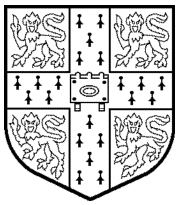
Theory and Aerospace Applications of Constrained Model Based Predictive Control

Mihai Huzmezan

Pembroke College
Department of Engineering
University of Cambridge



A dissertation submitted for
the degree of Doctor of Philosophy

March, 27 1998

"Who can control his faith?"

Othello

Cu dragoste intregii mele familii
(With love to my family)

Abstract

The thesis applies the Model Based Predictive Control ($MBPC$) technique, a relatively well known technique in the process industry, within the aerospace environment. Such an approach was taken because of some appealing attributes of $MBPC$ such as simplicity, richness and practicality. Moreover, the built in constrained optimisation gives to the flight control architecture more power in terms of monitoring actuator states or flight envelopes.

After an introduction, the second chapter contains a comprehensive overview of the methodology in an unified manner covering the state space and the input-output formulations of model based predictive control.

The third chapter deals with theoretical issues, such as stability feasibility and robustness, taking the reader into both unconstrained and constrained problem formulations.

The fourth chapter addresses some implementation issues of predictive controllers. The problems addressed range from tuning procedures and guidelines to a CAD development space used to tune and implement $MBPC$ controllers.

The fifth chapter contains a novel application of $MBPC$, as a Stability Augmentation System, to a Research Civil Aircraft Model (RCAM) under the auspices of the Group for Aeronautical Research and Technology in Europe (GARTEUR).

The next chapter presents an application of the $MBPC$ technique to Flight Management in a combined framework with an H_∞ controller used for the Guidance and Stability Augmentation System.

The seventh chapter describes advances towards reconfiguration and scheduling in flight control systems using high fidelity models expressed in a quasi-LPV form, Fault Detection and Isolation (FDI), model approximation/simplification and constrained Model Based Predictive Control. The strategy is applied to a well known missile example. This represents a new Reconfigurable Flight Control System. As well, this chapter explores the method developed by employing it to a High angle of Attack Research Model (HIRM) obtained by the courtesy of the Defence and Evaluation Research Agency (DERA).

Preface

I would like to express my gratitude to my supervisor Dr. Jan Maciejowski for providing me with his constant support, encouragement and guidance. His enthusiasm and willingness to allow me to pursue my own interest have been much appreciated. I am also very grateful to Professor Keith Glover, Dr. Glenn Vinnicombe and Dr. Malcom Smith from whom I have learned much about control theory. I would like also like to thank my previous supervisor Professor Vlad Ionescu for the guidance during my M.Sc. and the freedom he allowed me to choose the direction I was to take.

Since joining the group I have enjoyed the great company and help of all its members. I would like to thank George Papageorgiou, Michael Cantoni, Richard Ford and John Steele for their friendship and their time spent in discussing various control issues or proofreading various papers or versions of the present manuscript. I am also grateful to past and present members of the control group who have made pleasant my days in the Fallside Laboratory: Gavin, Giles, Philip, Tim, Teddy, Richard W., Nelson, Richard L., Sinichi, Thomas, Camile, Alex, Johannes, Paresh, Suren.

In particular I would like to thank my family for their continuing encouragement and support during my years abroad without whom I could never have finished this thesis. Also I would like to thank my friends, both near and far, who have made my experience here in Cambridge memorable.

I would like to acknowledge Pembroke College Cambridge, Lundgren Fund, ORS scholarship, Cambridge Overseas Trust, CT Taylor Fund, SM Invest Holding, DERA for their financial support during my studies in Cambridge.

As required by the University Statute, I hereby declare that this dissertation is not substantially the same as any that I have submitted for a degree at any other University, is the result of my own work, and includes nothing which is the outcome of work done in collaboration.

This thesis contains approximately 62000 words in length.

Mihai Huzmezan
Pembroke College,
Cambridge,
March 26, 1998

Acronyms

CARIMA — Controlled Auto-Regressive and Integrated Moving Average
CARMA — Controlled Auto-Regressive Moving Average
CRHPC — Constrained Receding Horizon Predictive Controller
CSGPC — Constrained Stable Generalised Predictive Control
DERA — Defence Evaluation and Research Agency
DS — Development Space
DMC — Dynamic Matrix Control
FDI — Fault Detection and Isolation
FM — Flight Management
GARTEUR — Group for Aeronautical Research and Technology in EURope
GS — Guidance System
GPC – Generalised Predictive Control
GUI — Graphical User Interface
HIRM — High Incidence Research Model
IDCOM — IDentification and Command
LHP — Left Half Plane
LMI — Linear Matrix Inequalities
LP — Linear Programming
LPV — Linear Parameter Varying
LTI — Linear Time Invariant
LTV — Linear Time Varying
MBPC — Model Based Predictive Controller
MCSGPC — Modified Constrained Stable Generalised Predictive Controller
MIMO — Multi Input Multi Output
MoD — Ministry of Defence
MPC – Model Predictive Control
MWLS — Mixed Weight Least Squares
QP — Quadratic Programming
RCAM — Research Civil Aircraft Model
RFCS — Reconfigurable Fligth Control System
SAS — Stability Augmentation System
SIORHC — Stable Input Output Receding Horizon Controller
SISO — Single Input Single Output
SGPC — Stable Generalised Predictive Control
PIM — Pseudo Inverse Method

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In this work we aim to improve the understanding of the Model Based Predictive Control ($MBPC$) technique and its potential in aerospace applications. This subject was generated by an attempt to apply a control technique mainly used in process industry in a completely new environment — the design and commissioning of flight controllers.

The idea of posing flight control problems as constrained optimisation problems was started by the needs developed by the military aerospace industry in the sixties when it addressed launching, guiding and landing space vehicles. The main feature of predictive controllers is their ability to handle constraints. In such a case the resulting control law is nonlinear but copes with panic kinds of situations when if a hard constraint is exceeded, catastrophic failures can occur. So a standard predictive controller will behave linearly when operating safely away from constraints but in a nonlinear manner when these are approached too closely.

Predictive control involves solving optimisation problems on-line which will yield computation time problems when addressing general aerospace applications. Of course this problem will gradually disappear as computer speeds increase but even now, as described in the thesis, there are applications such as flight management where $MBPC$ can provide real time solutions or, as another example, large space structures with solar panel arrays which have slow dynamics that do not require a fast update.

The reason for studying the theory of the Model Based Predictive Control algorithm was to enable us to gain necessary insight into the technique, to enable a full exploitation of its features and to generate proofs regarding the properties of our controllers. This study gave us the possibility to make some theoretical contributions. The main goal of our research was to push the application of $MBPC$ to its limits in terms of implementation possibilities and real time solution, with the aim of discovering the most realistic aerospace applications for it.

This thesis offers the necessary hints and guidelines for a time domain based tuning of the $MBPC$ in the case of customised applications such as aircrafts or missiles. The connection with the frequency domain can be observed when trying to design the $MBPC$ for robust performance or when various automatic tuning procedures required in reconfig-

urable control were employed.

1.2 A bit of history

Most of the surveys available in the field of Model Based Predictive Control ($MBPC$) include at their beginning a brief history of the field but none of them order the references using the year as a basis.

Therefore, before giving an outline of the thesis we consider a tabular, and therefore condensed format, historical presentation of the development of the predictive control field will be useful in order to locate our contribution and evaluate the state of the art in this particular research area. Of course this brief presentation is affected by the subjectivity of the author.

Table 1.1 offers the reader the possibility to see at a glance how this particular domain developed and when the major contributions were made, see Table 1.1. In the presentation few comments will be made with respect to the papers mentioned, but we invite the reader to familiarise her/himself with the author's view of this field as presented in the following chapters. This view was influenced by the work of the researchers enumerated.

Author and Year	Comments
[KP75]	Initial stability results in the receding horizon case including terminal constraints upon a number of states were presented.
[RRTP78]	It represents an early contribution to the IDentification and COMmand ($IDCOM$) algorithm.
[PGC82]	Dynamic Matrix Control (DMC) – an input output approach based on step/impulse response models.
[RM82]	Impulse response models are used to produce the control law based on the Model Algorithmic Control (MAC) approach.
[BV84]	A step response model, easy to obtain via experiments and on-line identification, was used in a DMC fashion.
[Pet84]	Transfer functions were intoduced in predictive control for the first time.
[MZM84]	The MUltiStep Adaptive Regulator ($MUSMAR$ is suggested to the control community as a first adaptive approach involving predictive control.
[DCT87]	The Generalised Predictive Control (GPC) algorithm is introduced together with an analysis of its properties.
[PG88]	In this book the authors for the first time have mentioned several application where predictive control was successful providing the reader with an accurate description of the DMC algorithm.

Author and Year	Comments
[CM89a]	The Long Range Predictive Control ($LRPC$) is introduced and analysed.
[BGW90]	The stability of the receding horizon scheme is analysed using Fake Algebraic Equations.
[Zaf90]	The issue of robust model predictive control is addressed for the first time together with the introduction of the new concept of hard and soft constraints.
[LM91b]	The robustness of the predictive control scheme is addressed based on an internal model obtained via identification.
[RC91]	Robustness effects of a pre-filter in Generalised Predictive Control are considered and analysed.
[CS91]	Constrained Receding-Horizon Predictive Control (CRHPC) is introduced and optimisation feasibility issues considered.
[MZ92]	Stabilising Input-Output Receding Horizon Control ($SIORHC$) is described as a control law obtained by optimising a quadratic function subject to condition that the output matches a reference value over a future constant range.
[BKC92]	Stable Generalised Predictive Control ($SGPC$) as an algorithm with guaranteed stability is formulated.
[AP92]	The connection between linear programming and robust model predictive control is made using impulse-response models.
[LGM92]	For the first time predictive control, as discrete time technique, is treated in a multi-rate sample-data framework
[Soe92]	One of the first contributions made at the level of unifying various formulations of predictive control existing up to date.
[KR93]	The multivariable $SGPC$ was derived as an extension of the original algorithm, more insight into the problem being added.
[RK93]	The original $SGPC$ formulation is extended to account for constraints, feasibility issues being addressed as well.
[DC93]	The stability of the GPC scheme is guaranteed using end-point weighting of the cost function which gives the required monotonicity to the cost function's associated Difference Riccati Equation.
[YC93]	Exponential weighting is employed for the first time in the context of predictive control cost function to achieve a prescribed degree of stability.

Author and Year	Comments
[MH93]	A heuristic robustness analysis to a predictive control scheme is provided.
[All94]	A robust predictive controller is designed on-line based on a min-max optimisation that accounts for the worse case — an operation extremely expensive from the computational point of view.
[HM94]	Based on an internal model structure the stability of the constrained $MBPC$ is analysed.
[LY94]	Guidelines for tuning the predictive controller for robust performance are provided based on a fixed tuning of the cost function weights and a manipulation of the observer covariance matrices.
[Mor94]	One of the leading authors in the field is surveying the area promoting at both industrial and academic level predictive control as a choice in the 1990's.
[YC95]	The robustness issue of the predictive control schemes is addressed in a similar manner like in Lee et al. but from the perspective of GPC, the authors' favourite algorithm.
[RKG95b]	The authors of the $SGPC$ formulate the optimisation in a new manner involving a mixed objective, to account for feasibility in the constrained case, and a modified version of the Lawson's least square algorithm to provide the solution.
[CKC96]	Various issues in multiple model predictive control are addressed emphasising the importance of how models are employed in prediction in order to maintain the optimisation convexity.
[RG96]	A two-degree of freedom GPC algorithm with better tracking is suggested, quite similar with the scheme used in H_∞ control.
[KBM96]	An original and neat formulation of predictive control in Linear Matrix Inequalities (LMI) is given, algorithm which accounts for stability, robustness and constraints by enlarging the LMI optimisation according with the number of extra features included. This leads to an expensive solution computationally wise.
[Sco97]	The scheme developed in 1993 by Rawlings et al. is improved leading to an infinite horizon Generalised Predictive Control formulation that enforces input and output constraints.

Table 1.1: Historical review of contributors to $MBPC$

1.3 The thesis overview

This thesis contains eight chapters grouped in two main parts. The first part (Chapters 2, 3, 4) gives the state of the art in $MBPC$ theory containing various formulation of predictive controllers, robust stability and feasibility issues together with a original contribution at the level of implementation guidelines and software. The second part (Chapters 5, 6, 7) describe the development of various aerospace applications for $MBPC$. These range from stability augmentation to flight management and reconfigurable control. In order to emphasise the connection between the two parts we have decided to present some of the theoretical achievements within the practical chapters. This decision is motivated by the fact that all our theoretical contributions were generated as a result of the practical investigations pursued.

The presentation made in the first chapters of the thesis, which have a theoretical content, represents a personal view of the author upon the predictive control field addressing general issues such as stability, performance and feasibility of the on-line optimisation.

The existing $MBPC$ software, unable to give a complete answer to our requirements developed during the research carried out, led us to produce a Development Space software package for Matlab which includes specific features that can have a more general impact upon the community designing and implementing predictive controllers.

Naturally before involving the $MBPC$ technique in various flight control systems structures we have passed through a learning stage followed by the development of various theoretical issues necessary for using the $MBPC$ method in three different aerospace applications: stability augmentation, flight management and reconfigurable control.

As a result, the design chapters contain not only the design procedure but the analysis and evaluation which had to be carried out in order to have a complete image about how our controllers behaved and to show their stability and performance. Several assumptions were made when performing the design, each of them being addressed and motivated within the corresponding context. The impact upon the $MBPC$ by these assumptions was analysed in the first two theoretical chapters.

A brief overview of the contents of each chapter is as follows:

Chapter 1: Introduction In this chapter after motivating the research developed and presented in the thesis we have offered a brief history of the predictive control field together with the present thesis outline.

Chapter 2: Predictive control formulations We gather the relevant model based predictive formulations including the state space and input-output such as: $MBPC$, DMC, GPC, $SGPC$ and infinite horizon GPC. Further understanding of these acronyms can be found using the list presented at the beginning of the thesis.

Chapter 3: Stability, feasibility and robustness The mathematical background addressing the stability of $MBPC$ schemes is developed. This involves both unconstrained and constrained controllers together with the discussion of various methods to ensure stability. The feasibility issue is developed in the case of constrained controllers based on the mixed weights least square algorithm. Towards the end of the chapter the design for robust performance is presented and several guidelines are

given with the aim to offer a better understanding of the advantages as well as the limitations of predictive control.

Chapter 4: Implementation of $MBPC$ schemes In this chapter tuning guidelines for the $MBPC$ parameters were given relying on the theoretical information offered in Chapters 2 and 3. Another aim was to provide the reader with more insight to the problem of filtering within $MBPC$ schemes. The issue of ensuring robust tracking and/or disturbance rejection for a wide class of signals is treated in an unified manner. As a result a novel $MBPC$ approach is considered. An answer to the question "Why we should have an $MBPC$ development space?" is also addressed in this chapter followed by a presentation of a predictive control Development Space (DS). This included a description of the graphic user interface and the a presentation of the DS at work with a MIMO aircraft model. This work has been published in [HM97b, HM97a]

Chapter 5: Stability augmentation using predictive control This chapter applies Model Based Predictive Control ($MBPC$) to the Garteur Research Civil Aircraft Model (RCAM) Design Challenge. Separate controllers are proposed for the longitudinal and lateral channels, each of these having $MBPC$ in the inner loop and a conventional controller in the outer loop. Emphasis is placed on describing the design process. As a main conclusion we can say that $MBPC$ is not recommended for routine use in flight control, but has good potential for higher level control functions such as: on-board flight management and reconfiguration of controllers in event of damage and to the aircraft structure or equipment. This work has been published in [HM96b]

Chapter 6: Flight management using predictive control The aim of this chapter is to investigate the role of the constrained Model Based Predictive Control strategy for flight management. As a conclusion we believe that $MBPC$ represents a technology which should be considered in the transition from stability augmentation to flight management systems. This work has been published in [PHGM97, HM97c].

Chapter 7: The model based predictive reconfigurable flight control The seventh chapter describes advances towards reconfiguration and scheduling in flight control systems using high fidelity models expressed in a quasi-LPV form, Fault Detection and Isolation (FDI), model approximation/simplification and constrained Model Based Predictive Control. The strategy is applied to a well known missile example. This represents a new Reconfigurable Flight Control System (RFCS). The chapter continues with the exploration of the reconfiguration strategy by employing it to a High Incidence Research Model (HIRM) obtained by the courtesy of the Defence and Evaluation Research Agency (DERA). This work will be published in [HM98a, HM98b, HM98c].

Chapter 8: Concluding remarks In the last chapter of the thesis we summarise the main contributions and make several suggestions for the future research to be carried in connection with the new developments presented here.

Chapter 2

Predictive Control Formulations

2.1 The fundamentals of $MBPC$ schemes

The defining feature of model based predictive control ($MBPC$) is the repeated optimisation of a performance objective over a finite horizon extending from a future time (N_1) up to a prediction horizon (N_2) [CM89a, Cla93]. Figure 2.1 characterises the way prediction is used within the $MBPC$ control strategy. Given a set-point $s(k+l)$, a reference $r(k+l)$ is produced by pre-filtering and used within the optimisation of the cost function (2.1). Manipulating the control variable $u(k+l)$, over the control horizon (N_u), the algorithm drives the predicted output $\tilde{y}(k+l)$, over the prediction horizon, towards the reference.

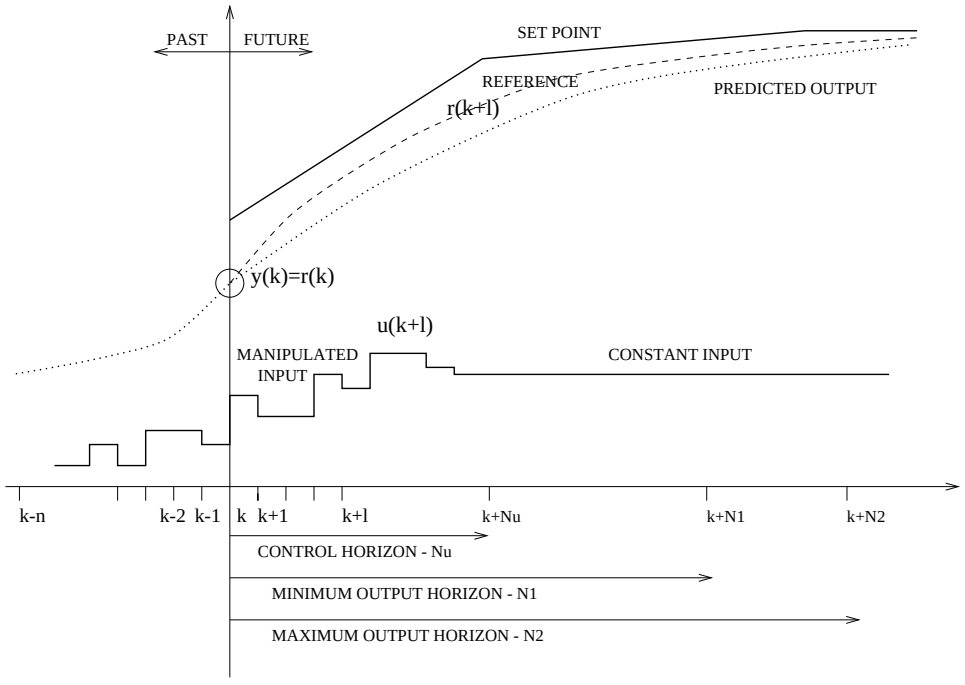


Figure 2.1: The $MBPC$ prediction strategy

The future control movement is determined by minimising the cost function:

$$J(k) = \sum_{l=N_1}^{N_2} \|(\tilde{y}(k+l) - s(k+l)\|^2_{Q(l)} + \sum_{l=0}^{N_u-1} \|\Delta u(k+l)\|^2_{R(l)} \tag{2.1}$$

subject to constraints on:

- the inputs levels $u_l(l) \leq u(l) \leq u_u(l)$ where $k \leq l \leq k + N_u - 1$
- the input rates of change $\Delta u_l(l) \leq \Delta u(l) \leq \Delta u_u(l)$ where $k \leq l \leq k + N_u - 1$
- the output (and state) levels $y_l(l) \leq \tilde{y}(l) \leq y_u(l)$ where $k + N_1 \leq l \leq k + N_2$

In the cost function (2.1) $\Delta u_i(k) = u_i(k+1) - u_i(k)$, $Q(l)$ and $R(l)$ are weights independent of time k and the norm $\|.\|^2_Q$ within the cost function is defined as $\|\alpha\|^2_Q = \alpha^T Q \alpha$. It is assumed that $\Delta u(l) = 0$ for $l \geq k + N_u$ and $s(k) = r(k)$. As in [CM89a, HM94] the optimisation is carried out using a quadratic program (QP) or other algorithms such as mixed weights least square (MWLS) and/or interior point optimisations.

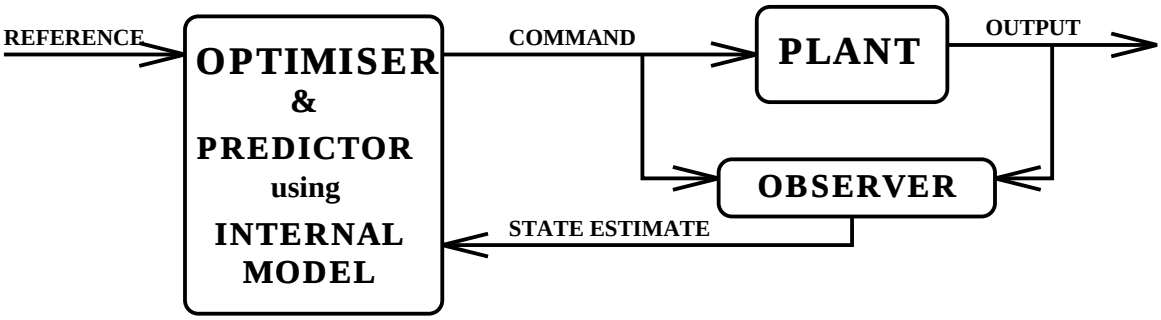


Figure 2.2: The structure of $MBPC$ schemes

The general structure of $MBPC$ schemes, given in Figure 2.2, is the following:

Optimiser contains the constrained cost function. The main task of the optimiser is to compute the present and future manipulated variable moves such that the predicted output follows the reference in a desirable manner.

Predictor employing the internal model and the measurement or estimate of the current state provides the optimiser with future predicted values of states and outputs.

Internal model represents the plant. State-space linear time invariant, impulse/step response or CARIMA models are used for the plant.

Observer provides current state estimates, which can be used in the predictor.

The $MBPC$ scheme has some very appealing attributes which are going to be exploited in this presentation:

- *Simplicity* — the basic idea of $MBPC$ is fairly intuitive, and can be understood without advanced mathematics;

- *Richness* — the common elements of $MBPC$ schemes, such as models, objective functions, prediction horizons, etc, can be tailored to specific problems;

- *Practicality* — the combination of linear dynamics and inequality constraints allows realistic nonlinearities to be handled.

For a plant model which is Linear Time Invariant (LTI), discrete time, with known parameters there are multiple predictive control formulations available such as: state space, preferred for most cases, and input-output, using transfer functions or matrices or finite impulse or step response models. For each process identification used and corresponding model there is a suitable $MBPC$ formulation. 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. More insight into this kind of conversion will be given in the appropriate sections.

Regarding the control and prediction horizons there are small differences between algorithms. Various authors relate the values for such horizons to simulation and theoretical results.

Weighting matrices in the cost function are used in a similar way for all methods although there are some differences regarding the values included. Their off-line tuning is compulsory for most of the algorithms in order to ensure real-time implementation. The exponential weighting formulation of the cost function can be regarded as an exception.

Constraints are handled in a similar manner by most of the $MBPC$ algorithms. Stacking them with the purpose of using standard QP routines is the usual technique. For the GPC method the algorithm did not initial address this issue but now it is possible to include it within the optimisation. Input constraints, as we will be able to see in the next chapter, are manageable from the feasibility and stability point of view but as it was shown in the literature the existence of output constraints can result in instability for processes with model error or discrete models with unstable zeros.

2.2 Model Based Predictive Control (MBPC)

This section addresses a multi-model state space formulation [HM96a] of model based predictive control ($MBPC$), a derivation inspired by the work of [LY94, Hei94, RM93]. The reason for such an approach is that in the alternative input/output formulation such as DMC or classical GPC states are not directly involved in the control algorithm so constraining them is difficult. The multi-model formulation opens the path for reconfigurable control in the case of failures.

In fact this is the formulation used within the $MBPC$ Development Space (DS), as described in Chapter 4 which was used for in the aerospace applications developed in the second part of this work.

2.2.1 Plant model requirements

Considering a possible multi-model context, let index i represent the plant in use:

$$\begin{aligned} x_i(k+1) &= A_i x_i(k) + B_{u_i} u_i(k) + B_{d_i} d_i(k) + B_{w_i} w_i(k) \\ y_i(k) &= C_i x_i(k) + \nu(k) \end{aligned} \tag{2.2}$$

where $x_i(k) \in \mathbb{R}^n$ is the system state vector, $u_i(k) \in \mathbb{R}^m$ is the system control input vector and $y_i(k) \in \mathbb{R}^p$ is the vector of outputs, all variables being measured at time k. The vector of outputs free from measurement noise is $y_{x_i}(k) = C_i x_i(k) + w_i(k)$. It is important to note that in the event of a complete actuator failure, the system and respective model state dimension may change such that $x_i(k) \in \mathbb{R}^{n_i}$.

The 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, $w_i(k)$ represents the input disturbance and $\nu(k)$ is an unmeasured output disturbance vector. It is assumed that each pair $(A_i, (B_{u_i}, B_{d_i}))$, with controllable modes of (A_i, B_{d_i}) being stable, is stabilizable and the pair (A_i, C_i) detectable. There is no other particular requirement for the plant model.

2.2.2 State space formulation

We express the state vector in terms of the change in the manipulated variable (at time k): $\Delta u_i(k) = u_i(k+1) - u_i(k)$, the state disturbance (at same time k): $\Delta d_i(k) = d_i(k+1) - d_i(k)$ and the output disturbance (at sampling time k): $\Delta w_i(k) = w_i(k+1) - w_i(k)$. In such a case, the state space model becomes:

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

where $y_{xi}(k+1)$ corresponds to a vector of outputs free from measurement noise. The augmented system will be referred to by the realization $\hat{A}$, $\hat{B}$, $\hat{C}$:

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

Certain elements of the state vector may not be available for measurement at time k so an observer will be employed.

2.2.3 State prediction

As mentioned, since certain elements of the state vector may not be available for measurement at time k we will determine a set of state estimates for each model of the plant

in use. The estimated states will be either the states of the main system or states of that system augmented by the disturbance variables, as shown in the previous section.

The vector of augmented state and output estimates $\left[\Delta x_i(k)^T \quad y_{x_i}(k)^T\right]^T$ will be denoted as $\hat{x}_i(k) \in \mathbb{R}^{(n+\hat{p})}$ (where $\hat{p}$ represents the number of outputs uncorrupted by measurement noise) and they will rely upon selecting a particular observer. So, in general, state estimation will take the following form at time $k+j$ ($j = 0 \ldots N_2 - 1$) from the prediction horizon N_2:

$$\hat{x}(k+j+1) = F_i(j)\hat{x}(k+j) + G_i(j)\Delta u_i(k+j) + H_i(j)y_i(k+j) \tag{2.3}$$

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

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

A typical simplification such as: $H_i(j) = 0$, $F_i(0) \neq F_i(1)$ and $F_i(j) = F_i(j+1)$ for $j > 0$ may occur. In this case the state estimator is constructed in the following way:

$$\hat{x}_i(k+1) = F_i(0)\hat{x}_i(k) + G_i(0)\Delta u_i(k) + H_i(0)y_i(k) \tag{2.4}$$

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

From the earlier definition of the state space description (2.2) and using (2.3) the vector of future state estimates may be written as:

$$\hat{X}_i(k) = \begin{bmatrix} \hat{x}_i(k+1) \\ \vdots \\ \hat{x}_i(k+N_2) \end{bmatrix} = \mathcal{F}_i(k)\hat{x}_i(k) + \mathcal{G}_i\Delta U_i(k) + \mathcal{H}_i(k)y_i(k) \tag{2.5}$$

In the above equations $\Delta U_i(k) = \left[\Delta u_i(k)^T, \ldots, \Delta u_i(k+N_u-1)^T\right]^T$ denotes the vector of future manipulated variable increments and $\hat{X}_i(k) = \left[\hat{x}_i(k+1)^T, \ldots, \hat{x}_i(k+N_2)^T\right]^T$ denotes the vector of the change of state estimates which is dependent only upon $\hat{x}_i(k)$ the state estimate at time k. Therefore it is necessary to have a state estimator to determine current state estimate $\hat{x}_i(k)$.

Now we are able to define the matrices involved in equation (2.5), these identities being

the result of stacking the equation (2.3) for $\hat{x}_i(k+1)$ up to $\hat{x}_i(k+N_2)$, see [Hei94] as well:

$$\mathcal{F}_i(k) = \begin{bmatrix} F_i(0) \\ F_i(1)F_i(0) \\ \vdots \\ \prod_{j=N_2-1}^{0} F_i(j) \end{bmatrix}$$

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

$$\mathcal{H}_i(k) = \begin{bmatrix} H_i(0) \\ F_i(1)H_i(0) \\ \vdots \\ \prod_{j=N_2-1}^{1} H_i(0) \end{bmatrix}$$

where $\prod_{\lambda=\epsilon}^{\delta} R(\lambda)$ denotes the product: $R(\epsilon)R(\epsilon-1)R(\epsilon-2)\dots R(\delta)$ for $\epsilon > \delta$, N_u and N_2 with $N_u \leq N_2$ are the control and prediction horizon, respectively. The optimisation assumes that $\Delta u_i(k+j) = 0$ for $j \geq N_u$.

Regarding the prediction equations a warning has to be made. The computations made can lead to wrong answers if some of the elements in A^i may become very large or small relative to others. This can lead to problems since the arithmetic employed by the current computers is finite-precision. In such cases iterative methods or "Horner's method" has to be employed.

2.2.4 The model associated constraints

For each model set i we have an associated constraint set expressed as linear inequalities. All the inequalities are defined by stacking together the corresponding scalar inequalities:

- Control signals (actuator limits)
  Stacking over the control horizon the inputs $U_i(k) = \left[u(k+1)^T, \dots, u(k+N_u)^T\right]^T$ we are able to write the linear inequalities that express the lower and the upper limit values in the following form:

$$m_{min_i}(k) \leq U_i(k) \leq m_{max_i}(k) \tag{2.6}$$

  where $m_{min_i}(k)$ and $m_{max_i}(k)$ contain all minimum and maximum rate values across the control horizon. An even more compact form of writing 2.6 is:

$$\begin{aligned} M_i U_i(k) &\leq m_i(k); \\ m_i(k) &= \begin{bmatrix} -m_{min_i}(k) \\ m_{max_i}(k) \end{bmatrix} \end{aligned}$$

- Control signals increments(actuator rates)
  Proceeding with a similar stacking procedure over the control horizon we write the linear inequalities that express the minimum and the maximum limit values in the form:

$$l_{min_i}(k) \leq \Delta U_i(k) \leq l_{max_i}(k) \tag{2.7}$$

  where $l_{min_i}(k)$ and $l_{max_i}(k)$ contain all the limit values across the control horizon. Following the same path as for the control signals we write the inequality (2.7) in the form of:

$$\begin{aligned} L_i \Delta U_i(k) &\leq l_i(k); \\ l_i(k) &= \begin{bmatrix} -l_{min_i}(k) \\ l_{max_i}(k) \end{bmatrix} \end{aligned}$$

- State constraints for the augmented system (variation of the states and the outputs of the model)
  Over the prediction horizon the vector: $\hat{X}_i(k)$ and the minimum and maximum values admissible for the system states across the N_2 horizon are defined by the following linear inequality:

$$n_{min_i}(k) \leq \hat{X}_i(k) \leq n_{max_i}(k). \tag{2.8}$$

  The compact form of the inequality 2.8 is:

$$\begin{aligned} N_i \hat{X}_i(k) &\leq n_i(k) \\ n_i(k) &= \begin{bmatrix} -n_{min_i}(k) \\ n_{max_i}(k) \end{bmatrix}. \end{aligned}$$

  If we would like to have more general inequalities concerning the states we might need to allow changes in the expression of N_i, in other words the present formulation might have to be slightly altered.

In order to pose the problem as a standard quadratic programming optimisation we need to have all the inequalities stacked one upon another as a single linear inequality constraint on the vector of future input changes ΔU_i. The final result of stacking constraints is:

$$\mathcal{D}_i \Delta U_i(k) \leq \mathcal{E}_i \begin{bmatrix} u_i(k-1) \\ \hat{x}_i(k) \\ y_i(k) \\ c_i(k) \end{bmatrix} \tag{2.9}$$

$$\mathcal{D}_i = \begin{bmatrix} L_i \\ M_i \Lambda \\ N_i \mathcal{G}_i \end{bmatrix}$$

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

where Λ and $\mathcal{I}$ are defined as follows:

$$\begin{aligned}\Lambda &= \begin{bmatrix} I_m & 0 & \dots & 0 \\ I_m & I_m & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ I_m & I_m & I_m & I_m \end{bmatrix} \\ \mathcal{I} &= [\mathcal{I}_m, \dots, \mathcal{I}_m]^T .\end{aligned}$$

where I_m is a unit matrix, $I_m \in \mathbb{R}^{m \times m}$.

The vector $c_i(k)$ is defined as: $c_i(k) = [l_i(k)^T, m_i(k)^T, n_i(k)^T]^T$ where $l_i(k) \in \mathbb{R}^\lambda$, $m_i(k) \in \mathbb{R}^\mu$ and $n_i(k) \in \mathbb{R}^\nu$. We should note that in general $\lambda \neq \mu \neq \nu$. Note that control $u_i(k+1)$ is still to be determined at time k.

2.2.5 Design objectives

The purpose of stacking inequalities in the form of (2.9) was to minimise the following cost functional:

$$J_i(k) = \sum_{j=1}^{N_2} \|(C_i \hat{x}_i(k+j) - s_i(k+j))\|^2_{Q_i(j)} + \sum_{j=0}^{N_u-1} \|\Delta u_i(k+j)\|^2_{R_i(j)} \tag{2.10}$$

using a constrained optimisation algorithm. In the cost function (2.10) $s_i(k)$ represent the reference vector for 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. The cost function can be written as in [Hei94]:

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

where

$$\begin{aligned}\mathcal{Q}_i &= diag\left[Q_i(1), \dots, Q_i(N_2)\right] \\ \mathcal{R}_i &= diag\left[R_i(1), \dots, R_i(N_u)\right] \\ \mathcal{C}_i &= diag\left[\hat{C}_i, \dots, \hat{C}_i\right] \\ \mathcal{S}_i(k) &= \left[s_i(k+1)^T, \dots, s_i(k+N_2)^T\right]^T .\end{aligned}$$

The term $\mathcal{K}_i(k)$ contains all the other terms independent of $\Delta U_i(k)$ and therefore can be omitted from $J_i(k)$ without affecting the optimisation.

In the cost function there are several types of variables:

- constants of the process model, connected with estimation and prediction: $\mathcal{G}_i$, $\mathcal{C}_i$, $\mathcal{Q}_i$, $\mathcal{R}_i$, $\mathcal{F}_i$ and $\mathcal{H}_i$.
- parameters varying with k and the model set i such as $\hat{x}_i(k)$, $y_i(k)$ and $\mathcal{S}_i(k)$, being updated at each new optimisation.

- parameters constant during the optimisation algorithm like N_2, and N_u that define the various horizons.
- the real optimisation variable vector $\Delta U_i(k)$.

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

$$\min_{\Delta U_i(k)} \left\{ \Delta U_i(k)^T \mathcal{A}_i \Delta U_i(k) + \left[\hat{x}_i(k)^T, y_i(k)^T, \mathcal{S}(k)^T\right] \mathcal{B}_i \Delta U_i(k) \right\} \tag{2.11}$$

subject to the constraints:

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

where the matrices $\mathcal{D}_i$ and $\mathcal{E}_i$ and the vector $c_i(k)$ are defined as shown in Section 2.2.4, equation (2.9). The other matrices involved in the optimisation problem (2.11 are defined as:

$$\begin{aligned} \mathcal{A}_i &= \mathcal{G}_i^T \mathcal{C}_i^T \mathcal{Q}_i \mathcal{C}_i \mathcal{G}_i + \mathcal{R}_i \\ \mathcal{B}_i &= 2\left[\mathcal{C}_i \mathcal{F}_i \quad \mathcal{C}_i \mathcal{H}_i \quad -I_{N_2 p}\right]^T \mathcal{Q}_i \mathcal{C}_i \mathcal{G}_i \end{aligned}$$

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

2.2.6 Further details on the unconstrained $MBPC$

The gain feedback case

The unconstrained state feedback case is considered in Figure 2.3 which shows a block diagram of $MBPC$ when states are available for feedback, and when there are no constraints.

For the computation of the matrices K_{MBPC} and L_{MBPC}, characterising the unconstrained $MBPC$, see [LY94] and Section 2.2.7.

From the Figure 2.3:

$$\Delta u(k) = K_{MBPC} \left[S(k) - L_{MBPC} \begin{bmatrix} \Delta x(k) \\ y(k) \end{bmatrix} \right]$$

where:

$$\begin{aligned} K_{MBPC} &= \begin{bmatrix} I & 0 & \dots & 0 \end{bmatrix} \left[(\mathcal{C}^u)^T \mathcal{Q}^T \mathcal{Q} \mathcal{C}^u + \mathcal{R}^T \mathcal{R} \right]^{-1} \mathcal{R}^T \mathcal{Q}^T \mathcal{Q}, \\ L_{MBPC} &= \begin{bmatrix} \mathcal{C}^x & \mathcal{C}^y \end{bmatrix}, \end{aligned}$$

$\hat{x}(k) = \begin{bmatrix} \Delta x(k) \\ y(k) \end{bmatrix}$, matrices $\mathcal{C}^u$, $\mathcal{C}^x$ and $\mathcal{C}^y$ are defined using a linear internal model of the plant and $\mathcal{R}$, $\mathcal{Q}$ are the cost-function weighting matrices over the prediction (N_y) and

control (N_u) horizon, respectively. $S(k)$ is the set-point and $R(k)$ the reference vectors across the N_y horizon at time k. In this case, for simplicity, we assume $F(z) = I$ (*i.e.* $R(k) = S(k)$).

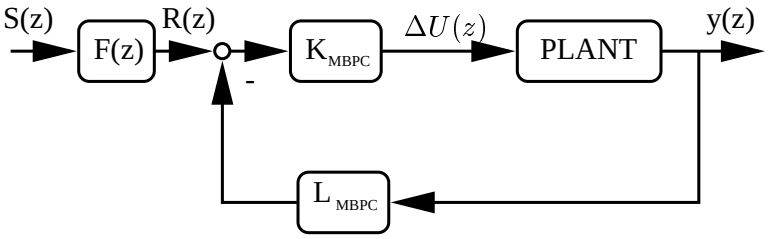


Figure 2.3: Filtering problem formulation

The parameter dependent $MBPC$ closed loop realisation

As we have seen in Section 2.2 the optimisation problem in the unconstrained case can be re-cast in the following form:

$$\min_{\Delta U_i(k)} \Delta U_i(k)^T \mathcal{A}_i \Delta U_i(k) + \left[\hat{x}_i(k)^T, y_i(k)^T, \mathcal{S}(k)^T\right] \mathcal{B}_i \Delta U_i(k) \tag{2.12}$$

The augmented system matrices will be referred to by the realization $\left[\begin{array}{c|c} \hat{A}_i & \hat{B}_i \\ \hline \hat{C}_i & 0 \end{array}\right]$ where it is assumed that the plant models are defined by $\left[\begin{array}{c|c} A_i & B_i \\ \hline C_i & D_i \end{array}\right]$ where the index i represents the model in use.

Using results from [Fle91] and assuming the availability of the state for measurement the optimal solution is given by the first control move $\Delta u_i(k) = \Sigma_1 \Delta U_i^*(k)$ where:

$$\Delta U_i^*(k) = -\frac{1}{2} \mathcal{A}_i^{-1} \mathcal{B}_i^T \Sigma_2 \left[\Delta x_i(k)^T, y_i(k)^T, s_i(k)^T\right]^T$$

for $\Sigma_1 = \begin{bmatrix} I_{m\times m} & 0_{m\times m} & \dots & 0_{m\times m} \end{bmatrix}$ and, assuming a step type of set-point, Σ_2 is defined as:

$$\Sigma_2 = \begin{bmatrix} I_{n\times n} & 0_{n\times p} & 0_{n\times p} \\ 0_{p\times n} & I_{p\times p} & 0_{p\times p} \\ 0_{p\times n} & I_{p\times p} & 0_{p\times p} \\ 0_{p\times n} & 0_{p\times p} & I_{p\times p} \\ \dots & \dots & \dots \\ 0_{p\times n} & 0_{p\times p} & I_{p\times p} \end{bmatrix}$$

Hence, the open loop is closed, as shown in Figure 2.4, with the following feedback:

$$\Delta u_i(k) = K_i \left(\left[\begin{array}{c|c} A_i & B_i \\ \hline C_i & D_i \end{array}\right], \Theta \right) \begin{bmatrix} \Delta x_i(k) \\ y_i(k) \\ s_i(k) \end{bmatrix} \tag{2.13}$$

where Θ is a vector defined by stacking together the diagonal entries of the weighting matrices Q and R. Ecuation (2.13) and the following development will be used throughout the automatic tuning procedure described in Section 7.2.4.

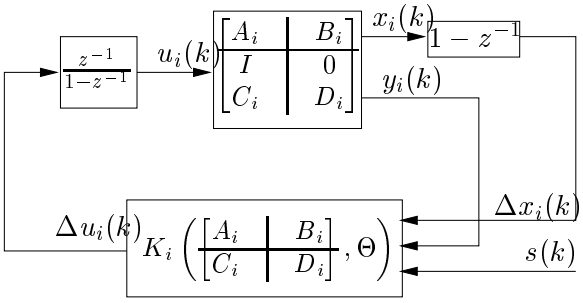


Figure 2.4: The unconstrained $MBPC$ closed loop

This loop can be restructured in the manner depicted in Figure 2.5 where the augmented plant, based on the model realisation $\left[\begin{array}{c|c} A_i & B_i \\ \hline C_i & D_i \end{array}\right]$, is:

$$\left[\begin{array}{c|c} \bar{A}_i & \bar{B}_i \\ \hline \bar{C}_i & \bar{D}_i \end{array}\right] = \left[\begin{array}{cc|c} A_i & 0 & B_i \\ C_iA_i & I & C_iB_i + D_i \\ \hline I & I & 0 \end{array}\right].$$

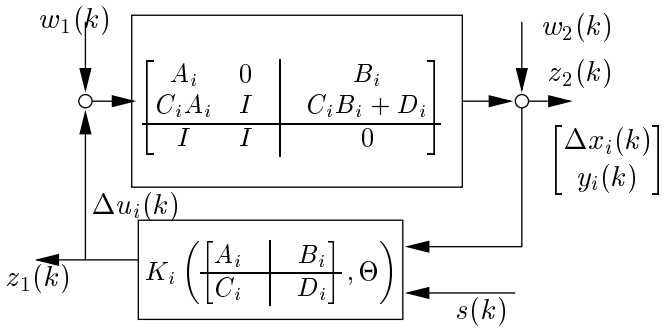


Figure 2.5: The unconstrained $MBPC$ restructured closed loop

For a zero reference vector the feedback law is:

$$\Delta u_i(k) = K_i \begin{bmatrix} \Delta x_i(k) \\ y_i(k) \end{bmatrix} = K_i \hat{x}_i(k)$$

where K_i has the following form:

$$K_i = \Sigma_1 \mathcal{A}_i^{-1} \mathcal{G}_i^T \mathcal{C}_i^T \mathcal{Q}_i^T \begin{bmatrix} \mathcal{C}_i & \mathcal{F}_i \end{bmatrix}$$

Considering the transfer matrix from the input and output disturbance signals ($w_1(k)$, $w_2(k)$), respectively, to the controller and plant output signals ($z_1(k)$, $z_2(k)$), respectively, we can picture the following structure, see Figure 2.6:

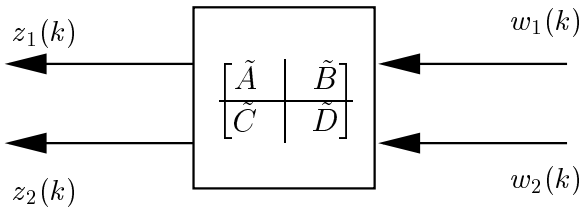


Figure 2.6: The four-block transfer matrix

Where matrices $\tilde{A}$, $\tilde{B}$, $\tilde{C}$, $\tilde{D}$ are defined as follows:

$$\begin{aligned}
\tilde{A} &= \bar{A} + \bar{B}(I - \bar{K}\bar{D})^{-1}\bar{K}\bar{C} \\
\tilde{B} &= \begin{bmatrix} \bar{B}\left[I + (I - \bar{K}\bar{D})^{-1}\bar{K}\bar{D}\right] & \bar{B}(I - \bar{K}\bar{D})^{-1}\bar{K} \end{bmatrix} \\
\tilde{C} &= \begin{bmatrix} (I - \bar{K}\bar{D})^{-1}\bar{K}\bar{C} \\ \bar{C}\left[I + \bar{D}(I - \bar{K}\bar{D})^{-1}\bar{K}\right] \end{bmatrix} \\
\tilde{D} &= \begin{bmatrix} (I - \bar{K}\bar{D})^{-1}\bar{K}\bar{D} & (I - \bar{K}\bar{D})^{-1}\bar{K} \\ \bar{D}\left[I + (I - \bar{K}\bar{D})^{-1}\bar{K}\bar{D}\right] & \left[I + (I - \bar{K}\bar{D})^{-1}\bar{K}\right] \end{bmatrix}
\end{aligned}$$

2.2.7 Comments on the state space formulation

In order to enable a comparison of the various formulations presented, a few characteristics that will summarise the $MBPC$ strategy are given. Regarding the type of model employed we acknowledge the following features:

- The state space formulation can be obtained in straightforward manner from most modelling techniques.
- Numerical algorithms based on state space representation used for prediction and estimation allow reliable computations.
- The state space models give the opportunity to model disturbances in a convenient way in either a deterministic or a stochastic sense.

Compared with some other approaches, to be presented, we remark that the assumption of a stable process is not necessary for computations to be done. The 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 a few simplifications can be made.

