closely related to the weights in the objective function and to the desired frequency response characteristics.

2.5.4 Feasibility

Because MPC requires the solution of an optimization problem at each open-loop step, the feasibility of that optimization problem must be ensured. Input, output and additional constraints to guarantee robust stability may generate an empty feasible solution space or they can generate a non-convex solution space (giving rise to multiple local minima and maxima).

One major reason for the industrial success of Model Predictive Control is the ability of the controller to enforce hard constraints on the process. There are two major classes of hard constraints: the input constraints and state or output constraints. The input constraints represent mostly physical limitations and they should be satisfied at all times. On the other hand state or output constraints are only desirables and cannot be strictly enforced at all times, especially if there are external disturbances.

For infinite horizon MPC feasibility and stability become identical making the importance of feasibility clearer.

There are several methods to ensure the feasibility of the MPC and thus the closed-loop stability indirectly.

1. Minimum time approach:

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k \\ J(x_0, \pi) &= \sum_{i=0}^{\infty} \left[x^T Q x + u^T R u\right] \\ \pi &= \{u_0, u_1, ..., u_{N-1}\} \ , \ u_k = 0 \ \ \forall k \geq N \\ Q &\geq 0, R \geq 0, \ \left[Q^{1/2} A\right] \text{ detectable} \end{aligned} \tag{2.91}$$

Rawlings and Muske [15] proposed a method where the time point is determined after which there are no more state or output constraint violations. Up to that point state

or output constraints are violated. At the subsequent open-loop steps the length of this relaxation time decreases at least by one time point and the closed-loop response reflects at most the first relaxation length. With this method the time to earliest possible constraint satisfaction is minimized, it is nominally stabilizing and for longer control horizons open-loop and closed-loop results converge, but the computation of the relaxation time is difficult and during the transient large constraint violations can occur.

2. Soft constraint approach: This method adds a penalty term for constraint violations to the objective function. If the weight of this penalty term becomes large, the soft constraints become practically hard ones. The computation is easy, the solutions are nominally stable, the states are continuous allowing easy establishment of robust stability, but closed-loop and open-loop behaviors are different and tuning is not trivial.

3. Mixed Methods: Rawlings and Scokaert [23] propose new methods for the minimal time and the soft constraint approaches. Their method addresses the shortcomings of the previous methods as described above.

2.6 Practical Properties

Other than the properties normally discussed in the context of feedback controllers and optimal control problems there are certain aspects of MPC which require some additional discussion.

2.6.1 Horizon Length

To ensure a number of desired theoretical properties, the horizon length should be infinitely long. Infinite horizon length MPC has the nice property that feasibility ensures nominal stability. Practically this is not realizable though. The on-line optimization has only a limited time to deliver a solution and this need results in finite horizon MPC. There are several types of finiteness though:

1. Finite control horizon, infinite prediction horizon

2. Finite control horizon, terminal state constraint

3. Finite control horizon and finite prediction horizon

Since degrees of freedom and thus the complexity of the optimization problem depends on the number of input variables, the biggest reduction in computation occurs when the control horizon is constrained to be finite and as short as possible. It is also possible to parametrize the control vector such that although the control horizon is infinite there are only finite number of decision variables. As expected this limitation causes performance degradation and even can destabilize certain systems.

The choice for the length of the control horizon is mostly a design decision and once fixed it remains constant. There were several attempts to formulate a variable horizon MPC [13], where the horizon length is adjusted to guarantee stability.

2.6.2 Tuning

Other than the horizon length there are the weights in the objective function which should be tuned for desired performance and stability. Especially robust performance and stability require very careful tuning.

For unconstrained MPC there are techniques to define the weights such that the resulting frequency response of the controller satisfies certain requirements [12]. Frequency shaped objective functions can only be used if there are no hard bounds. Hard bounds introduce events occurring at certain time points if they are activated and the frequency domain techniques automatically fail because they don't have any time localization. This limitation prohibits the use of powerful frequency domain techniques like loop shaping in constrained MPC.

Tuning in this setting refers to adjusting the matrices Q and R in the following objective function

$$J = \sum_{i=0}^{\infty} \left(x_{i+1}^T Q x_{i+1} + u_i^T R u_i \right) \tag{2.92}$$

There are also cases when the rate of change in inputs is included in the objective function. It

can be established either by converting

$$\begin{aligned} \Delta u^T S \Delta u &\rightarrow u^T P^T S P u \\ \Delta u &= P u \\ \Delta u_i &= u_{i-1} - u_i \end{aligned} \tag{2.93}$$

Addition of that type of penalty terms (soft constraints) allow the designer to shape the frequency response in an indirect and convoluted way. Adding these weights has also another beneficial effect: If the plant has right hand plane zeros the resulting system would have non-minimum phase characteristics which is difficult to handle, but the addition of input penalty terms push the right half plane zeros to the left and thus stabilize the system (and the controller).

2.6.3 Output Constraints

As mentioned before under the feasibility issue the output constraints are the main source of infeasibility. Most of the time these type of constraints represent the economical or environmental limitations. Unlike the input constraints which cannot be violated physically these constraints can be relaxed for certain period of time (see section 2.5.4).

These is also the problem of active output constraints at the steady state operating point. Since active constraints can be regarded as the rule rather than the exception in chemical process operations this is a frequently encountered issue. Rao and Rawlings [20] discuss how this issue can be addressed by computing the best target values by projecting the system onto the active constraints under the finite horizon parametrization of the input.

2.6.4 Disturbance and Model-Plant Mismatch

Classical Model Predictive Control algorithms incorporate disturbances by using the difference between the last predicted and measured outputs and assuming that it will remain constant along the prediction horizon in the open-loop optimization step. This constant term of course includes measurement errors (an observer can reduce it), model-plant mismatch and finally external disturbances. The history of the plant is most of the time neglected and just the last

disturbance term is used. This limited disturbance model prevents incorporation of periodic or long term event (like day-night temperature swings).

Extracting information to minimize the model-plant mismatch is very important. Instead of adding a constant term only, one can also update the parameters of the model. Such an adaptive strategy, though, would be very sensitive to measurement errors and external disturbances. There is research on this topic where additional constraints are incorporated into the MPC algorithm to ensure that the inputs provide the strong persistent excitation criterion [7] Through this method a time varying model can be identified on-line without causing major output fluctuations.

2.7 Summary of Issues

1. Nominal and robust stability with finite horizon MPC.
2. Feasibility of the open-loop problem with output constraints.
3. Limited disturbance models.
4. Determining the horizon length.
5. Specifying frequency domain characteristics.
6. Computational load.
7. Capturing multirate systems.

Bibliography

[1] M. Bell, D. Limebeer, and R. Sargent. Robust receding horizon optimal control. CPSE Consortium Meeting, 1996.

[2] R. G. B. Bequette. Multirate model predictive control, an analysis for single input single output systems. *Proceedings of PSE'91*, II(5), 1991.

[3] A. Bryson. *Applied Optimal Control : Optimization, Estimation, and Control.* Hemisphere Pub. Corp. Washington, 1975.

[4] D. Burghes and A. Graham. *Introduction to Control Theory, Including Optimal Control.* Number QA402.3.B788. John Wiley Sons, 1980.

[5] C. Cutler and B. Ramaker. Dynamic matrix control- a computer control algorithm. *Proceedings of the Joint Automatics Control Conference*, 1980.

[6] C. Garcia, D. Prett, and M. Morari. Model predictive control: Theory and practice, a survey. *Automatica*, 25(3):335–348, 1989.

[7] H. Genceli and M. Nikolaou. New approach to constrained predictive control with simultaneous model identification. *AICHE Journal*, 42(10):2857–2869, 1996.

[8] N. Gupta. Frequency-shaped cost functionals: Extension of linear-quadratic-gaussian design methods. *J. Guidance and Control*, 3(6):529–535, 1980.

[9] R. Kalman. Contributions to the theory of optimal control. *Bull. Soc. Math. Mex.*, 5:102–119, 1960.

[10] R. Kalman. A new approach to linear filtering and prediction problems. *Trans. ASME, J. Basic Engineering*, pages 35–45, 1960.

[11] J. Lee. Recent advances in model predictive control and other related areas. *CPC-V Conference Proceedings*, 1995.

[12] L. Lublin and M. Athans. Linear quadratic regulator control. *The Control Handbook*, 1996.

[13] D. Mayne. Nonlinear model predictive control: An assessment. *Preprints of Chemical Process Control - V*, 1996.

[14] M. Morari. Model predictive control: Multivariable control technique of CHoice in the 1990's? 1994.

[15] K. Muske and J. Rawlings. Model predictive control with linear models. *AICHE Journal*, 39(2):262–287, 1993.

[16] M. Nikolaou. Model predictive controllers: A crytical synthesis of theory and industrial needs. 1998. Preprint.

[17] M. Ohshima and I. Hashimoto. Multi-rate multivariable model predictive control and its application to a semi-commercial polymerization reactor.

[18] D. Prett and R. Gillette. Optimization and constrained multivariable control of a catalytic cracking unit. *Proceedings of the Joint Automatic Control Conference*, 1980.

[19] S. J. Qin and T. Badgwell. An overview of industrial model predictive control technology. *Preprints, University of Texas at Austin*, 1997.

[20] C. Rao and J. Rawlings. Steady states and constraints in model predictive control. *AICHE Journal*, 45(6):1266–1278, 1999.

[21] J. Rawlings, E. Meadows, and K. Muske. Nonlinear model predictive control: A tutorial and survey. *Proceedings of ADCHEM*, pages 203–214, 1994.

[22] J. Richalet, A. Rault, J. Testud, and J. Papon. Model predictrive heuristic control: Applications to industrial processes. *Automatica*, 14:413–428, 1978.

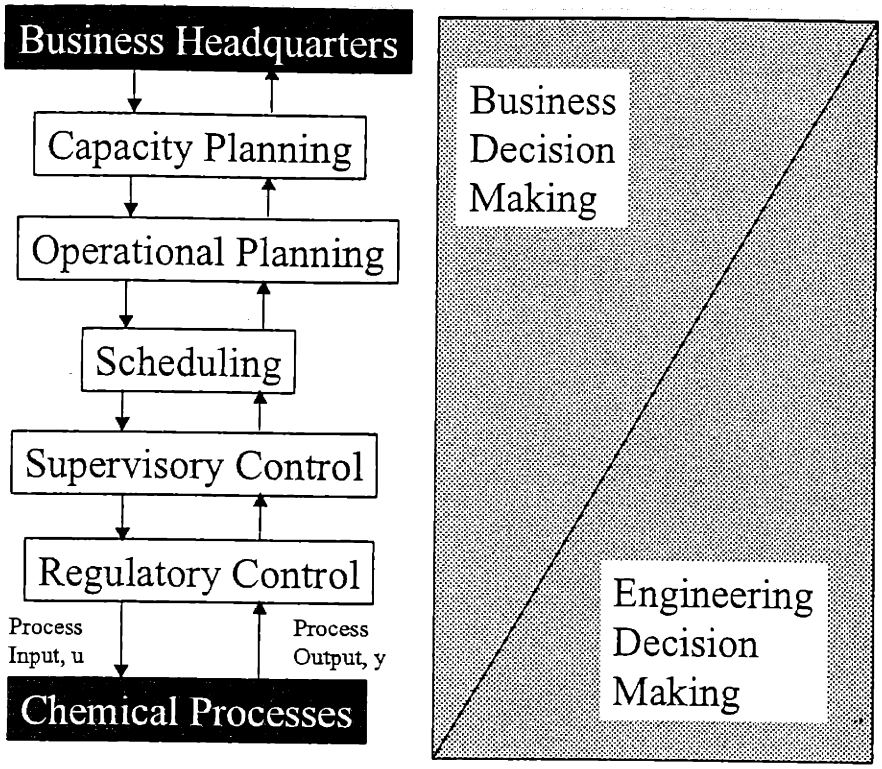
[23] P. Scokaert and J. Rawlings. Feasibility isuues in model predictive control. *AICHE Journal*, Submitted in August 1998.

Chapter 3

Multiscale Systems Theory

It is broadly accepted that physical phenomena occur at different time scales. However, it is not clear how to systematically incorporate this knowledge in the generation of adequate process models, or how to use it for the solution of some basic process engineering problems, e.g. control, estimation, diagnosis. Conventional models, causal and explicit, provide a convoluted representation of physics at various scales and hamper engineering analysis and interpretation. Furthermore, process models used in various engineering tools involve different time scales, e.g. closed-loop control, adaptive control, fault diagnosis, scheduling of control strategies, and planing of operating procedures, involved process models with dynamics of progressively increasing time constants. Current modeling practices are not very instructive on how to create consistent models for a sequence of interrelated engineering tasks, as the above, deployed at different time scales [19]. This fact has also implications on the business decision making process. Capacity planning, operational planning and scheduling are all parts of this process and these steps involve different time scales. In theory each step should be consistent with the rest of the steps, but in practice at every step a different model is used and consistency is rarely achieved. The interaction of these processes is ilustrated in figure (3-1)

In addition, sensors may provide measurements of process behavior at different sampling rates, invoking control actions at correspondingly different rates. The optimal fusion of measurement information or control actions relies on the availability of process models at time scales, which are commensurable with the sampling rate of measurements and the application intervals of control actions.



(Bassett, 1994)

Figure 3-1: Multiple characteristic times related to different engineering and management tasks

The above requirements indicate that we need representations, which capture scale-based characteristics of process models, measurements and control actions. However, the classical Fourier analysis, which has been used to provide such representations, is not adequate. It provides frequency-based information on the behavior of processes and measurements, by integrating the time behavior of such entities over an infinite time horizon, while processes and their characteristics change over finite segments of time. Therefore, we need a framework, which can provide explicit representation of process dynamics, measurements, and control actions, localized in both time and frequency (scale).

Table 3.1: Examples for Multiscale Phenomena

Multiscale temporal	· Multirate measurements · Multirate control action · Different process equipment with different characteristics times · Chemical reactions with different reaction rates (stiff ODE's) · Phenomena like heat, mass transport having different characteristic times
Multiscale spatial	· Crystallization, turbulance, vortex formation · Surface. interface reactions, interactions · Image analysis, patterns (coherent structures) · Compression

3.1 Previous Work

The need for tools to analyze multiresolution (multirate, multiscale) phenomena can be observed in many different fields. The first steps towards a generalized framework to represent multiscale phenomena were taken in the signal processing community.

Extracting multiresolution features from signals and images required mathematical tools similar to Fourier Transformation, but with temporal or spatial localization. Discrete windowed Fourier transformation (DWFT) was introduced long time ago but it was not providing enough flexibility to capture characteristics of a signal at various temporal or spatial scales. There were certain limitations on the size of the window and resulting statistics. So researchers started looking for other tools. Wavelets emerged at this point. Although they were known since early 20th century, their importance became clear after the seminal lectures and papers by Ingrid Daubeshies [10], [11], Stephan Mallat [17],...

Wavelets provide advantages working in multiscale domain such as optimal multiscale data fusion, statistical optimality, explicit calculation of error statistics, estimates at multiple resolutions and do forth [Willsky, Benveniste].

Although the bulk of the work on wavelets has focused on the representation of signals in the time-scale domain [Bakshi [2], [1], Daubeshies [10], Mallat [17]] recent work [Carrier [7]] has extended the original ideas to the formation of models, whose variables are localized in both time and scale. Such models, to be called "multi-scale" models, constitute the basis in the formation of an alternative framework for the solution of process control and estimation problems. This framework has several advantages:

1. Allows a direct correspondence between process behavior in time and scale, and the associated models describing such process.

2. Enables the deployment of naturally parallelizable algorithms for the solution of several basic tasks, e.g. simulation of process dynamics, optimal control, optimal state estimation.

3. Permits the optimal fusion of measurements or/and control actions which occur at different rates.

Next sections will provide an informal introduction to wavelet transformation and its representation on trees.

3.1.1 Wavelet Transformation

The last decade produced a lot of powerful theory and practical applications based on the wavelet transformation. Although the basic knowledge on wavelets existed for a long time the implications of wavelets on signal processing were introduced by Morlet.

The wavelet decomposition of signals

Wavelets are functions that satisfy certain requirements. They integrate to zero, waving above and below the x axis. They are localized and they insure quick and easy calculation of the direct and inverse wavelet transform.

There are many kinds of wavelets: Smooth wavelets, compactly supported wavelets, wavelets with simple mathematical expressions and simple associated filters.

Like sines and cosines in Fourier analysis wavelets are used as basis functions in representing other functions. Unlike the Fourier transformation though wavelets are local in both frequency (scale) (via dilation) and in time (via translation). This localization is the key advantage for control theoretic purposes.

Another important benefit of using wavelet transformation is its computational complexity. Fast Fourier Transformation (FFT) has $O(n \log_2 n)$ complexity whereas Fast Wavelet Transformation has only $O(n)$.

Now consider a continuous signal f_t which is in L^2 space (all square integrable functions), and generate the following sequence of approximations

$$f_{s,t} = \sum_{n=-\infty}^{+\infty} \widetilde{f}_{s,n}\phi_{s,2^s t-n} \quad s = 0, 1, 2, 3... \tag{3.1}$$

$\phi_{s,2^s k}$ is called the *scaling function* and is far from being arbitrary. It should satisfy certain criteria to be qualified as a wavelet scaling function. Those requirements are:

- The next level approximation (coarser level) $\phi_{s,2^s k}$ is a linear combination of the basis functions spanning the space of the $s + 1^{th}$ level of approximation $\phi_{s+1,2^{s+1}k}$, i.e.

$$\phi_{s,2^s l} = \sum_k h_k \phi_{s+1,2^{s+1}l-k} \tag{3.2}$$

- The scaling function should have a local support or one decaying very fast to zero.
- Integer translates of the scaling function should form an orthogonal set.

These requirements determine the filter coefficients h_k.

As we move from coarser approximation $f_{s,t}$ to the finer approximation $f_{s+1,t}$, we add new information. If V_{s+1} represents the space of all functions spanned by the orthogonal set, $\{\phi_{s+1,2^{s+1}k} : k \in Z\}$ and V_s the space of the coarser functions spanned by the orthogonal set, $\{\phi_{s,2^s k} : k \in Z\}$, then $V_s \subset V_{s+1}$. Let

$$V_{s+1} = V_s \oplus W_s \tag{3.3}$$

then W_s, is the space that contains the added information as we go from coarser, $f_{s+1,t}$, to the finer $f_{s,t}$, representation of the original function f_t. It can be shown that the space W_s is spanned by the orthogonal translates of a single function, $\psi_{s,2^s k}$, thus leading to the following equation:

$$f_{s+1,t} = f_{s,t} + \sum_{n=-\infty}^{\infty} \delta\widetilde{f}_{s,n}\psi_{s,2^s t-n} \tag{3.4}$$

The function $\psi_{s,2^s k}$ is called a *wavelet* and is related to the scaling function $\phi_{s+1,2^{s+1}k}$ through the following relationship

$$\psi_{s,2^s l} = \sum_k g_k \phi_{s+1,2^{s+1}l-k} \tag{3.5}$$

where $h(k)$ and $g(k)$ form a conjugate mirror filter pair.

Equation (A.5) constitutes the basis for the multitude of signal analysis techniques that have emerged over the last 8-10 years. It suggests the following:

- A continuous signal can be filtered to a desired approximation, $f_{s+1,t}$, using the scaling function, $\phi_{s+1,2^{s+1}k}$, as a low-pass filter. The dilation parameter, 2^{s+1}, signifies the temporal scale of sampling the continuous function.

- The difference in information content between the two approximations, $f_{s,t}$ and $f_{s+1,t}$, is characterized by the coefficients, $\delta\widetilde{f}_{s,n}$. This implies that the wavelet function can be seen as a band-pass filter that extracts information at a given scale (or equivalently, a given range of frequencies).

- By allowing $s \rightarrow \infty$, equation (A.5) implies that the continuous signal, f_t, can be expressed as,

$$f_t = \sum_{s=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} \delta\widetilde{f}_{s,n} \psi_{s,2^s t-n} \tag{3.6}$$

i.e. a linear combination of dilated and translated versions of the same wavelet function. The coefficient, $\delta\widetilde{f}_{s,n}$, signifies the contribution of the corresponding wavelet, $\psi_{s,2^s t-n}$, to the value of the signal, (a) over a localized segment of time, and (b) containing information in a localized range of frequencies. Equation (3.6) yields a non redundant representation of f_t for any orthogonal set of wavelets.

3.1.2 Tree Notation

Wavelet and scaling functions are defined in a resursive manner. Equation (3.5) clearly reflects the recursive relation between the scaling function and the wavelet function at consecutive scales. The recursive relationship is not just between the basis functions but also between the coefficients. The best way to represent such a relationship is the tree notation as depicted in Figure (3-2).

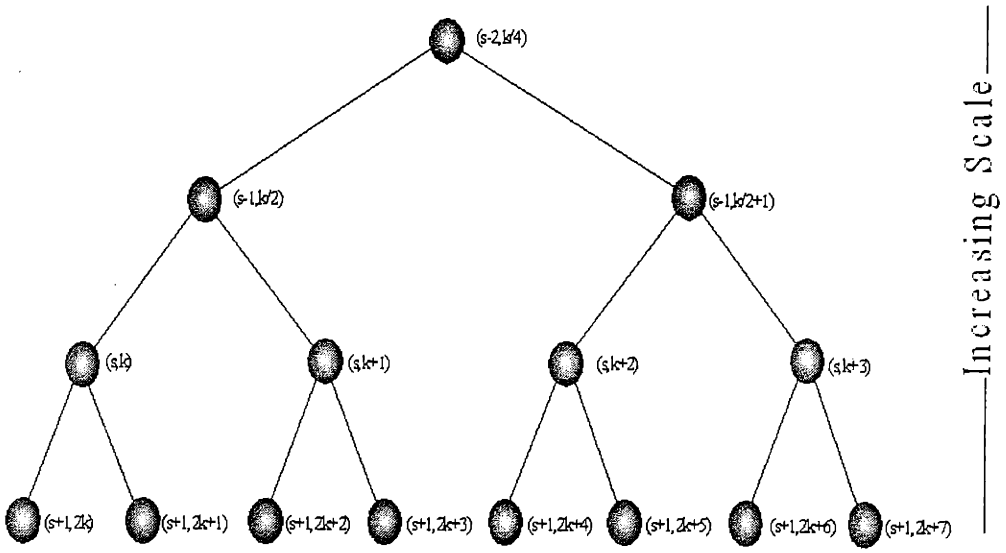


Figure 3-2: Indexing the coefficients of Haar based decomposition on the nodes of binary tree

The discussion in the following sections is applicable for any type of wavelet function, as long as it satisfies the requirements described above. For simplicity and efficiency Haar wavelets will be used in the rest of this thesis if not mentioned otherwise. Figure (3-2) depicts the recursive relationship of Haar wavelet on a binary tree. This relationship can be further described in terms of the signal coefficients.

Assume that there is a time series $f_t = \{f_0, f_1, f_2, ..., f_n, ...\}$, this series can be mapped to a scale with the same sampling interval on the binary tree, i.e.,

$\{f_0 = f_{s,k},\ f_1 = f_{s,k+1},\ f_2 = f_{s,k+2}, ...,\ f_n = f_{s,k+n}, ...\}$. Then the following relationships

hold

$$f_{s-1,\frac{k}{2}} = \frac{1}{\sqrt{2}} f_{s,k} + \frac{1}{\sqrt{2}} f_{s,k+1} \tag{3.7}$$

$$\delta f_{s-1,\frac{k}{2}} = \frac{1}{\sqrt{2}} f_{s,k} - \frac{1}{\sqrt{2}} f_{s,k+1} \tag{3.8}$$

where the coefficients in this averaging scheme are the filter coefficients of the Haar wavelet

$$\phi_t = \begin{cases} \frac{1}{\sqrt{2}} \; 0 \leq t < 1 \\ 0 \text{ otherwise} \end{cases} \quad \text{and} \quad \psi_t = \begin{cases} \frac{1}{\sqrt{2}} \; 0 \leq t < \frac{1}{2} \\ -\frac{1}{\sqrt{2}} \; \frac{1}{2} \leq t < 1 \end{cases} \tag{3.9}$$

for a continuous-time signal and

$$\phi_k = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} \quad \text{and} \quad \psi_k = \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \tag{3.10}$$

for a discrete-time signal. Subsequent filtering of $f_{s-1,\frac{k}{2}}$ leads to coarser depictions of f_t. Both the functions and wavelet coefficients can be mapped to the binary tree for a discrete signal.

Basseville et al. [3], [4] and [5] were the first to realize the significance of homogeneous trees in (a), indexing the wavelet coefficients of a signal, and (b), creating a systems theory for processes defined directly on trees. It is important to realize that the representational framework for signals (and models), offered by homogeneous trees, is far more flexible than figure (3-2) might suggest. For example, it is applicable to higher dimensional trees (for signals defined in multidimensional spaces, e.g. time, spatial coordinates), and to trees with asymmetry, or structures with unusual shapes. Such flexibility can be exploited to match the particular multi-scale structure of a signal's components, or to capture the localized behavior of physico-chemical phenomena with multi-scale features. Given the significant role that homogeneous trees and shifts on them will play, in the development of multiscale process models for estimation and control, in the next section we will try to summarize the main definitions and properties to be used in subsequent sections. The material has been primarily drawn from the works of Basseville et al. [3], [5], Chou [8], Luettgen [16], Irving [14], and, Fieguth [13].

3.1.3 Homogeneous Trees and Shift Operators on Trees an Signal Values

Homogeneous Trees

A homogeneous tree, T, is an infinite, acyclic, undirected, connected graph. If every node of T has exactly $(q+1)$ branches to other nodes, the tree is of order, q. The tree of figure (3-2) with $q = 2$ is called a dyadic tree and will be the primary focus in subsequent sections of this thesis.

A homogeneous tree, T, possesses a natural notion of distance. If s_1 and s_2 are two nodes on a tree, T, the distance between them, $d(s_1, s_2)$, is equal to the number of branches along the shortest path between the nodes s_1 and s_2. For a given tree we need to define a boundary point, before we can specify a partial ordering of the nodes on the tree and thus a convention for indexing the nodes. When $q = 1$, the corresponding tree represents the set of integers and possesses two boundary points, $+\infty$ and $-\infty$. Indeed, any two sequences of integers increasing towards $+\infty$ or decreasing towards $-\infty$ become equivalent, i.e. differ by a finite number of nodes. For a dyadic tree with $q = 2$ the set of boundary points in uncountable. Choose one of them and denote it as $-\infty$.

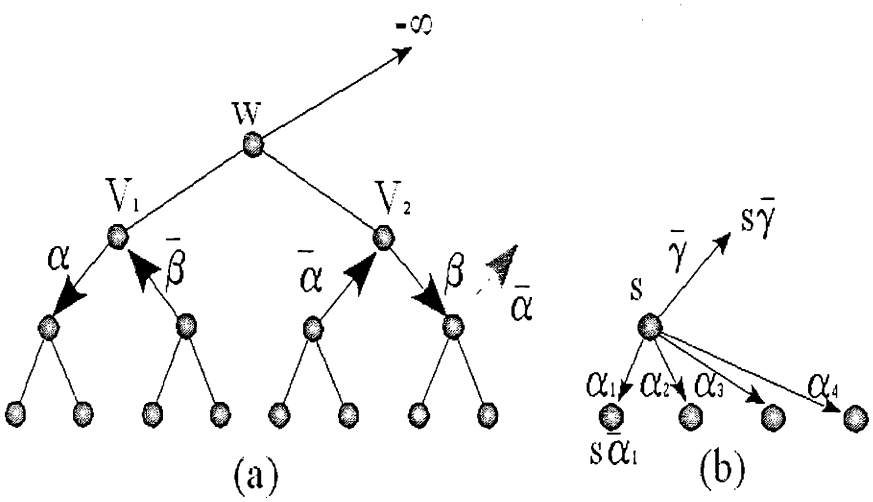


Figure 3-3: (a) Boundary point, horocycles, ans shifts on a dyadic tree. (b) Shift operators on a tree of order 4

Once a boundary point has been specified, draw the tree as shown in figure (3-3a). Nodes

s_1 and s_2 in figure (3-3a) are at the same distance from the boundary point of $-\infty$, since

$$\begin{aligned} d(s_1, -\infty) &= d(s_1, w) + d(w, -\infty) = 2 + d(w, -\infty) \\ d(s_2, -\infty) &= d(s_2, w) + d(w, -\infty) = 2 + d(w, -\infty) \end{aligned}$$

Nodes at the same distance from the boundary point are at the same horocycle, e.g. nodes s_1, s_2, s_3, s_4 in figure (3-3b). In the multi-scale representation of a signal, nodes at the same horocycle correspond to wavelet (or scaling function) coefficients at the same scale. An upward shift along the branches of the tree in figure (3-3a) represents a shift from a wavelet coefficient describing a signal at a finer resolution, to a wavelet coefficient describing the signal at a coarser one.

Having specified the notions of a boundary point and distance between two nodes on a homogeneous tree, it is a rather simple task to devise an indexing system that allows the unique and concise characterization of all nodes. We will discuss a specific indexing system after we have introduced the shift operators on trees.

Shift Operators on Trees

For a tree of order q we need two types of shift operators to traverse the tree as shown in figure (3-3b):

1. upward shift operator ; $\overline{\gamma} : s \rightarrow s\overline{\gamma}$, where node $s\overline{\gamma}$ is the parent node of s,
2. downward shift operator; $\alpha_i : s \rightarrow s\alpha_i$, where node $s\alpha_i$ is the child of node s, $1 \leq i \leq q$.

For the dyadic tree in figure (3a), we observe the following.

1. There exist two downward shift operators, which we will denote as follows:

$$\begin{aligned} \alpha \ : \quad & s \rightarrow s\alpha \equiv \text{left-offspring of } s \\ & \text{e.g. } s_1 = V_1\alpha \text{ and } s_3 = V_2\alpha \\ \beta \ : \quad & s \rightarrow s\beta \equiv \text{right-offspring of } s \\ & \text{e.g. } s_2 = V_1\beta \text{ and } s_4 = V_2\beta \end{aligned}$$

2. There exist two upward shift operators, defined as follows:

$$\overline{\alpha} \ : \quad s \rightarrow s\overline{\alpha} \equiv \text{parent of } s \text{ through a left up-shift}$$
$$\text{e.g. } V_1 = s_1\overline{\alpha} \text{ and } V_2 = s_3\overline{\alpha}$$
$$\overline{\beta} \ : \quad s \rightarrow s\overline{\beta} \equiv \text{parent of } s \text{ through a right up-shift}$$
$$\text{e.g. } V_1 = s_2\overline{\beta} \text{ and } V_2 = s_4\overline{\beta}$$

Note that any node on the tree can only have one parent through either a left- or a right-branch; e.g. there is no node, $s_4\overline{\alpha}$, since s_4 has a parent through a right-branch.

It should be noted that the shift operators, α and β, are the counterpart of the shift operator, z, in discrete Fourier transforms. Both shift the index of a value away from $-\infty$. However, in discrete Fourier transform the corresponding tree is of first-order (q = 1), requiring a single shift, while on a dyadic tree, with two offsprings from every node, we need two distinct shift operators to move away from the boundary point of -∞. Clearly, the upward-shift operators, $\overline{\alpha}$ and $\overline{\beta}$, correspond to z^{-1}.

Shift Operators on Signals

Consider two signals, $x(t)$ and $u(t)$, both of which are decomposed through Haar wavelet, with their wavelet (and scaling function) coefficients indexed by the nodes of the dyadic tree shown in figure (3-3a). Thus, by "signal" we will mean the array of scaling function (or, wavelet) coefficients indexed by the nodes of the dyadic tree. Let us define shift operators on the values of a signal which are consistent with the shift operators on trees, introduced in the previous paragraph. For any node, s, on the dyadic tree, we have the following:

1. If $x(s) = u(s\alpha)$, then let $x(s) = \alpha u(s) \rightarrow \alpha$ is the left branch down-shift operator

2. If $x(s) = u(s\beta)$, then let $x(s) = \beta u(s) \rightarrow \beta$ is the right branch down-shift operator

3. If $x(s) = u(s\overline{\alpha})$ with $u(s\overline{\beta}) = 0$, then let, $x(s) = \overline{\alpha}u(s) \rightarrow \overline{\alpha}$ is the left-branch up-shift operator

4. If $x(s) = u(s\overline{\beta})$ with $u(s\overline{\alpha}) = 0$, then let, $x(s) = \overline{\beta}u(s) \rightarrow \overline{\beta}$ is the right-branch up-shift operator.

It is rather straightforward to show that these shift operators on the signal values indexed by a dyadic tree satisfy the following equations:

$$\begin{aligned} \alpha\overline{\alpha} &= \beta\overline{\beta} = 1 \\ \beta\overline{\alpha} &= \alpha\overline{\beta} = 0 \\ \overline{\alpha}\alpha + \overline{\beta}\beta &= 1 \end{aligned} \tag{3.11}$$

Finally, it should be noted that α and β are one-to-one but not onto operators. In the floowing sections we will see how these shift operators on the values of signals provide the essential elements for the definition of multi-scale transfer functions of dynamic systems on trees.

Example 1 *A multiscale signal decomposition example*

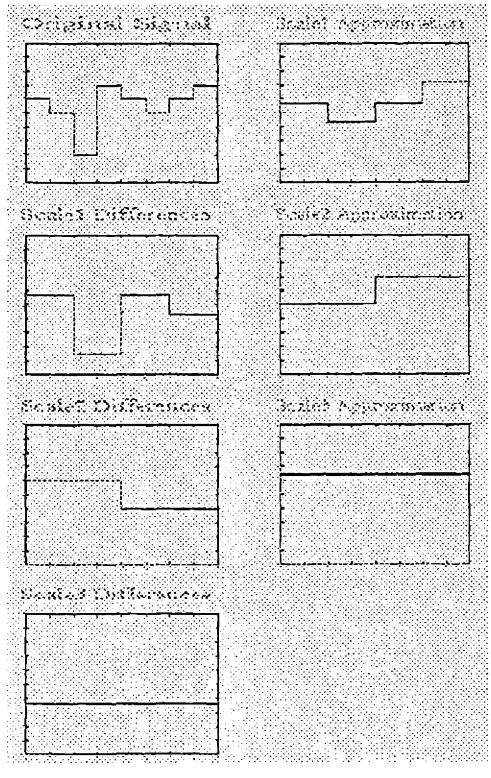


Figure 3-4: Multiscale decomposition of a signal

Approximations correspond to $f_{s,k}$ and differences correspond to $\delta f_{s,k}$. The original signal is

$$f = \{1, 0, -3, 2, 1, 0, 1, 2\}$$

The recursive transformation can be also represented as a matrix operation. The non-redundant information in this transformation is the differences at different scales (wavelet coefficients) and a reference point like the coarsest approximation of the signal. So the signal can be represented by the following coefficients

$$f = g\left(\delta f_{-1,1}, \delta f_{-1,2}, \delta f_{-1,3}, \delta f_{-1,4}, \delta f_{-2,1}, \delta f_{-2,2}, \delta f_{-3,1}, f_{-3,1}\right)$$

where s=0 is assumed at the signals physical level, i.e., $f_{0,1} = 1, f_{0,2} = 0...$ etc. The relationship between the wavelet coefficients and the signal is:

$$\begin{bmatrix} f_{-3,1} \\ \delta f_{-3,1} \\ \delta f_{-2,1} \\ \delta f_{-2,2} \\ \delta f_{-1,1} \\ \delta f_{-1,2} \\ \delta f_{-1,3} \\ \delta f_{-1,4} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} \\ \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} & \frac{-1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} & \frac{-1}{2} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ -3 \\ 2 \\ 1 \\ 0 \\ 1 \\ 2 \end{bmatrix}$$

$$\begin{bmatrix} f_{-3,1} \\ \delta f_{-3,1} \\ \delta f_{-2,1} \\ \delta f_{-2,2} \\ \delta f_{-1,1} \\ \delta f_{-1,2} \\ \delta f_{-1,3} \\ \delta f_{-1,4} \end{bmatrix} = \begin{bmatrix} 1.4142 \\ -1.4142 \\ 1.0 \\ -1.0 \\ .70711 \\ -3.5355 \\ .70711 \\ -.70711 \end{bmatrix}$$

3.1.4 Multiscale Systems Theory

Willsky et.al. developed a Multiscale System Theory for stationary systems described as dynamic systems on dyadic trees, and they investigated the statistical properties of the resulting models.

The description of multiresolution representations as dynamic systems on trees provides a setting for the multiresolution modeling of signals and phenomena which in turn lead directly to powerful methods for statistically optimal multiresolution signal and image processing. In particular these observations led Willsky et.al. to examine scale recursive models for stochastic processes, leading to multiscale generalizations of Schur-Levison techniques and Kalman filtering. These and related algorithms for likelihood calculation lead to new methods for a variety of important signal and image processing problems, including multiscale data fusion , motion estimation in image sequences, and texture discrimination. Furthermore these new methods offer considerable advantages over previous methods in terms of computational efficiency, statistical optimality, explicit calculation of error statistics, estimates at multiple resolutions and so forth.

The Multiscale System Theory framework developed by Willsky, Benveniste and Nikoukhah [6] defines the shift operators on a dyadic tree, which in some sense represent the transfer functions in frequency domain.

This framework with some system theoretic properties like realization, causality and transfer functions provides a good environment to describe images, stationary signals and even some simple dynamic systems, but it does not cover the requirements for forced dynamic systems

regularly used in process control. The existing framework is good to represent static information in a concise and optimal manner, but it does not offer any flexibility for modeling, simulation and control. The following sections will deal with this issue and provide a general framework for multiscale linear systems theory.

3.2 Multiscale Models of Linear Systems

Wavelet transformation is a linear transformation. There are approximation methods but non-linear models in general cannot be transformed into the multiscale domain without any information loss. The next sections will describe the framework to transform linear models from time domain to frequency/time domain both numerically and symbolically. It is the symbolic transformation and generalization of it what makes the Multiscale Linear System Theory very valuable and useful. Many system theoretic properties of existing linear system theory can be ported without any modification into multiscale domain, resulting in the direct ability to use methods developed in time or frequency domains without any major changes.

