Constrained Predictive Control Methods
Theory and Practice

First Year Report

Mihai Huzmezan
Pembroke College
Department of Engineering
University of Cambridge
Cambridge, CB2 1PZ, England

Supervisor: Dr. Jan Maciejowski

November 30, 1995

Contents

Preface

I would like to thank my supervisor Dr. Jan Maciejowski without whose help I could not have made the transition from avionics and classical control research into the field of H_∞ and Constrained Predictive Control Theory. I would also like to thank my previous supervisor, professor Vlad Ionescu, for the freedom he allowed me to choose the direction I was to take, and for his great support during the M.Sc. and the Ph.D. research started in Romania in 1994.

In particular I would like to thank my family, my friends and former colleagues for their emotional support. I am grateful as well to my sponsor "SM Invest Holding" for all financial support. Here companies like "Tarom SA" could be also mentioned for the free travel provided. Since joining the group I have enjoyed the great company and help of all its members.

This report is intended to provide a survey of preliminary work performed by the author and a plan for his intended Ph.D. research.

Chapter 1

Introduction

1.1 "The Decision Making Process"

Most of the economic activities that involve complex systems have a structure that can be described in terms of hierarchical layers. Hierarchical layers of "The Decision Making Process" are becoming more and more integrated. One that studies such systems may discover an overlapping structure presented in a simplified way in figure 1.1. This integration provides much flexibility and independence of action at all levels of this process. Each layer plays a unique but complementary role in automating "The Decision Making Process" and allows rapid reaction to changes. So, one technology cannot be effective and operable without the other and, in addition, the efficiency of the whole approach is possible when all components of the chemical process or flight system, for example, are part of the optimized system.

- **Plant:**

  In our context plant means physical systems. Real plant is the one which really flies or produce chemical components "out there".

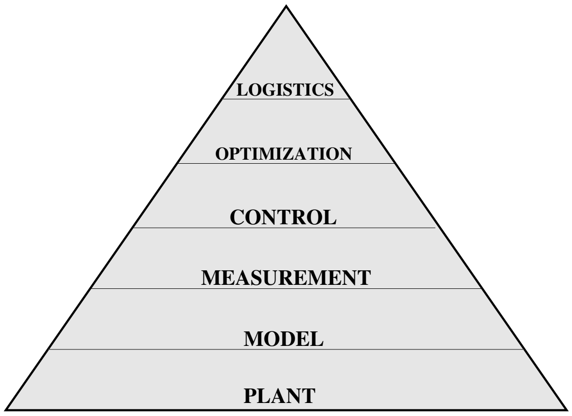

Figure 1.1: Hierarchical layers of "The Decision Making Process"

- **Model:**

  In this report the word "model" is associated with various meanings, so, it is necessary to distinguish among them:

  Ideal physical model is obtained by decomposing the real plant into ideal building blocks such as resistors, masses, Newtonian fluids, beams etc.

  Ideal mathematical model is obtained by applying natural laws to the above model which usually result nonlinear partial differential equations.

  Reduced mathematical model is obtained from an ideal mathematical model by liniarization around an equilibrium point and it is formed usually of rational transfer functions or state space descriptions. In such a model time delays are included as extra states.

  Problems occur when the mathematical model is supposed to approximate the real plant and, at that time, is necessary to emphasize the role of uncertainty. Uncertainty here means not being able to predict the output of the system. Even if we know the input, we are uncertain about the system. Generally uncertainty arises from two main sources: unknown or unpredictable inputs such as noise, disturbance etc. and unknown dynamics.

  The model we should provide in order to solve control problems is one that predicts the input/output response in such a way that we can use it for designing a control system and be confident about the resulting design will work on a real plant. Various models that suit our needs are available today.

  The model should be sufficiently complex to be able to simulate all components susceptible to failure but as simple as possible in order to improve speed of control. A certain complexity of the model is required in the case of using the system for training and certification.

  Mathematical models used for design are often *finite dimensional, linear and time invariant.* Main reason for using such models is that they are simplest for treating fundamental issues and the resulting design techniques work remarkably well for a large class of engineering problems (partly because lots of systems are built to be as close as possible to LTI in order to be easily controlled).

  Most *mathematical models* are obtained from *science* or *experimental data*:

  Models from science are derived by applying the laws of physics, chemistry etc. and making idealizing assumptions. The coefficients of the partial differential equations will depend on constants that can be measured. This kind of modelling will work for aerospace and electrical systems but other systems are difficult to model this way either because they are too complex or because their governing laws are unknown.

  Models from experimental data are obtained by performing experiments on the physical system. Ideally the model will cover data, which means that it should be capable of producing a determined input-output pair. A useful model in control design must have features like: being deterministic and having uncertainty built in it explicitly.

- **Measurements:**

  Instrumentation is used in order to monitor the process. Measurements include inputs, outputs and even states when available such as: speed, angle of attack, temperature, pressure, flow, level, composition.

- **Control:**

  Control represents the manipulation of plant (process) degrees of freedom for the satisfaction of various operating criteria. Synthesis, part of the design process, as well as real implementation represent important challenges. Control operation, part of "The Decision Making Process" is achieved using either analog controllers or digital signal processing computers. Such computers with large CPU capabilities are required in order to ensure real time processing. A controller is designed and implemented in "The Decision Making Process" following the typical scenario:

  Specify stability and performance requirements for the system

  Modelling the system imply a study of the plant and a careful choice of sensors and actuators used.

  Simplification of the model is a necessary step in order to have fast control algorithms and fairly simple controllers.

  Analyse the resulting model in order to determine its properties

  Design a controller to meet the specifications. It is possible that a tradeoff between specification could appear (i.e. tradeoff between sensor noise, disturbance rejection and input signal level).

  Simulating the complete system (plant plus controller) either on a computer or on a pilot plant

  Specifications impossible to meet mean to repeat few of above steps, so, iterative nature of the design is very important.

  Choice of hardware and software in order to implement the controller.

  Commissioning the controller if it is necessary

  It is desirable to minimize the amount of on-line tuning performed, a reasonable way of doing this being to use a representation of the process that includes uncertainty as well as statistical information on disturbances. Such a use of uncertainty description may allow the designer to evaluate various tradeoffs between performance, model accuracy and stability, and therefore determine a priori tuning parameters. Redefinition of some performance criterion and updating the model may be necessary. Adaptive control is connected with the idea of updating the model. Such updates could change the model in such a way that tuning on-line will be inevitable. From the design point of view, there is a strong connection between the number of application that will use a controller and expenses for its design. Sometimes it is better to use a unified approach for several controllers and an operator for tuning on-line the controller.

The fundamental control problem formulation, today, is concluded by a continuous challenge to met multiple performance criteria. Control systems are characterized by performance changing due to various operating strategies.

Solving the fundamental control problem posed involve knowing of the following entities:

- Performance Criteria
- Objectives
- Constraints (inequality or equality)
- Process Representation
- Model Equation
- Uncertainty

All these entities have to be allowed to change during the control action because of various reasons such as: varying conditions demanded by decision layer and plant failures. These are strong reasons that impose redesign on-line.

The spectrum of control methodologies is wide due to various assumptions and compromises made in the formulation of performance criteria. The goal of formulating a performance criterion and a corresponding representation is an interesting challenge for the designer.

- **Optimization:**

  Apart from short time control, "economic objectives" of the plant have to be satisfied, so optimization is necessary. Here by "economic objectives" the reader should understand management requirements as well as plant long term objectives. For an aircraft minimum time of flight or minimum fuel consumption could represent such requirements. The main difference between control and optimization is the frequency and the range of the implementation. So, control is defined as a short term strategies, that for example, maintain the aircraft at a certain flight level, comparative with the optimization which ensure minimization of certain economic objectives. Often optimization is implemented at such a rate that the controller is assumed to be steady-state. It is interesting to note the difference in using the word optimization across the following chapters.

  Main performance criteria in use today are:

  Economic - which is connected with the optimization layer of the decision making process and used to impose a minimizing of an operating cost function.

  Environmental and Safety - due to certain regulation and legal requirements.

  Equipment - there are physical limitation of the equipment in use.

  Product Quality - strong consumer specifications are posed in order to face competition.

  Human - when a human operator is required.

- **Logistics:**

  Logistics means the scheduling of operating plants or subsystems and allocation of raw materials and even personnel for the maximization of profits of the company and realization of the developing program.

1.2 Short overview of report's contents

This report will be devoted to a survey of various methods that lie under the MBPC umbrella as well as the author's work. It is intended to present various approaches as prediction and optimization techniques for several types of problems.

The second chapter gives a general presentation of MBPC starting with a bit of history and continuing with principal issues of conception and implementation of MBPC. A survey of methods like DMC or GPC as well as various issues connected with these algorithms such as: stability, robustness, control law etc. Several examples provided by the author's experience in the field of MBPC are contained in chapter three. The forth chapter gives a global picture of available areas to research. The focus from the author's point of view for next few years of research is presented in order to give a clear idea upon the way to be followed.

The principal idea at which the whole report is aimed at is to introduce the reader into the relatively new field of "Constrained Predictive Control" and to present the author's work and view upon the subject. At the end of the report reference to the future research plan will be made.

We intend to present as well in this survey the relatively new idea of reconfigurable control. As it is known there are virtually an unlimited number of possible failures that can occur on a modern aircraft of either a civil or a military nature.

Actuator, sensor, other subsystem failures or even structural failures can cause a catastrophe. Generally pilots execute, in the case of certain anticipated failures, preplanned procedures. Of course, certain unanticipated events can occur which complicate a successful failure accommodation. Experienced pilots can often compensate for failures but in certain emergency situations there is a need for computer assisted or, in special cases, fully automated reconfiguration laws to preserve the aircraft. Significantly enhanced performance can be achieved by using a proper failure detection and identification information system. We need to know how such subsystems react in particular situations, in order to accommodate their information in the "Automated Reconfigurable Control System". Paper [25] presents an alternative methodology to constrained predictive control which deals with failures. It is shown that, a fuzzy model reference learning controller, can be used to reconfigure the nominal system to compensate for various *actuator* failures without using explicit failure information.

Aircraft accident investigation shows that sometimes even the most severe, unanticipated, failures can be handled , and the aircraft saved, if the pilot takes the proper actions at the right time. In those cases simulation exercises can provide the pilot with necessary abilities to avoid the accident. Here is a powerful tradeoff between possibility of anticipating failures and the handling of these. Certain unanticipated failures, which in principle have to be detected by an FDI and handled by reconfigurable controller, are subject of our interest as well. An alternative way to cope with such kind of failures is, beyond physical redundancy, to use statistical prediction of failures (provided by pilot-manual or flight safety regulations) in order to add the controller with such new models (of a damaged subsystem or even whole aircraft) and try to use predictive control. This addition sometimes is made at the modelling stage but other times could be handled by constraints inside the algorithm.

Chapter 2

Survey of various methods

2.1 General presentation of the MBPC (Model Based Predictive Control) approach

2.1.1 Brief history

Practical Model Based Predictive Control was discovered in the early 80's by development of more heuristic than theoretical algorithms such as IDCOM (Identification and Command) by Richalet [21] and by Cutler [11] in DMC (Dynamic Matrix Control).

Another approach not entirely different is GPC (Generalized Predictive Control) due to Clarke [13]. This algorithm can produce robust adaptive MBPC (Model Based Predictive Control).

Despite relative improvements in the late 1980's, methods like "finite horizon control strategies" of MBPC appeared to be difficult to analyse and general stability results for such a class of problems were not obtained.

Evolution in this direction was enabled by the fact that " end point state constraints" could be used to give guarantees of stability under weak assumptions and led to theoretical indicators for succesfull design using MBPC. Another advance was in using "long enough" horizons to force a constrained plant into a region near the set point followed by linear stabilizing control.

New plants are required to respond to higher quality indicators and integration factors, that make the control by conventional means difficult. The original lack of comprehensive theory has been overcome by adapting earlier work in receding-horizon control. In the last period 1990-94 there has been significant progress in understanding of the design and use of MBP Control. The new finite horizon methodology enables the addition of actuator input or rates of change constraints as well of states constraints for which new stability theories are emerging. So, according to various authors [6, 31] model predictive control seems to be in many ways "the choice of multivariable technique of 90's".

2.1.2 Advantages and drawbacks of the method

Model Based Predictive Control has a set of characteristics that make it well suited for industrial control of discrete and continuous processes. Many methods are available but among them most known are DMC (Dynamic Matrix Control) and GPC (General Predictive Control).

The basic principle of MBPC is accessible and many experiments and implementations have demonstrated the that this technique is not difficult to use and it has a good performance/cost ratio. MBPC offer a wide range of choice in model structure, prediction methods and optimization criteria. In recent years it has been realized that many control methods, which were successfully applied to real processes have in fact much in common, being variations of the MBPC theme. So, today, the advantages of MBPC are widely recognized.

Properties like difficult dynamic behaviour, multivariable interactive systems, constraints handling, performance parameters are handled without problems by MBPC. The MBPC is gaining more and more attention during the last years essentially due to the certain state of the following technologies: modelling and identification techniques and digital computers.

Main steps in applying MBPC are: the identification of the process model and tuning the design parameters. As within other control strategies performance and robustness are correlated, so the set point (reference trajectory) mainly affects the control systems dynamics vs. the horizons that act upon the system robustness.

The possibility of including constraints explicitly into the process made MBPC very successful in the industrial world. Handling in real-time actuator constraints upon inputs and rates of change in a natural way, this approach enables plants to operate more closely to their profitable margins.

Of course, like any algorithm MBPC has drawbacks. The main drawbacks are located at the level of number of parameters that are involved in the algorithm. Another problem is if the set of active constraints is changed major computations have to be done.

2.1.3 Structure of MBPC schemes

The defining feature of MPBC is the repeated optimization of a performance objective over a finite horizon extending from the current or some future time up to a control horizon. The use of an explicit and identifiable model is a general requirement of MBPC. The generality of the performance objective provides the opportunity to design MBPC for higher goals such as energy consumption or waste minimization.

Inherent structures of most of MBC schemes, see figure 2.1, are identical in their main format but differ in details. Generally these details are critical for the success of the algorithm. The FDI system is not a standard component of MBPC schemes but presented here due to the future research proposed regarding the issue of reconfigurable control in case of failures.

THE OPTIMIZER contains all constraints (for possible configurations) and the main attribute of this system component is that it must work iteratively in real time. The main task of the optimizer is to compute the present and future manipulated variable moves such that the predicted outputs follow the set reference in a desirable manner.

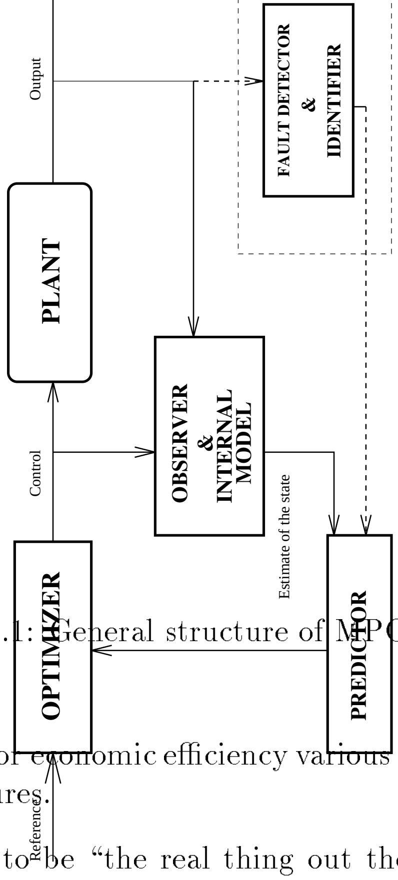

Figure 2.1: General structure of MPC schemes

In this case, sometimes, for economic efficiency various producers have developed short-cut optimization procedures.

THE PLANT is considered to be "the real thing out there", being attacked by various disturbances, that have to be modelled.

THE FDI is the key in reconfigurable control that generates inputs for other various systems describing the dimension, location and the way to deal with the failure.

THE OBSERVER in the case of using state space models is able to provide state estimation.

THE INTERNAL MODEL represents the plant, sufficiently complex to be able to simulate all components susceptible to failure but as simple as possible in order to improve speed of control. The use of the internal model for training and certification is appropriate in the global system not in the simple MBPC schemes.

THE PREDICTOR is the engine of the predictive control algorithms being able to provide the optimizer with future predicted values of states and outputs.

Attributes of MBPC that make it successful in industrial control design, but not only:

- Simplicity – basic idea of MBPC does not require complex mathematics like in H_∞ case and is fairly intuitive.

- Richness – all basic components of MBPC like model, horizons, objective function can be tailored with details of the current problem to be solved. The algorithm allows future extensions providing in this way a wide open methodology.

- Practicality – deals naturally with imposed constraints and in practice works routinely and profitably.

2.2 Review of DMC (Dynamic Matrix Control)

2.2.1 Input/Output models

All work developed in this section is based on the information from paper written by Dr. Maciejowski [28] and the book published by Prett and Garcia [35].

The Dynamic Matrix Controller for a system is based on a step response model of the process, given by:

$$y(t) = \sum_{i=1}^{\infty} g_i \Delta u(t-i) + n(t) \tag{2.1}$$

where $y(t)$ is the process output, $\Delta u(t)$ the increment of input variable (the manipulated variable), g_i are coefficients that correspond to the values of the step response model and $n(t)$ is the disturbance acting at time instant t. The use of $\Delta u(t) = u_k - u_{k-1}$ is required by the step response model and its corresponding coefficients.

Coefficients g_i of the model could be produced from state space description using following algorithm:

1. consider given A, B, C, D matrices of the linear discrete-time stable system. Initialize $i = 0$, $g_i = D$ and $Prod = I$.

2. compute $g_{i+1} = C \cdot Prod \cdot B + g_i$.

3. compute $Prod = Prod \cdot A$.

4. increment $i = i + 1$

5. cycle steps 2, 3 up to N after that for asymptotically stable system we consider that coefficients g_i for $i > N$ tend asymptotically to a constant value.

Another alternative is to compute first the impulse response model:

$$h_i = C{\cdot}A^{i-1}{\cdot}B \tag{2.2}$$

for $i{\in}\aleph$ and $h_0 = D$, then:

$$g_i = \sum_{j=0}^{i} h_j \tag{2.3}$$

In order to facilitate the computations of the control signals and to make things clear it is convenient to group those components of the prediction of the output that depend on $\Delta u(t+j)$ for $j \geq 0$ in:

$$y(t+j) = \sum_{i=1}^{j} g_i \Delta u(t+j-i) + \sum_{i=j+1}^{\infty} g_i \Delta u(t+j-i) + n(t+j) \tag{2.4}$$

Another useful idea in order to simplify computations is to consider that the estimate of future values of the disturbance $n(t+j)$ is the current value $n(t)$ included as well in the model of the process:

$$n(t+j) = n(t) = y(t) - \sum_{i=1}^{\infty} g_i \Delta u(t-i) \tag{2.5}$$

We should note that this is just a simple way to see disturbance problem and more sophisticated estimates are possible. As well this kind of approach assumes that all disturbances are steps. then it is possible to write the prediction of the output:

$$\hat{y}(t+j) = \sum_{i=1}^{j} g_i \Delta u(t+j-i) + \sum_{i=j+1}^{\infty} g_i \Delta u(t+j-i) + y(t) - \sum_{i=1}^{\infty} g_i \Delta u(t-i) \tag{2.6}$$

Because of the use of the backward shift operator q^{-1} in discrete case equation 2.6 may be written in a more compact form:

$$\hat{y}(t+j) = G_j(q^{-1})\Delta u(t+j) + p_j \tag{2.7}$$

where

- $G_j(q^{-1})$is a polynomial in the backward shift operator q^{-1}:

$$G_j(q^{-1}) = g_1 q^{-1} + \ldots + g_i q^{-j} \tag{2.8}$$

- and p_j represent the free response of the system, given by:

$$p_j = y(t) + \sum_{i=j+1}^{\infty} g_i \Delta u(t+j-i) - \sum_{i=1}^{\infty} g_i \Delta u(t-i) = y(t) + \sum_{i=1}^{\infty} (g_{j+i} - g_i)\Delta u(t-i) \tag{2.9}$$

The reason for calling p_j free response of the process is that if we consider $\Delta u(t+j)$ equal to zero for $j = 0 \ldots N_2$, then equation 2.7 becomes $\hat{y}(t+j) = p_j$ where N_2 is called maximum output horizon.

With the assumption of an asymptotically stable process, as discussed above, the coefficients g_i tend to a constant value, so equation 2.9 can be simplified to:

$$p_j = y(t) + \sum_{i=1}^{\mathbb{N}} (g_{j+i} - g_i)\Delta u(t-i) \tag{2.10}$$

where N is the number for which:

$$(g_{j+i} - g_i) \approx 0 \quad i > N \quad j = N_1 \ldots N_2 \tag{2.11}$$

An important note is that if the process is not asymptotically stable, then N does not exist and as result p_j cannot be computed using input output formulation. We should note here that with a state-space model other equivalent computations can be made.