The MIMO state space predictive control algorithm is characterised by the following features:

- The least squares method produces a nice analytical solution using the state space models in the unconstrained case.
- Constraints upon states can be handled with this approach in a convenient manner because of $\hat{x}_i(k) = \left[\Delta x_i(k)^T y_{x_i}(k)^T\right]^T$ which is part of the cost function.
- Packing constraints in a compact vector is a straightforward procedure.

Of course we can argue that none of foregoing features depend on using a state-space model but these models allow an exact and compact way to store the information about the plant, an easy manipulation of the closed loop for analysis purposes, more design freedom and last but not least an easier plant parameter estimation in the adaptive $MBPC$ case.

Constrained $MBPC$ algorithms with linear models result in piecewise LTI laws. If the plant is associated with a particular set of constraints:

$$\mathcal{D}_i \cdot \Delta U_i(k) \leq \mathcal{E}_i \cdot \begin{bmatrix} u_i(k-1) \\ \hat{x}_i(k) \\ y_i(k) \\ c_i(k) \end{bmatrix} \tag{2.14}$$

and using information from Fletcher's book [Fle91] when there is equality in the above linear inequality (*i.e.* for the active constraints), the optimal solution to the minimisation problem (2.11) is given by :

$$\begin{bmatrix} 2 \cdot \mathcal{A}_i & -\mathcal{D}_i^T \\ -\mathcal{D}_i & 0 \end{bmatrix} \cdot \begin{bmatrix} \Delta U_i^*(k) \\ \lambda^* \end{bmatrix} = - \begin{bmatrix} -\mathcal{B}_i^T \cdot \left[\hat{x}_i(k)^T, y_i(k)^T, \mathcal{S}_i(k)^T\right]^T \\ \mathcal{E}_i \cdot \left[u_i(k-1^T), \hat{x}_i(k)^T, y_i(k)^T, c_i(k)\right]^T \end{bmatrix} \tag{2.15}$$

where $\Delta U_i^*(k)$ is the optimal control sequence over the output horizon and λ^* is a set of Lagrange multipliers, corresponding to the optimisation of the $J_i(k)$ cost function.

If the solution of the optimisation problem is unique and the optimisation is feasible then the inverse of the matrix:

$$\begin{bmatrix} 2 \cdot \mathcal{A}_i & -\mathcal{D}_i^T \\ -\mathcal{D}_i & 0 \end{bmatrix}^{-1} = \begin{bmatrix} \mathcal{L}_{1i} & \mathcal{L}_{2i} \\ \mathcal{L}_{3i} & \mathcal{L}_{4i} \end{bmatrix} \tag{2.16}$$

exists and the optimal solution becomes:

$$\begin{aligned} \Delta U_i^*(k) = &-\mathcal{L}_{1i} \cdot \mathcal{B}_i^T \cdot \left[\hat{x}_i(k)^T, y_i(k)^T, \mathcal{S}_i(k)^T\right]^T + \\ &+\mathcal{L}_{2i} \cdot \mathcal{E}_i \cdot \left[u_i(k-1), \hat{x}_i(k)^T, y_i(k)^T, c_i(k)^T\right]^T \end{aligned} \tag{2.17}$$

As long as the plant model, the cost function and the constraints are independent of time then time invariance of the control law occurs. If the optimisation is repeated each step under the stated assumptions the feedback law is linear time invariant. A particular situation is when active constraints, which are associated with a particular model, remain unchanged as long as that model is employed. As a result, the control law is LTI as the set of of active constraints remains unchanged. In conclusion, the control law has a variable structure by switching between a finite number of control laws. This idea can be used for tuning the $MBPC$ parameters to guarantee robust stability by designing of a set of linear feedback controllers, each corresponding to a constraint combination.

It has been noted, see [MH93], 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.

If our controller is likely to operate a lot at the constraints, analysis has to be carried out in order to ensure the stability of the closed loop with that set of constraints considered active. In fact this can be regarded as the only situation when we can provide a tuning straightaway in the constrained case.

2.3 Dynamic Matrix Control (DMC)

One of the first formulations of predictive control was DMC [PGC82, PG88], commercial products still using this type of controllers which are based on impulse or step response models. As a result these type of models were the types that attracted the academic community until the development of the GPC formulation [CMT87] which popularised transfer function models.

Therefore the main reason for addressing here the DMC formulation is its historical value together with the aim of showing the equivalence between this formulation using step or impulse response models and the foregoing one employing state space models.

The equivalence between impulse and step response models which will be shown in a later section of this chapter allows us to discuss them together. Despite their property of being an intuitive concept the multivariable step or impulse response models come together with few drawbacks that point us towards the direction of using state space models and the corresponding formulation.

Among these drawbacks we can mention the fact that they can be used only with asymptotically stable plants for which it is sometimes impractical to excite them with step kind of inputs in normal operation. Despite a good estimation of the steady state gains, the identification of models via this method emphasises low frequencies which is sometimes insufficient for feedback control. Moreover these type of models are adequate only when all the controlled variables are outputs otherwise an auxiliary mechanism is necessary to deal with the unmeasured outputs. As mentioned in [Mac98], the necessity of impulse or step response models for capturing complicated patterns and for representing the plant delays are in fact false myths.

The DMC algorithm as an input/output formulation of predictive control works equally well with and without constraints. For the unconstrained case DMC has a simple least square solution, a feature shared by all predictive control algorithms. This is not the case when there are constraints. Then it requires constrained optimisation such as linear or quadratic programming. In practice it seems that DMC is usually implemented including a linear programming (LP) algorithm rather than a quadratic (QP).

During operation, the process constraints limit can be changed in response to instrument or plant failure. After performing the controller synthesis using DMC its properties such as stability is checked via simulation of the closed loop.

2.3.1 Equivalence of state space, impulse and step response models

Using the following state space representation for the plant model:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ y(k) &= Cx(k) + Du(k) \end{aligned}$$

and supposing that $x(0) = 0$, we apply a impulse input vector (i.e. $u(0) \neq 0$ and $u(k) = 0$, $\forall k > 0$, $k \in \mathbb{N}$) to get the following sequence of states and outputs:

$$\begin{array}{rcc|rcc} x(0) & = & 0 & y(0) & = & Du(0) \\ x(1) & = & Bu(0) & y(1) & = & CBu(0) \\ & \vdots & & & \vdots & \\ x(k) & = & A^{k-1}Bu(0) & y(2) & = & CABu(0) \\ & & & y(k) & = & CA^{k-1}Bu(0) \\ & \vdots & & & \vdots & \end{array}$$

From the above equations it is clear that the impulse response matrix sequence is given by:

$$\begin{aligned} h(0) &= D \\ h(1) &= CB \\ h(2) &= CAB \\ &\vdots \\ h(k) &= CA^{k-1}B \\ &\vdots \end{aligned}$$

where $CA^{k-1}B$ is often called the k'th Markov parameter of the state space model.

Using the definition of the step response matrix expressed using the impulse response matrix sequence we are able to write the step response sequence $g(t) = \sum_{k=0}^{t} h(k)$ as:

$$\begin{aligned} g(0) &= D \\ g(1) &= CB + D \\ g(2) &= CAB + CB + D \\ &\vdots \\ g(k) &= \sum_{i=0}^{k-1} CA^iB + D \\ &= C\left(\sum_{i=0}^{k-1} A^i\right)B + D \\ &\vdots \end{aligned}$$

The reverse transformations from impulse response models to state space can be obtained by building up the block Hankel matrix from the impulse response matrices and by recalling that:

$$h(k) = \begin{cases} D & (k = 0) \\ CA^{k-1}B & (k > 0) \end{cases}$$

Hence we have the following identity:

$$\begin{bmatrix} h(1) & h(2) & \dots \\ h(2) & h(3) & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} CB & CAB & \dots \\ CAB & CA^2B & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$
$$= \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \end{bmatrix} \begin{bmatrix} B & AB & A^2B & \dots \end{bmatrix}$$

In order to perform this conversion a finite impulse response has to be used. As well the foregoing equations assume that the data is generated exactly by a linear system with finite-dimensional state space.

2.3.2 From the plant dynamic matrix to the optimisation set-up

The representation given in this section is based on the information from [PG88]. The DMC controller for a system is based on a step response model of the process, given by:

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

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

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 $\Delta u(k+j)$ for $j \geq 0$ as:

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

A common assumption is that the estimate of future values of the disturbance $n(k+j)$ is the current value $n(k)$ (i.e. $n(k+j) = n(k) = y(k) - \sum_{i=1}^{\infty} g_i \Delta u(k-i)$). This is just a simple way to see disturbance problem and more sophisticated estimates are possible. Such an approach assumes that all disturbances are steps.

Hence the prediction of the output can be written as:

$$\hat{y}(k+j) = \sum_{i=1}^{j} g_i \Delta u(k+j-i) + \sum_{i=j+1}^{\infty} g_i \Delta u(k+j-i) + y(k) - \sum_{i=1}^{\infty} g_i \Delta u(k-i) \quad (2.18)$$

Using the backward shift operator z^{-1} in the discrete case equation (2.18) may be written in a more compact form:

$$\hat{y}(k+j) = G_j(z^{-1}) \Delta u(k+j) + p_j \quad (2.19)$$

where $G_j(z^{-1})$is a polynomial in the backward shift operator z^{-1} ($G_j(z^{-1}) = g_1 z^{-1} + \ldots + g_i z^{-i}$) and p_j represent the free response of the system, given by:

$$p_j = y(k) + \sum_{i=j+1}^{\infty} g_i \Delta u(k+j-i) - \sum_{i=1}^{\infty} g_i \Delta u(k-i) = y(k) + \sum_{i=1}^{\infty} (g_{j+i} - g_i)\Delta u(k-i) \tag{2.20}$$

The reason for calling p_j the free response of the process is that if we consider $\Delta u(k+j) = 0$ for $j = 0 \ldots N2$ in the DMC case then the equation (2.19) becomes $\hat{y}(k+j) = p_j$ where $N2$ is called maximum output horizon. As in Section 2.2 we define N_1 to be the minimum output horizon. With the assumption of an asymptotically stable process, as discussed above, the coefficients g_i tend to a constant value, so equation 2.20 can be simplified to:

$$p_j = y(k) + \sum_{i=1}^{N} (g_{j+i} - g_i)\Delta u(k-i) \tag{2.21}$$

where N is the number for which $(g_{j+i} - g_i) \approx 0 \quad i > N \quad j = N_1 \ldots N$.

An important note is that if the process is not asymptotically stable, then N does not exist and as result the simplification cannot be made (i.e. p_j cannot be computed using the formulation based on step or impulse response models). We should remark here that with a state-space model or the transfer matrix formulation the computations can be made.

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

$$J = \sum_{j=1}^{N_1} [\hat{y}(k+j) - s(k+j)]^2 + \sum_{j=0}^{N_u - 1} [\lambda \Delta u(k+j)]^2 \tag{2.22}$$

with respect to $\Delta u(k+j)$. In this expression $s(k)$ is a given reference at time instant t, λ a weighting factor for the increment of the manipulated variable $u(k)$ and $\hat{y}(k+j)$ are the output prediction values at time instants $(k+j)$. These predictions depend strictly upon the manipulated variables $\Delta u(k+j)$ and can be obtained from the process model. It is assumed that $\Delta u(k+j) = 0$ for $j \geq N_u$.

With the predictions as in equation (2.19) the cost function (2.22) can be written after a bit of algebra in the matrix form:

$$J = \Delta U^T(k)[G^T G + \lambda I]\Delta U(k) - 2{e_0}^T G \Delta U(k) + {e_0}^T e_0 \tag{2.23}$$

where $\Delta U(k)^T = [\Delta u(k), \Delta u(k+1), \ldots]$ is the vector of future controls to be computed, ${e_0}^T = [s(k+N_1) - p_{N_1}, s(k+N_1+1) - p_{N_1+1}, \ldots, s(k+N_2) - p_{N_2}]$ is the vector of known future errors between the set point $s(k+j)$ and the free response of the system p_j and G,

having real terms, is a matrix given by:

$$G = \begin{bmatrix} g_{N_1} & \dots & g_1 & 0 & \dots & & 0 \\ g_{N_1+1} & \dots & g_2 & g_1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ g_{N_2} & \dots & \dots & \dots & \dots & \dots & g_{N_2-N_{u+1}} \end{bmatrix}$$

The meaning of most of the variables involved in the DMC algorithm as well as the significance of the various horizons can be seen in Figure 2.1. In the next section the unconstrained as well as constrained case of DMC will be considered in relation to implementation issues. This section was included with the solely aim of giving the parallel with the development presented in Section 2.2 and Section 2.4.

2.3.3 The algorithm

The DMC algorithm has some special features in the unconstrained case. The cost function J, see equation (2.22), can be minimised using standard techniques like the "least square method" that provides a nice analytical solution in the unconstrained case:

$$\Delta U(k) = \left[G^T G + \lambda I\right]^{-1} G^T e_o$$

This solution will provide N_u values for the manipulated variable $\Delta u(k)$, 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 $\Delta u(k)$.

We intend to present here some special aspects regarding the implementation of the unconstrained case. Using the following form to express the model:

$$y(k) = \sum_{i=1}^{N} g_i \Delta u(k-i) + g_N u(k-N-1) + n(k) \tag{2.24}$$

we could use as well the relation between the coefficient of the step and impulse response models $h_i = g_i - g_{i-1} = \Delta g_i$ in order to reduce the impulse response model:

$$y(k) \approx \sum_{i=1}^{N} h_i u(k-i) + n(k) \tag{2.25}$$

Equation (2.25) provides a valid approximation for asymptotically stable processes for N big enough. Using equation (2.24) and following the procedure described in Section 2.3.2 for computing the predictions p_j we obtain a similar result:

$$\begin{aligned} \hat{y}(k+j) = \sum_{i=1}^{j} g_i \Delta u(k+j-i) + \sum_{i=j+1}^{N-1} \Delta u(k+j-i) + \\ + g_N u(k+j-N) + y(k) - \sum_{i=1}^{N-1} g_i \Delta u(k-i) - g_N u(k-N) \end{aligned} \tag{2.26}$$

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

$$\hat{y}(k+j) = G_j(z^{-1})\Delta u(k+j) + p_j$$

where the expression of the free response is:

$$p_j = y(k) + \sum_{i=1}^{N-j}(g_{j+i} - g_i)\Delta u(k-i) - \sum_{i=N-j+1}^{N-1} g_i \Delta u(k-i) + g_N(1-z^{-j})u(k+j-N)$$

In the constrained input/output formulation the quadratic cost function to be minimised is 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 $t + N_1 \leq j \leq t + N_2$.

The future estimated 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 (i.e.$y_l(j) \leq \hat{y}(j) \leq y_u(j)$ where $t+N_1 \leq j \leq t+N_2$). When we solve the minimisation 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 constrained case, it is better to have all constraints in a compact form such as $A \cdot x \leq b$, required by most of QP algorithms.

Denoting a lower triangular matrix $L_{N_u m}$:

$$L_{N_u m} = \begin{bmatrix} I_m & 0 & \dots & 0 \\ I_m & I_m & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ I_m & I_m & \dots & I_m \end{bmatrix} \tag{2.27}$$

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:

$$\begin{bmatrix} -L_{N_u}m \\ +L_{N_u}m \\ -I_{N_u}m \\ +I_{N_u}m \\ -S^{N_u}_{N_2-N_1} \\ +S^{N_u}_{N_2-N_1} \end{bmatrix} \cdot \Delta u(k+j) \geq \begin{bmatrix} u(k-1) - u_u(k) \\ \vdots \\ u(k-1) - u_u(k+N_u-1) \\ u_l(k) - u(k-1) \\ \vdots \\ u_l(k+N_u-1) - u(k-1) \\ -\Delta u_u(k) \\ \vdots \\ -\Delta u_u(k+N_u-1) + \Delta u_l(k) \\ \vdots \\ -\Delta u_l(k+N_u-1) \\ \vdots \\ p_{N_1} + s(k+N_1) - y_u(k+N_1) \\ \vdots \\ p_{N_2} + s(k+N_2) - y_u(k+N_2) \\ y_l(k+N_1) - p_{N_1} - s(k+N_1) \\ \vdots \\ y_l(k+N_2) - p_{N_2} - s(k+N_2) \end{bmatrix} \tag{2.28}$$

Here N_2 denotes the maximum output horizon, N_1 the minimum output horizon, $S^{N_u}_{N_2-N_1}$ represent a matrix that helps 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 the time of computation.

It is easy to prove that the constraints can be expressed in the above form (2.28) using certain separations. For example consider the constraints upon inputs $u_l(k+j) \leq u(k+j)$ where $1 \leq j \leq N_u$ we are able to express this linear inequality in the following form:

$$u_l(k+j) \leq u(k-1) + \sum_{i=1}^{j} \Delta u(k+i-1) \tag{2.29}$$

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

$$u_l(k+j) \leq u(k-1) + \begin{bmatrix} I_m & I_m & \dots & I_m \end{bmatrix} \cdot \begin{bmatrix} \Delta u(k) \\ \Delta u(k+1) \\ \vdots \\ \Delta u(t+j-1) \end{bmatrix} \tag{2.30}$$

Stacking the N_u inequalities, one on top of each other, gives the second column defined in relation 2.28. For the right inequality $u(k+j) \leq u_u(k+j)$ we apply a similar algorithm. In such a way the first $N_u \cdot m$ rows from the right column of relation (2.28) are defined. The input rate constraints as well as output constraints are treated in a similar manner. An interesting fact to be noted is that the total number of constraints is $4 \cdot N_u + 2 \cdot N_2 - N_1$.

2.3.4 A short summary

Using the formulation in the backward shift operator (z^{-1}) the model can be written in a more compact form, emphasising the free response of the system. On the other hand, the transparency of the algorithm was improved by using matrix computation. The assumption of an asymptotically stable process is necessary to use step response models.

The future values of the disturbance are approximated with the current value. This represents a strong assumption which despite of its simplicity has proved satisfactory in various applications. The constant output disturbance assumption can lead to various problems when used with unstable plants because of the state prediction which will get worse and worse as the horizon increases due to the fact that in the observer state matrix we retain the modes of the unstable plant model. Of course in the general case is possible to have such a type of disturbance assumption, but to make the observer stable in spite of the plant instability. But DMC does not do this, therefore although the algorithm scheme for estimating disturbances is simple and intuitive it has its own limitations.

Unfortunately constraints upon states are impossible to handle using the DMC standard formulation. Writing the cost function in a matrix form, the vector of future controls (including increments, rates and outputs) can be incorporated directly into the optimisation. When a QP algorithm is used to produce the solution, several assumptions for positivity of the matrices involved have to be made.

2.4 Generalised Predictive Control (GPC)

Papers [CM89a], [Cla93] presented by Clarke describe GPC as a new robust algorithm suitable for challenging adaptive control applications. The authors of GPC state also that the method is simple to derive and to implement. Simulation results show that GPC can cope with control of complex processes under realistic conditions.

In fact for us the main purpose of having included this section is not to present the GPC algorithm but to give the reader enough elements necessary to evaluate the perfect equivalence between this approach and the foregoing ones and to offer the basis in terms of structure which will be used in the next chapter when approaching issues like stability, feasibility and robustness. It can be argued that besides its historical importance the main feature of GPC is that it deals with models expressed in transfer functions which mainly for the SISO case make some of the observations and proofs more obvious than in the case of the state space formulation.

In the Section 2.3.1 we have shown the equivalence between state space, step, and impulse response models. Moving between state space and the transfer function descriptions is equally simple by writing $P(z) = C(zI - A)^{-1}B + D$, an equation which holds for both SISO and multivariable systems. Algorithms were developed and implemented for the more complicated excursion from transfer function to state space models.

In this section and the following we will use information from [CM89a] and [DCT87]. We will start by presenting the CARIMA model (Controlled Auto-Regressive and Integrated Moving Average) and considering a locally-linearised model:

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \nu(k) \tag{2.31}$$

where A and B are polynomials in the backward shift operator (z^{-1}):

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

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

and $u(k)$ is the control variable (input), $y(k)$ is the measured variable (output), $\nu(k)$ is the disturbance term. The disturbance $\nu(k)$ is considered to be of moving average form:

$$\nu(k) = T(z^{-1})\epsilon(k) = \frac{C(z^{-1})}{D(z^{-1})}\epsilon(k) \tag{2.32}$$

where

$$T(z^{-1}) = 1 + t_1 z^{-1} + \ldots + t_{nt} z^{-nt} \tag{2.33}$$

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

In the theory of predictions with a disturbance model the disturbance is modelled as a signal passed through a filter with monic numerator and denominator polynomials and a magnitude adjusted via $\epsilon(k)$. The signal $\epsilon(k)$ is assumed to be unpredictable because otherwise any additional knowledge should have been built into the choice of $C(z^{-1})$ and $D(z^{-1})$.

This model is general enough to allow both stochastic and deterministic disturbances to be modelled. The case of deterministic disturbances is modelled by taking $C(z^{-1}) = 1$ and making the first few values of $\epsilon(k)$ non-zero. In the case of a stochastic disturbance one can consider predicting $\epsilon(k)$ in such a way to optimise the predictions of $y(k+j)$ in some sense. The usual approach taken is the one of a minimum variance prediction (*i.e.* the error between the predicted and actual values is as small as possible).

The GPC algorithm assumes in the wide majority of cases stochastic disturbances for which the denominator of $T(z^{-1})$, the polynomial $D(z^{-1})$, appears as $D(z^{-1}) = \Delta A(z^{-1})$. This implies that the disturbances enter inside the plant, appearing at the output, as they would have been filtered through a transfer function involving the plant polynomials. Moreover such an expression for the disturbance when $\epsilon(k)$ is a sequence of pulses occurring at random times leads to piecewise disturbance that jumps at random times which can model accurately load kind of disturbances acting upon the plant. Including the factor $\Delta = 1 - z^{-1}$ leads to a controller that provides integral action. As a final note we need to mention that difficulties might arise when stochastic interpretation is adopted and the plant is unstable.

Coming back to our model, we combine equations (2.31) and (2.32) in order to obtain the CARMA model which is:

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + T(z^{-1})\epsilon(k)$$

For non-stationary disturbances such as random steps occurring at random time a more appropriate model is:

$$\nu(k) = \frac{T(z^{-1})\epsilon(k)}{\Delta} \tag{2.34}$$

where Δ is the differencing operator $\Delta = 1 - z^{-1}$ and $\nu(k)$ is now an integrated moving average sequence.

Combining equations (2.31) and (2.34) we obtain the CARIMA model:

$$A(z^{-1})y(k) = B(z^{-1})u(k-1) + \frac{T(z^{-1})\epsilon(k)}{\Delta}$$

The following interpretations of the disturbance term $\epsilon(k)$ and polynomial $T(z^{-1})$ in the CARIMA model can be given:

- With $T = 1$ and $\epsilon(k)$ a pulse sequence, the disturbance can be considered as random steps acting essentially on the plant input.
- When $\epsilon(k)$ is a sequence of independent random variables and T is considered to be known or estimated.
- $T(z^{-1})$ can be a fixed polynomial chosen to provide properties such as robustness to unmodelled dynamics [CM89a].

2.4.1 The prediction equations

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

$$A(z^{-1})\Delta y(k) = B(z^{-1})\Delta u(k-1) + T(z^{-1})\epsilon(k) \tag{2.35}$$

Consider now the Diophantine identity:

$$T(z^{-1}) = E_j(z^{-1})A(z^{-1})\Delta + z^{-j}F'_j(z^{-1}) \tag{2.36}$$

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

Multiplying (2.35) by $z^{-j}E_j(z^{-1})$ we obtain:

$$E_j A\Delta y(k+j) = E_j B\Delta u(k+j-1) + E_j T\epsilon(k+j)$$

Hence using equation (2.36):

$$Ty(k+j) = G'_j(z^{-1})\Delta u(k+j-1) + F'_j(z^{-1})y(k) + E_j(z^{-1})T\epsilon(k+j) \tag{2.37}$$

where $G'_j(z^{-1}) = E_j(z^{-1})B(z^{-1})$ and $z^{-j}y(k+j) = y(k)$.

By substituting $z = 1$ into equation (2.36) we have $F'_j(1) = T(1)$ and then we can write the following two equations:

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

and:

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

which can be proved by induction, see [CM89a].

Therefore equation (2.37) can be written as:

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

and

$$y(k+j) = G_j\Delta u(k+j-1) + \tilde{G}_j u^f(k) + F_j y^f(k) + E_j\epsilon(k+j)$$

with the final form for $y(k+j)$:

$$y(k+j) = \sum_{i=1}^{j} g_{i,j}\Delta u(k+j-i) + p_j + E_j\epsilon(k+j) \tag{2.38}$$

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

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

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

$$y^f(k) = \frac{\Delta y(k)}{T(z^{-1})}$$

$$u^f(k) = \frac{\Delta u(k)}{T(z^{-1})}$$

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:

$$TG_j + z^{-j}\tilde{G}_j = G'_j = BE_j = B(T - z^{-j}F'_j)/A\Delta$$

we have:

$$G_j = \frac{B}{A}\frac{1}{\Delta} - z^{-j}\left[\frac{F'_j}{A\Delta T} + \frac{\tilde{G}_j}{T}\right]$$

Therefore we are able to simplify equation (2.38) as follows:

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

because the first j terms of G_j are points on the plant step response model. Instead of using this theoretical approach 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 to the previous control $u(k-1)$ and the disturbance term is constant which means that $\epsilon(k+j) = 0$.

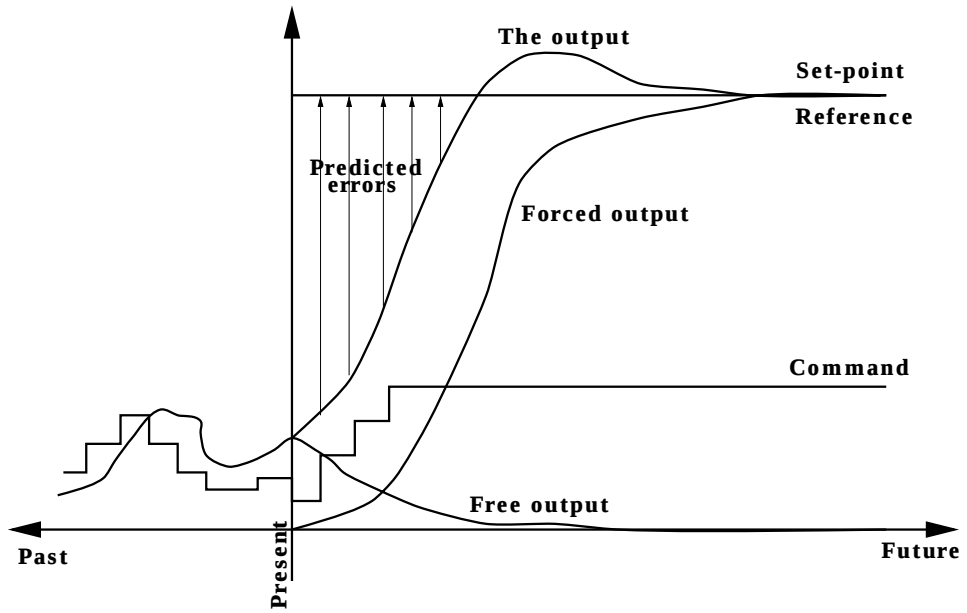


Figure 2.7: Variables involved in GPC schemes and other $MBPC$ algorithms

To have an overall view about the prediction output equation we will combine all outputs, for $j = 1, \ldots, N_2$, into a vector $y = G\tilde{u}+p+\epsilon$ where $y = [y(k+1)^T, y(k+2)^T, \ldots, y(k+N_2)^T]^T$ is the vector of future plant outputs, $\tilde{u} = [\Delta u(k)^T, \Delta u(k+1)^T, \ldots, \Delta u(k+N_u-1)^T]^T$ is the vector of potential future control increments, $p = [{p_1}^T, {p_2}^T, \ldots, {p_{N_2}}^T]^T$ is the vector of free response of the plant and $\epsilon = [\epsilon(k+1), \epsilon(k+2), \ldots, \epsilon(k+N_2)]^T$ is the vector of future disturbances.

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

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

The increments of control moves after the control horizon N_u are taken to be zero:
$\Delta u(k+j) = u(k+j) - u(k+j-1) = 0$ for $j > N_u$

2.4.2 The control algorithm

The future control sequence $u(k+j)$ is chosen by GPC algorithm to minimise the same cost-function as in the previously addressed approaches:

$$J(N_1, N_2, N_u, \lambda) = \sum_{j=N_1}^{N_2} e^2(k+j) + \lambda \sum_{j=1}^{N_u} \Delta u^2(k+j-1) \tag{2.39}$$

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 $e(k+j) = s(k+j) - \hat{y}(k+j)$ where $s(k+j)$ is the vector of the true set point. The optimisation is subject to the constraint that control increments $\Delta u(k+j) = 0$ for $j > N_u$.

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

Here another approach that uses a pseudo set-point to make smooth transitions from the current output $y(k)$ to the true set point $w(k+j)$ can also be mentioned. For a first order approximation this can be achieved by using $s^*(k+j+1) = (1-\alpha)s^*(k+j)+\alpha s(k+j)$ with $j > 0$ and $s^*(k) = y(k)$.

It is well to note that the prediction equation can be written as $\hat{y} = G\tilde{u} + p$ where $\tilde{u} = [\Delta u(k)^T, \Delta u(k+1)^T, \ldots, \Delta u(k+N_u-1)^T]^T$ is the vector of potential future control increments and $G \in \mathbb{R}^{(N_2-N_1+1)\times N_u}$ a matrix with $g_{i,j} = 0$ for $j - i > N_1$:

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

Substituting the equation $\hat{y} = G\tilde{u}+p$ in the equation (2.39) that defines the quadratic cost functional we obtain:

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

In the unconstrained case due to the least square approach we obtain the optimal control as $\tilde{u}_{opt} = (G^TG + \lambda I)^{-1}G^T(s-p)$ where $G^TG \in \Re^{N_u \cdot N_u}$.

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

The computation of GPC according to the author of paper [CM89a] 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 $\Delta u(k+j)$

2.4.3 A few concluding remarks

GPC as a control algorithm depends on integration of several ideas: the assumption of CARIMA rather than CARMA plant model, the use of long range prediction over a finite horizon (greater than the dead time of plant and at least equal with the model order), the recursion to the Diophantine equation, the consideration of weighting the control increments in the cost function and the choice of control horizon after which control increments are taken to zero.

Many of these ideas have arisen in the literature in one form or another but the authors of GPC try to consider these approaches as subsets of the GPC approach in such a way that theoretical results can be extended to this new method.

Regarding the type of model employed we have to note that: the CARIMA model was developed from the CARMA model, the disturbance is considered to be of a moving average form, the difference from the step response model based algorithms like DMC is given by the particular form required for the use of Diophantine equation when producing the prediction.

The Diophantine identity is used to predict several steps ahead the system response allowing the existence of a recursive relation that simplifies computation. The iterative computation of the solution is not only an important time saver but a key factor in the direction of adaptive applications.

The disturbances are either approximated as random steps occurring at random time or considered to be of moving average form. The classic GPC algorithms do not address the constraint handling but inequality constraints can be written in a similar form as in the input/output or the state space formulation and then augmented to the optimisation problem.

As mentioned in the interpretations developed in [Mac98, CM89a] sometimes we do not have to believe that there is a real stochastic disturbance. In fact we can regard the polynomial $C(z^{-1})$ or the observed gain as an additional tuning parameter which can be determined to give the predictive controller additional performance characteristics. As well in [Mac98] it was shown how with the state space formulation the observer parameters can be chosen with more design freedom than in the case of a traditional GPC formulation. Whether this additional freedom is actually useful is still debatable.

In the next chapter all these issues will be addressed in the framework of robust stability of the GPC, and in fact by the equivalence property, of other predictive control schemes as DMC or the state space $MBPC$.

2.5 Stable Generalised Predictive Control ($SGPC$)

One of the paths taken in proving stability of predictive control schemes involves the existence of a particular setup which will be described in this section. This is just a theoretical derivation from the GPC algorithm used in producing a good framework for the stability proofs which will follow in the next chapter. We have decided to include this presentation here due to the immediate connections with the foregoing material.

Kouvaritakis [BKC92, KR93] applies a GPC based control strategy once the system has been stabilised by means of an inner stabilising feedback loop. Such an idea is not in

fact suitable for the algorithm implementation but is useful in proving the stability of the closed loop. The reason for introducing this stabilising feedback loop is the replacement of certain loop transfer functions by finite impulse response filters (FIRs).

The stabilising feedback loop, given in Figure 2.8, has a form based on the Bezout identity written in the following form:

$$\begin{bmatrix} \tilde{Y} & \tilde{X} \\ -\tilde{N} & \tilde{M} \end{bmatrix} \begin{bmatrix} M & -X \\ N & Y \end{bmatrix} = I \tag{2.40}$$

where X, Y, $\tilde{X}$, $\tilde{Y}$, N, M, $\tilde{N}$, $\tilde{M} \in \mathbf{RH}_\infty$ (the space of all real rational stable transfer matrices) and $G(z) = N(z)M(z)^{-1} = \tilde{M}(z)^{-1}\tilde{N}(z) = \frac{1}{\Delta}G_p(z)$ with $\Delta = 1 - z^{-1}$. Here by $G_p(z)$ we denote the original plant.

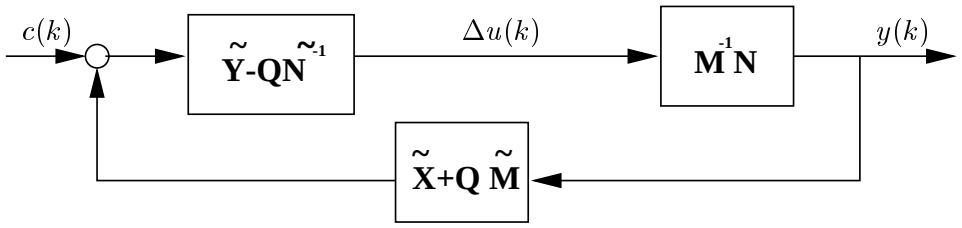


Figure 2.8: The $SGPC$ stabilising feedback loop

The parametrisation of all stabilising controllers is given by:

$$K = -(\tilde{Y} - Q\tilde{N})^{-1}(\tilde{X} + Q\tilde{M})$$

for any $Q \in \mathbf{RH}_\infty$.

The choice of Q does not affect the closed loop transfer functions from reference ($c(z)$) to input ($\Delta u(z) = M(z)c(z)$) or output ($y(z) = N(z)c(z)$):

$$\begin{aligned} \Delta u(z) &= (\tilde{Y} - Q\tilde{N})^{-1}[c(z) - (\tilde{X} + Q\tilde{M})NM^{-1}\Delta u(z)] \\ \Delta u(z) &= [(\tilde{Y} - Q\tilde{N}) + (\tilde{X} + Q\tilde{M})NM^{-1}]^{-1}c(z) \\ \Delta u(z) &= M[(\tilde{Y} - Q\tilde{N})M + (\tilde{X} + Q\tilde{M})N]^{-1}c(z) \\ \Delta u(z) &= Mc(z) \\ y(z) &= N(z)M^{-1}(z)u(z) = N(z)c(z) \end{aligned} \tag{2.41}$$

As stated in [ZDG96] the above equations hold for any choice of coprime factors $M(z)$ and $N(z)$. Therefore we can choose them as FIR operators:

$$\begin{aligned} M(z) &= T_u^0 + T_u^1 z^{-1} + \cdots + T_u^{\alpha+1} z^{-(\alpha+1)} \\ N(z) &= T_y^0 + T_y^1 z^{-1} + \cdots + T_y^{\alpha+1} z^{-(\alpha+1)} \end{aligned} \tag{2.42}$$

where $T_y^\tau \in \mathbb{R}^{p\times m}$ is a matrix with the ij^{th} element being n_{ij}^τ, elements of $N(z)$, and $T_u^\tau \in \mathbb{R}^{m\times m}$ a matrix with the ij^{th} element of the form $m_{ij}^\tau - m_{ij}^{(\tau-1)}$, elements of $M(z)$; α is the highest degree of any n_{ij}, m_{ij}; $T_{u_{ij}}^0 = m_{ij}^0$ and $T_{u_{ij}}^{m+1} = -m_+ ij^\alpha$.

We express the future system behaviour in terms of the following difference equations:

$$\begin{aligned} y(k+l) &= \sum_{\tau=0}^{\alpha} T_y^\tau c(k+l-\tau-1); l = 1 \dots N_y \\ \Delta u(k+l) &= \sum_{\tau=0}^{\alpha+1} T_u^\tau c(k+l-\tau); l = 0 \dots N_u - 1 \end{aligned}$$

The vector of future outputs $Y(k) = [y(k+1)^T, \dots, y(k+N_y)^T]^T$ and the vector of future inputs $\Delta U(k) = [\Delta u(k)^T, \dots, \Delta u(k+q-1)^T]^T$ are obtained by stacking the system inputs and outputs over the corresponding horizons.

Denoting by N_c the length of the reference horizon for $c(k+l)$ (note that in general $N_c \leq N_u \leq N_y$, see Figure 2.9) we denote the following vectors:

- the past values of the reference signal (known at time k):

$$C^{past}(k) = [c(k-1)^T, \dots, c(k-\alpha-1)^T]^T \tag{2.43}$$

- the future values of the reference signal:

$$C^{future}(k) = [c(k)^T, \dots, c(k+N_c-1)^T]^T \tag{2.44}$$

- the far future values of the reference signal (to be chosen)

$$C^{farfuture}(k) = [c(k+N_c)^T, \dots, c(k+N_y-1)^T]^T \tag{2.45}$$

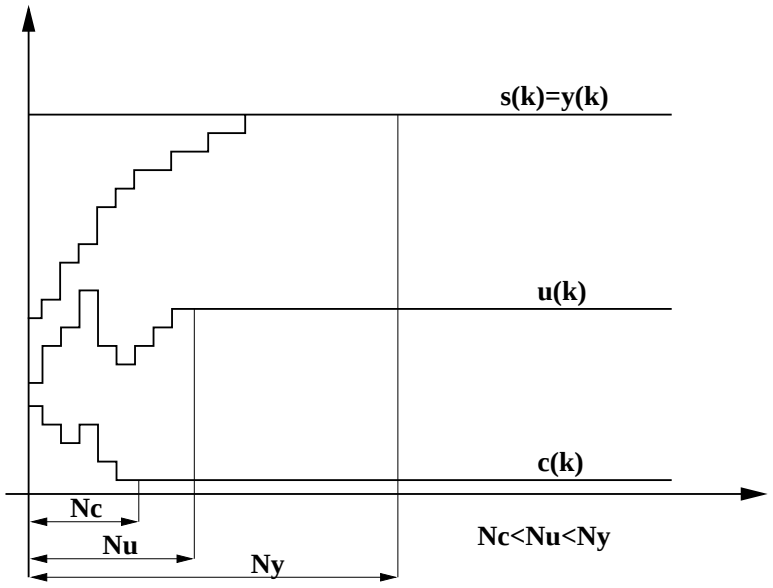


Figure 2.9: $SGPC$ horizon definition

As stated in [KR93] a convenient choice for the vector $C^{farfuture}(k)$ is to have a constant component $c(k+l) = c^\infty$ for $l = N_c \dots N_y - 1$ (i.e. $C^{farfuture}(k) = [c^{\infty T}, \dots, c^{\infty T}]^T$

where c^∞ is the solution of the system of equations $N(1)c^\infty = s^\infty$ where s^∞ is the steady state value of the output set-points which will be assumed constant while proving the stability of the $SGPC$ scheme. The c^∞ solution is unique only if $G(z)$ is a square system.

The choice c^∞ for the far future signals is an implicit terminal constraint on the output of the system requiring that the output should meet the set-point in a finite number of input moves N_u.

With the above definitions made now we are able to rewrite the predicted behaviour of the system as in [BKC92, Hei94]:

$$\begin{aligned} Y(k) &= \Gamma_y C^{future}(k) + \Xi_y C^{past}(k) + X_y C^{farfuture}(k) \\ \Delta U(k) &= \Gamma_u C^{future}(k) + \Xi_u C^{past}(k) + X_u C^{farfuture}(k) \end{aligned} \tag{2.46}$$

where $\Gamma_y \in \mathbb{R}^{N_y p \times N_c m}$, $\Xi_y \in \mathbb{R}^{N_y p \times (\alpha+1)m}$ and $X_y \in \mathbb{R}^{N_y p \times (N_y - N_c)m}$ are Hankel matrices depending on T_y^τ and the horizons N_c, N_u and N_y:

$$\Gamma_y = \begin{bmatrix} T_y^0 & 0 & \dots & \dots & \dots & 0 \\ T_y^1 & T_y^0 & 0 & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ T_y^\alpha & T_y^{\alpha-1} & \dots & T_y^0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \dots & T_y^1 & T_y^0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & \dots & \dots & 0 \end{bmatrix}$$

$$\Xi_y = \begin{bmatrix} T_y^1 & T_y^2 & \dots & T_y^\alpha & 0 \\ \dots & \dots & \dots & \dots & \dots \\ T_y^\alpha & 0 & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \dots & 0 \end{bmatrix}, \quad X_y = \begin{bmatrix} 0 & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots \\ T_y^0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & \dots & T_y^1 & T_y^0 \end{bmatrix}$$

The corresponding matrices for $\Delta u(k)$ are defined in a similar manner, see [Hei94].

With these new definitions the $SGPC$ cost function becomes:

$$J(k) = \| Y(k) - S(k) \|_Q^2 + \| \Delta U(k) \|_R^2$$

or

$$J(k) = [C^{future}(k) - C^0(k)]^T S^2 [C^{future}(k) - C^0(k)] + \gamma(k) \tag{2.47}$$

where $S^2 = \Gamma_y^T Q \Gamma_y + \Gamma_u^T R \Gamma_u$ and

$$\begin{aligned} \gamma(k) = &\| S(k) - \Xi_y C^{past}(k) X_y C^{farfuture}(k) \|_Q^2 + \\ &+ \| \Xi_u C^{past}(k) + X_u C^{farfuture}(k) \|_R^2 - \| C^0(k) \|_2^2 \end{aligned}$$

$$\begin{aligned} C^0(k) = S^{-2} \Gamma_y^T Q \left[S(k) - \Xi_y C^{past}(k) - X_y C^{farfuture}(k) \right] - \\ - S^{-2} R \Gamma_u^T \left[\Xi_u C^{past}(k) + X_u C^{farfuture}(k) \right] \end{aligned}$$

Since $\gamma(k)$ is constant then the cost function to be minimised is:

$$J(k) = \| S(C^{future}(k) - C^0(k)) \|_2^2$$

As discussed in [BKC92, KR93] N_u is in general sufficiently large to include the transient terms and $N_c < N_u \leq N_y$ such that $y(k+l)$ and $\Delta u(k+l)$ are functions of $C^{farfuture}(k) = [c^{\infty T}, \ldots, c^{\infty T}]^T$, N_c will depend on α – the truncation order of the FIR model.

2.6 Infinite Horizon Generalised Predictive Control ($IHGPC$)

The infinite horizon GPC is a natural extension of the classic GPC algorithm which bridges the gap between infinite and receding horizon methods. The main contribution of [Sco97] is the fact that for an infinite horizon, which is the central piece of the formulation, a finite number of decision variables is kept such that a quadratic program can be employed to solve the optimisation problem. This is not essentially different from the first contribution in this area which we owe to [RM93].

An infinite horizon $MBPC$ controller solves at each time step the following optimisation problem:

$$\min_{\Delta u(k)} J(k) = \min_{\Delta u(k)} \sum_{j=1}^{\infty} \| e(k+j) \|_2^2 + \| \Delta u(k+j) \|_R^2 \tag{2.48}$$

where $e(k+j) = y(k+j) - r(k+j)$, $\| \Delta u(k+j) \|_R^2 = \Delta u(k+j)^T R \Delta u(k+j)$. The optimisation is carried out subject to linear inequality constraints $A\Delta u(k) \leq b(k)$.

Based on the fact that $\Delta u(k+j) = 0$, $\forall j \geq N$ we rewrite the cost function as:

$$J(k) = \sum_{j=1}^{N} \left[\| e(k+j) \|_2^2 + \| \Delta u(k+l-1) \|_R^2 \right] + \sum_{j=1}^{\infty} \| e(k+N+j) \|_2^2 \tag{2.49}$$

Considering in the SISO case the CARIMA model:

$$y(k) = \frac{B(z^{-1})}{A(z^{-1})} u(k) + \frac{T(z^{-1})}{A(z^{-1})} \xi(k)$$

we perform a factorisation of $A(z^{-1}) = \bar{A}(z^{-1})\tilde{A}(z^{-1})$ where $\tilde{A}(z^{-1})$ of order $\tilde{n}_a$ contains all the unstable roots of $A(z^{-1})$.

In the SISO case after $(k+N)$ steps where $N = \max[N_u + n_b - 1, \tilde{n}_a, n_t]$, n_b is the order of $B(z^{-1})$, n_t is the order of $T(z^{-1})$, the tracking error of the process evolves freely according with the stable modes of the process defined by the roots of $\bar{A}(z^{-1})$, assuming that the unstable modes were stabilised first.

As stated in [RM93] the infinite sum of tracking errors can be calculated via a Lyapunov equation. We rewrite the cost function (2.49) in a matrix form:

$$J(k) = \mathcal{E}(k)^T \mathcal{Q} \mathcal{E}(k) + \Delta U(k)^T \mathcal{R} \Delta U(k) + \sum_{j=0}^{\infty} \| e(k+N+j) \|_2^2$$

where $\mathcal{Q} = \begin{bmatrix} I_{N-1} & 0 \\ 0 & 0 \end{bmatrix}$, $\mathcal{R} = diag([R, \ldots, R]^T)$, $\mathcal{E}(k) = [e(k+1)^T, \ldots, e(k+N)^T]^T$ and $\Delta U(k) = [\Delta u(k)^T, \ldots, \Delta u(k+N_u-1)^T]^T$.

Forming the vector of future errors $Z(k) = [e(k+N-\tilde{n}_a+1)^T, \ldots, e(k+N)^T]^T$ and defining the matrix $\Phi = \begin{bmatrix} 0_{\tilde{n}_a-1,1} & & I_{\tilde{n}_a-1} \\ -\bar{a}_{\tilde{n}_a} & \ldots & -\bar{a}_1 \end{bmatrix}$ it follows that $Z(k+j) = \Phi Z(k+j-1)$ which is equivalent to $Z(k+j) = \Phi^j Z(k)$.