3.3 From Time Domain to Time/Frequency Domain

The state, input, and measured variables of a dynamic process in a state-space or input-output model, are defined on the set of real numbers for continuous models, or the set of integers for discrete-time models. The wavelet transform of states, inputs and measurements allows us to define these variables in the time-scale domain, which is characterized by the nodes of a homogeneous tree. However, the character of the resulting dynamic model is not obvious, nor is the presence of any attractive characteristics that might facilitate or enhance the engineering tasks of simulating, estimating, controlling, and/or optimizing the dynamic behavior of linear processes. In this section we will attempt to elucidate these issues. Consider the input-output model given by the convolution integral

$$y(t) = \int_0^t g(t-\tau)\, u(\tau)\, d\tau \tag{3.12}$$

It is well known that the Fourier transform deconvolves $g(t-\tau)$ and $u(\tau)$ and leads to the following transfer function-based model in the frequency domain,

$$\bar{y}(\omega) = \bar{g}(\omega)\,\bar{u}(\omega) \tag{3.13}$$

Is it possible to establish a similar deconvolution of $g(t-\tau)$ and $u(\tau)$ under a wavelet transform and generate a transfer function-based model in the time-scale domain of the form

$$\tilde{y}(s) = \tilde{g}(s)\,\tilde{u}(s) \tag{3.14}$$

where, $\tilde{y}(s)$ and $\tilde{u}(s)$ are the wavelet coefficients of the corresponding variables at the node, s, of the associated homogeneous tree? The answer is no, as the following theorem states [15].

Theorem 3.1 *There is no wavelet function, which can transform the dynamic system of equation (3.12) into one of the form given by equation (3.14), i.e. into one where the transfer function and the input are completely deconvolved and separable.*

The result of Theorem 3.1 is not surprising. The transfer function model of equation (3.13) is possible because the e^{jwt} basis functions of the Fourier transform are also eigenfunctions of the differential operator. On the contrary, no wavelet function is an eigenfunction of the differential operator.

Simply, the transfer function we are looking for will not be related to the classical linear system theory, as is the Fourier-based transfer function in the frequency domain.

3.3.1 Homogeneous Linear systems

Consider the following multivariable, linear, discrete-time homogeneous model,

$$x_{k+1} = Ax_k \tag{3.15}$$

Using the indexing described in Figure (??) let us index the discrete-time values of the state, x_k, by the nodes at $s = 0$ (or equivalently, horocycle-0) of the dyadic tree, where the discretization interval is equal to the sampling period under which the model of equation (3.15) was generated. Then, equation (3.15) takes the following equivalent form,

$$x_{0,k+1} = A_0 x_{0,k} \tag{3.16}$$

where A_0 indicates the coefficient at scale zero, or the physical level. This model is still a purely temporal model. The transformation into multiscale domain occurs if Haar wavelet is applied to it,

$$x_{-1,\frac{k}{2}} = \frac{1}{\sqrt{2}}[x_{0,k} + x_{0,k+1}] \tag{3.17}$$

$$\delta x_{-1,\frac{k}{2}} = \frac{1}{\sqrt{2}}[x_{0,k} - x_{0,k+1}] \tag{3.18}$$

This is a straight forward implementation of the definition given at equations (3.7,3.8). Now substituting the original model into this equation,

$$x_{-1,\frac{k}{2}} = \frac{1}{\sqrt{2}}[x_{0,k} + A_0 x_{0,k}] = \frac{(I + A_0)}{\sqrt{2}} x_{0,k} \tag{3.19}$$

$$\delta x_{-1,\frac{k}{2}} = \frac{1}{\sqrt{2}}[x_{0,k} - A_0 x_{0,k}] = \frac{(I - A_0)}{\sqrt{2}} x_{0,k} \tag{3.20}$$

a relationship between scales 0 and -1 is formed. If we assume the sampling interval at scale 0 corresponds to T than at scale -1 the sampling interval is 2T because of the decimation built into the wavelet transformation. So ultimately equations (3.19,3.20) form a relationship between event occurring at different scales, thus they are aptly named Multiscale Models. For simplicity the multiscale model parameters will be named as,

$$A_{0,-1} = \frac{(I + A_0)}{\sqrt{2}} \tag{3.21}$$

$$\delta A_{0,-1} = \frac{(I - A_0)}{\sqrt{2}} \tag{3.22}$$

so that

$$x_{-1,\frac{k}{2}} = A_{0,-1}x_{0,k} \tag{3.23}$$

$$\delta x_{-1,\frac{k}{2}} = \delta A_{0,-1}x_{0,k} \tag{3.24}$$

The subscripts (0,-1) indicate that it is the model parameter between scales 0 and -1. See Figure (3-5) for more details. For consistency all multiscale models will be developed between a left child and its mother point. Through easy algebraic operations the relationship between the right child node and the mother point can be established.

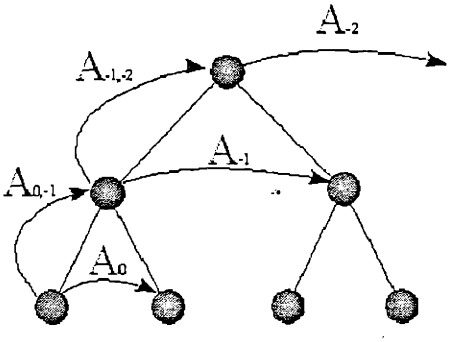


Figure 3-5: Model parameters at and between different scales

At scale -1 the model along the temporal direction becomes

$$\begin{aligned}
x_{-1,\frac{k}{2}} &= A_{0,-1}x_{0,k} \\
x_{-1,\frac{k}{2}+1} &= A_{0,-1}x_{0,k+2} \\
x_{0,k+2} &= A_0^2 x_{0,k} \\
x_{-1,\frac{k}{2}+1} &= A_{0,-1}A_0^2 x_{0,k} = A_0^2 A_{0,-1}x_{0,k} \\
x_{-1,\frac{k}{2}+1} &= A_0^2 x_{-1,\frac{k}{2}} = A_{-1}x_{-1,\frac{k}{2}}
\end{aligned} \tag{3.25}$$

From this

$$A_{-1} = A_0^2 \tag{3.26}$$

or in general (from the recursive nature)

$$A_{-m} = A^2_{-m+1} = A_0^{2^m} \tag{3.27}$$

Similarly

$$A_{-m+1,-m} = \frac{(I + A_0^{2^{m-1}})}{\sqrt{2}} \tag{3.28}$$

Remark 2 *Equation (3.23) provides a two-scale model of the linear system, since it relates the states at two distinct scales.*

Remark 3 *The state of the system at the higher scale, e.g.* $x_{-1,\frac{k}{2}} \rightarrow x_{-2,\frac{k}{4}}$ *or* $x_{-1,\frac{k}{2}+1} \rightarrow x_{-2,\frac{k}{4}}$ *captures all "past" history and plays a role analogous to that of* x_k *in the discrete-time model of equation (3.15).*

Remark 4 *For an n-th order, discrete-time, linear homogeneous system, it can be easily shown that the state at any node at level-m of the dyadic tree is given by the following multi-scale model*

$$x_{-m,k} = \sum_n \tilde{A}_{-m+n} x_{-m+n} \tag{3.29}$$

This representation does not change the order of the model. An nth-order model on time domain will remain nth order at any given scale along the temporal direction or across the scales. Of course, any higher order model can be converted into a first order model by state augmentation, too.

3.3.2 Forced Systems First and Higher Orders

Although models described in the previous section can be encountered in several physical systems most of the interesting systems are more complicated than that model. A more general model used extensively in linear control theory is

$$x_{k+1} = Ax_k + Bu_k \tag{3.30}$$

where u_k is an external input. This multivariable, linear discrete-time model can also be presented on the nodes of a dyadic tree by the following re-indexing

$$x_{0,k+1} = A_0 x_{0,k} + B_0 u_{0,k} \tag{3.31}$$

Like in the previous section a multiscale relationship can be formed by utilizing the Haar transformation from equation (3.17)

$$\begin{aligned} x_{-1,\frac{k}{2}} &= \frac{1}{\sqrt{2}}\left[x_{0,k} + x_{0,k+1}\right] \\ x_{-1,\frac{k}{2}} &= \frac{1}{\sqrt{2}}\left(I + A_0\right) x_{0,k} + \frac{1}{\sqrt{2}} B_0 u_{0,k} \\ x_{-1,\frac{k}{2}} &= \tilde{A}_{0,-1} x_{0,k} + \tilde{B}_{0,-1} u_{0,k} \end{aligned} \tag{3.32}$$

or in general

$$x_{-m-1,\frac{k}{2}} = \tilde{A}_{-m,-m-1} x_{-m,k} + \tilde{B}_{-m,-m-1} u_{-m,k} \tag{3.33}$$

where

$$\tilde{A}_{-m,-m-1} = \frac{\left(I + A_0^{2^m}\right)}{\sqrt{2}} \tag{3.34}$$

$$\tilde{B}_{-m,-m-1} = \frac{1}{\sqrt{2}} \left[\sum_{i=0}^{2^m-1} A_0^i\right] B_0 \tag{3.35}$$

This recursive relationship is only valid if the underlying states and inputs are correctly transformed, too. States are always averaged using equation (3.17), but inputs require a different averaging. As mentioned before there is no wavelet which can imitate the Fourier kernel to deconvolve the derivative operator. So a certain amount of convolution should be carried over the scales. To find out the special averaging for inputs, the dynamic model along the temporal

direction should be written for a higher scale.

$$\begin{aligned} x_{0,k+1} &= A_0 x_{0,k} + B_0 u_{0,k} \\ x_{0,k+2} &= A_0^2 x_{0,k} + A_0 B_0 u_{0,k} + B_0 u_{0,k+1} \\ x_{0,k+3} &= A_0^2 x_{0,k+1} + A_0 B_0 u_{0,k+1} + B_0 u_{0,k+2} \end{aligned}$$

These are the dynamic equations for 3 consecutive points on the time line (scale 0). From them two averages can be formed to express the states at scale -1

$$\begin{aligned} x_{-1,\frac{k}{2}} &= \frac{1}{\sqrt{2}} \left[x_{0,k} + x_{0,k+1} \right] \\ x_{-1,\frac{k}{2}+1} &= \frac{1}{\sqrt{2}} \left[x_{0,k+2} + x_{0,k+3} \right] = \\ &= \frac{1}{\sqrt{2}} \begin{bmatrix} A_0^2 x_{0,k} + A_0 B_0 u_{0,k} + B_0 u_{0,k+1} \\ + A_0^2 x_{0,k+1} + A_0 B_0 u_{0,k+1} + B_0 u_{0,k+2} \end{bmatrix} \\ &= \frac{A_0^2}{\sqrt{2}} \left[x_{0,k} + x_{0,k+1} \right] + \\ &\quad \frac{1}{\sqrt{2}} \left[A_0 B_0 u_{0,k} + (I + A_0) B_0 u_{0,k+1} + B_0 u_{0,k+2} \right] \\ x_{-1,\frac{k}{2}+1} &= A_0^2 x_{-1,\frac{k}{2}} + (I + A_0) B_0 u_{-1,\frac{k}{2}} \end{aligned} \tag{3.36}$$

where

$$u_{-1,\frac{k}{2}} = \frac{B_0^{-1} (I + A_0)^{-1}}{\sqrt{2}} \left\{ A_0 B_0 u_{0,k} + (I + A_0) B_0 u_{0,k+1} + B_0 u_{0,k+2} \right\} \tag{3.37}$$

In general

$$x_{-m,\frac{k}{2}+1} = \bar{A}_{-m} x_{-m,\frac{k}{2}} + \bar{B}_{-m} u_{-m,\frac{k}{2}} \tag{3.38}$$

$$u_{-m,\frac{k}{2}} = \frac{\bar{B}_{-m}^{-1}}{\sqrt{2}} \left\{ \begin{array}{c} \bar{A}_{-m+1} \bar{B}_{-m+1} u_{-m+1,k} + \left(I + \bar{A}_{-m+1} \right) \bar{B}_{-m+1} u_{-m+1,k+1} \\ + \bar{B}_{-m+1} u_{-m+1,k+2} \end{array} \right\} \tag{3.39}$$

with

$$\bar{A}_{-m} = A_0^{2^m} \tag{3.40}$$

$$\bar{B}_{-m} = \left[\sum_{i=0}^{2^m-1} A_0^i\right] B_0 \tag{3.41}$$

It is important to notice that the averaging scheme for the inputs is a natural outcome of the Haar transformation of the states. It is also interesting to note that the resulting averaging can be seen as a modified Hat wavelet transformation. Hat wavelet has the coefficients:

$$\phi_k = \left[\begin{array}{ccc} \frac{1}{2\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2\sqrt{2}} \end{array}\right] \quad \text{and} \quad \psi_k = \left[\begin{array}{ccc} -\frac{1}{2\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{2\sqrt{2}} \end{array}\right] \tag{3.42}$$

whereas the modified Hat has the following coefficients

$$\phi_k = \frac{\bar{B}_{-m}^{-1}}{\sqrt{2}}\left[\begin{array}{ccc} \bar{A}_{-m+1}\bar{B}_{-m+1} & \left(I+\bar{A}_{-m+1}\right)\bar{B}_{-m+1} & \bar{B}_{-m+1} \end{array}\right] \quad \text{and} \tag{3.43}$$

$$\psi_k = \frac{\bar{B}_{-m}^{-1}}{\sqrt{2}}\left[\begin{array}{ccc} -\bar{A}_{-m+1}\bar{B}_{-m+1} & \left(I+\bar{A}_{-m+1}\right)\bar{B}_{-m+1} & -\bar{B}_{-m+1} \end{array}\right] \tag{3.44}$$

The coefficients are model dependent and this fact reflects the inability of wavelets to deconvolve the dynamic model. Delayed inputs can be handled the same way, by just redefining the input terms participating in the averaging scheme.

3.3.3 Systems with Modeling Errors

Modeling systems with additive error terms is not difficult. Model Predictive Control formulation includes a constant term d in the dynamic model, but it can be a time varying variable, too

$$x_{k+1} = Ax_k + Bu_k + d_k \tag{3.45}$$

The multiscale model of this dynamic system can be formulated by first writing the model

on the nodes of the dyadic tree

$$x_{0,k+1} = A_0 x_{0,k} + B_0 u_{0,k} + d_{0,k} \tag{3.46}$$

and then writing the multiscale model

$$x_{-1,\frac{k}{2}} = \frac{1}{\sqrt{2}} (I + A_0) x_{0,k} + \frac{1}{\sqrt{2}} B_0 u_{0,k} + \frac{1}{\sqrt{2}} d_{0,k} \tag{3.47}$$

with the generalization

$$x_{-m-1,\frac{k}{2}} = \tilde{A}_{-m,-m-1} x_{-m,k} + \tilde{B}_{-m,-m-1} u_{-m,k} + \tilde{D}_{-m,-m-1} d_{-m,k} \tag{3.48}$$

where

$$\tilde{D}_{-m,-m-1} = \frac{1}{\sqrt{2}} \left[\sum_{i=0}^{2^m-1} A_0^i \right] \tag{3.49}$$

and

$$d_{-m,k} = \frac{1}{\sqrt{2}} \left\{ \bar{A}_{-m+1} d_{-m+1,2k} + \left(I + \bar{A}_{-m+1} \right) d_{-m+1,2k+1} + d_{-m+1,2k+2} \right\} \tag{3.50}$$

3.3.4 Systems with Unequal Number of States and Inputs

Almost all chemical processes have more states then inputs. The number of outputs on the other hand are fewer and quite frequently equal in number to the manipulated inputs. Since the transformation of time domain models into the multiscale domain requires inversion of the state space model parameters, the system of equations describing the dynamics of the states should be square. The easiest remedy for nonsquare systems is the introduction of dummy states or inputs. Since these models will be used in Model Predictive Control, and Model Predictive Control is based on optimization with constraints on states, outputs and inputs it is very easy to introduce new states and inputs and to constrain them to zero.

Example 2 *Converting a non-square system to a square system*

$$x_{k+1} = \begin{bmatrix} 0.1 & 0.3 & 0.2 \\ 0.7 & 0.2 & 0.5 \\ 0.6 & 0.2 & 0.4 \end{bmatrix} x_k + \begin{bmatrix} 1 & 2 \\ 2 & 7 \\ 0.3 & 3 \end{bmatrix} u_k$$

adding a new dummy input $u3_k$

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} 0.1 & 0.3 & 0.2 \\ 0.7 & 0.2 & 0.5 \\ 0.6 & 0.2 & 0.4 \end{bmatrix} x_k + \begin{bmatrix} 1 & 2 & 0 \\ 2 & 7 & 0 \\ 0.3 & 3 & 1 \end{bmatrix} \begin{bmatrix} u_k \\ u3_k \end{bmatrix} \\ \textit{subject to} \quad u3_k &= 0 \end{aligned}$$

The only important point in doing this addition is that the resulting matrix should be invertible.

3.3.5 Multirate Systems

Multirate systems are difficult to handle because they implicitly contain information at different scales or frequency bands. For each specific case one has to develop a new model. Multiscale formulation on the other hand provides a suitable framework to capture the behavior of the dynamic system at different sampling rates. In section 4.5 this representation is further exploited to represent different cases like multirate outputs or inputs or both simultaneously. The main idea behind the suitability of multiscale domain is that the inputs or states can be forced to remain constant over longer sampling intervals by just setting their corresponding wavelet (difference) coefficients to zero.

Reconstruction of the signal at finer scales then results in integration of dynamics at different scales, some changing at that finer scale (and thus nonzero wavelet coefficients) and some remaining constant as long as the sampling interval of the last scale at which they last had nonzero wavelet coefficients.

Example 3 *A very simple multirate system consists of two states and two inputs.* $u1_k$ *changes*

every sampling interval whereas $u2_k$ changes only every second sampling interval

$$x_{k+1} = \begin{bmatrix} 0.2 & 0.3 \\ 0.4 & 0,7 \end{bmatrix} x_k + \begin{bmatrix} 3 & 2 \\ 5 & 1 \end{bmatrix} \begin{bmatrix} u1_k \\ u2_k \end{bmatrix}$$

One way to express this multirate system in time domain would be

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} 0.2 & 0.3 \\ 0.4 & 0,7 \end{bmatrix} x_k + \begin{bmatrix} 3 \\ 5 \end{bmatrix} u1_k + \\ &\quad \begin{bmatrix} 2\left(\frac{[(-1)^{k+1}+1]}{2}\right) & 2\left(\frac{[(-1)^{k}+1]}{2}\right) \\ 1\left(\frac{[(-1)^{k+1}+1]}{2}\right) & 1\left(\frac{[(-1)^{k}+1]}{2}\right) \end{bmatrix} \begin{bmatrix} u2_{k-1} \\ u2_k \end{bmatrix} \end{aligned}$$

So for k=even cases $u2_k = u2_k$ and for k=odd $u2_k = u2_{k-1}$. But this formulation creates a time varying dynamic model and makes it difficult to use. Multiscale formulation of this problem redüces it again to a simple linear time invariant model subject to a equality constraint (blocking). The general formulation can be represented as

$$x_{-1,\frac{k}{2}} = \frac{1}{\sqrt{2}} (I + A_0) x_{0,k} + \frac{1}{\sqrt{2}} B_0 u_{0,k}$$

with

$$u2_{0,k+1} = u2_{0,k} \tag{3.51}$$

Since only the left nodes are defined through this representation the dyadic tree automatically handles the blocking of u2 without the necessity of a time varying model parameter.

3.4 Ideas on How to Model on Multiscale Domain

Transformation of linear models into multiscale domain allows generalization and duplication of the existing theories. One can describe this transformation as a re-grouping of variables such that they represent the frequency domain behavior in a far less convoluted manner. This direct transformation allows the use of existing tools and theories with minimal adjustments,

but being the extension of a limited representation (time domain models) it is also confined within the limits of linear systems theory.

A more general modeling paradigm is the multiscale domain itself. Instead of transforming models from time domain to the multiscale domain, models can be identified and created on the time/scale space. This type of models can capture scale dependent information with much more success and accuracy.

If models are transformed from time domain to time/scale domain then there is the additional very restrictive constraint of consistency and closure. Since computations at higher scales (coarser descriptions) should be correctly projected onto the finer scales (consistency) in order to correctly capture the time domain model at the physical level (closure) there is not much flexibility in terms of modeling and computations. On the other hand if models are created on the multiscale domain without considering that there is a need for closure or a definitive description of physical level then there is a great deal of freedom in terms of modeling and doing computations. Of course there is a need of a certain level of consistency but instead of being global this consistency requirement can be localized.

There is a lot of research around generating models across different scales. Semiconductor industry heavily depends on surface reactions in etching process and models describing this process have a big gap in terms of the scales involved. The reaction occurs on the surface and computed through Monte Carlo simulations, which are at very fine scales, whereas the gas phase is described through models which are at very coarse levels. Bringing these macro and micro scales together is achieved through trial and error based matching techniques. A continuous description of the whole process does not exist. Modeling in multiscale domain can fill this gap.

There are many other examples from chemical engineering field where micro scale properties define the bulk (macro or meso) properties but there are no consistent models describing both worlds.

Research on image analysis provided some tools to address this problem of multiscale or multiresolution modeling, but since images are just collection of data points and don't have necessarily a physical model, the knowledge is not readily expandable into other engineering fields.

This topic and the problems described in the following subsections define a very interesting

and hot research area. This thesis does not try to provide a solution to this issue but definitely techniques developed here can be readily applied to modeling in multiscale domain.

3.4.1 Identification of Models on Trees

Identification of models on the dyadic or higher order trees as described in the previous sections provides a great flexibility to incorporate frequency dependent characteristics of a system. There are many different approaches one can utilize to identify a model on the tree. One can either collect data at different frequencies and create different temporal models at each scale and reconcile it to satisfy the consistency requirement, or one can pose the problem as a large global identification problem subject to consistency constraints. The order of models at different scales does not need to be same, since at higher scales (lower frequencies) some fast components of the system can diminish. A reduced model can (a) reduce the complexity of the overall system (b) improve the robustness of the model.

3.4.2 Parameter Estimation

Having a linear model, identification of model parameters is a relatively easy task. The problem can be posed as a quadratic programing problem subject to consistency constraints and minimization of modeling error provides the model parameters in addition to the state estimates. Since the program contains constraints, Kalman Filter ideas cannot be used. A moving horizon parameter and state estimation algorithm can provide the solution and it is also compatible with Model Predictive Control algorithm. A multiscale moving horizon state estimator has been developed by Matthew Dyer and case studies demonstrated the efficiency of the algorithm.

3.4.3 Nonlinear Models: Multiscale Linearization (Multiband Modeling)

Since wavelet transformation is a linear transformation there is no explicit way to apply it to nonlinear systems, but if models are identified at multiple scales then the resulting model, even though it is linear, will not have a linear projection on the physical level. The closest model type on time domain would be time varying linear models, where the model is linear in terms

of states and inputs but nonlinear in terms parameters (bilinear in simplest case).

$$x_{k+1} = A_k x_k + B_k u_k \tag{3.52}$$

Sometimes nonlinearities arise at higher frequencies and they disappear at lower ones. In those cases such a representation can make the rich tool set of linear systems theory available to nonlinear systems. Again it is necessary to note that this thesis does not provide methods or algorithms to implement these ideas, but they are natural extensions of the ones described here.

3.5 Systems Theoretic Aspects

Since the wavelet transformation is a linear transformation the resulting multiscale domain models have exactly the same properties as their time domain projections. The additional flexibility in defining frequency domain properties in multiscale domain, though, gives rise to the need of redefining most important system theoretic properties of dynamic models in the multiscale domain. These definitions are mostly trivial extensions of time domain definitions, except in some cases where the multiscale domain has a more general and broader perspective. One should also notice that simple models in multiscale domain can have very convoluted and complex forms (same order, though) when they are projected onto the time domain.

3.5.1 Causality and Closure

The causality of the models in time domain is preserved after they are transformed into the multiscale domain, because the wavelet filter is a causal filter. The dyadic tree rendered according to the causality can be seen in figure (3-6). There is a more important issue related to causality though: closure of the models.

In going from the time-domain to the frequency-domain, during the Fourier transformation of a signal, we need only the transformation mapping; the two domains are orthogonal to each other. This is not the case as we go from the time-domain, where the discrete-time systems are defined, to the time-scale domain where the multiscale systems are defined. The consequences are interesting and need to be pointed out, as one contemplates the formulation of multi-scale models.

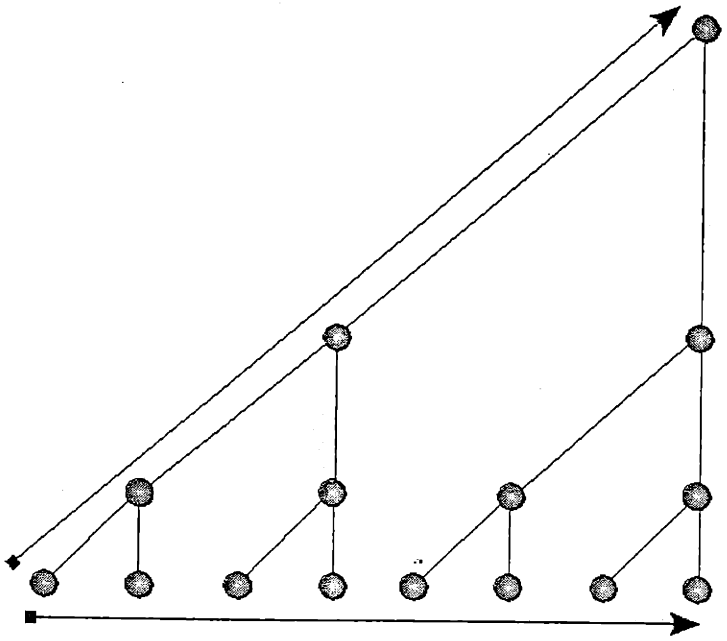


Figure 3-6: Direction of causality

Consider the two-scale, forced linear model of equation

$$\delta x_{-s,k} = A_{-s+1,-s} x_{-s+1,2k} + B_{-s+1,-s} u_{-s+1,2k} \tag{3.53}$$

Its variables are defined in the time-scale domain and their values are indexed by the nodes of a homogeneous dyadic tree. If we require that the model (3.53) in the time-scale domain is completely equivalent to the model (3.30) in the time-domain, we have imposed a requirement that constraints in a profound way any transformation from one domain to the other. This requirement can be exposed as follows:

"The values of the states and inputs on a homogeneous tree, evolving by equation (3.53), must achieve "closure", i.e. be equal, with the values of the states and inputs on the discrete-

time domain, evolving by equation (3.30)".

As a result of the "closure requirement" the two domains, i.e. the time-domain and the time-scale-domain, cannot be independent. As we move from the 1-dimensional domain of discrete-time (indexed by the integer numbers) to the 2-dimensional domain of time-scale (indexed by the nodes of a homogeneous tree) the additional freedom is taken up by the requirement that the state evolves by e.g. Haar scaling. As a result, the input variables are not free to be scaled by Haar, but must obey the resulting constraint. This leads to the modified Hat transform for the inputs.

It is rather straightforward to show that multiscale models with inputs transformed using Haar transformation (exactly like the states and outputs) cannot generate a discrete-time, causal model of the form (3.30) at any resolution. As a result, equations where inputs are scales using Haar wavelet cannot model adequately causal systems, where the inputs are tangible physical quantities, like those used as manipulated inputs for control. In this case the "closure requirement" is of essence and must be satisfied. (Stephanopoulos, Karsligil, Dyer [20])

One can further discuss a new property, which we can call computational causality. As it will be obvious later, most of the algorithms on homogenous dyadic trees involve some sort of sequential and parallel computations simultaneously. Section 3.6.3 discusses such an algorithm. The computations at a given node can only start if the information from the child or mother node is available depending on the direction of computational flow. The direction of the flow is not necessarily the temporal direction, and thus this definition of this type of causality is different from the classical definition.

3.5.2 Observability

The discrete-time system

$$x_{k+1} = A_0 x_k + B_0 u_k \tag{3.54}$$

with the output model

$$y_k = C_0 x_k \tag{3.55}$$

is equivalent to the multi-scale model

$$\delta x_{-s,k} = \frac{(I - A_{-s+1})}{\sqrt{2}} x_{-s+1,2k} - \frac{B_{-s+1}}{\sqrt{2}} u_{-s+1,2k} \tag{3.56}$$

with the output model

$$y_{-s,k} = C_{-s} x_{-s,k} \tag{3.57}$$

By analogy to the previous paragraph, let's define the observability of a multi-scale system as follows:

Definition 7 *The multi-scale system (3.56) with the output model (3.57), defined on a homogeneous tree, is observable, if, given a set of n measured outputs on the tree, we can compute the initial state on a node of the "initial time" path (figure(3-7)), provided that the measurements correspond to nodes of the tree which are at the same horocycle (scale) as the initial state or lower.*

The following theorem is a direct result of the above definition.

Theorem 3.2 *If the discrete-time system (3.54) with the output model (3.55) is observable, then the multi-scale model (3.56) with output model (3.57) is also observable, and vice versa.*

Proof. If system (3.54) with output model (3.55) is observable, then

$$rank \left[C^T | A^T C^T | \left(A^T\right)^2 C^T | \cdots | \left(A^T\right)^{2^m+(n-1)} C^T \right]^T = n \tag{3.58}$$

leading to the conclusion that at any scale-m, m = 0, 1, 2, 3,..., the corresponding discretized model

$$x_{-m,k+1} = \bar{A}_{-m} x_{-m,k} + \bar{B}_{-m} u_{-m,k} \tag{3.59}$$

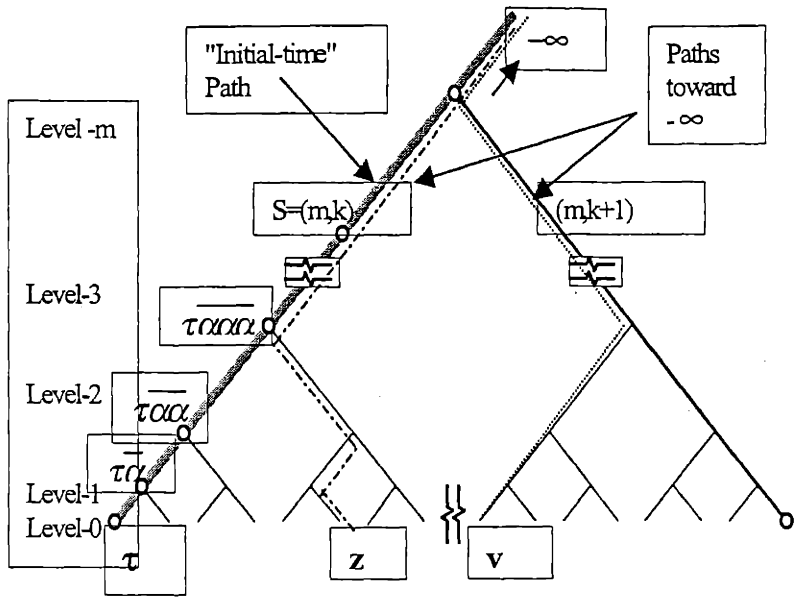


Figure 3-7: An infinite homogeneous tree, depicting the "Initial-Time" path. It is used in defining and proving conditions on stability, controllability and observability of multi-scale models on trees.

where

$$\bar{A}_{-m} = A_0^{2^m} \tag{3.60}$$

$$\bar{B}_{-m} = \left[\sum_{i=0}^{2^m-1} A_0^i\right] B_0 \tag{3.61}$$

with outputs

$$y_{-m,k} = C x_{-m,k} \tag{3.62}$$

is also observable. Consequently, given n measurements at n nodes of the same level (horocycle), we can compute the initial state at the node on the "initial time" path (figure(3-7)) and level-m.

Now, let us assume that the n measurements are at different level nodes on the "initial time" path. Then, assuming (without loss of generality) all inputs on this path to be zero, we

take,

$$
\begin{aligned}
y_{-m,k} &= Cx_{-m,k} && (3.63)\\
y_{-m-1,\frac{k}{2}} &= Cx_{-m-1,\frac{k}{2}} = C\left(I + A_{-m}\right)x_{-m,k} && (3.64)\\
y_{-m-2,\frac{k}{4}} &= Cx_{-m-2,\frac{k}{4}} = C\left(I + A_{-m-1}\right)\left(I + A_{-m}\right)x_{-m,k}\\
&= C\left(I + A_{-m}^2\right)\left(I + A_{-m}\right)x_{-m,k} && (3.65)\\
y_{-m-n+1,\frac{k}{2^{n-1}}} &= C\prod_{j=0}^{n-1}\left(I + A_{-m}^{2^j}\right)x_{-m,k}\\
&= C\left[\sum_{i=0}^{2^n-1} A_{-m}^i\right]x_{-m,k} && (3.66)
\end{aligned}
$$

The set of equations (3.63), (3.64), (3.65), (3.66) admits a unique solution, given condition (3.58).

When the measurements are distributed at various levels and are on different paths, similar reasoning, as above, allows the computation of the initial state on the "initial time" path and a level which is not lower than the level at which a measurement is available. (**Stephanopoulos, Karsligil,** Dyer [20]) ■

3.5.3 Controllability

Again, since systems (3.54) and (3.56) are equivalent, we expect that the controllability characteristics of the discrete-time system (3.54) will be possessed by the multi-scale system on a tree, and vice versa. We need though, to define the concept of controllability for a system on a tree.

Definition 8 *The multi-scale system of equation (3.56), defined on a homogeneous tree, is controllable, if, given an initial state at node, $\tau = (-s, k)$, on the left most branch of the homogeneous tree, the system can reach the desired state at any other node, $\gamma = (-m, n)$, of the tree, through a finite sequence of inputs, provided that*

- *distance between γ and τ is finite i.e. d(γ,τ)=finite, and*
- *the control inputs are at the same horocycle (scale) as γ or lower.*

The following theorem is a direct result of the above definition :

Theorem 3.3 *If the discrete-time system (3.54) is controllable, then the multi-scale model (3.56) is also controllable, and vice-versa.*

Proof. If system (3.54) is controllable, then

$$rank\left[B|AB|A^2B|\cdots|A^{n-1}B\right] = n \tag{3.67}$$

Where, n is the dimensionality of the state vector. Starting with node, $(-s,k)$, on the left most branch of the homogeneous tree, equation (3.56) yields

$$x_{-s-1,\frac{k}{2}} = \frac{(I+A_{-s})}{\sqrt{2}}x_{-s,k} + \frac{B_{-s}}{\sqrt{2}}u_{-s,k} \tag{3.68}$$

Take, without loss of generality, $x_{-s,k} = 0$. Then, recursive application of equation (3.68), along the nodes of the left most branch, yields the state at any node of the path, e.g.

$$x_{-s-1,\frac{k}{2}} = \frac{B_{-s}}{\sqrt{2}}u_{-s,k} \tag{3.69}$$

$$x_{-s-2,\frac{k}{4}} = \frac{\left(I+A_{-s}^2\right)B_{-s}}{2}u_{-s,k} + \frac{B_{-s-1}}{\sqrt{2}}u_{-s-1,\frac{k}{2}} \tag{3.70}$$

$$x_{-s-3,\frac{k}{8}} = \frac{\left(I+A_{-s}^4\right)\left(I+A_{-s}^2\right)B_{-s}}{2\sqrt{2}}u_{-s,k} + \frac{\left(I+A_{-s}^4\right)B_{-s-1}}{2}u_{-s-1,\frac{k}{2}} + \frac{B_{-s-2}}{\sqrt{2}}u_{-s-2,\frac{k}{4}} \tag{3.71}$$

$$x_{-s-n,\frac{k}{2^n}} = \sum_{l=1}^{n}\frac{1}{\sqrt{2}^l}\left[\prod_{j=n-l+1}^{n-1}\left(I+A_{-s}^{2^j}\right)\right]\left[\sum_{i=0}^{2^{(n-l)}-1}A_{-s}^i\right]B_{-s}u_{-s-n+l,\frac{k}{2^{n-l}}} \tag{3.72}$$

Equations (3.69), (3.70), (3.71), (3.72) can be solved for the unique solution in the inputs if and only if

$$rank\left[\left[\sum_{i=0}^{2^{n-1}-1}A_{-s}^i\right]B_{-s}|\left(I+A_{-s}^{2^{n-1}}\right)\left[\sum_{i=0}^{2^{n-2}-1}A_{-s}^i\right]B_{-s}|\cdots|\left(I+A_{-s}\right)^{-1}\left[\sum_{i=0}^{2^n-1}A_{-s}^i\right]B_{-s}\right] = n \tag{3.73}$$

or equivalently

$$rank\left[A_{-s}^{2^n-2^{n-1}}B_{-s}|A_{-s}^{2^n-2^{n-2}}B_{-s}|A_{-s}^{2^n-2^{n-3}}B_{-s}|\cdots|A_{-s}^{2^n-1}B_{-s}\right]=n$$

or

$$rank\left[B_{-s}|A_{-s}^{2^1}B_{-s}|A_{-s}^{2^2}B_{-s}|\cdots|A_{-s}^{2^{n-1}}B_{-s}\right]=n \tag{3.74}$$

which is true, given (3.67). Therefore, we can bring the zero state $x_{-s,k}$ to a desired value at any node on the left most branch of the tree, at a level higher than that of $(-s,k)$. If the nodes are in the same scale $(-s)$ then proving the controllability is trivial using the dynamic equation (3.38) ■

3.5.4 Stability

The fact that time domain models are equivalent to multiscale models implies that the stability characteristics of the discrete-time model are passed on to the multi-scale model on the tree, and vice versa. Note that since the stability of dynamic systems is an asymptotic property as time $\rightarrow+\infty$, in any stability treatment of a multiscale system we should consider a homogeneous tree with an infinite number of nodes at each level. However, before proceeding we need to define the concept of stability for a model on a tree.

Definition 9 *Consider the left most branch of an infinite tree. All nodes of the tree are considered to be to the "right" of the left most branch. The multiscale system of equation (3.56), defined on a homogeneous tree, is ℓ_p-stable in the Lyapunov sense, if, given a bounded state at any node, $(-s,k)$, on the left most branch i.e. $\|x_{-s,k}\|_p \leq C$, then the value of the state at any other node of the tree, computed through the recursive application of model (3.56) with bounded inputs, is also bounded.*

The following theorem is a direct result of the above definition:

Theorem 3.4 *If the discrete-time system of equation (3.54) is ℓ_p-stable in the Lyapunov sense, then the multi-scale system of equation (3.56) is also ℓ_p-stable in the Lyapunov sense, and vice versa.*

Proof. If a system (3.54) ℓ_p-stable in a Lyapunov sense, then $0 < ||A||_p \leq 1$ and consequently $0 < \left|\left|A^{2^m}\right|\right|_p \leq 1$ Therefore, the homogenous discrete-time model

$$x_{-m,k+1} = A_0^{2^m} x_{-m,k}$$

is ℓ_p-stable in the Lyapunov sense at any level, m, of discretization, and the state of the multiscale system is bounded at any node of the infinite tree.