In the DMC (Dynamic Matrix Control) method the control signal is obtained minimizing the cost function:

$$J = \sum_{j=N_1}^{N_2} [\hat{y}(t+j) - w(t+j)]^2 + \sum_{j=0}^{N_u-1} [\lambda \Delta u(t+j)]^2 \tag{2.12}$$

with respect to $\Delta u(t+j)$. In this expression $w(t)$ is a given set point at time instant t, λ a weighting factor upon manipulated variable $u(t)$ and $\hat{y}(t+j)$ are output prediction values at time instants $(t+j)$; these predictions depend strictly upon the manipulated variables $\Delta u(t+j)$ and can be obtained from the process model.

Of course it is assumed that $\Delta u(t+j) = 0$ for $j \geq N_u$.

With the predictions as in equation 2.7 the cost function 2.12 can be written after a bit of algebra in the matrix form:

$$J = \Delta U^T(t)[G^T G + \lambda I]\Delta U(t) - 2{e_0}^T G \Delta U(t) + {e_0}^T e_0 \tag{2.13}$$

where

- $\Delta U(t)^T = [\Delta u(t), \Delta u(t+1), \ldots, \Delta u(t)]$ is the vector of future controls to be computed
- ${e_0}^T = [w(t+N_1) - p_{N_1}, w(t+N_1+1) - p_{N_1+1}, \ldots, w(t+N_2) - p_{N_2}$ is the vector of known future errors between the set point $w(t+j)$ and the free response of the system p_j
- G, having real terms, is a matrix given by:

$$G = \begin{pmatrix} g_{N_1} & \ldots & g_1 & 0 & \ldots & & 0 \\ g_{N_1+1} & \ldots & g_2 & g_1 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ g_{N_2} & \ldots & \ldots & \ldots & \ldots & \ldots & g_{N_2-N_{u+1}} \end{pmatrix} \tag{2.14}$$

The meaning of most of the variables involved in the DMC algorithm as well the significance of the various horizons can be seen in figure 2.2. In the next section the unconstrained as well as constrained case of DMC will be considered in relation to implementation issues.

2.2.2 Unconstrained case

The DMC algorithm has some special features in the unconstrained case:

The index J, see equation 2.12, can be minimized using standard techniques like the "least square method" that provides a nice analytical solution in the unconstrained case:

$$\Delta U(t) = [G^T G + \lambda I]^{-1} G^T e_o \tag{2.15}$$

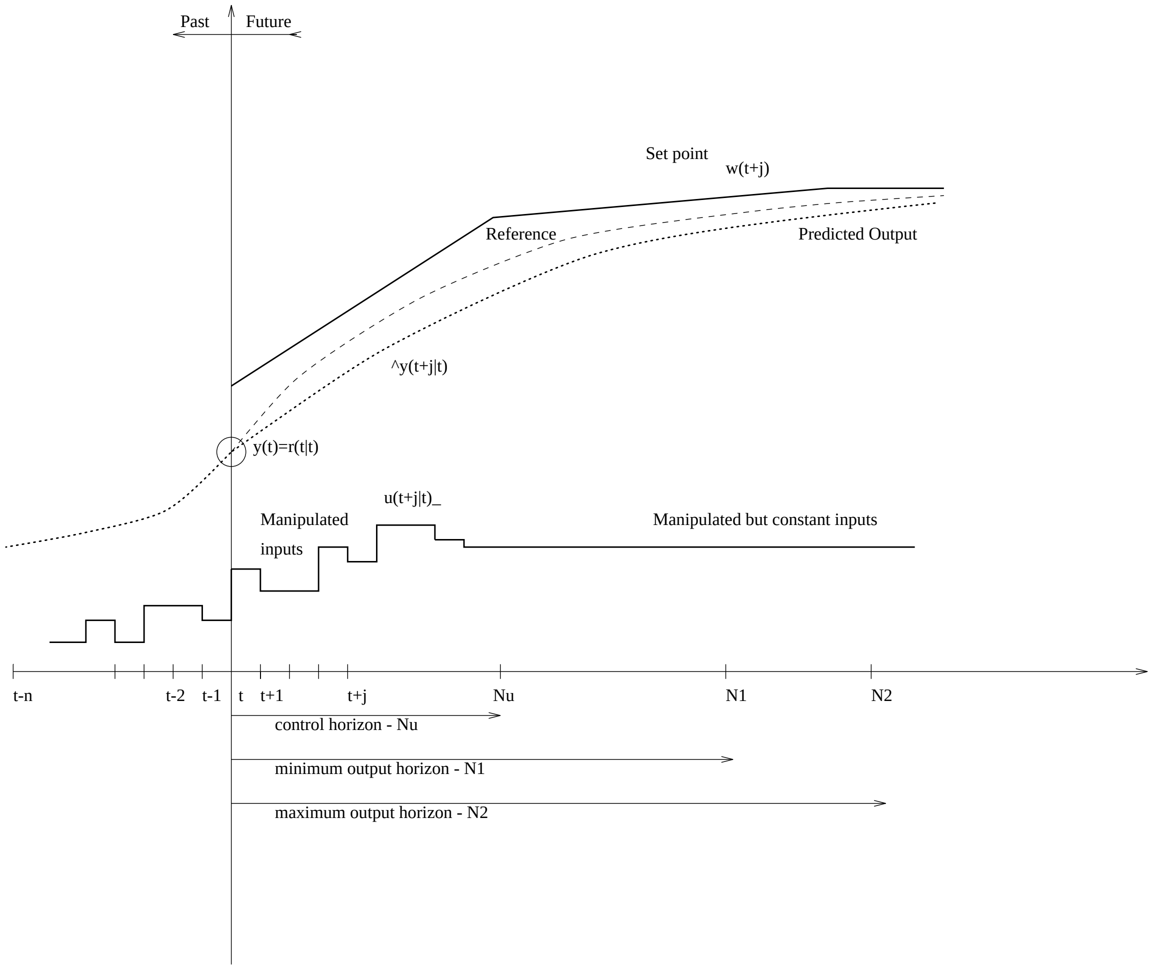

Figure 2.2: Global Picture of MBPC Algorithms

This solution will provide N_u values for the manipulated variable $\Delta u(t)$, but from those only the first one is applied to the process at time t. The next sampling period will be used to compute the next set of values for $\Delta u(t)$.

We intend to present here some special aspects regarding the implementation of the unconstrained case:

Using the following form to express the model:

$$y(t) = \sum_{i=1}^{N} g_i \Delta u(t-i) + g_N u(t-N-1) + n(t) \tag{2.16}$$

we could use as well the relation between the coefficient of the step and impulse response models:

$$h_i = g_i - g_{i-1} = \Delta g_i \tag{2.17}$$

in order to deduce the impulse response model to:

$$y(t) \approx \sum_{i=1}^{N} h_i u(t-i) + n(t) \tag{2.18}$$

Equation 2.18 provides a valid expression for asymptotically stable processes for N big enough. Now using equation 2.16 and following the procedure described in section "Review of DMC (Dynamic Matrix Control)" for computing the predictions p_j we obtain a similar result:

$$\begin{aligned}\hat{y}(t+j) = \sum_{i=1}^{j} g_i \Delta u(t+j-i) + \sum_{i=j+1}^{N-1} \Delta u(t+j-i) + g_N u(t+j-N) + y(t) - \\ - \sum_{i=1}^{N-1} g_i \Delta u(t-i) - g_N u(t-N)\end{aligned} \tag{2.19}$$

This equation can be transformed using the $G(q^{-1})$ polynomial into:

$$\hat{y}(t+j) = G_j(q^{-1})\Delta u(t+j) + p_j \tag{2.20}$$

where the expression of the free response is:

$$\begin{aligned}p_j = y(t) + \sum_{i=1}^{N-j}(g_{j+i} - g_i)\Delta u(t-i) - \sum_{i=N-j+1}^{N-1} g_i \delta u(t-i) + \\ + g_N(1 - g^{-j})u(t+j-N)\end{aligned} \tag{2.21}$$

2.2.3 Constrained case

This section will describe the particular case of the constrained input/output formulation. Because the issue of constraints was discussed before, here I will develop only the necessary tools required in to dealing with such a case.

The quadratic cost function is to be minimized subject to constraints on:

- the inputs levels: $u_l(j) \leq u(j) \leq u_u(j)$ where $t \leq j \leq t + N_u - 1$
- the input rates of change: $\Delta u_l(j) \leq \Delta u(j) \leq \Delta u_u(j)$ where $t \leq j \leq t + N_u - 1$
- the outputs levels: $y_l(j) \leq \tilde{y}(j) \leq y_u(j)$ where $t + N_1 \leq j \leq t + N_2$

Because future values of output levels ($\tilde{y}_i$) are unknown until the control inputs have been calculated the usual procedure in this case is to replace those values by their predictions: $y_l(j) \leq \hat{y}(j) \leq y_u(j)$ where $t + N_1 \leq j \leq t + N_2$

As known, when we solve the minimization problem subject to constraints, it is assumed that after N_u steps the manipulated variable is set to its previous value so $\Delta u(j) = 0$ for $j \geq t + N_u$.

In order to take advantage of the quadratic programming methods, mainly in the constraint case, it is better to have all constraints in a compact form such as $A \cdot x \leq b$ (form required by most of QP algorithms).

Denoting a lower triangular matrix such as:

$$L_{N_u m} = \begin{pmatrix} I_m & 0 & \dots & 0 \\ I_m & I_m & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ I_m & I_m & I_m & I_m \end{pmatrix} \tag{2.22}$$

where m is the number of plant inputs, we are able to pack the constraints in the following form by stacking inequalities on top of each other:

$$
\begin{pmatrix} -L_{N_u m} \\ +L_{N_u m} \\ -I_{N_u m} \\ +I_{N_u m} \\ -S_{N_2-N_1}^{N_u} \\ +S_{N_2-N_1}^{N_u} \end{pmatrix} \cdot \Delta u(t+j) \geq \begin{pmatrix} u(t-1) - u_u(t) \\ \vdots \\ u(t-1) - u_u(t+N_u-1) \\ u_l(t) - u(t-1) \\ \vdots \\ u_l(t+N_u-1) - u(t-1) \\ -\Delta u_u(t) \\ \vdots \\ -\Delta u_u(t+N_u-1) + \Delta u_l(t) \\ \vdots \\ -\Delta u_l(t+N_u-1) \\ \vdots \\ p_{N_1} + w(t+N_1) - y_u(t+N_1) \\ \vdots \\ p_{N_2} + w(t+N_2) - y_u(t+N_2) \\ y_l(t+N_1) - p_{N_1} - w(t+N_1) \\ \vdots \\ y_l(t+N_2) - p_{N_2} - w(t+N_2) \end{pmatrix} \tag{2.23}
$$

Here N_2 denotes the maximum output horizon, N_1 the minimum output horizon, $S_{N_2-N_1}^{N_u}$ represent a matrix that help us to define additional weighting matrices from the cost function and p_{N_1} or p_{N_2} are variables that contain all the terms in the cost function which are known before and fixed at a time of computation.

It is easy to prove that the constraints can be expressed in the above form 2.23 using certain separations. For example consider the constraints upon inputs $u_l(t+j-1) \leq u(t+j-1)$ where $1 \leq j \leq N_u$ we are able to express this linear inequality in the following form:

$$
u_l(t+j) \leq u(t-1) + \sum_{i=1}^{j} \Delta u(t+i-1) \tag{2.24}
$$

which at its turn can be described using the product between two vectors:

$$
u_l(t+j-1) \leq u(t-1) + \begin{pmatrix} I_m & I_m & \ldots & I_m \end{pmatrix} \cdot \begin{pmatrix} \Delta u_t \\ \Delta u_{t+1} \\ \vdots \\ \Delta u_{t+j-1} \end{pmatrix} \tag{2.25}
$$

Stacking the N_u inequalities, one on top of each other, gives the second column defined in relation 2.23. The right inequality $u(t+j-1) \geq u_u(t+j-1)$ is rewritten like $-u(t+j-$

1)$\leq -u_u(t+j-1)$ in order to apply a similar algorithm. In such a way the first $N_u \cdot m$ rows from the right column of relation 2.23 are defined.

The input rates constraints as well as outputs constraints are treated in a similar manner. An interesting fact to be noted is that the total number of constraints is $4 \cdot N_u + 2 \cdot N_2 - N_1$.

2.3 State space MBPC formulation

This presentation takes advantage of information presented in the papers [20] and [29].

In the following we will show how the constrained multivariable predictive control formulation may be implemented using state space methods sometimes including state estimation. This formulation even it address DMC include algorithms like GPC or MAC.

2.3.1 State space models

Consider an index i for the plant in use, we are able to assume that a model for the system is available in discrete state space form:

$$x_i(k+1) = A_i x_i(k) + B_{u_i} u_i(k) + B_{d_i} d_i(k) \tag{2.26}$$

$$y_i(k) = C_i x_i(k) + \nu(k) \tag{2.27}$$

where $x_i(k) \in \Re^n$ is the system state vector, $u_i(k) \in \Re^m$ is the system control input vector and $y_i(k) \in \Re^p$ is the vector of outputs, all variables being measured at time k. It is important to note that due to a complete actuator failure the system, respective model state dimension may change in such a way that $x_i(k) \in \Re^{n_i}$.The reader should note that input, state, and output (measurement) vectors are indexed using the variable i in order to indicate the model in use. Here $d_i(k)$ is considered to describe unmeasured state disturbance input and $\nu(k)$ is an unmeasured output disturbance vector. This set of models is considered to have each pair (A_i, B_i) stabilizable and (A_i, C_i) detectable.

Trying to cast the state space MBPC problem into a conventional QP form, we express the state vector in terms of the change in:

- the manipulated variable (at time k): $\Delta u_i(k) = u_i(k) - u_i(k-1)$.
- the state disturbance (at same time k): $\Delta d_i(k) = d_i(k) - d_i(k-1)$.
- the output disturbance (at sampling time k): $\Delta w_i(k) = w_i(k) - w_i(k-1)$.

In such a case, the state space model (using a matrix description) becomes:

$$\begin{aligned} \begin{pmatrix} \Delta x_i(k+1) \\ y_{x_i}(k+1) \end{pmatrix} = \begin{pmatrix} A_i & 0 \\ C_i \cdot A_i & I \end{pmatrix} \cdot \begin{pmatrix} \Delta x_i(k) \\ y_{x_i}(k) \end{pmatrix} + \begin{pmatrix} B_{u_i} \\ C_i \cdot B_{u_i} \end{pmatrix} \cdot \Delta u_i(k) \; + \\ + \begin{pmatrix} B_{d_i} & 0 \\ C_i \cdot B_{d_i} & I \end{pmatrix} \cdot \begin{pmatrix} \Delta d_i(k) \\ \Delta w_i(k) \end{pmatrix} \\ y_i(k) = \begin{pmatrix} 0 & I \end{pmatrix} \cdot \begin{pmatrix} \Delta x_i(k) \\ y_{x_i}(k) \end{pmatrix} + \nu(k) \end{aligned} \tag{2.28}$$

where $y_{x_i}(k+1)$ corresponds to a vector of outputs free from measurement noise.

The augmented system will be referred to by the realization $\hat{A}$, $\hat{B}$, $\hat{C}$:

$$\begin{aligned} \begin{pmatrix} \Delta x_i(k+1) \\ y_{x_i}(k+1) \end{pmatrix} = \hat{A}_i \cdot \begin{pmatrix} \Delta x_i(k) \\ y_{x_i}(k) \end{pmatrix} + \hat{B}_{u_i} \cdot \Delta u_i(k) + \begin{pmatrix} \hat{B}_{d_i} & \hat{B}_{w_i} \end{pmatrix} \cdot \begin{pmatrix} \Delta d_i(k) \\ \Delta w_i(k) \end{pmatrix} \\ y_i(k) = \hat{C}_i \cdot \begin{pmatrix} \Delta x_i(k) \\ y_{x_i}(k) \end{pmatrix} + \nu(k) \end{aligned} \tag{2.29}$$

Certain elements of the state vector may not be available for measurement at time k so a Kalman observer, described later in this chapter, will be employed. The state representation of the plant is not assumed to be minimal, so, the DMC algorithm can be included by letting $x_i(k)$ be a vector of past input changes.

2.3.2 Disturbance models

Unmeasured state disturbance input $d_i(k)$ and unmeasured output disturbance vector $w_i(k)$ may be modelled in either a deterministic or a stochastic sense, depending on the particular application.

For the *deterministic model*, disturbances are usually assumed to be in the form of: impulses, steps, ramps, or, the output of some other deterministic model suitable for representing slowly varying load disturbances on a process.

Because the estimation problem, via a Kalman observer, requires a *stochasting setting* we model the disturbance as a linear system driven by white noise: $\Delta\tilde{d}_i(k)$, $\Delta\tilde{w}_i(k)$. In order to take advantage of the Kalman filtering we will augment these two linear systems to the initial state equations of the main system, as follows:

$$\begin{aligned} \begin{pmatrix} \Delta x_i(k+1) \\ y_{x_i}(k+1) \\ \Delta d_i(k+1) \\ \Delta w_i(k+1) \end{pmatrix} = \begin{pmatrix} A_i & B_{d_i} & 0 & \\ C_i \cdot A_i & I & C_i \cdot B_{d_i} & I \\ 0 & 0 & A_{d_i} & 0 \\ 0 & 0 & 0 & A_{w_i} \end{pmatrix} \cdot \begin{pmatrix} \Delta x_i(k) \\ y_{x_i}(k) \\ \Delta d_i(k) \\ \Delta w_i(k) \end{pmatrix} + \begin{pmatrix} B_{u_i} \\ C_i \cdot B_{u_i} \\ 0 \\ 0 \end{pmatrix} \cdot \Delta u_i(k) + \\ + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ I & 0 \\ 0 & I \end{pmatrix} \cdot \begin{pmatrix} \Delta\tilde{d}_i(k) \\ \Delta\tilde{w}_i(k) \end{pmatrix} \end{aligned} \tag{2.30}$$

$$y_i(k) = \begin{pmatrix} 0 & I & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta x_i(k) \\ y_{x_i}(k) \\ \Delta d_i(k) \\ \Delta w_i(k) \end{pmatrix} + v_i(k) \tag{2.31}$$

where corresponding covariance matrices, that involve tuning parameters of the observer Q_d, Q_w, Q_v, are:

$$E\left[\begin{pmatrix} \Delta\tilde{d}_i \\ \Delta\tilde{w}_i \end{pmatrix} \cdot \begin{pmatrix} \Delta\tilde{d}_i^{\,T} \\ \Delta\tilde{w}_i^{\,T} \end{pmatrix} \right] = \begin{pmatrix} Q_d & 0 \\ 0 & Q_w \end{pmatrix} \tag{2.32}$$

$$E\left[v_i(k)\cdot v_i(k)^T\right] = Q_v \tag{2.33}$$

It is interesting to note that, according to the requirements in employing the "Kalman Observer" for state estimation, if the system being considered is such that disturbances $d_i(k)$ and $w_i(k)$ are direct stochastic inputs (white or zero mean noise, random amplitude impulses, steps or ramps) then it is not necessary to augment them to the state vector of the main system.

2.3.3 State prediction

As mentioned, since certain elements of the state vector may be not available for measurement at time k we will determine a set of state estimates for each model of the plant in use. The change of state estimates will be either the change of states of the main system or the change of states of that system augmented by the disturbance variables, as shown in the previous section.

The state estimates $\left(\Delta x_i(k) \quad y_{x_i}(k)\right)$ will be denoted as $\hat{x}_i(k)\in\Re^{(n+\hat{p})}$ (where $\hat{p}$ represents the number of outputs uncorrupted by measurement noise) and they will rely upon selecting a particular observer throughout the weighting matrices: Q_d, Q_w, Q_v.

So, in general, the state estimation will take the following form at a time $k+j$ ($j = 0\ldots N_u - 1$) from the prediction horizon N_u:

$$\hat{x}(k+j+1) = F_i(j)\cdot\hat{x}(k+j) + G_i\cdot\Delta u_i(k+j) + H_i(j)\cdot y_i(k+j) \tag{2.34}$$

where $\Delta u_i(k+j)$ is a vector of future control changes.

We should emphasize that $F_i(j)$, G_i, $H_i(j)$ matrices are derived, from the open-loop model itself, in a way that depends on the estimation/prediction scheme employed. This depends on the type of observer chosen and its weighting matrices as well as on the disturbances, that affect the plant, and on noise upon measurements. We need to assume that $F_i(j)$, G_i, $H_i(j)$ depend only on i (the model set) and j(time in the future) but not on k (current time) in order to exclude certain possibilities, such as non-stationary Kalman filters.