The error $e(k+N+j)$ can be expressed as $e(k+N+j) = CZ(k+j)$ where $Z(k+j) = [e(k+N-\tilde{n}_a+j+1), \ldots, e(k+N+j)]^T$ is the shifted vector of future errors and $C = \begin{bmatrix} 0 & \ldots & 0 & 1 \end{bmatrix}$.

Hence the infinite sum of tracking errors is:

$$\sum_{j=1}^{\infty} \| e(k+N+j) \|_2^2 = \sum_{j=1}^{\infty} Z(k)^T {\Phi^j}^T C^T C \Phi^j Z(k) =$$

$$Z(k)^T \left[\sum_{j=0}^{\infty} {\Phi^j}^T C^T C \Phi^j \right] Z(k) = Z(k)^T \bar{Q} Z(k)$$

where $\bar{Q} = \sum_{j=0}^{\infty} {\Phi^j}^T C^T C \Phi^j$.

The following Lyapunov equation $\bar{Q} = C^T C + {\Phi^j}^T \bar{Q} \Phi^j$ is satisfied by $\bar{Q}$. This equation has standard solution methods available. In the explicit form the equation is:

$$\sum_{j=0}^{\infty} {\Phi^j}^T C^T C \Phi^j = C^T C + \sum_{j=0}^{\infty} {\Phi^{j+1}}^T C^T C \Phi^{j+1}$$

The final form taken by the infinite sum of tracking errors is:

$$\sum_{j=0}^{\infty} \| e(k+N+j) \|_2^2 = Z(k)^T \bar{Q} Z(k) = \mathcal{E}(k+N)^T \tilde{Q} \mathcal{E}(k+N)$$

where $\tilde{Q} = \begin{bmatrix} Q_{N-\tilde{n}_a} & 0 \\ 0 & \bar{Q} \end{bmatrix}$. This leads to the following form of the infinite horizon cost function involving a finite number of decision variables:

$$J(k) = \mathcal{E}(k)^T \hat{Q} \mathcal{E}(k) + \Delta U(k)^T \mathcal{R} \Delta U(k)$$

where $\hat{Q} = \mathcal{Q} + \tilde{Q}$. Actually the only crucial role of the input terminal equality constraint $\Delta u(k+j) = 0$, $\forall j \geq N$ is that it allows us to calculate the optimal infinite horizon law based on a finite number of decision variables.

In conclusion the infinite horizon constrained optimisation is formulated as:

$$\min_{\Delta U(k)} J(k) = \min_{\Delta U(k)} \mathcal{E}(k)^T \hat{Q} \mathcal{E}(k) + \Delta U(k)^T \mathcal{R} \Delta U(k) \tag{2.50}$$

subject to the following constraints:

1. $\tilde{Y}(k) = \tilde{S}(k)$

2. $A\Delta U(k) \leq B(k)$

where $\tilde{Y}(k) = [\tilde{y}(k+N), \ldots, \tilde{y}(k+N+\tilde{n}_a)]^T$ describes the set of predictions after N steps ahead considering that the behaviour of the system is due to only the unstable part of $A(z^{-1})$, namely $\tilde{A}(z^{-1})$ (i.e. $\tilde{y}(k) = \bar{A}(z^{-1})y(k)$). The vector $\tilde{S}(k) = [\tilde{s}(k+N), \ldots, \tilde{s}(k+N+\tilde{n}_a)]^T$ ($\tilde{S}(k) = \bar{A}(z^{-1})S(k)$) represents the $\tilde{Y}(k)$ set point. Without the constraint $\tilde{Y}(k) = \tilde{S}(k)$ the optimisation is not equivalent the one defined in equation (2.48). A quadratic programming can be employed to compute the solution, the end point constraint requirement not affecting the optimal control obtained.

Chapter 3

Stability, feasibility and robustness

$MBPC$ solves on-line a constrained optimisation which 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 results in closed loop behaviour which is quite different from the one predicted by the optimal solution which assumed uninterrupted open-loop implementation.

As a result one of the main issues raised by the $MBPC$ technique is the one of stability. This is because predictive control is a feedback control strategy which brings up the risk of an unstable closed loop. Therefore using finite horizons the controller can put the system in such a state that it will eventually be impossible to stabilise it especially when constraints will be imposed by actuator limitations. In the case of the predictive control method the stability proofs, especially *a priori* guarantees, were required mainly by the academic community, people working in industry were fairly happy with this issue as long as all the $MBPC$ schemes implemented were working well.

There is a discussion in the $MBPC$ control community regarding the infinite and finite horizon approaches. Each one of them has its own advantages and drawbacks depending on the various applications tackled.

The initial approaches to $MBPC$ schemes considered finite horizons because at that time no methods were known of providing efficient on-line computation of the infinite horizon control law. Together with the finite horizon approach came the benefit of including linear constraints upon various variables involved in the controller cost function. In the finite horizon case using small prediction and control horizons leads to a reduced computational load, which is important since the requirements for the solution speed might be critical.

The unconstrained $MBPC$ control law, which is the result of a least square approximation, lack the ability to handle constraints. In order to account for them, a repeated optimisation is required which has as a solution the optimal feedback law together with the active set of constraints. As a result, the controller is nonlinear even for a linear plant and a linear set of inequality constraints.

In the absence of active constraints, it is known how to enforce stability. Essentially, either the prediction horizon N_2 must be made large enough, or 'terminal' equality constraints, which bind at time $k+N_2$, must be added to the problem formulation. It has been shown that, from the point of view of stability enforcement, terminal equality constraints

can be exchanged for an infinite prediction horizon [RM93]. For instance this is one of the issues upon which we concentrate in the following sections of this chapter.

Later approaches managed to cast infinite horizon $MBPC$ in a formulation involving a finite number of free variables. This resulted in the possibility of computing the solution using conventional quadratic programming.

Several methods are known of ensuring stability even in the presence of constraints, but under the assumption that the problem posed always remains feasible. This is a very strong and almost not verifiable assumption, and some current research is aimed at removing it.

Various assumptions were made in order to provide robust stability and feasibility for various cases. Most stability proofs are based on proving the monotonicity of the cost function $J(k)$ with increasing k, and hence using the cost function as a Lyapunov function. There have also been some attempts at exploiting the piecewise-linear nature of the controller to prove stability. Whereas obtaining stability is not difficult in practice for predictive control schemes, there are not yet standard procedures for obtaining specified stability margins.

In the following sections of this chapter we survey some of the essential techniques employed to guarantee stability in both unconstrained and constrained cases. The feasibility issues are addressed in the section corresponding to the constrained $MBPC$. The last section deals with two main methods used to provide robustness to $MBPC$ closed loops.

3.1 Achieving stability in the unconstrained case

Before presenting several approaches used in achieving the $MBPC$ stability in the unconstrained case we would like to point out that the inclusion of these methods in our work was generated by the necessity to use some of these results when designing the controller for the aerospace applications presented in the next chapters.

It is hard to say that we had a favourite method to check stability but it is true that we have used the information provided by these approaches in order to build up our tuning guide-lines and to check several of the assumptions made.

For instance the theoretical study made by Kouvaritakis et al. [BKC92] regarding an inclusion of a stabilising feed-back loop initiated somehow the investigations regarding the flight-management solution presented in Chapter 6.

On the other hand, even the technique of using a terminal constraint to ensure stability was not used in either the practical developments or in the Development Space — the in house software used to produce the simulation results shown in this work — we have included its presentation for two main reasons. One is concerned with the fact that historically speaking it is one of the initial approaches used to prove stability of the receding horizons scheme. The second reason is the parallelism drawn between other techniques and this one (*e.g* the use of an exponential weighting of the cost function).

A lot of insight in connecting the Difference Riccati Equation (DRE) and predictive control is provided in the work conducted by Bitmead et al. [BGW90]. For us this was not only a way to approach in systematic manner the issue of proving stability but as well represented a neat solution relying on tools with which we were familiar from robust

control theory.

In the class of infinite horizon approaches we situate the already classic contribution of Rawlings et al. [RM93] which in fact opened a new path in the theory and implementation of predictive control schemes. The results included in their early work together with the ones that can be found in Scokaert et al. [Sco97] offered a clear view upon how by manipulating the plant input for a finite number of steps can lead to a stable controller, the solution of which is computed as usual via a quadratic programme. This approach allowed the inclusion of linear constraints upon the variables of the cost function.

The approach to solving the stability of the predictive control scheme in an original and modern manner is the formulation in Linear Matrix Inequalities (LMI). The only problem with this approach is again connected with the real-time implementation because solving on-line LMI problems is still a challenge for present computers. Otherwise the problem is formulated in a clear manner and moreover there is a strong connection between the formulation, which allows a neat proof of stability, and the implementation which actually solves the corresponding LMI problem.

Note as well that in the presentation of the following techniques we tried to offer our view but at the same time to stick to the original formulation given by the authors.

3.1.1 The inclusion of a stabilising feedback loop

One of the possible approaches to stability for the $MBPC$ controller is the one developed in the work of Kouvaritakis et al. [BKC92, KR93].

The stability in the unconstrained case is proved under the assumption of a feasible optimisation problem that will have sufficiently large horizons to include all transient terms. We consider that for a given reference c^{∞}, which is a step, the predicted output will reach it in N_y steps. We consider that the reference is a step because the stability is not affected by the choice of the set-point.

In this section we state for the Stable Generalised Predictive Control ($SGPC$) the theorem which gives the sufficient stability condition. The proof of the theorem is based on the monotonicity of the cost function and is given with the aim of clarifying this way of tackling stability issues. For all the variables involved the reader should refer to Section 2.5:

Theorem 3.1.1 *[Hei94] The nominal SGPC controller is stable for all values of the reference horizon N_c such that $N_c < \min(N_y - \alpha, N_u - \alpha)$.*

Proof The feasibility of the $SGPC$ optimisation ensures that $y(k+l)$ will reach c^{∞} within N_y steps as long as $N_y > N_c + \alpha$, more precisely: $y(k+l) = c^{\infty}$ for $l \geq N_y$ (Recall that α is the order of the FIR operators (2.42)). In addition $\Delta u(k)$ will reach zero after N_u steps given that $N_u > N_c + \alpha$.

Considering the cost function 2.47 the optimal value of $C^{future}(k)$ (see 2.44 for its definition) at time k ($C^{future^*}(k)$) represents a feasible solution of the optimisation problem. The same $C^{future^*}(k)$ at time $k+1$ will be a feasible solution but at this time instant is not necessarily optimal.

Implementing this solution at time $k+1$ will result in a value of the cost function 2.47 not smaller than the optimal one:

$$J(k+1)^* \leq J(k+1) \leq J(k)^* \tag{3.1}$$

where $J(k+1)^*$ is the optimal value of the cost function at time $k=1$ and $J(k+1)$ is the value of the cost function obtained by plugging the optimal solution $C^{future^*}(k)$ obtained at time k.

The fact that $J(1)$ is finite, together with relation (3.1), prove the monotonically decreasing nature of the cost function. Then following a Lyapunov type of argument we can conclude that the closed loop system with the $SGPC$ controller is asymptotically stable.

□

As stated in Section 2.5, the inclusion of the feedback loop is used just to prove the stability of the $SGPC$ scheme. The existence of the inner loop, the design of which is based on the nominal model, cannot form the basis of a practical implementation because the output predictions will be affected by the model mismatches and disturbances. In other words the predictive control internal model defined by the plant and the inner stabilising feedback loop "hides" the real plant model from the outer $MBPC$. Therefore, the outer predictive controller, running at the same sampling rate as the inner one, will not be able to account for disturbances and model mismatches.

As a result the key to the implementation, given in [KR93], is the idea of rewriting the closed loop terms using relations that describe the system future behaviour using the FIR transfer functions, when the feedback inner loop is closed around the plant. The implementable feedback configuration is based on a pre-filter which produces a reference signal $c(k)$ as a function of the past inputs, outputs and future references. This represents actually the feedback mechanism required to respond to mismatch and disturbances.

3.1.2 The inclusion of terminal constraints

The essence of predictive control is to determine the control profile that optimises an open-loop performance objective in such a way that stability guarantees for the closed loop are provided. One method of overcoming the major drawback of the predictive control technique, which is the poor stability guarantees of the classic scheme, is to augment the cost function with terminal constraints that will bring the states to a steady state value at the end of the prediction horizon.

The first attempts to include terminal constraints were made by Kwon and Pearson [KP75, KP78, KP89] and then by Clarke et al. [CS91]. The idea mentioned in [KP78] is to bring all the closed loop eigenvalues of the system within the unit circle, which can be achieved by minimising the following finite horizon cost function:

$$J(k) = \sum_{j=1}^{N} \parallel y(k+j) - s(k+j) \parallel^2_{Q(k+j)} + \sum_{j=0}^{N} \parallel \Delta u(k+j) \parallel^2_{R(k+l)} \tag{3.2}$$

subject to the equality constraints for $l = 1 \dots m$:

$$\begin{aligned} y(k+N+l) &= s(k+l) \\ \Delta u(k+N+l) &= 0 \end{aligned}$$

where $y(k+l)$, $\Delta u(k+l)$ and $s(k+l)$ are the system output, input and future reference, respectively. The stability proof of this method assumes constant reference.

In the case of stable systems employing the end point constraint it can be shown [Sco97] that the cost function is finite. In the infinite horizon case a violation of this condition clearly results in an unbounded cost since violation of $y(k+N+l) = s(k+l)$ implies a nonzero steady state offset.

For unstable systems the end-point constraint specifies that the unstable modes of the system have to be at equilibrium values, in the linear case, or brought to a steady state, in the nonlinear case, at the end of the control horizon. Hence, the value of the unstable part of the free response is brought at the time $k+N_u$ to the value that will make $y(k+N)$ to converge to $s(k+N)$. If this condition is not met then the cost function, for the infinite horizon case, will be unbounded and instability might result.

The following theorem given by Clarke et al. [CS91] characterises the stability of the closed loop giving sufficient conditions in terms of the horizon lengths. We denote by n the degree of the internal plant model and by m the number of steps beyond the prediction horizon N for which the equality constraints are imposed.

Theorem 3.1.2 *[CS91] If the system matrix augmented to account for integral action 2.3 is nonsingular, the weights in the cost function are time invariant and the conditions $N \geq n+2, m = n+1$ is satisfied then the closed loop is asymptotically stable.*

The proof of this theorem is based on the observation that the output constraint can be transfered into a terminal state constraint, based on the observability assumption for the pair (A, C), and then the result given in Kwon et. al. [KP89] can be used. Recall equation (2.3) for the state matrix of the system augmented for integral action.

The state space solution of the above problem, suggested in [KP78], transforms the end-point constraint into an infinite end-point weighting matrix. If there is no weighting upon the state, the non-singularity of the system matrix is required.

3.1.3 The use of fake algebraic Ricatti equations (FARE)

Looking backwards in the history of Linear Quadratic (LQ) control, it is interesting to remember that only ten years ago, due to the work conducted by R.R. Bitmead [BGW90, NBG97] stability proofs were provided for LQ strategies based on finite horizon optimisation. As presented in Section 3.1.2 Kwon et al. [KP89, KP78] made some initial successful attempts that provided a special type of a receding horizon controller with zero-state terminal constraints.

The main difference between the receding horizon technique and the infinite horizon controller is the association of the optimisation solution with a Difference Riccati Equation (DRE) rather than a Algebraic Riccati Equation (ARE).

The inability to provide a neat stability result for the scheme involving finite horizons made several researchers suggest abandoning this strategy in favour of infinite horizon LQ optimisation, despite the wide application of finite horizons $MBPC$ in so many practical control problems. On the other hand such issues pushed several researchers to intensify their search. For them the requirement for adaptation together with the difficulty of solving the ARE on-line represented a serious argument in favour of the finite horizon approach.

Now that we know how the infinite horizon problem can be transformed in a problem that keeps a finite number of decision variables without losing the ability to include constraints, we are not going to argue that the necessity to include constraints was the reason to go for the finite horizon strategy.

One of the first paths taken to solve the receding horizon control problem was by means of a Fake Algebraic Riccati Equation (FARE), see [BGW90, NBG97]. This Section aims to provide an overall view of this approach, enabling also the understanding of the material contained in the Section 3.1.4. The following results represent a review of the main theorems regarding stability of receding horizon schemes, and is based on the work developed by [BGW90, NBG97].

In this case the problem statement is not much different from the state space approach presented in Section 2.2 but refers to the receding horizon case for which the state is included in the cost function. Considering an LTI system $x(k+1) = Ax(k) + Bu(k)$ with (A, B) stabilizable we associate the finite horizon cost function:

$$J(x(k), N) = x(k+N)^T P_0 x(k+N) + \sum_{j=0}^{N-1} x(k+j)^T Q x(k+j) + u(k+j)^T R u(k+j) \tag{3.3}$$

where P_0, $Q \geq 0$ (semi-positive definite) and $R > 0$ (positive definite). In the case of an LTI system used as a plant model, the unconstrained predictive control scheme is equivalent to a constant control law. The main drawback of the scheme is that for a given value of the horizons the closed loop might not be stable. It is known from the Linear Optimal control theory that for (A,Q) detectable the solution of the Discrete Riccati Equation (DRE) converges to the stabilising solution of the Algebraic Riccati Equation (ARE). As well from the LQ theory it is known that the solution of the problem:

$$\min_{u(k+j), j \geq 0} J(x(k), N) \tag{3.4}$$

is solved by the state feedback control law $u(k+j) = K(N+1-j)x(k+j)$ where $K(j) = -[B^T P(j) B + R]^{-1} B^T P(j) A$, $j = 0 \ldots N-1$ and $P(k)$ is the solution of the following DRE:

$$P(k+1) = A^T P(k) A + Q - A^T P(k) B [B^T P(k) B + R]^{-1} B^T P(k) A \tag{3.5}$$

with the initial condition $P(0) = P_0$.

The infinite horizon case ($N \to \infty$) is characterised by the stabilising feedback K_∞ which is the gain corresponding to $P(\infty)$. For the additional assumption (A, Q) detectable

or weaker (A, Q) with no unobservable modes on the unit circle we can state that the unique semi-positive definite solution of ARE:

$$P = A^T PA + Q - A^T PB[B^T PB + R]^{-1} B^T PA \tag{3.6}$$

is defined by the feedback gain $K_\infty = -[B^T P_\infty B + R]^{-1} B^T P_\infty A$ which asymptotically stabilises the closed loop (i.e. $A + BK_\infty$ is stable), see [BGW90].

One way to stabilise the closed loop with the $MBPC$ controller is to exploit the stability result reported above. The tool used in this approach is FARE which is obtained by defining $Q(N-1) = Q + P(N-1) - P(N)$ which represent an adjustment to the state weighting matrix and by rewriting the equation (3.5) as:

$$P(N-1) = A^T P(N-1)A + Q(N-1) - A^T P(N-1)B[B^T P(N-1)B + R]^{-1} B^T P(N-1)A \tag{3.7}$$

Hence comparing equation (3.7) with equation (3.6) we can regard the solution $P(N-1)$ as the solution used to produce the associated gain $K(N-1)$.

Extrapolating the result stated in the ARE case [BGW90] then for (A, B) stabilizable, $Q(N-1) \geq 0$ and the pair $(A, Q(N-1))$ detectable then the closed loop matrix $A + BK(N-1)$ is asymptotically stable. The initial assumption (A, B) stabilizable is necessary in order to perform control actions on the system, case in which is interesting to derive conditions for detectability of the pair (A,Q(N-1)).

Having a proper choice for P_0 and using the theorem [NBG97] which states that if $P(\cdot)$ is a solution of (3.5) and if $P(k+1) \leq P(k)$ for some k then $P(k+j+1) \leq P(k+j)$, $\forall j > 0$ then we can guarantee $Q(N-1) \geq 0$. In fact if P_0 is such that $P(1) \leq P_0$ then $P(k+1) \leq P(k)$, $\forall k \geq 0$ and as a result $Q(N-1) = Q + P(N-1) - P(N) \geq 0$, $\forall N > 0$.

Basically the choice of P_0 and N affects the detectability of the pair $(A, Q(N-1))$. For instance a lemma in [NBG97] proves that if $P(1) \leq P(0)$ then $P(N) \leq P(N-1)$ and $Q(N-1) = Q + P(N-1) - P(N)$. In such a case $(A, Q(N-1))$ is detectable $\forall N > 0$. This means that a good choice for P_0 guarantees non increasing monotonicity. In particular cases such as an open loop stable plant or when $Q = C^T C$ and (A, C) observable then the pair $(A, Q(N-1))$ is detectable. We conclude all the above statements with the following theorem:

Theorem 3.1.3 *[BGW90] Let $P(\cdot)$ be the solution of DRE (3.5) and assume that*

1. *$P(1) \leq P(0)$*
2. *(A, Q) detectable*

Then $A + BK(N-1)$ is asymptotically stable.

Remark 3.1.4 *As can be observed, if we have an initial solution $P(0) = P_0$ which guarantee the non increasing monotonicity of the cost function ($P(0) \geq P(1)$), then we can apply the above theorem for finite $N > 0$.*

3.1.4 The exponential weighting of the cost function

This section represents almost an independent approach to the problem of stability. The main idea, emphasised in the work of Yoon et al. [YC93], is to use exponentially increasing weightings in the cost function, with the aim of enhancing the stability margin, by placing more weighting on future tracking errors and control increments. All this was done in the context of ensuring the monotonicity of the cost function. One of the main advantages of exponential weighting, apart from ensuring stability, is the relatively easy way of implementing it in the existing $MBPC$ algorithms.

For the first time, Anderson et al. [AM71] introduced the idea of obtaining a prescribed degree of stability by using exponential weighting. This was done in continuous time infinite horizon optimal control. The approach developed by Yoon et al. [YC93] parallels the idea of Anderson et al. The motivation for using a monotonic weighting for the $MBPC$ cost function is the necessity of a monotonic solution of the associated Riccati equation as a sufficient stability condition. Moreover, according to Kwon et al. [KP89], increasing weighting improves the dynamic responses of the closed loop system.

The present formulation is concerned with SISO case. We concentrate on the following form of the cost function which, as we know already from Chapter 2, is minimised under the assumption $\Delta u(k+j) = 0$ for $j \geq N_u$:

$$J(k) = \sum_{j=1}^{N} Q(j)[y(k+j) - s(k+j)]^2 + R(j)\Delta u(k+j)^2 \tag{3.8}$$

The weighting $Q(j)$ and $R(j)$ have the following exponentially increasing forms $Q(j) = Q\lambda^{-2j}$ and $R(j) = R\lambda^{-2j}$ where $\lambda \in (0, 1]$ and $Q \geq 0$, $R > 0$. Without loss of generality we can assume $Q = 1$.

Recalling the GPC formulation from Section 2.4, when minimising the cost function (3.8) in the unconstrained case we can write for the SISO case [YC95] the optimal solution as $\Delta u(k) = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} \tilde{G}_g \tilde{G}_g^T M_g (F(k) - S(k))$ where:

$$\begin{aligned}
\tilde{G}_g &= [G_g^T M_g G_g + \Delta_g]^{-1} \\
M_g &= diag(\begin{bmatrix} \lambda^{-2} & \lambda^{-4} & \dots & \lambda^{-2N_u} \end{bmatrix}^T) \\
\Delta_g &= R \cdot diag(\begin{bmatrix} 1 & \lambda^{-2} & \lambda^{-4} & \dots & \lambda^{-2(N_u-1)} \end{bmatrix}^T) \\
F(k) &= \begin{bmatrix} f(k+1) & f(k+2) & \dots & f(k+N_2) \end{bmatrix}^T \\
S(k) &= \begin{bmatrix} s(k+1) & s(k+2) & \dots & s(k+N_2) \end{bmatrix}^T \\
G_g &= \begin{bmatrix} g_1 & 0 & \dots & 0 \\ g_2 & g_1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ g_{N_u} & g_{N_u-1} & \dots & g_1 \\ \dots & \dots & \dots & \dots \\ g_N & g_{N-1} & \dots & g_{N-N_u+1} \end{bmatrix}
\end{aligned}$$

Here $F(k)$ contains the free response of the system for $j \geq 0$, $S(k)$ represent the future reference vector and G_g contains the step response coefficients of the system. In such a

way the burden of producing a solution on-line can be reduced due to a large amount of computation carried out off-line.

The main argument used in providing stability properties for the predictive control scheme with exponential weighting relies on the monotonicity of the DRE equation associated with the unconstrained case, as shown in Section 3.1.3.

Using a minimal state space representation of the plant model we are able to rewrite the cost function (3.8) in the following form:

$$\begin{aligned} J(k) &= \lambda^{-2N_u} x(k+N_u)^T P_0 x(k+N_u) + \\ &+ \sum_{j=0}^{N_u-1} \lambda^{-2j} [x(k+j)^T C^T C x(k+j) + \Delta u(k+j)^T R \Delta u(k+j)] \end{aligned} \tag{3.9}$$

where we have used $P_0 = \sum_{j=0}^{N-N_u} {A^j}^T C^T C A^j$ to represent the system's free response.

Denoting $x_\lambda(k) = \lambda^{-k} x(k)$ and $\Delta u_\lambda(k) = \lambda^{-k} \Delta u_\lambda(k)$ we can rewrite the cost function (3.9) as:

$$\begin{aligned} J(k) &= x_\lambda(k+N_u)^T P_0 x_\lambda(k+N_u) + \\ &+ \sum_{j=0}^{N_u-1} [x_\lambda(k+j)^T C^T C x_\lambda(k+j) + \Delta u_\lambda(k+j)^T R \Delta u(k+j)] \end{aligned}$$

for which the corresponding model is $x_\lambda(k+1) = A_\lambda x_\lambda(k) + B_\lambda \Delta_\lambda u(k+j)$ with $A_\lambda = \lambda^{-1} A$ and $B_\lambda = \lambda^{-1} B$.

The predictive control problem defined by the cost function (3.9) has the following solution:

$$\Delta u_\lambda(k) = -K(N_u - 1) x_\lambda(k)$$

where

$$K(l) = \frac{B_\lambda^T P(l) A_\lambda}{R + B_\lambda^T P(l) B_\lambda}$$

The matrix $P(l)$ is produced via successive iterations using the following DRE:

$$P(l+1) = C^T C + [A_\lambda - B_\lambda K(l)]^T P(l) [A_\lambda - B_\lambda K(l)] + K(l)^T R K(l)$$

Returning to the original form written in λ we have:

$$\Delta u(k) = -K(N_u - 1) x(k)$$

where

$$K(l) = \frac{B^T P(l) A}{\lambda^2 R + B^T P(l) B}$$

and

$$P(l+1) = C^T C + \lambda^{-2}[A - BK(l)]^T P(l)[A - BK(l)] + RK(l)^T K(l)$$

Using the following theorem [YC95] we can state the monotonicity of the DRE solution. (We have decided to include the proof due to the insight given into the problem and the connections with the material presented in the previous section.)

Theorem 3.1.5 *If* $P(l) \geq \lambda^2 P(l+1)$ *for some* $l \in \mathbb{N}$ *and* $\lambda \in (0,1]$ *then* $P(l+q) \geq \lambda^2 P(l+q+1)\ \forall q \in \mathbb{N}$

Proof

Rewriting the DRE including the expression for the feedback matrix we have:

$$P(l+1) = C^T C + A_\lambda^T P(l) A_\lambda - \frac{A_\lambda^T P(l) B_\lambda B_\lambda^T P(l) A_\lambda}{R + B_\lambda^T P(l) B_\lambda}$$

We define $P'(l+2)$ based on the following DRE:

$$\lambda^2 P'(l+2) = C^T C + A_\lambda^T \lambda^2 P(l+1) A_\lambda - \frac{A_\lambda^T \lambda^2 P(l+1) B_\lambda B_\lambda^T \lambda^2 P(l+1) A_\lambda}{R + B_\lambda^T \lambda^2 P(l+1) B_\lambda} \qquad (3.10)$$

Knowing that $P(l) \geq \lambda^2 P(l+1)$ and comparing the previous two equations then we can state that $P(l+1) \geq \lambda^2 P'(l+2)$.

Now rewriting the first equation for $P(l+2)$ and comparing them:

$$\begin{aligned} P'(l+2) &= \lambda^{-2} C^T C + A_\lambda^T P(l+1) A_\lambda - \frac{A_\lambda^T P(l+1) B_\lambda B_\lambda^T P(l+1) A_\lambda}{\lambda^{-2} R + B_\lambda^T P(l+1) B_\lambda} \\ &\geq C^T C + A_\lambda^T P(l+1) A_\lambda - \frac{A_\lambda^T P(l+1) B_\lambda B_\lambda^T P(l+1) A_\lambda}{R + B_\lambda^T P(l+1) B_\lambda} = P(l+2) \end{aligned}$$

we can conclude that $P'(l+2) \geq P(l+2)$. Hence $P(l+1) \geq \lambda^2 P(l+2)$ and therefore by induction $P(l+q) \geq \lambda^2 P(l+q+1)\ \forall q \geq 0$.

□

Usually the condition $P(l) \geq P(l+1)$ is quite restrictive. The inclusion of λ^2 makes it achievable even when the system will be unstable for other horizons using the conventional cost function.

We conclude this section with a theorem [YC95] that provides the stability condition for the receding horizon scheme with exponential weighting:

Theorem 3.1.6 *Considering the system:*

$$\begin{aligned} x(k+1) &= Ax(k) + Bx(k) \\ y(k) &= Cx(k) \end{aligned}$$

controlled using the state feedback

$$\Delta u(k) = -K(N_u - 1)x(k)$$

where

$$K(l) = \frac{B^T P(l) A}{\lambda^2 R + B^T P(l) B}$$

and $P(l)$ obtained using the following DRE equation:

$$P(l+1) = C^T C + \lambda^{-2}[A - BK(l)]^T P(l)[A - BK(l)] + RK(l)^T K(l)$$

then the closed loop system is asymptotically stable if $P(l) \geq \lambda^2 P(l+1)$ for some $l \leq N_u - 1$.

Proof

The proof is based on finding a Lyapunov function $V(k)$ having the form $V(k) = x(k)^T P(N_u - 1)x(k)$.

$$\begin{aligned}
\Delta V(k) &= x(k+1)^T P(N_u - 1)x(k+1) - x(k)^T P(N_u - 1)x(k) = \\
&= x(k)^T \left[[A - BK(N_u - 1)]^T P(N_u - 1)[A - BK(N_u - 1)] - P(N_u - 1)\right] x(k) = \\
&= -x(k)^T Q x(k)
\end{aligned}$$

where Q is defined as:

$$\begin{aligned}
Q &= P(N_u - 1) - [A - BK(N_u - 1)]^T P(N_u - 1)[A - BK(N_u - 1)] = \\
&= P(N_u - 1) - \lambda^2 P(N_u) + \lambda^2 [C^T C + RK(N_u - 1)^T K(N_u - 1)]
\end{aligned}$$

Using the Theorem 3.1.5 results $P(N_u - 1) \geq \lambda^2 P(N_u)$ and hence

$$Q \geq \lambda^2 [C^T C + RK(N_u - 1)^T K(N_u - 1)] \geq 0$$

Due to the initial assumption (C, A) observable then we can claim that $x(K)^T [C^T C + RK(N_u - 1)^T K(N_u - 1)]x(k)$ is not identically zero along any trajectory of the closed loop system. Hence using the Lyapunov theorem it follows that x(k) converges to zero as $l \to \infty$

□

It is well known that $MBPC$ with constant weighting is also stabilising if the prediction and control horizons are sufficiently long. The problem in such a case, compared with the much shorter horizons required by the exponential weighting approach, is the computational load when the solution is computed on-line. In addition to the $MBPC$ with constant weighting for the exponential weighting approach a stability margin can be obtained.

The main influence of this strategy is in terms of the closed loop dynamic response which is speeded up and the overshoot reduced. Having the guarantee that the moduli of the closed loop poles are within a circle of a radius determined by λ we have a direct adjustment of the settling time. Hence once horizons are fixed, the radius λ becomes the only tuning parameter. Of course forcing the poles within a tight region will magnify the control effort required to change the dynamics of the original system.

Compared with the work done by Kwon et al. [KP75] which refers to the inclusion of a terminal state constraint, this approach removed that requirement completely.

3.1.5 Making the cost function horizons infinite

The basic idea for the infinite horizon assumption is that, for a problem which assumes no noise or disturbance and measurable states, it is possible to guarantee a stable solution of the system controller. A description of the infinite horizon problem involving a finite number of decision variables is given in Section 2.6. There is actually another approach which deals with stability issues by achieving monotonicity for the infinite cost functional written in the following form:

$$J(k) = \sum_{j=1}^{\infty} \left[(y(k+j) - s(k+j))^2 + R\Delta u(k+j) \right] = \sum_{j=1}^{\infty} \left[e(k+j)^2 + R\Delta u(k+j) \right] \tag{3.11}$$

subject to the input constraint $\Delta u(k+j) = 0$ for $j \geq L+1$.

A sufficient condition for solving the above minimisation problem, providing that the pair (A, B) is stabilizable, is that $L \geq r$ where r is the number of the unstable modes of the system state matrix (A). In fact we can state, see [NBG97] and Theorem 3.1.3, that assuming the pairs (A, B) stabilizable and (A, Q) detectable and having $P_0 = \Sigma_L$ (defined below) with $L \geq r$ then $A + BK(N-1)$ is stabilising $\forall N > 0$.

As a result two different cases are encountered:

1. A stable — this means $r = 0$ leading to a choice of $L = 0$ and hence $\Sigma_L = \sum_{j=0}^{\infty} A(j)^T Q A(j)$ which can be efficiently computed [Sco97] as a solution of a Lyapunov equation $\Sigma_L = A^T \Sigma_L A + Q$.

2. A unstable — a condition for which the controllers suggested by the teams Kwon and Pearson [KP89] and Muske and Rawlings [RM93] are equivalent.

Hence stability properties can be derived relatively easily due to the infinite horizon employed in the cost function. In [Sco97] a proof of stability is outlined based, on the argument of monotonicity of the cost function. Therefore, considering in the unconstrained case $\Delta U(k)$ as the optimal control vector at time k which gives the optimal prediction errors $\mathcal{E}(k)$ and the value $J(k)^*$ for the optimum cost function, we are able to express the cost function at time $k+1$:

$$J(k+1)^+ = J(k)^* - \| e(k+1)^* \|_2^2 - \| \Delta u(k) \|_R^2$$

The optimal cost at time $k+1$ has the property:

$$\begin{aligned} J(k+1)^* &\leq J(k+1)^+ \\ J(k+1)^* &\leq J(k)^* - \| e(k+1)^* \|_2^2 - \| \Delta u(k) \|_R^2 \end{aligned}$$

which means that $J(k)^*$ is non increasing and bounded by 0.

Hence $\| e(k+1)^* \|_2^2 - \| \Delta u(k) \|_R^2 \leq J(k)^* - J(k+1)^*$ which means that as $[J(k)^* - J(k+1)^*] \to 0$ the term $[\| e(k+1)^* \|_2^2 + \| \Delta u(k) \|_R^2] \to 0$. For positive values of R the stability follows. In [Sco97] details for the case $R = 0$ are given.

3.1.6 A formulation in linear matrix inequalities (LMIs)

A final step in our presentation of existing approaches used to address stability issues in the unconstrained case is the contribution of Kothare et al. [KBM96]. From the perspective of a practical implementation this method, which involves solving several LMIs on line, is very demanding from the computational point of view, but the current research in interior point algorithms might provide improvements in the quality and speed of the solution provided [BGFB94]. For a basic knowledge of the linear matrix inequalities (LMIs) concept we encourage the reader to consult the work of Boyd et al. [BGFB94].

In the nominal case the stability proof is based on the derivation of an upper bound for the cost function followed by a Lyapunov argument. In the nominal case the LQR regulator is obtained for which the feedback matrix F, where $u(k+j) = Fx(k+j), \forall j \geq 0$, is constant. The main difference from the LQ regulator is that in the presence of uncertainty for the unconstrained case the feedback matrix $F(x(k))$ exhibits a strong dependence on the current system state. Therefore using the predictive control approach which computes the feedback matrix at each sampling time the performance is substantially improved, compared with static feedback control [KBM96]. An important property of this approach is that if feasibility is achieved at time k, then it is maintained over the whole infinite prediction horizon.

In the case of the LMI formulation the infinite horizon objective has the following form:

$$J(k) = \sum_{j=0}^{\infty} \parallel x(k+j) \parallel_Q^2 + \parallel u(k+j) \parallel_R^2 \tag{3.12}$$

where $\parallel \alpha \parallel_Q = \alpha^T Q \alpha$. Driving the state to zero (a zero set-point) is a common procedure when proving stability of a controller based on a quadratic cost function involving the model states.

A quadratic function $V(x) = \parallel x(k) \parallel_P^2$ with $P > 0$ is derived as an upper bound on the performance index. With $V(0) = 0$ we suppose that $V(x)$ satisfies the following inequality at sampling time k:

$$V(x(k+j+1)) - V(x(k+j)) \leq -x(k+j)^T Q x(k+j) + u(k+j)^T R u(k+j) \tag{3.13}$$

for $x(k+j)$, $u(k+j)$ $\forall j \geq 0$. In order to have a finite $J(k)$ then $x(\infty) = 0$ is a condition and hence $V(x(\infty)) = 0$.

We sum (3.13) for $j = 0, \ldots, \infty$ and as a result we have $-V(x(k)) \leq J(k)$ which represents an upper bound on the performance index. Effectively the LMI approach to $MBPC$ designs a state feedback law F at each time step k which minimises the upper bound $V(x(k))$. Conditions on the existence of a $P > 0$ and the corresponding state feedback matrix F are given in the following theorem:

Theorem 3.1.7 *[KBM96] Let $x(k)$ be the state of the nominal system. Then the state feedback matrix that minimises the upper bound $V(x(k))$ on the performance objective $J(k)$ (3.12) is given by $F = YZ^{-1}$ where $Z > 0$ and Y are obtained from the solution, if it exists, of the following LMI problem:*

$$\min_{\gamma, Z, Y} \gamma$$

subject to

$$\begin{bmatrix} 1 & x(k)^T \\ x(k) & Z \end{bmatrix} \geq 0$$

and

$$\begin{bmatrix} Z & ZA^T + Y^T B^T & ZQ^{\frac{1}{2}} & Y^T R^{\frac{1}{2}} \\ AZ + BY & Z & 0 & 0 \\ Q^{\frac{1}{2}} Z & 0 & \gamma I & 0 \\ R^{\frac{1}{2}} Y & 0 & 0 & \gamma I \end{bmatrix} \geq 0$$

where $Z = \gamma P^{-1}$.

The matrices Z, F, Y are computed at each time k. The parameter γ is introduced in order to influence the speed of the closed loop response. In the unconstrained case several other LMIs can be augmented to the above optimisation which will force the closed loop poles into a predetermined stability region (*e.g* a sector in the LHP).

As a potential problem of this formulation we can observe the fact that the LMI optimisation always minimises an upper bound of the cost function, the quantity which is really of interest. Despite the fact that the authors of the strategy [KBM96] claim that the bounds are not too conservative we express our concern for some cases.

In the sections dedicated to the constrained case and robust performance we are going to address how the present LMI problem can be augmented with one that gives robust stability and two others in order to enable the algorithm to handle input and output constraints.

3.2 Ensuring stability and feasibility in the constrained case

One of the major reasons for using predictive control is its ability to handle constraints. This ability of course is conditioned by the feasibility of the optimisation problem over the future horizon. Therefore in the constrained case we cannot separate this two aspects: stability and feasibility.

First of all the problem has to be feasible at the time when the first on-line optimisation was solved. In other words the initial plant state has to be such that a control movement can be found in order to stabilise the plant in the presence of constraints. In normal operation (*i.e.* when none of the constraints are active) the above condition translates into a condition of plant stabilisability. Assuming such a condition being satisfied we can think about a second issue which refers the consistency between the set-point that the controlled system is requested to track and the system input and output constraints enforced.

In this section three different approaches used to tackle the issues of stability and feasibility in the constrained case are given.

The reason of including the first one is the potential that this algorithm has in dealing with the feasibility of the optimisation in the constrained case — problem that represents the bottleneck of the vast majority of $MBPC$ schemes.

The second approach was included because, relying on the new achievements in the field, we strongly believe that in the future we can think about implementations that will involve a constraint optimisation based on an infinite horizon cost function.

In connection with the constrained infinite horizon formulations, last but not least, we have decided to include in our presentation the predictive control scheme based on an LMI formulation which accounts for input and output constraints. This approach translates the usual problem of feasibility of the quadratic programme employed in obtaining the solution into a feasibility problem of the LMI problem solved at each time step.

3.2.1 A modified optimisation algorithm for the $SGPC$ scheme

In 1995 Rossiter et al. [RKG95b] augmented the $SGPC$ so as to include a penalty on constraint violation as well as on the predicted output error. Instead of producing the solution of the optimisation problem using the well known QP algorithm the authors have suggested a modification of the Lawson's weighted least square algorithm in a manner that considers as well the requirement for $\Delta u(k)$ to lie inside the feasible region as the one to minimise a quadratic cost defined by the $SGPC$ strategy. The authors proved that this modified algorithm together with the $SGPC$ strategy lead to a Constrained Stable Generalised Predictive Control ($CSGPC$) strategy which provides a systematic way of handling infeasibility.

As mentioned in Section 3.1.1, the arguments of the monotonicity of the cost function carry through when trying to prove the stability of the $CSGPC$ scheme, but in this case the proof requires the feasibility of the finite horizon optimisation problem. This fact cannot always be guaranteed. The requirement, similar to that in $SGPC$ schemes, is that the predicted output reaches its target value within a finite number of steps, within the prediction horizon N_y. The target value c^∞ is given by the set-point, which is assumed constant during the prediction horizon and the dynamics of the system: $N(1)c^\infty = s^\infty$ (where the plant $G(z)$ is factorised as $G(z) = N(z)M(z)^{-1}$).

During infeasibility it becomes necessary to relax this requirement, by allowing c^∞ to vary for as long as is necessary to recover from the infeasibility. This relaxation is obtained by:

- replacing the predicted part of the cost by a penalty on the deviation from c^∞.
- adding an additional constraint which defines how the relaxed c^∞ will move monotonically towards its target value.

The foregoing changes from the $CSGPC$ algorithm generated the Modified $CSGPC$ ($MCSGPC$) which has guaranteed stability and asymptotic tracking whereas $CSGPC$ represent a predictive control scheme that preserves optimality and stability in face of constraints.

The $CSGPC$ algorithm

In most of their papers [RKG95b, RKG95a] the authors consider time invariant constraints upon absolute input constraints ($|u(k+l) - u_{centre}| \leq u_{radius}$), input rate constraints ($|\Delta u(k+l)| \leq r$) and output constraints ($|y(k+l) - y_{centre}| \leq y_{radius}$). Considering

$C^{past}(k)$, $C^{future}(k)$ and $C^{farfuture}(k)$ defined in (2.43, 2.44, 2.45) and normalising them to 1 then:

$$|m(k)^T C^{future}(k) - v(k)| < 1, \quad k = 1, \ldots, N_y$$

where $m(k)$ and $v(k)$ are defined as follows:

- for absolute input constraints:

$$\begin{aligned} m(k)^T &= \frac{E(k)^T \Gamma_u}{u_{radius}}; \\ v(k) &= -\frac{E(k)^T + (\Xi_u C^{past}(k) + X_u C^{farfuture}(k)) + u(k-1) - u_{centre}}{u_{radius}} \end{aligned}$$

  where $E(k)$ denotes the sum of the first k column vectors of the identity I matrix.

- for input rates:

$$\begin{aligned} m(k)^T &= \frac{e(k)^T \Gamma_u}{\alpha}; \\ v(k) &= \frac{e(k)^T (\Xi_u C^{past}(k) - X_u C^{farfuture}(k))}{\alpha} \end{aligned}$$

- output absolute constraints

$$\begin{aligned} m(k)^T &= \frac{e(k)^T \Gamma_u}{y_{radius}}; \\ v(k) &= -\frac{e(k)^T (\Xi_y C^{past}(k) + X_y C^{farfuture}(k)) - y_{centre}}{y_{radius}} \end{aligned}$$

  where $e(k)$ is the k^{th} column vector of the identity matrix of conformal dimensions.

The above constraints can be stacked together in the following linear matrix inequality: $\| MC^{future}(k) - v(k) \|_\infty \leq 1$ with $v(k)$ varying irrespective of whether the constraints are time independent, $M \in \mathbb{R}^{3N_y \times N_u}$, $v(k) \in \mathbb{R}^{3N_y}$.

The above linear inequality implies that $C^{future}(k)$ must lie within a pair of hyper-planes each perpendicular to the corresponding vector from the M matrix. The convex region defined by the pairs of hyper-planes is referred as the feasible region. This region is non empty if the inequality holds for some future $C^{future}(k)$.

In contrast with this the expression "short term" infeasibility is used when the region is empty for a particular value of the N_c horizon. Here Lawson's weighted least square algorithm [LH74, KR93] comes into play. This algorithm deals with the situation when inequality constraints cannot be satisfied and when a C^{future} is chosen so as to minimise the worst case constraint violation. This is considered an important feature of the algorithm.

In the case of such short term infeasibility the strategy of choosing $C^{future}(k)$ is dominated by the minimisation of $J(k) = \| S(C^{future}(k) - C^0(k)) \|_2$. In order to account

for the infeasibility we augment the original $SGPC$ cost function forming the following optimisation index:

$$J(k+1) = \left\| \begin{array}{l} w(k+1)^{\frac{1}{2}} \left[S(C^{future}(k+1) - C^0(k+1)) \right] \\ W(k+1)^{\frac{1}{2}} \left[M(C^{future}(k+1) - v(k+1)) \right] \end{array} \right\|_2$$

The weights involved in the augmented cost function are updated in the following way:

$$\begin{aligned} w(k+1) &= \frac{w(k)}{\|W(k)(MC^{future}(k) - v(k))\|_1} \\ W_{jj}(k+1) &= \frac{W_{jj}(k)|(MC^{future}(k) - v(k))_j|}{\|W(k)(MC^{future}(k) - v(k))\|_1} \\ \|W(k)(MC^{future}(k) - v(k))\|_1 &= \sum_{j=1}^{m} W_{jj}(k)|(MC^{future}(k) - v(k))_j| \end{aligned}$$

The weights can be initiated as $w(0) = 1$ and $W(0) = I_m$ where m is the number of rows in M. When k increases the algorithm converges to $C^{future}(k)$ which minimises the cost function under constraints [RK93].

The following theorem gives a characterisation of the $CSGPC$ scheme:

Theorem 3.2.1 *[RK93] If the feasible region is nonempty for all k then the CSGPC scheme will cause the output y to follow asymptotically any set-point change to a new constant value.*

The feasibility condition is essential to prove the stability of $CSGPC$ schemes. Under "short term" feasibility the $C^{future}(k)$ converges to the constrained optimum compared with the case when infeasibility is encountered when it converges to the solution that minimises the maximum constraint violation. This fact enables the Mixed Weight Least Squares (MWLS) scheme to provide the feasibility of the optimisation problem [Hei94].