To prove the reverse, consider the node, $(-s,k)$, on the left most branch and a node, $\left(-s-l,\frac{k}{2^l}\right)$, along the same path (i.e. the left most branch). Using equation (3.56),

$$x_{-s,k} = \sqrt{2}^l \left(I + A_{-s}\right)^{-1} \left(I + A_{-s}^2\right)^{-1} \cdots \left(I + A_{-s}^{2^l}\right)^{-1} x_{-s-l,\frac{k}{2^l}} \tag{3.75}$$

or

$$\left|\left|x_{-s-l,\frac{k}{2^l}}\right|\right|_p = \frac{1}{\sqrt{2}^l} \left|\left|I + A_{-s}\right|\right| \left|\left|I + A_{-s}^2\right|\right| \cdots \left|\left|\left(I + A_{-s}^{2^l}\right)\right|\right| ||x_{-s,k}||_p \tag{3.76}$$

If $||x_{-s,k}||_p$ is bounded, then $\left|\left|x_{-s-l,\frac{k}{2^l}}\right|\right|_p$ is bounded for any l, as $l \to \infty$, if and only if $||A_{-s}||_p \leq 1$, which implies the ℓ_p-stability of system (3.54). Since system (3.56) is ℓ_p-stable in the Lyapanov sense, then if $||x_{-s,k}||_p$ is bounded, the state is also bounded at any node on the same horocycle as $(-s,k)$, i.e. $(-s,k+v)$ for $v \geq 1$, are all bounded. Then, the state at any node along the path from nodes $(-s,k+v)$. towards $-\infty$ is bounded implying $||\mathrm{A}_{-s}||_p \leq 1$. ■

3.6 Application of This New Framework

Multiscale Systems Theory can be applied in any area where linear systems theory has been applied before. Since this new framework is an extension to the existing theories every activity can be transformed into multiscale domain and performed there. As a result all the results can be duplicated. Furthermore since the new formulation provides a more attractive environment for parallel computations (see the next sections) and other computational efficiencies, the algorithms will work faster and larger problems can be solved. Localization of frequency ranges (scales) also allows utilization of powerful frequency domain techniques directly in time domain

even when there are hard constraints rendering the problem nonlinear and the use of frequency domain techniques impossible (for example Model Predictive Control with constraints on inputs). The following sections will briefly discuss how the new framework can be applied in different activities of process design and control.

3.6.1 Modeling

Previous sections described how models can be transformed into multiscale domain and how they can be formed in multiscale domain. There are many physical systems which have multiscale attributes. Self similarity, fractal nature, multi rate behavior, stiff ODE systems (reaction networks, set of dynamic models with distinct characteristic times etc.) are best described in an multiresolution / multiscale environment.

3.6.2 Estimation

State and parameter estimation can be performed on the dyadic tree and thus in multiscale domain much faster than the pure time domain. The representation allows simultaneous parallel computations and thus reduces the complexity with respect to the number of variables. Willsky et .al. [9] showed that Kalman Filter can be defined in multiscale domain. Their research also showed that the new formulation allows a better representation of the error statistics at multiple scales.

A major weakness of Kalman filtering is though that it cannot handle constraints. Kalman Filter is the mathematical dual of LQR and LQR does not contain constraints either. Model Predictive Control on the other hand can handle constraints on states and inputs, and this property is its strongest point. The extensive use of MPC in industry gave rise to the need of a state estimator which can handle the same constraints. Moving horizon state estimation algorithm developed by Rawling et.al. [18] provided a nice solution. A similar algorithm can be posed in multiscale domain which can perform exactly same as its time domain version but in addition it can handle multi rate measurements by optimally fusing data, can provide error statistics at each scale, which can generate very valuable information for model adaptation and above all can reduce the complexity of the problem at hand drastically by parallelizing the algorithm. Detailed information on the development of the algorithm and some case studies

can be found in the thesis of Dyer. [12] We will demonstrate here only the problem formulation and give some directions how it can be solved.

The moving horizon state estimation problem in time domain can be represented as follows:

$$\min_{\hat{x}_k} x_{0|-1}^{eT} P_{0|-1}^{-1} x_{0|-1}^{e} + \sum_{k=0..n} \hat{v}_k^T R_k^{-1} \hat{v}_k + \hat{w}_k^T Q_k^{-1} \hat{w}_k \tag{3.77}$$

subject to

$$\begin{aligned} \hat{x}_{k+1} &= A\hat{x}_k + Bu_k + \hat{w}_k \\ y_k &= C\hat{x}_k \\ \underline{g} &\leq G\hat{x}_k \leq \overline{g} \\ \underline{h} &\leq \hat{w}_k \leq \overline{h} \end{aligned}$$

where $\hat{x}_k$ is the resulting estimate of the state and $\hat{w}_k$ the associated modeling error or disturbance. The hat notation implies that they are both obtained from the solution to an estimation problem, as opposed to being true physical values, such as the measurement, y, and the process input, u.

The difficulty in applying the multiscale transformation directly to this cost function and constraints is the fact that using Haar transformation for the modeling error produces awkward state descriptions at higher levels. Matthew Dyer's [12] thesis delves in this problem. The resulting optimum description of the multiscale state estimation algorithm is as follows:

$$\begin{aligned} \Phi = & \sum_{\theta \in M} [\left(y_{top,1}^{\theta} - C\hat{x}_{top,1}\right)^T R_{top}^{\theta-1} \left(y_{top,1}^{\theta} - C\hat{x}_{top,1}\right) + \\ & \sum_{s,k \in T} \left(\delta y_{-s,k}^{\theta} - C\delta\hat{x}_{-s,k}\right)^T R_{-s,k}^{\theta-1} \left(\delta y_{-s,k}^{\theta} - C\delta\hat{x}_{-s,k}\right) + \qquad (3.78) \\ & + \sum_{s,k \in T} \left(\left(I + A_{-s}\right)\delta\hat{x}_{-s,k} - \left(I - A_{-s}\right)\hat{x}_{-s,k} + \sqrt{2}B_{-s}u_{-s,k}\right)^T \left(\sqrt{2}B_{-s}\right)^{-T} Q_{-s,k}^{-1} \cdots \\ & \cdots \left(\sqrt{2}B_{-s}\right)^{-1} \left(\left(I + A_{-s}\right)\delta\hat{x}_{-s,k} - \left(I - A_{-s}\right)\hat{x}_{-s,k} + \sqrt{2}B_{-s}u_{-s,k}\right) \qquad (3.79) \end{aligned}$$

subject to

$$\underline{g}_{-s} \leq G\hat{x}_{-s,k} \leq \overline{g}_{-s}$$

This formulation can incorporate different set of measurements (θ) and fuse them optimally to generate the optimum state estimates. One can control the level of errors at each scale and penalize them differently (frequency weighted cost functions). Formulated as quadratic program this formulation can easily handle constraints and finally a correct choice of weights generates exactly the same results as in (3.77).

3.6.3 Control

One of the most suitable application areas is the linear control theory. Traditionally linear control theory has been developed in frequency domain and very powerful tools came out of this research effort. During the last 20-25 years the demand for plant wide control became very strong. Simple localized controllers started becoming insufficient. Model Predictive Control and other advance control algorithms emerged as a result, but the nature of process control required equipments to work at physical constraints to maximize economic benefit. This requirement made the underlying models nonlinear preventing the use of powerful frequency domain techniques. Time domain methods like MPC have been developed, and some ad. hoc. methodologies are used to specify the frequency domain characteristics.

Multiscale system theory can successfully fill this gap. Being localized in both time and frequency domain it allows specifications in both. One can put bounds on time domain variables to satisfy physical limitations and one can put bounds on scale dependent variables to satisfy frequency domain limitations or requirements like suppression of high frequency changes or in general shaping the loop of frequency response. Of course the trade-offs between designing in time or frequency domain still exist. For a given set of specification it is possible that there is no feasible solution, but the framework allows correct specification of the requirements in a clear way.

One major reason why multiscale system theory can be used in optimal control and in related advanced control algorithms is the equivalency of the objective functions and constraints. The

wavelet transformation regroups the variables such that the new set of variables represent both frequency and time domain properties simultaneously at the mean time the objective function remains the same, i.e. the optimal solution is the same as in the pure time domain problem formulation.

Let us formulate a very simple optimal control problem and its transformation and show that they are equivalent:

$$\min_{u} (R - x)^T (R - x) \tag{3.80}$$

subject to

$$x_{k+1} = A_0 x_k + B_0 u_k \tag{3.81}$$

In order to transform this optimal control problem into multiscale domain we need to transform all of the variables, x, u, R:

$$\tilde{x} = Wx = \begin{bmatrix} x_{-smax,1} \\ \delta x_{-smax,1} \\ \vdots \\ \delta x_{-1,\frac{k_{max}}{2}} \end{bmatrix}$$

$$\tilde{R} = WR = \begin{bmatrix} R_{-smax,1} \\ \delta R_{-smax,1} \\ \vdots \\ \delta R_{-1,\frac{k_{max}}{2}} \end{bmatrix}$$

where

$$W = \begin{bmatrix} \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} \\ \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} & \frac{-1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} & \frac{-1}{2} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

for a system with 7 inputs, i.e. k_{max}=8. One important property of this matrix W is that its rows are orthogonal to each other. As a result W is a unitary matrix, meaning

$$W^T W = I \tag{3.82}$$

This property retains the equivalency.

Claim 3.5 *Time domain objective function is identical to its multiscale version*

Proof. Starting from the multiscale objective function, one can rewrite the variables as transformations and extract the unitary wavelet transformation matrix.

$$\begin{aligned} \left(\tilde{R} - \tilde{x}\right)^T \left(\tilde{R} - \tilde{x}\right) &= (WR - Wx)^T (WR - Wx) \\ &= (R - x)^T W^T W (R - x) \\ &= (R - x)^T (R - x) \end{aligned}$$

■

After showing that the objective functions are identical we also have to show that the underlying dynamic model is the same as in the time domain. As shown in the previous sections the most ideal representation for the dynamic model in multiscale domain is

$$\delta x_{-s,k} = A_{-s+1,-s} x_{-s+1,2k} + B_{-s+1,-s} u_{-s+1,2k} \tag{3.83}$$

where

$$A_{-s+1,-s} = \frac{(I - A_0^{2^s})}{\sqrt{2}} \tag{3.84}$$

$$B_{-s+1,-s} = -\frac{1}{\sqrt{2}} \left[\sum_{i=0}^{2s-1} A_0^i \right] B_0 \tag{3.85}$$

The solution algorithm for this unconstrained optimal control problem can be sketched as:

- Start with the initial point $x_{0,1}$, minimize $(\delta R_{-1,1} - \delta x_{-1,1})^T (\delta R_{-1,1} - \delta x_{-1,1})$ to determine the value of $u_{0,1}$ (from the state equation $\delta x_{-1,1} = A_{0,-1} x_{0,1} + B_{0,-1} u_{0,1}$).
- Once $u_{0,1}$ is determined compute the value of $x_{-1,1} = \tilde{A}_{0,-1} x_{0,1} + \tilde{B}_{0,-1} u_{0,1}$
- Start with the initial point $x_{-1,1}$, minimize $(\delta R_{-2,1} - \delta x_{-2,1})^T (\delta R_{-2,1} - \delta x_{-2,1})$ to determine the value of $u_{-1,1}$ (from the state equation $\delta x_{-2,1} = A_{-1,-2} x_{-1,1} + B_{-1,-2} u_{-1,1}$).
- Once $u_{-1,1}$ is determined compute the value of $x_{-2,1} = \tilde{A}_{-1,-2} x_{-1,1} + \tilde{B}_{-1,-2} u_{-1,1}$ and $x_{-1,2} = A_{-1} x_{-1,1} + B_{-1} u_{-1,1}$
- Start with the initial point $x_{-2,1}$, minimize $(\delta R_{-3,1} - \delta x_{-3,1})^T (\delta R_{-3,1} - \delta x_{-3,1})$ to determine the value of $u_{-2,1}$ (from the state equation $\delta x_{-3,1} = A_{-2,-3} x_{-2,1} + B_{-2,-3} u_{-2,1}$).
  - At the meantime start with the initial point $x_{-1,2}$, minimize $(\delta R_{-1,2} - \delta x_{-1,2})^T (\delta R_{-1,2} - \delta x_{-1,2})$ to determine the value of $u_{0,3}$, from the state equation

$$\begin{aligned}
\delta x_{-1,2} &= \frac{(I - A_0)}{\sqrt{2}} x_{0,3} - \frac{B_0}{\sqrt{2}} u_{0,3} \\
x_{-1,2} &= \frac{(I + A_0)}{\sqrt{2}} x_{0,3} + \frac{B_0}{\sqrt{2}} u_{0,3} \\
x_{0,3} &= \sqrt{2} (I + A_0)^{-1} x_{-1,2} - (I + A_0)^{-1} B_0 u_{0,3} \\
\delta x_{-1,2} &= \frac{(I - A_0)}{\sqrt{2}} \left[\sqrt{2} (I + A_0)^{-1} x_{-1,2} - (I + A_0)^{-1} B_0 u_{0,3} \right] - \frac{B_0}{\sqrt{2}} u_{0,3} \\
&= (I - A_0) (I + A_0)^{-1} x_{-1,2} - \left[\frac{(I - A_0) (I + A_0)^{-1}}{\sqrt{2}} + \frac{I}{\sqrt{2}} \right] B_0 u_{0,3} \\
&= (I - A_0) (I + A_0)^{-1} x_{-1,2} - \sqrt{2} B_0 u_{0,3}
\end{aligned}$$

– Once $u_{0,3}$ is determined calculate the value of $u_{0,2}$ from the modified Hat transformation:

$$\begin{aligned} B_0 u_{-1,1} &= \frac{(I+A_0)^{-1}}{\sqrt{2}} \left[A_0 B_0 u_{0,1} + (I+A_0) B_0 u_{0,2} + B_0 u_{0,3}\right] \\ u_{0,2} &= \sqrt{2} u_{-1,1} - B_0^{-1} (I+A_0)^{-1} A_0 B_0 u_{0,1} - B_0^{-1} (I+A_0)^{-1} B_0 u_{0,3} \end{aligned}$$

These steps are repeated at each scale. The basic idea behind is to go one scale up and at the meantime to go one scale down at the left hand side. Figure (3-8) depicts the steps of this algorithm. As one can see some of the steps can be performed in parallel, speeding the computation and reducing the complexity by implementing a divide and conquer type of algorithm.

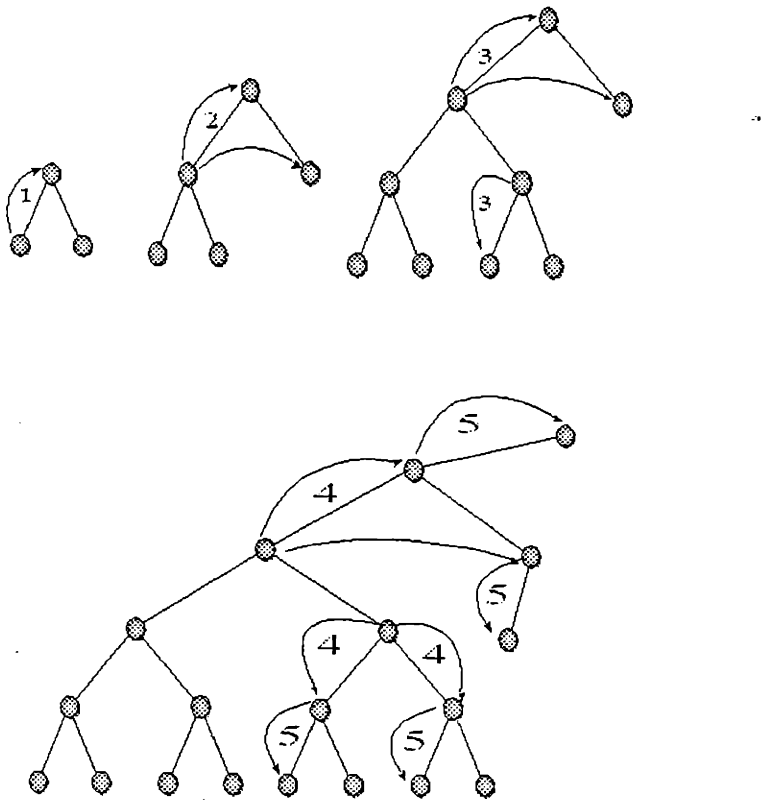


Figure 3-8: Steps of the optimization algorithm

Next chapter deals with the formulation of Multiscale Model Predictive Control and con-

straints on states and inputs are a part of it, so that issue will not be discussed here, but one important point to keep in mind is that the inputs in the dynamic equation are not transformed using the Haar wavelets, but the modified Hat wavelet. This brings additional burden if there are constraints on inputs. To guarantee the feasibility of the problem, i.e. the satisfaction of those constraints, some steps of the algorithm described above should be solved either simultaneously or iteratively. Both of these ideas will be discussed in the next chapter.

The benefit of this formulation besides being more efficient computationally is its ability to define scale dependent weights in the objective function.

$$\min \left(\tilde{R} - \tilde{x}\right)^T Q \left(\tilde{R} - \tilde{x}\right) \tag{3.86}$$

The elements of Q can be chosen such that deviations at some scales are penalized more than other. An obvious choice would be penalizing lower scales more in order to prevent high frequency changes.

Of course one important tool to shape the frequency response of the resulting outputs is the select a well designed reference path. Most of the control problems defined in the realm of optimal control have an additional penalty term:

$$\min \left(R - x\right)^T \left(R - x\right) + u^T P u \tag{3.87}$$

The reason to insert such a term into the objective function is two fold:

1. Soft constraints on inputs: Selection of the weight P determines how much input energy one is willing to exert onto the system in order to minimize the deviation from the reference path. This type of penalty terms are used extensively and the success of LQR formulation depends partially on this flexibility. One major weakness of this representation is though that there are some constraints that cannot be expressed as soft constraints or their weights should be set very high (infinity) to satisfy physical requirements.

2. Stabilizing effect: If the reference path and/or the underlying dynamic model is/are such that the resulting controller is unstable (like in non-minimum phase plants) then the additional penalty term in the objective function above pushes the positive poles to the

left and thus stabilizes the system.

The same effect can be generated if the reference path is designed accordingly. The simplest way to achieve this goal is to use a dynamic model for the reference path with the following form:

$$R_{k+1} = AR_k + (B + L)\, u_k \tag{3.88}$$

where

$$P = \begin{bmatrix} L & 0 & 0 & 0 \\ AL & L & 0 & 0 \\ \vdots & \vdots & \ddots & 0 \\ A^{N-1}L & A^{N-2}L & \cdots & L \end{bmatrix} \tag{3.89}$$

So it is relatively easy to formulate a nominal path we want the controller to follow and the additional penalty term in equation (3.87) can be incorporated in the reference path. This is a very useful fact since the wavelet transformation of inputs requires the use of modified Hat wavelets and the convolution one has to carry from scale to scale creates mathematical inefficiencies.

Bibliography

[1] B. Bakshi and G. Stephanopoulos. Wave-net: A multiresolution, hierarchical neural network with localized learning. *AICHE Journal*, 39, 1993.

[2] B. Bakshi and G. Stephanopoulos. Represenatation of process trends-III: Multiscale extraction of trends from process data. *Computers Chem. Engng*, 18:267–302, 1994.

[3] M. Basseville, A. Benveniste, K. Chou, S. Golden, R. Nikoukhah, and A. Willsky. Modeling and estimation of multiresolution stochastic processes. *IEEE Transactions in Information Theory*, 38, 1992.

[4] M. Basseville, A. Benveniste, and A. Willsky. Multiscale-autoregressive processes, part i: Lattice structures for whitening and modeling. *IEEE Transactions on Signal Processing*, 40, 1992.

[5] M. Basseville, A. Benveniste, and A. Willsky. Multiscale-autoregressive processes, part i: Schur-levinson parametrizations. *IEEE Transactions on Signal Processing*, 40, 1992.

[6] A. Benveniste, R. Nikoukhah, and A. Willsky. Multiscale system theory. *IEEE Transactions on Circuits and Systems-1: Fundamental Theory and Applications*, 41, 1994.

[7] J. Carrier and G. Stephanopoulos. Wavelet-based modulation in control relevant process identificaton. *AICHE Journal*, 1997.

[8] K. Chou. *A Stochastic Approach to Multiscale Signal Processing.* PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1991.

[9] K. Chou, A. Willsky, and R. Nikoukhah. Multiscale systems, kalman filters, and ricatti equations. *IEEE Transactions on Automatic Control*, 39, 1994.

[10] I. Daubechies. The wavelet transform, time-frequency localization and signal analysis. *IEEE Trans. Inf. Theory*, 36:961–1005, 1990.

[11] I. Daubechies. Ten lectures on wavelets. *SIAM*, 1992.

[12] M. Dyer. *A Multiscale Approach to State Estimation with Applications in Process Operability Analysis and Model Predictive Control.* PhD thesis, Massachusetts Institute of Technology, 2000.

[13] P. Fieguth. *Application of Multiscale Estimation to Large Scale Multidimensional Imaging and Remote Sensing Problems.* PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1995.

[14] W. Irving. *Multiscale Stochastic Realization and Model Identification with Applications to Large-Scale Estimation Problems.* PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1995.

[15] A. Lindsey. The non-existence of a wavelet function admitting a wavelet transform convolution theorem of the fourier type. *preprint, Ohio University, Athens, OH*, 1994.

[16] M. Luettgen. *Image Processing with Multiscale Models.* PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, 1993.

[17] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Patter Analysis and Machine Intelligence*, II:674–693, 1989.

[18] D. Robertson and J. Rawlings. A moving horizon-based approach for least squares state estimation. *AICHE Journal*, 42:2209–2223, 1994.

[19] G. Stephanopoulos. The shell process control workshop. 1989.

[20] G. Stephanopoulos, O. Karsligil, and M. Dyer. Multiscale aspects in linear and nonlinear estimation and control. *Proceedings of NATO-ASI on: Nonlinear Model Predictive Control*, 1997.

Chapter 4

Multiscale Model Predictive Control

This chapter will first define the Model Predictive Control (MPC) algorithm in multiscale domain. In order to make the transformations as transparent as possible at some points detailed derivations will be given. Second, the underlying optimization problem will be investigated and a solution algorithm will be devised. Towards the end of the chapter the new formulation and its properties will be used to address the weaknesses of classical MPC.

4.1 Introduction

Issues related to classical MPC described in the previous chapters can be addressed with the tools provided by the multiscale systems theory. There is a need of faster computations, and parallelizable algorithms for which multiscale domain can provide a solution. The classical MPC formulation cannot utilize frequency domain controller design methodologies, because of the existence of hard constraints, which cannot be violated. Multiscale domain framework, being localized both in time and scale (frequency), provides a convenient environment to express frequency domain specifications in a time domain-like fashion. One of the most important issues in classical MPC, the horizon length, can be addressed through a very different approach in multiscale domain. The resulting algorithm is a variable horizon one, satisfying feasibility and optimality requirements. Multiscale formulation of the state estimator also allows incorporation of multiple scale error statistics, incorporating more from the history of the plant and a richer feedback error.

4.2 Formulation of the Multiscale Model Predictive Control

Let us start from the dynamic model described in the previous chapter:

The multiscale state space model for the Multiscale MPC (MSMPC) is

$$\delta x_{-s-1,\frac{k}{2}} = \frac{(I-(A_{-s})_{n\times n})}{\sqrt{2}} x_{-s,k} - \frac{(B_{-s})_{n\times m}}{\sqrt{2}} u_{-s,k} - \frac{(D_{-s})}{\sqrt{2}} d_{-s,k} \tag{4.1}$$

$$y_{-s,k} = (C_{-s})_{m\times n}\, x_{-s,k} \tag{4.2}$$

where

$$A_{-s} = A_0^{2^s} \tag{4.3}$$

$$B_{-s} = \left[\sum_{i=0}^{2^s-1} A_0^i\right] B_0 \tag{4.4}$$

$$D_{-s} = \sum_{i=0}^{2^s-1} A_0^i \tag{4.5}$$

$$C_{-s} = C_0 \tag{4.6}$$

We choose to use the wavelet models describing $\delta x_{-s-1,\frac{k}{2}}$ instead of $x_{-s-1,\frac{k}{2}}$, because the set of wavelet coefficients form a non-redundant set of variables.

The objective function can be derived from the regular MPC formulation by transforming it into multiscale domain:

$$\min_{x} (\mathbf{R}-\mathbf{x})^T W^T W (\mathbf{R}-\mathbf{x}) =$$
$$\min_{\delta x} \left(\tilde{\mathbf{R}}-\tilde{\mathbf{x}}\right)^T \left(\tilde{\mathbf{R}}-\tilde{\mathbf{x}}\right)$$

where

$$\tilde{\mathbf{R}} = \mathbf{WR} = \begin{bmatrix} R_{-smax,1} \\ \delta R_{-smax,1} \\ \vdots \\ \delta R_{-1,\frac{k+N}{2}} \end{bmatrix}_{mN\times 1}, \tilde{\mathbf{x}} = \mathbf{Wx} = \begin{bmatrix} x_{-smax,1} \\ \delta x_{-smax,1} \\ \vdots \\ \delta x_{-1,\frac{k+N}{2}} \end{bmatrix}_{mN\times 1}$$

and $W^TW = I$ (orthogonal wavelet transformation matrix). This transformation preserves the original objective function both in value and solution. The transformation matrix W for the Haar wavelet can be generated through simple recursive computations. The matrix below is the W matrix for a SISO system with 3 scales (or 8 state points in time):

$$W = \begin{bmatrix} \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} \\ \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} & \frac{-1}{\sqrt{8}} \\ \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} & \frac{-1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{-1}{2} & \frac{-1}{2} \\ \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{\sqrt{2}} & \frac{-1}{\sqrt{2}} \end{bmatrix}$$

The coefficients at the top nodes $R_{-s\max,1}$ and $x_{-s\max,1}$ are fixed once an initial condition ($R_{0,1} = x_{0,1}$) is given. They can be represented as a linear combination of the initial condition on the time line and the wavelet coefficients on the left most branch of the homogenous tree. Since the coefficients at these nodes are redundant with the rest of the nodes, they should be expressed in terms of the non-redundant variables

$$x_{-s\max,1} = H_{m\times(N-1)}\delta\hat{x} + hR_{0,1} \tag{4.7}$$

$$R_{-s\max,1} = H_{m\times(N-1)}\delta\hat{R} + hR_{0,1} \tag{4.8}$$

where

$$\delta\hat{\mathbf{R}} = \begin{bmatrix} \delta R_{-smax,1} \\ \vdots \\ \delta R_{-1,\frac{k+N}{2}} \end{bmatrix}_{m(N-1)\times 1}, \delta\hat{\mathbf{x}} = \begin{bmatrix} \delta x_{-smax,1} \\ \vdots \\ \delta x_{-1,\frac{k+N}{2}} \end{bmatrix}_{m(N-1)\times 1}$$

The vectors H and h can be computed by the following relationship:

$$R_{-s\max,1} = -\sum_{i=0}^{n-1} \sqrt{2}^i \delta R_{-s\max+i,1} + \sqrt{2}^n R_{0,1} \tag{4.9}$$

$$x_{-s\max,1} = -\sum_{i=0}^{n-1} \sqrt{2}^i \delta x_{-s\max+i,1} + \sqrt{2}^n x_{0,1} \tag{4.10}$$

The vector H for a 3 scale system becomes

$$H = \begin{bmatrix} -1 & -\sqrt{2} & 0 & -2 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \delta R_{-3,1} \\ \delta R_{-2,1} \\ \delta R_{-2,2} \\ \delta R_{-1,1} \\ \delta R_{-1,2} \\ \delta R_{-1,3} \\ \delta R_{-1,4} \end{bmatrix} \tag{4.11}$$

and the vector h in this SISO case is a scalar of the value $2\sqrt{2}$.

So the objective function becomes

$$\begin{aligned} &\min_{\delta\hat{\mathbf{x}}} \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)^T \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) + \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) H^T H \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) = \\ &\min_{\delta\hat{\mathbf{x}}} \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)^T \left(I + H^T H\right) \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) \end{aligned} \tag{4.12}$$

Furthermore, the state equations over the horizon with length N $\left(\text{or scale } smax = \frac{\log(N)}{\log(2)}\right)$ can compactly be described as

$$\begin{aligned} \delta\hat{\mathbf{x}} &= \tilde{\mathbf{A}} x_{0,1} + \tilde{\mathbf{B}} \begin{pmatrix} u_{-smax+1,\frac{k}{2^{smax-1}}} \\ \vdots \\ u_{0,N-1} \end{pmatrix} + \tilde{\mathbf{D}} \begin{pmatrix} d_{-smax+1,\frac{k}{2^{smax-1}}} \\ \vdots \\ d_{0,N-1} \end{pmatrix} \\ \delta\hat{\mathbf{x}} &= \tilde{\mathbf{A}} x_{0,1} + \tilde{\mathbf{B}}\mathbf{u} + \tilde{\mathbf{D}}\mathbf{d} \end{aligned} \tag{4.13}$$

where the corresponding matrices can be computed by bringing all dynamic equations of the

form (4.1) together and expressing the wavelet coefficients in terms of inputs and the initial condition on the tree (note the inputs and the disturbances belong to the left nodes). For a 3 scale system (N=8) the system parameters can be computed through the following relationships:

$$\begin{bmatrix} x_{-2,1} \\ x_{-1,1} \\ x_{-1,3} \\ x_{0,1} \\ x_{0,3} \\ x_{0,5} \\ x_{0,7} \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 & -\sqrt{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 & 0 \\ -\frac{\sqrt{2}}{2} & -\frac{1}{2} & \frac{1}{2} & -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & -\frac{1}{2} & -\frac{1}{2} & -\frac{\sqrt{2}}{2} & 0 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} \begin{bmatrix} \delta x_{-3,1} \\ \delta x_{-2,1} \\ \delta x_{-2,2} \\ \delta x_{-1,1} \\ \delta x_{-1,2} \\ \delta x_{-1,3} \\ \delta x_{-1,4} \end{bmatrix} +$$

$$\begin{bmatrix} 2 \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} x_{0,1} \tag{4.14}$$

$$= \Phi \delta \hat{\mathbf{x}} + \phi x_{0,1} \tag{4.15}$$

The dynamic model on the tree is:

$$\begin{bmatrix} \delta x_{-3,1} \\ \delta x_{-2,1} \\ \delta x_{-2,2} \\ \delta x_{-1,1} \\ \delta x_{-1,2} \\ \delta x_{-1,3} \\ \delta x_{-1,4} \end{bmatrix} = \begin{bmatrix} x_{-2,1} \\ x_{-1,1} \\ x_{-1,3} \\ x_{0,1} \\ x_{0,3} \\ x_{0,5} \\ x_{0,7} \end{bmatrix} + \Psi \begin{bmatrix} u_{-2,1} \\ u_{-1,1} \\ u_{-1,3} \\ u_{0,1} \\ u_{0,3} \\ u_{0,5} \\ u_{0,7} \end{bmatrix} + \Delta \begin{bmatrix} d_{-2,1} \\ d_{-1,1} \\ d_{-1,3} \\ d_{0,1} \\ d_{0,3} \\ d_{0,5} \\ d_{0,7} \end{bmatrix} \tag{4.16}$$

where

$$= \begin{bmatrix} \frac{(I-A_{-2})}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{(I-A_{-1})}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{(I-A_{-1})}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{(I-A_0)}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{(I-A_0)}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{(I-A_0)}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{(I-A_0)}{\sqrt{2}} \end{bmatrix} \tag{4.17}$$

$$\Psi = \begin{bmatrix} \frac{-B_{-2}}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-B_{-1}}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-B_{-1}}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-B_0}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-B_0}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-B_0}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-B_0}{\sqrt{2}} \end{bmatrix} \tag{4.18}$$

$$\Delta = \begin{bmatrix} \frac{-D_{-2}}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{-D_{-1}}{\sqrt{2}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{-D_{-1}}{\sqrt{2}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-D_0}{\sqrt{2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-D_0}{\sqrt{2}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{-D_0}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-D_0}{\sqrt{2}} \end{bmatrix} \tag{4.19}$$

So using equation (4.15) we can replace the scaling coefficients:

$$\begin{aligned}
\delta\hat{\mathbf{x}} &= \hat{\mathbf{x}} + \Psi\mathbf{u} + \Delta\mathbf{d} \\
\delta\hat{\mathbf{x}} &= \Phi\delta\hat{\mathbf{x}} + \phi x_{0,1} + \Psi\mathbf{u} + \Delta\mathbf{d} \\
\delta\hat{\mathbf{x}} &= (I - \Phi)^{-1}\ \phi x_{0,1} + (I - \Phi)^{-1}\Psi\mathbf{u} + (I - \Phi)^{-1}\Delta\mathbf{d}
\end{aligned} \tag{4.20}$$

Finally we can define the parameters in equation (4.13):

$$\tilde{\mathbf{A}} = (I - \Phi)^{-1}\ \phi \tag{4.21}$$

$$\tilde{\mathbf{B}} = (I - \Phi)^{-1}\Psi \tag{4.22}$$

$$\tilde{\mathbf{D}} = (I - \Phi)^{-1}\Delta \tag{4.23}$$

The input constraints can be represented in the multiscale domain as:

$$\underline{\mathrm{U}}_{-s} \leq \mathbf{u}_{-s,k} \leq \bar{\mathrm{U}}_{-s}\ \forall k, s \in T \tag{4.24}$$

where the lower and upper bounds can be set separately for each scale as a design parameter, or they can be exact transformations of the time domain constraints. In that case

$$U_{-s} = \sqrt{2}^{s} U_0 \tag{4.25}$$

The output bounds are:

$$\underline{\mathrm{Y}}_{-s} \leq \mathbf{y}_{-s,k} \leq \bar{\mathrm{Y}}_{-s}\ \forall k, s \in T \tag{4.26}$$

Then similar to regular MPC, determining the model-plant mismatch including the measurement noise requires introduction of an observer:

$$\min_{\hat{\mathbf{w}}} \Phi_k = \hat{w}_{-s,k_s-N_s-1} P^{-1} \hat{w}_{-s,k_s-N_s-1} + \sum_{j_s=k_s-N_s}^{k_s-1} \left(\hat{w}_{-s,j}^T Q_w^{-1} \hat{w}_{-s,j}\right) + \sum_{j=k_s-N_s}^{k_s} \left(\hat{v}_{-s,j}^T R_v^{-1} \hat{v}_{-s,j}\right) \tag{4.27}$$

subject to:

$$\begin{aligned} \hat{x}_{-s,j+1|k_s} &= A\hat{x}_{-s,j|k_s} + Bu_{-s,j} + \hat{w}_{-s,j} \\ y_{-s,j_s} &= C\hat{x}_{-s,j|k_s} + \hat{v}_{-s,j} \end{aligned}$$

with initial estimate

$$\hat{x}_{-s,k_s-N_s|k_s} = \hat{x}_{-s,k_s-N_s|k_s-N_s-1} + \hat{w}_{-s,k_s-N_s-1} \tag{4.28}$$

and the constraints

$$\begin{aligned} \hat{w}_{\min,-s} &\leq \hat{w}_{-s,j} \leq \hat{w}_{\max,-s} \\ \hat{x}_{\min,-s} &\leq \hat{x}_{-s,j|k_s} \leq \hat{x}_{\max,-s} \end{aligned}$$

In summary the MSMPC with state estimation can be stated as:

$$\min_{\delta\hat{\mathbf{x}}} \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)^T \left(I + H^T H\right) \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)$$

subject to

$$\begin{aligned} \delta\hat{\mathbf{x}} &= \tilde{\mathbf{A}}x_{0,1} + \tilde{\mathbf{B}}\mathbf{u} + \tilde{\mathbf{D}}\mathbf{d} \\ \mathbf{y} &= \tilde{\mathbf{C}}\left(\Phi\delta\hat{\mathbf{x}} + \phi x_{0,1}\right) \end{aligned}$$

$$\underline{\mathbf{U}}_{-s} \leq \mathbf{u}_{-s,k} \leq \bar{\mathbf{U}}_{-s} \ \forall k, s \in T$$

$$\underline{\mathbf{Y}}_{-s} \leq y_{-s,k} \leq \bar{\mathbf{Y}}_{-s} \ \forall k, s \in T$$

with

$$x_{0,k=1} = x_0$$

where x_0 and d_k are found through

$$\min_{\hat{\mathbf{w}}} \Phi_k = \hat{w}_{-s,k_s-N_s-1} P^{-1} \hat{w}_{-s,k_s-N_s-1} + \sum_{j_s=k_s-N_s}^{k_s-1} \left(\hat{w}_{-s,j}^T Q_w^{-1} \hat{w}_{-s,j}\right) + \sum_{j=k_s-N_s}^{k_s} \left(\hat{v}_{-s,j}^T R_v^{-1} \hat{v}_{-s,j}\right)$$

subject to:

$$\begin{aligned} \hat{x}_{-s,j+1|k_s} &= A\hat{x}_{-s,j|k_s} + Bu_{-s,j} + \hat{w}_{-s,j} \\ y_{-s,j_s} &= C\hat{x}_{-s,j|k_s} + \hat{v}_{-s,j} \end{aligned}$$

with initial estimate

$$\hat{x}_{-s,k_s-N_s|k_s} = \hat{x}_{-s,k_s-N_s|k_s-N_s-1} + \hat{w}_{-s,k_s-N_s-1}$$

$$\begin{aligned} \hat{w}_{\min,-s} &\leq \hat{w}_{-s,j} \leq \hat{w}_{\max,-s} \\ \hat{x}_{\min,-s} &\leq \hat{x}_{-s,j|k_s} \leq \hat{x}_{\max,-s} \end{aligned}$$

and

$$\begin{aligned} x_0 &= \hat{x}_{0,1|k=0} + \hat{w}_{0,1} \\ d_{-s,1} &= \hat{w}_{-s,1} \ \forall s \in T \end{aligned}$$

In this formulation of the observer the index 1 refers to the last points at each scale, not to the first points.

Multiscale Model Predictive Control (MSMPC) and classical Model Predictive Control (MPC) share many of theoretical properties. Since MSMPC is derived directly from MPC formulation it can be tuned in such a way that it is equivalent to an MPC in time domain. This equivalency allows the adoption of almost all system theoretic characteristics of MPC.