Few simplifications, such as:

$H_i(j) = 0$ for $j > 0$

$F_i(0)\neq F_i(1)$ and

$F_i(j) = F_i(j+1)$ for $j > 0$

may occur. Therefore the state estimator is constructed in the following way:

$$\hat{x}_i(k+1) = F_i(0)\hat{x}_i(k) + G_i\Delta u_i(k) + H_i(0)y_i(k) \tag{2.35}$$

in which $H(0) = K$, $F(0) = \hat{A} - H(0)\hat{C}$, $F(1) = F(2) = \ldots = F(N_2 - 1) = \hat{A}$ and $G = \hat{B}$. For the future computation it is not necessary to assume the last two simplifications.

From the earlier definition of the state space description 2.26 and using 2.34 the vector of future state estimates may be written as:

$$\hat{X}_i(k) = \begin{pmatrix} \hat{x}_i(k+1) \\ \vdots \\ \hat{x}_i(k+N_2) \end{pmatrix} = \mathcal{F}_i(k)\cdot\hat{x}_i(k) + \mathcal{G}_i\cdot\Delta U_i(k) + H_i(k)\cdot y_i(k) \qquad (2.36)$$

We should note here that the vector of the change of state estimates denoted by:

$$\hat{X}_i(k) = \left[\hat{x}_i(k+1)^T, \ldots, \hat{x}_i(k+N_2)^T\right]^T \qquad (2.37)$$

is dependent only upon state estimate at time $j = 0$ which is $\hat{x}_i(k)$. Therefore it is necessary to have a state estimator that operates outside of model based controller to determine current state estimate $\hat{x}_i(k)$.

In the following, we shall use the notation $\prod_{\lambda=\epsilon}^{\delta} R(\lambda)$ to denote the product:

$$R(\epsilon)\cdot R(\epsilon-1)\cdot\cdot R(\epsilon-2)\ldots R(\delta) \qquad (2.38)$$

for $\epsilon > \delta$.

Now we are able to define the matrices involved in equation 2.36:

$$\mathcal{F}_i = \begin{pmatrix} F_i(0) \\ F_i(1)\cdot F_i(0) \\ \vdots \\ \prod_{j=N_2-1}^{0} F_i(j) \end{pmatrix} \qquad (2.39)$$

$$\mathcal{G}_i = \begin{pmatrix} G_i & 0 & \ldots & 0 \\ F_i(1)\cdot G_i & G_i & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \left[\prod_{j=N_u-1}^{1} F_i(j)\right]\cdot G_i & \left[\prod_{j=N_u-1}^{2} F_i(j)\right]\cdot G_i & \ldots & G_i \\ \vdots & \vdots & \ddots & \vdots \\ \left[\prod_{j=N_2-1}^{1} F_i(j)\right]\cdot G_i & \left[\prod_{j=N_2-1}^{2} F_i(j)\right]\cdot G_i & \ldots & \left[\prod_{j=N_2-1}^{N_u} F_i(j)\right]\cdot G_i \end{pmatrix} \qquad (2.40)$$

$$\mathcal{H}_i = \begin{pmatrix} H_i(0) \\ F_i(1)\cdot H_i(0) \\ \vdots \\ \prod_{j=N_2-1}^{1} H_i(0) \end{pmatrix} \qquad (2.41)$$

Here the significance of N_u and N_2, $N_u \le N_2$ is as defined in previous sections of this chapter (N_2 is considered to be the maximum optimization horizon).

The optimization is computed with respect to $\Delta u_i(k+j) = 0$ for $j \ge N_u$. The vector of future manipulated variable changes will be denoted by:

$$\Delta U_i(k) = \left[\Delta u_i(k)^T, \ldots, \Delta u_i(k+N_u-1)^T\right]^T \qquad (2.42)$$

2.3.4 Constraints

For each model set i we have an associated constraint set corresponding to:

- control signals themselves (corresponding to actuator limits):

$$M_i \cdot U_i(k) \leq \begin{pmatrix} m_{i1}(k) \\ \vdots \\ m_{i\mu}(k) \end{pmatrix} \tag{2.43}$$

  which is in fact:

$$M_i \cdot \begin{pmatrix} u_i(k) \\ \vdots \\ u_i(k+N_u-1) \end{pmatrix} \leq \begin{pmatrix} m_{i1}(k) \\ \vdots \\ m_{i\mu}(k) \end{pmatrix} \tag{2.44}$$

  where $U_i(k) = \left[U(k)^T, \ldots, U(k+N_u)^T\right]^T$ and μ are the number of constraints over the control horizon.

- changes of control signals (corresponding to actuator rates of change):

$$L_i \cdot \Delta U_i(k) \leq \begin{pmatrix} l_{i1}(k) \\ \vdots \\ l_{i\lambda}(k) \end{pmatrix} \tag{2.45}$$

  where $\Delta U_i(k) = \left[\Delta u_i(k)^T, \ldots, \Delta u_i(k+N_u-1)^T\right]^T$ and λ are the total number of rate constraints across the control horizon.

- states constraints (that now contain the output constraints):

$$N_i \cdot \hat{X}_i(k) \leq \begin{pmatrix} n_{i1}(k) \\ \vdots \\ n_{i\nu}(k) \end{pmatrix} \tag{2.46}$$

  which is in fact:

$$N_i \cdot \begin{pmatrix} \hat{x}_i(k+1) \\ \vdots \\ \hat{x}_i(k+N_2) \end{pmatrix} \leq \begin{pmatrix} n_{i1}(k) \\ \vdots \\ n_{i\nu}(k) \end{pmatrix} \tag{2.47}$$

  where $\hat{X}_i(k) = \left[\hat{x}_i(k+1)^T, \ldots, \hat{x}_i(k+N_2)^T\right]^T$

All above inequalities between vectors are defined by stacking together scalar inequalities. The right hand side of the last inequality may depend on predicted set-points for state and/or output variables. We should note that in these inequalities $\lambda \neq \mu \neq \nu$ in general.

Using a complete similar procedure as presented in section 2.2.3 all inequalities can be written stacked one upon another as a single linear inequality constraint on the vector of future input changes ΔU_i. Note that control $u_i(k)$ is still to be determined at time k.

We are going to show the way of stacking on control signal constraints. For this we need to note that:

$$u_i(k+j) = u_i(k-1) + \sum_{j=0}^{l} \Delta u_i(k+l) \tag{2.48}$$

therefore we can write:

$$U_i(k) = \Lambda \Delta U_i(k) + \mathcal{I} u_i(k-1) \tag{2.49}$$

The form for matrices Λ and $\mathcal{I}$ is given here:

$$\Lambda = \begin{pmatrix} I_m & 0 & \dots & 0 \\ I_m & I_m & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ I_m & I_m & I_m & I_m \end{pmatrix} \tag{2.50}$$

and

$$\mathcal{I} = [I_m, \dots, I_m]^T \tag{2.51}$$

Performing algebraic manipulation we are able to stack all constraints inequalities in the following form:

$$\mathcal{D}_i \cdot \Delta U_i(k) \leq \mathcal{E}_i \cdot \begin{pmatrix} u_i(k-1) \\ \hat{x}_i(k) \\ y_i(k) \\ c_i(k) \end{pmatrix} \tag{2.52}$$

where:

$$\mathcal{D}_i = \begin{pmatrix} L_i \\ M_i \Lambda \\ N_i \mathcal{G}_i \end{pmatrix} \tag{2.53}$$

and

$$\mathcal{E}_i = \begin{pmatrix} 0 & 0 & 0 & I_\lambda & 0 & 0 \\ -M_i \mathcal{I}_i & 0 & 0 & 0 & I_\mu & 0 \\ 0 & -N_i \mathcal{F}_i & -N_i \mathcal{H}_i & 0 & 0 & I_\nu \end{pmatrix} \tag{2.54}$$

In these equations identity matrices are consistent with the number of constraints involved in the following way $l_i(k) \in \Re^\lambda$, $m_i(k) \in \Re^\mu$ and $n_i(k) \in \Re^\nu$ ($\lambda = 2 \cdot N_u \cdot m$, $\mu = 2 \cdot N_u \cdot m$, $\nu = 2 \cdot N_2 \cdot (n + \hat{p})$)where:

$$l_{min_i}(k) \leq \Delta U_i(k) \leq l_{max_i}(k) \tag{2.55}$$

$$m_{min_i}(k) \le U_i(k) \le m_{max_i}(k) \tag{2.56}$$

$$n_{min_i}(k) \le \hat{X}_i(k) \le n_{max_i}(k) \tag{2.57}$$

and

$$l_i(k) = \begin{pmatrix} -l_{min_i}(k) \\ l_{max_i}(k) \end{pmatrix} \tag{2.58}$$

$$m_i(k) = \begin{pmatrix} -m_{min_i}(k) \\ m_{max_i}(k) \end{pmatrix} \tag{2.59}$$

$$n_i(k) = \begin{pmatrix} -n_{min_i}(k) \\ n_{max_i}(k) \end{pmatrix} \tag{2.60}$$

Now we are able to define the vector $c_i(k)$ as:

$$c_i(k) = \begin{pmatrix} l_i(k) \\ m_i(k) \\ n_i(k) \end{pmatrix} \tag{2.61}$$

The expression of L_i, M_i, N_i is given by:

$$L_i = M_i = \begin{pmatrix} -I_{N_u m} \\ I_{N_u m} \end{pmatrix} \tag{2.62}$$

$$N_i = \begin{pmatrix} -I_{N_2(n+\hat{p})} \\ I_{N_2(n+\hat{p})} \end{pmatrix} \tag{2.63}$$

where m is the number of inputs, $\hat{p}$ is the number of the outputs uncorrupted by measurement noise $(y_{x_i}(k))$, N_u is the control horizon and N_2 is the prediction horizon.

In the case of "receding horizon", in the literature, is assumed that the system after several steps achieve a steady-state: $\hat{x}_i(k + N_2) = const.$ which results in adding an equality to the set of constraints. This fact is much difficult to handle with previous approach because the equality constraints are employed over the prediction horizon.

2.3.5 The cost functional

The purpose of stacking inequalities in the form of 2.52 was to solve minimization of the following cost functional:

$$J_i(k) = \sum_{j=1}^{N_2} \|(C_i \hat{x}_i(k+j) - s_i(k+j)\|^2_{Q_i(j)} + \sum_{j=0}^{N_u} \|\Delta u_i(k+j)\|^2_{R_i(j)} \tag{2.64}$$

via QP algorithm. In the cost function the significance of variables is: $s_i(k)$ represent the vector of set-points for states and outputs (at time k). The weights $Q_i(j)$ and $R_i(j)$ are assumed to be independent of k, although they depend on i and may depend on j.

Now the cost function can be written as:

$$\begin{aligned} J_i(k) = \Delta U_i(k)^T \cdot \left[\mathcal{G}_i{}^T \cdot \mathcal{C}_i{}^T \cdot \mathcal{Q}_i \cdot \mathcal{C}_i \mathcal{G}_i + \mathcal{R}_i\right] \cdot \Delta U_i(k) + \\ +2 \cdot \left[\left[\mathcal{F}_i \hat{x}_i(k) + \mathcal{H}_i y_i(k)\right]^T \cdot \mathcal{C}_i{}^T - \mathcal{S}_i(k)^T\right] \cdot \mathcal{Q}_i \cdot \mathcal{C}_i \cdot \mathcal{G}_i \cdot \Delta U_i(k) + \\ \mathcal{K}_i(k) \end{aligned} \tag{2.65}$$

where:

$$\mathcal{Q}_i = diag\left[Q_i(1), \ldots, Q_i(N_2)\right] \tag{2.66}$$

$$\mathcal{R}_i = diag\left[R_i(1), \ldots, R_i(N_u)\right] \tag{2.67}$$

$$\mathcal{C}_i = diag\left[C_i, \ldots, C_i\right] \tag{2.68}$$

and:

$$\mathcal{S}_i(k) = \left[s_i(k+1)^T, \ldots, s_i(k+N_2)^T\right]^T. \tag{2.69}$$

The term $\mathcal{K}_i(k)$ is independent as seen of $\Delta U_i(k)$ and therefore can be omitted from $J_i(k)$ without affecting optimization.

It is interesting to note that in the cost function there are several types of variables:

- constants of the process model, connected with estimation and prediction done as $\mathcal{G}_i$, $\mathcal{C}_i$, $\mathcal{Q}_i$, $\mathcal{R}_i$, $\mathcal{F}_i$ and ,$\mathcal{H}_i$.
- parameters varying with k and the model set i such as $\hat{x}(k)$, $y_i(k)$ and $\mathcal{S}_i(k)$, being initialized at each new optimization.
- parameters constant during the optimization algorithm like N_2, and N_u that define horizons strategy.
- real optimization variable $\Delta U_i(k)$

Now it is possible to obtain the solution of the optimization problem as result of the following QP:

$$\min_{\Delta U_i(k)} \Delta U_i(k)^T \cdot \mathcal{A}_i \cdot \Delta U_i(k) + \left[\hat{x}_i(k)^T, y_i(k)^T, \mathcal{S}(k)^T\right] \cdot \mathcal{B}_i \cdot \Delta U_i(k) \tag{2.70}$$

subject to constraints:

$$\mathcal{D}_i \cdot \Delta U_i(k) \leq \mathcal{E}_i \cdot \begin{pmatrix} u_i(k-1) \\ \hat{x}_i(k) \\ y_i(k) \\ c_i(k) \end{pmatrix} \tag{2.71}$$

where the previous matrices $\mathcal{D}_i$ and $\mathcal{E}_i$,$c_i(k)$ are obtained as shown in section 2.3.4. The rest of unknown matrices are obtained in the following form:

$$\mathcal{A}_i = \mathcal{G}_i{}^T \mathcal{C}_i{}^T \mathcal{Q}_i \mathcal{C}_i \mathcal{G}_i + \mathcal{R}_i \tag{2.72}$$

and

$$\mathcal{B}_i = 2 \cdot \begin{pmatrix} \mathcal{C}_i \mathcal{F}_i & \mathcal{C}_i \mathcal{H}_i & -I_{N_2 p} \end{pmatrix}^T \mathcal{Q}_i \mathcal{C}_i \mathcal{G}_i \tag{2.73}$$

where $N_2 p$ is the dimension of the vector of set points at time k.

2.4 Review of GPC (General Predictive Control) algorithm

2.4.1 The CARIMA model

In this section and the following we will use information from the author of GPC algorithm D. Clarke, information that can be founded in papers [7] and [13]. We will start by presenting the CARIMA model (Controlled Auto-Regresive and Integrated Moving Average) and considering a locally-linearized model:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \nu(t) \tag{2.74}$$

where A and B are polynomials in the backward operator q^{-1}:

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ldots + a_{na} q^{-na} \tag{2.75}$$

$$B(q^{-1}) = 1 + b_1 q^{-1} + \ldots + b_{nb} q^{-nb} \tag{2.76}$$

and $u(t)$ is the control variable (input), $y(t)$ is the measured variable (output), $\nu(t)$ is the disturbance term.

The reason for the name CARIMA is that in the literature the disturbance $\nu(t)$ is considered to be of moving average form:

$$\nu(t) = T(q^{-1})\epsilon(t) \tag{2.77}$$

where

$$T(q^{-1}) = 1 + t_1 q^{-1} + \ldots + t_{nt} q^{-nt} \tag{2.78}$$

and $\epsilon(t)$ which is an uncorrelated random sequence.

First we combine equations 2.74 and 2.77 in order to obtain the CARMA model which is:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + T(q^{-1})\epsilon(t) \tag{2.79}$$

For unstationary disturbances such as random steps occurring at random time another appropriate model is necessary:

$$\nu(t) = \frac{T(q^{-1})\epsilon(t)}{\Delta} \tag{2.80}$$

where Δ is the differencing operator $\Delta = 1 - q^{-1}$.

Combining equations 2.74 and 2.80 we obtain the CARIMA model:

$$A(q^{-1})y(t) = B(q^{-1})u(t-1) + \frac{T(q^{-1})\epsilon(t)}{\Delta} \tag{2.81}$$

Here several interpretations of the disturbance term $\epsilon(t)$ and polynomial $T(q^{-1})$ in the CARIMA model could be useful for future designs:

- With $T = 1$ and $\epsilon(t)$ a pulse sequence, the disturbance can be considered as random steps acting essentially on the plant input.
- When $\epsilon(t)$ is a sequence of independent random variables and T is considered known or estimated. A predictor based on this model will then have optimality properties.
- $T(q^{-1})$ can be a fixed polynomial (case when not an optimal, but a good design, may well provide properties such as robustness to unmodelled dynamics) see paper [7].

2.4.2 The prediction equations

As in the original GPC derivation an approach is to start with previous the CARIMA model written in the form:

$$A(q^{-1})\Delta y(t) = B(q^{-1})\Delta u(t-1) + T(q^{-1})\epsilon(t) \tag{2.82}$$

Consider now the Diophantine identity:

$$T(q^{-1}) = E_j(q^{-1})A(q^{-1})\Delta + q^{-j}F'_j(q^{-1}) \tag{2.83}$$

from which unique polynomials E_j and F'_j can be obtained for given $A(q^{-1})$, $T(q^{-1})$ and j (Δ is the differencing operator $\Delta = 1 - q^{-1}$).

Multiplying 2.82 by $E_j(q^{-1})$ we obtain:

$$E_j A \Delta y(t+j) = E_j B \Delta u(t+j-1) + E_j T \epsilon(t+j) \tag{2.84}$$

Replacing $E_j A \Delta$ from equation 2.83 gives:

$$Ty(t+j) = G'_j(q^{-1})\Delta u(t+j-1) + F'_j(g^{-1})y(t) + E_j(q^{-1})T\epsilon(t+j) \tag{2.85}$$

where $G'_j(q^{-1}) = E_j(q^{-1})B(q^{-1})$ and $q^{-j}y(t+j) = y(t)$.

By substituting $q = 1$ into equation 2.83 that makes $F'_j(1) = T(1)$ and using induction to prove we can write the following two identities:

$$F'_j(q^{-1}) = T(q^{-1}) + F_j(q^{-1})\Delta \tag{2.86}$$

and:

$$G'_j(q^{-1}) = T(q^{-1})G_j(q^{-1}) + q^{-j}\tilde{G}_j(q^{-1}) \tag{2.87}$$

Therefore equation 2.85 is:

$$Ty(t+j) = TG_j\Delta u(t+j-1) + \tilde{G}_j\Delta u(t-1) + Ty(t) + F_j\Delta y(t) + E_jT\epsilon(t+j) \tag{2.88}$$

and

$$\begin{aligned} y(t+j) &= G_j\Delta u(t+j-1) + \tilde{G}_j u^f(t) + F_j y^f(t) + E_j\epsilon(t+j) = \\ &= \sum_{i=1}^{j} g_{i,j}\Delta u(t+j-i) + p_j + E_j\epsilon(t+j) \end{aligned} \tag{2.89}$$

where p_j, being the quadratic j-step ahead predictor, has the form:

$$p_j = y(t) + \sum_{i=0}^{nf} f_{i,j} y^f(t-i) + \sum_{i=0}^{n\tilde{g}} \tilde{g}_{i,j} u^f(t-i) \tag{2.90}$$

and the $u^f(t-i)$, $y^f(t-i)$ represent filtered signals by a bandpass transfer function from the plant I/O data (so the high frequency noise and disturbance are removed):

$$y^f(t) = \frac{\Delta y(t)}{T(q^{-1})} \tag{2.91}$$

$$u^f(t) = \frac{\Delta u(t)}{T(q^{-1})} \tag{2.92}$$

Even though it seems that the previous equations included into an algorithm will involve much prior computations to evaluate the coefficients f_ij and $\tilde{g}_ij$, calculation will be simplified by using the recursive relation between pairs (E_j, F_j) and (E_{j+1}, F_{j+1}).

Moreover because:

$$TG_j + q^{-j}\tilde{G}_j = G'_j = BE_j = B(T - q^{-j}F_j)/A\Delta \tag{2.93}$$

we have:

$$G_j = \frac{B}{A}\frac{1}{\Delta} - q^{-j}\left[\frac{F_j}{A\Delta T} + \frac{\tilde{G}_j}{T}\right] \tag{2.94}$$

Therefore we are able to simplify equation 2.89 such as:

$$\hat{y}(t+j) = \sum_{i=1}^{j} g_{i,j}\Delta u(t+j-i) + p_j \tag{2.95}$$

because the first j terms of G_j are points on the plant step response model, so that coefficients associated with future controls are stationary. Instead of using this theoretical approach in to compute p_j we will use a recursion algorithm. The main reason for this iterative computation is that p_j is simply the response of plant assuming that future controls are equal with the previous control $u(t-1)$ and the disturbance term $x(t+j)$ is constant, which means that $\epsilon(t+j) = 0$.