The weights' behaviour is such that these will converge to 0 if the corresponding elements of the constraint part of the cost function are less than one and some finite non-zero value otherwise [RK93]. During the optimisation some of the diagonal elements of W(k) do not tend to zero [Hei94]. These are the ones corresponding to the active set of constraints at that moment, providing an effective way of identifying them. In such a way the optimisation time is shortened if a suboptimal solution is accepted.

The $MCSGPC$ algorithm

As a natural extension to the $CSGPC$ another modified algorithm $MCSGPC$ was developed. A slack variable end-point constraint is introduced in order to deal with the "short term" infeasibility. The cost function is identical to the one provided by $CSGPC$ at all times except when this kind of infeasibility is encountered. The only difference from the $CSGPC$ is that c^∞ is a degree of freedom, and an extra constraint has been introduced to ensure that the current optimal value of c^∞ is closer to the one desired and obtained by solving the equation $N(1)c^\infty = s^\infty$ denoted by c^∞_{old}.

Therefore we redefine $C^\star(k) = \left[c^{future}(k)^T \quad c^{\infty T}\right]^T$ and we write:

$$\begin{aligned} Y(k) &= \Gamma_{m_y} C^\star + \Xi_y C^{past}(k) \\ \Delta U(k) &= \Gamma_{m_u} C^\star + \Xi_u C^{past}(k) \end{aligned}$$

Then the problem $\|MC(k) - v(k)\|_\infty < 1$ becomes $\|M^\star C^\star(k) - v^\star(k)\| < 1$ where:

$$\begin{aligned} M^\star(k) &= \begin{bmatrix} M & m \\ 0 & \frac{1}{c^\infty_{old} - \frac{s^\infty}{N(1)}} \end{bmatrix} \\ v^\star(k) &= \begin{bmatrix} v(k) + mc^\infty \\ \frac{1}{\frac{N(1)c^\infty_{old}}{s^\infty} - 1} \end{bmatrix} \end{aligned}$$

As a consequence the cost function is:
$J(k) = \|S^\star(k) - C^0(k)\|^2$ where $S^\star = (Q\Gamma^T_{m_y}\Gamma_{m_y} + R\Gamma^T_{m_u}\Gamma_{m_u})^{-2}$.

The main assumption made in the $MCSGPC$ case is that the set-point has been chosen sensibly otherwise the algorithm will cause c^∞ to settle at the boundary of the constraint interval. The strategy guarantees the "short term" feasibility by assuming feasibility at the start of the optimisation. Note that no other assumptions are required.

3.2.2 A constrained infinite horizon formulation

In the setting of an infinite horizon $MBPC$ formulation with constraints a separation between stable and unstable plants is required. The two cases are fairly different from the perspective of the stability proof mechanism and the feasibility of the solution.

Constraints that normally impose a pair of bounds on future values of control inputs, increments and outputs can be expressed for linear models as pairs of inequalities which are linear in the future control increments. Therefore constraints of the type $Du(k+j) \leq d$, $\forall j \geq 0$ corresponding for inputs and $Hx(k) \leq h$, $\forall j \geq 1$ corresponding for outputs can be defined and then stacked together in a similar way using the procedure shown in Section 2.2.3. Before stacking them we can employ a simple method to convert the input and output constraints into a finite set by assuming as usual $u(k+j) = 0$ for $j \geq N_u$.

For stable systems the input constraints are always feasible independent of the pair (A, B). In the same case things are a bit different for state and/or output constraints due to the possible infeasibility that may occur. Of course there are methods to convert them into a feasible set by removing them for small j. In [RM93] it was shown that there exists such a j_1 providing feasibility. The argument is based on the fact that for a large enough j the states become arbitrarily small and the constraints are satisfied. It was possible to determine the value of j_1 for the nominal model and no disturbances acting on the plant. In fact Rawlings et al. [RM93] show that in order to guarantee satisfaction of the constraints on an infinite horizon it is enough to have them satisfied on a horizon $[j_1, j_2]$ where both j_1 and j_2 are finite.

In the unstable plant case neither input nor state and/or output constraints feasibility can be checked independently of the pair (A, B), the initial state and the horizon N_u. Rawlings et al. in [RM93] define what an admissible x_0 means by denoting a set X_{N_u} of x_0 for which there exists, a command $u(k+j) \in U$ for $j \in [0, N_u]$ and $u(k+j) = 0$ for

$j \geq N_u$ such that $\lim_{k\to\infty} x(k) = 0$. The claim made was that the system can be stabilised by controllers based on the infinite horizon cost function if and only if $x_0 \in X_{N_u}$. In such a case the state constraints can be handled as in the previous case of the stable plant.

Theorem 3.2.2 *[RM93] For a stabilizable pair (A, B) with A unstable having r unstable modes $x(k+j) = 0$ is an asymptotically stable solution of the closed loop closed with an $MBPC$ controller if $N_u \geq r$ and the QP program is feasible for every $x_0 \in X_{N_u}$.*

It is obvious that if the $x_0 \notin X_{N_u}$ then X_{N_u} has to be enlarged by increasing N_u. A more difficult case is when $x_0 \notin X_\infty$ which means that the plant cannot be stabilised matching the present constraint. The enlargement of N_u is just one alternative that has to be applied with care because of the increase in the computational load. Another choice that can work is to make j_1 big, which is equivalent to the removal of the constraints for the initial part of the optimisation. For instance, in the case of a nonminimum phase system constraints can be violated during the initial behaviour. Increasing j_1 will make the constraints feasible therefore giving the controller the opportunity to assess the situation.

3.2.3 Augmenting the linear matrix inequalities (LMIs) with constraints

In the constrained case the LMIs stated in Theorem 3.1.7 have to be augmented with several others which account for the input and output constraints. The basic idea of such an inclusion can be found in [BGFB94] but Kothare in his work [KBM96] presented for the first time the extension to discrete time systems.

The input constraints are imposed over the whole horizon of the future manipulated variable in two specific ways for the infinite horizon $MBPC$ problems. The average bounds such as $\| u(k+j) \|_2^2 \leq u_{max}$, $\forall j \geq 0$ are translated into LMI inequalities in the following form:

$$\max_{j\geq 0} \| u(k+j) \|_2^2 = \max_{j\geq 0} \| YZ^{-1}x(k+j) \|_2^2 \, Z^{-\frac{1}{2}} \leq \max_{z\in\mathcal{C}} \| YZ^{-1}z \|_2^2 = \lambda_{max}(Z^{-\frac{1}{2}}Y^TYZ^{-\frac{1}{2}}) \tag{3.14}$$

where $\mathcal{C}$ represent the state invariant ellipsoid, see [KBM96].

Using Schur complements we translate the above inequality into an LMI:

$$\begin{bmatrix} u_{max}^2 I & Y \\ Y^T & Z \end{bmatrix} \geq 0$$

The peak bounds on each component of $u(k+j)$ at sampling time j can be formulated as $|u_l(k+j)| \leq u_{lmax}$, $\forall j \geq 0$, $l = 1, \ldots, N_u$. The translation of the above inequality in an LMI follows the next few steps which involves the Cauchy-Schwartz inequality and the Schur complement lemma:

$$\max_{j\geq 0} |u_l(k+j)|^2 = \max_{j\geq 0} |(YZ^{-1}x(k+j))_l|^2 \leq \max_{z\in\mathcal{C}} |(YZ^{-1}z)_l|^2 \leq \| (YZ^{-\frac{1}{2}})_l \|_2^2 = (YZ^{-1}Y^T)_{ll}$$

Therefore the existence of a matrix X such that:

$$\begin{bmatrix} X & Y \\ Y^T & Z \end{bmatrix} \geq 0, \tag{3.15}$$

with $X_{ll} \leq u_{l_{max}}^2$, $l = 1, \ldots, N_u$, guarantees that $|u_l(k+j)| \leq u_{l_{max}}$.

Along the same lines as in [KBM96] it can be proved that the output constraint $\| y(k+j) \|_2 \leq y_{max}$, $\forall j \geq 1$ can be represented in the following LMI form:

$$\begin{bmatrix} Z & (AZ+BY)^T C^T \\ C(AZ+BY) & y_{max}^2 I \end{bmatrix} \geq 0 \tag{3.16}$$

This is achieved as follows:

$$\begin{aligned} \max_{j \geq 1} \| y(k+j) \|_2 &= \max \| C(A+BF)x(k+j) \|_2 \\ \max \| C(A+BF)x(k+j) \|_2 &\leq \max_{z \in \mathcal{C}} \| C(A+BF)z \|_2 \\ \max_{z \in \mathcal{C}} \| C(A+BF)z \|_2 &= \bar{\sigma}[C(A+BF)Z^{\frac{1}{2}}] \end{aligned}$$

Thus $\| y(k+j) \|_2 \leq y_{max}$ if $\bar{\sigma}[C(A+BF))Z^{\frac{1}{2}}] \leq y_{max}$ or $Z^{\frac{1}{2}}[A+BF]^T C^T C[A+BF]Z^{\frac{1}{2}} \leq y_{max}^2 I$ which via Schur complements is equivalent to the LMI (3.16).

As a result, the theorem proved by Kothare et al. [KBM96], which refers to the constrained case, states that for an LMI problem as in Theorem 3.1.7 augmented with the corresponding LMIs for the input (3.14, 3.15) and output constraints having a feasible solution Y and Z the $MBPC$ infinite horizon constrained solution is $F = YZ^{-1}$. The feasibility problem is addressed in [KBM96] by a lemma stating that any feasible solution F at time k is also feasible at all times $k+j$, $\forall j \geq 1$. In this way a feasible asymptotically stable $MBPC$ state feedback control law is obtained by solving an LMI problem.

3.3 Designing for robust performance in $MBPC$ control

Robust performance of $MBPC$ schemes is a key issue. By definition a robust control design ensures a particular performance specification in closed loop to the controlled plant which is assumed to lie in a set that can be characterised in a quantitative manner.

The next natural step after solving the nominal stability/performance problem using the approaches described in Section 3.1 and Section 3.2 is to address the robust stability/performance. In industrial application this issue is tackled at the moment via extensive closed loop simulations prior to implementation, an expensive method due to the large number of possible combinations of plant dynamics and active constraints.

In order to achieve robust stability/performance in $MBPC$ schemes two main paths were taken by various authors:

1. The conventional way – which involves specific tuning of the controller either via the observer included in the output feedback schemes or by formulating the problem as min-max optimisation that can be solved in a classic manner or as an LMI.

2. The multi-model adaptive approach – which assumes an existence of a plant model obtained either via identification or via on-line linearisation of a high fidelity model associated with the plant [HM98a, HM98c, Mac97]

In our work we went for the second one due to its possibilities to encompass the problem of reconfigurable control. Nevertheless the first one, as seen in the literature, can provide good means for achieving robust performance.

Apart from the observer tuning method any other alternative is extremely demanding from the computational point of view. All of them worked fairly well in simulation but, when coming down to a real-time implementation, apart from the observer tuning method, all other approaches were somehow difficult to apply.

3.3.1 The conventional approach

Robustness achieved via tuning the observer

Many researchers regarded the design of the observer as a way to achieve robustness in $MBPC$ schemes. Lee et al. [LM91b] have initial tried to provide robustness to the $MBPC$ scheme via additional tuning parameters (*i.e.* parameters of the filter gain and disturbance time constraints) which together with the conventional ones (*i.e.* cost function horizons and weights) can provide a large amount of possible tuning combinations. Unfortunately these parameters offer unnecessary flexibility in the controller, complicating tuning since many of them have overlapping effects.

As a result the authors of [LY94] suggest the tuning for robustness via the parameters of the observer employed to estimate the plant states. This idea is based on the fact that some of these parameters have more direct connections to the closed loop robust stability and performance than others which cannot be tuned in a straightforward manner. For these it is best to set them at a certain pre-specified value due to their indirect effect.

In [LY94] guidelines for tuning these parameters, which have independent and well understood effects on the closed loop were given. The unconstrained controller is designed first for the best nominal performance and then detuned to achieve robust performance. The closed loop stability is checked *a posteriori* once all the tuning parameters have been determined, based on the results given in Section 3.1. Lee et al. provided such guide lines by looking at systems with a first order disturbance model. By writing the closed loop transfer function from output disturbance and reference to the output, the dependence on the observer parameters [LY94] is revealed. Once nominal stability and performance for the observer and the $MBPC$ controller were achieved, in SISO or MIMO cases, rules are provided to shape the frequency response of the sensitivity function and its complement via the observer parameters.

Another perspective upon robustness is given by the work of Bitmead et al. [BGW90, NBG97] where Loop Transfer Recovery (LTR) is advocated in order to design the $MBPC$ controller.

As explained in [Mac89] a state feedback controller, obtained as a solution of an LQ problem, is known to have excellent robustness properties in continuous time and, providing the use of a very small sampling time, in discrete time as well. But, these properties hold only in the case of full state measurements and not when an observer is used to estimate the plant state. The solution for this problem – called Loop Transfer Recovery (LTR) – is to design an observer after a state feedback was designed using LQ. The idea, first time mentioned by [Mac89], is such that by using artificial covariance matrices in the optimisation cost function, which has as a solution the observer gain matrix, the overall

loop will converge to the good gain and phase margin properties of the state feedback alone.

The other alternative of detuning the controller via adjusting the control cost function increment weighting matrix R is disregarded due to its less predictable effect with process dynamics and the impossibility of retaining the same robustness margin in the face of active output state constraints [Zaf90, RM93, Sco97].

Kouvaritakis et al. in their work [KR93] give an alternative to the problem of achieving robustness for $MBPC$ closed loops. Recalling the presentations of $SGPC$ made in Section 2.5 we know already that the whole family of controllers can be expressed in terms of the Youla parameter, $Q(z)$, a stable transfer matrix:

$$K_{SGPC}(z) = \Delta(z)[\tilde{Y}(z) - Q(z)\tilde{N}(z)]^{-1}[\tilde{X}(z) + Q(z)\tilde{M}(z)] \tag{3.17}$$

This parameter represents a degree of freedom which can be exploited in order to maximise robustness properties of the loop. Using standard arguments like in [ZDG96] for an additive unstructured plant uncertainty $\Delta G(z)$ of the plant $G(z) = \tilde{M}(z)^{-1}\tilde{N}(z) = N(z)M(z)^{-1}$, which has the property $\|\Delta G(z)\|_\infty \leq W(z)$, the necessary and sufficient robust stability conditions is:

$$\|W(z)K_{SGPC}(z)[I + G(z)K_{SGPC}(z)]^{-1}\|_\infty < 1 \tag{3.18}$$

Hence the optimal $K_{SGPC}(z)$ controller for robust stability, see Figure 3.1, is the one that minimises the infinity norm of the expression (3.18). Rewriting (3.18) in terms of $Q(z)$ we have:

$$\|W(z)[\tilde{Y}(z)M(z) + \tilde{X}(z)N(z)]^{-1}[\tilde{X}(z) + Q(z)\tilde{M}(z)]\|_\infty < 1$$

which is equivalent to the short form $\|T_1(z) - T_2(z)Q(z)T_3(z)\|_\infty < 1$ for which well known techniques developed for model matching are available. Solving the above optimisation for $Q(z)$ will determine the $MBPC$ controller out of the family defined in equation (3.17) which will give the required robust stability properties.

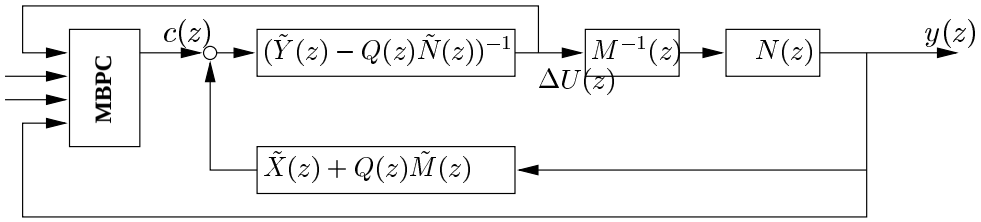


Figure 3.1: The block diagram showing the $SGPC$ $Q(z)$ parametrisation

Along the same lines the work of Yoon et al. [YC95] concentrates on the achievement of robustness in the case of GPC schemes via a clever design of the observer. As a result his approach concentrates on the design of the $T(z^{-1})$ polynomial of the CARIMA model (2.31). Generally the polynomial $T(z^{-1})$ is regarded as a fixed observer which is well known to play an important role in robust realisation of GPC predictive controllers [CM89a].

A systematic design strategy for the GPC observer $T(z^{-1})$ has not been produced yet. Even if it is commonly assumed that filtering results in good robust properties of the closed loop, this is contradicted by several simple examples [YC95]. Therefore designs based on such ideas may fail to achieve robustness. Here we will try to emphasise, based on the work done by [YC95], the role of $T(z^{-1})$ as a key element in designing for robustness within the GPC formulation, presented in Section 2.4.

The GPC control structure referred to in this section is shown in Figure 3.2

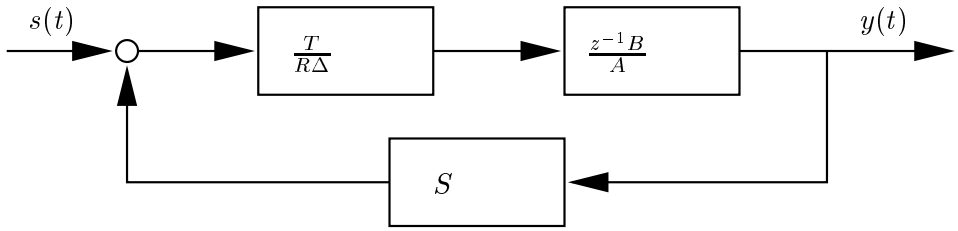


Figure 3.2: The structure of GPC schemes

for which we have:

$$R(z^{-1})\Delta u(t) = T(z^{-1})s(t) - S(z^{-1})y(t) \tag{3.19}$$

The theorem summarising Yoon et al.'s work addresses the robustness issue in a small gain theorem fashion. The condition stated in the theorem is then used in the $T(z^{-1})$ filter design.

Theorem 3.3.1 *[YC95] Considering a controller (*e.g *the GPC controller) given by* $R(z^{-1})\Delta u(t) = T(z^{-1})w(t) - S(z^{-1})y(t)$ *and designed based on the plant model* $A(z^{-1})y(t) = B(z^{-1})u(t-1)$ *then the characteristic polynomial of the closed loop given by*

$$A_0(z^{-1})R\Delta + z^{-1}B_0(z^{-1})S(z^{-1}) = P_c(z^{-1})$$

remains stable if $A(z^{-1})$ *and* $A_0(z^{-1})$ *have the same number of unstable roots and if* $\forall\omega \in [0,\pi]$*:*

$$\left| \frac{B(e^{-j\omega})}{A(e^{-j\omega})} - \frac{B_0(e^{-j\omega})}{A_0(e^{-j\omega})} \right| \leq \left| \frac{P_c(e^{-j\omega})}{A(e^{-j\omega})} \right| \left| \frac{T(e^{-j\omega})}{S(e^{-j\omega})} \right| \tag{3.20}$$

where the pair $(A_0(z^{-1}), B_0(z^{-1}))$ *represents the real plant and* $A(z^{-1})R(z^{-1})\Delta + z^{-1}B(z^{-1})S(z^{-1}) = P_c(z^{-1})T(z^{-1})$ *is the closed loop characteristic equation in the case of perfect modelling.*

The design objective is to ensure a prescribed high frequency roll-off for the transfer function $\frac{T(z^{-1})}{S(z^{-1})}$ in order to achieve a larger value of the upper bound, at high frequencies, in (3.20). Methods to address this issue were given by Clarke at al. [RC91, YC95].

From the perspective of the relation between $T(z^{-1})$ and the $Q(z^{-1})$ parametrisation used in the $SGPC$ case we can compare them in terms of block diagrams. By observing that $Q(z^{-1}) = \frac{M(z^{-1})}{T(z^{-1})}$ where $T(z^{-1})$ and $M(z^{-1})$ satisfy the following Diophantine

equations:

$$\begin{aligned} R(z^{-1}) &= R'(z^{-1})T(z^{-1}) - z^{-1}B(z^{-1})M(z^{-1}) \\ S(z^{-1}) &= S'(z^{-1})T(z^{-1}) - \Delta A(z^{-1})M(z^{-1}) \end{aligned}$$

we can claim that the $T(z^{-1})$ based tuning, pictured in Figure 3.3, is similar with the scheme suggested by [BKC92] illustrated in Figure 3.1.

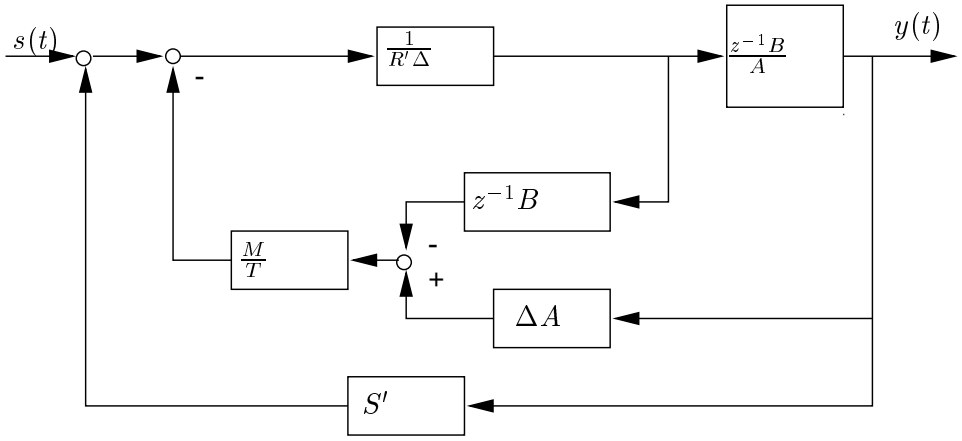


Figure 3.3: The block diagram showing the equivalence between $T(z^{-1})$ and the $Q(z)$ parametrisation when tuning for robustness

The min-max approach

In [Lee96, All94] the "min-max" algorithm for solving the robust stability problem is developed. The main idea of the algorithm consists of a minimisation of the maximum cost function for all possible plants in the uncertainty ball. The drawback of this method is that the procedure accounts only for the optimisation of the worst case performance which sometimes results in conservative control actions. Moreover, the complexity of the algorithm is quite involved since it requires a multilevel optimisation.

In [All94] the "min-max" algorithm is considered for FIR models. The minimisation is carried out with respect to the time invariant impulse response, as it ranges over a pre-specified polytope of plants, of the tracking error. According to Allwright et al. [All94] the ∞-norm min-max problem has a solution independent of whether the uncertainty in the impulse response is time variant or invariant.

The minimisation problem based on FIR models is formulated in the following way:

$$\begin{aligned} &\min_{u \in \mathbb{R}^{m \times N_u}} \quad \max_{\delta \in \Delta} \|y(\delta, U, V) - s\|_\infty \\ &\underline{U} \le U \le \bar{U} \\ &\underline{Y} \le y(\delta, U, V) \le \bar{Y} \end{aligned}$$

where $U = [u(k)^T, \ldots, u(k+N_u)^T]^T$ and $V = [u(k-1)^T, \ldots, u(k-N)^T]^T$. Here the input ($\underline{U}$, $\bar{U}$) and output ($\underline{Y}$, $\bar{Y}$) constraints are time invariant but the uncertainty ($\delta \in \Delta$) is time variant.

This optimisation involves only the maximum deviation of the worst case tracking error, which is therefore affected by even one output point far from the set-point. As an alternative we can use, as in [All94], a minimisation problem that makes the problem still solvable via an LP program paying less attention to outlier points.

The polytopic approach using LMIs

In Kothare et al. [KBM96], two models for uncertain systems were used to address the robust constrained predictive control using linear matrix inequalities (LMIs).

The polytopic approach to robust constrained $MBPC$ makes use of the linear time varying system (LTV) expressed in discrete time:

$$\begin{aligned} x(k+1) &= A(k)X(k) + B(k)u(k) \\ y(k) &= Cx(k) \end{aligned}$$

where $u \in \mathbb{R}^m$, $x \in \mathbb{R}^n$, $y \in \mathbb{R}^p$ and $\Omega = C_o\{[A_1, B_1], \ldots, [A_L, B_L]\}$ is a polytope, where C_0 denotes a convex hull (*i.e.* if $[A, B] \in \Omega$, then $[A, B] = \sum_{i=1}^{L} \lambda_i [A_i, B_i]$). Methods to develop such models can be found in [SA90, Woo95].

In the polytopic case the robust performance optimisation problem is formulated as:

$$\begin{array}{ccc} & \min & J_\infty(k) \\ u(k+j) & j = 0, \ldots, N_u - 1 & \\ [A_i, B_i] \in \Omega & i \geq 0 & \end{array}$$

where the performance index is defined as:

$$J_\infty(k) = \sum_{j=0}^{\infty} x(k+j)^T Q x(k+j) + \sum_{j=0}^{N_u - 1} u(k+j)^T R u(k+j)$$

The min-max optimisation problem is performed over the set Ω and corresponds to choosing the time varying plant $[A_i, B_i] \in \Omega$, $\forall i \geq 0$ in such a way that if it is used as a model for prediction then it will lead to the largest (worst case) value of $J_\infty(k)$, the value of which minimised with respect to the control moves $u(k+j)$, $j = 0, \ldots, N_u - 1$.

As stated in [KBM96] this min-max problem,convex for finite N_u, was not addressed in the literature up to now in this set-up. The only exception is the work of Allwright et al. [All94] that simplifies the problem using FIR models. In [KBM96] as described in Section 3.1.6 the problem is addressed by finding an upper bound on the performance objective.

The robust performance constrained minimisation written in terms of LMIs is described in the following theorem:

Theorem 3.3.2 *[KBM96] Let $x(k)$ be the state of the uncertain system. Then the state feedback matrix that minimises the upper bound $V(x(k))$ on the performance objective $J_\infty(k)$ is given by $F = YZ^{-1}$ where $Z > 0$ and Y are obtained from the solution, is exists, of the following LMI problem:*

$$\min_{\gamma, Z, Y} \gamma$$

subject to

$$\begin{bmatrix} 1 & x(k)^T \\ x(k) & Z \end{bmatrix} \geq 0$$

and

$$\begin{bmatrix} Z & ZA_i^T + Y^T B_i^T & ZQ^{\frac{1}{2}} & Y^T R^{\frac{1}{2}} \\ A_i Z + B_i Y & Z & 0 & 0 \\ Q^{\frac{1}{2}} Z & 0 & \gamma I & 0 \\ R^{\frac{1}{2}} Y & 0 & 0 & \gamma I \end{bmatrix} \geq 0$$

where $Z = \gamma P^{-1}$ *and* $i = 1, \ldots, L$. *The minimisation problem is subject to either*

$$\begin{bmatrix} u_{max}^2 I & Y \\ Y^T & Z \end{bmatrix} \geq 0$$

or

$$\begin{bmatrix} X & Y \\ Y^T & Z \end{bmatrix} \geq 0,$$

depending on the input constraint to be imposed and

$$\begin{bmatrix} Z & (A_i Z + B_i Y)^T C^T \\ C(A_i Z + B_i Y) & y_{max}^2 I \end{bmatrix} \geq 0$$

reflecting the output constraint to be enforced.

The proof of this theorem can be found in the Appendix of [KBM96].

Other approaches

Other ideas regarding a solution for the robust design of $MBPC$ for stable plants involve the suppression of the input movement on the basis of a Lyapunov function, as in [VGN95].

In [Bad97] an extension of the infinite horizon algorithm suggested by Rawlings et al. [RM93] is given. The model uncertainty is parametrised by a list of possible plants. Constraints are added to prevent the cost function from increasing for the true plant. The authors claim the robust stabilisation in the presence of constraints for the whole range of the tuning parameter. In fact, the cost function minimisation is augmented in the unconstrained case by an inequality constraint which requires the cost function values, computed for each member of the plant family living in the uncertainty ball, to remain constant or decrease at each time step relative to the cost values computed using the current measured state and the input restriction. The restriction for the input is obtained by shifting from the optimal input vector computed at previous time.

This approach, similar to [Zaf90] can be thought as a generalised state contraction, defined on an infinite horizon including the contributions from the inputs and states without any additional tuning factors. The feasibility of the optimisation is ensured due to the restricted output. The theorems stated in [Bad97] give the necessary conditions for the

robust stability of the above strategy in the unconstrained case. For the constrained case the necessary conditions stated in the case of stable systems are required. This result was extended by Badgwell along the lines from [RM93, Sco97, RK93] by relaxing the hard constraints using a slack variable. As a result the $MBPC$ controller finds both the control increment and the value of the slack variable, an idea first introduced by [Zaf90].

3.3.2 The multi-model adaptive approach

As discussed in the initial presentation of the state space $MBPC$ approach there are means of involving a different model at each time step in the optimisation. We are aware that systems are highly nonlinear with dynamics that change from one set-point to another. Although there are methods of dealing with such type of systems they require a good knowledge of the plant. The usual approach to such problems is to design robust controllers (a conventional approach) as in Section 3.3.1 but these controllers suffer from poor performance.

The alternative to achieve robustness of the $MBPC$ design, suggested in this section, is that of an adaptive controller using a multi-model approach. This means that the internal model of the $MBPC$ controller will reflect at each time instant the system in use. Of course such a controller has got its own drawbacks when the nonlinearity is hard to characterise, the parameter variations are fast with respect to the process dynamics or the input excitation is insufficient if an identification method is employed to produce the internal model.

Before tackling the problem with an $MBPC$ multi-model regulator, which is more complicated than a fixed gain controller, we need to check if the problem can be solved by a simple linear controller. Note that it is not possible to judge the requirement for an adaptive controller just by looking at the variations of the process open loop.

The design stages for the $MBPC$ multi-model regulator are the following:

1. The whole range of the plant operating envelope is covered with a grid sufficiently fine to capture the nonlinearities of the model. The grid depends on the controller design method and specification.

2. Point-wise linearisations are produced in the vertices of this grid.

3. $MBPC$ controllers are designed for each of these models.

4. A global compensator is obtained by switching between these LTI laws.

5. Analysis has to be carried out as described in the work of Shamma et al. [SA90].

In the case of $MBPC$ the approach taken by Chow et al. [CKC96] is an interpolation of the linear designs for intermediate values of the scheduling variable done at the level of controller parameters. Additional problems connected with this approach arise when the scheduling variable is the system output. In this case it is difficult to choose which model from the several that cover the set-point change should be used. A multiple model design using switching should typically round the scheduling variable to the nearest vertex of the parameter grid and use the corresponding controller.

More details about the path taken in our work are to be found in Chapter 7 where we approach robust control as a multi-model adaptive control problem, namely one in which adaptation occurs by discrete changes to the control algorithm, rather than continuous tracking of a gradually-changing model, see as well [HM98a, HM98c].

Chapter 4

The tuning and implementation of $MBPC$ schemes

This chapter is devoted to the tuning and implementation of $MBPC$ schemes. This process consists of getting the process model, performing the off line design and the on line implementation. Therefore during the design of an $MBPC$ controller the following steps have to be followed:

1. Determine the requirements for the closed loop behaviour in terms of time and frequency criteria.

2. Produce the linear internal model of the plant.

3. Define the constraints related to inputs, rates of change in the inputs, outputs and states. Construct matrices that represent these.

4. Choose appropriate dimensions for control and prediction horizons

5. Choose the cost function weighting matrices.

6. Tune the performance of the closed loop system by performing closed loop linear and nonlinear simulations. This involves iteration of steps 4 and 5.

7. Perform a stability and robustness analysis.

This standard design cycle will be addressed in the next sections of this chapter with emphasis on the steps related to the $MBPC$ control scheme.

The linear internal model should relate to all manipulated variables as well the states and/or outputs depending on the formulation adopted (input/output or state space). After a nominal, unconstrained design is performed stability and performance of the nominal $MBPC$ controller are verified via theoretical analysis and simulation simultaneously. The tradeoff between robust stability and performance is explored generally on line by tuning the controller, so, weights in the cost function can be updated on-line to affect the solution.

4.1 From design criteria into method dependent parameters

In general the design criteria, given by the control problem to be tackled, are set up in method independent terms. This section considers possible requirements and provides a procedure to approximate them by means of objectives compatible with the $MBPC$ design methodology.

There are many adjustable tuning parameters in the case of $MBPC$ schemes. This section examines the effects of these upon closed loop performance trying to establish a systematic way of choosing them. In the case of $MBPC$ designs the design criteria are used to tune the method parameters which are: control and prediction horizons, weighting matrices, sampling time and sometimes even constraint boundaries. Therefore performance criteria defined in time domain such as rise time or overshoot are passed directly to the control method via these tuning parameters. The choice of them is based on several theoretical results (see previous chapter), but certain rules of thumb as well, integrated together in a trial-and-error tuning procedure.

Other main design criteria which are passed directly to the method are the constraints on actuators (rate and position limits) and safety constraints regarding the minimum or maximum allowable output values. The on-line optimisation will produce a solution with respect to these requirements. It is the designer's task to ensure the feasibility of the optimisation in the constrained case, generally by a suitable manipulation of the cost function and constraints.

Before starting the presentation of the translation of design criteria into method dependent objectives it is well to have defined the performance concept. Two types of performance can be distinguished: servo and regulator performance. Several aspects of the servo behaviour of a closed loop system can be measured by using a step as the reference trajectory. The variables defining part of the performance, used in the following part of this section, are:

- $t_r = \min_t(t \mid y(t) \geq 0.9)$ [s] – rise time (the time required for the system to reach 90 % of the unit step input value).

- $t_p = \arg(\max_t(y(t)))$ [s] – peak time (the time at which the response reaches its maximum value).

- $t_s = \min_t(t \mid\mid y(t) - y_{ss} \mid < 0.02 \mid y_{ss} \mid)$ [s] – settling time (time required for the system to settle within 2 % of its steady state value).

- $M_p = (y(t_p) - y_{ss}) \times 100$ [%] – the overshoot.

- $e_{ss} = \mid (y_{ref} - y_{ss}) \mid \times 100$ [%] – the steady state error.

where y_{ss} represent the steady state value of $y(k)$.

Figure 4.1 shows the variables involved in the servo performance definition:

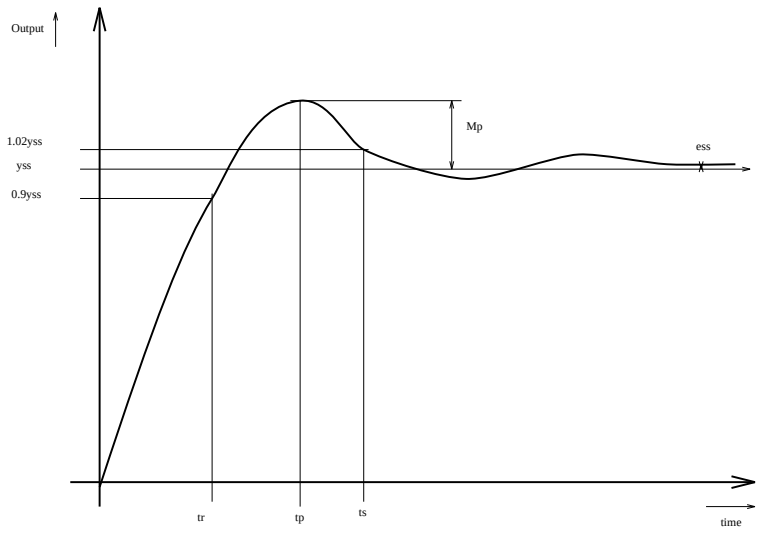


Figure 4.1: Servo performance criteria

Another useful relation is between the maximum modulus r of the discrete time closed-loop eigenvalues and the settling time t_s: as r approaches zero the settling time decreases. This is a useful expression while employing the exponential weights for the $MBPC$. In the case of a first or second order systems the settling time can be given by the minimum value of n satisfying $r^n \leq 0.01$ (*i.e.* by the value of $\min_n(n : n \geq \log_r 0.01)$). This kind of relation becomes more difficult to find as order increases [YC93].

4.1.1 Control and prediction horizons

The major influence of the control horizon is on stability of the closed loop. Once this was achieved by an appropriate tuning its main influence is upon the performance of the controlled system. Some influences upon robustness could be seen as well [Soe92]. The control horizon has to be bigger that the number of unstable plant modes in order to ensure $MBPC$ nominal stability [RM93, Sco97], see as well Theorem 3.1.2. Therefore, in case of the control horizon tuning we perform a step response analysis of the system assuming a predefined sampling period followed by a trial-and-error tuning process.

In general the choice of the prediction horizon is made in connection with the knowledge of the future set point and the dynamics of the system. Therefore this horizon is derived from the settling time. The solution speed will be decreased to an unacceptable extent for big values of the prediction horizon. An increase of this horizon should be considered when the closed loop proved to have long settling time. The prediction horizon must contain at least the non-minimum phase behaviour of the system to be controlled using $MBPC$.

To complete the reader's image about tuning we recall some special cases. For instance, the example of mean-level control ($N_u = 1$) applied to a stable plant is one of them. In this case it is possible to find one control increment such that the reference will be met by the plant output in steady state. This will allow the plant to react to set-point changes while being able to reject disturbances in the shortest time possible.

The case of the perfect controller which sets both the control and prediction horizons

to 1 is definitely a particular one. This because assuming that the plant has unstable zeros these will lead to an unstable controller which in the case of mismodeling can drive the plant unstable.

In the other cases discussed in the literature, such as dead-beat control, we need a control horizon as large as the number of states in the plant model and a prediction horizon having a value twice the control horizon. In this case the idea is to bring all the plant modes to zero and then to allow enough time to the output to settle to the value of the reference making sure that are no delayed modes inside the controller which might emerge in the future.

4.1.2 The sampling period

The sampling period plays an important role in $MBPC$ controllers. It is important to be aware that the sampling time has a strong influence upon all tuning parameters of the MBPC technique. This parameter also defines the discrete time state space representation of the continuous time linear model used as the internal model within the $MBPC$ scheme. The smaller the sampling period, the more accurate the discretisation will be. On another hand, for a small sampling period, a big increase in the computational load can be observed. This limits the controller bandwidth.

The way we usually set the sampling time in the MIMO case is by looking to all the transfer functions from inputs to outputs and using the one which exhibits the largest bandwidth. Then the frequency in rad/s at the crossover is augmented with another decade – upon which we require accuracy from the discrete time model. This is considered as being the Nyquist frequency ($\frac{\pi}{T_s}$) and therefore defines the sampling time. In other words the sampling time corresponds to the smallest time constant which appears in the continuous time model. Another restriction for T_s is given by the application. For instance, a civil flight control system operates using a sampling frequency in the range $8 - 20Hz$.

There is a tradeoff between decreasing the sampling period and increasing dimensions of matrices involved in the $MBPC$ algorithm or the number of optimisations performed in the time unit. To conclude, if the sample period is small in relation with the settling time this will result in large control and prediction horizons possibly causing numerical problems. However, the smaller the sampling period, the better can a reference trajectory be tracked or disturbance rejected assuming no inter-sampling behaviour problems. The attempts of [LGM92] are the only available ones trying to deal with $MBPC$ tuning in a sample data system framework.

4.1.3 Weighting matrices

The weighting matrices upon the outputs and control increments are important design parameters. The matrix that weights the difference between the set point and the plant model output is positive definite, as well as the command increment weighting matrix. Both influence the performance properties required from the closed loop system.

The control increment weighting matrix affects the control increment magnitude. For instance increasing this will reduce the control activity leading to a "switch off" of the feedback action. Therefore, in the case of a stable plant we can expect to obtain a stable

closed loop by increasing the control weighting sufficiently. The drawback of this is slow response to disturbances, since the resulting control actions will be small.

The output (error) weighting matrix starts to be used extensively when dealing with MIMO systems. In the SISO case having just the control weighting matrix is enough. For complex MIMO plants manipulating both the control and output weighting matrices allows the designer to decouple input/output channels providing the closed loop system with the required performance criteria.

At the moment, we have limited theoretical tools to guarantee stability in the case of the constrained optimisation. As a result the first step in the controller design has to be the tuning of the controller in the absence of constraints. Then, a softening mechanism can be employed as in [Zaf90], a formulation that covers any mixture of hard and/or soft constraints. We allow a violation of the hard constraint by an amount $\nu \geq 0$. Next, we add to the cost function 2.2.5 the term $\| \nu \|^2_W$, where W is the weight that determines the effect of softening: for $W = \infty$ we get hard constraints; $W = 0$ corresponds to a complete removal of the constraints.

Other methods of dealing with infeasibility situations were already described in the previous chapter. When using receding control with exponential weighting a special meaning is provided to both weighting matrices. In this case they represent the tool to provide a prescribed degree of stability to the $MBPC$ scheme.

The tuning of weighting matrices is not a one step process. At the beginning we tune the weights of the MBPC controller using the time simulations involving the linear model of the plant. This step is followed by fine tuning done by time simulations employing the full nonlinear model of the plant and constraints. This tuning is another example of a trial-and-error procedure involved in choosing the MBPC controller parameters. Such a procedure is time consuming due to the mixture of discrete and continuous time blocks which requires the Simulink software to use an integration step upper bounded by the smaller sampling time.

4.1.4 Set point generation

Traditionally set-point filtering has been used to reduce actuator saturation. This was achieved by passing the set-point changes through a filter and only then forming the error vector. In such a way we limit its amplitude such as not to have the actuators being saturated.

Even in the $MBPC$ case, when we can handle actuator saturation via constraints, there are reasons to employ such pre-filters such as: the lack of understanding of the closed loop behaviour when a set of constraints are active, the necessity to keep control authority at any time to deal with large disturbances and the fact that set-point filtering will remove only high frequency changes (low frequency changes can still saturate the actuators). Therefore, the set point filter represents an indirect tuning parameter for $MBPC$ schemes. More theoretical insight into this aspect is offered in Section 4.2.

When an $MBPC$ controller was used as inner loop stability augmentation system *a priori* knowledge of the set point was not assumed. There are other choices to generate the set-point from the present command. For example a simple way to generate the reference

trajectory $r(k+l)$ is via a first order filter:

$$\begin{aligned} r(k) &= y(k) \\ r(k+l) &= \alpha r(k+l-1) + (1-\alpha)s(k) \end{aligned}$$

where $y(k)$ is the current output, $s(k)$ is current set-point at time k for $l = 0 \dots N_y$ and $\alpha \in [0, 1]$.

In general the reference trajectory generation assumes a certain knowledge of the set-point to be followed which is embedded in the pre-filter design. By making an *a priori* decision on the pre-filter structure it is possible to degrade the achievable performance because the pre-filter is usually chosen to give the fastest possible response during typical demands.

In the next section we will show what are the trade-offs between designing an appropriate filter and choosing a specific pattern of cost function weights.

4.2 Filtering and plant shaping

In the literature, see [GPM89], filtering of the future set-point is often recommended as shown in Figure 2.2 and is considered as an additional degree of freedom used together with the traditional ones which are horizons and weighting matrices.

In [RG96] the authors developed an algorithm with better tracking based on minimising a cost function which is a measure of the tracking across the whole simulation. A similar approach developed in [ANA96] enhances the tracking performance by adapting a pre-filter on-line with the aim of achieving good tracking and low sensitivity properties for the closed loop. In other papers, such as [CZ91], the authors have used filtered set-points, inputs and measurements in the cost function.