There are two major areas where the properties of MSMPC should be investigated:

1. System theoretic aspects such as stability, feasibility, performance and robustness
2. MPC specific aspects like horizon length, constraints, blocking, condensing, reference path

As mentioned before since MSMPC shares many properties with MPC most of the theorems developed for classical MPC apply to MSMPC, too.

4.3 System Theoretic Aspects

System theoretic aspects of MSMPC deal with general behavior of the controlled system under the effect of the model predictive controller.

4.3.1 Nominal Stability

Stability in the context of MPC means that one can bring the states or outputs from a given initial condition to a predefined vicinity of target values in a given horizon length. As the horizon length gets larger and approaches infinity the deviation of states or outputs from the target values should approach to zero, too, i.e. asymptotic stability.

$$\begin{aligned} ||R_N - x_N||_2 &\leq \varepsilon_N \text{ and } \varepsilon_N \geq 0 \\ \varepsilon_N &\rightarrow 0 \text{ as } N \rightarrow \infty \end{aligned}$$

Since stability of multiscale systems is equivalent to time domain systems, all the discussion about stability of MPC in Chapter 2 is valid here, too.

4.3.2 Open-Loop Stability

Stability of the open loop system does not guarantee closed loop stability, but it is required for the closed loop to be stable. Open-loop stability is determined by the stability characteristics of the underlying process and controller. Since MSMPC uses the same state space representation and the same set of output and input constraints as the regular MPC all of the stability properties apply to MSMPC, too, with the exception of feasibility of the control horizon. The regular MPC has the horizon length as a tuning parameter and a bad choice of this horizon can create an infeasible problem in terms of output constraints (input constraints are always feasible)(see section 2.5.4) . MSMPC chooses its horizon length at each step emulating an infinite horizon algorithm such that the output constraints are always feasible (of course if at least an infinitely long control horizon is feasible). With the variable horizon and reference path

tracking settings MSMPC is pratically an infinite horizon formulation. The objective function consists of two parts:

$$\Phi_k = \sum_{1}^{N} (R_i - x_i)^T (R_i - x_i) + \sum_{N+1}^{\infty} (R_i - x_i)^T (R_i - x_i)$$

but we know that N is chosen such that from N+1 on

$$(R_i - x_i) = 0 \ \forall i > N+1 \tag{4.29}$$

so

$$\Phi_k = \sum_{1}^{N} (R_i - x_i)^T (R_i - x_i)$$

This formulation does not require the artificial infinite prediction horizon and finite control vector parametrization, but it is similar to the formulation (**Lee**, 1995) where after the control horizon the following assumption is made for the inputs

$$u_i = K x_i \ \forall i > N+1$$

This is very similar to the result we have in equation (4.29).

$$\begin{aligned} (R_i - x_i) &= 0 \\ R_i &= A R_{i-1} + B u_{i-1} \\ u_{i-1} &= B^{-1} [R_i - A R_{i-1}] \ \forall i > N+1 \end{aligned} \tag{4.30}$$

4.3.3 Closed-Loop Stability

Closed loop properties of MSMPC are also very similar to regular MPC. Asymptotic stability can be shown by Lyapunov type arguments. MSMPC does not need any terminal constraints to guarantee stability, since the algorithm expands the horizon as much as needed to reach the reference path at the end. From a point of view this can be seen as a contraction method. The reference path is a filter which defines how much the open loop states should approach the

real target values at the end of the horizon so that the closed loop problem is asymptotically stable (the end of the horizon is the point N, but one knows there exists a solution for inputs which brings the states asymptotically to real target values and this solution can be computed through solution of an algebraic set of equations (4.30)). For unstable dynamic systems, if there exist a solution (i.e. the system is stabilizable) then MSMPC will expand its horizon length as much as required to bring the unstable modes to the target values.

4.3.4 Robust Stability

For MSMPC robust stability or in other words the ability of the system to handle modeling errors or parametric changes can be treated the same way it is treated for MPC. Using a different computational structure for plant-model mismatch over different scales can only improve the robustness since the open loop optimization step has more information about the system across different scales or frequency bands.

4.3.5 Performance

One can select tuning parameters of MSMPC such that it exactly reproduces the same results as MPC, thus it has the same performance, but the flexibility of the formulation and computation of MSMPC enables the nominal controller to perform better than MPC. In addition under certain disturbance loads and plant-model mismatch cases MSMPC again performs much better than MPC. In the next Chapter, case studies will demonstrate these claims.

4.4 Tuning Parameters

4.4.1 Horizon Length

Values of tuning parameters of MPC are difficult to determine. Especially determination of the length of the *control and prediction horizons* are very tricky. A wrong choice can render the open loop problem for an unstable system to become infeasible and thus the closed loop response to become unstable . MSMPC has the unique ability to choose its horizon length. If the reference path can be reached fast the horizon length gets short and under certain circumstances the MSMPC can become a simple proportional controller (horizon length 1). If reference path

cannot be reached fast then the horizon length gets expanded until the reference path is reached. For computational reasons one can define an upper bound for the horizon length and then add a contraction type of inequality constraint to satisfy asymptotic stability of the closed loop.

4.4.2 Constraints

If a large disturbance hits the system the problem can become infeasible. Inverse response systems can also make the QP infeasible at the beginning of the open loop horizon. Regular MPC handles that problem either by neglecting output constraints for a short time period at the beginning of the horizon or by converting the hard constraints (output constraints) to soft constraints and then penalizing the deviation from the constraints. At that point prioritizing of the constraints can be used as a design methodology. MSMPC needs also such a tool since not all infeasibilities can be handled by increasing the horizon length. Rawlings and Muske [6] and Rawlings [5] developed methodologies to incorporate the constraint relaxation and cost function modifications. Those techniques can be used directly in MSMPC too. The modified cost function would look like

$$\min_{\delta\hat{\mathbf{x}}} \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)^T \left(I + H^T H\right) \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) + \varepsilon^T Q \varepsilon$$

and additional "bandwidth" constraints would look like

$$\begin{aligned} \underline{Y}_{-s} - \varepsilon_{-s,k} &\leq y_{-s,k} \leq \bar{Y}_{-s} + \varepsilon_{-s,k} \ \forall k, s \in T \\ \underline{E}_{-s} &\leq \varepsilon_{-s,k} \leq \overline{E}_{-s} \ \forall k, s \in T \end{aligned}$$

The choice of E would determine which constraints are to be relaxed (and by how much) and which ones are kept to be as hard constraints.

4.4.3 Weights in the Objective Function

Determination of the weight in the objective function is really a design problem. A particular choice of weights reflect a desired solution with certain characteristics. Large weights for the inputs or change of inputs (move suppressant) cause a smoother input profile with better robustness characteristics but inferior performance. Large weights for the states enforce the

system to reject deviations from the set point, but the resulting input profile could be frantic and model uncertainty and/or external disturbances can make the problem easily unstable. There is no solid framework yet available to determine those weights.

MSMPC objective function is equivalent to MPC objective function but the regrouped variables (averages and differences of original variables) give us a great opportunity to define the weights such that they correspond to frequency response characteristics of the desired controller. Since the new variables represent different scales (frequency bands) one can use loop shaping ideas from frequency domain to design a nominal controller. For unconstrained problems this nominal controller can indeed be realized (frequency domain LQR formulation). For constrained problems the QP will decide for the best possible solution where the designed weights will define the target controller.

Frequency Domain Requirements

First we will introduce some basic definitions how to design controllers with frequency domain specifications and then we will introduce techniques to apply these requirements on time domain problems:

1. Command Following
2. Disturbance Rejection
3. Insensitivity to Measurement Noise

Singular values are one of the most important tools in frequency domain controller design, because the bounds on norms can be expressed using them:

In a closed loop system the feedback error can be described as:

$$e(s) = (I + GK)^{-1} r(s) - (I + GK)^{-1} d(s) + (I + GK)^{-1} GK\eta(s) \tag{4.31}$$

and if there are no disturbances and measurement noise terms (4.31) becomes:

$$e(s) = (I + GK)^{-1} r(s) \tag{4.32}$$

$$= S(s)r(s) \tag{4.33}$$

and if r is assumed to be a sinusoidal signal

$$r(t) = \underline{r}e^{iwt} \tag{4.34}$$

$$e = S(jw)r \tag{4.35}$$

$$||e||_2 \leq \sigma_{\max}[S(jw)]\,||r||_2 \tag{4.36}$$

if is the range of frequencies for the command signal then $\sigma_{\max}[S(jw)] \ll 1$ for all w$\in$ for good command control. Furthermore one can band the error term by the possible minimum and maximum values:

$$\sigma_{\min}[S(jw)] \leq ||e||_2 \leq \sigma_{\max}[S(jw)] \tag{4.37}$$

Using some singular value properties:

$$\sigma_{\min}(A) - 1 \leq \sigma_{\min}(I + A) \leq \sigma_{\min}(A) + 1 \tag{4.38}$$

$$\sigma_{\max}(A^{-1}) = \frac{1}{\sigma_{\min}(A)} \tag{4.39}$$

Then

$$\sigma_{\max}[S(jw)] \ll 1 \text{ (small sensitivity)} \tag{4.40}$$

$$\sigma_{\max}\left[(I + GK)^{-1}\right] \ll 1 \tag{4.41}$$

$$\sigma_{\max}\left[(I + GK)^{-1}\right] = \frac{1}{\sigma_{\min}[(I + GK)]} \tag{4.42}$$

$$\sigma_{\min}[(I + GK)] \gg 1 \tag{4.43}$$

$$\sigma_{\min}[(I + GK)] < \sigma_{\min}[GK] + 1 \tag{4.44}$$

$$\sigma_{\min}[G(jw)K(jw)] \gg 1 \ \forall w \in \quad \text{(large loop gain)} \tag{4.45}$$

The same arguments hold for disturbance rejection too.

Remark 5 *With these specifications the sensitivity to measurement noise is set to the maximum possible value because the complementary sensitivity function becomes I: the result is a flat closed-loop response* $\sigma_{\min}[C(jw)] \approx \sigma_{\max}[C(jw)] \approx 1$

Remark 6 *Good command following and disturbance rejection conflicts with low sensitivity to measurement noise.*

Remark 7 *Cannot do good command following and disturbance rejection with noisy sensors that make low frequency errors (drifts, bias etc.)*

One can modify the plant by adding some filters in front of the inputs and measurements and thereby emphasizing just certain frequency ranges.

Closed Loop Control Requirements

1. $S(jw)$ (sensitivity) should be kept small indicating

   $\sigma_{\max}\left[(I+GK)^{-1}\right] \ll 1$

2. $T(jw)$ (closed loop transfer function) should be kept small indicating

   $\sigma_{\max}\left[I-(I+GK)^{-1}\right] \ll 1$

3. Tracking of a reference path indicating

   $\sigma_{\max}\left[I-(I+GK)^{-1}\right] \approx \sigma_{\min}\left[I-(I+GK)^{-1}\right] \approx 1$

4. Minimize control energy indicating keeping $\sigma_{\max}[K]$ small.

Open Loop Control Requirements

1. $\sigma_{\min}[GK]$ large (sensitivity, tacking of reference path)

2. $\sigma_{\max}[GK]$ small (noise propagation)

3. $\sigma_{\max}[K]$ small (minimize control energy)

Now these requirements should be translated into the weights in the LQR (LQG) problem so that they can be applied in time domain. There is not any method to do it in time domain systematically for all cases. One can still use a trial and error based tuning of weights to satisfy certain robustness and performance characteristics by changing the Q and R matrices in the cost function used in LQR design.

Robustness Enhancement via Frequency - Weighted Cost Function [1] The suppression of certain vibration modes may be crucial. It is possible to extend the infinite time LQR optimization theory to frequency sensitive cost functions by applying Parseval's Theorem

$$\int_0^\infty |f(t)|^2\,dt = \frac{1}{2\pi}\int_{-\infty}^\infty |F(j\omega)|^2\,d\omega \tag{4.46}$$

The cost function can also be written the same way

$$\begin{aligned}\int_0^\infty |\mathrm{f}(t)|^2\,dt &= \int_0^\infty \{x^TQx + u^TRu\}\,dt \\ &= \frac{1}{2\pi}\int_{-\infty}^\infty \{x^*(j\omega)Q(j\omega)x(j\omega) + u^*(j\omega)R(j\omega)u(j\omega)\}\,d\omega\end{aligned} \tag{4.47}$$

And proper selection of $Q(j\omega)$ and $R(j\omega)$ provides the ability to weigh different vibration modes selectively.

Let us assume that the system is:

$$\begin{aligned}\dot{x} &= Ax + Bu + w \\ y &= Hx + v\end{aligned}$$

and the objective function is:

$$J = \int_0^\infty \left(x^TQx + u^TRu\right)dt$$

Using Parseval's theorem this objective function becomes:

$$J = \frac{1}{2\pi}\int_{-\infty}^{+\infty} \left(x^*\left(j\omega\right)Qx\left(j\omega\right) + u^*\left(j\omega\right)Ru\left(j\omega\right)\right)d\omega \tag{4.48}$$

Here the weighting factors are assumed to be constant, but if we let them vary with frequency them we can find a desirable set of weighting factors which would force the solution of the optimization to satisfy certain frequency response characteristics. If we further assume that the

weights can be made of the form:

$$Q(j\omega) = P_1^*(j\omega) P_1(j\omega) \tag{4.49}$$

$$R(j\omega) = P_2^*(j\omega) P_2(j\omega) \tag{4.50}$$

then we can transform the state equations as:

$$x^1 = P_1 x \tag{4.51}$$

$$u^1 = P_2 u \tag{4.52}$$

Then the objective function becomes:

$$J = \frac{1}{2\pi} \int_{-\infty}^{+\infty} \left(x^{1*}(j\omega) x^1(j\omega) + u^{1*}(j\omega) u^1(j\omega) \right) d\omega$$

and this objective function can be transformed into the time domain objective:

$$J = \int_0^{\infty} \left(x^{1T} x^1 + u^{1T} u^1 \right) dt$$

Now we need to find out the dynamics of the filters P_1 and P_2 in the time domain [3]:

$$P_1(s) = C_1 (sI - A_1)^{-1} B_1 + D_1 \tag{4.53}$$

$$P_2(s) = C_2 (sI - A_2)^{-1} B_2 + D_2 \tag{4.54}$$

Assume these are the realizations of the filter dynamics. Then the augmented system becomes:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ \dot{z}^1 &= B_1 C x + A_1 z^1 \\ \dot{z}^2 &= A_2 z^2 + B_2 u \\ x^1 &= D_1 C x + C_1 z^1 \\ u^1 &= C_2 z^2 + D_2 u \end{aligned}$$

or in summary:

$$\begin{bmatrix} \dot{x} \\ \dot{z}^1 \\ \dot{z}^2 \end{bmatrix} = \tilde{A} \begin{bmatrix} x \\ z^1 \\ z^2 \end{bmatrix} + \tilde{B}u$$

$$\begin{bmatrix} x^1 \\ u^1 \end{bmatrix} = \tilde{C} \begin{bmatrix} x \\ z^1 \\ z^2 \end{bmatrix} + \tilde{D}u$$

and the objective function becomes:

$$\begin{aligned} J &= \int_0^\infty \left(\begin{bmatrix} x^1 \\ u^1 \end{bmatrix}^T \begin{bmatrix} x^1 \\ u^1 \end{bmatrix} \right) dt \\ J &= \int_0^\infty \left(X^T \tilde{C}^T \tilde{C} X + 2X^T \tilde{C} \tilde{D} u + u^T \tilde{D}^T \tilde{D} u \right) dt \end{aligned} \tag{4.55}$$

where

$$X = \begin{bmatrix} x \\ z^1 \\ z^2 \end{bmatrix}$$

So the resulting controller is *a dynamic compensator.*

Multiscale Loop Shaping with Hard Constraints in Time Domain The major difficulty in designing an optimal controller in frequency domain, with hard constraints in time domain, is that frequency domain representation of such a controller does not exist. The hard constraints render the controller nonlinear. A way around this difficulty is designing the controllers frequency response characteristics in frequency domain and then transforming those into time domain as weights in the objective function. Enforcing the time domain hard constraints upon the optimization problem will result in a trade-off: The hard constraints will be satisfied under any condition to comply with the requirement of feasibility, but frequency response characteristics will only be followed if there is enough degrees of freedom. So basically the system

will follow the frequency domain design settings closely if the optimization problem is interior feasible, but these settings will not be realized as desired if some constraints are activated.

Derivation of MS-Loop Shaping Before going into the formulation there are some important points to be mentioned:

Remark 8 *In the multiscale representation there is a finite number of frequency bands or scales (discrete frequency bands).*

Remark 9 *If we choose to use the weights in the optimization function as means of designing the frequency response characteristics of the system then these specifications will appear as soft constraints rather than hard bounds and under limited degrees of freedom (active time domain hard bounds) those criteria will not be satisfied.*

Remark 10 *If we have a longer horizon length in the MPC formulation then we have access to slower frequencies.*

Remark 11 *High scales represent lower frequencies and penalizing them more than the lower frequencies will result in rapid transients and steady state offsets if the horizon length is not long enough. (Top δx can not be zero if there are active constraints, and forcing it to be zero would leave an offset)*

Remark 12 *Lower scales represent faster frequencies (fastest frequency, ω_0, being at the time line or 0th level or physical level or at the closure and $\omega_0 = \frac{1}{T_s}$, where T_s is the smallest sampling period). Penalizing them will result in slower transients and steady state offsets if the horizon is not long enough. The response will be smooth but slow.*

Problem Formulation There are two different ways to formulate the problem. The one similar to the LQR is to define frequency response characteristics as weights and thus as soft constraints. The other formulation is to express the frequency response characteristics as hard constraints. The second formulation is more powerful and the result is guaranteed to satisfy the requirements, but the feasibility of the problem formulation becomes an issue, which can be addressed by solving an LP problem first to show that the feasible space is not an empty set.

MS-MPC Formulation

$$\min_{\delta\hat{\mathbf{x}}} \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)^T \left(I + H^T H\right) \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) \tag{4.56}$$

subject to

$$\begin{aligned} \delta x_{-s-1,\frac{k}{2}} &= \frac{\left(I - (A_{-s})_{n\times n}\right)}{\sqrt{2}} x_{-s,k} - \frac{(B_{-s})_{n\times m}}{\sqrt{2}} u_{-s,k} - \frac{(D_{-s})}{\sqrt{2}} d_{-s,k} \\ y_{-s,k} &= (C_{-s})_{m\times n}\, x_{-s,k} \end{aligned}$$

$$\underline{\mathrm{U}}_{-s} \leq \mathbf{u}_{-s,k} \leq \bar{\mathrm{U}}_{-s} \ \forall k, s \in T \tag{4.57}$$

$$\underline{\mathrm{Y}}_{-s} \leq y_{-s,k} \leq \bar{\mathrm{Y}}_{-s} \ \forall k, s \in T \tag{4.58}$$

with

$$x_{0,k=0} = x_{0,0} \tag{4.59}$$

Soft Constraint Formulation

$$\min_{\delta\hat{\mathbf{x}}} \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)^T \left(Q + H^T P H\right) \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) \tag{4.60}$$

where the weighting matrices Q and P are the design parameters. The simplest case is that Q is a diagonal, positive semidefinite matrix with a penalty term associated with every scale, and P a diagonal, positive semidefinite matrix penalizing the deviation from the reference path at

the highest scale (top node) i.e. :

$$Q = \begin{bmatrix} \underline{q_3} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \underline{q_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \underline{q_2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \underline{q_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \underline{q_1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \underline{q_1} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \underline{q_1} \end{bmatrix}$$

$$P = [p_1]$$

A $q_1 > q_2 > q_3$ strategy would result in slow, smooth response where as a $q_1 < q_2 < q_3$ strategy would result in a fast transient. A large p_1 would force the system to distribute the total deviation (lack of performance, total error) such that the algebraic average of it is minimized. Since the algebraic average is minimized the resulting response can be very noisy with large overshoots

Although through this representation we have access to different frequency bands (scales) of the system we cannot specify the loop shape since we don't have an explicit frequency domain transfer function between the states (outputs) and the inputs and disturbances. Even if we had one the controller gain is not static, but it is changing at each step of the open loop prediction and closed loop implementation (the first input computed through the open loop prediction). This can be easily shown for LQR systems where the gain is computed through the dynamic Ricatti equation. (In MPC the hard bounds prevent formulation of a Ricatti equation, before each implementation one can compute an artificial gain though) An algorithm can be devised to check the performance of the controller and modify it if it does not satisfy the requirements.

Hard Constraint Case The hard constraint case requires a problem formulation which is more conservative than the other but in return it guarantees satisfaction of all design criteria

both in time and frequency.

$$\min_{\delta\hat{\mathbf{x}}} \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)^T \left(I + H^T H\right) \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)$$

subject to

$$\begin{aligned} \delta x_{-s-1,\frac{k}{2}} &= \frac{\left(I - (A_{-s})_{n\times n}\right)}{\sqrt{2}} x_{-s,k} - \frac{(B_{-s})_{n\times m}}{\sqrt{2}} u_{-s,k} - \frac{(D_{-s})}{\sqrt{2}} d_{-s,k} \\ y_{-s,k} &= (C_{-s})_{m\times n}\, x_{-s,k} \end{aligned}$$

$$\underline{\mathrm{U}}_{-s} \leq \mathbf{u}_{-s,k} \leq \bar{\mathrm{U}}_{-s} \ \forall k, s \in T$$

$$\underline{\mathrm{Y}}_{-s} \leq y_{-s,k} \leq \bar{\mathrm{Y}}_{-s} \ \forall k, s \in T$$

$$\left| T \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) \right| \leq \gamma \tag{4.61}$$

with

$$x_{0,k=0} = x_{0,0}$$

The extension is the equation (4.61). That additional bound provides a means of controlling the shape of the response, or better the shape of the deviation. The matrix T selects the scales and γ determines the maximum allowed deviation. Small γ at lower scales indicates a smooth transition and slow response and reduces sensitivity to measurement noise and disturbances (including both external and plant-model mismatch). γ being equal to zero at a given scale corresponds to a kind of condensing (condensing the deviation but neither the states nor the inputs). Small γ at higher scales indicates a fast transient and a low tolerance to steady state offsets. (Notice that one can include a γ_T so that one can explicitly bound the deviation; although it seems to be a desirable design variable it can cause the problem to be infeasible within the hard bounds on inputs and outputs)

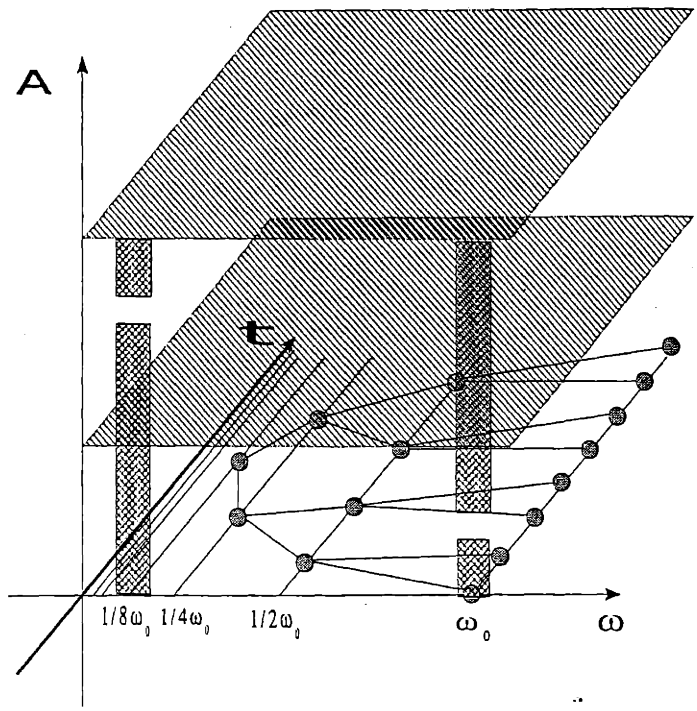


Figure 4-1: A time / frequency representation with hard bounds in both domains

4.4.4 Reference Path

Reference path is also an important factor for robustness and performance. Ideally the reference path should reflect the inverse of the system times a filter, in other words a nominal solution.

First let us show that there exists a reference path trajectory such that the classical formulation can be exactly replicated:

Classical formulation has the objective function

$$\Phi_C = \mathbf{x}^T Q \mathbf{x} + \mathbf{u}^T S \mathbf{u}$$

And incorporation of reference path renders the following objective function:

$$\Phi_R = (\mathbf{R} - \mathbf{x})^T (\mathbf{R} - \mathbf{x})$$

If we use the state equation (over the length of control horizon)

$$\mathbf{x} = Ax_0 + B\mathbf{u}$$

and incorporate it into the objective functions, the classical becomes

$$\Phi_C = \mathbf{u}^T(B^TQB + S)\mathbf{u} + 2x_0^TA^TQB\mathbf{u} + x_0^TA^TAx_0$$

and the rendered objective function becomes:

$$\Phi_R = \mathbf{u}^TB^TB\mathbf{u} + 2(\mathbf{R} - Ax_0)^TB\mathbf{u} + (\mathbf{R} - Ax_0)^T(\mathbf{R} - Ax_0)$$

Now further assume that there are no constraints (or they are not active) and B is invertible. Then at the optimal point both inputs should be same (Q, S are symmetrical positive definite matrices) :

$$\begin{aligned}
\frac{\partial \Phi_C}{\partial \mathbf{u}} &= 0 \\
\frac{\partial \Phi_R}{\partial \mathbf{u}} &= 0 \\
\frac{\partial \Phi_C}{\partial \mathbf{u}} &= 2(B^TQB + S)\mathbf{u} + 2B^TQAx_0 = 0 \\
\frac{\partial \Phi_R}{\partial \mathbf{u}} &= 2B^TB\mathbf{u} + 2B^T(\mathbf{R} - Ax_0) = 0
\end{aligned}$$

From there

$$\begin{aligned}
\mathbf{u}_C &= -(B^TQB + S)^{-1}B^TQAx_0 \\
&= -(QB + B^{-T}S)^{-1}QAx_0 \\
\mathbf{u}_R &= \left(B^TB\right)^{-1}B^T(\mathbf{R} - Ax_0) \\
&= B^{-1}(\mathbf{R} - Ax_0)
\end{aligned}$$

and both u's should be same, so:

$$\begin{aligned} B^{-1}(\mathbf{R}-Ax_0) &= -(QB+B^{-T}S)^{-1}Ax_0 \\ \mathbf{R} &= \left[I-(Q+B^{-T}SB^{-1})^{-1}Q\right]Ax_0 \end{aligned}$$

So for every pair $\{Q,S\}$ there is an $\mathbf{R}$ giving the same nominal solution (input profile). Furthermore we are not restricted to Q and S but since we have $\mathbf{R}$ at multiple scales we have the option of designing such an $\mathbf{R}$ that penalizes changes at different scales differently thus we can shape the frequency behavior of the resulting system (loop shaping). As one can see easily the reference path can create the same stabilizing effects as S, namely a correct choice of $\mathbf{R}$ (or S) can shift the right hand side zeros to the left half plane and thus eliminate unstable modes. For inverse response systems a well designed reference path can eliminate output constraint violations by forcing the system into the opposite direction at the beginning of the horizon.

4.4.5 Blocking and Condensing

Blocking and condensing are tools to reduce the complexity of an optimization algorithm involving a large number of inputs and outputs. Blocking (B) is the assumption that inputs do not change for a specified number of time steps, thus a reduction in the number of free variables, and condensing (C) is the assumption that the output variables do not change for the specified range of time steps, thus reducing the number of inequality constraint on the outputs if they have been posted.

B&C are known to reduce the complexity and in the case of MPC and other types of optimal control, they are also known to enhance robustness at the cost of performance. Reducing the number of free variables makes the resulting policy less flexible, and this in effect is a disadvantage for the performance characteristics of the system. On the other hand, this reduced flexibility or sensitivity boosts the robustness of the system: an unavoidable trade-off.

B&C can easily be incorporated into MSMPC by the way of equality constraints, the same way it is incorporated into the classical MPC. A similar strategy (a special case of blocking) can be used to handle *multirate systems* (Karslıgil,1999 [2])

Formulation

B&C can be performed by manipulating the system equations:

$$x_{k+1} = Ax_k + Bu_k \tag{4.62}$$

$$y_k = Cx_k \tag{4.63}$$

For a horizon length of N one can bring these equations in a compact form as follows:

$$\begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} A \\ \vdots \\ A^N \end{pmatrix} x_0 + \begin{pmatrix} B & \cdots & 0 \\ \vdots & \vdots & \vdots \\ A^{N-1}B & \cdots & B \end{pmatrix} \begin{pmatrix} u_0 \\ \vdots \\ u_{N-1} \end{pmatrix} + \begin{pmatrix} I & \cdots & 0 \\ \vdots & \vdots & \vdots \\ A^{N-1} & \cdots & I \end{pmatrix} \begin{pmatrix} d_0 \\ \vdots \\ d_{N-1} \end{pmatrix}$$

$$\mathbf{x} = \tilde{\mathbf{A}}x_0 + \tilde{\mathbf{B}}\mathbf{u} + \tilde{\mathbf{D}}\mathbf{d}$$

$$\mathbf{y} = \tilde{\mathbf{C}}\mathbf{x}$$

$$\mathbf{y} = \tilde{\mathbf{C}}\tilde{\mathbf{A}}x_0 + \tilde{\mathbf{C}}\tilde{\mathbf{B}}\mathbf{u} + \tilde{\mathbf{C}}\tilde{\mathbf{D}}\mathbf{d}$$

where

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}, \mathbf{u} = \begin{pmatrix} u_0 \\ \vdots \\ u_{N-1} \end{pmatrix}, \mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \tag{4.64}$$

Then the blocking can be done by:

$$\mathbf{u} = P_B\tilde{\mathbf{u}} \tag{4.65}$$

and the condensing by:

$$y = P_C\tilde{y} \tag{4.66}$$

These blocking and condensing matrices can be created by repeating rows and deleting columns form the identity matrix. As an example take the following blocking strategy:

$$\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \end{bmatrix} \longrightarrow \tilde{\mathbf{u}} = \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \\ \tilde{u}_5 \end{bmatrix} \longrightarrow \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \\ \tilde{u}_5 \\ \tilde{u}_5 \end{bmatrix}$$

This can be achieved by

$$\begin{aligned} \mathbf{u} &= \mathbf{P}_B \tilde{\mathbf{u}} \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \\ \tilde{u}_3 \\ \tilde{u}_4 \\ \tilde{u}_5 \end{bmatrix} \end{aligned}$$

So the blocked and condensed state and output equations are then:

$$\begin{aligned} \mathbf{x} &= \tilde{\mathbf{A}} x_0 + \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}} + \tilde{\mathbf{D}} \mathbf{d} \\ P_C \tilde{\mathbf{y}} &= P_C \tilde{\mathbf{C}} \tilde{\mathbf{A}} x_0 + P_C \tilde{\mathbf{C}} \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}} + P_C \tilde{\mathbf{C}} \tilde{\mathbf{D}} \mathbf{d} \end{aligned}$$

If we write the objective function for the tracking problem

$$\min_u (\mathbf{R} - \mathbf{x})^T (\mathbf{R} - \mathbf{x}) \tag{4.67}$$

where

$$\mathbf{R} = \begin{pmatrix} R_1 \\ \vdots \\ R_N \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix}$$

The input constraints are:

$$\underline{U} \leq u_k \leq \overline{U}, \ \forall \ k = 0..N-1 \tag{4.68}$$

The output constraints are:

$$\underline{Y} \leq y_k \leq \overline{Y}, \ \forall \ k = 1..N \tag{4.69}$$

Now if we substitute the state equation into the objective function then the minimization problem becomes

$$\min_u \left(\mathbf{R} - \tilde{\mathbf{A}}x_0 - \tilde{\mathbf{B}}\mathbf{u} - \tilde{\mathbf{D}}\mathbf{d}\right)^T \left(\mathbf{R} - \tilde{\mathbf{A}}x_0 - \tilde{\mathbf{B}}\mathbf{u} - \tilde{\mathbf{D}}\mathbf{d}\right)$$

and renaming

$$\tilde{R} = R - \tilde{\mathbf{A}}x_0 - \tilde{\mathbf{D}}\mathbf{d}$$

the objective function simplifies to:

$$\begin{aligned} & \min_u \left(\tilde{\mathbf{R}} - \tilde{\mathbf{B}}\mathbf{u}\right)^T \left(\tilde{\mathbf{R}} - \tilde{\mathbf{B}}\mathbf{u}\right) \\ \iff \quad & \min_u \frac{1}{2}\mathbf{u}^T\tilde{\mathbf{B}}^T\tilde{\mathbf{B}}\mathbf{u} - \tilde{\mathbf{R}}\tilde{\mathbf{B}}\mathbf{u} \end{aligned}$$

subject to

$$\begin{aligned} \underline{\mathbf{U}} \ &\leq \ \mathbf{u} \leq \overline{\mathbf{U}} \\ \underline{\mathbf{Y}} \ &\leq \ \mathbf{y} \leq \overline{\mathbf{Y}} \end{aligned}$$

Now analyze the number of variables:

Inputs: $\mathbf{u}_{Nm\times 1}$

Outputs: $\mathbf{y}_{Np\times 1}$

Input constraints: Converting to the standard form :

$$\begin{bmatrix} \mathbf{u} \\ -\mathbf{u} \end{bmatrix} \leq \begin{bmatrix} \overline{\mathbf{U}} \\ -\underline{\mathbf{U}} \end{bmatrix}$$

$$\begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} \mathbf{u} \leq \begin{bmatrix} \overline{\mathbf{U}} \\ -\underline{\mathbf{U}} \end{bmatrix}$$

$$V\mathbf{u} \leq \mathbf{v}$$

and from there $V_{2Nm\times 2*Nm}$ and $v_{2Nm\times 1}$

Output constraints: Converting to the standard form :

$$\begin{bmatrix} \mathbf{y} \\ -\mathbf{y} \end{bmatrix} \leq \begin{bmatrix} \overline{\mathbf{Y}} \\ -\underline{\mathbf{Y}} \end{bmatrix}$$

$$\begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix} \mathbf{y} \leq \begin{bmatrix} \overline{\mathbf{Y}} \\ -\underline{\mathbf{Y}} \end{bmatrix}$$

$$W\mathbf{y} \leq \mathbf{w}$$

and from there $W_{2Np\times 2Np}$ and $w_{2Np\times 1}$

So total numbers can be summarized as follows:

Inputs	Nm
Outputs	Np
Constraints	$2\,(Np + Nm)$

If we apply blocking and condensing to this system and let us say: we reduce the number of inputs from Nm to (N-b)m and outputs from Np to (N-c)p. The problem then can be redefined

as:

$$\begin{aligned} & \min_u \left(\tilde{\mathbf{R}} - \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}}\right)^T \left(\tilde{\mathbf{R}} - \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}}\right) \\ \Longleftrightarrow \quad & \min_u \frac{1}{2} \tilde{\mathbf{u}}^T P_B^T \tilde{\mathbf{B}}^T \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}} - \tilde{\mathbf{R}} \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}} \end{aligned} \tag{4.70}$$

subject to

$$\begin{aligned} P_B \underline{\mathbf{U}} \quad &\leq \quad P_B \tilde{\mathbf{u}} \leq P_B \overline{\mathbf{U}} \\ P_C \underline{\mathbf{Y}} \quad &\leq \quad P_C \tilde{\mathbf{y}} \leq P_C \overline{\mathbf{Y}} \end{aligned} \tag{4.71}$$

and introducing the output equation, the output constraints become:

$$\begin{aligned} P_C \underline{\mathbf{Y}} \quad &\leq \quad \tilde{\mathbf{C}} \tilde{\mathbf{A}} x_0 + \tilde{\mathbf{C}} \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}} + \tilde{\mathbf{C}} \tilde{\mathbf{D}} \mathbf{d} \leq P_C \overline{\mathbf{Y}} \\ P_C \left(\underline{\mathbf{Y}} - \tilde{\mathbf{C}} \tilde{\mathbf{A}} x_0 - \tilde{\mathbf{C}} \tilde{\mathbf{D}} \mathbf{d}\right) \quad &\leq \quad P_C \tilde{\mathbf{C}} \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}} \leq P_C \left(\overline{\mathbf{Y}} - \tilde{\mathbf{C}} \tilde{\mathbf{A}} x_0 - \tilde{\mathbf{C}} \tilde{\mathbf{D}} \mathbf{d}\right) \end{aligned} \tag{4.72}$$

Now this system has less free variables and constraints, depending on the blocking and condensing strategies. Let us define the blocking and condensing matrices as:

$$P_B|_{Nm \times (N-b)m} \quad : \quad \Re^{(N-b)m} \longrightarrow \Re^{Nm} \tag{4.73}$$

$$P_C|_{Np \times (N-c)p} \quad : \quad \Re^{(N-c)p} \longrightarrow \Re^{Np} \tag{4.74}$$

This type of blocking strategy would result in the following reduction in number of variables and constraints

Inputs: $\tilde{\mathbf{u}}_{(N-b)m \times 1}$

Outputs: $\tilde{\mathbf{y}}_{(N-c)p \times 1}$

Input constraints: $\tilde{V}_{2(N-b)m \times 2(N-b)m}$ and $\tilde{v}_{2(N-b)m \times 1}$

Output constraints: Converting to the standard form :

$$\begin{aligned} \begin{bmatrix} P_C \tilde{\mathbf{C}} \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}} \\ -P_C \tilde{\mathbf{C}} \tilde{\mathbf{B}} P_B \tilde{\mathbf{u}} \end{bmatrix} \quad &\leq \quad \begin{bmatrix} P_C \left(\overline{\mathbf{Y}} - \tilde{\mathbf{C}} \tilde{\mathbf{A}} x_0 - \tilde{\mathbf{C}} \tilde{\mathbf{D}} \mathbf{d}\right) \\ -P_C \left(\underline{\mathbf{Y}} - \tilde{\mathbf{C}} \tilde{\mathbf{A}} x_0 - \tilde{\mathbf{C}} \tilde{\mathbf{D}} \mathbf{d}\right) \end{bmatrix} \\ \tilde{W} \tilde{\mathbf{u}} \quad &\leq \quad \tilde{\mathbf{w}} \end{aligned}$$

and from there $\tilde{W}_{2(N-c)p\times 2(N-b)p}$ and $\tilde{\mathbf{w}}_{2(N-c)p\times 1}$

So total numbers can be summarized as follows:

Inputs	$(N-b)\,m$
Outputs	$(N-c)\,p$
Constraints	$2\left((N-c)\,p+(N-b)\,m\right)$

Multiscale Blocking and Condensing

Wavelet basis can be used to transform the input and output signals so that tresholding can be applied according to information theoretic criterion (minimizing the deviation from the signal). This method provides a wavelet domain blocking and condensing strategy Joseph et.al. [4] used this method. Although the method they are describing is a much better approach than the conventional ad hoc time domain blocking and condensing it still suffers from various weaknesses:

1. The blocking and condensing of the input and output signals are done during the design phase and kept the same afterwards.