To have a global vectorial image about the prediction output equation we will combine all outputs, for $j = 1, \ldots, N_2$, into a vector y:

$$y = G\tilde{u} + p + \epsilon \tag{2.96}$$

where:

$$y = [y(t+1), y(t+2), \ldots, y(t+N_2)]^T \tag{2.97}$$

is the vector of future plant outputs.

$$\tilde{u} = [\Delta u(t), \Delta u(t+1), \ldots, \Delta u(t+N_u-1)]^T \tag{2.98}$$

is the vector of potential future control increments.

$$p = [p_1, p_2, \ldots, p_{N_2}]^T \tag{2.99}$$

is the vector of free response of the plant.

$$\epsilon = [\epsilon(t+1), \epsilon(t+2), \ldots, \epsilon(t+N_2)] \tag{2.100}$$

The matrix G is significant in determining all properties of long range predictive control algorithm:

$$G = \begin{pmatrix} g_1 & 0 & \ldots & 0 \\ g_2 & g_1 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ g_{N_2} & g_{N_2-1} & \ldots & g_1 \end{pmatrix} \tag{2.101}$$

It is interesting to note that increments of control moves after the control horizon N_u are taken to be zero $\Delta u(t+j) = u(t+j) - u(t+j-1) = 0$ for $j > N_u$

2.4.3 The control Algorithm

The future control sequence $u(t+j)$ is chosen by GPC algorithm to minimize a cost-function:

$$J(N_1, N_2, N_u, \lambda) = \sum_{j=N_1}^{N_2} e^2(t+j) + \lambda \sum_{j=1}^{N_u} \Delta u^2(t+j-1) \tag{2.102}$$

where

- N_1 is the minimum prediction horizon
- N_2 is the maximum prediction horizon
- N_u is the control horizon
- λ is the control weighting

and

$$e(t+j) = w(t+j) - \hat{y}(t+j) \tag{2.103}$$

where $w(t+j)$ is the vector of the true set point. The optimization is subject to the constraint that control increments $\Delta u(t+j) = 0$ for $j > N_u$.

Quadratic programming for minimizing J subject to constraints can be used here in order to improve the GPC algorithm and to open up new areas of use.

More it can also be mentioned that another approach that uses a pseudo set-point to make smooth transitions from the current output $y(t)$ to the true set point $w(t+j)$.For a first order approximation this can be achieved by using:

$$w^*(t+j+1) = (1-\alpha)w^*(t+j) + \alpha w(t+j) \tag{2.104}$$

with $j > 0$ and $w^*(t) = y(t)$.

The receding horizon used in the GPC approach is characterized by the following steps pursued at each instant t:

1. given $y(t)$ and previous values of: y, u, Δu the predictions of the free response of the plant are made to form the vector e by comparison with w.

2. for the given N_1, N_2, N_u, λ, e the vector $\tilde{u}$ is calculated by minimizing J.

3. only the first element of the command: $u(t) = u(t-1) + \Delta u(t)$ is applied and the whole sequence is repeated for the next sample interval.

It is well to note that the prediction equation with the assumption present in the cost-functional 2.102 can be written as:

$$\hat{y} = G\tilde{u} + p \tag{2.105}$$

where

$$\tilde{u} = [\Delta u(t), \Delta u(t+1), \ldots, \Delta u(t+N_u-1)]^T \tag{2.106}$$

is the vector of potential future control increments

and G is a $\Re^{(N_2-N_1+1)\cdot N_u}$ matrix with $g_{i,j} = 0$ for $j - i > N_1$:

$$G = \begin{pmatrix} g_{N_1} & g_{N_1-1} & \cdots & \cdots & 0 \\ g_{N_1+1} & g_{N_1} & g_{N_1-1} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_{N_2} & g_{N_2-1} & \cdots & \cdots & g_{N_2-N_u+1} \end{pmatrix} \tag{2.107}$$

Substituting the equation 2.105 in the equation 2.102 that define the quadratic cost functional we obtain:

$$J = (w - G\tilde{u} - p)^T (w - G\tilde{u} - p) + \lambda \tilde{u}^T \tilde{u} \tag{2.108}$$

In the unconstrained case due to the least square approach we obtain the optimal control as:

$$\tilde{u}_{opt} = (G^T G + \lambda I)_{-1} G^T (w - p) \tag{2.109}$$

where $G^T G \in \Re^{N_u \cdot N_u}$

The invertibility of matrix $G^T G + \lambda I$ is crucial for a feasible control optimization. Control weighting λ is hard to determine apriori but a good choice would be to make the previous matrix invertible as well to improve the control performance.

The computation of GPC according to the author of paper [7] is as follows:

1. computation of the step response model
2. computation of the free response of the system based on initial conditions of the plant
3. computation of the j step ahead predictor
4. computation of the command $\Delta u(t+j)$

Choice of output and control horizons

Using simulation examples on a variety of plant models, including stable, unstable and nonminimum-phase processes, provided in Clarke's papers, we are able to give little advice about choosing output and control horizons.

- For *the minimum input horizon* N_1 and models with known dead-time l it is well to have $N_1 = l$ in order to avoid superfluous calculations. For variable l, N_1 can be set to 1 with no loss of stability.
- *The maximum output horizon* should be chosen in such a way that N_2 will be equal with the degree of $B(q^{-1})$ matrix.
- *The control horizon* is a very important parameter that can be chosen in various ways depending on the particular situation. For simple plants a value of 1 is appropriate for N_u; for complex plants increased values are more suitable. For high performance applications such as flight control a large value of N_u is desirable.

2.5 Internal model controller approximation used in MBPC

Internal Model Control (IMC) as an alternative to solve predictive control problems was initiated by Garcia and Morari [5].

The main idea behind IMC is that a feedback control system, see figure 2.4, is restructurated into the IMC form, see figure 2.3,where:

- y – plant output
- y_m – model output
- S_i – vector of set points
- e – error between y and y_m

The internal model controller (IMC) is defined by:

$$Q_i(z) = \left[I + C_i(z)\tilde{G}_i(z)\right]^{-1} C_i(z) \tag{2.110}$$

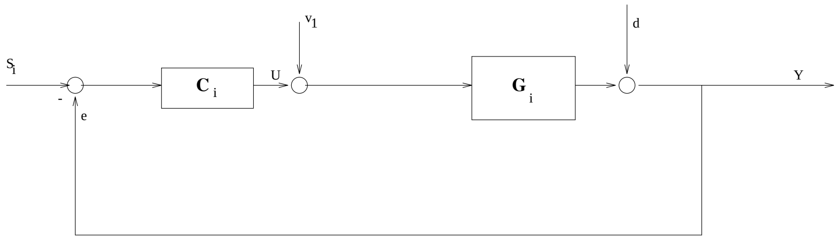

Figure 2.3: Feedback Closed Loop

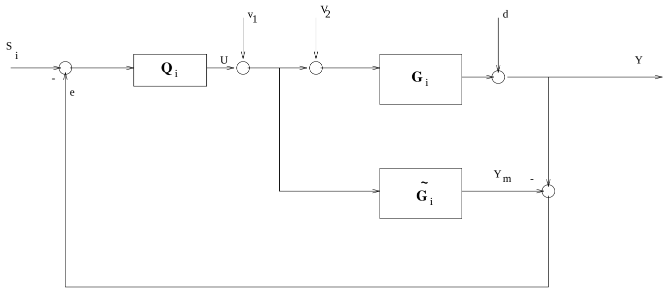

Figure 2.4: Internal Model Structure

where $C_i(z)$ is the feedback controller.

It is important to note that in the case of predictive control the IMC consists of both the optimizer and the predictor.

The controller and feedback loop according to Maciejowski and Heise [20] possesses several useful characteristics, assuming a perfect model $\tilde{G}$ for the plant:

dual stability: For a perfect model a sufficient condition for system stability is that both plant and model are stable. This is evident examining the closed loop transfer matrices for $G = \tilde{G}$:

$$\begin{pmatrix} y \\ u \\ y_m \end{pmatrix} = \begin{pmatrix} G_iQ_i & G_i & (I - G_iQ_i)G_i \\ Q_i & 0 & -Q_iG_i \\ G_iQ_i & G_i & -G_iQ_iG_i \end{pmatrix} \begin{pmatrix} S_i \\ v_{1i} \\ v_{2i} \end{pmatrix} \tag{2.111}$$

perfect controller: Examining equation:

$$y_i(z) = G_i(z)\left[I + Q_i(z)(G_i(z) - \tilde{G}_i(z))\right]^{-1} Q_i(z)(S_i(z) - d_i(z)) + d_i(z) \tag{2.112}$$

and having $Q_i(z) = \tilde{G}_i^{-1}(z)$ it can be seen that $y_i(z)$ will follow $d_i(z)$ which means that IMC has the inverse structure of the model. Stability of $G_i(z)$ and $Q_i(z)$ is necessary and sufficient for stability of the "perfect controller". In practice $Q_i(z) = \tilde{G}_i^{-1}(z)$ is usually infeasible.

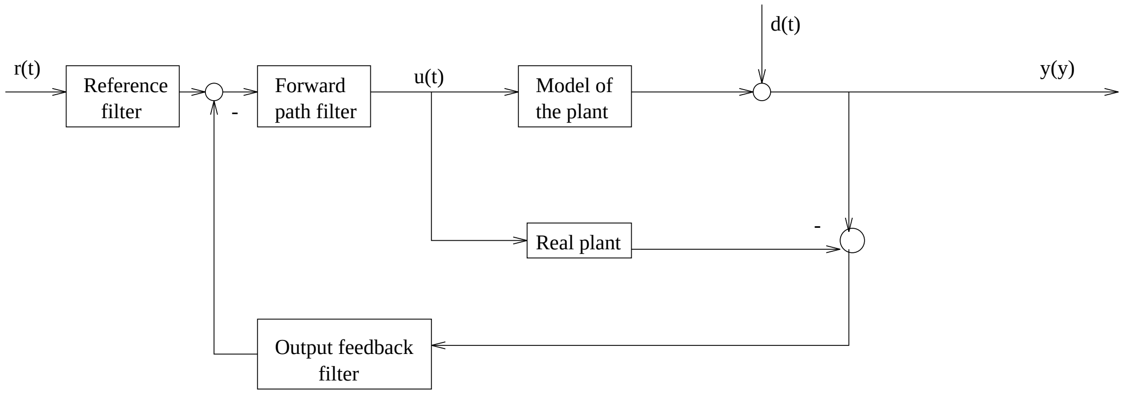

Figure 2.5: Internal Model Controller Structure

zero offset

According to Morari [5] the motivation behind using the IMC was to take advantage of the unconstrained MBPC schemes :

- tuning on-line via adjusting meaningful physical parameters
- no concern about closed loop stability
- good performance
- the ability to cope with other inputs than steps

and to avoid their disadvantages.

The design procedure described in the paper [5] consists of two main steps:

1. Design a pre IMC controller $\hat{Q}$ to have a good response without regard for constraints on manipulated variables and robustness.
2. The real IMC controller is designed by augmenting the pre IMC $\hat{Q}$ with a low pass filter, the parameters of which are adjusted either off or on-line to reduce the action of controlled variable and to improve robustness $Q = \hat{Q}F$.

The classical block diagram approach for either unconstrained or constrained method is the two degrees of freedom internal model controller structure, see fig 2.5. Filters, in this scheme, are included in order to fine tune the closed loop.

There is much freedom in order to choose the filter such as a diagonal matrix with one tuning parameter. Usually this is adequate enough to achieve satisfactory response and robust characteristics. In paper [5], for special cases, a gradient search procedure is developed. Nowadays the development of a more reliable technique is the subject of current research.

2.6 Piecewise LTI models and control laws

By a model set we understand that the nonlinear system is approximated by a finite number of linear models that compose this set. These models obtained via small perturbation algorithm are successful in describing the plant, even the failed one. So, we assume that for an aircraft we will have this set of models describing the whole flight envelope plus several failure configurations. Constraints are associated with a particular model remaining unchanged as long as that model is employed. As a result, the control law is LTI (linear time invariant) as the set of of active constraints remains unchanged. In conclusion the control law has a variable structure and the property to switch between a finite number of control laws.

In order to prove the claim made in the above paragraph that MPBC laws are piecewise linear time invariant with switches between LTI laws occurring when the plant model or corresponding active constraint or setpoints or cost function changes, we present the following demonstration:

Supposing that the plant is associated with a particular set of active constraints:

$$\mathcal{D}_i \cdot \Delta U_i(k) \geq \mathcal{E}_i \cdot \begin{pmatrix} u_i(k-1) \\ \hat{x}_i(k) \\ y_i(k) \\ c_i(k) \end{pmatrix} \tag{2.113}$$

and using information from Fletcher's book [15] when there is equality in the above linear inequation, the optimal solution is given by :

$$\begin{pmatrix} 2 \cdot \mathcal{A}_i & -\mathcal{D}_i^T \\ -\mathcal{D}_i & 0 \end{pmatrix} \cdot \begin{pmatrix} \Delta U_i^*(k) \\ \lambda^* \end{pmatrix} = -\begin{pmatrix} -\mathcal{B}_i^T \cdot \left[\hat{x}_i(k)^T, y_i(k)^T, \mathcal{S}_i(k)^T\right]^T \\ \mathcal{E}_i \cdot \left[u_i(k-1^T), \hat{x}_i(k)^T, y_i(k)^T, c_i(k)\right]^T \end{pmatrix} \tag{2.114}$$

where $\Delta U_i^*(k)$ is the optimal control sequence over the output horizon and λ^* is a set with a Lagrange multipliers vector, corresponding to each optimization of the $J_i(k)$ cost function. If the solution of the optimization problem is unique and the optimization is feasible then the inverse of the matrix:

$$\begin{pmatrix} 2 \cdot \mathcal{A}_i & -\mathcal{D}_i^T \\ -\mathcal{D}_i & 0 \end{pmatrix}^{-1} = \begin{pmatrix} \mathcal{L}_{1i} & \mathcal{L}_{2i} \\ \mathcal{L}_{3i} & \mathcal{L}_{4i} \end{pmatrix} \tag{2.115}$$

exists and the optimal solution becomes:

$$\begin{aligned} \Delta U_i^*(k) = -\mathcal{L}_{1i} \cdot \mathcal{B}_i^T \cdot \left[\hat{x}_i(k)^T, y_i(k)^T, \mathcal{S}_i(k)^T\right]^T + \\ + \mathcal{L}_{2i} \cdot \mathcal{E}_i \cdot \left[u_i(k-1), \hat{x}_i(k)^T, y_i(k)^T, c_i(k)^T\right]^T \end{aligned} \tag{2.116}$$

If the optimization is repeated each step under the stated assumptions the feedback law is linear time invariant. It has been noted by Maciejowski [29] that it is plausible that a system will be stable if each of its constituent LTI laws gives closed loop stability and the switching between laws occurs rarely enough. Therefore one way to obtain stability for such systems is to choose tuning parameters such that constraints will not be activated.

2.7 Further comments on MBPC methods

All MBPC methods lie under the umbrella of two different types of control horizons *"The finite horizon assumption"* and *"The infinite horizon assumption"*. The control horizons have major influences on the MBPC method and their numerical implementation.

The essence of *"receding horizon control"* is to determine a control profile that optimizes an open-loop performance objective on a time interval extending from the current time to a defined future time called the prediction horizon and then implementation of the control profile. The feedback is included by using the output measurement to update the optimization for the next step. One of the major drawbacks of the receding horizon technique is poor stability of the control laws. One method to achieve stability is to add terminal constraints that will bring the states to zero at the end of the control horizon, when the response of the system becomes free. This approach is restricted to controllable and stabilizable systems.

The main features of *"The finite horizon assumption"* according to various authors are:

simpler computation: in the unconstrained cases the use of the least square technique is simpler than solving Riccati equations. For the constrained case new algorithms like the "Method of Analytic Centres" or the "Quadratic Programming Algorithm" solve the problem fast enough.

constraints: because at this moment this is an active research area constraints on both controlled variables and process outputs are difficult to manage in an infinite horizon setting. When the state is far from the origin we should consider a constrained problem with a finite horizon. As the state approaches the steady zero value the algorithm should switch to an infinite horizon cost function, unconstrained, which is in fact a Riccati problem.

tuning flexibility: the variable horizon may represent another tuning parameter to achieve performance and stability. Sometimes such tuning is difficult because of the lack of guarantee of stability or the unknown effect of increasing a certain parameter (such as the input weighting).

The basic idea for *"The infinite horizon assumption"* is that, for a problem which assumes no major noise or disturbance and measurable states, it is possible to guarantee a stable solution of the system controller, the solution being obtained as a result of the quadratic program defined. If the problem is not feasible of course the solution consisting of the set of future control movements it is not defined and an alternative problem, with relaxed constraints, has to be defined.

A new approach due to Mayne and Michalska proposes a formulation for nonlinear or linear plants that switches to a stabilizing linear controller after the state is brought near the steady position. The objective function will be a receding horizon portion for the nonlinear system and an infinite horizon portion for the approximately linear system around the steady state.

Assuming that every variable needed for control and optimization is reachable through direct measurement or estimation and transforming all performance criteria into mathematical expressions it follows that any of the above stated criteria can be mathematically stated as being one of the following two types: *objectives* - functions of variables to be optimized dynamically and *constraints* - function of variables to be kept within bounds. Such constraints in turn can be: *hard constraints* - no violation of bounds are allowed at any time and *soft constraints* - violation of several bounds can be allowed temporarily for satisfaction of other criteria.

In this report several methods dealing with both types of control horizons are presented, methods that share certain similarities, as well as differences.

The DMC algorithm deals equally well with and without constraints. For the unconstrained case DMC has a simple least square solution. This is not the case when there are constraints, when, it requires quadratic programming (QP). In practice it seems that DMC is usually implemented including a linear programming (LP) algorithm rather than a QP.

Another facility of the algorithm is that during operation, the process constraints limit can be changed in response to instrument or plant failure. It is assumed that the stable inverse model of the plant represents a reasonable controller for it, that, obviously, requires optimization. After the controller synthesis using DMC its stability is checked via simulation of the closed loop.

Implementation of DMC is characterized by three main steps that ensure success in most of the cases:

1. *Identification of the process model*: For any type of model, this model should relate to all manipulated variables as well the states or outputs depending on the formulation adopted (input/output or state space). In order to deal with disturbances using a feedforward compensation a model is required.

2. *Off line design*: In any off-line environment only a nominal design can be performed. Stability and performance of the nominal DMC controller are verified via simulation simultaneously. As mentioned above, stability in the control law can be verified off-line by imposing various conditions and moving horizon simulation tests. During operation imposing very hard constraints to the process may result in instability.

   DMC can guarantee the optimal response in the face of input constraints according to [35]. Constrained control has nonlinear characteristics, therefore stability in the face of output constraints becomes an issue. So, in such a case, because of nonlinear behaviour, it is fairly complicated to obtain rigorous stability tests. Only for few particular cases is such a test performed nowadays. At this stage stability in front of process constraints is very hard to define.

   For the the assumption of complete knowledge of the model and its inputs and off-line receding horizon, simulation is very helpful as long as initial conditions match the plant. In practice it is well to perform a series of simulation case studies in which possible initial conditions are considered. Weights in the objective function are changed accordingly to obtain desirable dynamic responses. Frequency response analysis can be performed here to verify disturbance rejection property.

3. *On line implementation*: The tradeoff criteria between stability and performance is reached on line by tuning the controller, so, weights in the cost function can be updated on-line to affect the quadratic program solution. Here useful knowledge of how model

parameters will change with operating conditions is obtained, so that, either weights of even step response model coefficients, in the case of using such a model, can be adapted on-line. Alternatively in the absence of a model dependent on the operating point conditions one could have an on-line identification routine, continuously updating the linear plant model, and in such a way an adaptive controller will result.

The algorithm is used to produce the whole control sequence required but only the first element of the solution is applied.

GPC as a control algorithm depends on integration of several ideas:

- assumption of CARIMA rather than CARMA plant model
- the use of long range prediction over a finite horizon (grater than the dead time of plant and least equal with the model order).
- recursion to the Diophantine equation
- the consideration of weighting the control increments in the cost function
- choice of control horizon after which projected control increments are taken to zero

Because many of these ideas have arisen in the literature in one form or another the author of GPC tries to consider these approaches as subsets of the GPC approach in such a way that theoretical results can be extended to this new method. Papers [7], [6] presented by Clarke describe GPC as a new robust algorithm suitable for challenging adaptive control applications. The author of GPC states also that the method is simple to derive and to implement. Another interesting idea is that for short control horizons where the GPC algorithm can be mounted in a microcomputer. Simulation results show that GPC can cope with control of complex processes under realistic conditions even where the design phase is not so prolonged.

For a controlled model that is linear time invariant (LTI), discrete time, with known model parameters there are multiple formulations available such as: state space, preferred for most cases, and input-output, using transfer functions or matrices. At a certain extent methods presented are similar. For example, even though different models are used for prediction, there is a possibility to convert them in such a way that a state space formulation can be employed. This remark covers most of the algorithms with an exception due to the CARIMA model used in GPC that permits the use of the Diophantine identity. Regarding the control and prediction horizons there are small differences between algorithms. Some authors relate the values for such horizons to different simulation or theoretical results. The possibility to use estimation inside the optimization algorithm is clear and easy within state space framework. Weighting matrices in the cost function are used in a similar way for all methods even though there are some differences regarding the values included. The off-line computation is compulsory for most of the algorithms to speed-up the control and ensure real-time implementation. For each process identification used and corresponding model there is a suitable MBPC method.