In the following we investigate the cases when the above formulations are equivalent:

Theorem 4.2.1 *[HM97b] Consider the MBPC problem:*

$$J(k) = \sum_{l=1}^{N_y} \|(\tilde{y}_f(k+l) - r(k+l)\|^2_{Q(l)} + \sum_{l=1}^{N_u} \|\Delta u_f(k+l-1)\|^2_{R(l)} \tag{4.1}$$

If $e_f(z) = W_y(z)e(z)$ *with* $e(z) = \tilde{y}(z) - s(z)$, $r(z) = F(z)s(z)$, $\tilde{y}_f(z) = W_y(z)\tilde{y}(z)$, $\Delta u_f(z) = W_u(z)^{-1}\Delta u(z)$ *and moreover* $W_y(z) = F(z)$ *with* $W_y(z)$ *and* $W_u(z)^{-1}$ *being FIR filters having a causal and anti-causal part:*

$$W_y(z) = a_{-N_p}z^{-N_p} + \dots + a_{-1}z^{-1} + a_0z^0 + a_1z^1 + \dots + a_{N_f}z^{N_f}$$

$$W_u(z)^{-1} = b_{-N_q}z^{-N_q} + \dots + b_{-1}z^{-1} + b_0z^0 + b_1z^1 + \dots + b_{N_g}z^{N_g}$$

where $N_f \leq N_y$ *and* $N_g \leq N_u$ *then the MBPC problem can be reformulated as:*

$$J(k) = \|E(k)\|^2_{\hat{\mathcal{Q}}_f} + \|\Delta U(k)\|^2_{\hat{\mathcal{R}}_f} \tag{4.2}$$

where $\hat{\mathcal{Q}}_f = \hat{T}^T \mathcal{Q}_f \hat{T}$ *and* $\hat{\mathcal{R}}_f = \hat{X}^T \mathcal{R}_f \hat{X}$ *and*

$$\begin{aligned} \Delta U(k) &= \begin{bmatrix} \Delta u(k-N_q)^T & \dots & \Delta u(k+N_g+N_u-1)^T \end{bmatrix}^T \\ E(k) &= \begin{bmatrix} e(k-N_p)^T & \dots & e(k+N_f+N_y)^T \end{bmatrix}^T \\ \mathcal{Q}_f &= diag[Q(1), \dots, Q(N_y)] \\ \mathcal{R}_f &= diag[R(1), \dots, R(N_u)] \end{aligned}$$

and

$$\hat{T} = \begin{bmatrix} a_{-N_p} & \dots & a_0 & \dots & a_{N_f} & 0 & 0 & \dots & 0 \\ 0 & a_{-N_p} & \dots & a_0 & \dots & a_{N_f} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & a_{-N_p} & \dots & a_0 & \dots & a_{N_f} \end{bmatrix}$$

$$\hat{X} = \begin{bmatrix} a_{-N_q} & \dots & a_0 & \dots & a_{N_g} & 0 & 0 & \dots & 0 \\ 0 & a_{-N_q} & \dots & a_0 & \dots & a_{N_g} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & \dots & 0 & a_{-N_q} & \dots & a_0 & \dots & a_{N_g} \end{bmatrix}$$

Proof Let $E_f(k) = \begin{bmatrix} e_f(k+1)^T & \dots & e_f(k+N_y)^T \end{bmatrix}^T$ and $\Delta U_f(k) = \begin{bmatrix} \Delta u_f(k)^T & \dots & \Delta u_f(k+N_u-1)^T \end{bmatrix}^T$.
Then the cost-function (4.1) can be written as:

$$J(k) = \|E_f(k)\|^2_{\mathcal{Q}_f} + \|\Delta U_f(k)\|^2_{\mathcal{R}_f}. \tag{4.3}$$

The filtered error is:

$$\begin{aligned} e_f(k+1) &= a_{-N_p} e(k-N_p) + \dots + a_{-1} e(k-1) + a_0 e(k+1) + \dots + 0e(k+N_f+N_y) \\ &\quad \dots\dots\dots\dots\dots \\ e_f(k+N_y) &= 0e(k-N_p) + a_{-N_p} e(k+N_y-N_f-N_q 1) + \dots \\ &\quad + a_0 e(k+N_y-N_f) + \dots + a_{N_f} e(k+N_f+N_y) \end{aligned}$$

Thus the expression for the filtered error is $E_f(k) = \hat{T} E(k)$. For $\Delta u_f(k), \dots, \Delta u_f(k+N_u)$ a similar expression can be derived such that $\Delta U_f(k) = \hat{X} \Delta U(k)$. Therefore with $\hat{\mathcal{Q}}_f = \hat{T}^T \mathcal{Q}_f \hat{T}$ and $\hat{\mathcal{R}}_f = \hat{X}^T \mathcal{R}_f \hat{X}$ the cost function 4.3 can be written as:

$$J(k) = \|E(k)\|^2_{\hat{\mathcal{Q}}_f} + \|\Delta U(k)\|^2_{\hat{\mathcal{R}}_f}$$

□

The conclusions of the above theorem can be summarised in the following four remarks:

Remark 4.2.2 *Note that the use of an anti-causal part in the FIR filters is possible when there is* apriori *knowledge of the set-point and of the predicted output over the relevant horizon.*

Remark 4.2.3 *Solving the formulation (4.1) or equivalently (4.3) is computationally less expensive than formulation (4.2) since the prediction and control horizon are considerably shorter. Therefore we recommend use of filtering for actual implementation.*

Remark 4.2.4 *The positive definite weighting matrices $\hat{\mathcal{Q}}_f$ and $\hat{\mathcal{R}}_f$ no longer have the diagonal block structure as in the case of cost function (4.3). This shows the way in which the filters ensure extra degree of freedom to the optimisation cost function. Note that with full structured $\hat{\mathcal{Q}}_f$ and $\hat{\mathcal{R}}_f$ the problem can no longer be written in the form:*

$$J(k) = \sum_{l=-N_p}^{N_f+N_y} \|(\tilde{y}(k+l) - s(k+l)\|^2_{\hat{Q}(l)} + \sum_{l=-N_q}^{N_g+N_u} \|\Delta u(k+l-1)\|^2_{\hat{R}(l)}$$

Remark 4.2.5 *The assumption $F(z) = W_y(z)$, although quite restrictive and unnatural, was made in order to simplify the formulation and show how one can reformulate the original problem.*

4.3 Ensuring robust tracking and disturbance rejection

During the development of $MBPC$ techniques researchers were concerned with ensuring integral action with their control schemes. The aim was to have this property built into the algorithm in order to cope with random steps occurring at random times or, in the stochastic sense, with integrated white noise. These signals represent a class of disturbances encountered in many practical systems, see [CMT87].

Inherent integral action, in conjunction with an appropriate prediction model, written in the control increment $\Delta u(k)$, ensures zero offset in the case of constant disturbances [CM89b].

Recall that the conventional setup of the $MBPC$ formulation in the state space form which include integral action is characterised by:

- The plant model:

$$\begin{aligned} x(k+1) &= Ax(k) + Bu(k) \\ \tilde{y}(k) &= Cx(k) + n(k) = y_x(k) + n(k) \end{aligned}$$

  where $u(k)$, $x(k)$, $\tilde{y}(k)$, $n(k)$ are the input, state, output, output disturbance vectors, respectively, and $y_x(k)$ is the output vector free of disturbance.

- The cost function described in equation (2.1)

- The LTI state space prediction model:

$$\begin{aligned} \begin{bmatrix} \Delta x(k+1) \\ y_x(k+1) \end{bmatrix} &= \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ y_x(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} \Delta u(k) \\ y(k) &= \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ \tilde{y}_x(k) \end{bmatrix} + n(k) \end{aligned}$$

- The linear inequalities expressing the $MBPC$ constraints:

$$\mathcal{D}\Delta U(k) \leq \mathcal{E}\begin{bmatrix} u(k-1) \\ \hat{x}(k) \\ y(k) \\ c(k) \end{bmatrix}, \mathcal{D} = \begin{bmatrix} L \\ M\Lambda \\ N\mathcal{G} \end{bmatrix}$$

$$\mathcal{E} = \begin{bmatrix} 0 & 0 & 0 & I_\lambda & 0 & 0 \\ -M\mathcal{I} & 0 & 0 & 0 & I_\mu & 0 \\ 0 & -N\mathcal{F} & -N\mathcal{H} & 0 & 0 & I_\nu \end{bmatrix}$$

All the matrices involved in the above equations are defined in Section 2.2.

In this section we develop a scheme to deal with a wider class of disturbance and/or reference signals within $MBPC$ controlled plants. This scheme will generalise the usual integral action of $MBPC$ which deals with piecewise constant signals.

The idea behind this development is the well known *internal model principle* which states that, in order to have a disturbance which does not decay to zero rejected or a reference tracked, its corresponding model has to be included in the controller. Our approach is to augment the plant model with the disturbance model and then use the augmented model as a prediction model to construct the optimiser cost function.

By assuming that the disturbance and/or reference signal satisfies the difference equation (4.4) this formulation allows a wide variety of signals such as steps, ramps or sinusoids, to be described:

$$n(k+1) + \alpha_1 n(k) + \alpha_2 n(k-1) = 0 \tag{4.4}$$

This equation can be rewritten in the state space representation:

$$\begin{aligned} \begin{bmatrix} n(k) \\ n(k+1) \end{bmatrix} &= \begin{bmatrix} 0 & I \\ -\alpha_2 I & -\alpha_1 I \end{bmatrix} \begin{bmatrix} n(k-1) \\ n(k) \end{bmatrix} \\ n(k) &= \begin{bmatrix} 0 & I \end{bmatrix} \begin{bmatrix} n(k-1) \\ n(k) \end{bmatrix} \end{aligned}$$

For instance, using the above model a step can be obtained when $n(0) = 0$, $n(1) = a$, $\alpha_1 = -1$, $\alpha_2 = 0$, a ramp when $n(0) = 0$, $n(1) = a$, $\alpha_1 = -2$, $\alpha_2 = 1$, $a \in \mathcal{R}$ or the sinusoid when $n(0) = b\sin(\phi)$, $n(1) = b\sin(aT_s + \phi)$, $\alpha_1 = -2\cos(aT_s)$, $\alpha_2 = 1$, $a \in \mathcal{R}$.

We define the generalised input:

$$\psi(k) = u(k) + \alpha_1 u(k-1) + \alpha_2 u(k-2)$$

the generalised state:

$$\xi(k) = x(k) + \alpha_1 x(k-1) + \alpha_2 x(k-2)$$

and the corresponding output:

$$\begin{aligned} \zeta(k) &= \tilde{y}(k) + \alpha_1 \tilde{y}(k-1) + \alpha_2 \tilde{y}(k-2) \\ &= \tilde{y}_x(k) + \alpha_1 \tilde{y}_x(k-1) + \alpha_2 \tilde{y}_x(k-2) \\ \zeta(k) &= C\xi(k) \end{aligned}$$

Hence, the generalised state space representation of the plant model is:

$$\begin{aligned} \xi(k+1) &= A\xi(k) + B\psi(k) \\ \zeta(k) &= C\xi(k) \end{aligned}$$

together with

$$\begin{aligned} \eta(k+1) &= A_n\eta(k) \\ n(k) &= C_n\eta(k) \end{aligned}$$

where $\eta(k) = \begin{bmatrix} n(k-1) \\ n(k) \end{bmatrix}$ represent the state of the disturbance model.

This state space representation, augmented with the noise model (4.4) and using the vector $y_x(k)$ of outputs free of disturbance, can be written as:

$$\begin{bmatrix} \xi(k+1) \\ \eta(k+1) \\ y_x(k) \\ y_x(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 & 0 & 0 \\ 0 & A_n & 0 & 0 \\ 0 & 0 & 0 & I \\ CA & 0 & -\alpha_2 I & -\alpha_1 I \end{bmatrix} \begin{bmatrix} \xi(k) \\ \eta(k) \\ y_x(k-1) \\ y_x(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \\ CB \end{bmatrix} \psi(k)$$

$$\tilde{y}(k) = \begin{bmatrix} 0 & C_n & 0 & I \end{bmatrix} \begin{bmatrix} \xi(k) \\ \eta(k) \\ y_x(k-1) \\ y_x(k) \end{bmatrix}$$

Actually this is the only way we can connect the output $\tilde{y}(k)$ with the generalised state vector. Let $z(k) = \begin{bmatrix} \xi(k)^T & \eta(k)^T & y_x(k-1)^T & y_x(k)^T \end{bmatrix}^T$ then the model written in a compact form is:

$$\begin{aligned} z(k+1) &= \hat{A}z(k) + \hat{B}\psi(k) \\ \tilde{y}(k) &= \hat{C}z(k) \end{aligned} \tag{4.5}$$

In fact this is going to be the model used to build the prediction across the N_y horizon and finally the cost function to be optimised.

In this process the state corresponding to the compact form will be partly measured and/or partly estimated depending on the type of formulation employed (e.g. with or without state measurement available). The difference between the measured and the estimated output will be used to start the predictor which will provide the predicted values of $\tilde{y}$ across the N_y horizon. This is actually how feedback information comes into play.

The cost function of the generalised formulation is:

$$J(k) = \sum_{l=1}^{N_2} \|(\hat{C}z(k+l) - r(k+l)\|^2_{Q(l)} + \sum_{l=1}^{N_u} \|\psi(k+l-1)\|^2_{R(l)}$$

Here the reader should note the differences compared with the conventional cost function (2.1). The model used for prediction was defined by (4.5).

Associated with the cost function and the prediction model we introduce the following constraints upon the variables involved in the control process:

- the inputs levels: $u_l(l) \leq u(l) \leq u_u(l)$ where $k \leq l \leq k + N_u - 1$
- the generalised input levels: $\psi_l(l) \leq \psi(l) \leq \psi_u(l)$ where $k \leq l \leq k + N_u - 1$
- the output (in general other linear combinations of states): $y_l(l) \leq \tilde{y}(l) \leq y_u(l)$ where $k + N_1 \leq l \leq k + N_2$

Following the procedure of stacking constraints developed in Section 2.2 we present in the next part of this section a way to impose constraints directly onto the system state despite of the use of the generalised state vector in the robust tracking formulation.

We start by considering the state constraints written in the following linear inequality over the whole prediction horizon $PX(k) \leq o(k)$. Over the same horizon the augmented generalised state is computed using the equation $Z(k) = \mathcal{F}(k)z(k) + \mathcal{G}\Psi(k) + \mathcal{H}(k)y(k)$ where $\mathcal{F}(k)$, $\mathcal{G}$, $\mathcal{H}(k)$ are defined in Section 2.2 and $z(k) = [\xi(k)^T, \nu(k)^T, y_x(k-1)^T, y(k)^T]^T$.

Defining the following matrix having compatible dimensions $\bar{S} = \begin{bmatrix} I & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots \\ I & 0 & 0 & 0 \end{bmatrix}$ we write the generalised state as $\xi(k) = \bar{\mathcal{F}}(k)z(k) + \bar{\mathcal{G}}\Psi(k)\bar{\mathcal{H}}(k)y(k)$ where $\bar{\mathcal{F}}(k) = \bar{S}\mathcal{F}(k)$, $\bar{\mathcal{G}} = \bar{S}\mathcal{G}$ and $\bar{\mathcal{H}}(k) = \bar{S}\mathcal{H}(k)$.

Then we consider the vector of future states defined over the prediction horizon with respect to $\xi(k)$:

$$\begin{bmatrix} x(k+1) \\ \vdots \\ x(k+N_2) \end{bmatrix} = \begin{bmatrix} \xi(k+1) \\ \vdots \\ \xi(k+N_2) \end{bmatrix} + \begin{bmatrix} 0 & 0 & \dots & 0 & 0 & 0 \\ -\alpha_1 & 0 & \dots & 0 & 0 & 0 \\ -\alpha_2 & -\alpha_1 & \dots & 0 & 0 & 0 \\ 0 & -\alpha_2 & \dots & 0 & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 0 & \dots & -\alpha_2 & -\alpha_1 & 0 \end{bmatrix} \begin{bmatrix} x(k+1) \\ \vdots \\ x(k+N_2) \end{bmatrix} + \begin{bmatrix} -\alpha_2 & -\alpha_1 \\ 0 & -\alpha_2 \\ 0 & 0 \\ \dots & \dots \\ \dots & \dots \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(k-1) \\ x(k) \end{bmatrix}$$

which boils down to:

$$\begin{aligned} X(k) &= I\xi(k) + \bar{\Delta}X(k) + \Lambda \begin{bmatrix} x(k-1) \\ x(k) \end{bmatrix} \\ X(k) &= (I - \bar{\Delta})^{-1} \left(\xi(k) + \Lambda \begin{bmatrix} x(k-1) \\ x(k) \end{bmatrix} \right) \end{aligned}$$

if matrix $(I - \bar{\Delta})$ is non-singular.

Hence the constraints imposed upon the state over the future prediction horizon $PX(k) \leq o(k)$ can be written as:

$$\begin{aligned} P(I - \bar{\Delta})^{-1} \left(\xi(k) + \Lambda \begin{bmatrix} x(k-1) \\ x(k) \end{bmatrix} \right) &\leq o(k) \\ P(I - \bar{\Delta})^{-1} \left(\bar{\mathcal{F}}(k)z(k) + \bar{\mathcal{G}}\Psi(k) + \bar{\mathcal{H}}(k)y(k) + \Lambda \begin{bmatrix} x(k-1) \\ x(k) \end{bmatrix} \right) &\leq o(k) \end{aligned}$$

which leads to:

$$P(I-\bar{\Delta})^{-1}\bar{\mathcal{G}}\Psi(k) \leq o(k) - P(I-\bar{\Delta})^{-1}\left(\bar{\mathcal{F}}(k)z(k) + \bar{\mathcal{H}}(k)y(k) + \Lambda\begin{bmatrix}x(k-1)\\ x(k)\end{bmatrix}\right)$$

These constraints can be cast into the following linear inequality by stacking together scalar inequalities over the control and prediction horizon, respectively :

$$\bar{\mathcal{D}}\Psi(k) \leq \bar{\mathcal{E}}\begin{bmatrix}\begin{bmatrix}u(k-2)\\ u(k-1)\end{bmatrix}\\ \begin{bmatrix}x(k-1)\\ x(k)\end{bmatrix}\\ z(k)\\ y(k)\\ c(k)\end{bmatrix}$$

where $c(k) = [l(k)^T, m(k)^T, n(k)^T, o(k)]^T$ and the matrices $\bar{\mathcal{D}}$ and $\bar{\mathcal{E}}$ embedding elements defined in Section 2.2 have the following form:

$$\bar{\mathcal{D}} = \begin{bmatrix} L \\ M(I-\bar{\Delta})^{-1} \\ N\mathcal{G} \\ P(I-\bar{\Delta})^{-1}\bar{\mathcal{G}}\end{bmatrix}$$

$$\mathcal{E} = \begin{bmatrix} 0 & 0 & 0 & 0 & I_{\Psi(k)} & 0 & 0 & 0 \\ -M(I-\bar{\Delta})^{-1}\bar{\Lambda} & 0 & 0 & 0 & 0 & I_{U(k)} & 0 & 0 \\ 0 & 0 & -N\mathcal{F}(k) & -N\mathcal{H}(k) & 0 & 0 & I_{Z(k)} & 0 \\ 0 & -P(I-\bar{\Delta})^{-1}\left(\bar{\mathcal{F}}(k)+\bar{\Lambda}\right) & -P(I-\bar{\Delta})^{-1}\bar{\mathcal{H}}(k) & 0 & 0 & 0 & 0 & I_{X(k)}\end{bmatrix}$$

Now the constrained quadratic programming problem to be solved on-line is completely defined. In terms of implementation Figure 4.2 defines the structure used and the corresponding signals.

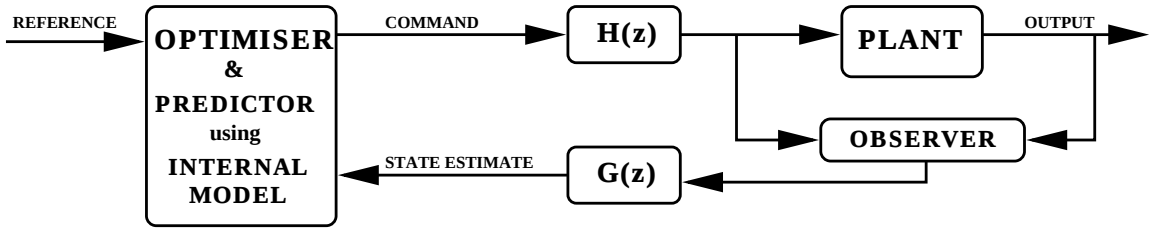


Figure 4.2: The structure of the generalised $MBPC$ controller

The filters $G(z)$ and $H(z)$ have the following form $G(z) = 1 + \alpha_1 z^{-1} + \alpha_2 z^{-2}$ and $H(z) = (1 + \alpha_1 z^{-1} + \alpha_2 z^{-2})^{-1}$ where $\xi(k) = G(z)x(k)$ and $u(k) = H(z)\psi(k)$.

The proposed formulation results in the disturbance being rejected asymptotically if the $MBPC$ controller stabilises the system (4.5) which means that $z(k)$ tends to zero in an asymptotic manner. Therefore assuming $r(k+l) = 0$ for $l = 1 \ldots N_y$ then $\tilde{y}(k)$ tends asymptotically to zero which ensures the required property. As a result we require that the original plant, in its discrete time representation, should have zeros differently located than the poles of the filter $H(z)$ in order to avoid their cancelling.

For example, in the case of ramp rejection/following since the plant operates on $\Delta\Delta u(k)$ rather than $u(k)$ (by "plant" here we mean the model used for prediction), we require that the original plant should have no zeros located at $z = 1$ which otherwise will cancel the ramp model ($(1 - z^{-1})^2$) introduced by this way of operation. For instance, in Section 6.5, this approach is used in following a landing path reference which has the form of a ramp.

Note that higher order disturbance models than (4.4) can be handled in an analogous manner. This gives a way of handling more than one type of disturbance simultaneously.

Robust tracking or disturbance rejection can now be achieved due to a new formulation which provides rejection and/or tracking for a wider class of reference and/or disturbance signals via an appropriate prediction model constructed using the *internal model principle*. Therefore, including the disturbance model together with the prediction model ensures offset-free control even when there is a significant mismatch between the model and the actual plant. This algorithm was employed as a basis for the "Development Space" described in the next section.

4.4 The $MBPC$ Development Space (DS)

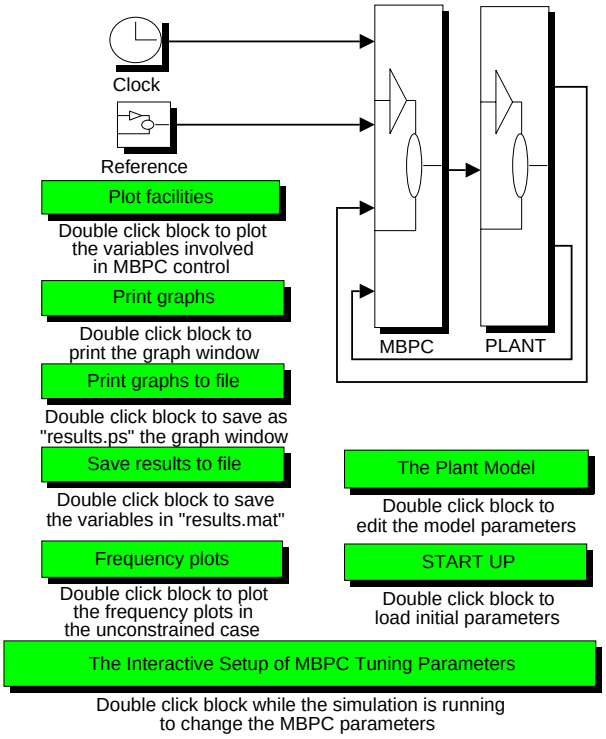


Figure 4.3: The SIMULINK $MBPC$ development space

The section aims to provide the reader with an answer to the question "Why a $MBPC$ development space?" and to present a particular proposal for such a software. Then a

description of the Graphical User Interface (GUI) is given followed by a presentation of the Development Space (DS) at work.

The design environment was intended to be a useful tool for designers involved in $MBPC$ type of control. The architecture was defined in a flexible way to give opportunity to the designer to use this tool as a Development Space which can be customised to his needs [HM97a]. For example the use of predefined blocks and embedded algorithms will enable him to construct simulation models. Otherwise, a too dedicated user interface will limit the user to several predefined operations which will be too restrictive for an experienced user. The features of the Development Space were defined after a careful look at the previous attempts such as: the Model Predictive Control Toolbox for use with Matlab [MR95] and the Predictive Adaptive Control Environment [SCC95].

We call this CAD application a "Development Space" because we would like to emphasise the difference between this and a standard MATLAB toolbox [HM97a]. Having a user interface which utilises SIMULINK features, in an open environment for developers, this application goes beyond a normal toolbox in terms of customisation but, not as far as an application dedicated to a particular $MBPC$ control problem. The tuning interface is constructed in such a way as to allow the designer to adjust various design parameters while the simulation is running.

Convenient general purpose $MBPC$ design software for MIMO systems using state space internal models is not available. The existing software does not seem to allow a straightforward access from the MATLAB–SIMULINK environment [MR95] or when MIMO plants are required to be controlled [SCC95]. In [SCC95] facilities such as an automatic procedure to define the optimisation horizons may fail in some particular cases. This limits the usefulness of the on-line identification feature which is provided by the toolbox.

Our Development Space is constructed using the MATLAB–SIMULINK software and its associated standard user interface. This environment uses state-space discrete $MBPC$ internal models, but the simulation plant model can be nonlinear. It handles non square systems and disturbances. The Development Space assumes that either direct measurements of the plant states are available, or that they are accurately estimated using a suitable observer, which the user must supply in the form of a Simulink block.

In effect the Development Space allows nonlinear simulations with the $MBPC$ controller appearing as a "block" in the conventional way. By contrast the MPC Toolbox absorbs the whole SIMULINK model inside the function, so that the usual SIMULINK environment is not available while running a simulation.

Explicit constraints on manipulated input changes, input variables, outputs and states are handled. Tools allow the user to handle unconstrained problems, characterised by an analytic solution, and constrained problems solved by means of hill climbing algorithms.

The environment shown in Figure 4.3 provides functions and has features that allow the user to design and test in simulation $MBPC$ controllers for simple and complex plants. Such an approach supports once more the idea of a "Development Space" instead of a toolbox. As the only option, state space based internal models were used because of the reliability of the numerical algorithms involved and the possibility to interface them with other state-space based tools such as the subspace method – used in identification. The use of nonlinear internal models for prediction and therefore, for the computation of the

optimal control law is not currently possible.

4.4.1 The Graphical User Interface (GUI)

The main functions of the development space (see Figure 4.3) are: *Interactive setup of the MBPC tuning parameters*, *Plant model setup*, *Start up*, *Frequency plot*, *Plot facilities* and *Print graph*.

The tuning parameters of the *MBPC* method are requested by the dialog box shown in Figure 4.4. It is important to note here that any change in the parameters while the simulation is running is passed immediately to the *MBPC* algorithm. This feature can be used to understand the influence of the various parameters and the way of tuning them.

The *Plant model setup* function enables us to setup the internal plant model for the *MBPC* control algorithm within a text editor window via a template file. The file to be edited has comments included that are intended to help the user in changing the parameters according to his particular problem.

Activating the *Start up* push button places all variables required by the *MBPC* method within the MATLAB workspace by running the functions `setup.m`, see Appendix A.1, and `algorithm.m`, see Appendix A.2. SIMULINK uses specific variables from the workspace and passes them to the DS functions as parameters. When a SIMULINK model is created, a new function, called *S-function*, becomes available in MATLAB. An *S-function* behaves like any other MATLAB function. A full description of the concept is given in [Mat95]. Once all the variables are loaded and computed we can access the *Frequency plot* facilities that will provide us with information regarding the closed loop transfer function (complementary sensitivity) in the unconstrained case (see Figure 4.6). The frequency response of the controlled plant is saved as well in MVFR format for later use within the MFD Toolbox [FMB90].

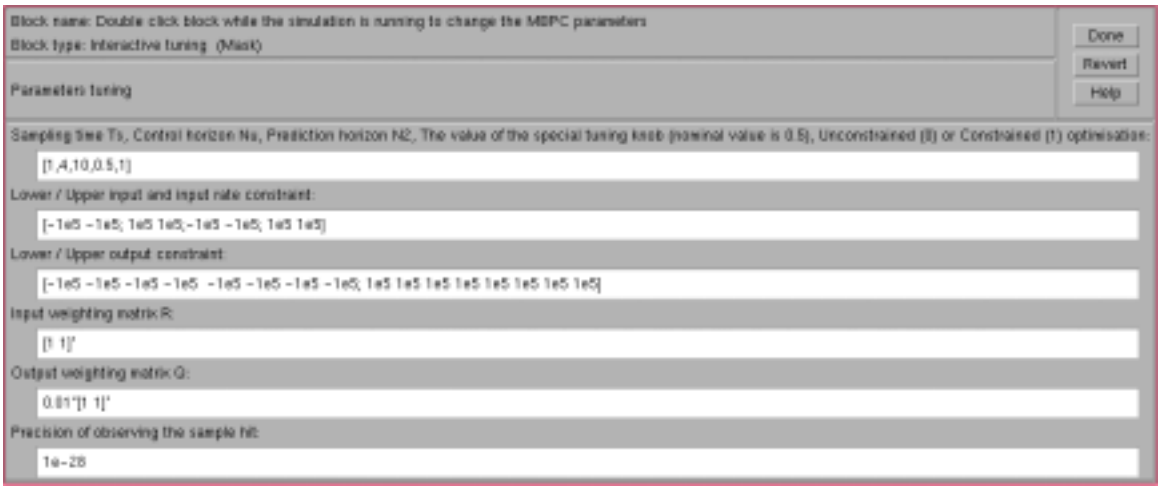


Figure 4.4: The *MBPC* parameters dialog box

When the simulation is finished, clicking on the *Plot facilities* push button brings

forward a multiple graph (shown in Figure 4.7) showing measurements of the plant together with the variation of the states (*i.e.* Δx and the command provided by the $MBPC$ controller.

4.4.2 Development Space's architecture

Functional blocks are: the reference block, the plant block and the $MBPC$ block. The engine of the design environment is the $MBPC$ controller, having the optimiser as a central point.

The blocks shown in Figure 4.5 are used to produce the signals required by the $MBPC$ algorithm. The MATLAB workspace is provided with the signals involved in the control for analysis and plotting purposes. The optimiser, see Appendix A.3, is written as a SIMULINK S-function that has no states but runs a quadratic program. Only the first movement of the solution vector is used for control purposes.

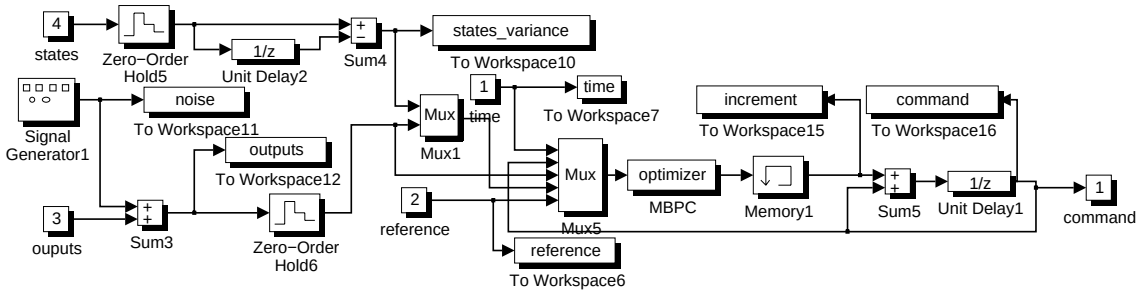


Figure 4.5: The $MBPC$ block

The plant block hosts the linear or nonlinear model of the plant. We note here that the main requirement of the Development Space is that the state measurements from the plant model are available as long as estimation of them is not used.

When a reduced model of the plant is employed via truncating the states, as the $MBPC$ internal model, a state transformation is needed in order to provide the reduced model, employed for prediction, with the necessary initial measurement. Therefore, in the SIMULINK plant block we have included the necessary transformation matrix. Within the reference block the designer has to provide reference signals to the algorithm, reference signals which can be changed during the run. The reference trajectory is either generated during the simulation run, or it can be pre-computed and read from a file. The latter approach allows the $MBPC$ to anticipate the future set-point movements.

4.4.3 The Development Space at work

In order to give more insight into the use of the Development Space we are going to take a longitudinal aircraft model (a linear representation of the nonlinear model used in the next chapter) and pass it through all the stages required for a proper set-up and simulation using the $MBPC$ environment.

The setting up involves two main steps:

1. The definition of the linearised plant model, used as internal model within the $MBPC$ algorithm. Therefore, we are using two models, one to represent the plant (non-linear) and one (linear) for the $MBPC$ internal model.
2. The definition of the $MBPC$ tuning parameters (control and prediction horizons, command and output weighting matrices).

The first step can be accomplished once a linear continuous representation of the plant is produced using standard trim and linearisation procedures. Such a representation is used in order to define the discretized version of the model at the sampling time required. This discrete system is produced from the continuous time one by using specific commands from the MATLAB μ-*tools* or Control Toolbox. Model reduction can be applied in order to reduce the computational burden due to the large dimension of the internal model. This internal model will assume a simple way to model disturbances as being the difference between the real output and the model output and assumes that the same disturbance persists across the prediction horizon.

The linear representation of the plant is given in Table 4.1 (see next chapter for the meaning of inputs, states and outputs). Constraints involved in the optimisation algorithm are saturation of actuators and actuator rate limits (see Table 4.2).

$$A = \begin{bmatrix} -0.9825 & 0.0000 & -0.0007 & -0.0161 \\ 1.0000 & 0 & 0 & 0 \\ -2.1937 & -9.7758 & -0.0325 & 0.0743 \\ 77.3570 & -0.7675 & -0.2264 & -0.6684 \end{bmatrix}$$

$$B = \begin{bmatrix} -2.4379 & 0.2912 & 0.2912 \\ 0 & 0 & \\ 0.1837 & 9.8100 & \\ -6.4785 & 0 & \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & -79.8667 & -0.0283 & 0.9996 \\ 0 & 0.0000 & 0.9996 & 0.0290 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Table 4.1: The aircraft linear model

Limits	Constrained Variable Name	Unit
$-0.436 \leq \delta_E \leq 0.174$	elevator deflection saturation	rad
$-0.261 \leq \dot{\delta_E} \leq 0.261$	elevator deflection rate limits	rad/s
$0.009 \leq \delta_{Th} \leq 0.174$	engines throttle limits	-
$-0.028 \leq \dot{\delta_{Th}} \leq 0.028$	engine throttle slew rates	-

Table 4.2: Longitudinal channel actuator constraints

The next step involves the definition of the plant within the corresponding block as a nonlinear model. This assumes that the plant is known and that we are able to implement

it via SIMULINK standard blocks or a *S-function*. Both linear and nonlinear models can be implemented at the designer's choice. In the case of the nonlinear implementation we have to make sure that the assumption of measurable states is not violated so that, we are able to provide the $MBPC$ block with these signals.

Before starting a simulation it is as well to check the simulation parameters of the standard SIMULINK pull-down menu. The maximum step of the integration method has to be an adequate sub multiple of the sampling time at which the $MBPC$ controller operates. In our case the *Min Step Size* $= 0.001 \times T_s$, *Max Step Size* $= 0.25 \times T_s$ and *Tolerance* $= 1 \times 10^{-3}$.

Sampling time, horizons, type of optimisation	[1,4,10,0.5,1]
Input and rate constraints	$\begin{bmatrix} -0.436 & +0.009 & +0.009 \\ +0.174 & +0.174 & +0.174 \\ -0.261 & -0.028 & -0.028 \\ +0.261 & +0.028 & +0.028 \end{bmatrix}$
States constraints	$\begin{bmatrix} -1e3 & -1e3 & -1e3 & -1e3 & -2 & -40 & 55 \\ 1e3 & 1e3 & 1e3 & 1e3 & 2 & 40 & 120 \end{bmatrix}$
Input weighting matrix	[3e5 6e5 6e5]'
Output weighting matrix	[1 1 1]'
Precision in observing the sample hit	1e-10
Offset	2e-8

Table 4.3: Aircraft model $MBPC$ tuning parameters

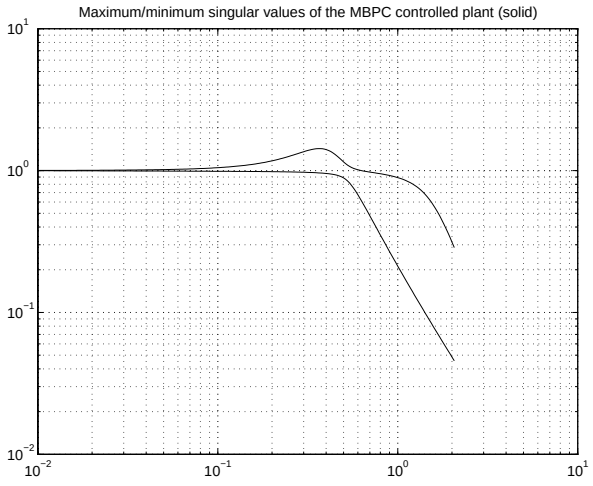


Figure 4.6: The frequency plot facility output

For the unconstrained case and the linear plant model we can plot the frequency response of the closed loop transfer function of the minimum and maximum singular

values (see Figure 4.6).

When running the simulation it is important to make sure that the precision required in detecting the sampling hit, the time spent in the on-line optimisation procedure and the offset time parameter present in most of the discrete time blocks of the $MBPC$ controller are consistent. The sample time variable sets the sample time at which the discrete block is updated. It possible to specify an offset time that will give information about which blocks should be updated sooner or later then others (see [Mat95]). For our example $T_s = 0.1$ s and the offset is 1×10^{-8}. The amount of computation time for the quadratic programming solver block is varying from one internal model to another depending on the horizons used. Therefore, selecting the offset is a task that cannot be provided as an automatic feature of the Development Space due to the rules of thumb involved.

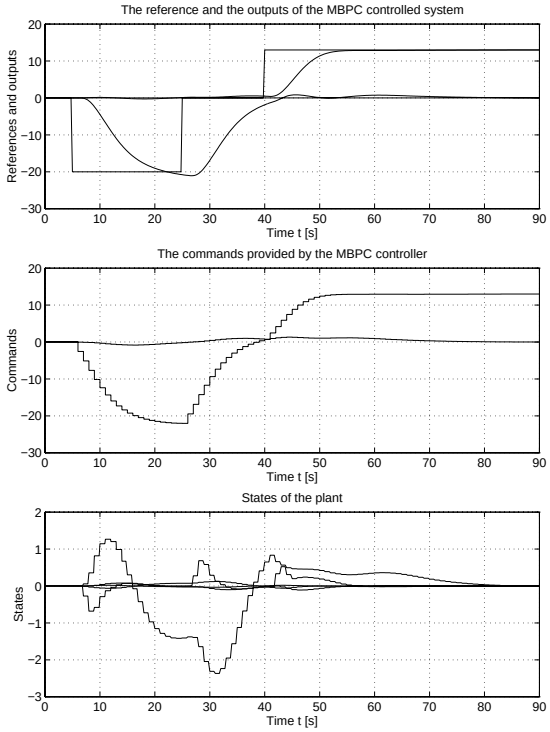


Figure 4.7: The plotted simulation results

Once we have decided the initial tuning parameters we can proceed to time simulations. For the first stages of the tuning procedure it is recommended to have a short simulation time (six times the maximum time constant of the plant) and at the beginning to start in the unconstrained case and then to include the constraints. At this stage a trial and error process complemented by the designer's knowledge has to be performed in order to achieve the required performance for the controlled plant. For our example, the values within the dialog window for tuning parameters chosen finally are given in Table 4.3.

With these values the responses to step changes in vertical and air speed, shown in Figure 4.7, were obtained in the absence of disturbances. Any change of the design parameters will reflect instantaneously within the simulation being carried out.

Chapter 5

Stability Augmentation Using Predictive Control

5.1 Introduction

The world experience in fly-by-wire is now more than 20 years old but the complex problems encountered in designing and implementation of automatic pilots for civil or military aircraft makes the research in this area concentrated on providing alternatives for replacing controllers designed in a classical manner with new ones based on modern design techniques. The general objective of flight controllers is to improve the natural flying qualities of the aircraft, in particular stability and flight envelope protections. The stability augmentation system (SAS) improves the flying qualities, contributes to safety and provides stability in face of perturbations such as gust and engine failures. All these features are provided together with the advantage of a reduced crew workload.

In principle one control surface is sufficient to control as many modes as independent measurements are available. Of course increased stability characteristics can be obtained while degrees of freedom are available. Moreover the existence of protections for the system excursions out of the normal flight envelope can provide the pilot with the flexibility to react without hesitation in various situations. These reasons lead us to consider the constrained predictive control method as a potential tool in addressing SAS type of functions. We have decided to investigate this because constrained $MBPC$ is radically different from other control approaches currently employed for flight control. This difference is not only at the level of the design method used but the implementation as well because it involves an on line computation of the controller. Several studies of using predictive control in such applications have been reported, though only a minority of these have really addressed the on line computation problem [GE91, SPVvL91, BS93, SSD95, WB95].

This chapter investigates the dimension of the problems encountered during a straight replacement of aircraft SAS conventionally used controllers. A principal reason for developing this design was to demonstrate that $MBPC$ can be applied to a realistic complex benchmark problem and to demonstrate the limitations of this technique. Even further advances in computing hardware will bring the solution time down to acceptable levels for high-bandwidth systems. The $MBPC$ will give flight certification difficulties especially in

the constrained case.

The design method is illustrated by designing an autopilot for the Research Civil Aircraft Model (RCAM) used in the Group for Aeronautical Research and Technology in Europe (GARTEUR) design challenge [MBT97]. To enable comparisons with other GARTEUR designs a standard evaluation over a *Frankfurt descent procedure* was performed which tested the controller in conditions of mismodelling, gust disturbances and an engine failure. The main challenge was to retain the $MBPC$ benefits: its very nonlinear behaviour when the constraints are approached, its ability to anticipate pilot commands while providing low complexity for the overall controller.

5.2 The controller architecture for the RCAM problem

In this section the control system architecture for the overall system will be described. The controller structure we have adopted was a $MBPC$ controller (inner loop) in combination with a conventional controller (outer loop). The design consisted of separate design of the inner and outer loops for each of the channels. We have adopted this hybrid structure for several reasons part of them technical and part strategic. Previous design examples from the literature tackle the issue of $MBPC$ control by using such a controller as a single controller but not in conjunction with others. Therefore this design challenge was regarded as a good opportunity to check a different scheme and check its strong and weak points.

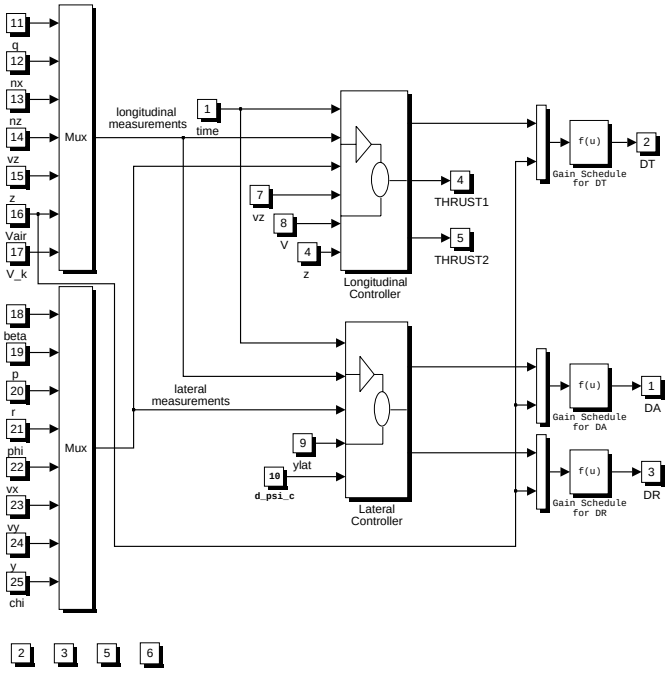


Figure 5.1: The aircraft overall controller structure

Two more problems should be mentioned here. Proceeding to a design that has a conventional inner loop controller and including actuator models the number of states that

should be measured or estimated in order to use $MBPC$ technique is increased. Moreover the challenge imposes restrictions about the available measurements (no actuator states or outputs are assumed to be measurable). Therefore, the inner $MBPC$ controller does not include the actuator models as part of its internal model. The second problem concerns the available linear model. The linearisation tool uses a 9 state aircraft nonlinear model that makes impossible the choice of the altitude as an output of the resulting linear system.

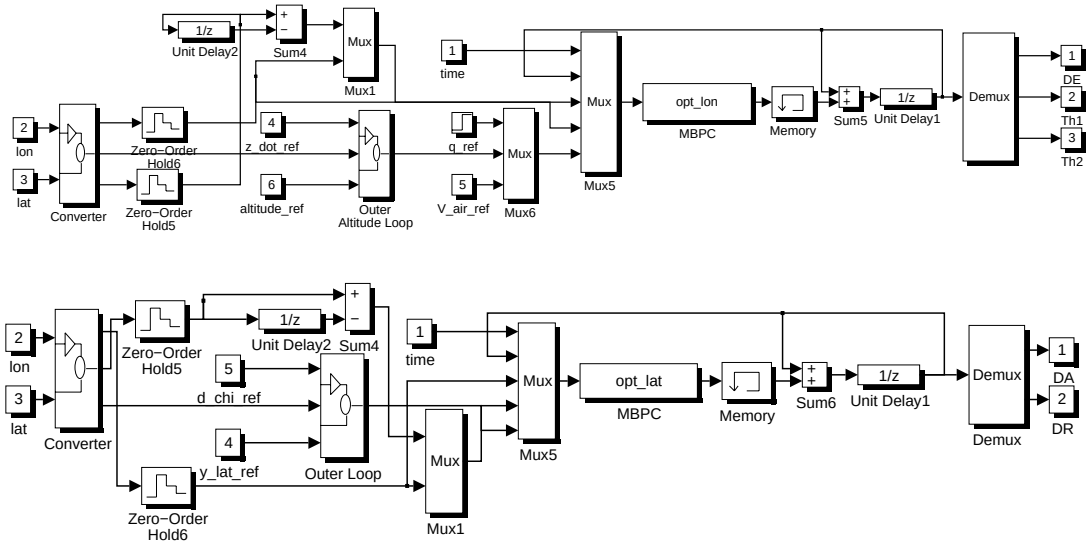


Figure 5.2: The aircraft longitudinal and lateral controller structure

Figure 5.1 shows the overall framework making use of separate controllers for each channel. Figure 5.2 shows more details of the solution adopted. The approach taken in this design is to split the aircraft controller into two parts: longitudinal and lateral. Each controller is divided into two loops, inner and outer. This approach was used to give the pilot the ability to intervene manually throughout the landing manoeuvre. The behaviour of both inner loop $MBPC$ controllers is that of Stability Augmentation System (SAS).

5.2.1 Models

The RCAM plant model, as described in the GARTEUR book[MBT97], is a full six degree of freedom nonlinear model generated as a Matlab MEX file by Dymola from an accurate mathematical model of an experimental transport aircraft.

The model used within the $MBPC$ algorithm to generate the control law is a linearised representation of the full six degrees of freedom nonlinear model. This model approximates the behaviour of the nonlinear model (see Figures 5.3, 5.4). The nonlinear model having the inputs, states and outputs presented in Table 5.2 was linearised around an operating point situated in the middle of the flight envelope, see Table 5.1.

Variable Name	Trim Value	Unit
mass	120000	Kg
centre of gravity	[0.23 0.0 0.0]	-
airspeed	80	m/s
flight path angle	0	rad
heading angle	0	rad
initial position	[0 0 -1000]	m
delay	0	sec
gust scale length	150	m
deviation gust	1.5	m/s
constant wind speed	[-10 0 0]	m/s

Table 5.1: The operating point trim parameters

The design uses only this linearisation in order to demonstrate the robustness of the controller for the whole flight envelope. Moreover the lack of actuator models within the internal model of $MBPC$ will show the ability of the control technique to cope with mismodelling problems.

The decoupling into longitudinal and lateral channels of the aircraft model was based on the author's experience regarding flight control systems for civil aircraft. The level of coupling between channels, analysed using the full nonlinear model, was observed to be small. The analysis was carried out by providing steps in the commands and checking the cross coupling between channels. The small level of coupling is a general characteristic for a commercial aircraft supporting the decision of splitting the aircraft dynamics in two distinct loops.

The linear internal models for the $MBPC$ control were obtained in two stages: first using the full nonlinear model a linear model was produced, then a selection of the corresponding model was employed for each channel based on the decoupling in between channels.

For the longitudinal channel the model is:

```
A_lon =   -0.9825    0.0000   -0.0007   -0.0161
           1.0000         0         0         0
          -2.1937   -9.7758   -0.0325    0.0743
          77.3570   -0.7675   -0.2264   -0.6684
B_lon =   -2.4379    0.2912    0.2912
                0         0         0
           0.1837    9.8100    9.8100
          -6.4785         0         0
C_lon =    1.0000         0         0         0
                0  -79.8667   -0.0283    0.9996
                0    0.0000    0.9996    0.0290
D_lon =    0     0     0
           0     0     0
           0     0     0
```

The eigen values of the A_{lon} matrix:

```
      real      imaginary   frequency      damping
(1)  -0.01139    -0.1264       0.126         0.0897
(2)  -0.01139     0.1264       0.126         0.0897
(3)  -0.83029    -1.1069       1.383         0.6000
(4)  -0.83029     1.1069       1.383         0.6000
```

In addition we provide a brief analysis of the dynamics of this model in terms of flight dynamics. Hence, the two modes that characterise the longitudinal channel are phugoid (1,2) and short period (3,4) (time responses are shown in Figure 5.3).