2. There is no physical reasoning behind it. It is just a numerical approximation to reduce the computational load without suffering from performance and stability degradation.

3. Time domain blocking and condensing strategy described in the paper [4] does not guarantee that the output and input constraints will be satisfied.

4. There is a direct relationship between the outputs and the inputs. So doing blocking and condensing independently can cause inconsistent specifications and infeasible optimization problems.

Blocking as mentioned before is a natural outcome of Multiscale MPC formulation. Enforcing child nodes to remain constant automatically creates a blocking strategy, which will be described in the next section and an example will be given later in Chapter 5.

4.5 Multirate Multiscale MPC

4.5.1 Multirate Measurements

The most obvious case for multirate systems is the measurements taken at different sampling intervals. The problem becomes obvious if one tries to incorporate all of the information collected from the sensors at different rates into the MPC algorithm. Since the MPC algorithm utilizes a shifting frame of prediction and control horizons, at the start of each horizon not all measurements will be available. So the model-plant mismatch phase should fuse the available measurement to provide the best available information at the beginning of each open-loop optimization step. In Figure 4-2 you can see the measurements taken at 4 different rates.

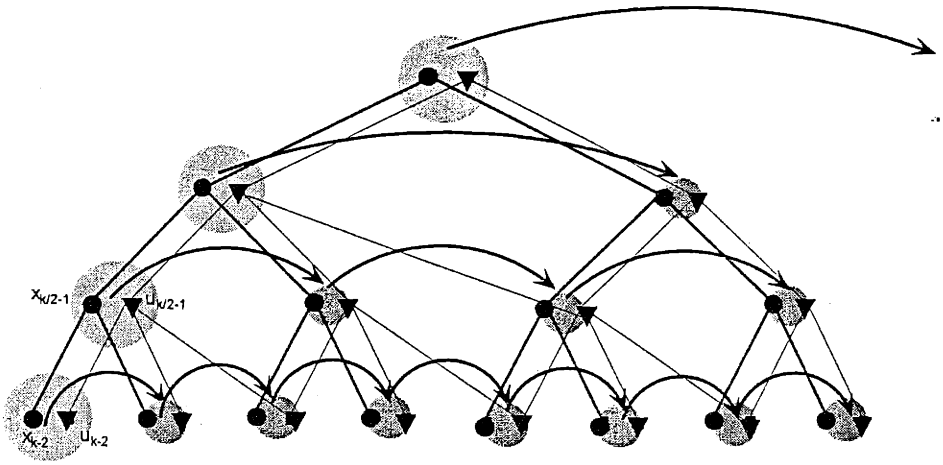


Figure 4-2: Multiscale Tree notation with multirate measurements

The multirate measurement case is simply providing the optimal model-plant mismatch information at the beginning of each open-loop optimization step.

There are two possible variations of multirate measurements:

1. *Different output variables are measured at different rates* : This case is very simple to handle. One only needs to take corresponding averages at the required scales and below the measurement scale one can either assume constant values or one can solve a state estimation problem to determine the finer scale values (with state estimator one optimally

fuses and reconciles the measurement data).

2. *Same output variable is measured with multiple sensors at different rates* : This type of measurements are done for calibration purposes and detection of sensor faults. This additional information (lower rate measurements) is valuable and should be used by optimally fusing it to the other measurement data. This can again be done by a multiscale state estimator.

The use of a multiscale state estimator greatly simplifies the multirate MPC algorithm, since all the synchronization is done by the estimator optimally. A more challenging problem is when there are actuators operating at different rates. This time not the estimator but its dual, the optimal controller, should be manipulated to incorporate the multiscale behavior of the inputs.

4.5.2 Multirate Measurements and Inputs

If we want to apply the inputs computed at higher scales we have to think about their meaning, i.e. about the physicality of the values.

There are two approaches to think about measurements and/or inputs (Figure 4-3):

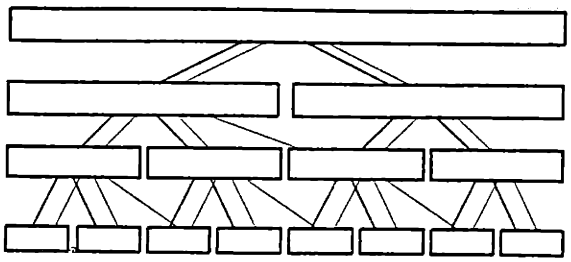
- Edge driven
- Interval driven

In both cases we assume that the measurements, taken practically at a single time point, correspond to an output value over a sampling interval. The starting position of that interval determines the type of the measurement model. The length of the interval is the same in both cases and is equal to the sampling interval.

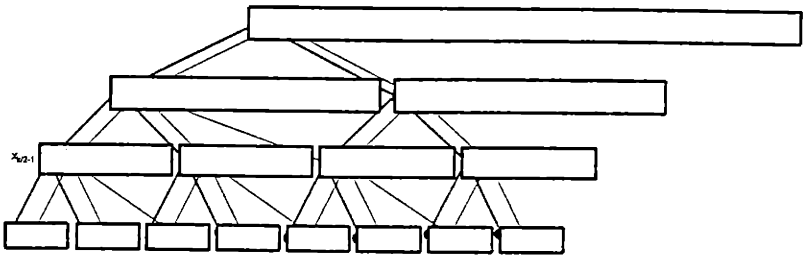
4.5.3 Development of Multirate Multiscale MPC

There are certain assumption we have to make:

1. Regularly spaced measurements and control actions



(a)



(b)

Figure 4-3: (a) Interval Driven (b)edge driven measurements

2. Modified edge driven type of measurement models incorporating the causal effects (important to determine when the inputs will be applied) (Figure $4-4$)

3. Below the physical scale δx and δu's are assumed to be zero.

There are two basic classes of this type of problems:

1. States evolve separately from each (the dynamic system is diagonal, i.e. there are no of diagonal terms which indicate interaction between states) other and can be computed so.

2. States interact. Models should be fused.

The first case is not interesting since the models can be applied separately, but the second case needs some attention. If the inputs are applied at different levels then the effect of an input at higher scales on the states at lower scales is as if the input is blocked at the children nodes. Figure 4-4 shows the measurements as intervals at different scales.

The easiest way to see how the fusion can be done is to create a simple example:

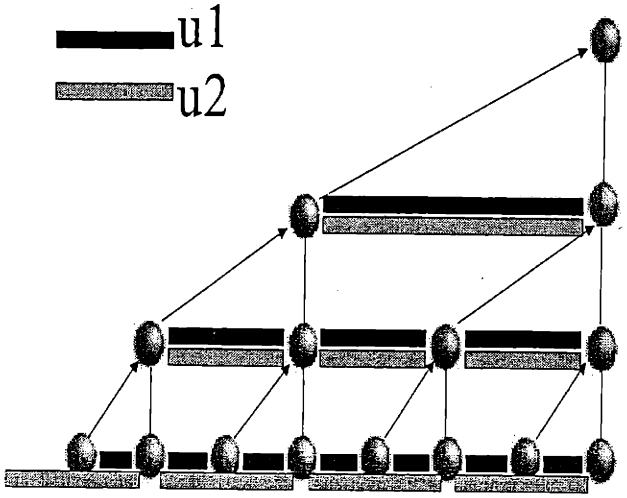


Figure 4-4: Multirate Multiscale

Every step of the derivation of the problem is the same except the additional constraint on the inputs u. Inputs which are applied at lower frequencies (thus belonging to higher scales) have children with the exact same constant value.

$$u_{-l,k} = \frac{1}{\sqrt{2}^{s-l}} u_{-s,m} \; \forall l < s \text{ and } 2^{s-l}m \leq k \leq 2^{s-l}(m+1) - 1 \tag{4.75}$$

if $u_{s,m}$ is applied at scale s. The rest of the problem remains the same and equation 4.75 is added to the set of equality constraints similar to the blocking constraints. Actually inputs applied at higher scales behave like blocked inputs at lower scales. The only point which needs to be clarified is how and when the inputs at higher scales should be applied. If we assume interval based representation then the input should be applied at the beginning of the interval (same time point as the first time line input point)

The state equations are (if the measurements are multirate too then estimation algorithm generates the best disturbance structure from all the measurements fused optimally):

$$\begin{aligned} \delta x_{-s-1,\frac{k}{2}} &= \frac{(I - (A_{-s})_{n\times n})}{\sqrt{2}} x_{-s,k} - \frac{(B_{-s})_{n\times m}}{\sqrt{2}} u_{-s,k} - \frac{(D_{-s})}{\sqrt{2}} d_{-s,k} \\ y_{-s,k} &= (C_{-s})_{m\times n} x_{-s,k} \end{aligned}$$

where

$$A_{-s} = A_0^{2^s} \tag{4.76}$$

$$B_{-s} = \left[\sum_{i=0}^{2^s-1} A_0^i\right] B_0 \tag{4.77}$$

$$D_{-s} = \sum_{i=0}^{2^s-1} A_0^i \tag{4.78}$$

$$C_{-s} = C_0 \tag{4.79}$$

The objective function can be derived from the regular MPC formulation by transforming it into multiscale domain:

$$\begin{aligned} &\min_{\delta\hat{\mathbf{x}}} \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)^T \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) + \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) H^T H \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) \\ &\min_{\delta\hat{\mathbf{x}}} \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right)^T \left(I + H^T H\right) \left(\delta\hat{\mathbf{R}} - \delta\hat{\mathbf{x}}\right) \end{aligned} \tag{4.80}$$

and additionally state equations over the horizon with length N $\left(\text{or scale } smax = \frac{\log(N)}{\log(2)}\right)$ can compactly be described as

$$\begin{aligned} \delta\hat{\mathbf{x}} &= \tilde{\mathbf{A}} x_{0,0} + \tilde{\mathbf{B}} \begin{pmatrix} u_{-smax+1,1} \\ \vdots \\ u_{0,N-1} \end{pmatrix} + \tilde{\mathbf{D}} \begin{pmatrix} d_{-smax+1,1} \\ \vdots \\ d_{0,N-1} \end{pmatrix} \\ \delta\hat{\mathbf{x}} &= \tilde{\mathbf{A}} x_{0,0} + \tilde{\mathbf{B}}\mathbf{u} + \tilde{\mathbf{D}}\mathbf{d} \end{aligned} \tag{4.81}$$

Then the input constraints are:

$$\underline{\mathrm{U}}_{-s} \leq \mathbf{u}_{-s,k} \leq \bar{\mathrm{U}}_{-s} \ \forall k, s \in T \tag{4.82}$$

The output bounds are:

$$\underline{\mathrm{Y}}_{-s} \leq \mathbf{y}_{-s,k} \leq \bar{\mathrm{Y}}_{-s} \ \forall k, s \in T \tag{4.83}$$

And the blocking because of the multirate character is :

$$u_{-l,k} = \frac{1}{\sqrt{2}^{s-l}} u_{-s,m} \; \forall l < s \text{ and } 2^{s-l} m \leq k \leq 2^{s-l}(m+1) - 1 \tag{4.84}$$

This type of blocking requires an additional information: When will the higher scale inputs be applied? (Check Figure 4-4)

4.6 The MSMPC Algorithm

In this section we will describe the algorithm to solve the multiscale problem. The MATLAB code for this algorithm can be found in Appendix B.

4.6.1 General Description

MPC

1. Define the model and initial conditions
2. Define specific inputs and disturbances
3. Define the horizon length
4. Start the loop by time interval k=1, if k>M go to 9
5. Call MATQP , feed initial **state**, model-plant **mismatch** and get the first element of the computed **input** profile, predicted **state** and the **deviation** from the desired reference path.
6. Calculate the real plant **output**, and **deviation** from the set point
7. Add the results to the **history** information, determine the model-plant **mismatch**
8. **k=k+1** and go to 4
9. **Plot** the results

MSMPC

1. Define the model and initial conditions
2. Define specific inputs and disturbances
3. Define the upper bound for scales
4. Start the loop by time interval k=1, if k>M go to 10
5. Call FINDSCALES , feed initial **state**, model-plant **mismatch** and get the required **scale** number, multiscale model-plant **mismatch** information.
6. Call MSMATQP , feed initial **state**, multiscale model-plant **mismatch** and get the first element of the computed **input** profile, predicted **state**.
7. Calculate the real plant **output**, and **deviation** from the set point
8. Add the results to the **history** information, determine the model-plant **mismatch**
9. k=k+1 and go to 4
10. **Plot** the results

4.6.2 Subprogram I: matqp

In:Initial state value x_0, model-plant mismatch at previous step d_0

Out:First element of the input profile u_0, predicted state value x_1,and deviation from the reference path

1. Create reference path

$$R_i = a - C^i b \; \forall i \in \{1, .., N\} \tag{4.85}$$

2. Create the matrices and vectors for

$$\mathbf{x} = \begin{pmatrix} A \\ \vdots \\ A^N \end{pmatrix} x_0 + \begin{pmatrix} B & \cdots & 0 \\ \vdots & \vdots & \vdots \\ A^{N-1}B & \cdots & B \end{pmatrix} \begin{pmatrix} u_0 \\ \vdots \\ u_{N-1} \end{pmatrix} + \begin{pmatrix} I & \cdots & 0 \\ \vdots & \vdots & \vdots \\ A^{N-1} & \cdots & I \end{pmatrix} \begin{pmatrix} d_0 \\ \vdots \\ d_0 \end{pmatrix} \tag{4.86}$$

$$\mathbf{x} = \mathbf{C}x_0 + \mathbf{Du} + \mathbf{D1d} \tag{4.87}$$

$$\mathbf{y} = \mathbf{C}_{output}\mathbf{x} \tag{4.88}$$

and the input, output bounds.

$$AA = \begin{bmatrix} \mathbf{C}_{output}\mathbf{D} & -I * slack \\ -\mathbf{C}_{output}\mathbf{D} & I * slack \end{bmatrix} \tag{4.89}$$

$$BB = \begin{bmatrix} \overline{Y} - \mathbf{C}_{output}\left(\mathbf{Cx}_0 + \mathbf{D1d}\right) \\ -\underline{Y} + \mathbf{C}_{output}\left(\mathbf{Cx}_0 + \mathbf{D1d}\right) \end{bmatrix} \tag{4.90}$$

3. Define

$$\tilde{\mathbf{R}} = \mathbf{R} - \mathbf{Cx}_0 - \mathbf{D1d} \tag{4.91}$$

$$H = \mathbf{D}^T\mathbf{D} \tag{4.92}$$

$$f = -\mathbf{D}^T\tilde{\mathbf{R}} \tag{4.93}$$

Optionally one can add slack variables for output constraint relaxation

$$\tilde{H} = \begin{bmatrix} H & 0 \\ 0 & I * pen \end{bmatrix} \tag{4.94}$$

$$\tilde{f} = \begin{bmatrix} f \\ 0 \end{bmatrix} \tag{4.95}$$

4. Solve the QP by

$$U = qp(\tilde{H}, \tilde{f}, AA, BB, ULB, UUB); \tag{4.96}$$

5. u_0 is the first element of U.

6. $x_1 = Ax_0 + Bu_0$

4.6.3 Subprogram II: findscales

In:Initial state value x_0, model-plant mismatch at previous steps $d_{0..k}$

Out:Required number of scales, multiscale model-plant mismatch vector,and

1. Create multiscale model parameters by

$$A_{-s} = A_0^{2^s} \tag{4.97}$$

$$B_{-s} = \left[\sum_{i=0}^{2^s-1} A_0^i\right] B_0 \tag{4.98}$$

$$D_{-s} = \sum_{i=0}^{2^s-1} A_0^i \tag{4.99}$$

$$C_{-s} = C_0 \tag{4.100}$$

2. Create the blocking parameters for inputs according to the equation:

$$\begin{aligned} u_N^{BLOCK} = {} & B^{-1}\left[\sum_{k=1}^{2^{N-1}} kA^{2^{N-1}-k}\right]^{-1}\left\{\sqrt{2}^{N-1}\left[\sum_{k=0}^{2^{N-1}-1} A^k\right]B\right\} u_{-N+1,1}^{AVE} - \\ & B^{-1}\left[\sum_{k=1}^{2^{N-1}} kA^{2^{N-1}-k}\right]^{-1}\left\{\sqrt{2}^{N-2}\left[\sum_{k=2^{N-2}}^{2^{N-1}-1} A^k\right]B\right\} u_{-N+2,1}^{AVE} - \\ & B^{-1}\left[\sum_{k=1}^{2^{N-1}} kA^{2^{N-1}-k}\right]^{-1}\left\{\left[\sum_{k=1}^{2^{N-2}-1} k\left(I + A^{2^{N-2}}\right)A^k\right]B\right\} u_{0,N-1}^{BLOCK} \end{aligned}$$

or briefly

$$u_N^{BLOCK} = c_2^N u_{-N+1,1}^{AVE} - c_3^N u_{-N+2,1}^{AVE} - c_4^N u_{0,N-1}^{BLOCK} \tag{4.101}$$

3. Create the model-plant mismatch multiscale representation $d_{0-s_{max},1}$ either by Haar or Modified Hat averaging scheme.

4. Create the left most nodes of the multiscale reference path $R_{-s,1}$ and $\delta R_{-s,1}$

5. Start from scale $s = 0$ and prepare the objective function. The main objective is to find an feasible upper bound for the number of scales. For that purpose the regular objective function for each scale is converted into:

$$\min_u \left(R_{-s,2} - x_{-s,2}\right)^T \left(R_{-s,2} - x_{-s,2}\right) \tag{4.102}$$

Then manipulating this objective function by the following identities and substitutions

$$\begin{aligned} R_{-s,2} &= R_{-s,1} - \sqrt{2}\delta R_{-s-1,1} \\ x_{-s,2} &= x_{-s,1} - \sqrt{2}\delta x_{-s-1,1} \end{aligned} \qquad \text{(Haar)}$$

$$\min_u \left(R_{-s,1} - \sqrt{2}\delta R_{-s-1,1} - x_{-s,1} + \sqrt{2}\delta x_{-s-1,1}\right)^T \left(R_{-s,1} - \sqrt{2}\delta R_{-s-1,1} - x_{-s,1} + \sqrt{2}\delta x_{-s-1,1}\right)$$

$$\varepsilon_{-s-1,1} = \frac{1}{\sqrt{2}}\left(R_{-s,1} - x_{-s,1}\right) \tag{4.103}$$

$$\min_u 2\left(\delta R_{-s-1,1} - \delta x_{-s-1,1} - \varepsilon_{-s-1,1}\right)^T \left(\delta R_{-s-1,1} - \delta x_{-s-1,1} - \varepsilon_{-s-1,1}\right) \tag{4.104}$$

$$\delta x_{-s-1,1} = \frac{(I - A_{-s})}{\sqrt{2}} x_{-s,1} - \frac{B_{-s}}{\sqrt{2}} u_{-s,1} - \frac{D_{-s}}{\sqrt{2}} d_{-s,1} \tag{4.105}$$

$$\delta \tilde{R}_{-s-1,1} = \delta R_{-s,1} - \frac{(I - A_{-s})}{\sqrt{2}} x_{-s,1} + \frac{D_{-s}}{\sqrt{2}} d_{-s,1} - \varepsilon_{-s-1,1} \tag{4.106}$$

$$H = B_{-s}^T B_{-s} \tag{4.107}$$

$$f = \sqrt{2} B_{-s}^T \delta \tilde{R}_{-s-1,1} \tag{4.108}$$

$$AA_{out} = \begin{bmatrix} C_{-s} B_{-s} \\ -C_{-s} B_{-s} \end{bmatrix} \tag{4.109}$$

$$BB_{out} = \begin{bmatrix} \sqrt{2}^s \overline{Y} - C_{-s} \left(A_{-s} x_{-s,1} + D_{-s} d_{-s,1} \right) \\ -\sqrt{2}^s \underline{Y} + C_{-s} \left(A_{-s} x_{-s,1} + D_{-s} d_{-s,1} \right) \end{bmatrix} \tag{4.110}$$

$$AA_{in} = \begin{bmatrix} c_2^s \\ -c_2^s \end{bmatrix} \tag{4.111}$$

$$BB_{in} = \begin{bmatrix} \overline{U} + c_3^N u_{-N+2,1}^{AVE} + c_4^N u_{0,N-1}^{BLOCK} \\ -\underline{U} - c_3^N u_{-N+2,1}^{AVE} - c_4^N u_{0.N-1}^{BLOCK} \end{bmatrix} \tag{4.112}$$

So the QP problem becomes

$$uscale = qp(H, f, [AA_{in}; AA_{out}], [BB_{in}; BB_{out}], ULO, UUP); \tag{4.113}$$

4.6.4 Subprogram III: msmatqp

In:Initial state value x_0, model-plant mismatch at previous steps $d_{0..k}$

Out:Applied input, predicted state

1. Create the objective function

$$\left(\delta R - \delta x \right)^T \left(I + G^T G \right) \left(\delta R - \delta x \right) \tag{4.114}$$

$$H = I + G^T G \tag{4.115}$$

$$f = -H^T \delta R \tag{4.116}$$

where

$$x_{TOP} = G\delta x + g x_{0,1} \tag{4.117}$$

2. Create the input bounds

$$\begin{aligned}
\underline{U} &\leq \mathbf{u} \leq \overline{U} && (4.118)\\
\delta x &= \tilde{\mathbf{A}} x_{0,1} + \tilde{\mathbf{B}}\mathbf{u} + \tilde{\mathbf{D}}\mathbf{d} && (4.119)\\
\mathbf{u} &= \tilde{\mathbf{B}}^{-1}\left(\delta x - \tilde{\mathbf{A}} x_{0,1} - \tilde{\mathbf{D}}\mathbf{d}\right) &&\\
\underline{U} + \tilde{\mathbf{B}}^{-1}\left(\tilde{\mathbf{A}} x_{0,1} - \tilde{\mathbf{D}}\mathbf{d}\right) &\leq \tilde{\mathbf{B}}^{-1}\delta x \leq \overline{U} + \tilde{\mathbf{B}}^{-1}\left(\tilde{\mathbf{A}} x_{0,1} - \tilde{\mathbf{D}}\mathbf{d}\right) && (4.120)\\
\begin{bmatrix} \tilde{\mathbf{B}}^{-1} \\ -\tilde{\mathbf{B}}^{-1} \end{bmatrix} \delta x &\leq \begin{bmatrix} \overline{U} + \tilde{\mathbf{B}}^{-1}\left(\tilde{\mathbf{A}} x_{0,1} - \tilde{\mathbf{D}}\mathbf{d}\right) \\ -\underline{U} - \tilde{\mathbf{B}}^{-1}\left(\tilde{\mathbf{A}} x_{0,1} - \tilde{\mathbf{D}}\mathbf{d}\right) \end{bmatrix} && (4.121)\\
AA_{in} &= \begin{bmatrix} \tilde{\mathbf{B}}^{-1} \\ -\tilde{\mathbf{B}}^{-1} \end{bmatrix} && (4.122)\\
BB_{in} &= \begin{bmatrix} \overline{U} + \tilde{\mathbf{B}}^{-1}\left(\tilde{\mathbf{A}} x_{0,1} - \tilde{\mathbf{D}}\mathbf{d}\right) \\ -\underline{U} - \tilde{\mathbf{B}}^{-1}\left(\tilde{\mathbf{A}} x_{0,1} - \tilde{\mathbf{D}}\mathbf{d}\right) \end{bmatrix} && (4.123)
\end{aligned}$$

3. Create the output bounds

$$\begin{aligned}
\underline{Y} &\leq \mathbf{y} \leq \overline{Y} && (4.124)\\
\mathbf{y} &= C\mathbf{x} && (4.125)\\
\mathbf{x} &= \Phi\delta x + \phi x_{0,1} && (4.126)\\
\underline{Y} &\leq C\Phi\delta x + \leq \overline{Y} \\
\underline{Y} - \mathbf{C}\phi x_{0,1} &\leq C\Phi\delta x \leq \overline{Y} - \mathbf{C}\phi x_{0,1} && (4.127)\\
\begin{bmatrix} C\Phi \\ -C\Phi \end{bmatrix}\delta x &\leq \begin{bmatrix} \overline{Y} - \mathbf{C}\phi x_{0,1} \\ -\underline{Y} + \mathbf{C}\phi x_{0,1} \end{bmatrix} && (4.128)\\
AA_{out} &= \begin{bmatrix} C\Phi \\ -C\Phi \end{bmatrix} && (4.129)\\
BB_{out} &= \begin{bmatrix} \overline{Y} - \mathbf{C}\phi x_{0,1} \\ -\underline{Y} + \mathbf{C}\phi x_{0,1} \end{bmatrix} && (4.130)
\end{aligned}$$

4. Solve the QP by

$$\delta x = qp(H, f, [AA_{in}; AA_{out}], [BB_{in}; BB_{out}], ULB, UUB); \quad (4.131)$$

5.

$$\mathbf{u} = \tilde{\mathbf{B}}^{-1}\left(\delta x - \tilde{\mathbf{A}}x_{0,1} - \tilde{\mathbf{D}}\mathbf{d}\right)$$

u_0 is the first element of $\mathbf{u}$.

6. $x_1 = Ax_0 + Bu_0$

4.7 Notation

$x_{-s,k}$: States at node k and scale s

$u_{-s,k}$: Inputs at node k and scale s

$y_{-s,k}$: Outputs at node k and scale s

$R_{-s,k}$: Reference path at node k and scale s

$d_{-s,k}$: Disturbance at node k and scale s

$v_{-s,k}$: Measurement noise at node k and scale s

$w_{-s,k}$: Modeling noise at node k and scale s

$A_{-s}, B_{-s}, C_{-s}, D_{-s}$:System parameters at scale s

T: The binary tree

$\underline{U}_{-s}$: Lower input constraint at scale s

$\bar{U}_{-s}$: Upper input constraint at scale s

$\underline{Y}_{-s}$: Lower output constraint at scale s

$\bar{Y}_{-s}$: Upper output constraint at scale s

Bibliography

[1] N. Gupta. Frequency-shaped cost functionals: Extension of linear-quadratic-gaussian design methods. *J. Guidance and Control*, 3(6):529–535, 1980.

[2] O. Karsligil, M. Dyer, and G. Stephanopoulos. Multiscale model predictive control. AICHE 1999 Annual Meeting, Nov 1999.

[3] L. Lublin and M. Athans. Linear quadratic regulator control. *The Control Handbook*, 1996.

[4] S. Palavajjhala, R. Motard, and B. Joseph. Blocking and condensing design for QDMC using wavelets. *I E C Res.*, 33:1159–1173, 1994.

[5] C. Rao and J. Rawlings. Steady states and constraints in model predictive control. *AICHE Journal*, 45(6):1266–1278, 1999.

[6] J. Rawlings and K. Muske. The stability of constrained receeding horizon control. *IEEE Transactions on Automatic Control*, 38(10):1512–1516, 1993.

Chapter 5

Case Studies

To demonstrate the effectiveness of the MSMPC formulation and the algorithm to solve it several case studies will be used. In each case study the performance of MSMPC will be compared to the classical MPC algorithm.

The workings of the algorithms are discussed in great detail in the previous chapter and the MATLAB codes of the subroutines are listed in Appendix B.

5.1 Case 1: Set Point Change

In this case we will look at a system with the following dynamic model[1]:

$$A = \begin{bmatrix} 0.2 & 0.3 \\ 0.4 & 0.7 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & -2 \\ 1 & -1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

with the input constraints

$$\bar{U} = \begin{bmatrix} -1.7100 \\ -1.6200 \end{bmatrix} \text{ and } \underline{U} = \begin{bmatrix} -2.5300 \\ -1.9800 \end{bmatrix}$$

[1] $x_{k+1} = Ax_k + Bu_k$
$y_k = Cx_k$

This is a stable dynamic system (eig(A)=[0.0228 0.8772]) We assume that all the states are fully observable. There are no modeling or measurement errors and external disturbances. The reference path we want the dynamic model to follow is a step from the starting point to the final target:

$$x_{ini} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \rightarrow x_{fin} = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

The horizon length for the classical MPC is fixed at 7 time units, whereas MSMPC has a variable horizon length with an upper bound of 7 time units and the simulation is done for 15 time units. Prediction and control horizons are of equal length.

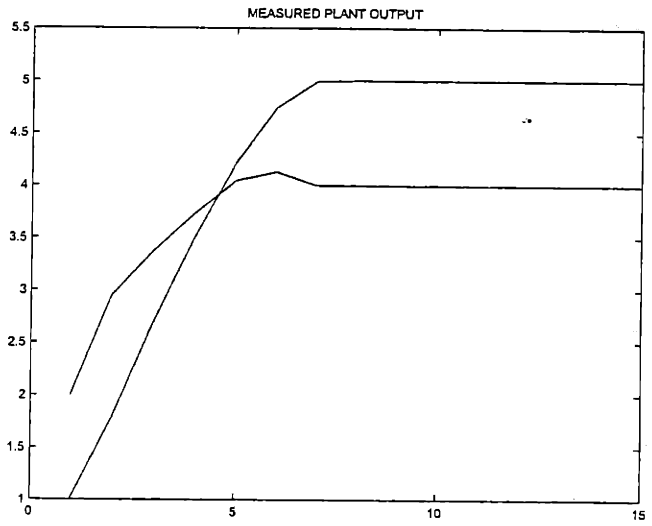


Figure 5-1: (case1) Outputs (solid line: MPC, dashed line: MSMPC)

As the results suggest there is practically no difference in the solutions. Both algorithms perform equally well. The very important point is though, that MSMPC had a horizon length of 7 only for 6 time units, then the horizon length fell to 1, indicating a transition from an optimal control strategy to a simple feedback control scheme (a proportional controller with a time varying gain) without any degradation in performance.

At the beginning of the simulation the difference between the starting point and the target is largest, so there is a strong need for input energy. Both inputs are at the constraint up to

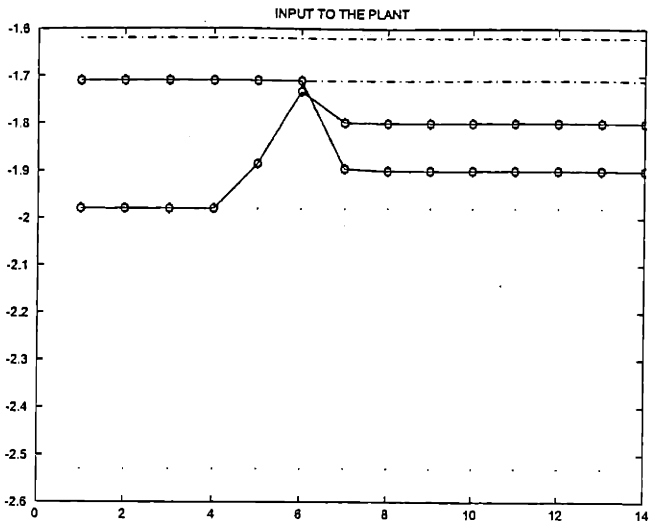


Figure 5-2: (case1) Inputs (squares: MSMPC, diamonds: MPC, _._: upper bounds ...: lower bounds)

time point 4 (Figure (5-2)). At that point input u_2 leaves the constraint and starts increasing while the first input remains at its upper bound. At time point 6 the first input also leaves the bound and it marks the moment when there is no more need for the longer control horizon: the horizon length falls to a unit length. That there are no active constraints indicates that at that point there exists an interior solution for the inputs, which can bring the states to their final set point, and this can be seen in figure (5-1) at time point 7: both states reach their new target and remain there.

Conclusion 5.1 *The main lessons from this case is that the MPC and MSMPC algorithms perform equally well, and provide as expected the same solution, but MSMPC arrives at the solution faster by applying a variable control horizon strategy.*

The complexities of both algorithms can be compared indirectly by measuring the number of floating point operations:

$$flops_{MPC} = 4.023.850$$

$$flops_{MSMPC} = 2.246.188$$

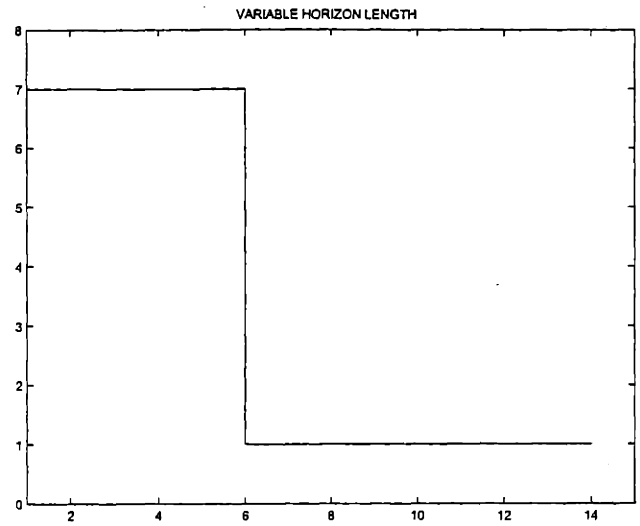


Figure 5-3: (case1) Change in the horizon length in MSMPC

One should take these numbers as approximate measures for complexity though, since the underlying code is not written for maximum efficiency.

5.2 Case 2: Disturbance Rejection

In this case we will look at a system with the following dynamic model:

$$A = \begin{bmatrix} 0.2 & 0.3 \\ 0.4 & 0.7 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & -2 \\ 1 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

with the input constraints

$$\bar{U} = \begin{bmatrix} 0.5500 \\ -0.5400 \end{bmatrix} \text{ and } \underline{U} = \begin{bmatrix} 0.0900 \\ -0.6600 \end{bmatrix}$$

This is a stable dynamic system (eig(A)=[0.0228 0.8772]) We assume that all the states are fully observable. There are no modeling or measurements errors, but an external disturbance,

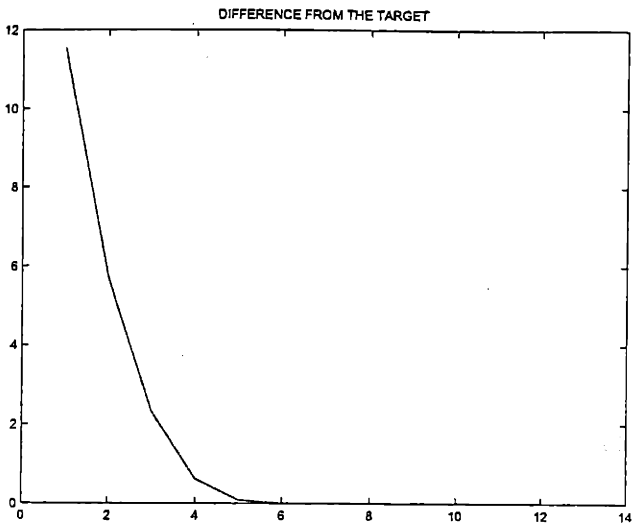


Figure 5-4: (case1) Deviation from the target points (solid line: MPC, dashed line: MSMPC)

which hits state 1 with a magnitude of 1 at time point 5 and continues for two time units. The set point is fixed at:

$$x_{set} = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

The horizon length for the classical MPC is fixed at 7 time units, whereas MSMPC has a variable horizon length with an upper bound of 7 time units and the simulation is done for 15 time units. Prediction and control horizons are of equal length.

As the results suggest there is practically no difference in the solutions. Both algorithms perform equally well. The very important point is though, that MSMPC has a horizon length of 7 only for 3 time units, then the horizon length falls to 3 for one time unit and finally it is reduced to 1, indicating a transition from an optimal control strategy to a simple feedback control scheme without any degradation in performance.

At the beginning of the disturbance the difference between the state point and the set point is largest, so there is a strong need for input energy. Both inputs are at the constraint up to time point 6 (Figure (5-6)). At that point input u_2 leaves the bounds and starts decreasing while the first input jumps to its upper bound. At time point 9 the first input also leaves the

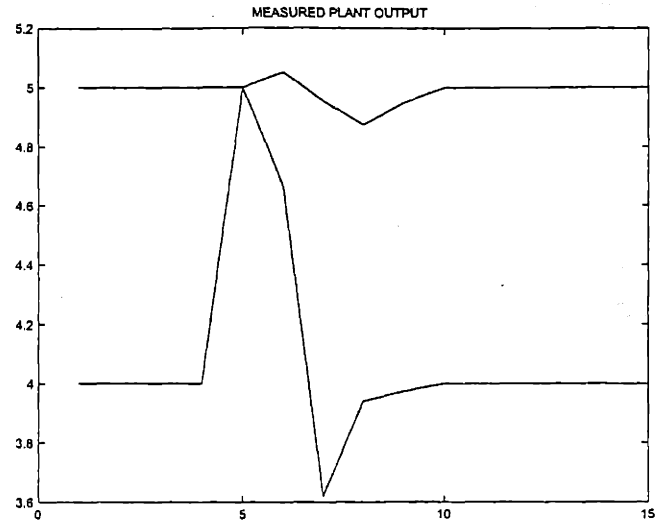


Figure 5-5: (case 2) Outputs (solid line: MPC, dashed line: MSMPC)

bound and it marks the moment when there is no more need for the longer control horizon: the horizon length falls to a unit length. That there are no active constraints indicates that at that point there exists an interior solution for the inputs, which can bring the states to their final set point, and this can be seen in figure (5-5) at time point 10: both states reach their target and remain there.