Constraints are handled in a similar manner by most of MBPC algorithms. Stacking them with the purpose to use standard QP routines become an usual habit. For the GPC method even initially the algorithm did not address this issue now it is possible to precise it within the optimization. The idea of making the problem mathematically manageable resides from the fact that for example to deal with MBPC with constraints there is a need for a quadratic programming method that solves efficiently such a problem with a standard software. The conversion of constraints into objective criteria is useful from the control point of view because actual hardware deals more easily with even a complicated constrained cost function then directly with constraints. Another reason is that for example in H_∞ technology, because every thing is developed in a frequency domain it is hard to handle explicitly time domain constraints.

Model based predictive control solves on-line a constrained optimization via QP at each sampling time. The optimization is carried out over the prediction horizon assuming that no feedback is used during this future period. Applying only the first sequence of optimal future inputs, computed via QP, results in closed loop behaviour. This behaviour is quite different from the one predicted by the optimal solution which assumes uninterrupted open-loop implementation. It was shown in the literature that output constraints can result in instability for processes with model error or discrete models with unstable zeros, see paper [43].

As shown in section 2.6 constrained MBPC algorithms with linear models are controlled by piecewise LTI laws. Conform [44] such a framework, as the one presented in section 2.6, can be used for tuning MBPC parameters to guarantee robust stability by designing of a set of linear feedback controllers, each corresponding to a constant combination. The use of hard constraints restricts the degree of freedom available to the designer. When a particular combination of constraints result in instability often the only action is to remove or relax them during the on-line optimization. This is usually the case with output constraints placed on the first few points of the on-line optimization.

A short survey upon remarks that involve stability and robustness of MBPC systems is presented in order to give a complete idea of results achieved:

From the controllable point of view paper [41] shows that a discrete-time controllable linear system is globally controllable to the origin, with bounded inputs, if and only if A is stable.

Paper [32] differentiates the infinite and finite horizon approaches from the stability point of view. A complete method for unstable systems is provided assuming that the input sequence drives the predictions of the model unstable modes to zero after $k+N_u$ steps, when the change in control variable becomes zero. Otherwise, if unstable modes are not zero, they evolve uncontrolled and do not converge. Paper [33] states that for positive definite matrices the QP is strictly convex which implies that the solution is unique and global. In [36] it is explained that an MBPC with an infinite horizon can globally stabilize a linear discrete-time system, provided that the QP is feasible.

In a 1992 paper of Zafiriou [43] it is mentioned that the presence of hard constraints in the on-line optimization procedure often produces a nonlinear controller. There is difficulty in the study of stability properties because of the lack of analytical expressions available for the controller. The main conclusion of that paper is that in order to increase the applicability of the "contraction mapping method" it is necessary to develop a computationally feasible search procedure of finding and testing all control functions that occur during the on-line optimization. The author only deals with some objective functions subject to particular hard constraints.

By considering QP problems, as a particular set of nonlinear programming problems, we can emphasize a few points. QP problems are defined as maximizing the following quadratic objective function:

$$F(x) = a_0 + c^T x + x^T Q x \tag{2.117}$$

subject to the linear inequality constraints:

$$\begin{aligned} a^T x &\geq b \\ x &\geq 0 \end{aligned} \tag{2.118}$$

.

Special cases of nonlinear problems involve variables that are in a certain set (e.g. nonlinear integer programming) or constraints that depend on time in the form of differential equations (e.g. optimal control, dynamic optimization). The convexity of QP problems assists in determining under what conditions a local optimal solution is also the global optimal solution.

An important result in mathematical programming evolves from the concepts of convexity, so, if the nonlinear programming problem, known as the convex programming problem, has following properties:

1. F(x) is a convex function
2. each inequality constraint is a concave function (the constraints form a convex set)

then the local minimum is also the global minimum. In LP problems the objective function is always convex, and the constraints form a convex set, so, the local optimum is always the global one.

For the QP problem the objective function is convex if $x^T Q x$ is positive semidefinite hence Q must be positive semidefinite. Assuming that the QP problem has a convex objective function the second requirement is to have constraints forming a convex set.

The feasible point or vector of a QP problem is any vector that satisfies the equality and inequality constraints. The set of all vectors that satisfies the constraints constitutes the feasible domain of the function. A constrained optimum is one in which the local optimum lies on the boundary of the feasible region. In relation with just the inequality constraints a point may be classified as an interior point (a feasible point) or an exterior point (a non feasible point). So, an interior point is one for which all the inequality constraints are strictly respected. The set of points for which the objective function has a constant value is called a contour of the cost function.

In the case of nonlinear programming with inequality and equality constraints we define a point to be a local minimum if the objective function cannot decrease along any smooth arc directed from that point into the feasible region.

Because linear programming methods have been successfully applied to large dimensional problems (containing linear inequality constraints) the liniarization of the nonlinear problem seems to be a sensitive approach in solving such problems.

2.7.1 Comparison of MBPC methods presented in the report

We intend to develop the study regarding the MBPC methods presented in the report:

- The input/output formulation
- The state space formulation
- The general predictive control method

Few criteria will be used such as: model employed, assumptions made, the unconstrained and constrained case and the objective function.

For each method following models are used:

The step response model:

-easy to be obtained via identification

-could be produced from state space descriptions

-use simple matrix computation in an iterative manner that could simplify the algorithm

-by using the backward shift operator it is possible to write the model in a more compact form emphasizing the free response of the system

The state space model:

-useful because most of mathematical models used for design produce state space description -prediction and estimation can be handled in an integrate environment

-the matrix form represent an easy way to compute and improve QP algorithms

The CARIMA model:

-evolute from CARMA model

-the disturbance is considered to be of a moving average form

-the difference from the step response model is given by the particular form required for the use of Diophantine equation for prediction

Regarding the assumptions made in every case (method) some ideas can be stated:

For the I/O formulation:

-the assumption of an asymptotically stable process is necessary to simplify the free response of the system up to a manageable form

-a strong assumption is made upon the future values of the disturbance that is approximated with the current value in the process model

For the state space formulation:

-the assumption of a stable process is not necessary for computations to be done

-state space models give opportunity to model disturbance in either a deterministic or a stochastic sense

-matrices involved in prediction are derived from the open loop model depending on the model set, the time in the future but not the current time, so, few simplification are made

For the GPC:

-the Diophantine identity is used to predict several steps ahead the system response

-recursive relation that simplify computation are used

-disturbances are either approximated as random steps occurring at random time or considered to be of moving average form

In both unconstrained and constrained case there are particularities corresponding to each method, the overall characteristic being the similarity between them:

For the I/O formulation:

-least square method is used to produce an analytical solution

-using this approach stability can be analysed and few results have been produced already

-constraints upon states are impossible to be handled using this formulation

-in order to simplify computation a stacking of the constraints is used for a compact form

For the state space formulation:

-the least square method produces a nice analytical solution using the state space model

-constraints upon states can be handled with this approach in a convenient manner

-packing constraints in a compact vector is a standard procedure

For the GPC:

-classic GPC algorithms do not develop a particular section to handle constraints but inequality constraints can be written in a similar form as in the input/output or the state space formulation

A few remarks will help to make a comparison of the three approaches with respect to the objective function and control algorithm:

For the I/O formulation:

-minimization of the objective function is done over the prediction horizon

-a weighting factor upon the manipulated variable is used

-by writing the cost function in a matrix form, the vector of future controls (including increments, rates and outputs) can be used directly into the optimization

-a QP algorithm is used, so, several assumptions for semipositivity of the matrices involved were made

For the state space formulation:

-only two horizons participate in the optimization

-several terms independent of the vector of future controls are omitted from the matrix form involved in the optimization.

-several rules of thumb (checked via simulation) produce useful values for the control and prediction horizon

For the GPC:

-classic GPC (without constraints) involve as well a reduction to the least square algorithm

-in the constraint case several linear matrix inequalities are employed and the solution of the optimization is produced using quadratic programming

-the choice of horizons, that depends on the CARIMA model and the complexity of the controlled plant, is very important

Chapter 3

Design examples

3.1 Chemical process automated with an MBPC

Simulation examples provide an accurate image about the MBP control of the SISO and MIMO systems. SISO case was developed in order to prove the algorithm efficiency and MBP control method performance. The SISO case as in many control techniques represents the first step toward the multivariable case.

All diagrams illustrate responses obtained using a flexible "home" software connected with Simulink, the Matlab extension used to simulate dynamical systems. Figure 3.17 presents Simulink test framework that provided us with a complete tool for MBP control problems. The complete algorithm for the general case of a MIMO system is described from the theoretical view point in section 2.3 and is presented in section 3.2.

As shown in section 2.3, that presents the theoretical base of the algorithm, an on-line computation using a quadratic program is carried out. Because real time implementation is carried out it is necessary to have an efficient algorithm. The speed of this algorithm is mainly affected by the dimension of a wide variety of tuning parameters each with a precise influence upon the stability and performance of this control method. Parameters like prediction and control horizon influence the dimension of the matrices involved in prediction/estimation in an increasing sense. Simulations reveal that large horizons are time consuming from the computation point of view without major improvements in tracking of a set point comparative with considerable small horizons that affect the closed loop system from the stability point of view. Due to this fact an on-line computation of matrices involved in the algorithm (especially connected with estimation/prediction process) must be avoided and corresponding off-line computations are recommended even when a change in the LTI model used occurs.

Decision time for a possible failure in the real system or a change in the trim point represent an extra time that should be added to the previous evaluated computation time. In such cases (eg. an actuator failure) it is better to change the vector of constraints associated with the model than to alter the whole model used in the optimization.

Weighting matrices used in the algorithm for Kalman filter design and the cost function expression represent sensitive tuning parameters that could influence the solution of the QP algorithm. An inaccurate slow prediction as well as an unappropiate influence of the set-point tracking error or command increment may produce different results than those expected.

Constraints and hardness or softness of these play an important role, so, various configurations were checked. Output response problems could appear when the constraints are enforced as in the case when the dynamics of the real system are changed in such a way that severe increments are required.

The long simulation time suggests the requirement of a fast computer. For the real time implementation on a plant of MBPC algorithm advanced DSP boards that have a user friendly interface with Matlab and even Simulink are needed. From the software point of view a complete package has to have features mentioned in the section 4.1.2.

Often for aerospace problems that will represent the next step for future research we will take advantage of the assumption of accessible states for measurement that will simplify the control structure of the estimator/predictor. Such a reduction will normally speed up the algorithm by avoiding estimation computing time.

Concluding strictly upon the process case it is important to precise a few details about an Evaporator which performs an industrial process namely to evaporate solvent from a feedstream.The main variable which needs to be controlled is the 'Product Composition', which we will call $X2$. Keeping variations in the composition as small as possible maximizes the profitability of the Evaporator, because it minimizes production of out-of-spec product, while minimizing production costs, since it is possible to aim for a composition only a little better than the minimum acceptable one. However, it is also necessary to operate the Evaporator safely, and without damaging the installed equipment. This requires the pressure in the Evaporator ($P2$), and the level of liquid in the separator ($L2$), to be controlled. There are three variables which we assume we can adjust, in order to control the Evaporator: the mass flow rate of the product being drawn off from the recirculating liquor ($F2$), the pressure of the steam entering the Evaporator ($P100$), and the mass flow rate of the cooling water entering the condenser ($F200$). These three variables will be our 'control inputs'. There are several other variables which affect the Evaporator's performance, but we assume that these are not under our control. These will act as 'disturbances' which will keep pushing the controlled variables away from their target values. In particular, the Feed flow rate ($F1$), the circulating flow rate ($F3$), the Feed composition ($X1$), the Feed temperature ($T1$), and the cooling water inlet temperature ($T200$) all give the process significant disturbances. The figure 3.1 represent the nonlinear model of the Evaporator described using the Simulink package.

The system used in MBPC algorithm is an LTI system discretized and linearized using Matlab functions:

```
% Pair of commands and states around the liniarization is made

Uo=[0;
    0;
    0];

Xo=[ 1.0;
    25.0;
    50.5];

% Trim function used to get the exact trim point
```

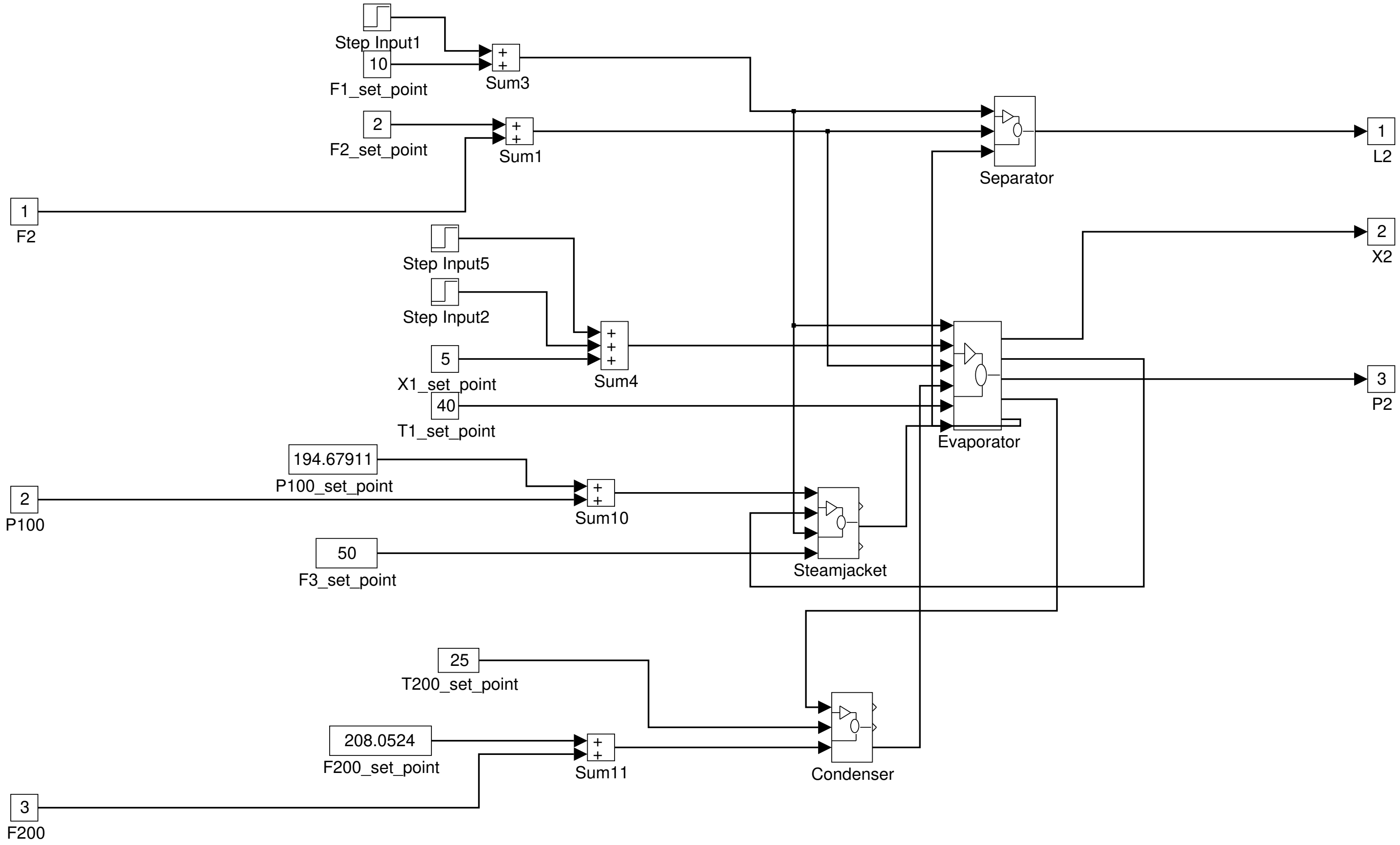

Figure 3.1: Nonlinear Model of The Evaporator

```
[X,U,Y,DX] = trim('PLANT_Evap',Xo,Uo,[1 2 3],[1 2 3]);

% Linmod function help us to get the linear model
%at a sample time = 1 time unit

[A,B,C,D] = linmod('PLANT_Evap',X,U);
```

The state space form of the linearized system of the simplified nonlinear Evaporator (see figure 3.1) at a sampling rate of $T_s = 1$ time unit is:

```
% AA,BB,CC,DD represent the discrete model

AA = [ -0.001    0.0042    0.0075;
        0.000   -0.1000         0;
        0.000   -0.0209   -0.0558];

BB = [-0.0500   -0.0019         0;
      -1.2500         0         0;
            0    0.0096   -0.0018];

CC = [ 1.0000         0         0;
            0    1.0000         0;
            0         0    1.0000];

DD = [  0     0     0
        0     0     0
        0     0     0];
```

The model contains three states which are as well measured outputs:

State variable	Set point
separator level L2	1 m
product composition X2	25%
operating pressure P2	50.5 KPa

Most of the constraints as explained are due to physical limitations, we refer to control constraints:

Control variable	Constraint
product flow rate F2	unconstrained
steam pressure P100	$\leq$ 400 KPa
cooling water flowrate F200	$\leq$ 400 Kg/min

The nominal values for disturbance variables are:

Disturbance variable	Operating value
circulating flowrate F3	50 Kg/min
feed flowrate F1	10 Kg/min
feed composition X1	5%
feed temperature T1	40°C
cooling water inlet temperature T200	25°C

In order to prove the reliability of the algorithm used and support the various remarks in the report, presented here are several simulation results. The software and model used are described in sections 3.2 and 3.1 respectively.

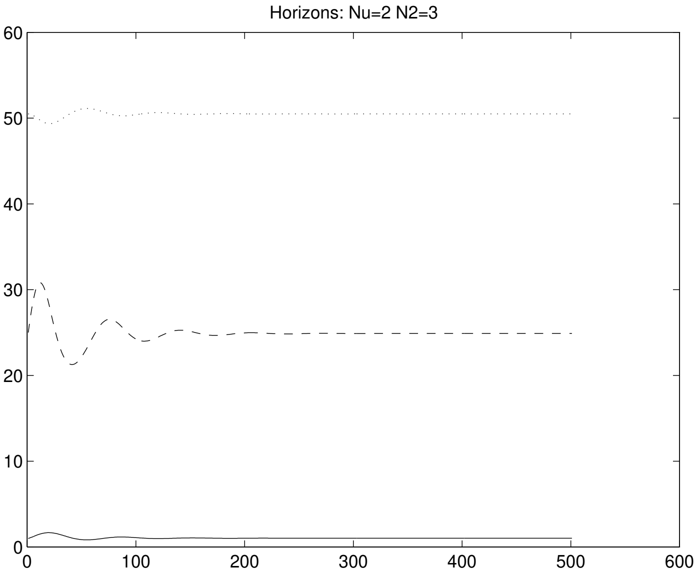

Figure 3.2: Horizon influence upon output response in unconstrained case

Few horizons in the unconstrained case were tested (see figures 3.2, 3.4 and 3.5) and results proved that suitable values for the respective control prediction horizons are around $N_u = 2$ respective $N_2 = 3$. These values correspond with those from other papers (see [7], [31] and [1]). With the aim to give a complete image of the efficiency of the control algorithm the graphs of control increment for suitable horizons are presented (see figure 3.3).

Further tests related to the steady state achievement (see figures 3.6, 3.7 and 3.8) were pursued assuming values $N_u = 2$, $N_2 = 3$ for the control and prediction horizons. In these cases two kinds of weighting matrices were tested in order to show their influences upon steady state achievement.

MBPC methods give different responses in the unconstrained and constrained case, so, these two cases were simulated and corresponding results presented. Set points during these simulations are shown as well in figures 3.9 and 3.11. For the command increment we need to add two more self explanatory graphs 3.10 and 3.12.

Most of the processes require a state estimate to be controlled even by MBPC. It is this fact that imposes the use of a Kalman estimator or of an observer in the MBPC schemes. Pole allocation for the filter, via weighting matrices is extremely important because of the required estimate accuracy. Two different values for these matrices were used during the tests and corresponding estimates are presented in figures 3.13 and 3.14. The poles for the $A - K{\cdot}C$ matrix are shown in the appendix.

The MBP control algorithm should provide satisfactory disturbance rejection, as presented in figures 3.15 and 3.16. The measurement of the input and output is affected in practice by noise that influences the behaviour of the estimator, so, adequate weighting matrices are used.