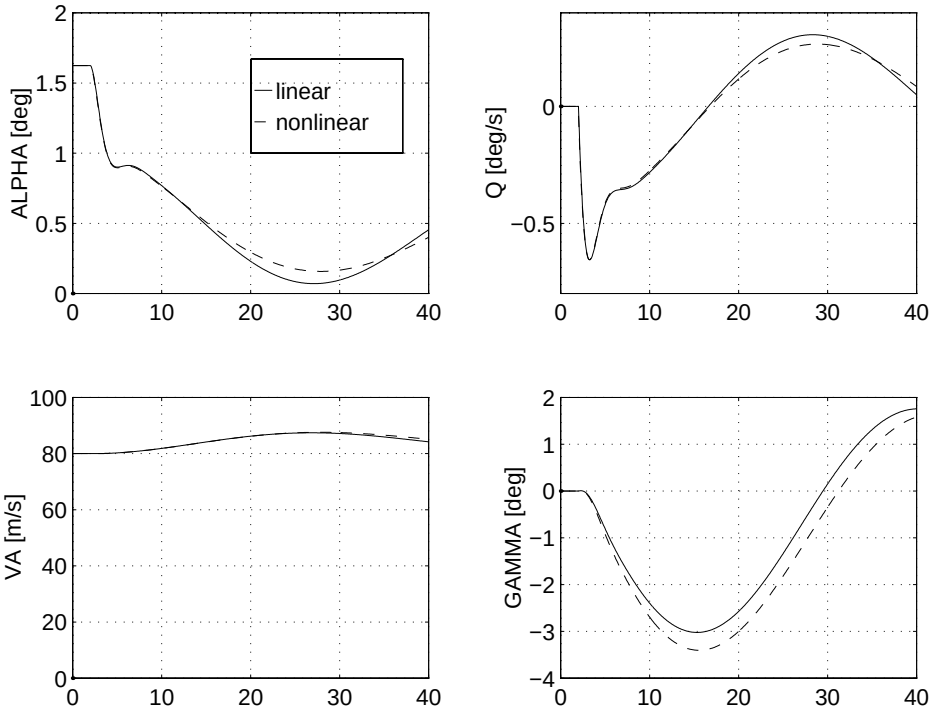


Figure 5.3: Time responses of the longitudinal channel linear model

Developing a similar analysis for the lateral channel we end up with the model:

```
A_lat =   -1.2667    0.5498    0.0000   -0.0242
           0.0522   -0.5207   -0.0000    0.0045
           1.0000    0.0283   -0.0000         0
           2.2667  -79.9679    9.7897   -0.1699
BS_lat =  -0.8402    0.2904
          -0.0176   -0.3325
                0         0
                0    2.0384
CS_lat =      0         0   -0.0000    0.0125
              0         0    1.0000         0
DS_lat =      0     0
              0     0
```

The eigen values of the A_{lat} matrix:

	real	imaginary	frequency	damping
(1)	-0.183	0.0000	0.1836	1.0000
(2)	-0.235	-0.5954	0.6404	0.3684
(3)	-0.235	0.5954	0.6404	0.3684
(4)	-1.301	0.0000	1.3017	1.0000

characterised by the following modes: lateral (4), spiral (1) and Dutch roll (2,3) having the corresponding time responses shown in Figure 5.4.

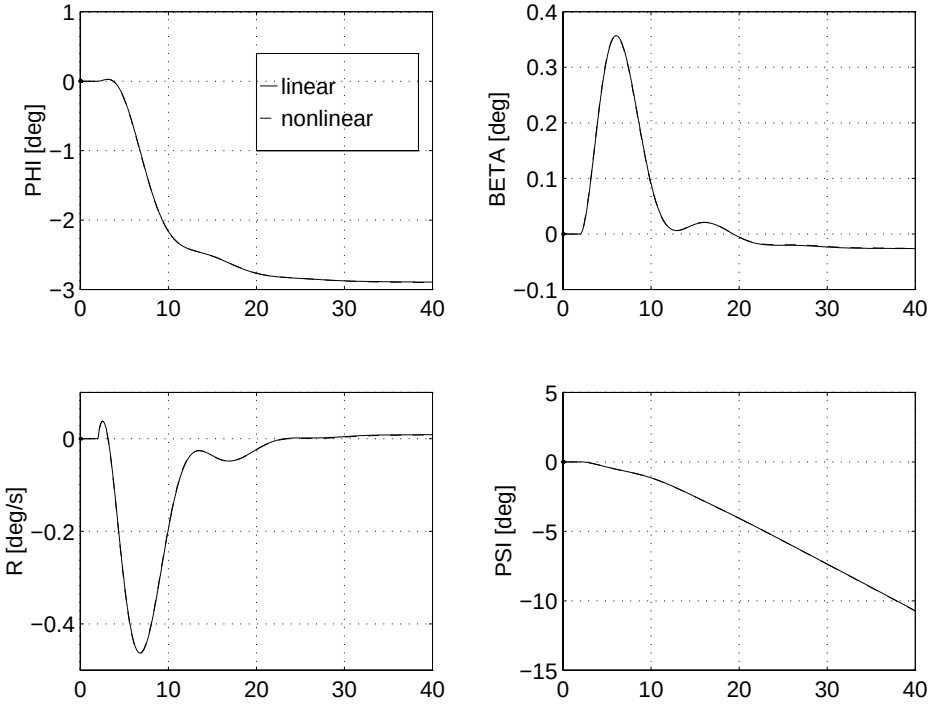


Figure 5.4: Time responses of the lateral channel linear model

As stated, in order to prove some of the features of the $MBPC$ algorithm, such as constraint handling and adaptability, we are not going to use an augmented model containing actuator or delay models. Therefore, instead of using the actuator models we prefer to specify their characteristics and bandwidth via constraints. This will keep low the dimension of the optimisation problem to be solved at each time step.

5.2.2 Actuator signals

Control effectors, well described in the RCAM design challenge [MBT97], are the following:

- Actuators used to control the longitudinal aircraft channel are the tail-plane (δ_T) and the engine commands (δ_{Th1} and δ_{Th2}). Constraints involved in the optimisation algorithm are saturation of actuators and actuator rate limits (see table 5.3):

Symbol	Input Variable Name	Unit
δ_A	aileron deflection	rad
δ_E	elevator deflection	rad
δ_R	rudder deflection	rad
δ_{Th_1}	engine 1 throttle position	-
δ_{Th_2}	engine 2 throttle position	-
Symbol	State Variable Name	Unit
p	roll rate	rad/s
q	pitch rate	rad/s
r	yaw rate	rad/s
ϕ	roll angle	rad
θ	pitch angle	rad
ψ	yaw angle	rad
u_B	velocity in body axis x direction	m/s
v_B	velocity in body axis y direction	m/s
w_B	velocity in body axis z direction	m/s
Symbol	Output Variable Name	Unit
q	pitch rate	rad/s
n_x	horizontal load factor	-
n_z	vertical load factor	-
w_V	velocity in vehicle carried system along z direction	m/s
z	height	m
V_a	calibrated air speed	m/s
V	total velocity	m/s
β	side-slip angle	rad
p	roll rate	rad/s
r	yaw rate	rad/s
ϕ	roll angle	rad
u_V	velocity in vehicle carried system along x direction	m/s
v_V	velocity in vehicle carried system along y direction	m/s
y	position in earth fixed frame	m
χ	flight path heading angle	rad
Symbol	Simulation Variable Name	Unit
ψ	yaw angle	rad
θ	pitch angle	rad
α	angle of attack	rad
γ	flight path angle	rad
x	position in earth fixed frame	m
n_y	lateral load factor	-

Table 5.2: Inputs, States and Outputs of the linear model

Limits	Constrained Variable Name	Unit
$-0.436 \leq \delta_T \leq 0.174$	saturation of tail-plane deflection	rad
$-0.261 \leq \dot{\delta_T} \leq 0.261$	rate limits of tail-plane deflection	rad/s
$0.009 \leq \delta_{Th} \leq 0.174$	engines throttle limits	rad
$-0.028 \leq \dot{\delta_{Th}} \leq 0.028$	engine throttle slew rates	rad/s

Table 5.3: Longitudinal channel actuator constraints

- The lateral channel is controlled by means of the rudder (δ_R) and ailerons (δ_A). Within the lateral channel controller constraints correspond to saturation of actuators and actuator rate limits (see table 5.4):

Limits	Constrained Variable Name	Unit
$-0.436 \leq \delta_A \leq 0.436$	saturation of aileron deflection	rad
$-0.436 \leq \dot{\delta_A} \leq 0.436$	rate limits of aileron deflection	rad/s
$-0.523 \leq \delta_R \leq 0.523$	saturation of rudder deflection	rad
$-0.436 \leq \dot{\delta_R} \leq 0.436$	rate limits of rudder deflection	rad/s

Table 5.4: Lateral channel actuator constraints

We have assumed an independent control of these channels by means of the respective actuators (see Figure 5.2). This is justified by the model analysis performed in the preceding Section 5.2.1 which provided us with information regarding the coupling level between the lateral and longitudinal channels.

5.2.3 Measurement signals

The choice of measurement signals depends on the quality of available signals. Sensor models are not provided because they are assumed to be perfect, which makes the usual choice of measurement signals difficult. Like other control techniques $MBPC$ works best when state measurements are available. Therefore the approach to RCAM design challenge assumed a state computation from the output measurements of the nonlinear model instead of state estimation. Such a computation is possible under the assumption that the model is perfect. This assumption will be reconsidered at the implementation stage. In our future research an estimation of states will be employed to provide a more realistic design.

The inner loop will stabilise and augment flying qualities. The outer loop controller in both longitudinal and lateral case is designed taking advantage of classic design techniques. This is a simple controller that ensures tracking of the altitude in the longitudinal channel and the heading rate in the lateral channel case. This was a reliable method to obtain a good outer controller in terms of performance and robustness.

In this chapter our aim is to present extensively the design of the $MBPC$ controller. The choice of measurement signals for this controller was made in accordance with the author's flight control design experience in the case of civil aircraft. The longitudinal inner controller uses pitch rate (q), air speed (V_a), vertical rate ($\dot{z}$) as feedback signals in the inner loop. Corresponding internal states of the model are pitch rate (q), pitch angle (θ), velocity in the Ox direction of body axes (u_B) and velocity in the Oz direction of body frame (w_B). The outer loop provides altitude tracking, and uses altitude as an output feedback signal.

The lateral inner controller employs side slip (β) and roll angle (ϕ). States used in feedback are: roll rate (p), yaw rate (r), roll angle (ϕ), and velocity along y axis in body frame (v_B). The outer lateral loop provides tracking for the heading angle rate ($\dot{\psi}$) and the horizontal flight path using its measured lateral deviation (y_{lat}).

5.2.4 Reference signals

During automatic flight control references for the controller are generated primarily based on time rather than on geometrical information. Therefore time is used as an independent variable for parameterising the trajectory. Effectively, this implies the possibility of a smooth transition from 3D to 4D scenarios. In the actual trajectory generator the time exists as an internal state. At each integration step the aircraft position is compared with the desired one, resulting in an "along the track error", a parameter of importance in 4D navigation. Once the aircraft moves away from the desired trajectory due to disturbances or poor controller performance the time error, simulation base time minus the aircraft time along the track computed will act as an extra reference signal.

The trajectory generator provides following outputs, mostly given in earth frame: x_E, y_E, z_E - desired position vector, u_E, v_E, w_E - desired velocity vector, V_E - modulus of total velocity, y_{lat} - lateral deviation from the horizontal flight path and $\dot{\psi}$ - heading angle rate. Other outputs refer to delay time resulting from deviations from the desired trajectory including engine failure or wind disturbances. All signals used as references are consistent despite a certain redundancy. In our design we use all these signals in order to ensure the fly by wire facilities as well to provide a possible command to the pilot by conventional means – commands given to the $MBPC$ controlled aircraft. For the longitudinal channel the $MBPC$ controller is provided with the following reference signals: pitch rate (q), air speed (V_a), vertical rate ($\dot{z}$). The lateral controller has following references provided: side slip (β) and roll angle (ϕ).

Most $MBPC$ schemes assume knowledge of the future set point. In this report we have employed $MBPC$ as a SAS for which no knowledge of the future set point sequence is assumed. The reference trajectory is generated assuming that current set-point (at time k) will be maintained over the whole prediction horizon ($r_i(k) = \ldots r_i(N_2)$). In our particular case the plant is a transport aircraft therefore such an assumption can be maintained because of the slow set-point variation from one discrete time step to another. The set-point is provided either by the pilot or by the outer loop controller. It is important to note that no terminal constraints are used during the optimisation.

5.3 Designing the $MBPC$ controllers

This section considers the requirements set up in method independent terms in the Garteur book [MBT97] and provide a procedure to approximate them by means of objectives compatible with the $MBPC$ design methodology.

In this section, following the procedure presented in Chapter 4, a description of the necessary actions that were taken for each iteration within the design cycle will be given. In order to produce the model required within $MBPC$ algorithm we linearised the nonlinear aircraft model excluding sensors, actuators and time delays around an operating point at the middle of the flight envelope. This model was linearised using equilibrium inputs outputs and states defined in Section 5.2.1. Then the problem was split in two separate channels: longitudinal and lateral. For each channel the inner loop was analysed and a controller designed. Various responses mentioned in the design challenge [MBT97] were checked using linear model as an intermediate analysis. The design for the outer loops was achieved using a linear time invariant representation of the $MBPC$ controller, namely the unconstrained case. The aircraft linear model was augmented with the LTI controller in order to produce the model required in the outer controller design. The next step was to perform a more extended analysis of the inner and outer loops using simulations involving the full nonlinear aircraft model. This helped us to get a fine tuning of the $MBPC$ controller, process that was done in few iterations. The first design was achieved for the longitudinal channel, this gave useful experience for designing the lateral one. Therefore the time involved in the inner lateral channel $MBPC$ controller design was shortened by a considerable amount. The fine tuning of this controller when full nonlinear model simulations were done took the same amount of time as in the longitudinal case.

5.3.1 Longitudinal channel

Using the strategy presented in the Section 4.1.1 we have to take into consideration requirements regarding: the flight path angle response (vertical rate response), the altitude response and the cross coupling between altitude and airspeed in order to tune the $MBPC$ parameters.

The choice of the control horizon was made by increasing it from a value of 1 (which correspond to stable internal models) to a value of 4 (the model order). This trades off reduction in control surface activity against computational complexity.

The tuning of the prediction horizon was made with respect to the settling time in the airspeed response for a step of 13 m/s in airspeed command. Disturbances can lead to constraint violations which can in turn lead to instability. We started with a prediction horizon $N_2 = 15$ and found that this value gave a certain degree of instability in the presence of vertical wind shear. Trading off the computational complexity against performance we have increased N_2 and finally set it to a length $N_2 = 18$.

At the beginning of the design procedure the sampling time for the $MBPC$ controller was set to $0.1s$ (a sampling frequency of $10Hz$). This was made because of the dramatic increase in the simulation time owing to the small time step in the integration method and due to the fact that a sampling frequency of $10Hz$ was observed to be at the bottom of the range used in civil aircraft industry (in this case the aircraft is assumed to be a rigid

body all the higher modes being treated as uncertainties).

The choice of the weighting matrices was made in order to track in a good manner the commands provided by the outer loop and to minimise the control effort. In accordance with the internal model dimensions $R = diag(\begin{bmatrix} 1 \times 10^5 & 7 \times 10^5 & 7 \times 10^5 \end{bmatrix})$ and $Q = diag(\begin{bmatrix} 1 & 1 & 1 \end{bmatrix})$.

The reference to be followed by the output was constructed from the set-point under the assumption of a priori unknown commands, a realistic expectation for the $MBPC$ used as a Stability Augmentation System (SAS).

5.3.2 Lateral channel

The lateral channel $MBPC$ controller was tuned mainly by translating RCAM design challenge requirements such as roll angle response, side-slip angle response, heading rate response and lateral deviation response into the tuning parameters specific for the method.

As in the longitudinal case we have fixed from the beginning a few of the parameters such as: $T_s = 0.1$, $N_u = 4$ and $Q = diag(\begin{bmatrix} 1 & 1 \end{bmatrix})$. The tuning involved the other parameters in a trial-and-error process. The prediction horizon was tuned first with respect to the settling time requirements for the side-slip response. The final choice was $N_2 = 18$. When tuning N_2, in parallel, the input weighting matrix was adjusted up to the final values of $R = diag(\begin{bmatrix} 20 & 60 \end{bmatrix})$. This choice provides the performance specifications of the roll and side-slip response to be consistent with the RCAM design requirements and reduces the lateral deviation error to zero quickly enough. We found that in order to meet the specification during the engine failure we had to increase the prediction horizon up to values of $N_2 = 22$. The reference to be followed by the predicted output was constructed in a similar way as in the longitudinal channel case.

5.4 Linear analysis

In order to perform the fine tuning of the $MBPC$ controller various responses were calculated and compared to the requirements in the RCAM design challenge manual.

5.4.1 Longitudinal channel

Flight path angle response

We subject the system to a step of 3° in commanded flight path angle. The response is shown in Figure 5.7. The tracking ensured is good (rise time of $5s$, settling time $15s$), the overshoot is small (less than 1 %), and the cross coupling to velocity is negligible (less than $0.4m/s$).

Velocity response

The system response to a step of $13m/s$ in commanded velocity is shown in Figure 5.5. Most of the requirements were satisfied. The rise time in air speed is near the specified $12s$ and the settling time is at about $45s$. The overshoot is 15 %, higher the the specified 5 %. The cross coupling to flight path angle is also a bit higher than the specified 0.5°.

As the second performance parameter the system response to a step change in airspeed of $13m/s$, caused by a wind shear, was calculated and plotted in Figure 5.6. The flight path angle was perturbed by approximately $4°$, which is higher than the specified value on cross coupling from air speed to path angle.

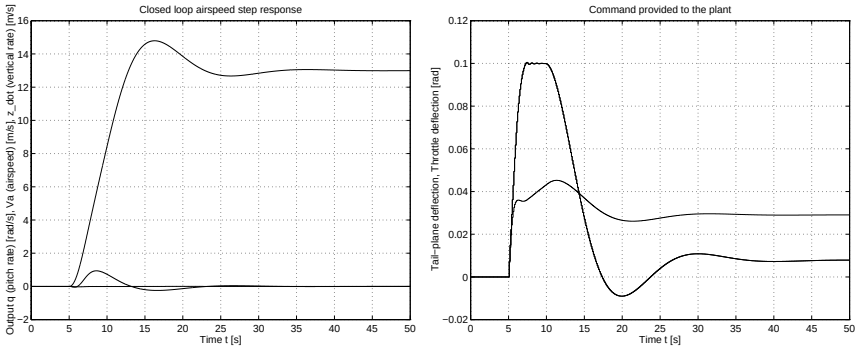


Figure 5.5: Linear system - Velocity response

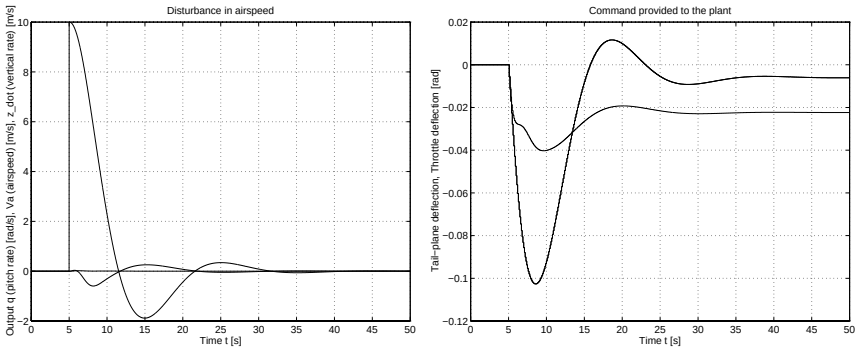


Figure 5.6: Linear system - Air speed response

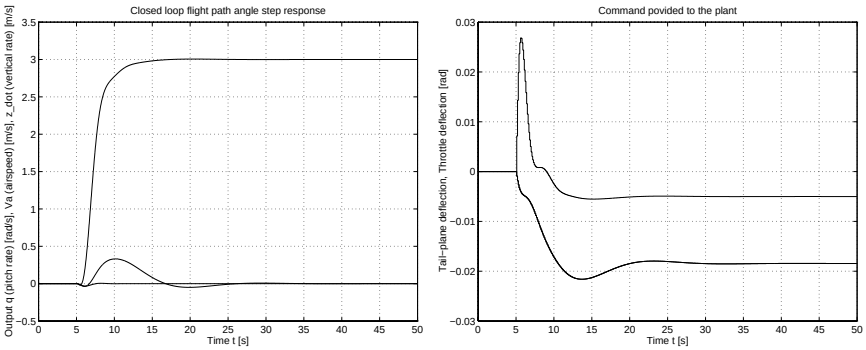


Figure 5.7: Linear system - Flight path angle response

5.4.2 Lateral channel

Roll angle response

The response of the system to $0.34rad$ ($20°$) step in the roll angle command is shown in Figure 5.8. The cross coupling from roll to side slip is specified to be less than $1°$. The tradeoff between a short rise time and a low overshoot or limit upon the aileron rate was solved by employing adequate values for horizons and weighting matrices. The procedure of choosing the values for horizons and weights is described in Section 5.3.2.

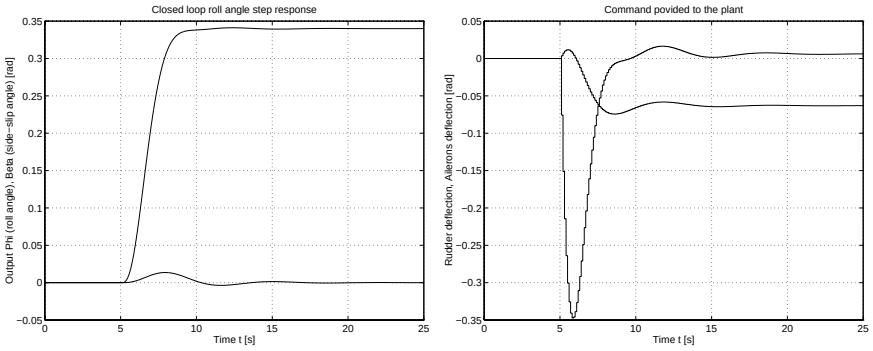


Figure 5.8: Linear system - Roll angle response

Side slip angle response

The response in side-slip command of $0.034rad$ ($2°$) is shown in Figure 5.9. Associated commands for ailerons and rudder are presented. The system response is not delayed significantly by introducing the $MBPC$ controller and the rise time is $4s$.

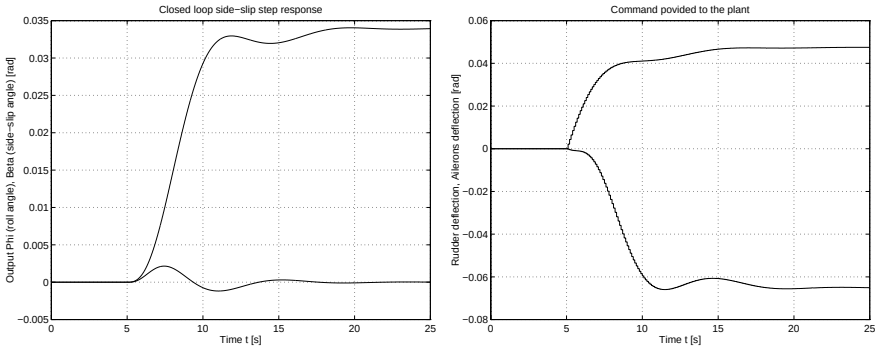


Figure 5.9: Linear system - Side-slip response

A summary of how the RCAM design requirements were met is given in the following Table 5.5:

Channel	Responses	Performance requested	Performance achieved	Decision
Longitudinal	flight path angle	$t_r < 5s$, $t_s < 20s$ $M_p < 5\%$	$t_r = 3s$, $t_s = 12s$, $M_p = 0\%$	+,+,+
	velocity	$t_r < 12s$, $t_s < 45s$, $M_p < 5\%$	$t_r = 12s$, $t_s = 35s$, $M_p = 15\%$	+,+,–
Lateral	roll angle (not specified)	$cross - coupling < 1^\circ$	$cross - coupling = 0.8^\circ$	+
	side-slip angle	$t_r < 5s$, $t_s < 20s$, $M_p < 5\%$	$t_r = 3s$, $t_s = 12s$, $M_p = 0\%$	+,+,+

Table 5.5: $MBPC$ controlling the linear plant – analysis results

5.5 Nonlinear analysis of the resulting $MBPC$ controller

The principal analysis method used in the case of $MBPC$ algorithms is time domain simulation. In this section all simulations have been performed using the nonlinear model. The purpose of this analysis is to make clear to what extent the controller satisfies the design objectives formulated in [MBT97].

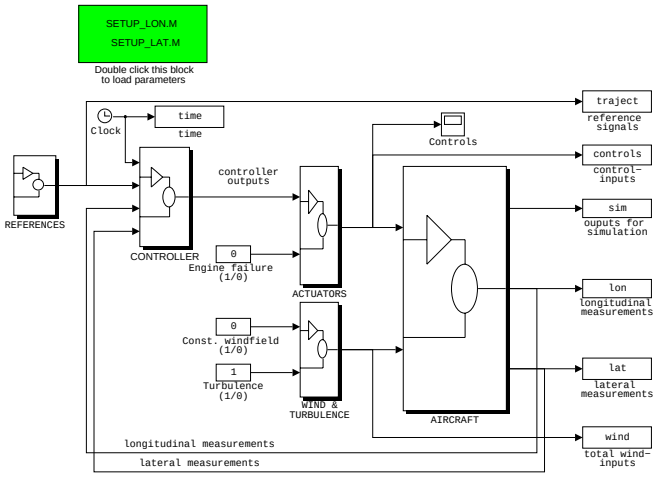


Figure 5.10: The Simulation Framework for the linear aircraft system

In order to perform synthesis and analysis in the $MBPC$ case, a particular test bed has been developed. The test framework involved is presented in Figure 5.10. This test bed is based on Development Space described in Chapter 4 providing, in the user interface, all facilities required for flight control design.

5.5.1 Longitudinal channel

Inner loop – Flight path angle response

Altitude tracking ensured by the longitudinal outer loop relies on performance characteristics of the flight path angle response (vertical rate response). Therefore, in the first graph of Figure 5.11 we show the behaviour of the nonlinear aircraft (longitudinal channel) when we subject the system to a step demand of 3°. The step occurred $10s$ after the simulation was started in order to avoid influence of simulation transients. The rise and settling time met the specifications ($t_r < 5s$ and $t_s < 20s$). A small overshoot can

be observed as a way to improve the rise time in order to ensure better altitude tracking performance. The altitude, as shown in the the second graph of Figure 5.11, is influenced by this constant rate of descent. Both engines are reduced by the $MBPC$ controller to the minimum thrust. Still this is not enough to maintain the airspeed steady state error at zero.

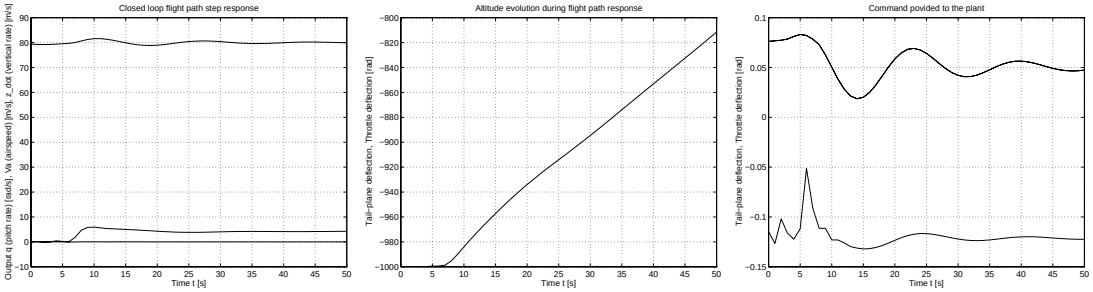


Figure 5.11: Nonlinear system - Vertical rate response

Inner loop – Airspeed response

The velocity response of the nonlinear aircraft model controlled by $MBPC$ is shown in Figure 5.12. The controlled system's speed was subject to a step of $13m/s$ (occurring at $5s$ after the start of the simulation). The rise time of 10 s and the settling time of $25s$ are within the RCAM manual specifications. The overshoot is less than 5 %. The control movement, shown in the second graph of Figure 5.12 has a satisfactory behaviour across the whole manoeuvre. From the first graph of Figure 5.12 it can be seen that no significant cross coupling to vertical rate was shown.

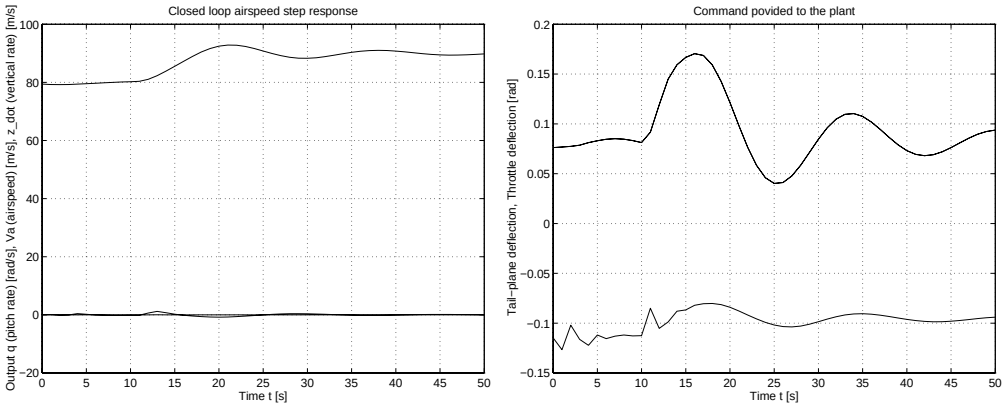


Figure 5.12: Nonlinear system - Speed response to a step demand of $13m/s$

The system response of the nonlinear system to a step change in airspeed of $13m/s$, caused by wind shear, is shown in Figure 5.13 (the step occurred at $5s$). The resulting response shows that the effect of such a constant disturbance, the wind shear, can be reduced in $15s$ with a small coupling to vertical rate. The settling time, which is within the

specification, is $35s$. During the wind shear rejection the control movement has a satisfactory behaviour. In the second graph of Figure 5.13 the law provided by the constrained $MBPC$ is presented in comparison with the actuator output.

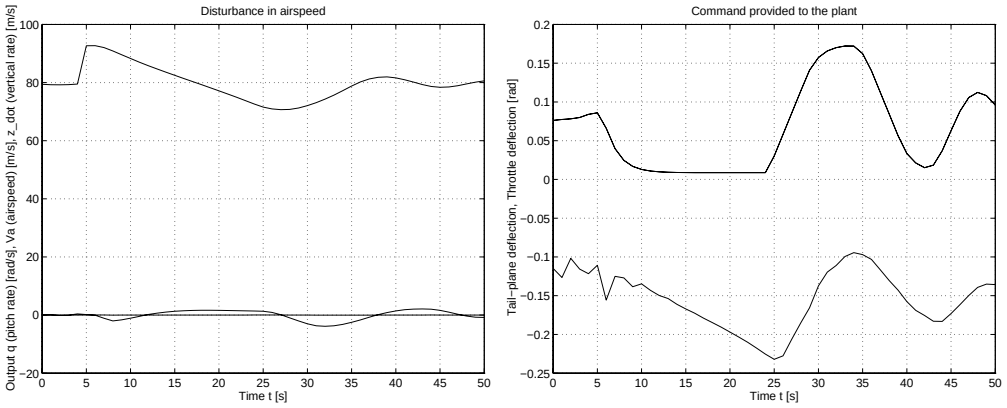


Figure 5.13: Nonlinear system - Step in airspeed

Outer loop – Altitude response

In this section, we present the results obtained in the case of altitude tracking employing the $MBPC$ controller as an inner loop controller and a conventional controller as an outer controller for a step of $30m$ in altitude demand. In this case the closed inner loop will be considered as a block with flight path angle and velocity commands as inputs. The outer loop used for tracking purposes provides us with the responses shown in Figure 5.14. Altitude response is shown in the first graph while the command movement in the second. The tracking ensured by the outer loop in conjunction with the $MBPC$ controlled inner loop is within the specifications (the rise time smaller than $12s$ and the settling time smaller than $45s$). The overshoot is bigger than the prescribed one of 5 % at altitudes above $305m$.

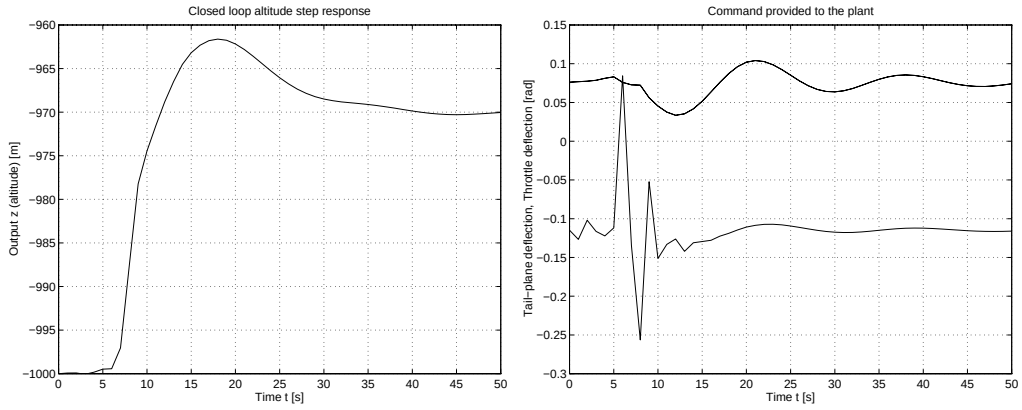


Figure 5.14: Nonlinear system - Altitude step response

Outer loop – Cross coupling between airspeed and altitude

When the velocity step response was simulated the altitude behaviour, shown in the second graph of Figure 5.15 met the design challenge specifications [MBT97]. These stated that the peak value of the transient of the absolute error between the aircraft altitude and its reference should be smaller than $10m$ (see second graph of the Figure 5.15). Conversely for an altitude step of $30m$ the peak value of the transient of the absolute error between the aircraft and the commanded airspeed is smaller than $1kt$ (see Figure 5.15, first graph).

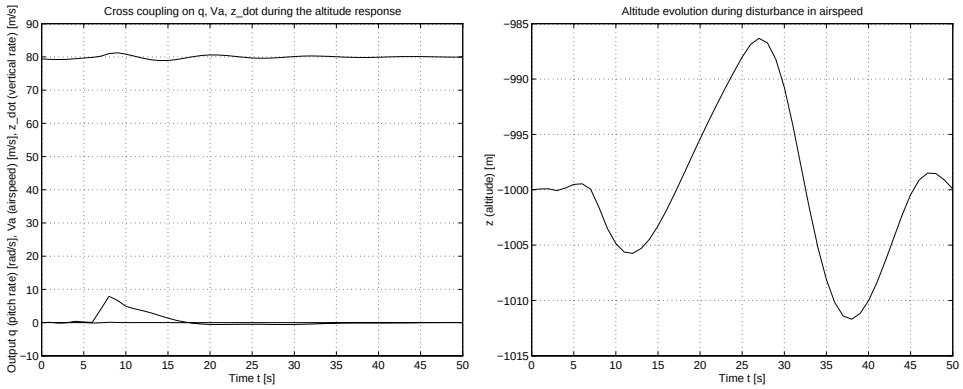


Figure 5.15: Nonlinear system - Cross coupling between airspeed and altitude

5.5.2 Lateral channel

Inner loop – Roll angle response

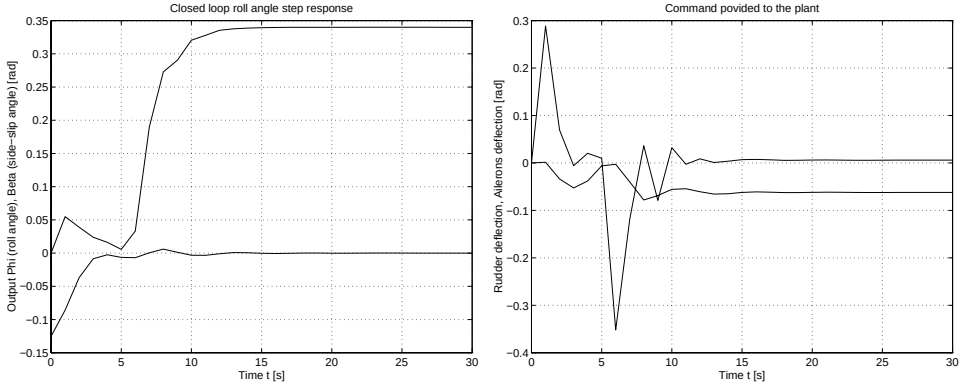


Figure 5.16: Nonlinear system - Step in roll

The roll angle response was analysed by providing the nonlinear system with a step of $0.34rad$ $(20°)$(see Figure 5.16). The rise time of $7s$ has proved to be a bit slow when performing a coordinated turn in the horizontal plane. Even though no overshoot can be seen and the settling time is $10s$ the response time of the inner loop has to be improved. During the step in roll angle the side slip angle β is small, having peak values below

$0.05rad$ (1°). In the case of engine failure all specifications were met, as can be seen in Figure 5.23. The specified turbulence conditions still keep the roll angle within bounds (less than 5°). The maximum roll angle is limited using the constrained $MBPC$ algorithm to 30° in order to comply with safety criteria.

Inner loop – Side slip angle response

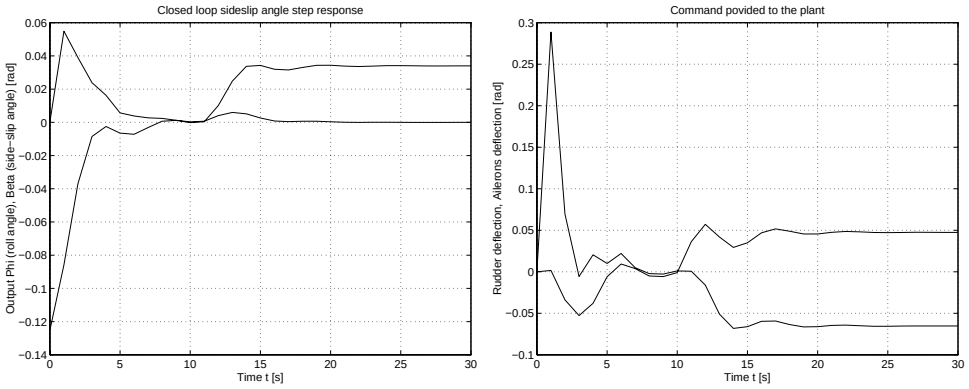


Figure 5.17: Nonlinear system - Step in side-slip

The nonlinear aircraft side slip angle control implemented by $MBPC$ is shown in Figure 5.17 when a step of $0.034rad$ (2°) was applied. The rise time of $4s$ and the settling time of $25s$ comply with the specs. The drawback is in terms of an overshoot that is bigger than 5%. Fortunately the side slip angle control involves small angles, so the overshoot does not cause any problems. Unfortunately there is no overshoot specified in the RCAM manual [MBT97]. The cross coupling to roll angle is less than 0.5°, well below the specified value of 1°.

Outer loop – Lateral deviation

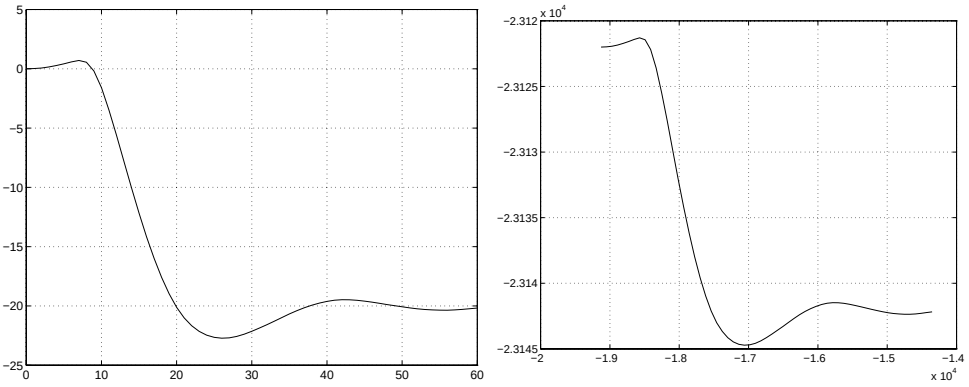


Figure 5.18: Nonlinear system – Response to a lateral deviation of $20m$

We subject the aircraft to a disturbance step of $20m$ in lateral deviation after $5s$. The response (see Figure 5.18, first graph) shows that the error from the desired track is reduced to 10 % in $30s$. The overshoot in the response to the step in lateral command signal at altitudes above $305m$ is bigger than the specification, which causes considerable errors when following a reference trajectory that has coordinated turns. There is no steady state error due to constant wind disturbances (Figure 5.24) and the lateral controller copes well with the specified turbulence conditions.

Outer loop – Heading rate

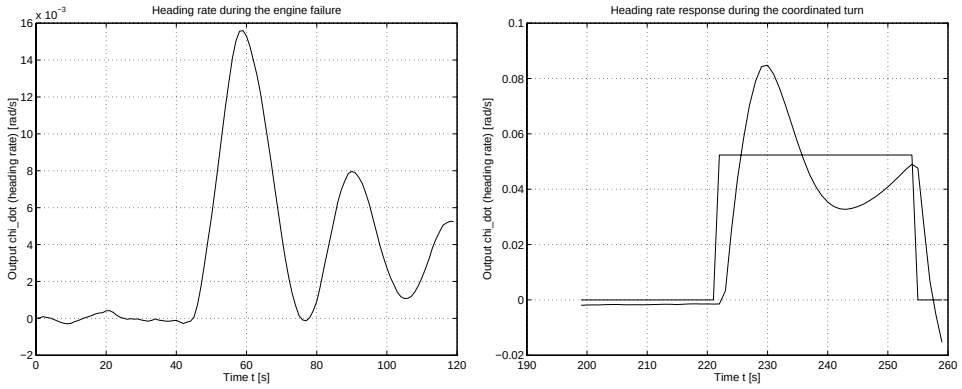


Figure 5.19: Nonlinear system – Response to a step in heading rate of $0.1rad/s$

As part of the RCAM design challenge requirements we minimise the heading rate in case of engine failure. The results are shown in the second graph of Figure 5.19. The heading rate error during lateral Dryden gust is minimised as well. The commanded heading rate is tracked well by the aircraft heading rate, as shown in Figure 5.19.

5.5.3 Other criteria

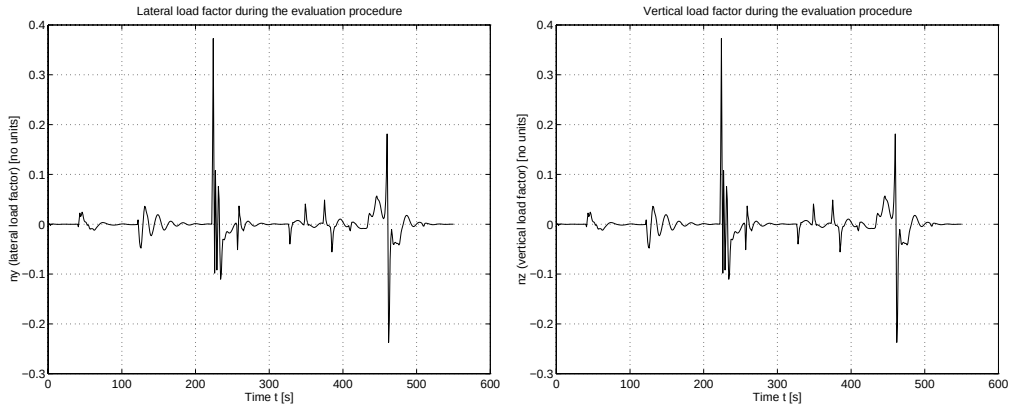


Figure 5.20: Ride quality criteria: lateral and vertical accelerations

As regards robustness analysis, the most significant model uncertainties are: centre of gravity variations, mass variations and time delays. As shown in Section 5.6.1, controller behaviour is generally satisfactory when the plant involves uncertainties.

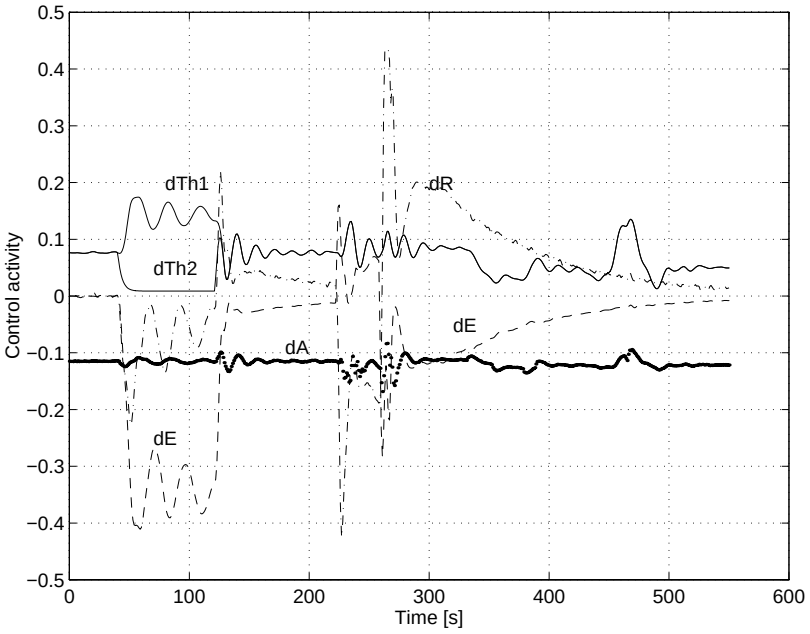


Figure 5.21: Actuators and engines control activity

Ride quality criteria such as maximum vertical acceleration, maximum lateral acceleration and damping were computed after the evaluation mission was completed, and are presented in table 6.5. Corresponding values across the evaluation procedure (the landing manoeuvre in the absence of gust) are shown in Figure 5.20.

All safety criteria specified in the manual [MBT97] were met either because of constraints imposed by $MBPC$ or because of saturation blocks included in the controller outer loops.