Conclusion 5.2 *The main lessons from this case is that the MPC and MSMPC algorithms perform equally well, and provide as expected the same solution, but MSMPC arrives at the solution faster by applying a variable horizon strategy.*

The complexities of both algorithms can be compared indirectly by measuring the number of floating point operations:

$$flops_{MPC} = 3.882.595$$

$$flops_{MSMPC} = 1.495.681$$

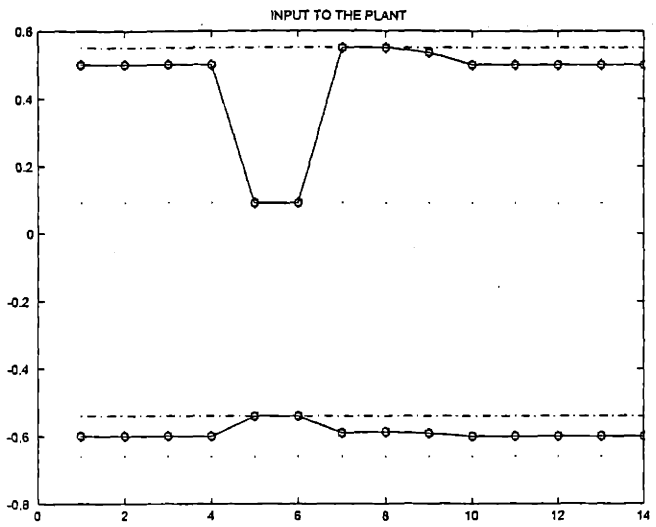


Figure 5-6: (case 2) Inputs (squares: MSMPC, diamonds: MPC, _.._: upper bounds ...: lower bounds)

5.3 Case 3: Unstable Dynamic System

In this case we will look at a system with the following dynamic model:

$$A = \begin{bmatrix} 1.1 & 0.3 \\ 0.6 & 1.1 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

with the input constraints

$$\bar{U} = \begin{bmatrix} 8.0850 \\ 5.0400 \end{bmatrix} \text{ and } \underline{U} = \begin{bmatrix} 2.9450 \\ 1.7100 \end{bmatrix}$$

This is an unstable dynamic system (eig(A)=[1.5243 0.6757]) We assume that all the states are fully observable. There are no modeling or measurements errors,and no external disturbance The reference path we want the dynamic model to follow is a first order response starting from

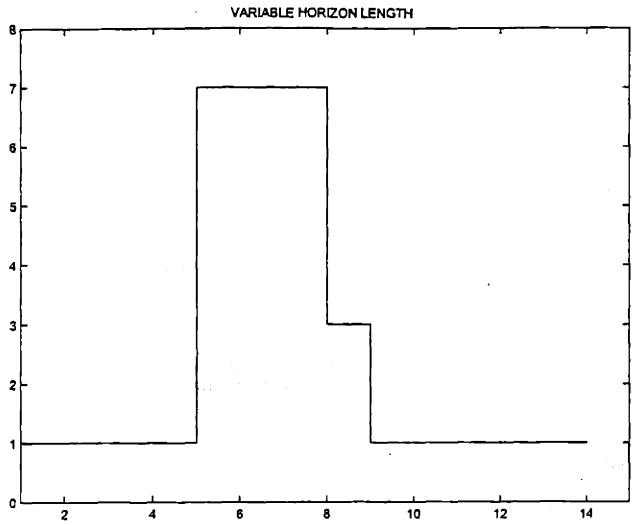


Figure 5-7: (case 2) Change in the horizon length in MSMPC

the current state value to the final target (figure (5-10)) :

$$R_k = x_{fin} - \begin{bmatrix} 0.8 & 0 \\ 0 & 0.3 \end{bmatrix}^k (x_{fin} - x_{cur})$$

The initial and final setpoints are:

$$x_{ini} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \rightarrow x_{fin} = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

The horizon length for the classical MPC is fixed at 3 time units, whereas MSMPC has a variable horizon length with an upper bound of 7 time units and the simulation is done for 25 time units. Prediction and control horizons are of equal length.

This case is designed to demonstrate the flexibility gained through variable control horizon. Although the MSMPC is allowed to have longer control horizons than the classical MPC the number of total computations performed by the MSMPC algorithm is less.

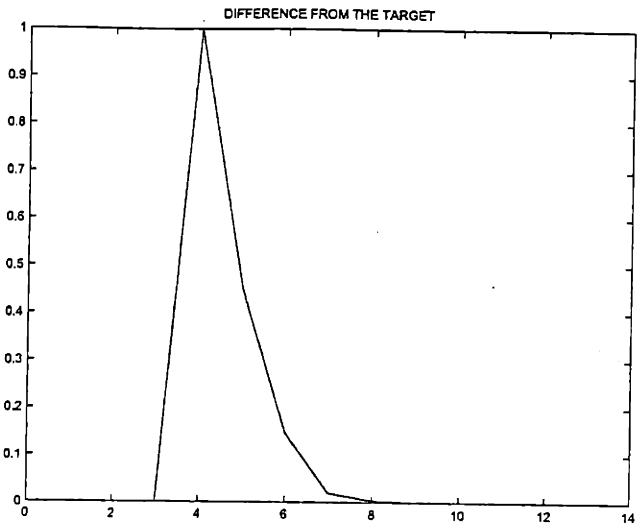


Figure 5-8: (case 2) Deviation from the target points (solid line: MPC, dashed line: MSMPC)

$$flops_{MPC} = 2.640.600$$

$$flops_{MSMPC} = 1.955.519$$

Still the performance of the MSMPC is superior. Since MSMPC can see further in time (prediction) at each open-loop step it can take more optimal steps to minimize deviation from the reference path. At the beginning of the simulation the performance of MSMPC is worse compared to MPC for a short time period, because MSMPC deploys the inputs such that the deviation is minimized globally, whereas the MPC algorithm with its short horizon length can only do local minimizations at each open-loop step. As a result MSMPC reacts fast, brings the states close enough to the reference path to deactivate the active input constraints, and thereafter the horizon length drops to 1. A final note on this case is that the MSMPC algorithm activates the input constraints for a shorter time period.

Conclusion 5.3 *Variable control horizon improves the speed and performance. MSMPC algorithm brings the states faster into the interior feasible region, where no input constraint is active.*

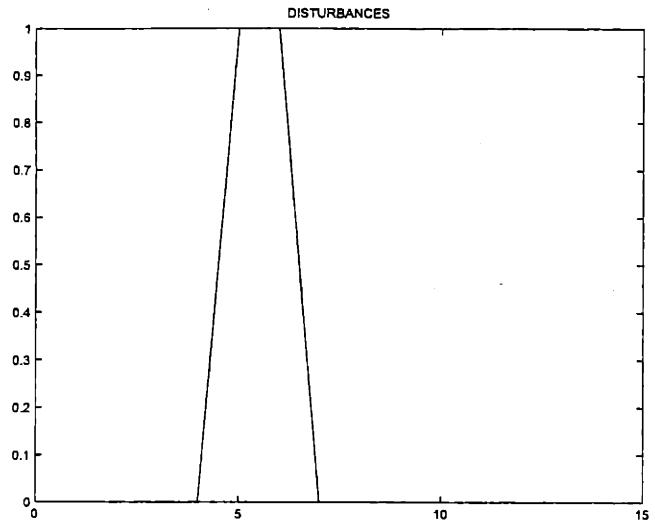


Figure 5-9: (case 2) External disturbance in state x_1, measured as the difference between the predicted output and measured output

5.4 Case 4: Modeling Error and a Multiscale Disturbance Model

This case demonstrates the effect of modeling error and different disturbance models. Before going into the details, let us describe what we mean by different disturbance models. One major weakness of MPC formulation is its limited use of disturbance models. Most of the MPC algorithms use the simple error feedback as a disturbance model:

$$d = x_{measured} - x_{computed}$$

This error includes modeling and measurement errors and the external disturbances. This term is assumed to remain constant along the prediction horizon in the open-loop step and so it is added as a constant correction term to the dynamic model:

$$x_{k+1} = Ax_k + Bu_k + d$$

In reality though modeling error changes in magnitude as the values of states change, and measurement errors are never constant. This type of model update is useful if there is a bias term either in measurements (sensor failure), plant-model mismatch or in external disturbances

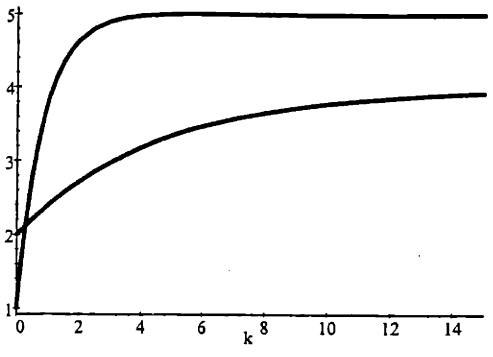


Figure 5-10: Reference path for Case3 at the initial point

(like in integrating systems), but it fails if the deviation between the measured and predicted states vary with time, which occurs most of the time. It is almost impossible to predict the future deviations in the open-loop step, but one can extract more information from the history of the plant at various scales, i.e. at various time intervals. This additional information can be very valuable. In situations like day-night temperature swings the only way to capture such changes and incorporate them into the open-loop optimization is to use a multiscale disturbance model. Also in cases where there are multirate measurements, the related deviations should be incorporated in a multiscale manner (**Dyer [1]**).

Among many other possible multiscale disturbance models we will use the most obvious one: The history of all deviations will be stored. They will be indexed such that the last deviation measured will be the first element and they will be transformed into the multiscale domain by using the modified Hat wavelet, since the deviations behave like inputs. The coefficients on the left most branch are the multiscale deviations which will be fed to the MSMPC algorithm (see figure **(5-15)**).

This case uses the following dynamic model:

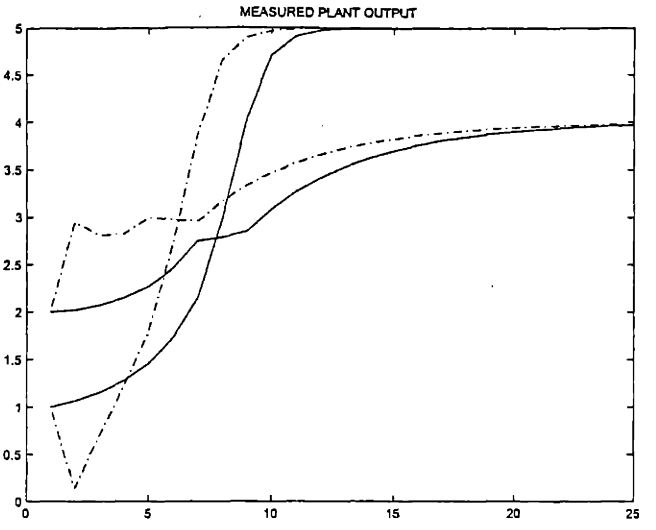


Figure 5-11: (case 3) Outputs (solid line: MPC, dashed line: MSMPC)

$$A = \begin{bmatrix} 1.1 & 0.3 \\ 0.6 & 1.1 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & -2 \\ 1 & 1 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

with the input constraints

$$\bar{U} = \begin{bmatrix} -0.7689 \\ 0.5377 \end{bmatrix} \text{ and } \underline{U} = \begin{bmatrix} -1.8872 \\ -0.0002 \end{bmatrix}$$

The plant has a slightly different dynamic model (modeling error):

$$A_P = \begin{bmatrix} 1.25 & 0.39 \\ 0.48 & 0.89 \end{bmatrix} \text{ and } B_P = \begin{bmatrix} 1.1 & -1.9 \\ 1.05 & 1.01 \end{bmatrix}$$
$$C_P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

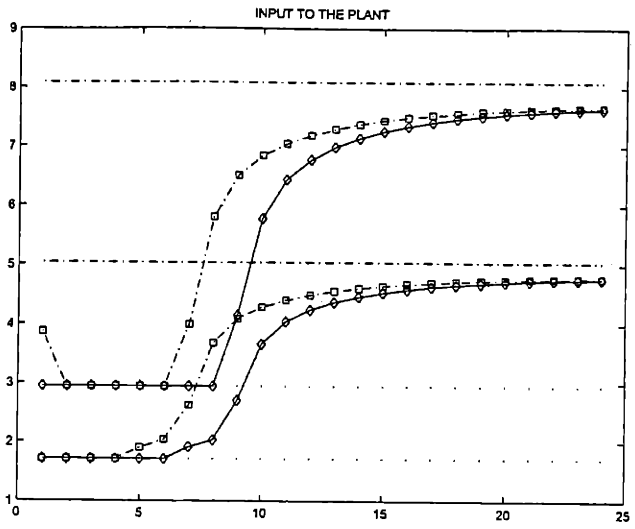


Figure 5-12: (case 3) Inputs (squares: MSMPC, diamonds: MPC, _.._: upper bounds ...: lower bounds)

In addition to the modeling error there is also an external disturbance between 5th and 15th time units during the simulation. The magnitude of the disturbance is 0.1 in the first output and -0.1 in the second output. Measurements are assumed to be perfect, and all states are observable as before. The object is to reject the disturbance, and compensate for the modeling error.

To be able to make a reasonable comparison both algorithms are forced to use a 7 point control and prediction horizons. The plots for outputs and deviations start from time point 7, because the MSMPC algorithm requires at least 7 deviation points to be able to provide multiscale deviation information at all 3 scales. So the first 7 points are like training, because the missing deviation points are assumed to be zero which is not true.

Conclusion 5.4 *The MSMPC algorithm with the multiscale disturbance model performs better than the MPC algorithm, because the history data initiates a stronger and faster response to the external disturbance and almost constant modeling error. After the external disturbance stops, MSMPC still contains enough information about the magnitude of the modeling error and returns the states faster to their set points.*

Besides MSMPC is still faster than the MPC algorithms as one can see from the flops

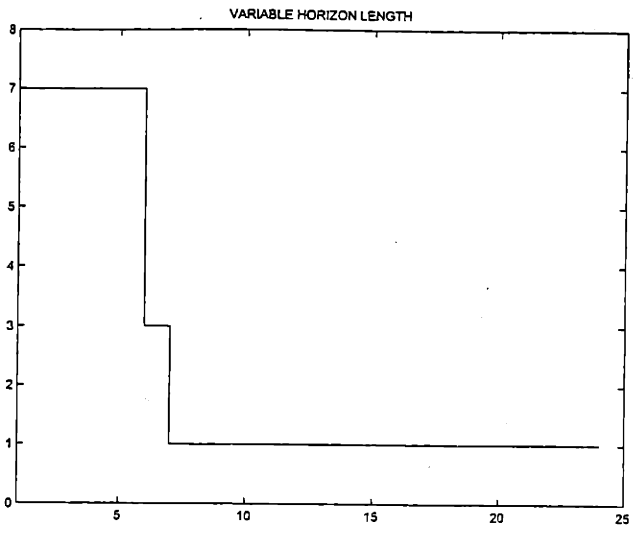


Figure 5-13: (case 3) Change in the horizon length in MSMPC

numbers.

$$flops_{MPC} = 19.676.297$$

$$flops_{MSMPC} = 11.583.903$$

5.5 Case 5: Modeling error and Output Constraints (feasibility issues)

As mentioned before input constraints are mostly physical constraints and they are by their nature form an always feasible solution space. On the other hand output constraints can generate infeasibilities, especially if the horizon length is chosen inappropriately. The following case demonstrates such an issue. The problem definition includes modeling error and external disturbances, too. Regular MPC has a horizon length of 5; the MSMPC can have a horizon length up to 15 points. MSMPC algorithm uses a multiscale disturbance model described as in the previous case. The objective of the case is to reject the disturbance and compensate for

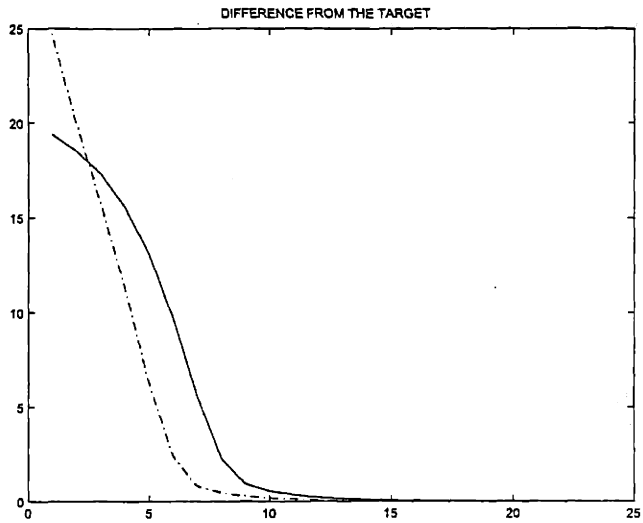


Figure 5-14: (case 3) Deviation from the target points (solid line: MPC, dashed line: MSMPC)

the modeling error. The dynamic model used for the prediction is:

$$A = \begin{bmatrix} 1.1 & 0.3 \\ 0.6 & 1.1 \end{bmatrix} \text{ and } B = \begin{bmatrix} 1 & -2 \\ 1 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 2 & -1 \end{bmatrix}$$

with the input constraints

$$\bar{U} = \begin{bmatrix} -0.7770 \\ 0.5793 \end{bmatrix} \text{ and } \underline{U} = \begin{bmatrix} -2.0863 \\ 0.0120 \end{bmatrix}$$

and output constraints:

$$\bar{Y} = \begin{bmatrix} 5 \\ 1 \\ 5.5 \end{bmatrix} \text{ and } \underline{Y} = \begin{bmatrix} 0 \\ -2 \\ 2.5 \end{bmatrix}$$

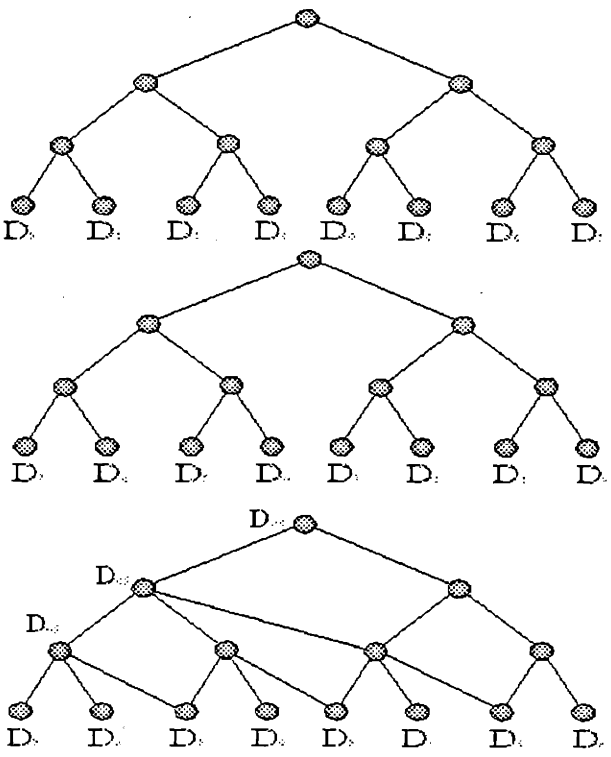


Figure 5-15: Multiscale Disturbance Model

The plant has a slightly different dynamic model parameter A (modeling error):

$$A = \begin{bmatrix} 1.25 & 0.39 \\ 0.48 & 0.89 \end{bmatrix}$$

Like in the previous case up to point 7 the results generated are not reliable since there is no previous history of deviations (including modeling error or disturbances) and they are assumed to be zero in MSMPC algorithm.

Conclusion 5.5 *MPC having a very short horizon length (for demonstration purposes) cannot generate a feasible solution space and states cannot be brought to their set points. As a result first inputs start violating their bounds and then outputs start also diverging. MSMPC on the other hand extends its horizon length as much as possible to provide enough degrees freedom to*

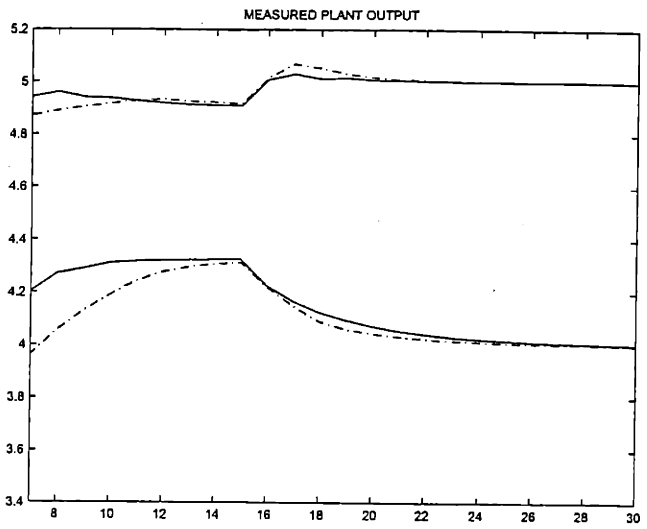


Figure 5-16: (case 4) Outputs (solid line: MPC, dashed line: MSMPC)

reject the disturbance and compensate for the modeling error. MSMPC manages to bring the states back to their targets and no constraints are violated.

5.6 Case 6: Heavy Oil Fractionator (HOF)

The process considered here is a heavy oil fractionator as shown in figure (5-25). The column has three product drawns and three side circulating loops. The heat requirement of the column enters with the feed, which is a gaseous stream. Product specifications for the top and side draws are determined by economics and operating requirements. There is no product specification for the bottom draw, but there is an operating constraint on the temperature in the lower part of the column. The three circulating loops remove heat to achieve the desired product separation. The heat exchangers in these loops reboil columns in other parts of the plant, and therefore have varying heat duty requirements. The bottom loop has an enthalpy controller which regulates heat removal in the loop by adjusting steam make: its heat duty can be used as a manipulated variable to control the column.

This problem is presented in the **Shell Process Control Workshop** [3] and does not represent an existing process, but it contains all significant elements of a real fractionator

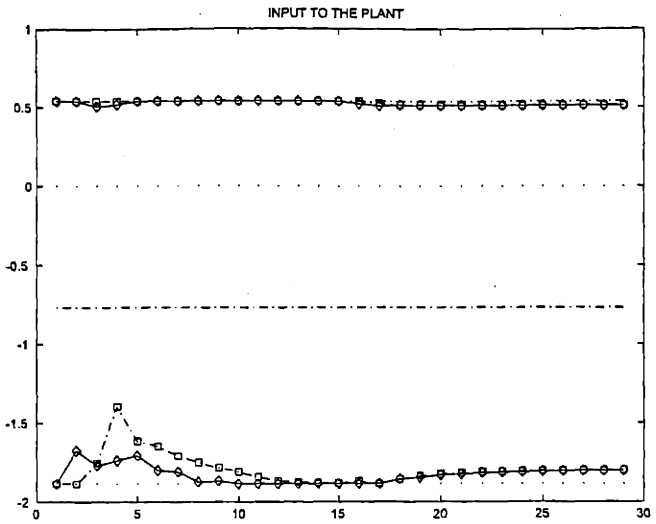


Figure 5-17: (case 4) Inputs (squares: MSMPC, diamonds: MPC, _.._: upper bounds ...: lower bounds)

control problem.

The model of this plant is given in frequency domain and needs to be transformed into time domain in order to be used in MPC and MSMPC algorithms. The form of the given model is

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = H \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} \tag{5.1}$$

where the outputs and inputs correspond to:

- y_1: Top end point
- y_2: Side end point
- y_3: Top temperature

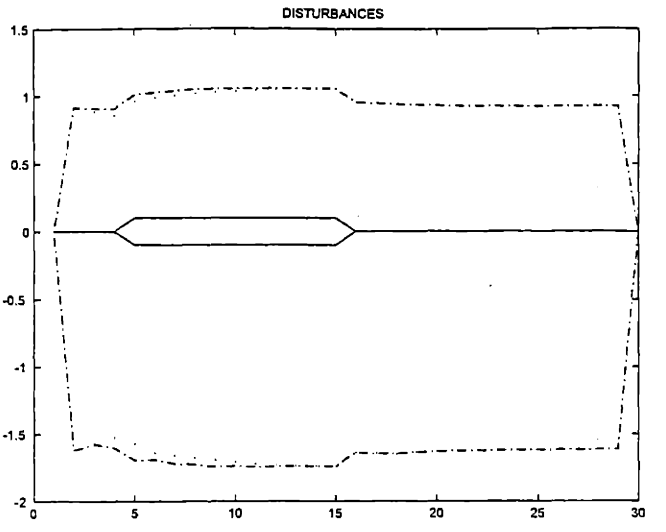


Figure 5-18: (case 4) Deviation from the target points (solid line: External Disturbance, dashed line: MPC deviations, dotes: MSMPC deviations)

- y_4: Upper reflux temperature
- y_5: Side draw temperature
- y_6: Intermediate reflux temperature
- y_7: Bottoms reflux temperature
- u_1: Top draw
- u_2: Side draw
- u_3: Bottoms reflux duty
- u_4: Intermediate reflux duty
- u_5: Upper reflux duty

As mentioned in the description above observing the outputs y_3,y_4,y_5 and y_6 is not necessary, since they cannot be used as feedback information. A similar elimination can be done also for

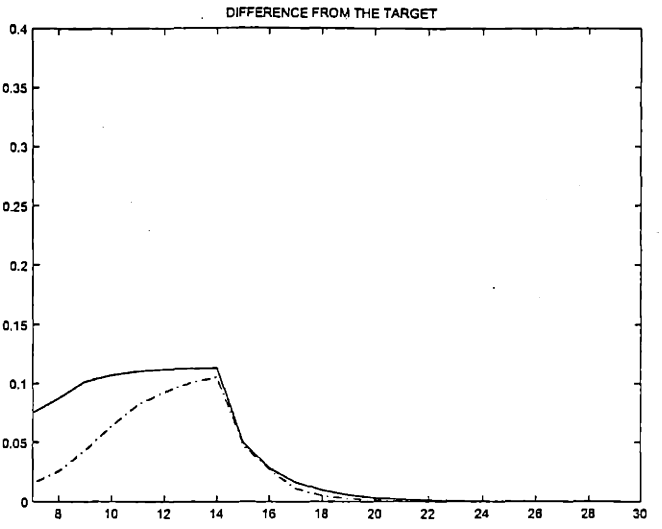


Figure 5-19: (case 4) Deviation from the target points (solid line: MPC, dashed line: MSMPC)

inputs: u_4 and u_5 can not be manipulated, but they effect the system as external disturbances.

$$H_r = \begin{bmatrix} \frac{0.08019z+0.2313}{z^8-0.9231z^7} & \frac{0.1142}{z^8-0.9355z^7} & \frac{0.1164z+0.3357}{z^8-0.9231z^7} \\ \frac{0.2113z+0.2032}{z^6-0.9231z^5} & \frac{0.1876z+0.181}{z^5-0.9355z^4} & \frac{0.1704z+0.4864}{z^5-0.9048z^4} \\ \frac{0.4998}{z^6-0.8859z^5} & \frac{0.1962z+0.1879}{z^7-0.9131z^6} & \frac{1.367}{z-0.8102} \end{bmatrix} \tag{5.2}$$

$$H_d = \begin{bmatrix} \frac{0.0264z+0.0756}{z^8-0.925z^7} & \frac{0.03557z+0.1015}{z^8-0.9048z^7} \\ \frac{0.05958z+0.1652}{z^5-0.8521z^4} & \frac{0.0893z+0.2425}{z^5-0.8187z^4} \\ \frac{0.157}{z-0.8623} & \frac{0.148}{z-0.8825} \end{bmatrix} \tag{5.3}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_7 \end{bmatrix} = H_r \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + H_d \begin{bmatrix} u_4 \\ u_5 \end{bmatrix} \tag{5.4}$$

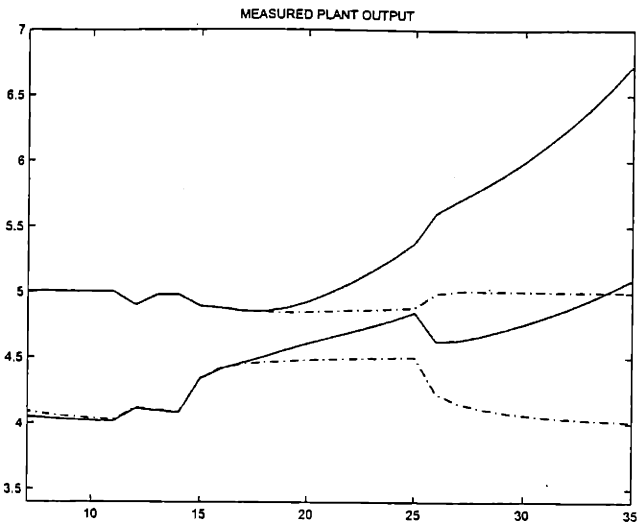


Figure 5-20: (case 5) Outputs as in the previous cases (not the constrained ones)(solid line: MPC, dashed line: MSMPC)

There are also uncertainties in steady state gains, any they introduce the modeling errors:

$$H_{r\varepsilon} = \begin{bmatrix} \frac{(0.08019z+0.2313)}{z^8-0.9231z^7}\left(1+\frac{2.11\varepsilon_1}{4.05}\right) & \frac{0.1142}{z^8-0.9355z^7}\left(1+\frac{0.39\varepsilon_2}{1.77}\right) & \frac{0.1164z+0.3357}{z^8-0.9231z^7}\left(1+\frac{0.59\varepsilon_3}{5.88}\right) \\ \frac{0.2113z+0.2032}{z^6-0.9231z^5}\left(1+\frac{3.29\varepsilon_1}{5.39}\right) & \frac{0.1876z+0.181}{z^5-0.9355z^4}\left(1+\frac{0.57\varepsilon_2}{5.72}\right) & \frac{0.1704z+0.4864}{z^5-0.9048z^4}\left(1+\frac{0.89\varepsilon_3}{6.9}\right) \\ \frac{0.4998}{z^6-0.8859z^5}\left(1+\frac{3.11\varepsilon_1}{4.38}\right) & \frac{0.1962z+0.1879}{z^7-0.9131z^6}\left(1+\frac{0.73\varepsilon_2}{4.42}\right) & \frac{1.367}{z-0.8102}\left(1+\frac{1.33\varepsilon_3}{7.2}\right) \end{bmatrix}$$

$$H_{d\varepsilon} = \begin{bmatrix} \frac{0.0264z+0.0756}{z^8-0.925z^7}\left(1+\frac{0.12\varepsilon_4}{1.2}\right) & \frac{0.03557z+0.1015}{z^8-0.9048z^7}\left(1+\frac{0.16\varepsilon_5}{1.44}\right) \\ \frac{0.05958z+0.1652}{z^5-0.8521z^4}\left(1+\frac{0.13\varepsilon_4}{1.52}\right) & \frac{0.0893z+0.2425}{z^5-0.8187z^4}\left(1+\frac{0.13\varepsilon_5}{1.83}\right) \\ \frac{0.157}{z-0.8623}\left(1+\frac{0.18\varepsilon_4}{1.14}\right) & \frac{0.148}{z-0.8825}\left(1+\frac{0.18\varepsilon_5}{1.26}\right) \end{bmatrix} \quad (5.5)$$

Detailed information about the nature of these modeling errors can be found in **Shell Process Control Workshop** [3].

So a state space model describing this system should have the following form:

$$[x_{k+1}]_{n\times 1} = A_{n\times n}[x_k]_{n\times 1} + B_{n\times 3}[u_k]_{3\times 1} + G_{n\times 2}[d_k]_{2\times 1} \quad (5.6)$$

$$[y_k]_{3\times 1} = C_{3\times n}[x_k]_{n\times 1} + D_{3\times 3}[u_k]_{3\times 1} \quad (5.7)$$

As expected there are many different ways for the realization of this system. The minimum

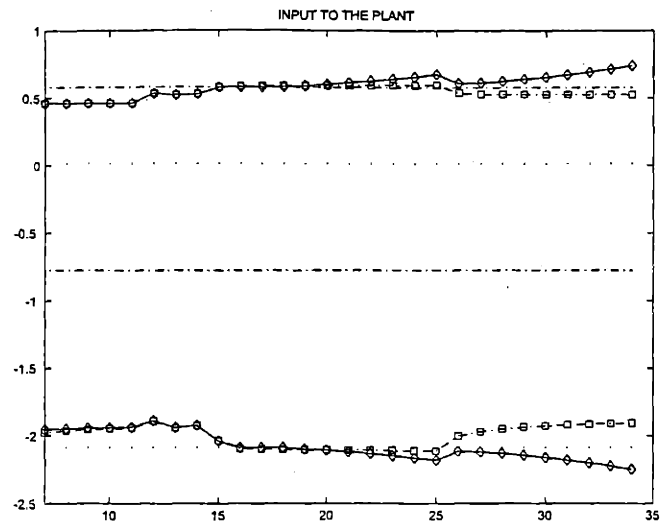


Figure 5-21: (case 5) Inputs (squares: MSMPC, diamonds: MPC, _.._: upper bounds ...: lower bounds)

realization of the system is very large and not controllable, so there is a need for model reduction. Model reduction is a major topic for itself and we don't go much into detail, but one can check the grammian of the system and determine a cut-off point. Such a reduction technique would preserve as much as possible from the system both in time and frequency domain. If the technique described above is applied to the HOF one can reduce the original 32 by 32 (not controllable) state model to a 8 by 8 controllable and observable system:

$$\begin{aligned} [x_{k+1}]_{8\times 1} &= A_{8\times 8}\,[x_k]_{8\times 1} + B_{8\times 3}\,[u_k]_{3\times 1} + G_{8\times 2}\,[d_k]_{2\times 1} \\ [y_k]_{3\times 1} &= C_{3\times 8}\,[x_k]_{8\times 1} \end{aligned}$$

One can check the quality of the model reduction by comparing the step responses and looking at the singular values along a wide range of frequencies.

As one can see from the figures (5-26,5-27) the reduced model matches the original systems characteristics closely. There are some discrepancies in frequency domain and steady state gains, but they are negligible and besides MPC and MSMPC algorithms both will be tested with the same reduced model. With this level of matching the reduced model does not require modified

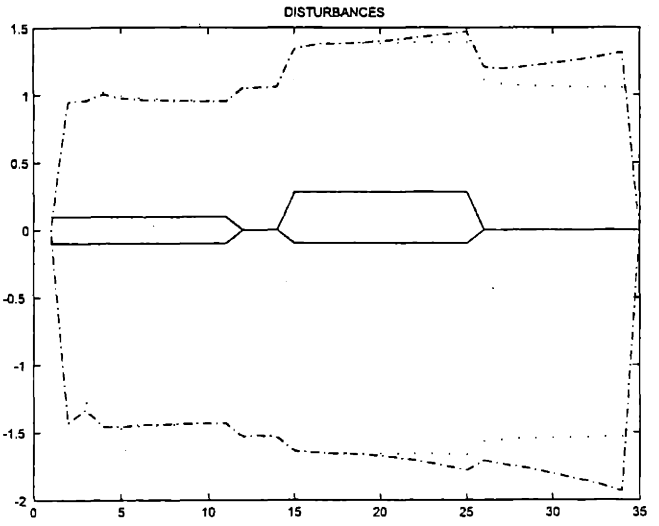


Figure 5-22: (case 5) Deviation from the target points (solid line: External Disturbance, dashed line: MPC deviations, dotes: MSMPC deviations)

criteria for control performance either. The next section lists the original control performance criteria which we will apply to our reduced system, too.

5.6.1 Control performance criteria

The HOF problem definition also includes the requirements for the control scheme:

- Maintain

$$-0.005 \leq y_1, y_2 \leq 0.005 \tag{5.8}$$

  at steady state.

- minimize u_3

- reject disturbances

$$-0.5 \leq u_4, u_5 \leq 0.5 \tag{5.9}$$

  (these are disturbances, even though they appear as inputs)

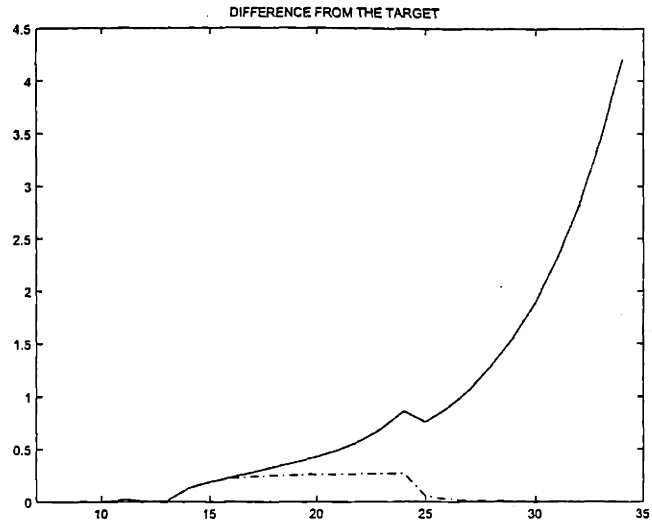


Figure 5-23: (case 5) Deviation from the target points (solid line: MPC, dashed line: MSMPC)

- Bound

$$-0.5 \leq u_1, u_2, u_3 \leq 0.5 \tag{5.10}$$

- Bound

$$-0.5 \leq y_7 \leq 0.5 \tag{5.11}$$

- Bound

$$-0.5 \leq y_1, y_2 \leq 0.5 \tag{5.12}$$

There is further discussion on additional assumptions and requirements and more information on those details can be found in **Prett and Garcia** [2].

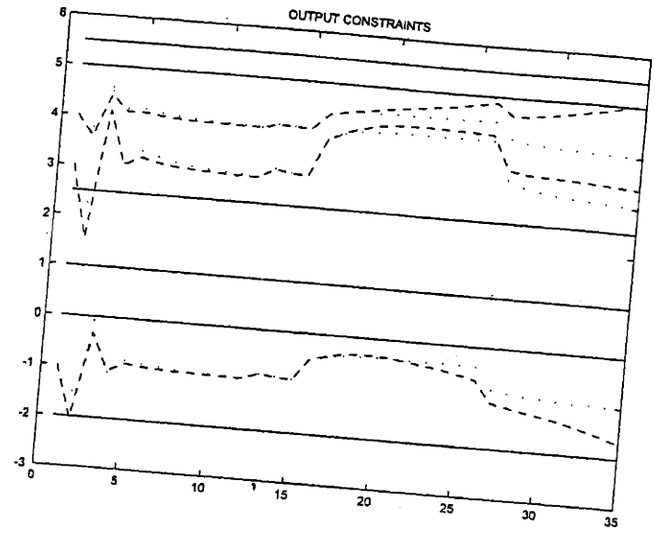


Figure 5-24: (case 5) Constrained outputs (Solid lines: bounds dashed lines: MPC outputs dotes: MSMPC outputs)

5.6.2 No Modeling Error, but Uncontrolled Inputs

So this first scenario has the following configuration:

$$\begin{aligned} \varepsilon_1 &= \varepsilon_2 = \varepsilon_3 = \varepsilon_4 = \varepsilon_5 = 0 \\ d_k &= \begin{bmatrix} 0.5 \\ -0.5 \end{bmatrix} \end{aligned}$$

Meaning that the upper reflux duty is at 0.5 and the intermediate reflux duty is at -0.5 (all these variables are dimensionless, and in the form of deviation variables). To make a legitimate comparison MPC will have a maximum control horizon length of 7 time units and MSMPC will be allowed to have a maximum of 3 scales, so that MSMPC has at most the same size as the classical MPC.

All of the figures (5-29, 5-30, 5-28, 5-31) start at time 10 because up to that time the system has to run without a disturbance load to settle at its steady state. The results show no significant difference in performance, except the MSMPC algorithm shrinks to a 1 unit long control horizon after time 20, transforming the advanced control algorithm into a simple proportional controller with a time varying gain.