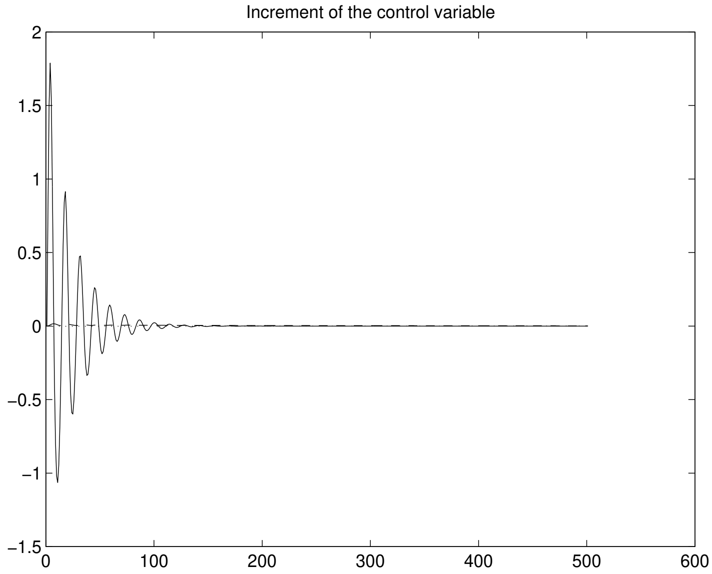

Figure 3.3: Increment of control variable for $N_u = 2$ and $N_2 = 3$ in unconstrained case

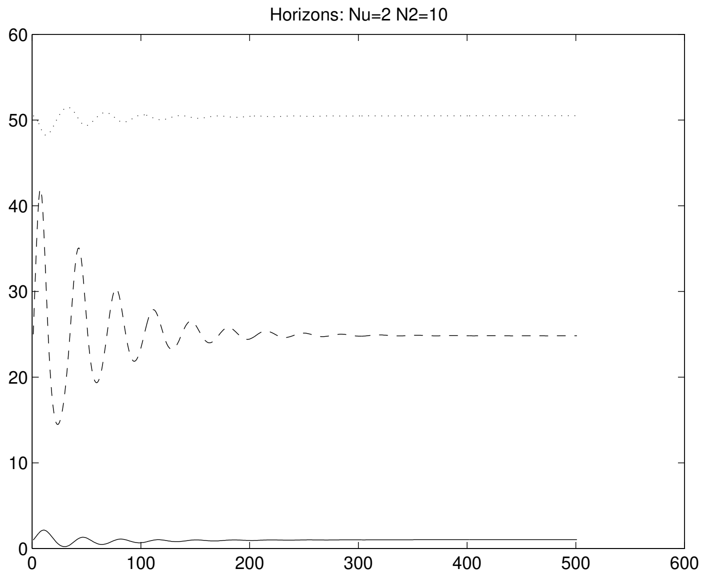

Figure 3.4: Horizon influence upon output response in unconstrained case

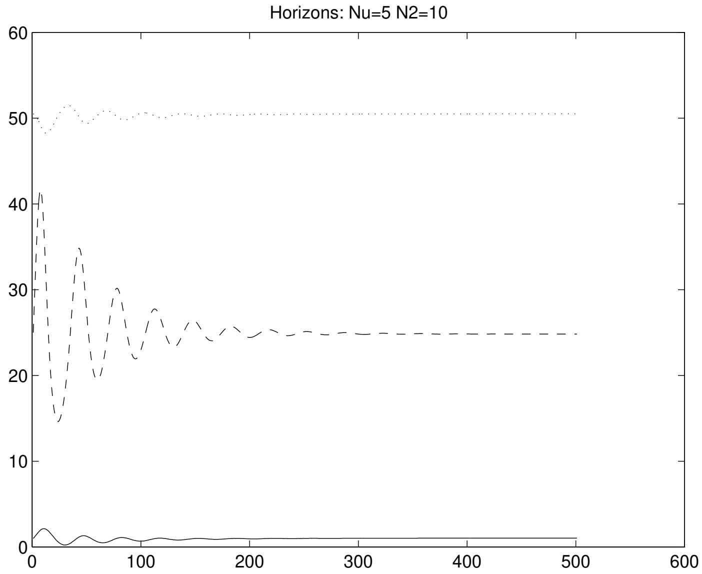

Figure 3.5: Horizon influence upon output response in unconstrained case

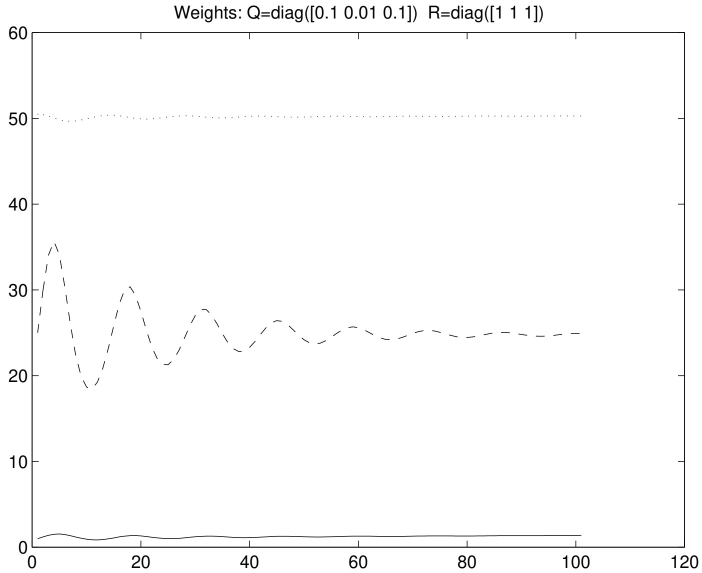

Figure 3.6: Influence of cost function weighting matrices upon output response in unconstrained

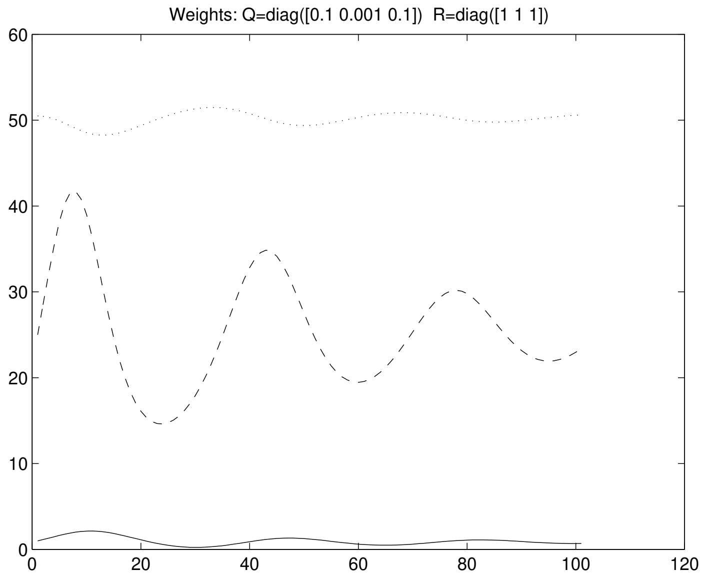

Figure 3.7: Influence of cost function weighting matrices upon output response in unconstrained case

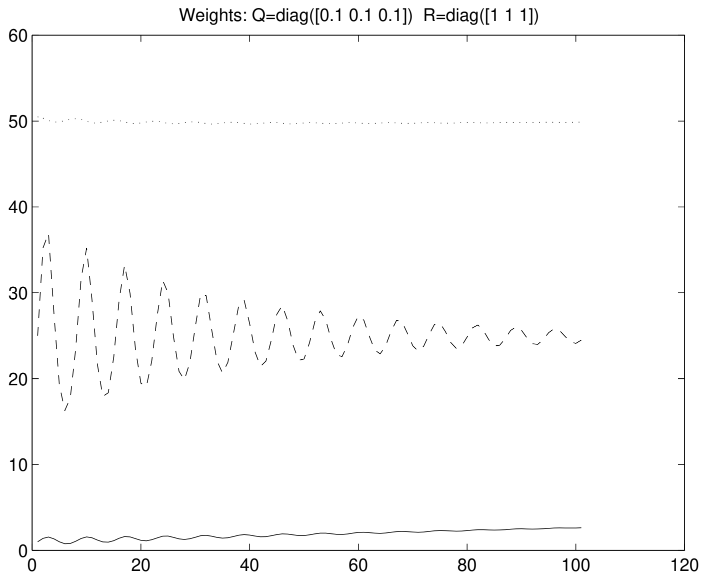

Figure 3.8: Influence of cost function weighting matrices upon output response in unconstrained case

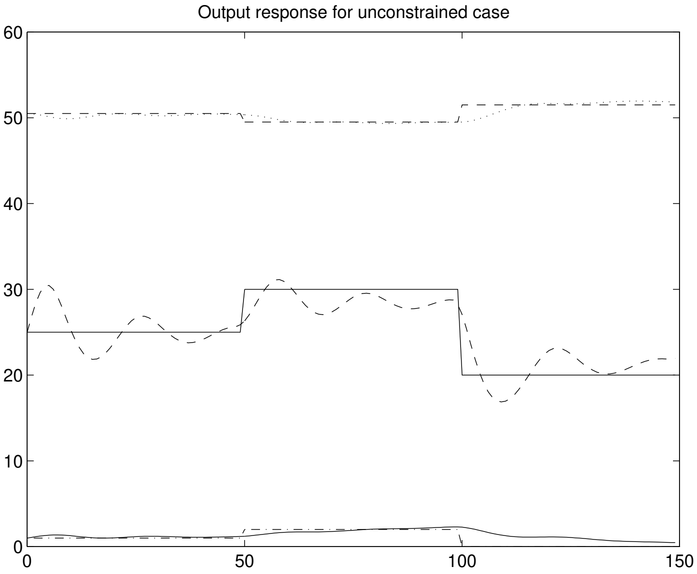

Figure 3.9: Output response in unconstrained case for specified set points

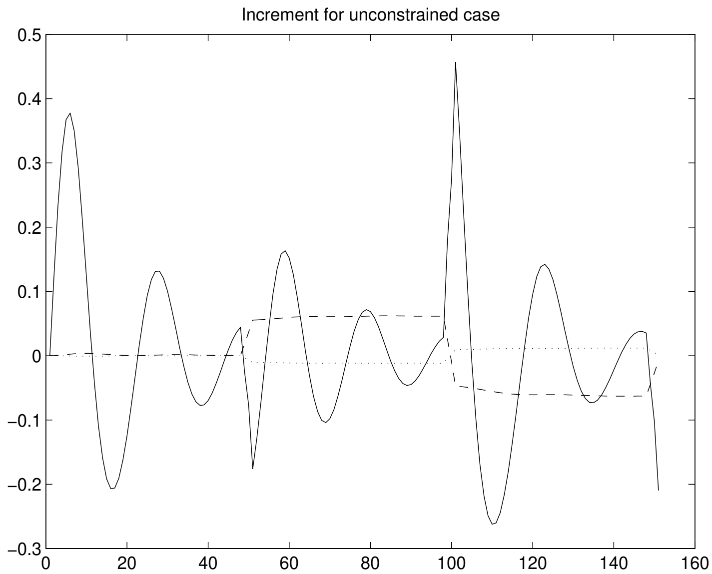

Figure 3.10: Increment of control variable in unconstrained case

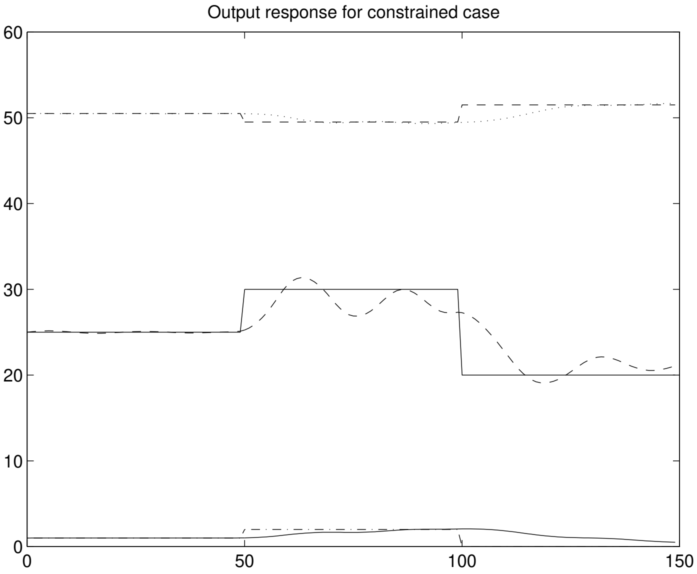

Figure 3.11: Output response in constrained case for specified set points

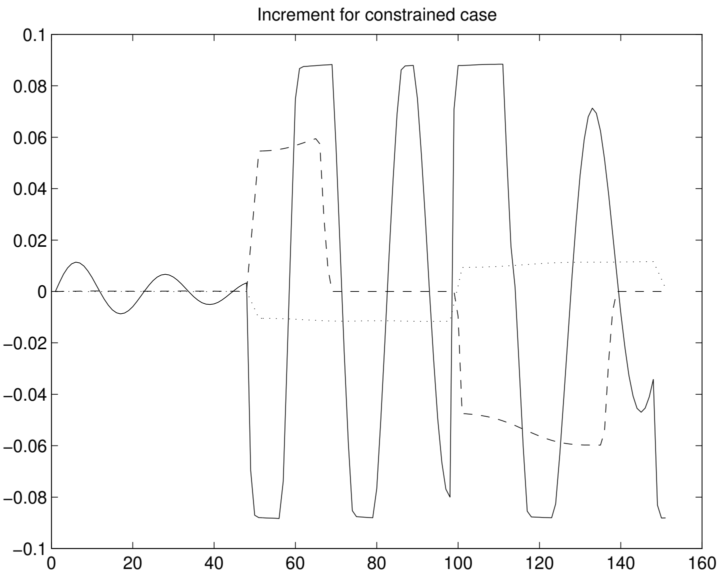

Figure 3.12: Increment of control variable in unconstrained case

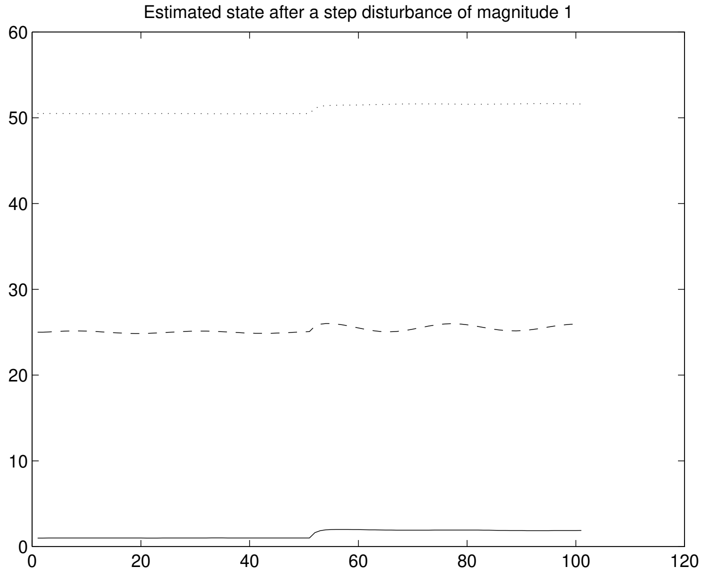

Figure 3.13: State estimate using $G = I$, $Q = I$ $R = I$

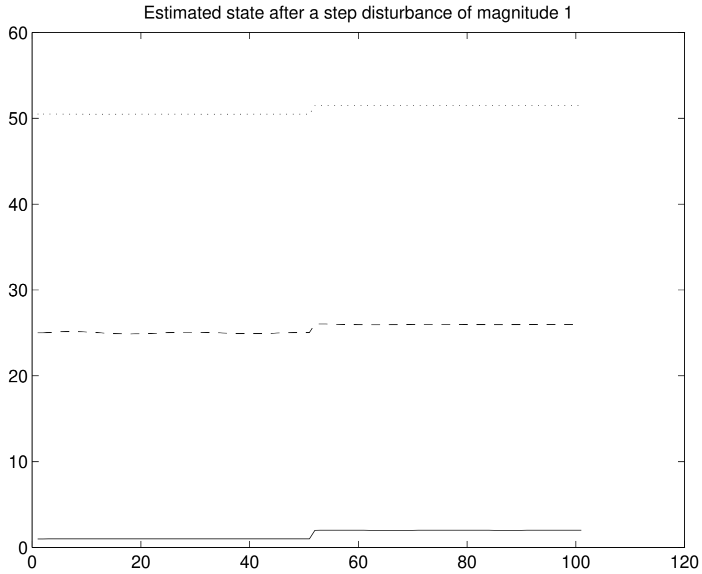

Figure 3.14: State estimate using $G = 10{\cdot}I$, $Q = 100{\cdot}I$ $R = 100{\cdot}I$

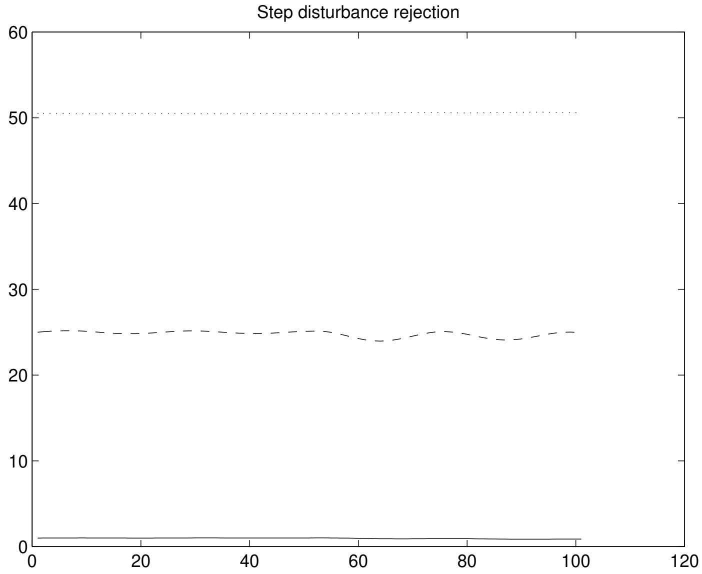

Figure 3.15: System output with disturbance rejected

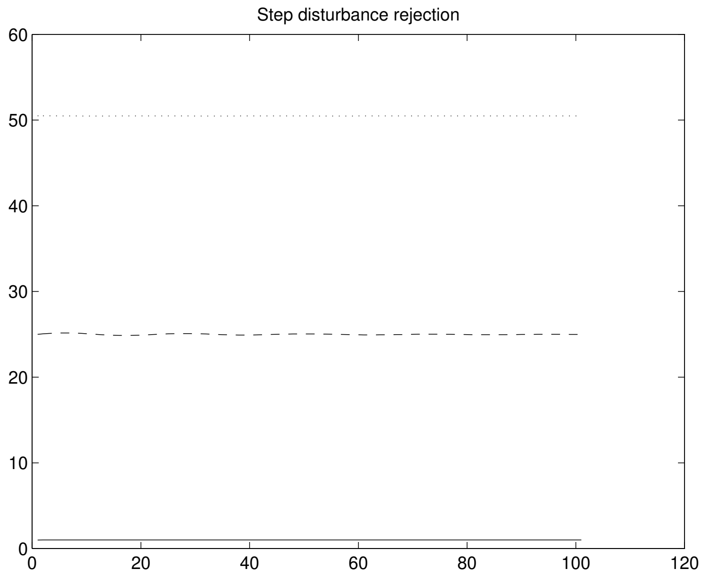

Figure 3.16: State estimate using $G = 10 \cdot I$, $Q = 100 \cdot I$ $R = 100 \cdot I$

3.2 MBPC design software

3.2.1 The simulation framework

Figure 3.17 shows the structure of the Simulink test framework. Software components are located within different blocks inside the scheme. The simulation framework consists of three different subsystems: the nonlinear model, the estimator and the optimizer. Various variables are taken out to the Matlab workspace in order to check and visualize the algorithm efficiency.

In order to setup the linearized model, to compute off-line matrices involved in optimization and to evaluate precise values for constraints two programs: **setup.m** and **algorithm.m** are employed by a double click on the corresponding block in the Simulink window. The software that manages all off-line computations is fairly general with a small exception given by tuning parameters such as weights for the Kalman filter and the cost function. Comments make it easy to conduct changes regarding these variables.

Inside the block "Matlab Function" lies the QP solver: **solver_qp.m** that performs the on-line optimization and provides the control increment for the next step. The delay is used to store the previous value of the command vector in order to be added with the new increment.

The estimator structure involves a standard structure that contains augmented system matrices as well as the delay block characteristic for such a state space representation of the discrete Kalman filter. The zero-order holds are included to simulate the function of an A/D converter based on a sample-hold mechanism.

The control algorithm flowchart (see figure 3.18) shows steps applied at each sampling time to the controlled system.

We prefer to organize the material in such a way, with a dedicated appendix for programs, because it is interesting to show the correspondence with notations inside the theoretical section and the rest of the material presented.