The use of $MBPC$ as an inner loop controller keeps the control activity small both for actuators and engines, as seen in Figure 5.21 where all units are common (*i.e.rad*).

5.6 Results of the automated evaluation procedure

This section presents the methodology-independent results of the controller designed in the previous sections. It is mostly based on the evaluation mission and scenario defined in [MBT97]: both 'overall tracking performance' and 'inner-loop behaviour' of the controlled system will be evaluated by means of bounds on key variables. The evaluation procedure is fully automated. No changes were made to any of the automatically generated figures or the table with numerical results.

First a general view of the results will be given in Section 5.6.1. Next, a more detailed discussion of the behaviour of the controlled aircraft will be based on the four separate

flight segments defined in [MBT97]. Finally, a summary of the numerical results of the evaluation will be given in Section 5.6.6.

5.6.1 A general view of the results

Before we go into detail on the specific properties of the controller with respect to aircraft behaviour, we will consider a view of the entire trajectory in the 3D plot given in Figure 5.22.

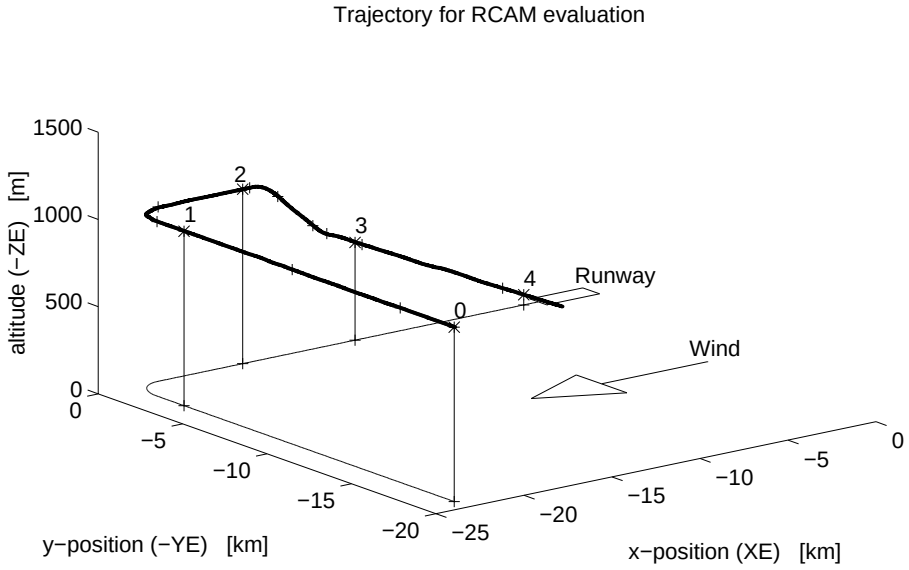


Figure 5.22: The trajectory response of the controlled RCAM model

In this plot, results are combined with respect to reference trajectory following, disturbance rejection and robustness. However, the scale is such that deviations from the trajectory should be hardly noticeable. A major robustness demand is taken into account by plotting the four trajectories resulting with nominal, most forward and most aft positions for the centre of gravity. The four trajectories can be seen and evaluated better in the following figures.

5.6.2 Segment I: the effect of engine failure

As the RCAM aircraft model is twin-engined, a single engine failure (between points a and b) will mainly result in lateral deviation. Hence Figure 5.23 provides a top view of the first trajectory segment. During the engine failure the dynamic response is acceptable (the roll angle does not exceed $5deg$) but the steady state deviation does not return to zero sufficiently quickly (less than $25s$ after the engine failure) as a result of our omitting a disturbance model. The overshoot after the engine reactivation is within bounds. Furthermore, the airspeed does not drop below $1.2 \times V_{\text{stall}}$, which is ensured by the $MBPC$ constrained control.

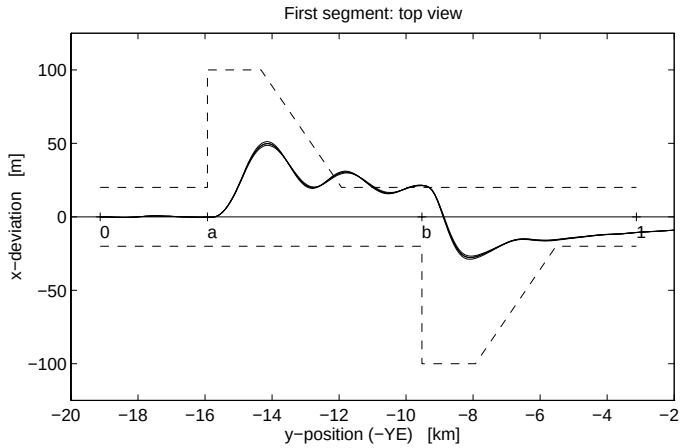


Figure 5.23: Segment I: the effect of engine failure

5.6.3 Segment II: the $3deg/s$ turn

At the moment the turn starts, the perfect following of the required trajectory and the desire to perform a coordinated turn would imply a sudden change in the aircraft's bank angle, which is only possible with an infinitely high roll rate. Obviously this is impossible, so that deviations from the desired trajectory at the start (and the finish) of the turn are unavoidable. To demonstrate this, Figure 5.24 – first plot gives a top view of the trajectory for all three centre of gravity locations plus bounds within which an acceptable performance should lie. Figure 5.24 – second plot provides a closer look at the actual lateral deviations: the bounds are equivalent with those in Figure 5.24 – first plot.

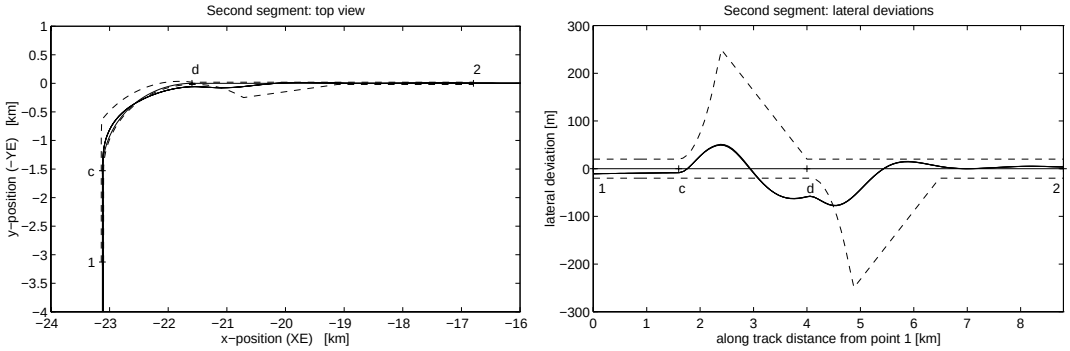


Figure 5.24: Segment II: the $3deg/s$ turn

5.6.4 Segment III: the capture of the $-6°$ and $-3°$ glide-slope

We start with a glide-slope of $-6°$; again it is unavoidable that the aircraft leaves the desired trajectory. It should return to the trajectory without overshoot and well within a period of $30s$. After that we go to a glide-slope of $-3°$ such that we get an inverse behaviour with respect to the desired trajectory, that should be about half the size of the

first response (if the system has a more or less linear behaviour). In the first plot of the Figure 5.25 the longitudinal response of the aircraft is plotted for three centre of gravity locations and with bounds that specify acceptable behaviour. The vertical deviations from the desired glide-slope are plotted in the second plot of the Figure 5.25.

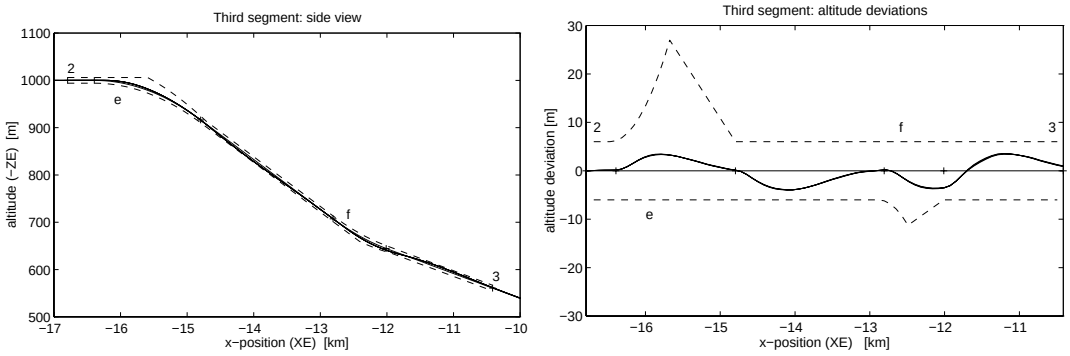


Figure 5.25: Segment III: the capture of the -6° and -3° glide-slope

5.6.5 Segment IV: the final approach with wind-shear

While on final approach with a glide-slope of -3° the effect of a wind-shear model is considered. Figure 5.26 – first plot shows the longitudinal response of the aircraft for three centre of gravity locations and with bounds that specify acceptable behaviour. The vertical deviations from the desired glide-slope are plotted in Figure 5.26 – second plot.

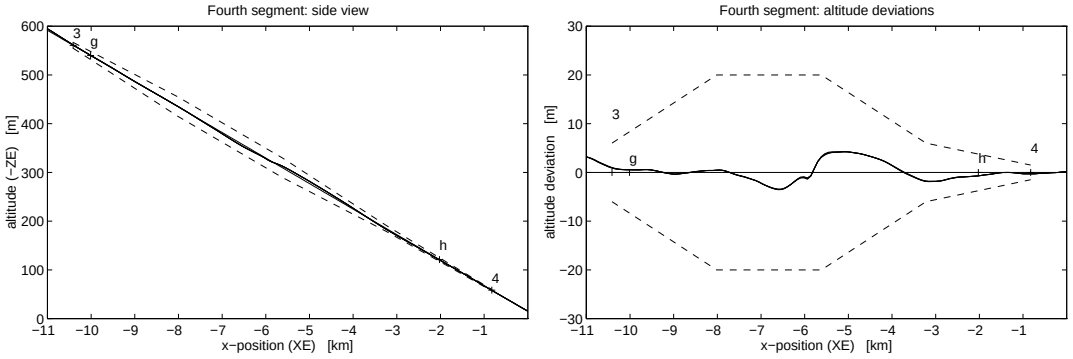


Figure 5.26: Segment IV: the final approach with wind-shear

5.6.6 Numerical results

Table 6.5 gives numerical results based on the discussed simulation results. For the motivation and calculation principle of the various results see [MBT97]. Each segment of the evaluation procedure it is considered via the design criteria: performance, robustness, comfort, safety and power. Because the evaluation criteria are independent of the type of controller used the table contains calculable indicators that enable us to obtain an objective comparison between completely different controllers. For each of the above items and

for each of the four trajectory segments a single number was calculated.

This number is an important indicator for several aspects of controller performance. In most cases a value smaller than one is acceptable. When saying most cases we would like to point out that in segment II a high roll rate, that is required to meet the given trajectory requirements, will decrease the comfort during this segment, which can be seen as a value greater than one of the corresponding indicator. Also, in segment IV, in order to comply with the vertical wind-shear disturbances, the passengers comfort is decreased (indicator > 1). Apart from these values greater than one of the comfort indicator, all other indicators show a good achievement of the design challenge requirements.

	Segment I	Segment II	Segment III	Segment IV	Total
Performance	0.5153	0.2792	0.1943	0.1138	0.2756
Robustness	0.0797	0.0675	0.0882	0.0601	0.0739
Comfort	0.6027	1.7584	0.6723	1.2348	1.0671
Safety	0.0081	0.0228	0.0100	0.0250	0.0165
Power	0.0055	0.0077	0.0151	0.0295	0.0145

Table 5.6: Numerical results of the evaluation procedure

5.7 Conclusions and lessons learned

The design of the automatic pilot, done for the final approach of an experimental fictitious aircraft (RCAM) is robust with respect to variation of several model parameters, time delays, nonlinearities and engine failures. As regards robustness analysis, in Section 5.6.1 we have shown that the controller behaviour is generally satisfactory with significant model uncertainties. This exercise also had the goal of evaluating and presenting the method's limitations. Ride quality criteria such as maximum vertical acceleration, maximum lateral acceleration and damping were computed after the evaluation mission was completed, and are presented in table 6.5. Apart from two values greater than one of the comfort indicator, all other indicators from table 6.5 show good achievement of the design challenge requirements.

We want to stress again the issue of robustness addressed during the design. The main achievement is that we use a $MBPC$ controller based on a single linear internal model – a state space realisation that does not contain actuators models. The behaviour of the controlled aircraft was proved to be good across the whole flight envelope. Moreover, the implementation of the $MBPC$ controller as a Stability Augmentation System (SAS) proved to be satisfactory in terms of performance achieved during various manoeuvres. Disturbance rejection of wind-shear and gust is good as well as tracking during engine failure or constant wind shear. Despite the non-conventional structure of $MBPC$, the resulting controller has standard interfaces to the evaluation software, and at a high level is "plug-compatible" with a conventional controller.

Another problem is that a constrained optimisation problem must be solved on-line, and therefore the computational complexity of implementing the controller is high. The main implementation difficulty is in reducing the computation time sufficiently to allow real-time operation. The latter problem is probably not insuperable, because progress can

be expected both in the efficiency of solution algorithms for $MBPC$ [CZ91], and in the power of the hardware on which they run.

It was possible to obtain a satisfactory controller despite a very simple model used as an internal model for the $MBPC$. But, some aspects of its performance, particularly robustness, were achieved in a trial-and-error manner, by post-design testing. In order to move towards more systematic design, from the robustness point of view, it is necessary to use a multi-model approach to ensure robustness against large perturbations (eg. engine failures).

Constrained Model Based Predictive Control $MBPC$ may not offer any advantages over more conventional control algorithms for routine flight control applications, apart from the constraint handling. As mentioned at the beginning of this chapter, however, we believe that it has good potential for higher-level control functions, such as on-board flight management, or reconfiguration of controllers in the event of damage to the aircraft structure or equipment.

In the next chapter we extend our research to the combined use of H_∞ loop-shaping and $MBPC$ as a method of designing automatic pilots for civil aircraft. An H_∞ loop-shaping controller in the inner loop will provide the stability augmentation and guidance functions; an $MBPC$ controller in the outer loop controller will act as a flight manager and overall supervisor [PHGM97].

Chapter 6

Flight Management Using Predictive Control

6.1 An introduction on flight management systems

The state of the art in flight management systems is represented by a set of computers that allow the pilot to program an entire flight from takeoff to landing and allows the aircraft to navigate directly between two points.

Among several approaches to flight management systems the one based on the total energy concept employed for a simplification of the aircraft to a point mass can be distinguished. In recent years this concept has been widely applied in flight performance optimisation and integrated control problems [WG94, Lam83b, Lam83a]. As a result vertical flight profiles for a specific range are optimised with a direct operating cost as an index function [WG94] or coordinated elevator and thrust commands are developed providing decoupled flight path and speed control for all the conventional features of the autopilot and autothrottle.

The development of modern control techniques allows aircraft control systems to be increasingly integrated. The flight management (FM) system is just a typical example for which integrating flight performance optimisation, thrust control together with guidance computation as well as navigation management will result in having the aircraft flying at its optimum performance with minimum cost fulfilling flight missions in an automatic manner.

As a result of the research, presented in the previous chapter, we have considered a flight management system as an alternative application for the $MBPC$ technique going beyond the stability augmentation system. We consider that at a high level the structure of the $MBPC$ controller, as well as the idea, are simple, but the transparency of the controller is reduced by the use of an on-line optimiser. Although the elements of the controller, namely the predictor and the optimiser, correlate clearly with the functionality of the control strategy the constrained optimisation makes a precise task allocation harder.

The next table summarises all the functions of the three system involved in our contribution: SAS, GS and FM.

The Automatic Flight System (AFS)

Flight Management System (FMS)

Inputs:

- Flight Plan Definition and/or Alternatives
- Departure and Arrival Procedures
- Time, Altitude and Speed at all waypoints
- Airline Cost Index

Functions:

- Flight Plan Optimisation (using a built-in data base)
  - optimum speeds
  - optimum and maximum altitudes
- Computations
  - flight conditions
  - estimated time of arrival
  - distance to destination
  - estimated fuel on board at destination
- Advisory Optimisations
  - fuel planning
  - optimum altitude and step climb/descend
- Aircraft Position Determination
- Aircraft Position Referenced to the Flight PLan
- Navigation Mode Selection
- Ground Speed and Wind Computations
- Optimum Radio and Inertial Sensor MIxing
- Provide Guidance Commands Along Flight Plan

Outputs:

- Heading Reference
- Lateral Deviation Error
- Speed Reference
- Altitude Reference
- Vertical Speed Reference
- Various Computation Results

Provides Long Term Control

Guidance System (GS)

Inputs:

- Short Term Guidance Information from the Flight Control Unit (FCU)
  - heading rate
  - track
  - speed
  - altitude
  - vertical speed
- Flight Guidance Information from the Flight Management System (FMS)
  - heading reference
  - lateral deviation error
  - speed reference
  - altitude reference
  - vertical speed reference

Functions:

- Lateral Navigation
  - turn coordination
  - yaw damping
  - rudder trim and limiting
  - automatic engine failure compensation
  - lateral deviation minimisation
  - constant wind disturbance rejection
- Longitudinal Navigation
  - flight envelope protection
  - speed control
  - altitude control
  - vertical speed control
  - thrust control
  - gust disturbance rejection
  - winshear disturbance rejection
- Updating Measurements using Radio/Inertial Navaids

Outputs:

References for the Flight Augmentation System (FAS)

Provides Short Term Control

Flight Augmentation System (FAS)

Inputs:

- Lateral Stabilisation
  - sideslip angle reference
  - roll angle reference
- Longitudinal Stabilisation
  - vertical speed reference
  - airspeed reference

Functions:

- Stabilisation of the Aircraft Along the Track
- Sideslip Angle Minimisation
- Actuator Effort Minimisation
- Engine Effort Minimization
- Ensure Good Ride Quality
- Robustification in Face of Modeling Errors
- Disturbance Rejection
- Ensure Low Channel Cross Coupling

Outputs:

- Lateral Channel
  - Rudder
  - Ailerons
  - Engines
- Longitudinal Channel
  - Tailplane
  - Canards
  - Airbrakes
  - Flaps
  - Flaperons
  - Engines

Provides Very Short Term Control

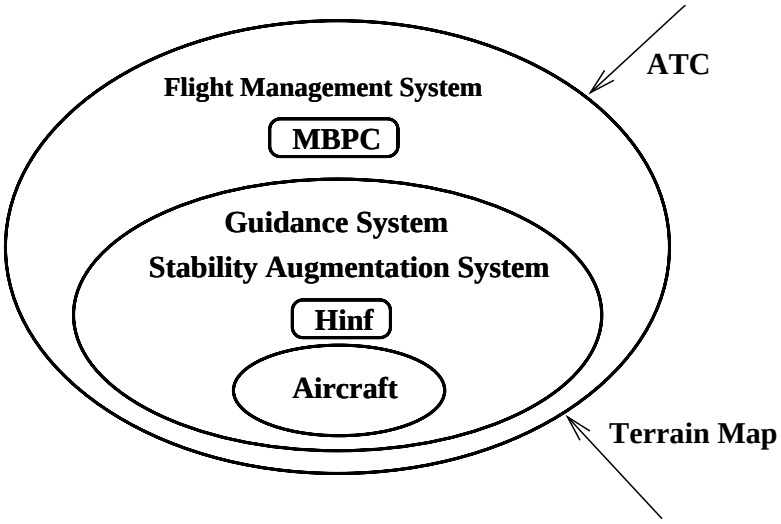


Figure 6.1: The autopilot architecture

Constrained $MBPC$ can remedy some of the drawbacks associated with fixed gain controllers:

- Disturbances of large amplitude on the plant output might saturate the actuators. This could lead to poor output decoupling and potential loss of stabilisability in the case of an unstable open loop plant.
- Complying with flight envelope constraints is not straightforward and requires *a priori* decisions.
- Pre-filters are usually chosen to give the fastest possible response without saturating the actuators during a typical pilot demand. This implies that small demands are achieved in the same time as large demands.

Here a novel architecture is presented and tested in the same framework as that of the RCAM model. This structure provides a robust controller, from the point of view of performance and stability, maintaining relatively low complexity and overcoming the negative aspects in the role considered so far. The next sections summarise our experiences and give some discussion of the potential of $MBPC$ for flight management.

6.2 A combined $MBPC/H_\infty$ autopilot for flight management guidance and stabilisation

A flight management system must optimise long term objectives such as passenger comfort and fuel consumption over *a priori* known "way-points" provided by air traffic control. Moreover, current requirements in flight management are increasingly concerned with 4D navigation – the flight path is time stamped and corrected with the along-track error as defined in Section 6.5. The use of $MBPC$ as a method to design a flight management

system seems desirable, as all the above goals can be expressed by the standard $MBPC$ problem formulation.

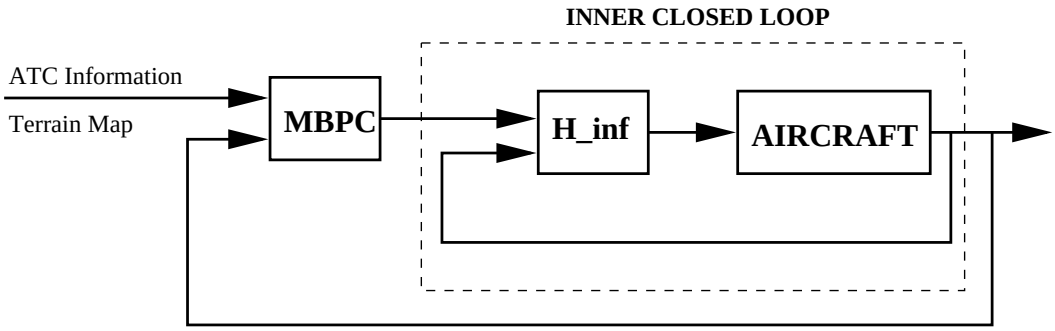


Figure 6.2: The autopilot architecture

Stability augmentation and guidance systems must be designed by a method that provides: disturbance rejection at both the input and output of the plant, noise rejection, robust stability, flexibility in specifying the bandwidth of the controller, tracking and a rich variety of analysis tools that will aid certification. For these reasons H_∞ loop-shaping was used to design the stability augmentation and guidance system. The structure of the H_∞ loop-shaping controller is shown in Figure 6.3.

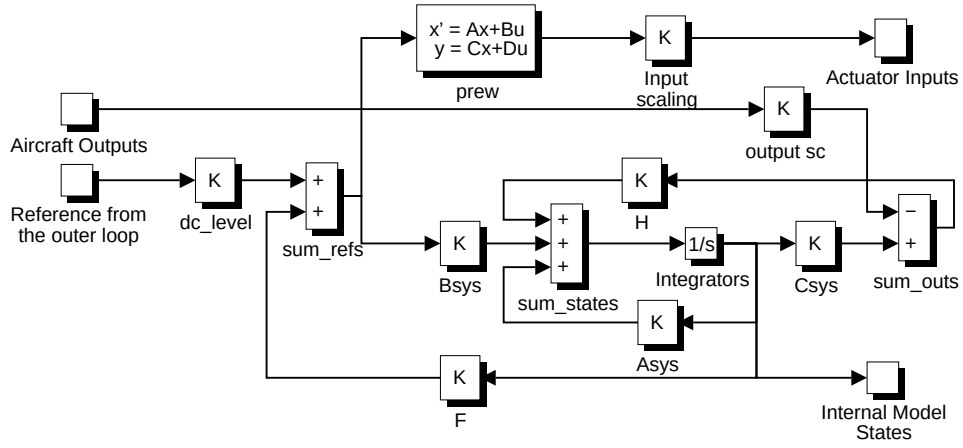


Figure 6.3: The inner H_∞ controller

The reader should consult the [PGS+97] for the design procedure employed. We do not claim that an H_∞ loop-shaping controller is the only suitable choice, but in the aerospace field, where it is appropriate to address the worst case signals, the goals concerning guidance and stability augmentation are well expressed by the standard H_∞ loop shaping formulation.

By monitoring the actuators and plant outputs and imposing constraints on their behaviour [Mor94, KRC92, RM93] $MBPC$ can help deal with the drawbacks associated with the fixed gain controllers. For example, the actuators can be used to their limits.

$MBPC$ will adjust the reference to the inner closed loop (H_∞ controller and RCAM aircraft) so as to avoid violation of the flight envelope and ensure good time response characteristics. All the above motivate the following structure shown in Figure 6.4, few explanations being necessary for a better understanding.

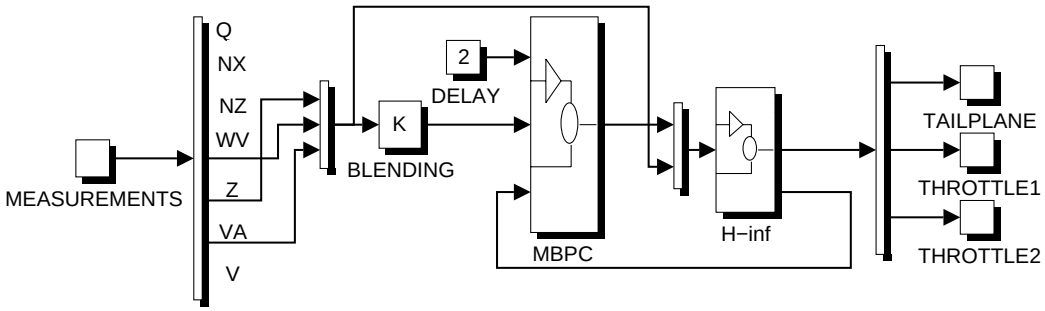


Figure 6.4: The combined $MBPC/H_\infty$ controller

The observer implementation of the H_∞ loop shaping controller, see Figure 6.3, has the well recognised advantages described in [PGS+97]. Among them we can mention that this implementation is equivalent to the stabilising feedback loop configuration shown in Figure 2.8. For this structure, as described in [Vin93], the disturbance rejection design problem is "decoupled" from the problem of achieving tracking of the reference signal with a nominal plant model.

The key result that we are going to use in our presentation is: $y(z) = N(z)c(z)$, see equation (2.41). Having a robust controller accomplishing the functions of the SAS and GS we can consider that we operate with the closed inner loop in the region where model uncertainty was accounted and as a result the robust performance limits offered by this controller are sufficiently good.

Hence the internal model of the $MBPC$ controller is going to be $N(z)$. The property mentioned in the foregoing paragraph represents a special advantage which enable us to produce the internal model without applying model reduction techniques as in [PHGM97]. Moreover, the transfer matrix $N(z)$ is stable and as a result well developed and understood $MBPC$ design procedures can be applied. An added feature is that for a certain class of aircrafts once the inner controller was designed the transfer matrices from reference to outputs will be similar for all the set members, complying with the MIL regulations. Therefore we can employ the same $MBPC$ controller, designed initial for one aircraft, to the whole class.

Between the states of $N(z)$ we find explicitly all the actuator states. In this case the actuator constraints are enforced using the method described in Section 4.3. The disadvantage of enforcing such constraints as state and not input constraints is that the enforcement is subject to the accuracy of the system modelling. This disadvantage is minimised by the existence of good actuator models.

The flight management system must not destabilise or cancel the effect of the inner-loop and must be robust to modelling uncertainty. When employing this novel architecture the $MBPC$ controller, see Figure 6.5, has to be robust to uncertainty within the inner

closed loop bandwidth. Hence the $MBPC$ controller can be less robust than the H_∞ controller as the uncertainty within the bandwidth is much smaller because it has been reduced by the H_∞ controller. It is our intention that the $MBPC$ controller, shown in Figure 6.5, provides an optimised reference to the inner closed loop without interfering with the stabilisation and guidance. Note that the $MBPC$ decision variable – $\psi(k+l)$, denoted by command in Figure 6.5 – depends on the inner loop design due to the choice of the $MBPC$ internal model. The outputs of the $MBPC$ controller (inputs to the inner-loop) are the blended altitude and vertical speed ($k_p z + k_i \dot{z}$) and the airspeed (V_a). The references are assumed to be provided from a data base provided by the air traffic control (ATC), possibly via Data Link. The estimated or measured states used within the $MBPC$ belong to the aircraft, actuators, H_∞ controller and the output mixer.

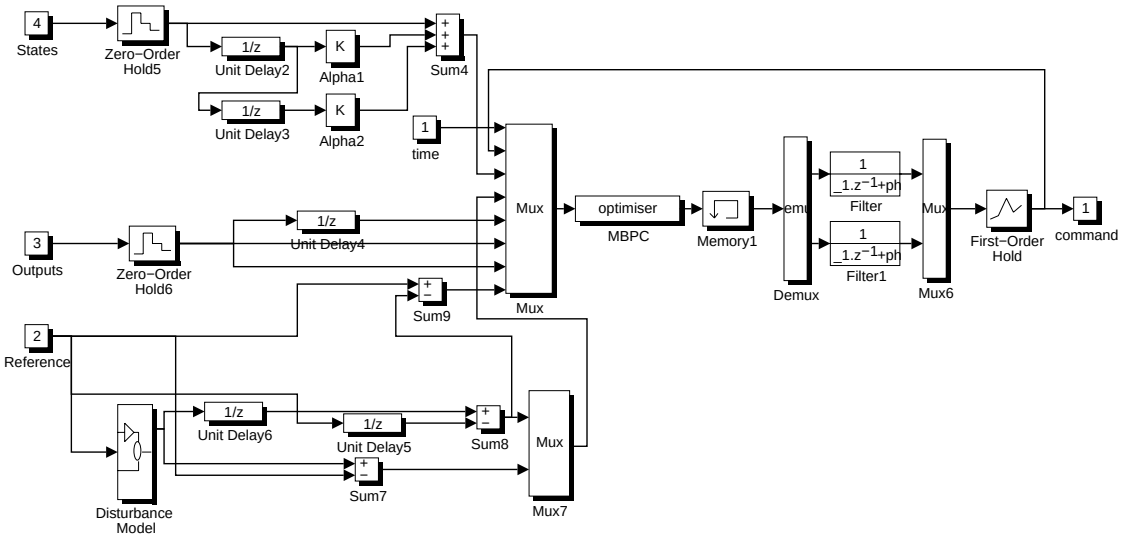


Figure 6.5: The outer $MBPC$ controller

The combined controller structure, shown in Figure 6.4, operates as a multi-rate system having a high sampling rate ($T_s = 0.01s$) for the H_∞ controller and a low one, $T_s = 1s$, for the $MBPC$ controller.

A single inner-loop controller, shown in Figure 6.3, was designed for the longitudinal channel covering both stability augmentation and guidance functions. Even though it is conventional to design an inner loop to provide stability and outer loop for tracking we consider that such an architecture has a number of advantages.

One of them is that the designer obtains insight in how robustness is traded off for good performance in altitude following. In the design actuator states and loop delays to simulate computational delays were also used. For further details of the H_∞ controller consult the design example in [PGS+97].

The single controller for both SAS and GS reduces the number of states of the H_∞ controller. This becomes very important when employing an $MBPC$ outer loop as the $MBPC$ uses an internal model of the closed loop for prediction. The higher the complexity

of the closed loop the longer the optimisation problem will take to solve. Note that the H_∞ loop-shaping controller K_∞ is placed in the forward loop with no pre-filters.

The flight management role of the $MBPC$ controller leads us to a way of generating the trajectory for the $MBPC$ off-line based on the generator provided within the conventional RCAM environment. This generator is not a part of our management system, because we have assumed that the trajectory information is coming from a data base given via Data Link by the air traffic control (ATC). Then the $MBPC$ controller will use this information on-line in order to produce the optimised references for the inner loop.

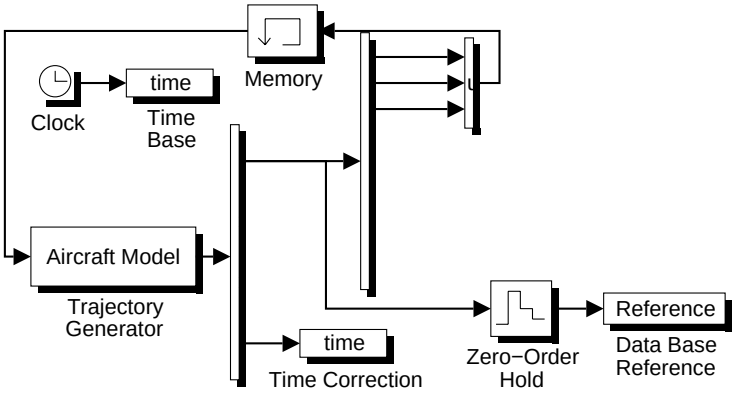


Figure 6.6: The flight manager $MBPC$ trajectory generator

The scheme shown in figure 6.6 was employed to produce the 4D reference trajectory, using the dynamical model of the aircraft sampled using the same sampling rate of the $MBPC$ controller. In practice this will be replaced by a system which interpolates the ATC way-points using the same aircraft dynamic model.

6.3 The flight management controller design procedure

6.3.1 The design cycle

1. Determine the requirements for the flight management, guidance and stabilisation systems behaviour

2. Design an H_∞ loop-shaping controller that will ensure disturbance rejection, noise rejection and robust stability as described in [PGS+97] by translating the pertinent requirements into dynamic pre and post compensators.

3. Choose the implementation and location of the H_∞ loop-shaping controller in the inner-loop. Produce a low order state-space model of the inner-loop that will serve as the $MBPC$ internal model.

4. Define the constraints related to inputs, rates of change of the inputs, outputs (flight envelope limits) and states (actuator limits). Construct matrices that represent these over the control and prediction horizons. Choose appropriate values for the $MBPC$ tuning parameters: control and prediction horizons and the cost function weighting

matrices in order to meet safety, comfort and overall flight management system performance requirements.

5. Tune the closed loop cost function parameters via closed loop linear and nonlinear simulations. This involves iteration of step 4.

6. Perform a stability and robustness analysis using the analytic expressions in the unconstrained case or time simulations in the constrained one.

6.3.2 From design criteria to $MBPC$ method dependent objectives

The design criteria for the outer $MBPC$ controller involve safety, comfort, control activity and performance of the overall system. These have to be translated into the choice of several $MBPC$ tuning parameters — control and prediction horizons, weighting matrices, sampling time and sometimes even constraint boundaries. The choice of these parameters is based on several theoretical results, as well as some rules of thumb, integrated together in a trial-and-error tuning procedure.

The safety criteria, reflecting the envelope safeguards, will provide the constraint limits upon the variables involved in the on-line optimisation. Table 6.1 gives the output constraints used and their physical interpretation.

Flight envelope constraints	Minimum value	Maximum value
Airspeed (m/s)	51.8×1.05	51.8×2.5
Vertical speed (m/s)	-30	$+30$
Altitude (m)	0	15000

Table 6.1: Flight envelope constraints as $MBPC$ constraints

We are able to impose constraints – as stated in Table 6.2 – on actuator rates and positions which are available as states of the aircraft model augmented with models of the actuators and the H_∞ controller.

Constrained Variable Name	Limits	Unit
Tail-plane Deflection	$-0.436 \leq \delta_T \leq +0.174$	rad
Tail Plane Deflection Rate	$-0.261 \leq \dot{\delta}_T \leq +0.261$	rad/s
Engines Throttle Limits	$+0.009 \leq \delta_{Th} \leq +0.174$	rad
Engine Throttle Slew Rates	$-0.028 \leq \dot{\delta}_{Th} \leq +0.028$	rad/s

Table 6.2: Actuator constraints as $MBPC$ constraints

The way to translate requirements upon comfort within $MBPC$ is by employing cost function weights that will give a suitable trade-off between tight following of a given trajectory and large loads on the aircraft structure.

The robustness of the combined $MBPC/H_\infty$ autopilot is obtained by appropriate design of the inner H_∞ loop shaping controller taking advantage of its main feature.

Now we clarify how the performance criteria of the $MBPC$ flight manager are translated into the available tuning parameters. In the following , where we use the same notation for the tuning parameters as in Chapter 4, we give all the technicalities of the design cycle step involving tuning of the $MBPC$ parameters.

The control N_u and prediction N_2 horizons

The influence of the control and prediction horizons is primarily on the performance of the controlled system, however they also have some influence upon robustness. As discussed, a smaller control horizon makes the controlled system more robust to uncertainties such as parameter variations. The choice of these horizons takes into account knowledge of the dynamics of the inner-loop. In case of the control horizon we perform a step response analysis of the system assuming a predefined sampling period. Our final choice for the control horizon was N_u=4 obtained after increasing it from the minimum value of 1 characteristic for stable systems.

The prediction horizon is derived from the settling time having in general a length greater than the system order. A small horizon will reduce the computational complexity, but in the case of non-minimum phase systems to be controlled using $MBPC$ it must contain at least the non-minimum phase behaviour. The robust performance and stability impose extra boundaries on the horizons.

For our cost function trading off computational complexity against robust stability we have increased the prediction horizon from 7 (the $MBPC$ internal model order) to N_2=20. The small control and prediction horizons ensured that the optimisation can be solved in real time ($0.9s$ on a Sun SPARCstation 20).

The sampling period T_s

The sampling period plays an important role in $MBPC$ controllers. A possible choice for this parameter is ten times smaller than the fastest settling time in the closed loop system (the value of T_s is obtained using linear time response analysis assuming constraints are inactive).

In order not to interface with the inner controller, the $MBPC$ control loop should have a lower bandwidth than the inner loop. This allows a big value for the sampling time T_s=$1s$, but does not require it. This makes possible a real time implementation of the controller. The simulation results with this value proved satisfactory.

The weighting matrices R and Q

The weighting matrices upon the outputs and control increments are important design parameters. Both give a measure of the tracking properties required from the closed loop system.

Since the references for the inner-loop have already been scaled, as part of the inner loop design, such that a unit change of each reference is equally significant, it is possible to

set R=*diag*($\begin{bmatrix}1 & 1\end{bmatrix}$), and avoid tuning this weight altogether. This leads to considerable simplification of the tuning procedure.

The tuning is an iterative process typically starting with Q=*diag*($\begin{bmatrix}1 & 1\end{bmatrix}$). The first step in the Q controller parameter design was to tune it in the absence of constraints. At first we tune using the time simulations involving the linear model of the plant. This step is followed by fine tuning, done by time simulations employing the full nonlinear model of the plant.

In order to improve the passenger comfort, which means that the control is less tight, we have to reduce Q. The H_∞ loop-shaping controller reduces the amount of uncertainty in the inner-loop, therefore we do not require so much robustness from the $MBPC$ controller. Hence this allows small values for Q. The final value of the output weighting matrix was Q=*diag*($\begin{bmatrix}0.007 & 0.02\end{bmatrix}$).

Once we have decided the initial tuning parameters we can proceed to time simulations. For the first stages of the tuning procedure it is recommended to have a short simulation time (six up to ten times the maximum time constant of the plant) and at the beginning to start in the unconstrained case and then to move towards the constraint one.

6.4 Analysis of the interaction between the inner and outer controllers

One of the main questions raised when a controller contains an inner and an outer loop is how they interact when ensuring disturbance rejection and if they destabilise or cancel their effects.

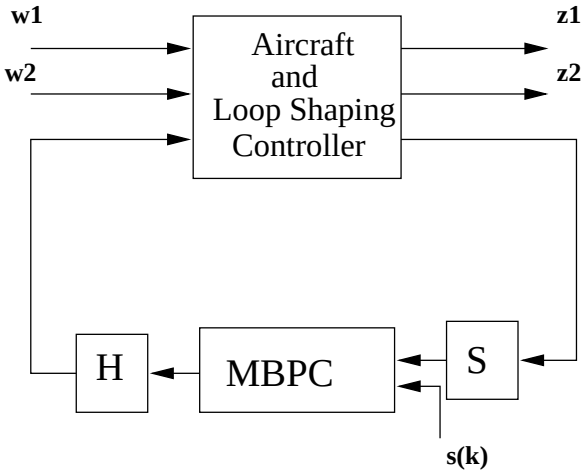


Figure 6.7: The generalised plant including the $MBPC$ introduced using sample and hold operators

The method chosen to analyse this type of behaviour is based on the computation of the norm $\|T_{w\to z}\|_\infty$, see Figure 6.7 which is the so called four block transfer matrix from input and output disturbances to the output of the plant and the controller, respectively. The value of this norm provides a level of robust stability to coprime factor uncertainty

which is a general type of uncertainty possible to use in aerospace applications due to similarities with conventional gain and phase margins.

As described in Section 6.3.2 there are two parameters that have a significant influence on the interaction of the inner and outer controllers. These are the weighting matrix R of the reference to the inner loop (the $MBPC$ output) and the sampling time T_s of the outer $MBPC$ controller. Moreover knowing the meaning of the $MBPC$ controller output (the reference to the inner loop) it is clear that imposing rate constraints will provide as well a way to decide the separation of the FM and GS+SAS controllers.

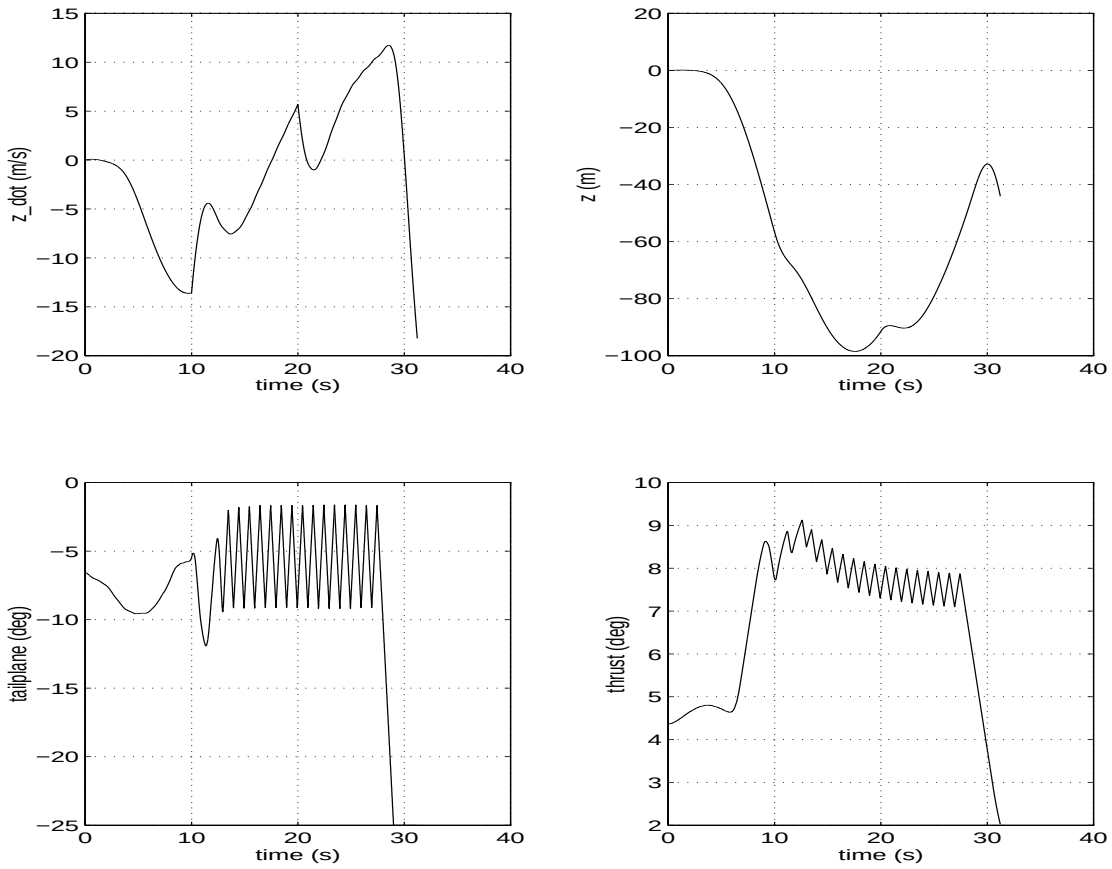


Figure 6.8: Responses when the inner and outer loop controllers are running at the same sampling time

Care was taken when computing this norm by regarding the system as hybrid and employing a sample data procedure developed and coded by the authors of [CG97]. The generalised plant necessary in this procedure was obtained by augmenting the inherent continuous time aircraft with the H_∞ loop-shaping controller which was designed in continuous time. All these operations were performed with the help of specific functions from the MATLAB Control Toolbox used for sub-systems interconnection. Then the discrete time $MBPC$ controller was wrapped around the plant and the norm $\|T_{w\to z}\|_\infty$ computed for various combinations of R and T_s. Results of this computation are shown in the

Table 6.3.

R \ T_s	0.5s	1s	1.5s
0.50	4.85	4.49	4.43
1	4.18	4.10	4.05
2	3.85	3.80	3.77

Table 6.3: Analysis results showing the interaction between the inner and outer controllers

The main conclusion is that by increasing the sampling time and enforcing small variations of the reference signal to the inner loop via a large R weight we can obtain a complete decoupling between the inner and outer controller, with a small deterioration of the $\|T_{w\to z}\|_\infty$ norm at the expense of an outer loop which will be less sensitive to the set-point changes.

In fact the combined autopilot has to be tuned in such a way to ensure the design criteria from the GARTEUR design challenge which made us to use the tuning $R = diag(\begin{bmatrix}1 & 1\end{bmatrix})$, $Q = diag(\begin{bmatrix}0.007 & 0.02\end{bmatrix})$ and $T_s = 1s$ for which $T_{w\to z} = 4.10$ comparative with the case without the $MBPC$ controller when $T_{w\to z} = 3.39$. This ensured a good ratio between the robust stability and performance for the closed loop plant.

Operating both inner and outer controllers at the same sampling rate while trying to reject a wind type of disturbance started in simulations at time $t = 10s$ we can observe how by having their functions overlapped the outer controller is driven into instability. As a result the reference to the inner loop exceeds by far the nominal values having as a result an unstable controller. This shows once more that having an inner stabilising loop and an outer $MBPC$ controller running at the same sampling rate might lead to instability.

6.5 Analysis of the longitudinal channel autopilot

At this stage a trial and error process complemented by the designer's knowledge have been performed in order to achieve the required performance for the controlled plant. The performance was tested with the full nonlinear model by providing steps in the reference signal of the commanded altitude and velocity.

The environment shown in Figure 6.9 provides functions and has features that give the user possibility to design and simulate the RCAM aircraft controllers. The main requirement is that the state measurements from the plant model are available if estimation of them is not used. The simulation framework, based on the Development Space presented in Chapter 4, was constructed using the MATLAB–SIMULINK in order to enable us to perform analysis and simulations with various $MBPC$ configurations.