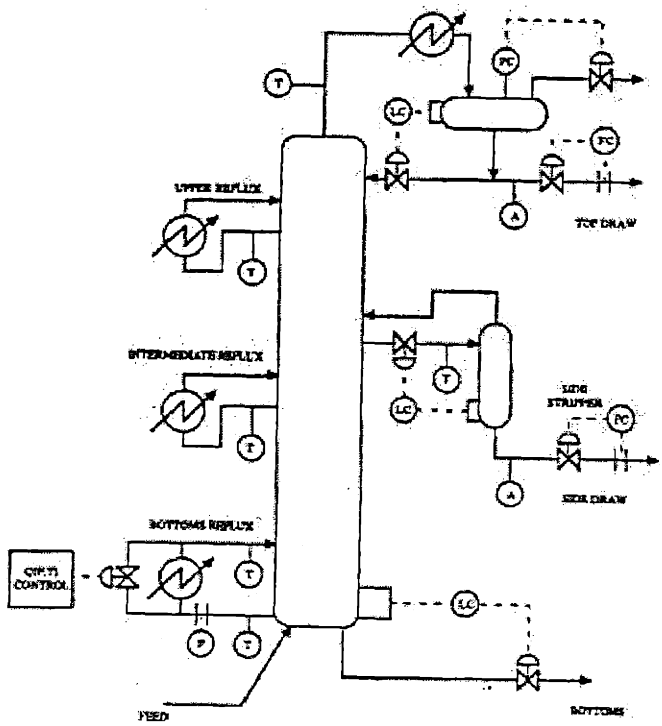


Figure 5-25: (case 6) Diagram of a Heavy Oil Fractionator (Shell Control Problem)

5.6.3 Modeling Error, Uncontrolled Inputs

The second scenario has the following configuration:

$$\begin{aligned} \varepsilon_1 &= \varepsilon_2 = \varepsilon_3 = -0.5 \;\; \varepsilon_4 = \varepsilon_5 = 0.5 \\ d_k &= \begin{bmatrix} -0.2 \\ -0.2 \end{bmatrix} \end{aligned}$$

Meaning that the upper reflux duty and the intermediate reflux duty are at -0.2. To make a legitimate comparison MPC will have a maximum control horizon length of 7 time units and MSMPC will be allowed to have a maximum of 3 scales, so that MSMPC has at most the same size as the classical MPC. In addition to having a variable control horizon MSMPC also uses a multiscale error feedback structure as described in case 5.4.

The results (figures (5-39, 5-33, 5-34, 5-35, 5-36, 5-37, 5-38)) indicate the positive effect

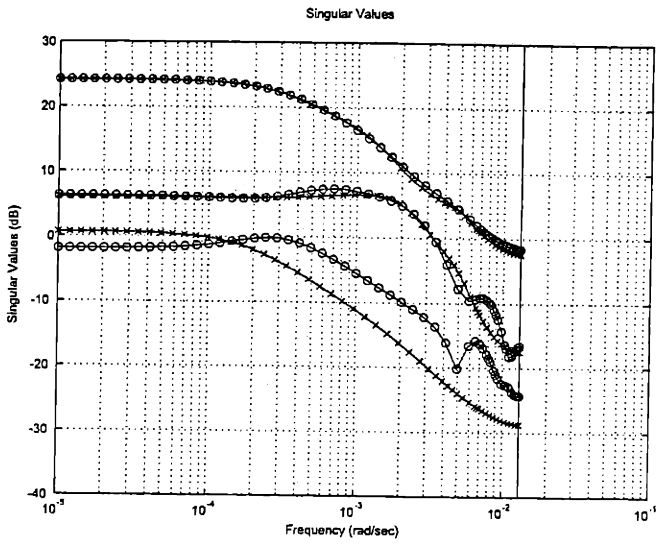


Figure 5-26: (case 6) Change of singular values with frequency (o: original system *: reduced system)

of the multiscale error feedback. Since the controller knows more about the systems behavior over different scales it can respond accordingly and eliminates the oscillations observed in the MPC solution. The deviation from the target is a good measure for the performance, but more importantly in the MSMPC solution the states and outputs reach their target values with much less variation than the MPC solution. This is very important since the outputs represent the quality of the end product and oscillations can create a lot of waste product. Of course MSMPC solution can be considered also as a better control strategy since the inputs do not oscillate much either, requiring less change in the inputs increasing the life time of the actuators.

5.7 Case 7: A Multirate Multiscale MPC Example

This case has the following dynamic model parameters:

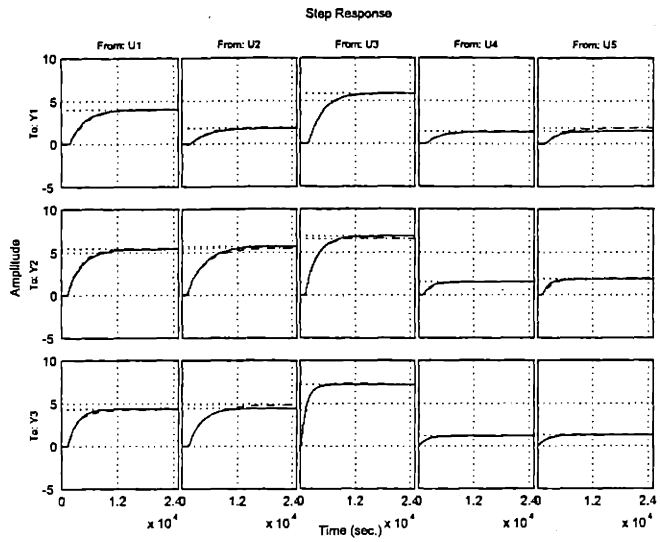


Figure 5-27: (case 6) Step response comparisons (solid line: original dashed: reduced model)

$$A = \begin{pmatrix} 0.2 & 0.3 \\ 0.4 & 0.7 \end{pmatrix}$$

$$B = \begin{pmatrix} 1 & -2 \\ 1 & 1 \end{pmatrix}$$

$$C = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

u_1 is applied every time interval and u_2 every second time interval and held constant during that period.

$$x_{ini} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

$$x_{fin} = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$$

Maximum number of scales is 4.

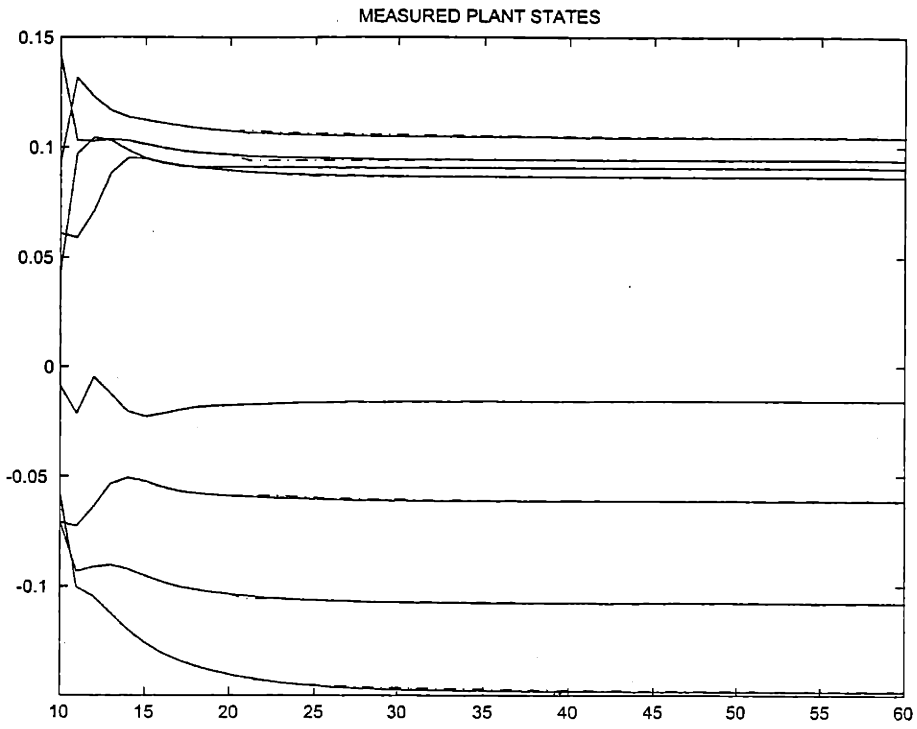


Figure 5-28: (case 6.1) The dynamics in the states as a result of the disturbance hitting the system. (solid: MPC dashed: MSMPC) (here states do not necessarily correspond to physical variables)

The algorithm to solve this problem should be separated into two sections for both time domain and multiscale MPC:

1. Open loop optimization starts with fresh inputs, i.e. the previous input had the last blocked component.

2. Open loop optimization starts with a blocked input, i.e. there will be an equality constraint on the first input to satisfy the blocking and thus the multirate behavior in the closed loop solution.

For time domain MPC blocking is easy: Block every second u_2 starting either from the beginning of the open loop horizon (case 1) or from the second input point on where the u_2 element of the first input is fixed by the previous applied input to satisfy the closed loop

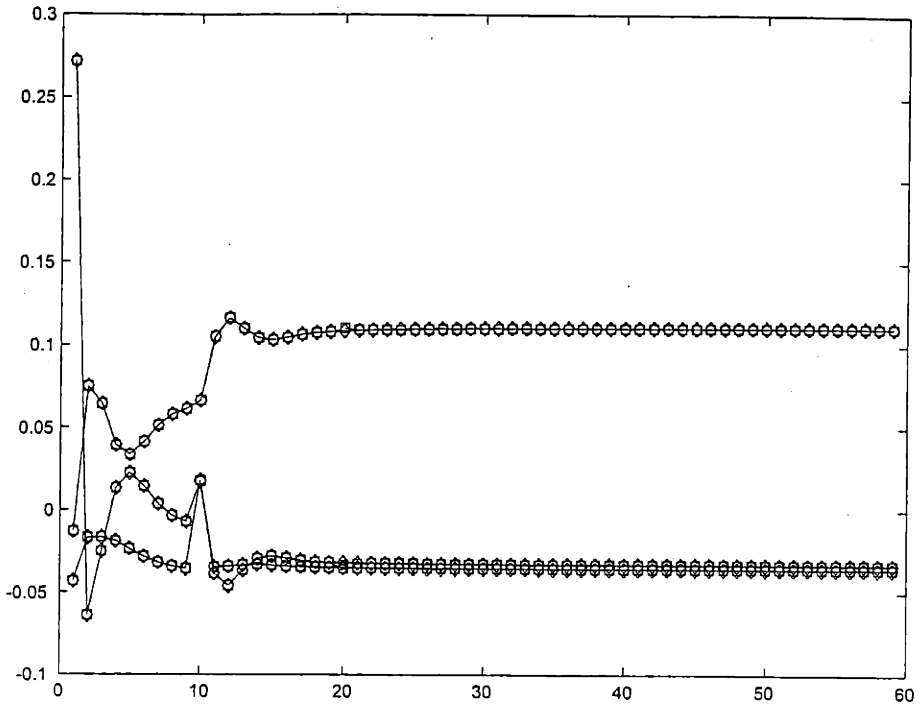


Figure 5-29: (Case 6.1) Inputs to the plant (diamonds: MPC squares: MSMPC)

blocking. Blockings appear as equality constraints and are very easy to incorporate into the problem.

For the MSMPC the situation is more difficult. We can do the same type of blocking in MSMPC but then that blocking will be only along the 0^{th} scale (time line). If we want to utilize the multiscale characteristics i.e. that the inputs at higher scales are real inputs which can be applied, then we can compute u_2 at scale 1 and u_1 at scale 0. The u_2 values at scale 0 will be blocked accordingly and will be the same as their parent nodes. The major difficulty in applying this scheme is case 1, when both inputs start fresh. Since the tree in figure (4-4) allows blocking only starting from the second input (which is ideal for case 2) we have to change the definition of the initial point: We start with the same tree but we force the first input to be the same as the last computed and applied input and the initial point of the tree is computed by

$$x_0 = A^{-1}\left(x_{lastmeasurement} - Bu_{lastapplied}\right) \tag{5.13}$$

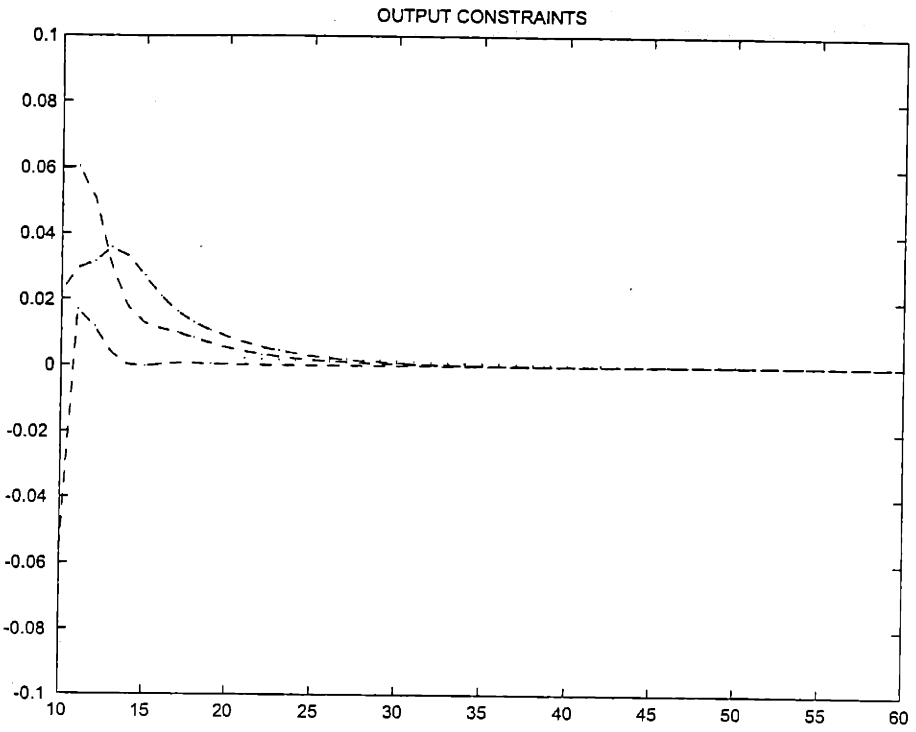


Figure 5-30: (case 6.1) The Outputs and constraints (here the constriants are at -0.5 and 0.5, so they are not shown in the fiigure, all outputs satisfy the constraints)

so that the second node on the time line has the correct measured state value.

The comparison of the solutions indicates that the MSMPC performs better than the MPC, not necessarily because MSMPC is a better computational environment for multirate systems (in terms of formulation it is definitely more convenient than the regular MPC formulation), but because the variable control horizon length allows great flexibility both in computational aspects and in capturing the multirate behavior of the system.

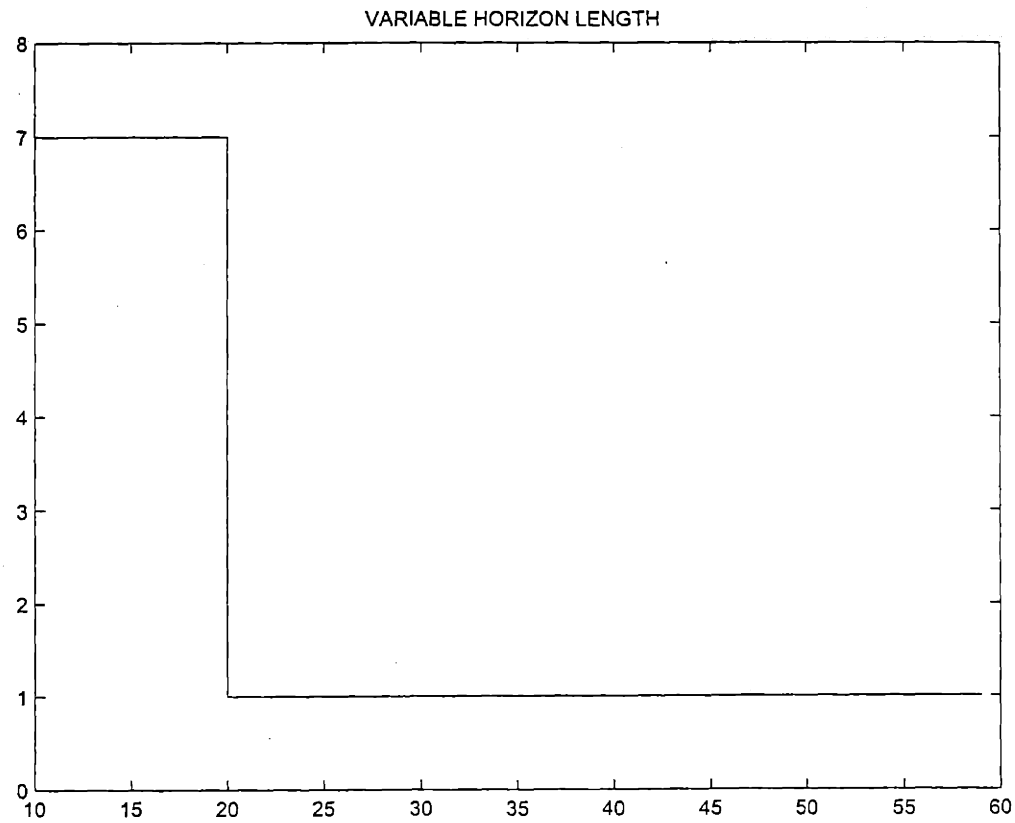


Figure 5-31: (case 6.1) The variable horizon length of the MSMPC

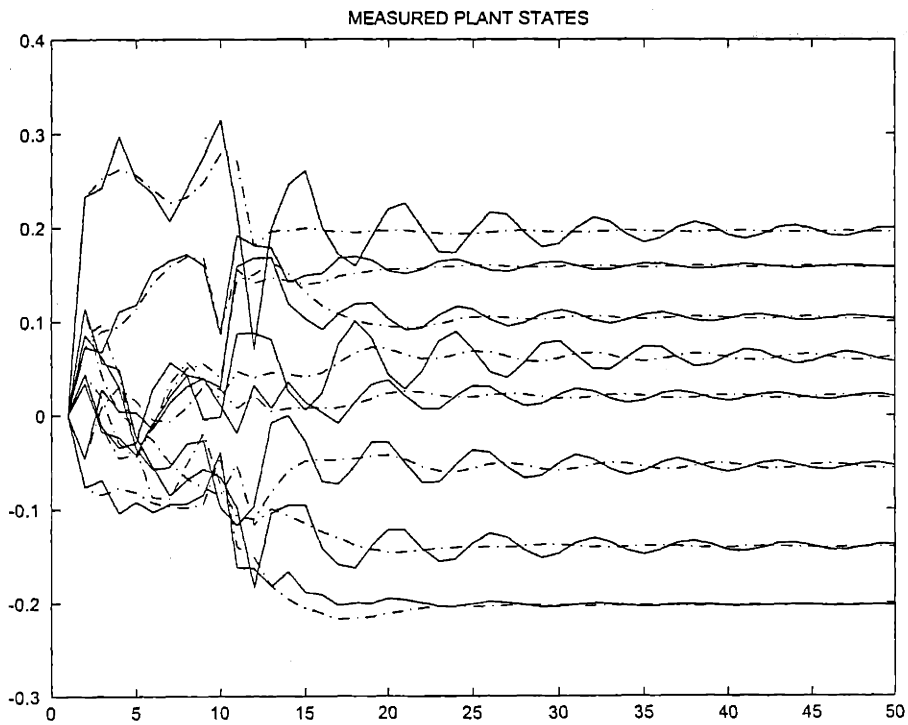


Figure 5-32: (case 6.2) The dynamics in the states as a result of the disturbance hitting the system. (States do not necessarily represent any physical parameters) (solid: MPC dashed: MSMPC)

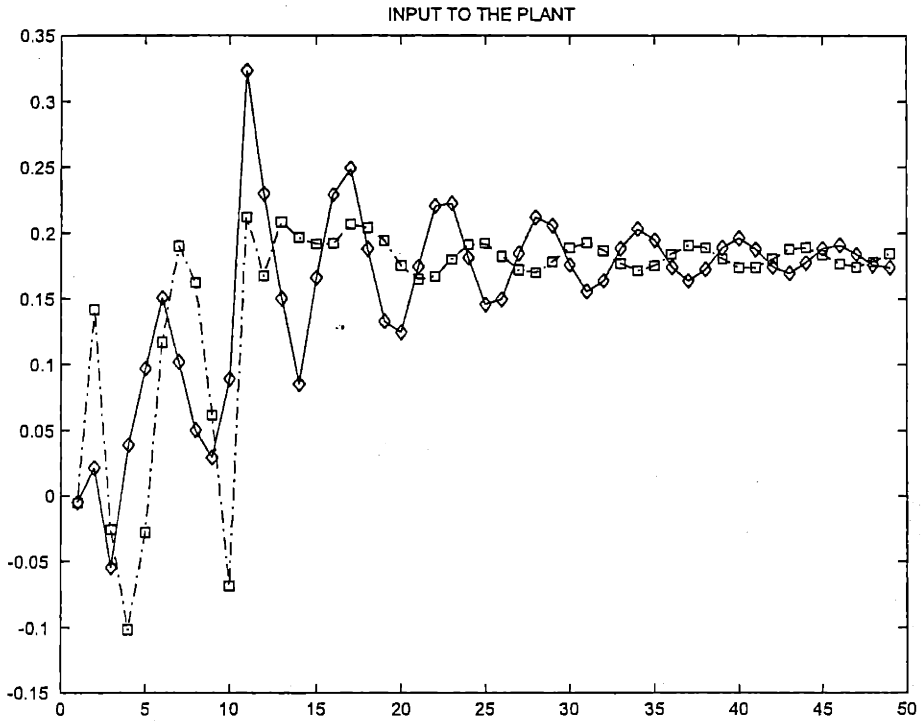


Figure 5-33: (Case 6.2) Input1: Top Draw (diamonds: MPC squares: MSMPC)

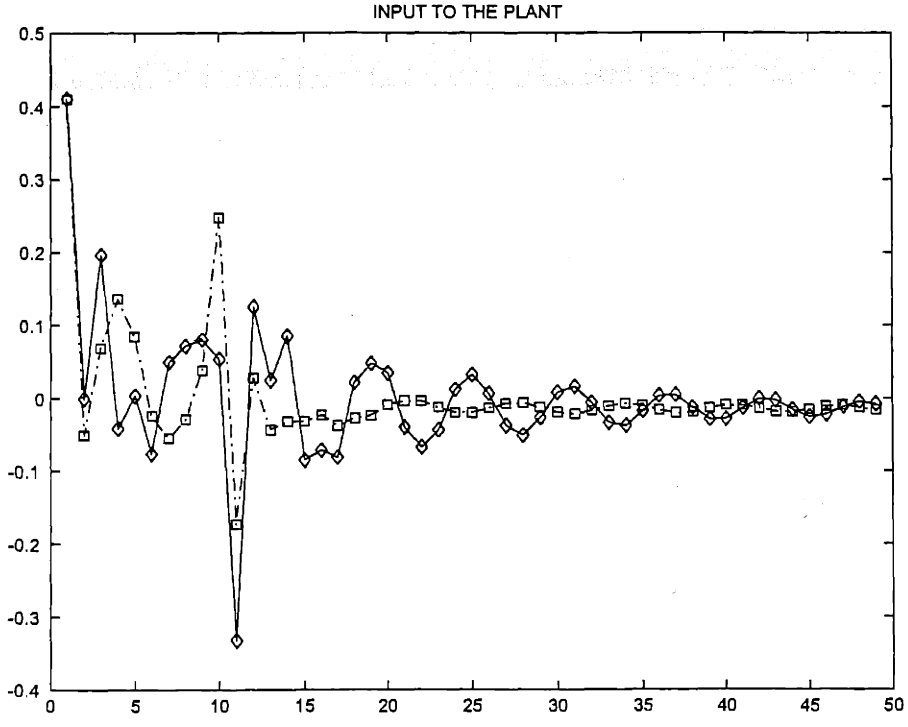


Figure 5-34: (Case 6.2) Input2: Side Draw (diamonds: MPC squares: MSMPC)

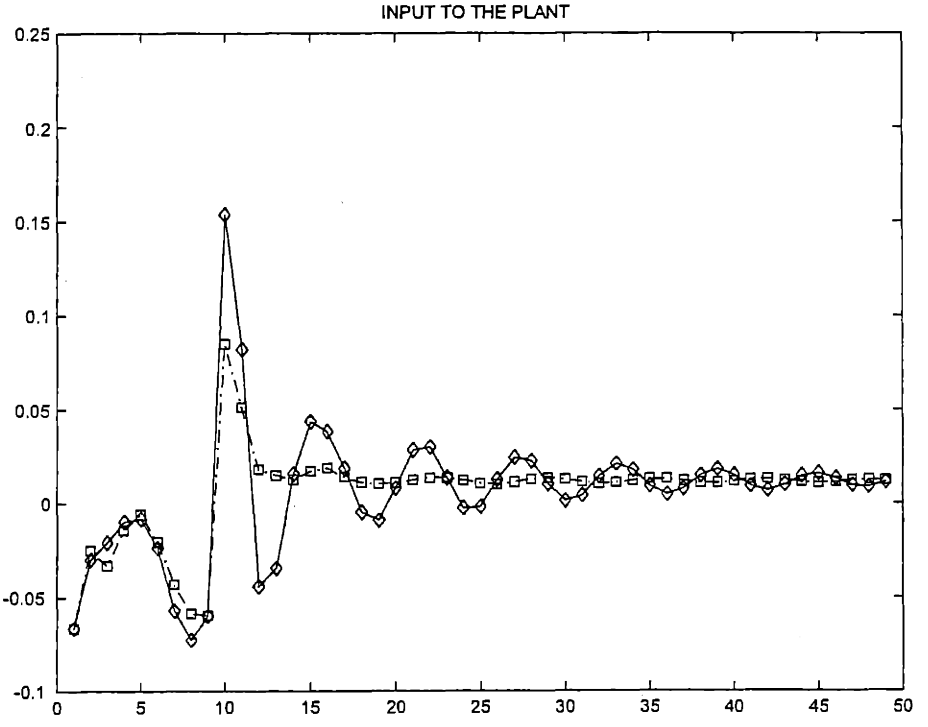


Figure 5-35: (Case 6.2) Input3: Bottom Reflux Duty (diamonds: MPC squares: MSMPC)

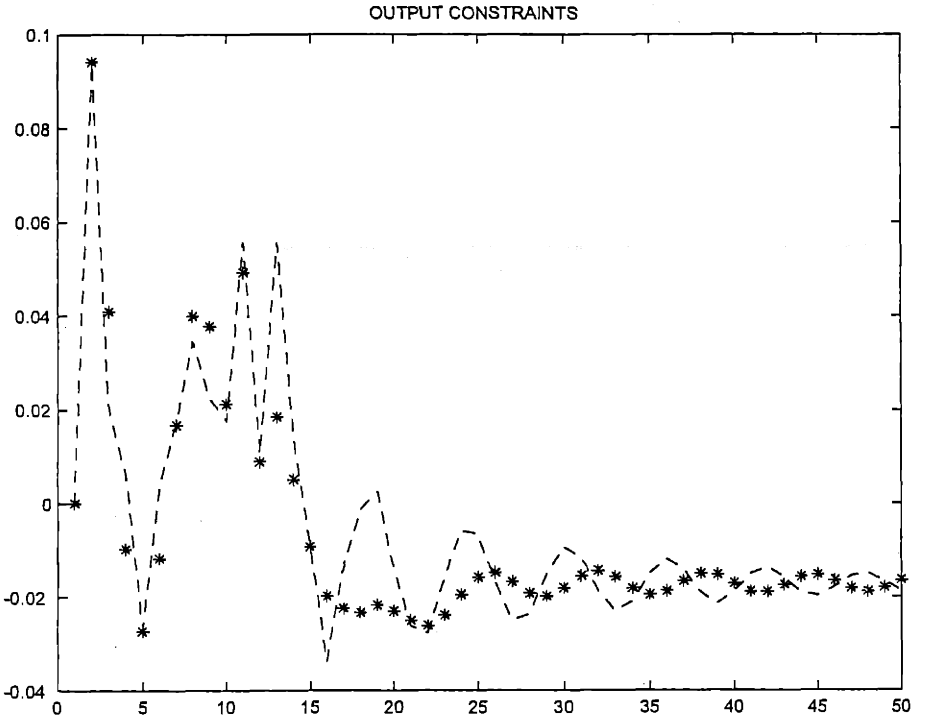


Figure 5-36: (case 6.2) Output1: Top End Point (dashed: MPC stars: MSMPC) (here the constraints are at -0.5 and 0.5, so they are not shown in the figure, all outputs satisfy the constraints)

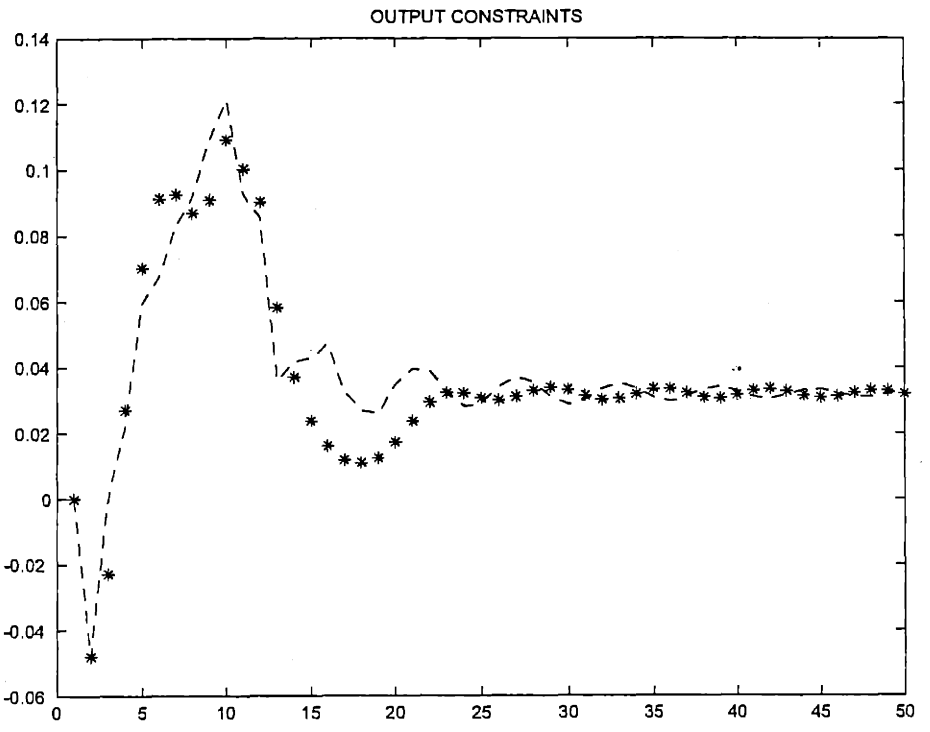


Figure 5-37: (case 6.2) Output2: Side End Point (dashed: MPC stars: MSMPC) (here the constraints are at -0.5 and 0.5, so they are not shown in the figure, all outputs satisfy the constraints)

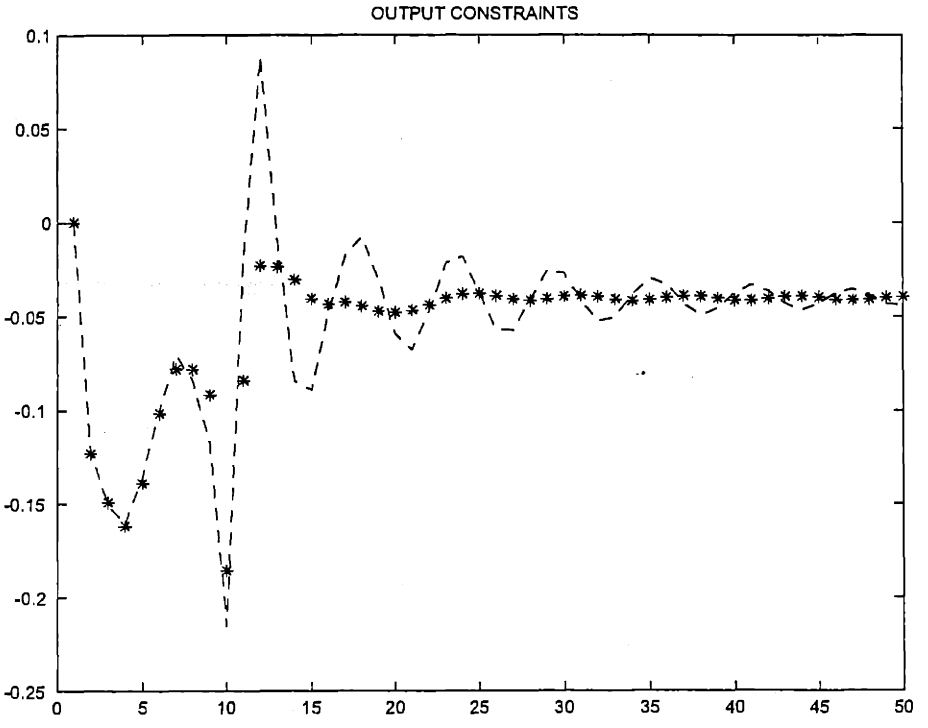


Figure 5-38: (case 6.2) Output3:Bottoms Reflux Temperature (dashed: MPC stars: MSMPC) (here the constraints are at -0.5 and 0.5, so they are not shown in the figure, all outputs satisfy the constraints)

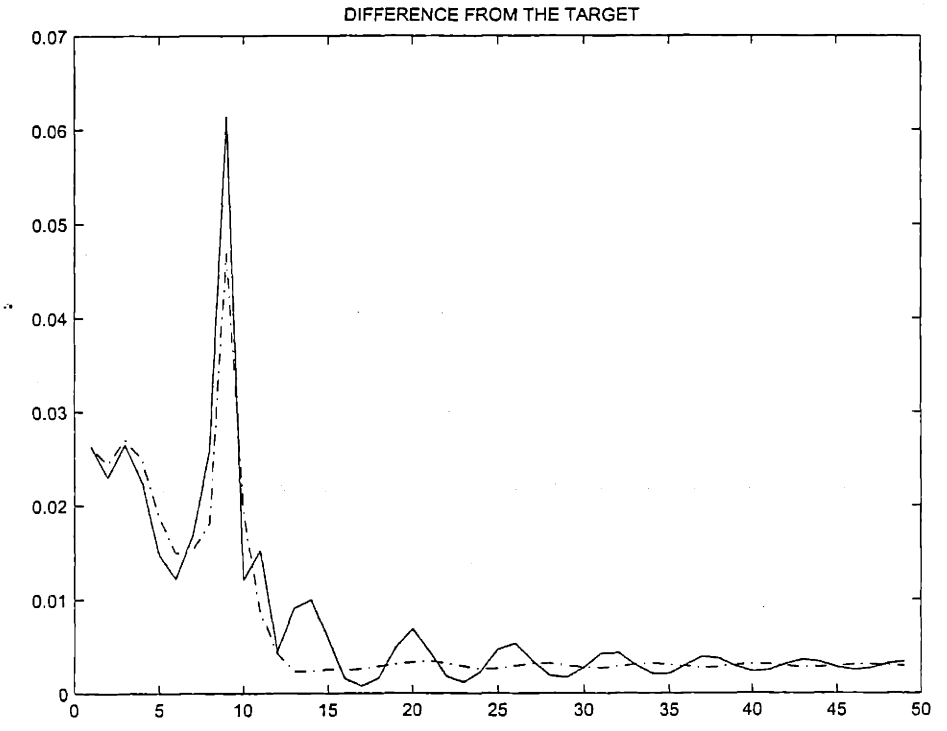


Figure 5-39: (case 6.2) Deviation from the target points (solid line: MPC, dashed line: MSMPC)

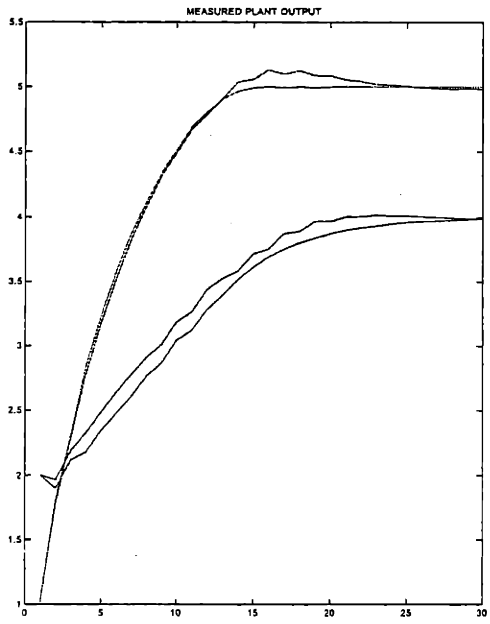


Figure 5-40: (case 7) Comparison of the regular MPC solution (solid) with the Multirate MS-MPC (dashed)

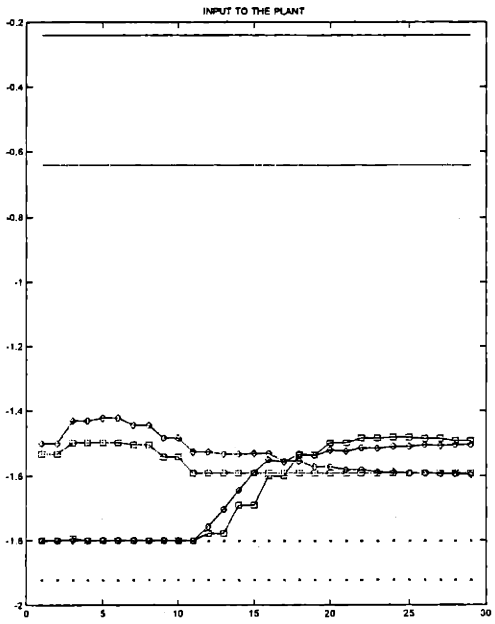


Figure 5-41: (case 7) Comparison of the inputs. Diamonds are Multirate MPC and squares are Multirate MS-MPC. Notice that the u2 starts at a higher value and changes only every second time step.