3.2.2 Quadratic programming (QP) functions

The method used in most of quadratic programming algorithms are due to Danzig (1963) and Lemke (1965) but some of them are subject of early techniques developed by Beale (1959) as an extension of linear programming.The minimization of a cost functional subject to linear inequalities is often referred as linear complementarity problem (LCP) which in fact is a Lagrangian method.

Matlab function block contains the procedure **solver_qp.m** listed in appendix that perform the QP on-line optimization. As shown, it is important to note the two variants of QP within Matlab, procedures that provide similar solutions with the only difference being that the C routine interfaced with Matlab is ten times faster than the standard "qp". It is important to pay attention to the different ways of describing the constraint inequality in these procedures. Considering the cost function 2.70 it is important to note that in order to have a well posed QP it is necessary to have the $\mathcal{A}_i$ matrix positive definite.

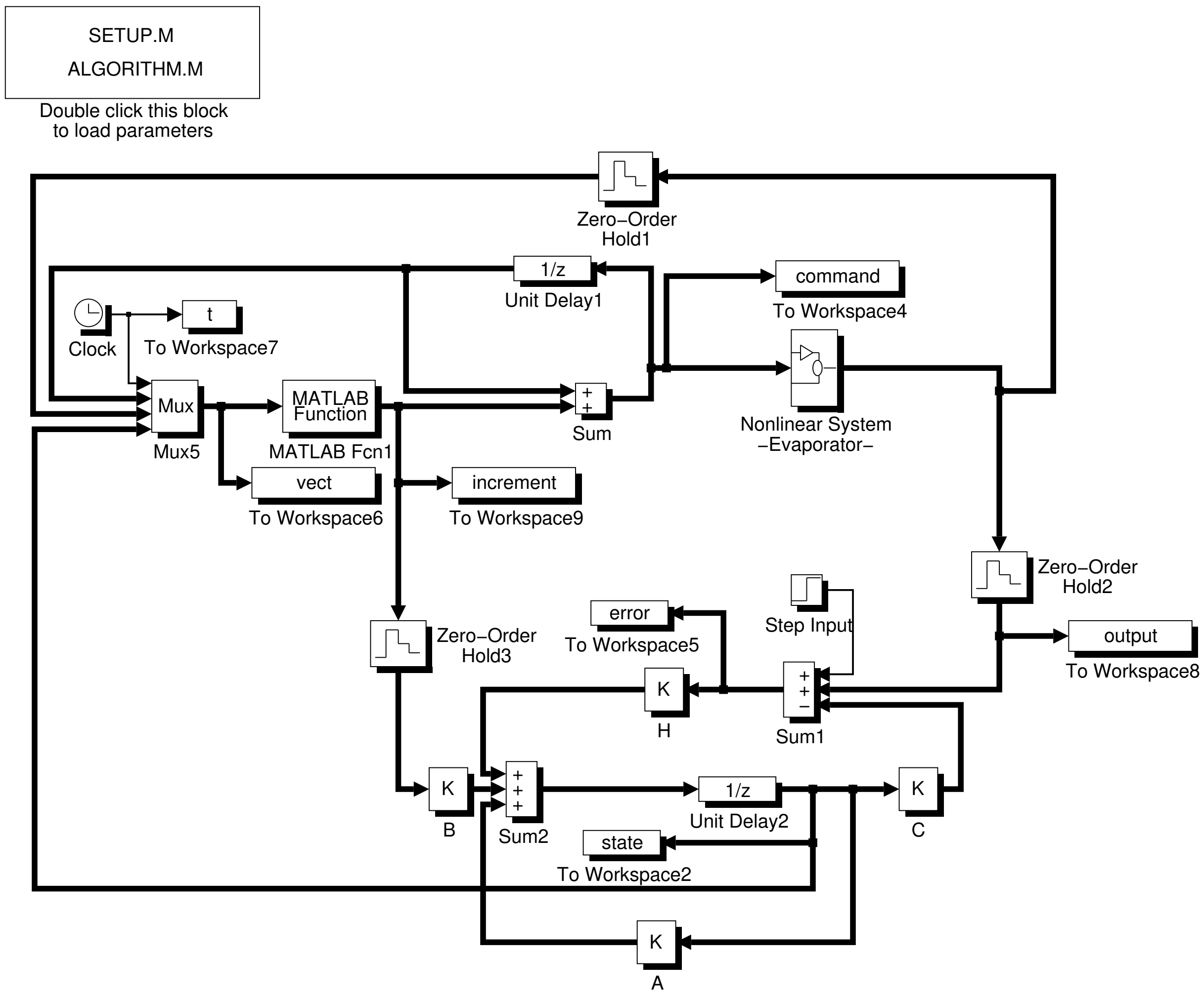

Figure 3.17: MBPC Test Framework

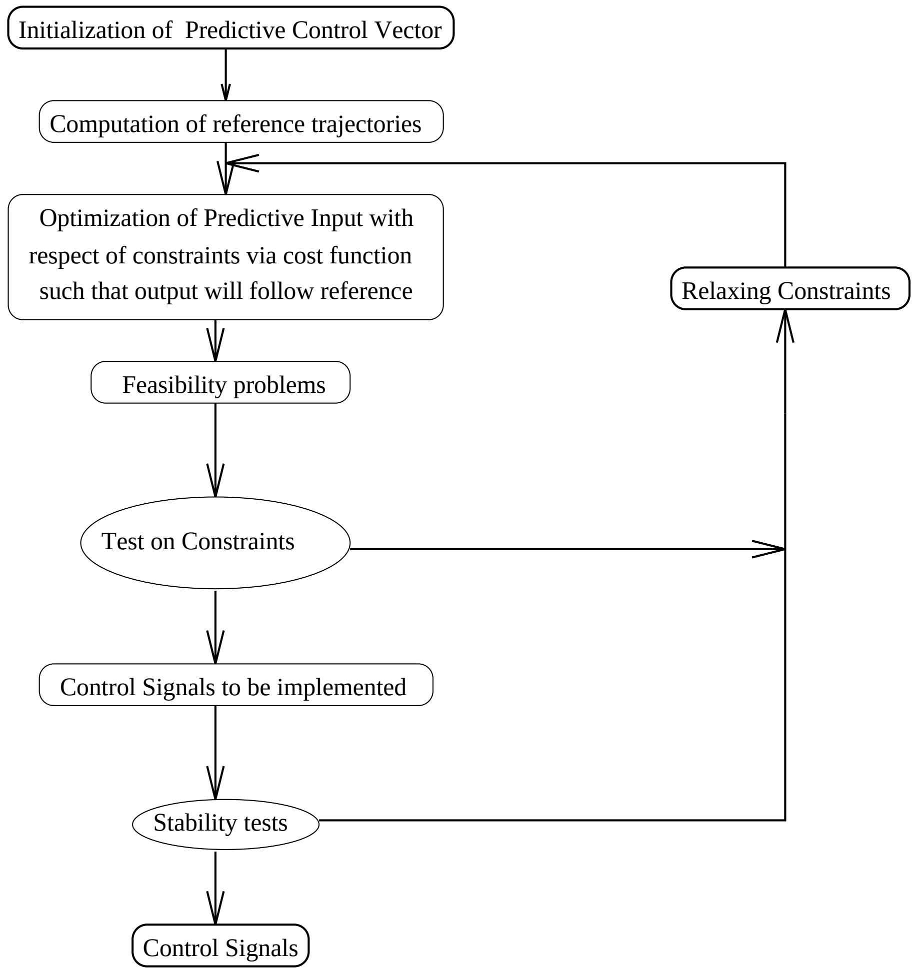

Figure 3.18: Control Algorithm Flowchart

Chapter 4

Conclusions

4.1 Research fields for the future

4.1.1 Reconfigurable control in case of failures

In order to prove statements made in the introduction we quote a few details from a recent report on an A-310 tragic event [10]. The scenario is very common for most take-offs from major world airports. The only difference is that due to a failure in the system an accident occurs. The report of the Romanian Accident Investigation Committee (AIC) says that " at about the time the event began" the copilot was flying the take-off from Bucharest manually, but with the autothrottle system (ATS) engaged and select on "take-off" mode. The aircraft was climbing through 2,000 ft at 190kt, with the crew retracting flap according to the usual procedure. At that point the crew had selected "apparently" the "climb" mode, which normally result in power reduction in both engines. The right throttle lever stayed at the previous position "as would be the case if it had been mechanically prevented, due to an unidentified reason, from responding to the ATS servo motor". Meanwhile the left throttle lever retarded slowly, taking 42s to reach idle. The pilot had already begun a 25° bank left turn as required by departure procedure, but the increasing power asymmetry caused the aircraft 18° nose-up to decrease to zero, and the bank to increase "as the aircraft continued to roll over laterally to more than 170° to the left , while the pitch seems to have reached 80° nose-down". The report mentioned that "only small rudder and elevator deflection" were recorded throughout the event. The flight-data recorder shows that engine 2 (right hand throttle) was reduced to idle only a few seconds before end of recording.

In this case the right proper action taken by the pilots, assisted by an automated flight system that include reconfigurable control in case of such failures, could have saved the aircraft.

The use of MBP control in order to solve reconfigurable control problem is suggested because of main keys of the algorithm which were proved to be:

- prediction by means of a process model
- generation of a reference trajectory
- ability to structure the control postulated law

- algorithmic control of the best control scenario
- ability to handle in a straightforward way constraints
- possibility to introduce feedforward control in order to compensate measured disturbance
- inherent dead-time compensation

Due to these properties certain changes in the flight envelope, the function of several actuators or the dynamics of the aircraft can be managed with satisfactory performance. The future study will investigate such issues with respect to stability and robustness of the system controlled with MBPC laws.

Severe changes, impossible to be handled via MBPC, will impose the use of different reference models using a compound between gain scheduling or switching and the MBP control algorithm.

Because the issue of a FDI (fault detection and identifier) system is in a way collateral to our research we assume that signals provided by such a system are accurate enough to be used in the reconfigurable control process.

The role of identification in the algorithm is to alarm, isolate and estimate the failure. The alarm task involves making a "logic decision" between wrong or fine in the system function. The problem of isolation is that of determining the source of failure (which sensor or actuator had failed and what type of failure has occurred). Estimation involves the determination of the extent of failure (a sensor may become completely non-operational or it may suffer a degradation in the form of increased inaccuracies).

The reference models must be designed carefully to represent achievable performance levels of the failed aircraft. Intuitively, the pilot will not push the performance of an aircraft that has a failure to the same level as the nominal one.

In the case of an aileron actuator malfunction, the pilot loses the primary roll control effector. In such a case, a differential elevator can be used to reconfigure for this failure but the pilot will never expect full recovery of the original performance. In all cases when an aircraft loses a control surface, it looses some performance. As soon as a failure is detected a different set of reference models should be used. In order to prevent the controller from pushing the aircraft beyond its capabilities a "slower"reference model is usually used in the case of failures.

The fault dynamics of the aircraft is completely different from the nominal one. Mechanism such as "gain scheduling" should be employed in order to move between these controlled designs as the flight envelope is traversed. Direct switching between models can lead to problems regarding stability. It is our idea to use a linear interpolation between models. Gain scheduling trays to capture the nonlinearities of the real plant via parametric linearity of the system.

Various ways of gain scheduling (interpolation between models) are known. Among of these developments on forward speed and on speed and incidence are most commonly used.

Of course, "Gain Scheduling" approach has certain limitation. For example the number of parameters that have to be stored across the flight envelope include the models that

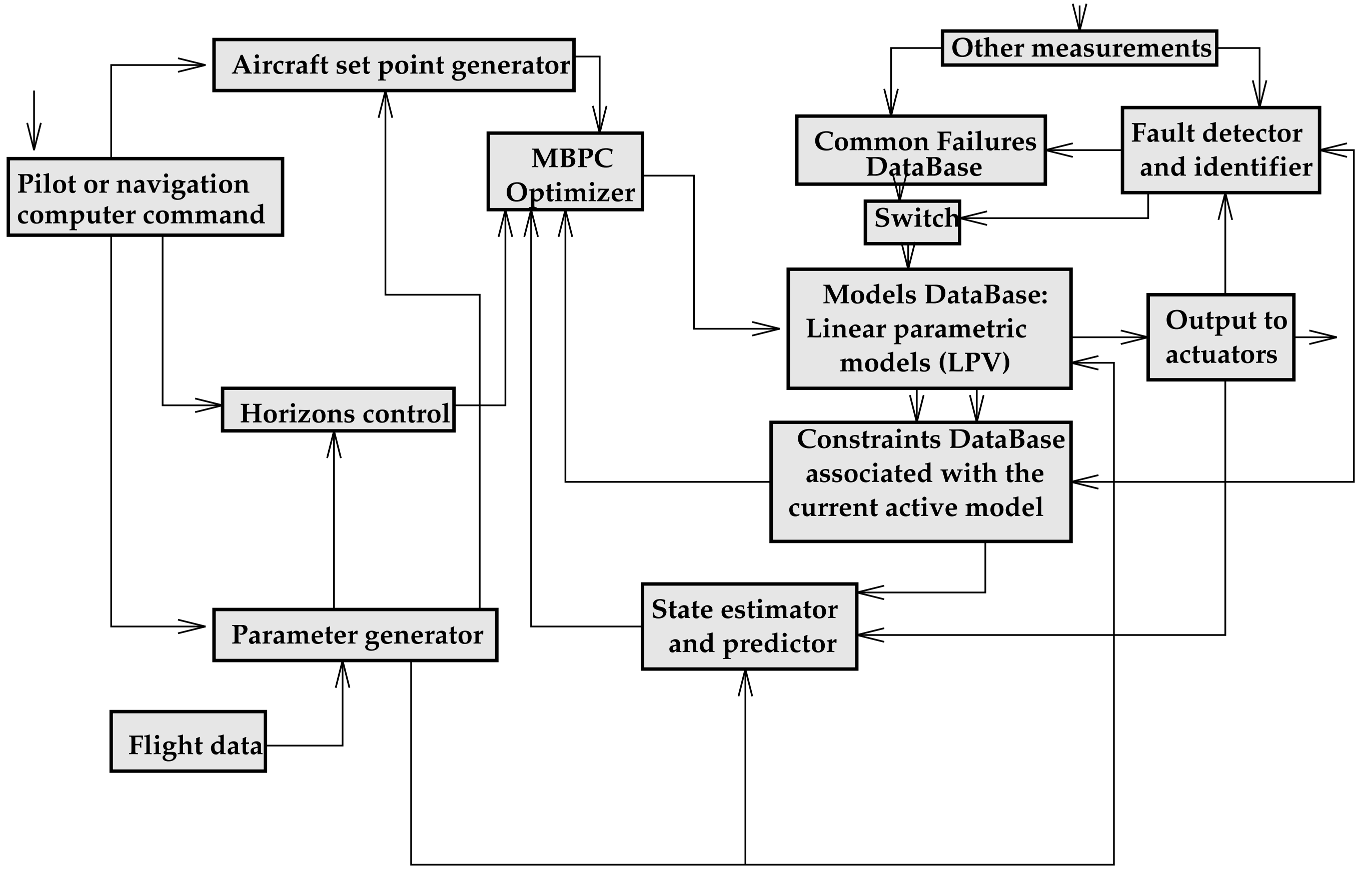

Figure 4.1: Configuration of a Flight MBPC System

incorporate failures. Another limitation is due to the fact that not all parameters need scheduling, so a careful selection is required.

The simulation developed in future work will deal at the beginning with few typical failures such as:

1. a failure on the pitch channel, with corresponding testing ways such as: bank to bank reversal attempts using full lateral stick inputs.

2. a failure in the horizontal stabilizer with corresponding ways of testing such as: full lateral stick input or a bank to bank reversal attempts using lateral stick commands.

3. a rudder failure that is tested with a 360° roll attempt using full coordinated stick and pedal inputs.

   All of these ideas will be handled by the system presented in figure 4.1. That figure represents in terms of blocks a possible structure that, using MBPC techniques, deals with the unusual issue of reconfigurable control in the case of failures.

4.1.2 Matlab and Simulink constrained predictive toolbox

A valuable toolbox should have the general scheme shown in figure 4.1. In fact this will be much more a modular structure than a Matlab toolbox because the whole purpose of this is to provide a flexible way to deal with various MBPC problems. This tool will be tested on two major examples described in chapter 3.

The system presented in figure 4.1 is required to have following features:

- On-line change of the model set, associated constraints, sampling rate, control and prediction horizons, constraints relaxing.
- Various optimization techniques such as QP, LP, LMI
- Automatic switch between nominal, fault, uncertainty models
- Warnings and alarms
- Various ways to display simulation outputs
- Standard measurement interface (DSpace DSP boards)
- Real time implementation using DSpace soft interface and boards

The design and evaluation environment used in the Garteur project will be extended as a flexible part of the toolbox that we intend to present at the end of the research.

4.1.3 Real time implementation of MBPC algorithms

The hard requirement for the algorithm speed is due to the following main problems:

1. For a fast system, such as an aircraft, the solution should be provided as soon as computed in order to avoid any delays.

2. For a spread system the time gained via a fast computation can be used to manage the lack of communication between consisting modules.

The interface to Matlab provided by the DSpace module will be employed in order to produce the required machine code for the DSpace floating point processor boards, the heart of future simulation systems installed. The possibility of producing accurate machine code direct from Matlab via such an interface will be investigated. Because of system complexity an approach similar to objectual programming is needed. After a careful study of known methods such as containers etc. a decision of which is the most suitable approach will be made.

The necessity to speed up the solution of the optimization might lead us with some theoretical problems. We will research the possibility to solve the optimization via a modified QP that will take advantage of the particular structure of the matrices involved in the quadratic cost function.

4.2 Research plan

The appropriate research will be developed around the project initiated by the GARTEUR group that presents the challenge of designing a flight control system that will control a civil aircraft, characterized by the RCAM model. All stages of design and result presentation are summarized in the table 4.2.

Connected with this project and as a tool to design future control systems using the MBPC algorithm we will develop a modular toolbox to be used with Simulink and Matlab. Finally the possibility of real time implementation will be investigated using DSpace software and DSP (digital signal processing) boards. As well, we will pay attention to the issue of reconfigurable control in order to accommodate the idea of failure. The corresponding mathematical environment will be developed.

The long term research plan covers the period 1996, 1997 (two full years). During this time we will develop a complete framework that is able to deal with linear parametric models (aircraft models that change in an LPV manner according to the flight envelope) and reconfigurable control in case of failures (most of the failures according with the pilot manual). Apart from the "Frankfurt approach" scenario provided in the "Garteur" (RCAM) project, other test benchmarks will be developed according to practical reasons. The main features of MBPC as such are the ability to deal with constraints and to cope with changes in the structure of the real plant, changes that will keep the plant close enough to its linearized model, will be used throughout future research regarding reconfigurable control. The perspective of using MBPC control laws for fighters represent another issue to be investigated.