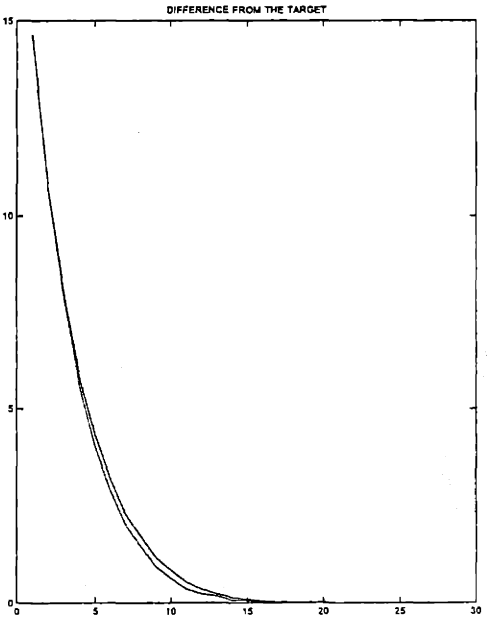


Figure 5-42: (case 7) Comparison of the deviation from the target values.

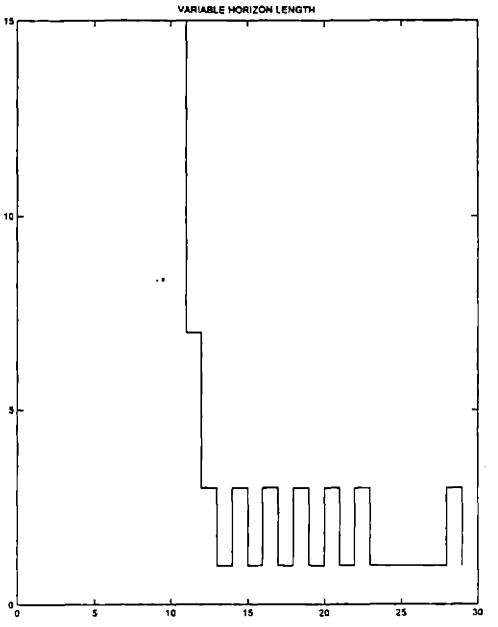


Figure 5-43: (case 7) And finally the comparison showing the variable horizon length. Regular multirate MPC has 8 points fixed control horizon.

Bibliography

[1] M. Dyer. *A Multiscale Approach to State Estimation with Applications in Process Operability Analysis and Model Predictive Control.* PhD thesis, Massachusetts Institute of Technology, 2000.

[2] D. Prett and C. Garcia. *Fundamental Process Control.* Butterworths Series in Chemical Engineers. Butterworths Publishers, 1988.

[3] D. Prett and M. Morari. *Shell Process Control Workshop.* Butterworths Publishers, 1987.

Chapter 6

Conclusions and next steps

Development of technology in instrumentation and computers in an economic environment, where it becomes more and more important to react to market conditions, changing product quality and composition requirements, gives rise to the urgent need for a more integrated approach in operations of processing plants in order to reduce the cost, waste and production time.

The need for a plant wide design and control approach resulted in the development and implementation of Model Predictive Control and other similar advance control algorithms which can incorporate economic cost functions and more importantly constraints, reflecting the physical feasibility of any solution they provide. Although this great framework has very desirable properties it also has some important shortcomings. These shortcomings are described in detail in chapter 2. The common characteristic of these shortcomings is the necessity of solving a problem in time domain, where some of the specifications are in the frequency domain. This issue can be overcome by introducing a new framework where the same problem can be formulated and solved with greater flexibility.

6.1 Contributions

In order to address the issues raised above there was a need for the introduction of a new framework which would capture not only all of the previous formulations, but also add more flexibility to handle the issues of existing formulations.

The separation of time and frequency domains causes most of the issues. Techniques developed in one domain can only be applied in the other if a long list of assumptions are justified. This list of assumptions unfortunately includes also the fact that saturations in input and output variables are neglected. This assumption is against the whole idea of an integrated control scheme which can push a process to its limits to minimize its objective function and maximize its performance therefore.

This thesis introduces a time/scale (frequency) domain framework, developed originally in our group and refined in this thesis to be used in control relevant problems. This new framework allows transformation of linear time domain models into time/scale domain, similar to Fourier transformation, but being localized in both domains, it also allows the transformation of hard constraints on input and/or output variables. This multiscale transformation is achieved by utilizing the wavelet basis functions. Unlike in many other applications of wavelets, our work deals with the symbolic transformation of models rather than just the numerical transformation of time series. As a result we were able to generate models valid at each scale (similar to discretization at different sampling intervals) , and between different scales (multiscale models). This gave us the ability to form relationships between scales and to do operations at different scales in a consistent manner.

Having a framework to define phenomena occurring at different rates or scales we applied this methodology to transform the Model Predictive Control algorithm into multiscale domain. The first and most direct result of this transformation is the additional flexibility in formulation both in the modeling and the optimization phase. The next result is the ability to capture multirate processes naturally without any additional modifications. Blocking and condensing also are simplified to very basic equality constraints. One of the most important impacts of this formulation is, though, the opportunity to design a low complexity parallel algorithm to solve the underlying quadratic programing problem. Reduction of complexity is achieved partially by relaxing the fixed control horizon length. This relaxation has two very important consequences:

1. Reduction in the complexity, if there is no need to have long control horizons (most of the time).

2. Generating an MPC formulation which behaves like an infinite horizon MPC, and thus

has superior stability and feasibility properties.

Especially the latter effect is very desirable. Since the horizon length extends to satisfy the feasibility of the open loop step, the closed loop becomes also stable as a result.

Besides the favorable computational properties the new formulation also provides a natural framework to incorporate frequency domain specifications. Instead of assigning ad hoc weights in the objective function to shape the frequency response, the multiscale formulation separates the objective function into different frequency bands (scales) and they can be penalized differently in order to achieve frequency domain specifications.

The flexibility of the multiscale formulation can also be seen in the dual formulation : the multiscale estimation problem. It incorporates richer depiction of error feedbacks and provides an environment to identify plant-model mismatch at multiple scales and to fuse multirate measurements and control actions naturally.

6.2 Future Work

There are two different areas where further research should be directed to. The first area is the possible improvements on the framework presented in this thesis. The algorithms presented in this thesis are not designed to be optimal. Their main purpose is the demonstration of the flexibilities introduced by the new formulation and solution techniques. The second area is the creation of models directly in multiscale domain rather than the transformation from the time domain. Achieving this goal requires further exploration in the meaning of closure and consistency.

6.2.1 Improving the new framework

Computational: The formulation of MPC in multiscale domain is very flexible and there are many different ways one can explore in order to achieve the best performance in terms of both controller quality and computational efficiency. The work presented in this thesis just demonstrates a subset of possible variations and their solutions.

Great computational performance improvements can be achieved by developing a more efficient Quadratic Program algorithm to solve the optimization problem. The underlying

optimization problem in multiscale domain has a very special form which can be easily exploited. There are two major possibilities for this exploitation:

1. Parallelization of the whole algorithm, and solving it on multiple processors
2. Using a sequential optimization (like in the FINDSCALES algorithm) followed by an efficient update step based on the Karush Kuhn Tucker conditions and local gradients (similar to Laemke's Algorithm).

A problem specific algorithm can accelerate the solution time drastically and allow the incorporation of larger problems with more free variables and constraints.

Adaptive MSMPC: Identification of models used in the MPC formulation is very difficult and expensive. Certain variables in the plant are perturbed by introducing artificial disturbances and the effects of those disturbances are measured in order to identify a model. The conditions surrounding a chemical plant change continuously, though. Equipment get older, raw material quality changes, outside temperature swings, sensors fail, market conditions change. As a result models once identified (under very specific conditions) loose their validity. A solution to this problem is continuous online model identification and reconciliation. Multiscale modeling and MPC formulation proves itself extremely useful for this requirement. Multiscale state estimation algorithm developed by **Matthew Dyer** can be easily extended to identify model parameters and since the data is collected at various scales one can specify which frequency bands are more important to monitor and control. Combining that proposed identification and estimation algorithm with MSMPC one can simultaneously identify the process and adapt the controller accordingly. The adaptive MSMPC would perform drastically better in practical applications where the models are never static.

NMSMPC: Nonlinear MPC is a very active research topic. Linear models are very simple and not enough to express the complex nature of real processes. Theoretical and computational problems prevented the use of nonlinear models extensively in industry. Recent developments in the theoretical aspects of nonlinear MPC makes their use more and more feasible, though. Incorporation of nonlinear systems in MSMPC formulation is a difficult task. There are many reasons for it, starting from a very similar argument like in the Fourier Transformation case: Only linear systems can be transformed into the frequency domain. Multiscale Systems Theory

improves on this limitation and allows the transformation of systems which have hard bounds, i.e. saturations can occur. This is a very limited subset of nonlinear systems and further research should be performed on this topic in order to capture more of the nonlinear systems into the multiscale domain. One way to do it is Multiscale Linearization of nonlinear models. If consistency of models at various scales is kept in mind one can generate models of different orders at each scale and can also define scale dependent constraints to accurately describe the nonlinear system. Knowledge from model reduction, multiscale system theory and nonlinear systems theory should be fused in order to achieve this goal. Once scale dependent linear models are obtained the rest is simple application of the algorithms developed in this thesis.

There are also some minor but interesting side topics requiring further investigation.

Richer Disturbance Models: Classical MPC algorithms use the last deviation between the measured and computed outputs as a constant modeling error in the open loop optimization step. MSMPC goes further and uses the history of the deviations so that more about the real behavior of the plant can be incorporated in the optimization step. Multiscale decomposition of the deviations can be subjected to more elaborate filtering and identification techniques. Thresholding at lower scales can eliminate noise, where thresholding at higher scales (low frequencies) can eliminate bias terms originating from measurement errors. One can go one step further and identify models describing the behavior of the deviations so that the optimization step can use a more reliable disturbance projection to generate more accurate predictions. Of course one cannot expect to do fortune telling but multiscale models derived from the deviation data can easily incorporate events occurring at different time scales (like day-night temperature swings, or a sinusoidal fluctuation in flow rate originating from the nature of a pump).

Fault detection: Using the variable control horizon profile one can deduce when big disturbances hit the system. The disturbances can be in the form of modeling error, i.e. a change in the plant (equipment failure, change in the operating regime, unmodeled reactions or products in the process) or in the form of measurement errors (sensor failure) or of course it can be a totally external disturbance. A change in the horizon length can initiate a more elaborate algorithm to determine the real source of the disturbance and the results can be incorporated into MSMPC to improve its performance.

Soft Constraints: Incorporation of soft constraints in MPC formulation eliminates the

infeasibility originating from tight output constraints. The same modification can be done to MSMPC algorithm, too. Although MSMPC can extend its horizon length to eliminate such infeasibilities, it also increases the computational load. Soft constraints have additional benefits in improving robustness and in multiscale formulation they can be used as an extended tool to shape the frequency characteristics of the outputs.

6.2.2 Modeling in Multiscale Domain

A more ambitious part of the future work is in the area of developing models directly in the multiscale domain. As mentioned before in chapter 3, any valid multiscale model should satisfy certain criteria. The most important two of them are closure and consistency. Closure is required if there is a real model which should be valid at the physical level. Closure bounds the tree representing the multiscale model.

Consistency is required if the underlying system is a causal physical system. An image has no causality and as result a multiresolution model describing that image can have different models at each scale (**Willsky**).

It is not obvious how to design models directly in the multiscale domain considering the requirements described above. There are some relaxations though; first of all if there is no preset physical level, closure requirement is not needed anymore. The lack of a physical level is difficult to grasp, but considering the fact that many phenomena occur at different rates (different scales, characteristic times etc.) it is the most natural representation. Each level will have a physical meaning but there will be no single and compact representation in the time domain, because there will be no single scale which can be called the "time level".

Such a modeling environment will still require a certain amount of consistency. Inputs at a node should still generate states at the next temporal node and the mother node at the next scale, but it is possible to have models of different order at different scales, which are still consistent. Especially at higher scales (low frequencies) fast components (short characteristic times) disappear and corresponding states become constant. Consequently the dynamic model describing those fast components becomes unnecessary at higher scales. A reduced model can be used to describe the remaining system. This procedure further reduces the amount of complexity and can be called "compressing the dynamic model".

This work combined with the introduction of nonlinear systems can lead to a new paradigm where micro and macro scale models can be combined consistently for any physical system.

Appendix A

LQR

Given the system dynamics

$$\dot{x} = Ax(t) + Bu(t) \tag{A.1}$$

$$x(t=0) = x_0 \tag{A.2}$$

with $x(t) \in \Re^n$ and $u(t) \in \Re^m$ along with a linear combination of states to keep small

$$z(t) = Cx(t) \tag{A.3}$$

with $z(t) \in \Re^l$. We define a quadratic cost functional

$$J = \int_0^\infty \left[z^T(t) z(t) + u^T(t) Ru(t) \right] dt \tag{A.4}$$

in which the size of the states of interest, $z(t)$, is weighted relative to the amount of control action in $u(t)$ through the weighting matrix R.

If the following assumptions hold:

1. The entire state vector $x(t)$ is available for feedback.
2. $\begin{bmatrix} A & B \end{bmatrix}$ is stabilizable and $\begin{bmatrix} A & C \end{bmatrix}$ is detectable.
3. $R = R^T > 0$

Then

1. The Linear Quadratic Controller is the unique, optimal, full state feedback control law

$$u(t) = -Kx(t) \text{ with } K = R^{-1}B^T S \tag{A.5}$$

that minimizes the cost, J, subject to the dynamic constraints imposed by the open-loop dynamics in (A.1).

2. S is the unique, symmetric, positive semi-definite solution to the algebraic Riccati equation

$$SA + A^T S + C^T C - SBR^{-1}B^T S = 0 \tag{A.6}$$

3. The closed-loop dynamics arrived at by substituting (A.5) into (A.1)

$$\dot{x}(t) = [A - BK]\, x(t) \tag{A.7}$$

are guaranteed to be asymptotically stable.

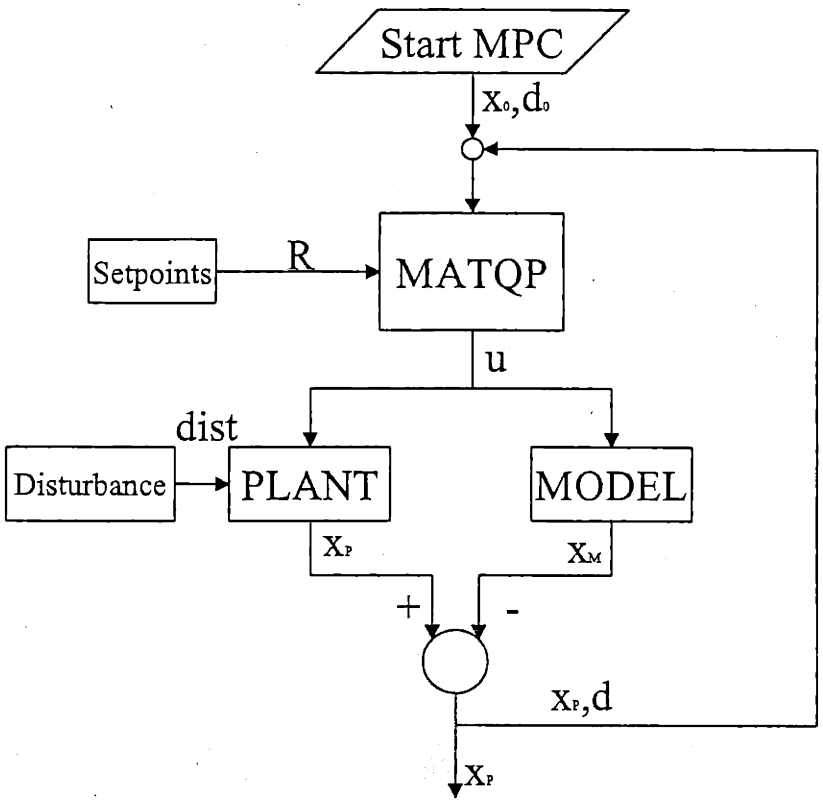
4. The minimum value of the cost J in (A.4) is $J = x_0^T S x_0$.

Appendix B

The MATLAB Code for MSMPC Algorithm

This section lists the major parts of the code underlying the case studies. One should note that this code is not written for efficiency, but to demonstrate the properties of MSMPC. The code can be optimized drastically. Another note is that some very simple functions are omitted here; if needed they can be found at **http://lispe.mit.edu/Students/Orhan/**.

B.1 Flowcharts of the Program



Flowchart of the Code for the Classical MPC Formulation

B.2 The Main Code: TESTSYS

```
clear
%Define the regular MPC problem
inputsys;
%The disturbance section
for j=1:len
dist(:,j)=inp(:,j);
end
%The initial conditions
xmpc=x1;
xstart=xfin;
Xmodel=[xmpc];
Xmodel1=[xmpc];
Xplant=[xmpc];
Umodel=[];
OBJ=[];
```

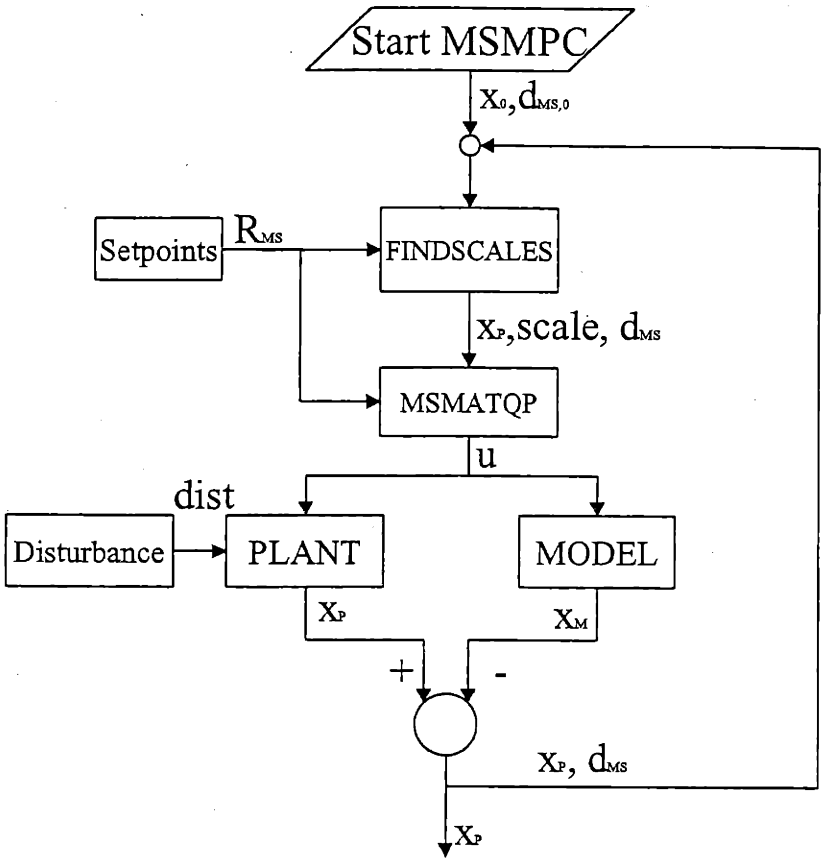


Figure B-1: Flowchart of the Code for the Multiscale MPC Formulation

```
xp=xmpc;
UUP=[];
ULO=[];
ERR=[0];
flops(0)
%Computation at each time step
for k=1:len-1
%Measure the disturbance by taking the
%difference between the output measurement and
%model prediction
dist(:,k)=xp-xmpc;
%Do the optimization over a fixed horizon and
%determine the input and predicted output
[u1,xmpc,error]=matqp(xp,dist(:,k));
ERR=[ERR,error];
%Predicted states
```

```
Xmodel=[Xmodel,xmpc];
Xmodel1=[Xmodel1,xmpc+dist(:,k)];
%Real Plant output
xp=AP*xp+BP*u1+dist(:,k+1);
Xplant=[Xplant,xp];
Umodel=[Umodel,u1];
%Compare the distance from the ultimate targets
OBJ=[OBJ,(xstart-xp)'*(xstart-xp)];
UUP=[UUP,uup];
ULO=[ULO,ulo];
end
flops
%Plotting section
%regular_plot
regularmultiple
%MS-MPC section
%Defining the problem
inputsys;
%Defining the disturbances
distms=zeros(n*len,1);
for j=1:len
distms(in(j,n))=inp(:,j);
end
%The initial conditions
xmsmpc=x1;
xstart=xfin;
Xmodelms=[xmsmpc];
Xmodelms1=[xmsmpc];
Xplantms=[xmsmpc];
Xprevms=[xmsmpc];
Umodelms=[];
OBJms=[];
Uprevms=[];
scale=1;
SC=[];
xp=xmsmpc;
distu=[];
%Computations at each time step on multiple scales
flops(0)
for k=1:len-1
%Measure the disturbance by taking the
%difference between the output measurement and
%model prediction
distms(in(k,n))=xp-xmsmpc;
distu=[distu,distms(in(k,n))];
```

```
%Determine the number of scales: variable horizon
[scale,uprev,D]=findscales2(xp,scale,distms(1:n*k));
Uprevms=[Uprevms,uprev];
%Having the scales create a qp over that
scale=limit1*const+(1-const)*scale;
SC=[SC,2^scale-1];
[u1,xmsmpc]=msmatqp(scale,xp,D);
Xmodelms=[Xmodelms,xmsmpc];
Xmodelms1=[Xmodelms1,xmsmpc+distms(in(k,n))];
%Real plant output
xp=AP*xp+BP*u1+distms(in(k+1,n));
Xplantms=[Xplantms,xp];
Umodelms=[Umodelms,u1];
%Compare the performance by computing the
%the difference from the target values
OBJms=[OBJms,(xstart-xp)'*(xstart-xp)];
end
flops
%multiscale_plot
multimultiple
output_violation_plot
```

B.3 Subprogram I: matqp

```
function [u,xmodel,error]=matqp(xplant,dist)
moder=dist;
%QP for MPC
model;
a=xfin;
b=xfin-xplant;
%Reference Path
R=[];
for i=1:L
R(in(i,n))=a-Cr^i*b;
end
R1=R;
D=[B];
C=[A];
Er=[moder];
VLB=[ulo];
VUB=[uup];
D1=[eye(n)];
Ymin=[yl];
Ymax=[yu];
for i=1:L-1
```

```
Z=D(in(i,n),:);
Y=D1(in(i,n),:);
D=[D zeros(i*n,size(B,2));[A*Z,B]];
D1=[D1 zeros(i*n,n);[A*Y,eye(n)]];
C=[C;A^(i+1)];
Er=[Er;moder];
VLB=[VLB;ulo];
VUB=[VUB;uup];
Ymin=[Ymin;yl];
Ymax=[Ymax;yu];
end
bout=C*xplant+D1*Er;
Rm=R1'-bout;
H=D'*D;
f=-D'*Rm;
I=eye(size(Cy,1)*L);
%Adding the output constraints
Coutput=[];
[xx,yy]=size(Cy);
for i=1:L
Coutput(in(i,xx),in(i,yy))=Cy;
end
Dout=Coutput*D;
BB=[Ymax-Coutput*bout;-Ymin+Coutput*bout];
AA=[[Dout,-I*slack];-[Dout,-I*slack]];
Hbar=[H,zeros(size(H,1),size(I,2));zeros(size(I,1),size(H,2)),I*pen];
fbar=[f;zeros(size(I,1),1)];
U=qp(Hbar,fbar,AA,BB,VLB,VUB);
uuu=length(U);
error=0;
S=U(L*n+1:uuu);
for i=1:L
error=error+S(in(i,xx))'*S(in(i,xx));
end
u=U(in(1,size(B,2)));
xmodel=A*xplant+B*u;
```

B.4 Subprogram II: findscales

```
function [scale,uprev,D]=findscales2(x1h,scale1,dist)
%The object of this M file is to determine the length of the horizon
%and a initial guess for the inputs. Besides it produces the bounds
%for the blocked variables
%The data section, define the problem
model;
```

```
%Create the MS coeficients
[MSA,MSB]=coef(A,B,limit1);
[MSA,MSD]=coef(A,eye(n),limit1);
%-----------------------------------------
xin=x1h;
nn=length(dist);
%-----------------------------------------
a=xfin;
b=xfin-xin;
%The optimization step:
stop = 0;
scale= 0;
user=zeros(n,1);
ublock=zeros(n,1);
xl=x1h;
X=[xl];
mim=scale1;
%Cumulative block bounds
AB=[];
bb=[];
for j=1:scale1
c1(:,:,j)=zeros(size(A));
c2(:,:,j)=zeros(size(A));
c3(:,:,j)=zeros(size(A));
c4(:,:,j)=zeros(size(A));
for i=1:2^(j-1)
c1(:,:,j)=c1(:,:,j)+i*A^(2^(j-1)-i);
c2(:,:,j)=c2(:,:,j)+A^(i-1);
end
for i=2^(j-2):2^(j-1)-1
c3(:,:,j)=c3(:,:,j)+A^i;
end
for i=1:2^(j-2)-1
c4(:,:,j)=c4(:,:,j)+(i*(eye(size(A))+A^(2^(j-2)))*A^i);
end
c1(:,:,j)=inv(B)*inv(c1(:,:,j));
c2(:,:,j)=c1(:,:,j)*sqrt(2)^(j-1)*c2(:,:,j)*B;
c3(:,:,j)=c1(:,:,j)*sqrt(2)^(j-2)*c3(:,:,j)*B;
c4(:,:,j)=c1(:,:,j)*c4(:,:,j)*B;
end
D2=[];
C2=[];
D=disthist(dist,limit1);
while stop < 1
scale=scale+1;
```

```
if scale>mim
for j=mim+1:scale
c1(:,:,j)=zeros(size(A));
c2(:,:,j)=zeros(size(A));
c3(:,:,j)=zeros(size(A));
c4(:,:,j)=zeros(size(A));
for i=1:2^(j-1)
c1(:,:,j)=c1(:,:,j)+i*A^(2^(j-1)-i);
c2(:,:,j)=c2(:,:,j)+A^(i-1);
end
for i=2^(j-2):2^(j-1)-1
c3(:,:,j)=c3(:,:,j)+A^i;
end
for i=1:2^(j-2)-1
c4(:,:,j)=c4(:,:,j)+(i*(eye(size(A))+A^(2^(j-2)))*A^i);
end
c1(:,:,j)=inv(B)*inv(c1(:,:,j));
c2(:,:,j)=c1(:,:,j)*sqrt(2)^(j-1)*c2(:,:,j)*B;
c3(:,:,j)=c1(:,:,j)*sqrt(2)^(j-2)*c3(:,:,j)*B;
c4(:,:,j)=c1(:,:,j)*c4(:,:,j)*B;
end
mim=scale;
end
[Rsl,Rsr,Rw]=rof(x1h,scale);
el=(Rsl-xl)/sqrt(2);
Al= MSA(in(scale,n),:);
Bl= MSB(in(scale,n),:);
DL= MSD(in(scale,n),:);
dRm= Rw-(eye(size(A))-Al)/sqrt(2)*xl-el+DL*D(in(scale,n))/sqrt(2);
f = (sqrt(2)*dRm'*Bl)';
H = Bl'*Bl;
VUP = sqrt(2)^(scale-1)*uup;
VLO = sqrt(2)^(scale-1)*ulo;
% Create the output constraints
Aout= [Cy*Bl;-Cy*Bl];
bout= [yu*sqrt(2)^(scale-1)-Cy*(Al*xl+DL*D(in(scale,n)));...
-yl*sqrt(2)^(scale-1)+Cy*(Al*xl+DL*D(in(scale,n)))];
% Create the additional blocking bound
d=-c3(:,:,scale)*user(:,scale)-c4(:,:,scale)*ublock(:,scale);
AConst=[c2(:,:,scale);-c2(:,:,scale)];
bconst=[uup-d;-ulo+d];
uscale=qp(H,f,[AConst;Aout],[bconst;bout],VLO,VUP);
ub=c2(:,:,scale)*uscale+d;
xr=Al*xl+Bl*uscale+DL*D(in(scale,n));
xl=((eye(size(A))+Al)*xl+Bl*uscale+DL*D(in(scale,n)))/sqrt(2);
```

```
X=[X,xl];
user=[user,uscale];
ublock=[ublock,ub];
if norm(abs(xr-Rsr))<0.01
stop=1;
end
if scale>limit1-1
stop=1;
end
end
uprev=user(:,2);
```

B.5 Subprogram III: msmatqp

```
function [u1,xres]=msmatqp(scale,xc,DF)
model;
%A) Objective function
D=disturb(DF,scale);
[H,f]=objective(scale,xc);
%B) The bounds
[A5,b5]=bounding(scale,xc,D);
%C) The Blocking
[A4,b4]=blocking(scale,xc,D);
N=size(A4,1);
%D) Output Constratints
[A6,b6]=outputconst(scale,xc);
AAA=[A6;A5];
BBB=[b6;b5];
AB=[A4*vblock;AAA*wbound];
BB=[b4*vblock;BBB*wbound];
dx=qp(H,f,AB,BB,-ones(size(H,1),1)*inf,ones(size(H,1),1)*inf,...
zeros(size(H,1),1),N*vblock);
[A2,b2]=u_to_dx(scale,xc,D);
%[A1,b1]=x_to_dx(scale,n,xc);
u=A2*dx+b2;
%xx=A1*dx+b1;
u1=u(in(1,n));
xres=A*xc+B*u1;
```

B.5.1 Subsubprogram I: msmatqp-disturb

This function creates the multiscale disturbance structure from the history of deviations.

```
function D1=disturb(D,scale)
model;
```

```
DPSEUDO=D;
fd=D(in(1,n));
D=[];
for i=1:scale
D=[D;fd*sqrt(2)^(i-1)];
end
if dstructure==1
D=DPSEUDO;
end
d1=[zeros(n,1);D];
db1=zeros(n,1);
for j=1:scale
c1(:,:,j)=zeros(size(A));
c2(:,:,j)=zeros(size(A));
c3(:,:,j)=zeros(size(A));
c4(:,:,j)=zeros(size(A));
for i=1:2^(j-1)
c1(:,:,j)=c1(:,:,j)+i*A^(2^(j-1)-i);
c2(:,:,j)=c2(:,:,j)+A^(i-1);
end
for i=2^(j-2):2^(j-1)-1
c3(:,:,j)=c3(:,:,j)+A^i;
end
for i=1:2^(j-2)-1
c4(:,:,j)=c4(:,:,j)+(i*(eye(size(A))+A^(2^(j-2)))*A^i);
end
c1(:,:,j)=inv(c1(:,:,j));
c2(:,:,j)=c1(:,:,j)*sqrt(2)^(j-1)*c2(:,:,j);
c3(:,:,j)=c1(:,:,j)*sqrt(2)^(j-2)*c3(:,:,j);
c4(:,:,j)=c1(:,:,j)*c4(:,:,j);
end
for i=1:scale
db=c2(:,:,i)*d1(in(i+1,n))-c3(:,:,i)*d1(in(i,n))-c4(:,:,i)*db1(:,i);
db1=[db1,db];
end
db=db1(:,2:scale+1);
dib=[];
[As,Bs]=coef(A,B,scale);
for i=1:scale
for j=1:2^(i-1)
dib=[dib;db(:,i)];
end
end
DB=modhat(dib,A,eye(n));
P=[];
```

```
%Now extract all left nodes from this matrix
for i=1:scale
for j=1:2:2^(scale-i+1)
P=[P;DB(in(j,n),i)];
end
end
D1=P;
```

B.5.2 Subsubprogram II: msmatqp-objective

This function generates the objective function parameters.

```
function [H,f]=objective(scale,xc)
model;
[A1,b1]=x_to_dx(scale,n,xc);
H=eye((2^(scale)-1)*n);
t=1;
for i=1:scale
H(in(t,n),in(t,n))=eye(n)*(2^(scale-i)+1);
k=t+2^(scale-i);
for j=1:scale-i
H(in(t,n),in(k,n))=eye(n)*sqrt(2)^(2*(scale-i+1)-j);
k=k+2^(scale-j-i);
end
t=t+2^(scale-i);
end
H=(H+H')/2;
R=[];
a=xfin;
b=xfin-xc;
for i=1:2^(scale)
R(in(i,n))=a-Cr^(i-1)*b;
end
[MSR,MSDR]=msref(R,n);
dR=[];
for i=1:scale
dR=[dR;MSDR(1:n*2^(scale-i),i)];
end
f=-H*dR;
```

B.5.3 Subsubprogram III: msmatqp-bounding

This function generates the input bounds in terms of δx's.

```
function [A5,b5]=bounding(scale,xc,D)
```

```
model;
[A3,b3]=u_to_u(scale,xc,D);
ULO=[];
UUP=[];
%for i=1:scale
% Making changes for just bounding inputs on
% the timeline
for j=1:2^(scale)-1
ULO=[ULO;ulo];
UUP=[UUP;uup];
end
A31=A3(1:(2^scale-1)*n,:);
b31=b3(1:(2^scale-1)*n,:);
A5=[A31;-A31];
b5=[UUP-b31;-ULO+b31];
```

B.5.4 Subsubprogram IV: msmatqp-blocking

This function generates the equality constraints for blocking.

```
function [A4,b4]=blocking(scale,x01,D)
[A3,b3]=u_to_u(scale,x01,D);
n=length(x01);
bl=[eye(n),-eye(n)];
x=length(b3);
y=2^scale-1-scale;
B=zeros(y*n,x);
f1=1;
f2=0;
for i=1:scale-1
for j=1:2^i-1
f2=f2+1;
B(in(f1,n),f2*n+1:(f2+2)*n)=bl;
f1=f1+1;
end
f2=f2+1;
end
A4=B*A3;
b4=-B*b3;
```

B.5.5 Subsubprogram V: msmatqp-outputconst

This function generates the output constraints.

```
function [A6,b6]=outputconst(scale,xc)
```

```
model;
[A1,b1]=x_to_dx(scale,n,xc);
[xx,yy]=size(Cy);
count=1;
count2=1;
YU=[];
YL=[];
Cout=[];
for i=1:scale
for j=1:2^(scale-i)
Cout(count:count+xx-1,count2:count2+yy-1)=Cy;
YU=[YU;yu*sqrt(2)^(i-1)];
YL=[YL;yl*sqrt(2)^(i-1)];
count2=count2+yy;
count=count+xx;
end
end
A1r=A1;
[rr,ss]=size(A1);
for i=1:rr
if A1r(i,i)==0
A1r(i,i)=-sqrt(2);
else
A1r(i,i)=-sqrt(2)/2;
end
end
AR=Cout*A1r;
AL=Cout*A1;
b=Cout*b1;
bR1=YU-b;
bR2=-YL+b;
AL=AL(xx+1:2^(scale-1)*xx,:);
b=b(xx+1:2^(scale-1)*xx);
AR=AR(1:2^(scale-1)*xx,:);
bR1=bR1(1:2^(scale-1)*xx);
bR2=bR2(1:2^(scale-1)*xx);
YU=YU(xx+1:2^(scale-1)*xx);
YL=YL(xx+1:2^(scale-1)*xx);
bL1=YU-b;
bL2=-YL+b;
A6=[AL;-AL;AR;-AR];
b6=[bL1;bL2;bR1;bR2];
```

B.5.6 Subsubprogram VI: msmatqp-xtodx

This function provides the matrix transformation between the states and wavelet coefficients on the dyadic tree.

```
function [A1,b1]=x_to_dx(scale,n,x01)
%the function part that handles the computation of the creation of A and b
A1=zeros((2^scale-1)*n,(2^scale-1)*n);
f=1;
for i=1:scale
t=1;
for j=f:2^(scale)-1
BL=blockmatrix(i,n);
[x,y]=size(BL);
A1(t:t+x-1,in(j,n))=BL;
t=t+n*2^(i-1);
end
f=f+2^(scale-i);
end
b1=abs(A1(:,in(2^(scale)-1,n)));
c=-A1(1:n,:)*sqrt(2)^scale;
A1=A1+b1*c;
b1=b1*sqrt(2)^scale*x01;
```

B.5.7 Subsubprogram VII: msmatqp-utodx

This function forms the relationship between the inputs and wavelet coefficients on the tree.

```
function [A2,b2]=u_to_dx(scale,xc,D)
model;
[A1,b1]=x_to_dx(scale,n,xc);
[As,Bs]=coef(A,B,scale-1);
I=[];
for j=1:scale
I=[I;eye(n)];
end
AW=1/sqrt(2)*(I-As);
AS=1/sqrt(2)*(I+As);
BS=1/sqrt(2)*Bs;
BW=-BS;
A11=zeros((2^scale-1)*n,(2^scale-1)*n);
B11=zeros((2^scale-1)*n,(2^scale-1)*n);
D11=zeros((2^scale-1)*n,(2^scale-1)*n);
f=1;
for i=1:scale
```

```
for j=1:2^(scale-i)
AL=AW(in(i,n),:);
BL=BW(in(i,n),:);
DL=BS(in(i,n),:)*inv(B);
A11(in(f+j-1,n),in(f+j-1,n))=AL;
B11(in(f+j-1,n),in(f+j-1,n))=BL;
D11(in(f+j-1,n),in(f+j-1,n))=DL;
end
f=f+2^(scale-i);
end
I=eye((2^scale-1)*n);
A2=inv(B11)*(I-A11*A1);
b2=-inv(B11)*(A11*b1-D11*D);
```

B.5.8 Subsubprogram VIII: msmatqp-utou

This function generates inputs on the right nodes of the tree from the left node inputs.

```
function [A3,b3]=u_to_u(scale,x01,D)
model;
[A2,b2]=u_to_dx(scale,x01,D);
[As,Bs]=coef(A,B,scale-1);
a1=[];
a2=[];
a3=[];
for i=1:scale-1
a1=[a1;-inv((eye(n)+As(in(i,n),:))*Bs(in(i,n),:))*As(in(i,n),:)*Bs(in(i,n),:)];
a2=[a2;-inv((eye(n)+As(in(i,n),:))*Bs(in(i,n),:))*Bs(in(i,n),:)];
a3=[a3;sqrt(2)*eye(n)];
end
% Creation of D's
% start with the countdown
D=[];
Z=1;
s1=2^scale-3;
for i=1:scale-1
s=scale-i;
D=eye((2^scale-2+2^i-i)*n);
block=modaveblock(a1(in(s,n),:),a2(in(s,n),:),a3(in(s,n),:),i);
[x,y]=size(block);
[x1,y1]=size(D);
for j=1:(2^i-1)
D=[D(1:(s1+2*(j-1))*n,:);zeros(x,(s1-2+j)*n),block,...
zeros(x,y1-y-(s1-2+j)*n);D((s1+2*(j-1))*n+1:x1,:)];
[x1,y1]=size(D);
```

```
end
Z=D*Z;
s1=s1-2^(i+1);
end
% Now the u_to_dx
A3=Z*A2;
b3=Z*b2;
```