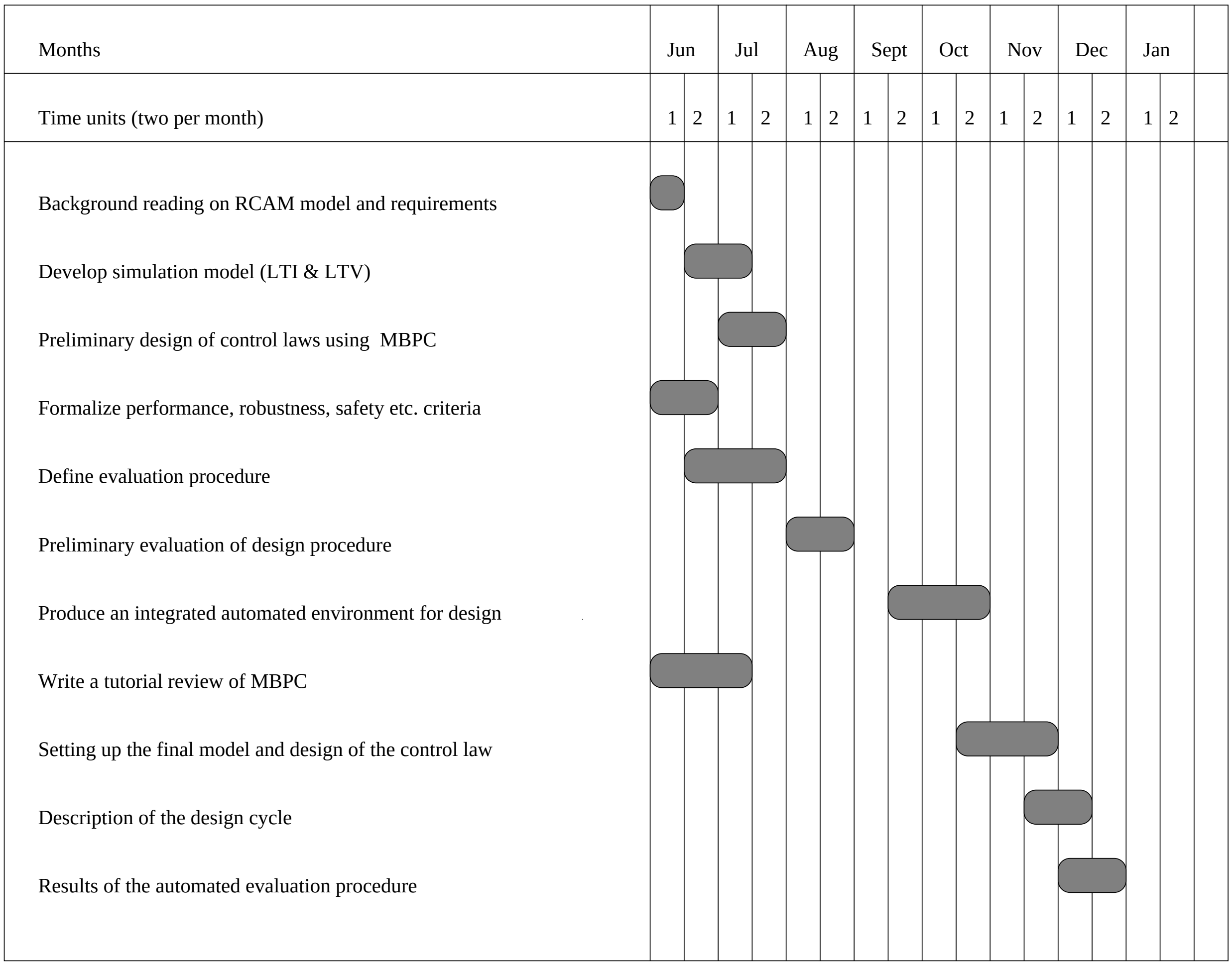

Figure 4.2: Time scale for GARTEUR (RCAM model) – The Applied MBPC Project

Chapter 5

Appendix

5.1 Matlab program SETUP.M

```
clear
% AA,BB,CC,DD represent the discrete model

AA = [ -0.001    0.0042    0.0075;
        0.000   -0.1000         0;
        0.000   -0.0209   -0.0558];

BB = [-0.0500   -0.0019         0;
      -1.2500         0         0;
            0    0.0096   -0.0018];

CC = [ 1.0000         0         0;
            0    1.0000         0;
            0         0    1.0000];


DD = [  0     0     0
        0     0     0
        0     0     0];

% mu_ini = number of system input
mu_ini = 3;

% ns_ini = number of system states
ns_ini = 3;

% po_ini = number of system outputs
po_ini = 3;

save model AA BB CC DD mu_ini ns_ini po_ini;

%Uo = initial command
Uo = zeros(mu_ini,1);

%XO = initial state
Xo = [1 25 50.5]';

%Yo = initial output
Yo = [1 25 50.5]';

%X_pred=initial prediction
X_pred = [[0 0 0]';Yo];

save initial Uo Xo X_pred Yo;
```

```
% NNu = control horizon
NNu = 3;

% NN2 = prediction horizon
NN2 = 10;
save horizons NNu NN2;


% ll = constraints upon rate of inputs change
ll1 = 1000*ones(6,1);
ll = ll1;
for k = 2:NNu
ll = [ll; ll1];
end

% mm = constraints upon inputs
mm1 = 1000*ones(6,1);
mm = mm1;
for k = 2:NNu
mm = [mm; mm1];
end

% nn = constraints upon states
nn1 = 1000*ones(12,1);
nn = nn1;
for k = 2:NN2
nn = [nn; nn1];
end

save constraints ll mm nn;

% Benchmark setup
Ysel1 = Yo;
S1 = Ysel1;
for k = 2:100
S1 = [S1;Yo];
end

Ysel2 = Yo+[0.03 10 1]';
S2 = Ysel2;
for k = 2:50
S2 = [S2;Ysel2];
end

Ysel3 = Yo-[0.03 10 1]';
S3 = Ysel3;
for k = 2:50
S3 = [S3;Ysel3];
end

% Benchmark 1
SS = [S1; S2; S3; S1; S2; S3; S1; S1; S1; S1];

% Benchmark 2
%SS = [S1; S1; S1; S1; S1; S1; S1; S1];

% Benchmark 3
%SS = [S2; S2; S2; S2; S2; S2; S2; S2];

% Benchmark 4
%SS = [S3; S3; S3; S3; S3; S3; S3; S3];

save set_points SS;
```

5.2 Matlab program ALGORITHM.M

```
global ll mm nn A B C D mu_sys ns_sys po_sys NNu NN2 SS st K KK S Uo Xo Yo
X_pred Delta_prev H_new F_new G_new C_new Lambda_new I_new Q_new R_new
A_new B_new L_new M_new N_new D_new E_new;


%Uo = initial command
%XO = initial state
%Yo = initial output
%X_pred=initial prediction
load initial;


%SS vector of set points over the whole future horizon
load set_points

%Number of steps in simulation
st=1;

% ll = constraints upon rate of inputs change
% mm = constraints upon inputs
% nn = constraints upon states
load constraints;


% AA,BB,CC,DD represent the discrete model
% mu_ini = number of system inputs
% ns_ini = number of system states
% po_ini = number of system outputs
load model;

% A,B,C,D represent the discrete augmented model
% mu_ini = number of system inputs of augmented model
% ns_ini = number of system states of augmented model
% po_ini = number of system outputs of augmented model
A = [ AA    , zeros(ns_ini,po_ini);
     CC*AA ,   eye(po_ini,ns_ini)];

B = [    BB;
      CC*BB];

C = [zeros(po_ini,ns_ini) , eye(po_ini,ns_ini)];

D = DD;

% Compute augmented system dimensions
mu_sys =mu_ini;
ns_sys =ns_ini+po_ini;
po_sys =po_ini;

% NNu = control horizon
% NN2 = prediction horizon
load horizons;

if NNu>NN2
stop;
end


% Setup the environment for Kalman filter computation,
% Kalman filter associated with the augmented system

% Input weight matrix GK
GK = 0.1*eye(ns_sys);
```

```
% State weight matrix QK
QK = 1000*eye(ns_sys);

% Output weight matrix RK
RK = 1000*eye(po_sys);

% Kalman Filter matrix computation
KK = dlqe(A,GK,C,QK,RK);

% Computation of constraint vectors dimension
[sl,cl] = size(ll);
[sm,cm] = size(mm);
[sn,cn] = size(nn);



% Precomputational movements
F_aux_0 = A-KK*C;
G_aux_0 = B;
H_aux_0 = KK;



% F_new computation
F_aux_1 = A*F_aux_0;
F = [F_aux_0;
     F_aux_1];

for p = 3:NN2
F_aux_1 = A*F_aux_1;
F = [F      ;
     F_aux_1];
end
F_new = F;



% H_new computation
H_aux_1 = A*H_aux_0;

H = [H_aux_0;
     H_aux_1];
for p = 3:NN2
H_aux_1 = A*H_aux_1;
H = [H      ;
     H_aux_1];
end
H_new = H;



% G_new computation
G_aux_1 = A*G_aux_0;

Z = zeros(ns_sys,mu_sys);
G_col_aux = [G_aux_0;
            G_aux_1];

for p = 3:(NN2-NNu+1)
G_aux_1 = A*G_aux_1;
G_col_aux = [G_col_aux;
             G_aux_1  ];
end

G_col = G_col_aux;

for p = 1:(NNu-1)
```

```
G_col = [Z    ;
         G_col];
end

G_matrix = [G_col G_matrix];

for q = 2:(NNu-1)

G_aux_1 = A*G_aux_1;
G_col_aux = [G_col_aux;
             G_aux_1  ];
G_col = G_col_aux;
for p = 1:(NNu-q)
G_col = [Z    ;
         G_col];
end

G_matrix = [G_col G_matrix];

end

G_aux_1 = A*G_aux_1;
G_col_aux = [G_col_aux;
             G_aux_1  ];
G_col = G_col_aux;
G_matrix = [G_col G_matrix];
G_new = G_matrix;



% C_new computation
C_first = C;
C_matrix = C_first;
for q = 1:NN2-1
Z_l = zeros(po_sys,(ns_sys*q));
Z_u = zeros((po_sys*q),ns_sys);

C_matrix = [C_matrix  ,     Z_u;
                 Z_l  , C_first];
end
C_new = C_matrix;



% Lambda_new computation
Z_m = zeros(mu_sys);
I_m = eye(mu_sys);

for qq = 1:NNu

if qq == 1
Lambda_line_aux = I_m;
else
Lambda_line_aux = [Lambda_line_aux,I_m];
end

Lambda_line = Lambda_line_aux;

for q = qq:NNu-1
Lambda_line = [Lambda_line , Z_m];
end

if qq == 1
Lambda = Lambda_line;
else
Lambda = [Lambda    ;
```

```
        Lambda_line];
end

end
Lambda_new = Lambda;



% I_new computation
I_new = Lambda_line';



% Q_new computation
Q = 0.007*eye(po_sys*NN2);


% R_new computation
R = 0.001*eye(mu_sys*NNu);



% A_new computation
A_new = G_new'*C_new'*Q*C_new*G_new+R;


% B_new computation
B_new = 2*[C_new*F_new C_new*H_new -eye(NN2*po_sys)]'*Q*C_new*G_new;



% L_new computation
L_new = [-eye(NNu*mu_sys);
          eye(NNu*mu_sys)];


%M_new computation
M_new = L_new;



%N_new computation
N_new = [-eye(NN2*ns_sys);
          eye(NN2*ns_sys)];



%D_new computation
D_new = [            L_new;
          M_new*Lambda_new;
               N_new*G_new];



% E_new computation
[r1,c1] = size(M_new*I_new);
[r2,c2] = size(N_new*F_new);
[r3,c3] = size(N_new*H_new);

E_new = [ zeros(2*NNu*mu_sys,c1) , zeros(2*NNu*mu_sys,c2) ,
zeros(2*NNu*mu_sys,c3) ,
 eye(2*NNu*mu_sys,sl)   , zeros(2*NNu*mu_sys,sm) ,
zeros(2*NNu*mu_sys,sn);
         (-M_new*I_new) ,    zeros(2*NNu*mu_sys,c2) ,
zeros(2*NNu*mu_sys,c3) ,
 zeros(2*NNu*mu_sys,sl) , eye(2*NNu*mu_sys,sm)   ,
zeros(2*NNu*mu_sys,sn);
```

```
        zeros(2*NN2*ns_sys,c1) , (-N_new*F_new) ,
(-N_new*H_new)  ,
 zeros(2*NN2*ns_sys,sl) , zeros(2*NN2*ns_sys,sm)    ,
eye(2*NN2*ns_sys,sn)];
```

5.3 Matlab program SOLVER_QP.M

```
function[Delta_U]=solver_qp(vector)
%********** Simulink-Matlab Function *************************
%
%
%Performs on line decomposition of multiplexed input vector and
%feed in an appropriate manner function quadratic that performs
%future moovement computation
%
%
%************************************************************



global ll mm nn A B C D mu_sys ns_sys po_sys NNu NN2 SS st KK S
Uo  Xo Yo X_pred Delta_prev H_new F_new G_new C_new Lambda_new I_new
Q_new R_new A_new B_new L_new M_new N_new D_new E_new;


time = vector(1);
U = vector(2:(mu_sys+1));
Y = vector((mu_sys+2):(mu_sys+po_sys+1));
X_pred1 = vector((mu_sys+po_sys+2):(mu_sys+ns_sys+po_sys+1));

if time == 0
U = Uo;
Y = Yo;
X_pred1 = X_pred;
end

S = SS(((time*po_sys)+1):((time*po_sys)+(NN2*po_sys)));


% QP initialization
ccc = [X_pred1' Y' S']*B_new;
bbb = E_new*[U;X_pred1;Y;ll;mm;nn];
AAA = D_new;
QQQ = A_new;

% QP solver

%First method
%[Delta_U_O,OBJ,FAIL] = lqp(sparse(QQQ),ccc,-sparse(AAA),-bbb,[],[],[],0);
%Second method

Delta_U_O = qp(QQQ,ccc,AAA,bbb);

% First step ahead implementation of QP Delta U solution
Delta_U = Delta_U_O(1:mu_sys);


else

end
```

5.4 C routine LQP interfaced with Matlab

```
LQP  Linear and quadratic programming using LQP
     [X,OBJ,FAIL]=LQP([],c,A,b) solves the LP problem
            min c'x   subject to:  Ax >= b
                                   ======= (note direction)
     [X,OBJ,FAIL]=LQP(Q,c,A,b) solves the QP problem
            min 0.5*x'Qx + c'x   subject to:  Ax >= b
     (Q must be nonnegative of course)
     A, and Q if present, must be sparse (if not, then use
     [X,OBJ,FAIL]=LQP(sparse(Q),c,sparse(A),b) )
     [X,OBJ,FAIL]=LQP(Q,c,A,b,l,u,r,verbose,inftol,sigfig)
     defines a set of lower and upper
     bounds on the design variables, X, so that the solution
     is always in the range l < x < u, and a set of ranges on the
     constraints so that b <= Ax <= b+r.
     If any of l, u or r are not specified, or are empty, they default to
       l=-Inf*ones(size(c))
       u=Inf*ones(size(c))
       r=Inf*ones(size(b))
     (i.e. LQP(Q,c,A,b,[],[],[]) is equivalent to  LQP(Q,c,A,b) )
     See the LOQO manual for the meaning of verbose, sigfig and inftol
       - if empty, or not specified, they are left at their defaults
     lqp.mex is a matlab interface to the code LOQO
```

5.5 Classic QP routine part of "Matlab Optimization Toolbox"

QP Quadratic programming. X=QP(H,f,A,b) solves the quadratic programming problem:
min 0.5*x'Hx + f'x subject to: Ax ¡= b x
[x,LAMBDA]=QP(H,f,A,b) returns the set of Lagrangian multipliers, LAMBDA, at the solution.
X=QP(H,f,A,b,VLB,VUB) defines a set of lower and upper bounds on the design variables, X, so that the solution is always in the range VLB ¡ X ¡ VUB.
X=QP(H,f,A,b,VLB,VUB,X0) sets the initial starting point to X0.
X=QP(H,f,A,b,VLB,VUB,X0,N) indicates that the first N constraints defined by A and b are equality constraints.

QP produces warning messages when the solution is either unbounded or infeasible. Warning messages can be turned off with the calling syntax: X=QP(H,f,A,b,VLB,VUB,X0,N,-1).

5.6 Poles of $A - K{\cdot}C$ for various weighting matrices

```
A-KC =
   -0.0000
   -0.1008
   -0.0559
    0.3820
    0.3811
    0.3816

A-KC =
   -0.1091
   -0.0580
   -0.0000
    0.0185
    0.0098
    0.0122
```

Bibliography

[1] P.Banerjee adn S.L.Shah. The role of signal processing methods in the robust design of predictive control. *Automatica*, 31:681–695, 1995.

[2] J.C. Allwright. On min-max model-based predictive control. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 415–426. Oxford University Press, 1993.

[3] J.A. Rossiter B. Kouvaritakis and A.O.T. Chang. Stable generalized predictive control: An algorithm with guaranteed stability. *IEE Procedings-D*, 139(4):349–362, 1993.

[4] F. Berlin and P.M. Frank. Design and realization of a mimo predictive controller for a 3-tank system. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 446–457. Oxford University Press, 1993.

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

[6] D. Clarke. Advances in model-based predictive control. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 3–21. Oxford University Press, 1993.

[7] D.W Clarke and C. Mohtadi. Properties of generalized predictive control. *Automatica*, 25(6):859–875, 6 1989.

[8] R.M.C de Keyser. The MBPC approach. In *CIM - Europe Workshop on Industrial Application of MBPC*. ESPRIT CIM, 1992.

[9] R.A.J. de Vries and H.B. Verbruggen. Multivariable unified predictive control. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 84–102. Oxford University Press, 1993.

[10] D.Learmount. Throttles and pilots are key to tarom issues. *Flight International 19-25 April*, page 6, 1995.

[11] G.E. Garcia D.M. Prett and C.R. Cutler. The dynamic matrix control method. *US Patent 4349869*, 9 1982.

[12] D. Dumur and P. Boucher. Predictive control application in the machine-tool field. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 471–482. Oxford University Press, 1993.

[13] C. Mohtadi D.W Clarke and P.S. Tuffs. Generalized predictive control i and ii. *Automatica*, 23(2):137–160, 1987.

[14] K. Gousman F.A.Zaid, P.Ionnou and R. Rooney. Accomodation of failures in the F-16 aircraft using adaptive control. In *Submitted to.* IEEE Transactions on Automatic Control, 1991.

[15] R. Fletcher. *Practical Optimization.* John Wiley an Sons, first edition, 1991.

[16] B.A. Francis. *A Course in H infinity Control Theory.* Prentice-Hall, first edition, 1991.

[17] J.D. Powell G.F. Franklin and A. Emami-Naeini. *Feedback Control of Dynamic Systems.* Macmillan Publishing Company, third edition, 1994.

[18] G.H. Golub and C.F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, 1983.

[19] A. Halanay and V. Ionescu. *Time Varying Discrete Linear Systems.* Birkhauser, first edition, 1994.

[20] S.A. Heise and J.M. Maciejowski. Stability constrained MBPC using a internal model control structure. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 230–244. Oxford University Press, 1993.

[21] J.L. Testud J.A. Richalet, A. Rault and J. Papon. Model predictive heuristic control: Aplications to an industrial process. *Automatica*, 14:413–428, 1978.

[22] B.A. Francis J.C. Doyle and A.R. Tannenbaum. *Feedback Control Theory.* Macmillan Publishing Company, first edition, 1992.

[23] J.Doyle K. Zhou and K. Glover. *Robust and Optimal Control.* unpublished draft, 1993.

[24] E.G. Kassapakis and K. Warwick. Predictive control application for autopilot design. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 458–470. Oxford University Press, 1993.

[25] A.W. Kwonk and K.M. Passino. Expert supersision fo fuzzy learning systems for fault tolerant aircraft control. In *Submitted to.* IEEE Transactions on Automatic Control, 1993.

[26] D.A. Linkens and M. Mahfouf. Generalized predictive control in clinical anaesthesia. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 429–445. Oxford University Press, 1993.

[27] J.M. Maciejowski. *Multivariable Feedback Design.* Addison Wesley, first edition, 1989.

[28] J.M. Maciejowski. Dynamic matrix control. In *Report produced for Cambridge Control Ltd.* Cambridge Control Ltd., 1991.

[29] J.M. Maciejowski and S.A. Heise. Heuristic robustness analysis of model-based predictive controllers. In *Proceedings of the 12th World Congress*. International Federation of Automatic Control, 1993.

[30] D.C. McFarlane and K.Glover. *Robust Controller Design Using Normalized Coprime Factor Plant Descriptions*. Springer-Verlag, first edition, 1989.

[31] M. Morari. Model predictive control: Multivariable control technique of choice in the 1990's. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 22–37. Oxford University Press, 1993.

[32] K.R. Muske and J.B. Rawlings. Linear model predictive control of unstable processes. *Journal of Process Control*, 1993.

[33] K.R. Muske and J.B. Rawlings. Model predictive control with linear models. *Process Systems Engineering*, 39(2):262–287, 1993.

[34] K. Dass P. Krauss and H. Rarke. Model-based predictive control with kalman filtering for state estimation. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 69–83. Oxford University Press, 1993.

[35] D.M. Prett and C.E. Garcia. *Fundamental Process Control*. Butterworths Series in Chemical Engineering, first edition, 1988.

[36] J.B. Rawlings and K.R. Muske. The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 1992.

[37] J.A. Rossiter and B. Kouvaritakis. Advances in generalized and constrained predictive control. In D. Clarke, editor, *Advances in Model-Based Predictive Control*, pages 303–317. Oxford University Press, 1993.

[38] J.A. Rossiter and B. Kouvaritakis. Constrained stable generalized predictive control. *IEE Procedings-D*, 140(4):243–254, 1993.

[39] T. Soderstrom and P. Stoica. *System Identification*. Prentice Hall, first edition, 1989.

[40] T.T.C. Tsang and D.W. Clarke. Generalized predictive control with input constraints. *IEE Procedings-D*, 135(6):1451–1460, 1988.

[41] A. Zheng and V.Balakrishnan. Constrained stabilization of discrete-time systems.

[42] A.S. Willsky. A survey of design methods for failure detection in dynamic systems. *Automatica*, 12:601–611, 1976.

[43] E. Zafiriou. Robust model predictive control of processes with hard constraints. *Computers Chemical Engineering*, 14(4/5):359–371, 1990.

[44] E. Zafiriou and H-W. Chiou. On the effect of constraint softening on the stability and performance of model predictive controllers. In *Submitted to*. Anual AIChE Meeting Nov. 1-6 1992, session 123, 1993.

[45] A. Zeng and M. Morari. Stability of model predictive control with soft constraints. In *Submitted to*. IEEE Transactions on Automatic Control, 1993.