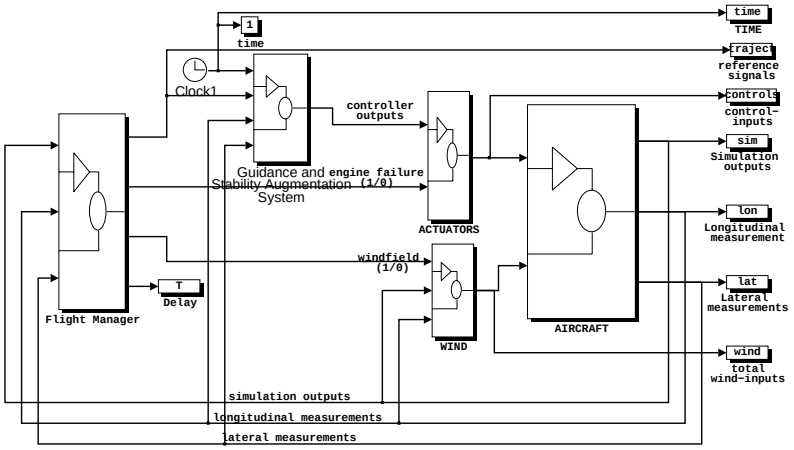


Figure 6.9: The simulation framework for the non-linear aircraft

In the altitude case the controlled aircraft was subjected to a step of $30m$ for various gravity centre positions, mass variations and time delays (as described in [MBT97]. The altitude response for various combinations of these, together with the cross couplings in air speed and corresponding actuators (the engine throttle and tail-plane) movements are shown in Figure 6.10. The step tracking is within the specifications. The biggest variation from the nominal case being $4m$. The cross coupling between altitude and airspeed is within bounds (eg. smaller than $1kt$). The control activity is limited within the constraints imposed for the $MBPC$ design.

Conversely, checking the airspeed response subject to a step of $13m/s$ we can conclude that the response is satisfactory in the face of the same uncertainties as in the previous case, see Figure 6.12. The trajectory following of the nonlinear aircraft was studied using time simulations for various configurations of the plant.

Channel	Responses	Performance achieved	Decision
Longitudinal	altitude	$t_r = 30s,\ t_s = 50s,\ M_p = 0\%$	+,–,+
	velocity	$t_r = 4s,\ t_s = 30s,\ M_p = 5\%$	+,+,+

Table 6.4: Analysis results achieved with the nonlinear longitudinal plant

As shown in Figures 6.10, 6.12 and Table 6.4 results were satisfactory. The use of the *a priori* information on the reference trajectory can be observed when the receding horizon mechanism brings it into the prediction horizon of the $MBPC$ controller.

In Figure 6.11 we depict time responses of the RCAM longitudinal channel full non-linear model. $MBPC$ was implemented with constraints placed upon the actuator deflections and rates and on the outputs as flight envelope restrictions.

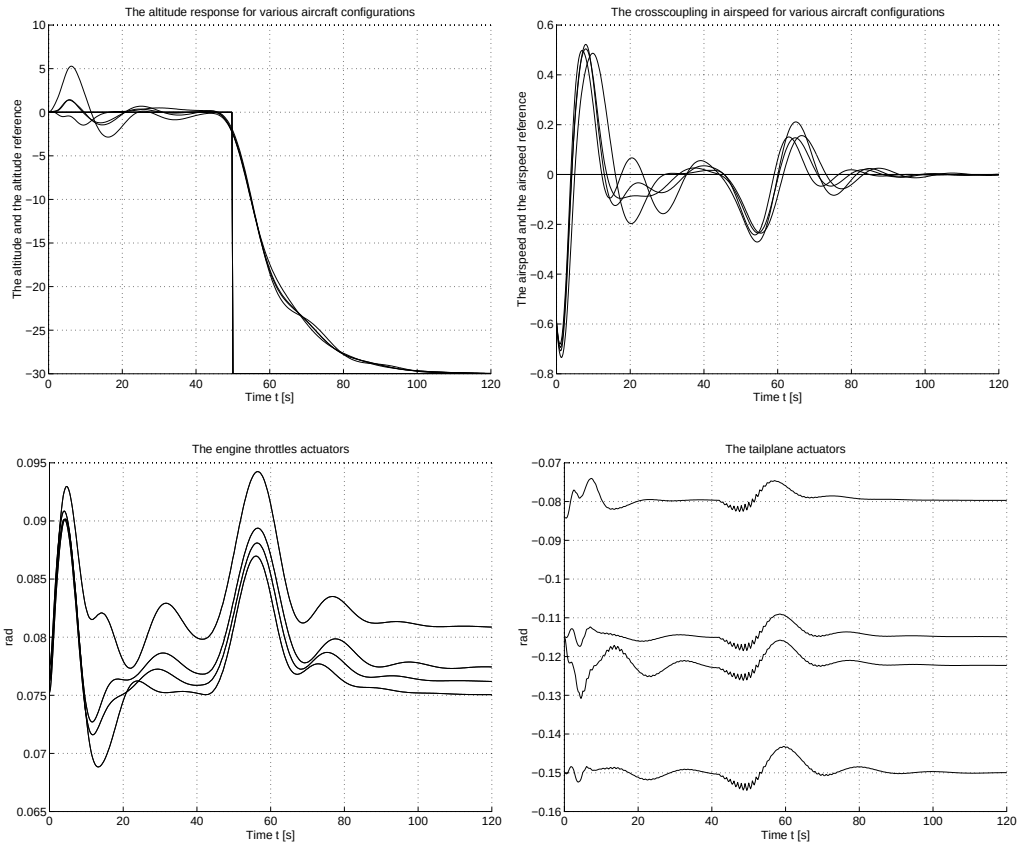


Figure 6.10: Results of the longitudinal analysis in the case of $MBPC$ as a Flight Management System (altitude $30m$ step response)

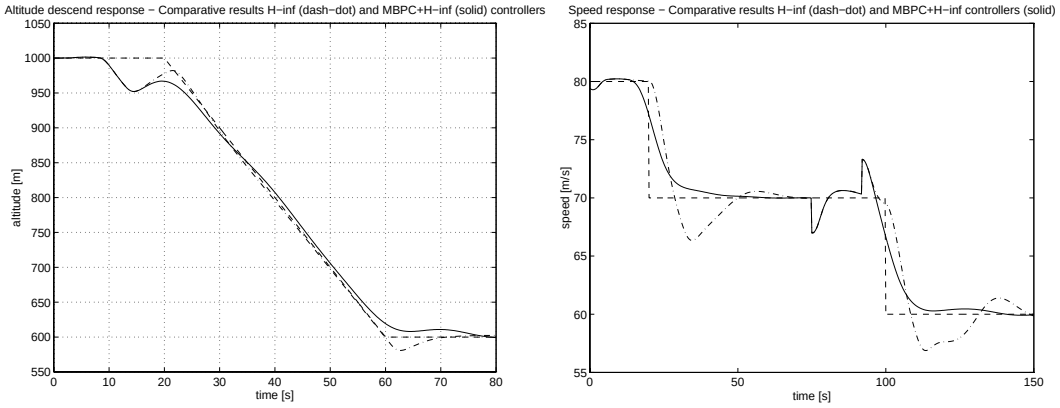


Figure 6.11: Altitude and airspeed response (subject to wind-shear) comparative results: the H_∞ controller (dash-dotted) and the $MBPC/H_\infty$ controller (solid)

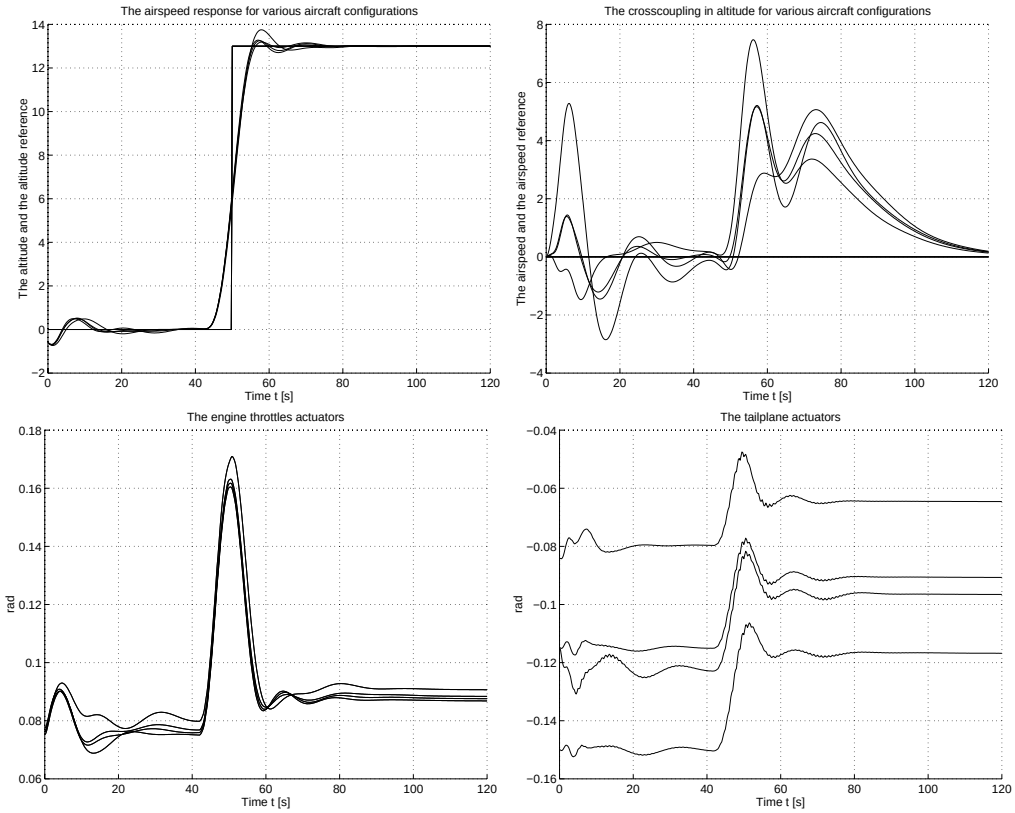


Figure 6.12: Results of the longitudinal analysis in the case of $MBPC$ as a Flight Management System (airspeed $13m/s$ step response)

We have subjected the controlled aircraft to different scenarios, relevant for the longitudinal case (Figure 6.11). At $20s$ the RCAM goes into a descent at a rate of $10m/s$. At $10s$ there is a wind-shear of $10m/s$ for a $10s$ duration. The prediction horizon of the $MBPC$ is $N_2 = 10s$. The H_∞ controller by itself tries to recover from the disturbance as fast as possible and return to the original altitude of $1000m$. The combined $MBPC/H_\infty$ structure, though, takes into account that the aircraft is going to start descending at $20s$ and does not try to reach the original set-point hence improving passenger comfort. In effect the $MBPC$ controller is modifying the reference to the inner closed loop. At the bottom of the descent there is no overshoot for similar reasons.

The transients in the first $5s$ (Figure 6.11) are due to the non-linear model not being perfectly trimmed. The combined controller minimises the overshoot and follows the reference well within the design specs from [HM97c]. At $75s$ there is a head wind of $3m/s$ that lasts for $17s$. As the prediction horizon is $10s$ it can be deduced that the disturbance rejection capability of both controllers is the same in these circumstances.

In the case of tracking of a simulated landing path, in Figure 6.13 and Figure 6.15 we show the responses of the combined controller for two tuning choices of the $MBPC$ controller in terms of the R matrix and the constraint enforcement. In both cases we consider *a priori* knowledge about the RCAM model landing trajectory.

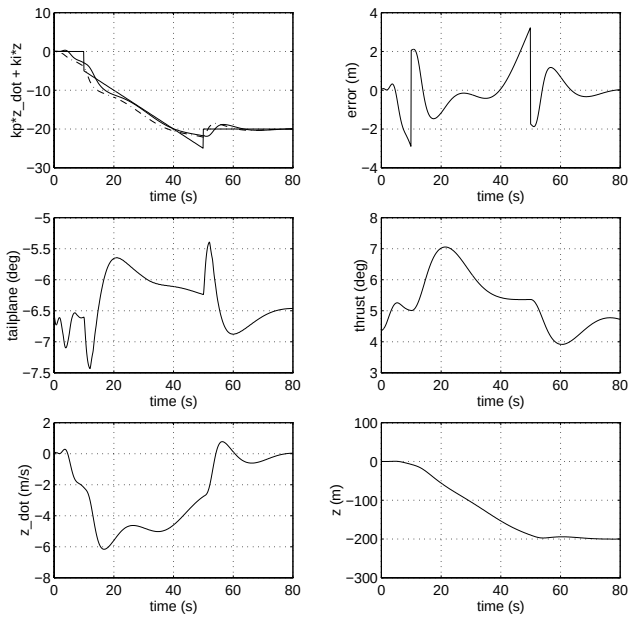


Figure 6.13: The tracking of a landing path using an $MBPC$ controller as a flight manager with mild tuning and no constraints upon the rate of change of the reference to the inner loop

The main purpose of showing these simulations is to demonstrate in the absence of wind disturbances and when the system is subjected to steps and ramps in the reference, which reflect the shape of the landing path, the behaviour of the closed loop. Here the ramp reference to be tracked is the blended variable $k_p\dot{z} + k_i z$ obtained using the vertical speed and altitude. Such a choice was required by the architecture of the combined guidance and stability flight controller. This feature was ensured using the approach from Section 4.3 which was coded within the Development Space software, see the Appendix.

In these simulations the prediction horizon was set to a value of $N_2 = 20$ steps ahead. Note that the inner H_∞ controller was not designed to account for ramp type of following and therefore this task was transfered to the outer controller. The $MBPC$ control algorithm used was the same as the one developed in the Section 4.3 providing robust tracking.

In Figure 6.14, for the same tuning of the controller as in the Figure 6.13, we show the responses when we have subjected the nonlinear aircraft to a wind disturbance of $+10m/s$ between times $t = 30s$ and $t = 40s$. The dash-dotted line represents the reference provided by the $MBPC$ controller to the inner closed loop. It can be observed a good separation between the tasks of the two controllers (the outer is used primarily for tracking and the inner for disturbance rejection).

The improvement in comfort, done at the expense of tracking, led us to consider the tuning of the second controller as appropriate from the perspective of overall design criteria. This controller provided small errors and minimised the control effort. The responses achieved are shown in Figure 6.15.

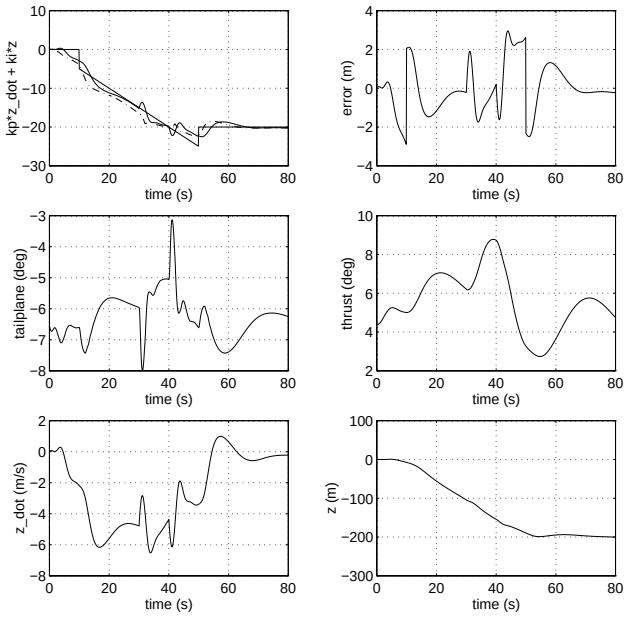


Figure 6.14: The tracking of a landing path in the presence of disturbance using an $MBPC$ controller as a flight manager having mild tuning ($R = 5500$) and no constraints on the reference to the inner loop

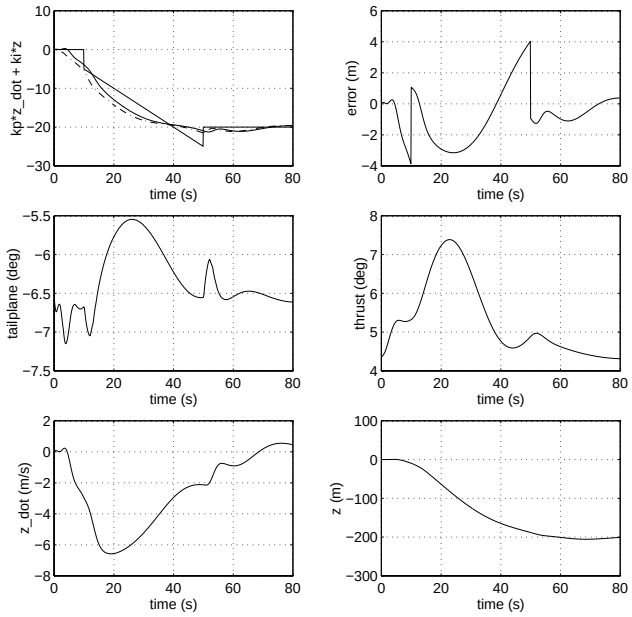


Figure 6.15: The tracking of a landing path using an $MBPC$ controller as a flight manager with tide tuning ($R = 7500$) and constraints upon the rate of change of the reference to the inner loop

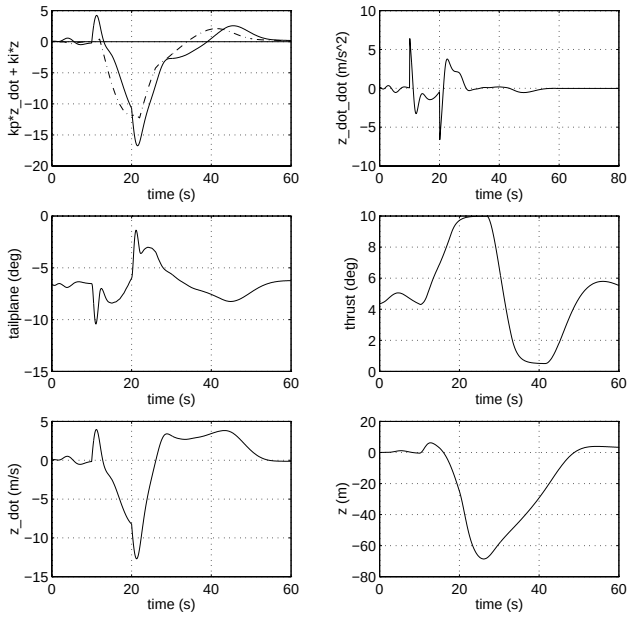


Figure 6.16: The evaluation of the disturbance rejection properties of the combined autopilot with an $MBPC$ having mild tuning ($R = 5500$) and no constraints on the reference to the inner loop but constraints on actuators

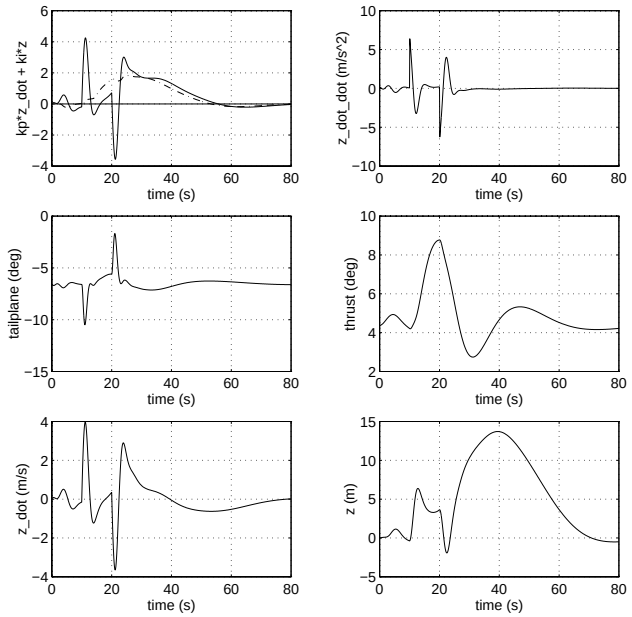


Figure 6.17: The evaluation of the disturbance rejection properties of the combined autopilot with an $MBPC$ having strong tuning ($R = 7500$) and constraints on the reference rate to the inner loop and actuators

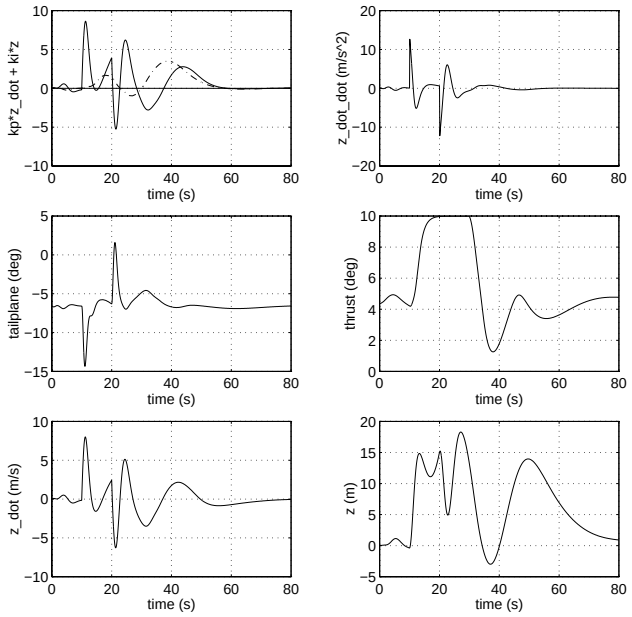


Figure 6.18: The evaluation of large disturbance rejection properties of the combined autopilot with an $MBPC$ having strong tuning ($R = 7500$) and constraints on the reference rate to the inner loop and actuators

The previous three Figures 6.16, 6.17 and 6.18 show in simulation the disturbance rejection properties of the combined controller (FM+(GS+SAS)). Different choices for the tuning of the $MBPC$ controller with and without constraints on rate of change of the reference to the inner loop were tested.

In Figures 6.16 and 6.17 the disturbance starting at time $t = 10s$ and ending at time $t = 20s$ had a magnitude of $10m/s$. The main purpose of the simulation was to test the passenger comfort in these two cases and check how by monitoring the actuators we can avoid their saturation despite of the large magnitude of the disturbance.

In Figure 6.18 the severity of the disturbance was increased up to $20m/s$ which drove the controller to operate with the actuators at their limits. In spite of this large disturbance the error in flight level tracking was acceptable from the design criteria point of view, not exceeding $20m$.

6.6 Results of the automated evaluation procedure

The RCAM design challenge involves designing a control law that is able to perform an approach to landing in the presence of turbulence and wind-shear, whilst remaining robust to mismodeling [MBT97].

This section presents the methodology-independent results of the designed controller. It is mostly based on the evaluation mission and scenario defined. Both 'overall tracking performance' and 'inner-loop behaviour' of the controlled system will be evaluated by means of bounds on key variables.

A discussion of the behaviour of the controlled aircraft will be based on the relevant flight segments for the longitudinal channel. A comparison between the guidance and stabilisation functions and the overall autopilot is provided. Finally, a summary of the comparative numerical results of the evaluation will be presented.

To prove the idea of the combined controller structure the most relevant segment of the approach manoeuvre is extracted. This segment, that represents the final descend to land, requires the capture of $-6°$ and $-3°$ glide-slopes.

We start with a glide-slope of $-6°$; again it is unavoidable that the aircraft leaves the desired trajectory. It should return to the trajectory without overshoot and well within a period of $30s$. After that we go to a glide-slope $-3°$ of such that we get an "inverse" behaviour with respect to the desired trajectory, that should be about half the size of the first response (if the system has a more or less linear behaviour). In Figure 6.19 the longitudinal response of the aircraft is plotted for three centre of gravity locations and with bounds that specify acceptable behaviour.

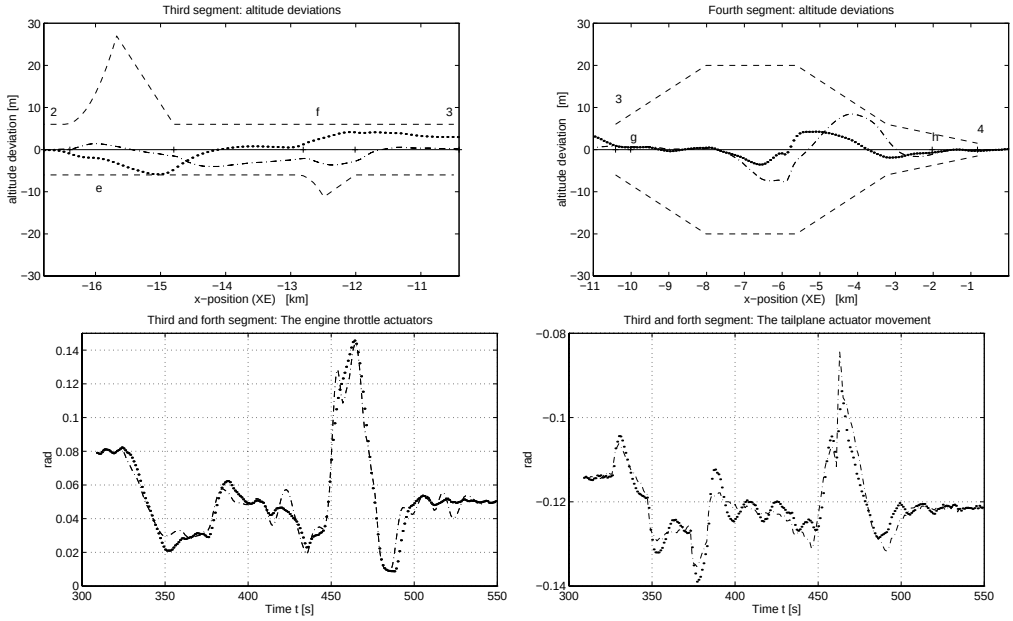


Figure 6.19: Segment III: vertical deviations from the desired glide-slope and Segment IV: vertical deviations from the desired glide-slope and corresponding actuators movements (The $MBPC/H_\infty$ combined autopilot (dotted) and the H-inf controller (dash-dotted)

Both controllers, $MBPC/H_\infty$ (dotted) and H_∞ on its own (dash-dot), behave in an acceptable manner. The difference arises when considering the vertical deviation from the desired glide slope. While the H_∞ controller tries to follow the descent reference trajectory as closely as possible resulting in overshoots, the $MBPC/H_\infty$ controller takes advantage of the *a priori* known trajectory, optimising and improving comfort and safety. The vertical deviations from the desired glide-slope are plotted in Figure 6.19.

While on final approach with a glide-slope of $-3°$ the effect of a wind-shear is considered (see segment IV g–h in Figure 6.19, second graph). The vertical deviations from the desired

glide-slope are plotted.

Criteria	$MBPC/H_\infty$	H_∞
Comfort	0.5137	1.3316
Safety	0.0075	0.0084
Power	0.0155	0.0150

Table 6.5: Comparative numerical results

Table 6.5 gives comparative numerical results based on the two simulations with the two distinct controllers, the combined autopilot $MBPC/H_\infty$ and the H_∞ loop shaping controller on its own. Each segment of the evaluation procedure is considered via the design criteria: comfort, safety and power. For each of the trajectory segments a single number was calculated. The smaller the numbers the better the design. The motivation and calculation principles of these figures can be found in [MBT97]. The performance and robustness criteria addressed by the inner controller are not included. Because the evaluation criteria are independent of the type of controller used the table contains calculable indicators that enable us to obtain an objective comparison between this one and completely different controllers from other design chapters.

6.7 Conclusions and lessons learned

The main difficulty in designing $MBPC$ is in the tuning of the many parameters available in the algorithm. But we found that, with experience, partly systematic tuning procedures were developed, also see [LY94].

It was possible to obtain a satisfactory controller despite a very simple model used as an internal model by the $MBPC$. In order to move towards more systematic design, from the robustness point of view, it was necessary to use an inner robust stabilising H_∞ loop-shaping controller to ensure robustness against large perturbations (eg. delays, mass and gravity centre variations).

The absence of well defined rules for choosing the tuning parameters requires serious experience of the designer in order to reach acceptable results. But assuming a good analysis tool set and adequate rules of thumb, the problems can be overcome. It may be noted that this problem is not specific for $MBPC$; for example the learning curve that will bring a classical control engineer to the stage when a complex MIMO design can be succesfull, designed using H_∞ will take up to several months, as discussed in [PGS+97]. The difference is that for the H_∞ loop-shaping controller synthesis more systematic procedures have been developed. Once the above mentioned obstacles are overcome, a fairly systematic redesign process can be developed, if the designer who inherits the redesign has experience of the technique and of the procedures used at the first stage of design.

The constrained optimisation, which has to be solved on-line, increases the computational complexity of implementing the controller. The difficulty to be overcome was reducing the computation time sufficiently to allow real-time operation. This was ensured

by the low sampling rate employed for the outer $MBPC$ controller. Moreover, progress can be expected both in the efficiency of solution algorithms for $MBPC$ and in the power of the hardware on which they run.

Constrained $MBPC$ may not offer any advantages over more conventional control algorithms apart from straightforward constraint handling. As a result we have presented in Section 6.6 an automatic pilot based on a combined use of an H_∞ loop-shaping controller that will provide the stability augmentation and guidance functions and an $MBPC$ controller that will act as a flight manager and overall supervisor. As expected, the $MBPC$ proved to be an effective technique for flight control, once this higher-level objective of flight management was included.

From the perspective of a supervisory loop based on a predictive controller, the closest approach to the one presented in this chapter is in [PG88]. There an architecture based on PID local controllers operating in continuous time and receiving the steady-state settings for each unit from an outer $MBPC$ is outlined.

Current qualification and certification procedures are not appropriate for $MBPC$ or some other modern control solutions. As long as constraints are not active our solution, having an analytical form, can be certified just as well as any other linear control law. But that provides only a partial analysis of the controller, when it is operating in its linear mode. Moreover, it could be argued that if the autopilot goes unstable the $MBPC$ controller could be switched off.

Chapter 7

The Model Based Predictive Reconfigurable Flight Control

"Control reconfiguration" has a broad meaning. This kind of technique is applied mainly in three situations:

1. the establishment of the system operating regime
2. performance improvement during operation
3. as part of fault accommodation.

Control reconfiguration is a critical technology in terms of safety. All methods of reconfiguration presume a certain initial knowledge of the system. In time, this may change gradually due to environmental effects or rapidly on account of faults. Hence, various impairments may significantly degrade the performance of the aircraft as well as the decoupling of the aircraft longitudinal and lateral axes. A reconfigurable controller possesses the capability to redistribute and coordinate the control effort during a system failure among the aircraft's remaining effective control surfaces, such that satisfactory flight performance is retained if possible. As the system changes the necessity for updating the control strategy increases. The flight control system is designed not only to cope with failures but to be robust with respect to variations in aerodynamic stability derivatives, high frequency unmodelled dynamics, time delays, sensor noise and atmospheric turbulence. If the aircraft dynamics is altered due to uncontrollable changes such as failed sensors, actuators or damaged structural elements the reconfigurable control should ensure a response still close to the nominal performance, if possible.

Reconfigurable control tries to avoid excessive hardware replication. Of course, the set of redundant components allows the flight control system to perform some of its functions in spite of the failures some of its components might have. Increased survivability and reduced support requirements involve redundant aerodynamic control surfaces and spare power during manoeuvres. The trade off in terms of replication is between the weight, size, complexity and power consumption of such redundant sets of equipment. Instead of relying on hardware redundancy a reconfigurable control system has to exploit inherent control redundancies in order to restructure in real time and preserve stability and performance.

Owing to time constraints, in many failure scenarios the automated control law redesign process has to make use of the fastest algorithm possible. When an impairment occurs the most urgent task is to stabilise the system so as to avoid catastrophic situations. Once the stabilisation has been carried out, there is more time for the control reconfiguration mechanism to manipulate the control in order to obtain better system performance.

In this chapter after surveying the literature for available solutions we suggest a new approach to fault tolerant control based on $MBPC$. This choice is motivated and then the controller architecture and tuning presented for two different cases: a missile and a fighter aircraft. Next, an automatic tuning procedure for the $MBPC$ controller is developed. The chapter ends with evaluation results in both cases and some conclusive remarks.

7.1 Literature review

7.1.1 General issues

There are authors focusing on these various aspects of reconfigurable system design, among them relevant studies were generated by [Bla95, Pat96].

According to [Bla95] several steps must be taken prior to designing a reconfigurable control system:

1. Essential properties of the system must be designed with respect to previous and likely fault conditions.

2. All potential failures and their effects have to be systematically determined, if possible. Techniques such as failure mode effect analysis (FMEA) can be used and their results eventually used in a supervisor type of algorithm.

3. System reliability analysis should provide information on various component failure rates. Additionally a reliability distribution chart must be produced.

Considerable effort has to be put into all stages of the system design in order to enable it to cope with the various requirements of reconfigurable control. Before carrying out the effective design process, an integrated and systematic understanding of the system structure, reliability and types of redundancy in sensors and actuators is required.

Practical control problems often involve more actuators and sensors than are needed for designing a control system in an economic way, as a result an appropriate set of actuators and sensors must be selected from the available candidates. In general establishing a control structure refers to both actuator/sensor selection and the process of partitioning and pairing.

The choice of partitioning/pairing problem in the case of decentralised control was studied by [SP96] and many practical tools such as relative gain array and other interaction measures have been developed. The RGA is used as a measure of the sensitivity of a control structure to diagonal input uncertainty.

The complementary problem of actuator/sensor selection was addressed in [LBMP95]. Its authors try to answer the question "What makes a control structure more desirable than others?". Screening tools that select the desirable candidate from a set of control structures were developed relying on measuring the potential performance of the candidates.

Most of the research in the area of partitioning/pairing was carried out in the stochastic setting and in general the criteria developed were based on nominal performance, as in [HMW80]. More advanced and systematic developments were provided by [BT78] relying on the "Condition Number Criterion" which is valid for some types of model redundancy.

Other advances were made by [LM91a] suggesting a criterion based on the structured singular value, a criterion that can deal with structured type of uncertainty. It is possible to see now, after having browsed through several techniques, that most of the criteria rely on a particular uncertainty description or a specific design approach.

On the other hand, during operation we encounter similar problems in the sense that plant inputs and/or outputs can be lost or appear at any moment, as an effect of diverse hardware failures. This will lead to a dynamic change of degrees of freedom available to the controller.

With respect to the two distinctive notions: controlled and manipulated variables we are able to provide a system classification, used by [ZDG96], into square (rare), "fat" (more manipulated than control variables, a type common in the real world) and "slim" plants (the reverse case to the "fat" one). The case of a "fat" plant is desired due to the extra degrees of freedom that can be used for control. In the case of reconfigurable control the regulator has to be designed in such a way that even if signals are lost it will stay awake trying to make the best of the sub-plant in control in terms of stability and performance. Unfortunately a well conditioned plant might contain a number of un-conditioned sub-plants.

We consider that another problem raised by the existence of such diverse sub-plants is that the controller has to provide them with a relatively similar behaviour in closed loop, which might cause completely different weights to be used in the process of translating the control requirements into method specific parameters and might require sophisticated automatic tuning procedures.

It is impossible to ensure control reconfiguration in the case of actuator or sensor failure without redundancy in the initial plant. As an inherent part of the design process for control reconfiguration, the nature and location of the available redundancy has to be analysed. If the level of safety for the closed loop does not match the level required then additional redundancy has to be provided at this stage in either direct or indirect form. Aerospace applications usually take advantage of both types of redundancies so as to cut the costs of flight control systems without affecting their safety.

The extent of redundancy therefore has a direct connection with the system reliability and the complexity of the FDI system in use. The type and level of redundancy provided determines the way in which control reconfiguration is provided. Hence the failed sensors or actuators in systems with redundancy will sometimes imply dramatic changes in the controller structure (*i.e.* at the algorithm level).

Direct redundancy

By definition direct redundancy means that a number of independent hardware channels are used. Generally a voting system is used in order to decide in between the good and impaired channels. A discussion of such voting systems is provided in [Wes77].

As long as several hardware devices automatically provide redundant signals the analysis being carried out in the design process will detect them and use them to keep the number of physical devices low. Often, different channels are supplied with different types of sensors, actuators or even software that have the same input-output features. Actually what is referred to by parallel redundancy is equivalent to direct redundancy with dissimilar pieces of hardware and/or software.

Analytical redundancy

In most control systems that rely on dynamic models there is an amount of built in analytical redundancy. For example [LP93] in a flight control system the roll rate p can be estimated/computed analytically by assuming measurements of yaw (Ψ) and roll (Φ) angle derivatives obtained via a filter having the Laplace transform $\frac{1}{1+0.1s}$. This way to compute p can be applied for all aircraft.

Other researchers [Wil76] provide analytical redundancy using the mathematical model. Kalman filtering can be considered as another mean of providing redundancy. The accuracy of all these techniques used to provide redundancy in an analytical manner suffers from the system nonlinearity and estimation errors.

7.1.2 Reconfigurable/fault tolerant control methods

Reconfigurable control can be tackled by various methods described in a succinct manner by the tree chart depicted in Figure 7.1. As we are able to see from this figure the precise nature of reconfigurable control depends upon whether the approach is passive or active.

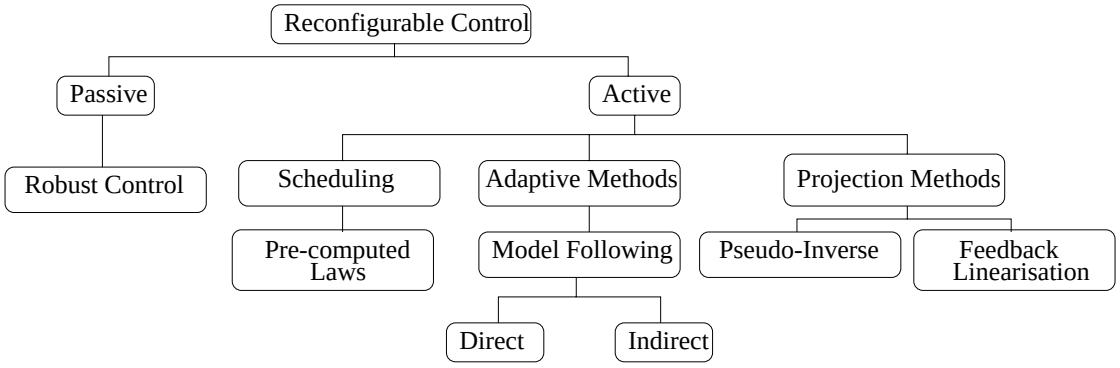


Figure 7.1: The tree diagram of reconfigurable control methods

As it can be seen from the Figure 7.1 one of the approaches to fault tolerant control is passive. This method use the robust control theory and techniques [ZDG96, GL95] to ensure robust stability of the closed loop in face of certain faults that can be regarded as various types of model uncertainties: additive, multiplicative or more general coprime factors which are well suited for aerospace applications [PGS+97].

The impaired system will provide an acceptable performance with the same controller designed for the nominal case, providing that it is situated within the robust performance radius ensured by the robust controller. Therefore, the performance of the closed loop

system is going to be affected to an extent depending on the robust performance provided by the same controller. Hence, it is considered that passive approaches are suitable in restricted cases when a fault has an relatively small effect on the controlled system. Of course with one single controller we have small such regions and therefore the requirement arises for active approaches.

Among various active approaches to reconfigurable control suggested in the literature we see a possible organisation in terms of following methods: model following, model reference adaptive control, multi-model adaptive identification and reconfigurable control, pseudo-inverse reconfiguration, control distribution and polynomial networks supercontroller strategy. The reader will be able in the next few sections to get more insight into those methods which will be required later to evaluate the newly suggested strategy, and see how the drawbacks of these methods are addressed.

Pre-computed laws

The concept of scheduling originated in the flight control area as an effective solution to deal with aerodynamic coefficient changes with respect to airspeed, angle of attack (AOA) or altitude. In such applications measured data from the environment triggers the right control action. This principle can be used within the reconfigurable flight control framework as well.

The idea extended in the field of reconfigurable flight control is altogether different. Pre-computed control laws are stored and activated in accordance with the information from an FDI system that monitors the plant and provides state estimates used as basis of a controller scheduling scheme.

In the case of complex systems for which fast parameter changes might be encountered due to the system envelope, failures or external disturbances, we measure the intelligence of a controller for such a system via its speed and accuracy in responding to such changes [Bla95]. From the set of controllers corresponding to such environments encountered during operation we have to determine the best one to be used.

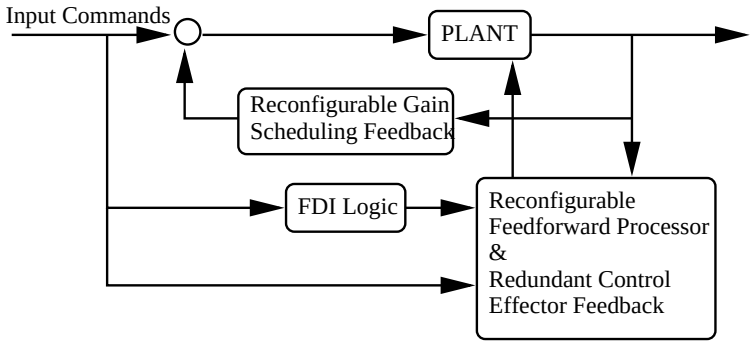


Figure 7.2: Restructurable flight control based on pre-computed laws

One of the schemes suggested by [MHBC89], quoted in [Pat96], is based on the following principles emphasised in the block diagram of Figure 7.2. The role of the processor shown in the figure is to cancel out the effect of the command error integrators as "surfaces are taken out" from the system (i.e. the plant becomes "slim"). The FDI logic is

designed to predict the plant states and to trigger the scheduled feedback. This feedback is in fact adjusted by a feedforward compensator (i.e. in open loop fashion) which means that no information about the closed loop is available resulting in a great drawback during failures, when the FDI system may induce instability and poor performance as a result of malfunctioning. This represent a major concern in most of the FDI based schemes.

Two main streams can be distinguished in the literature: the first one is the direct switching (*i.e.* an output based choice switching towards the next controller in a predetermined structure) and the second one called indirect switching is a strategy aiming to have multiple models used in determining when and to which controller we should switch. Researchers like [MGHM88], the initiator of the indirect switching, and [Mor93, NB97] concentrate their attention on proving the stability of such schemes.

Another approach that falls into the category of pre-computed control laws was developed by [MS91]. This method involves multiple model adaptive estimation and control as shown in the structure of Figure 7.3.

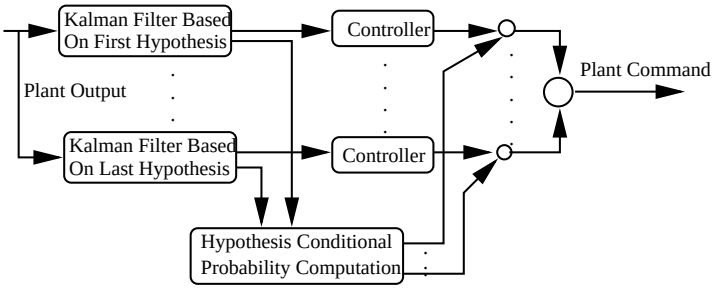


Figure 7.3: Restructurable flight control based on pre-computed laws and a probabilistic structure

For an adequate model of the aircraft and parameters being allowed to take discrete values, a Kalman filter is designed for each parameter value resulting in a bank of separate filters. Based on the residuals of these filters the conditional probability of each discrete parameter value is evaluated at each time step. The output value of each controller from the bank is weighted by its corresponding probability.

Examples of this method in the literature, when several controllers were designed for various conditions such as: healthy aircraft, failed pitch rate sensor, failed tail-plane and failed combination of other control surfaces equivalent to a "pseudo structural damage", can be found in [MS91].

The third approach of interest to this presentation is due to [NB97]. In their work Narendra et al. [NB97] suggests the architecture of the reconfigurable/adaptive control system presented in Figure 7.4.

In this approach the objective is to minimise the control error $e_i = \hat{y}_i - y$ between the real and desired output. All the errors between the output of the identification model I_i and the plant output are computed. Corresponding to each identification model I_i there is a controller C_i whose parameter q_i is chosen such that C_i achieves the control objective for I_i.

The switching rule that selects the appropriate $\hat{y}_i$ tries to yield the best performance for a given objective while stabilising the system. The choice for the switching rule was

to determine the performance cost indexes of the form:

$$J_i(k) = \alpha e_i^2(k) + \beta \sum_{j=0}^{k} \lambda^{-(k-j)} e_i(j)$$

Using the parameters α and β of the above performance index we can describe various choices of instantaneous and long term accuracy measures. The exponential weighting determines the memory of the cost function in rapid switching environments and ensures its boundedness when e_i is bounded.

The index $J_i(k)$ is read at each time step. An elapsed time $T_m > 0$ is allowed after every switch and only then the controller corresponding to the model having minimum $J_i(k)$ is chosen for control. The dwell time T_m is required in order to prevent arbitrarily fast switching.

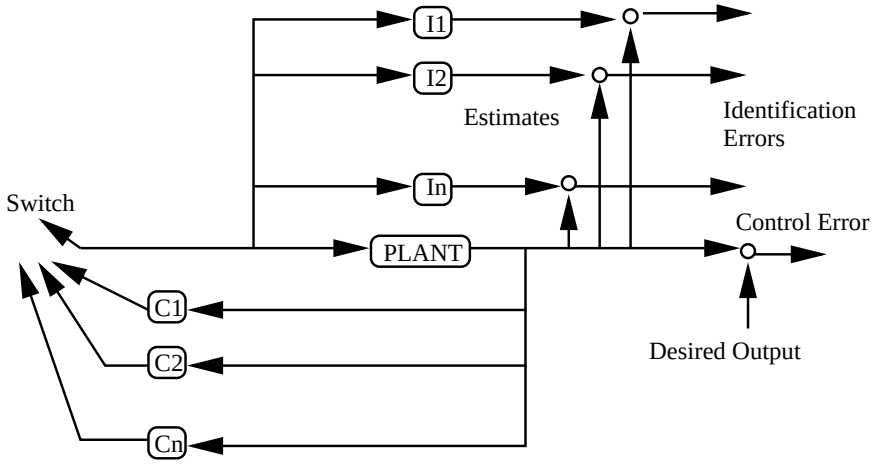


Figure 7.4: Restructurable flight control based on n identification models and controllers

The complexity of the problem in the multiple model case increases when a wider range of failure conditions is addressed. So, the additional computing resources can constitute a serious disadvantage. Examples of simplifications made to improve the computational speed for a terminal missile guidance can be found in [Rau95]. In [FR94] a novel approach employs each model from the bank in simulation done off-line storing the results. Just one Kalman filter is running on-line of which estimated results are compared with the stored ones enabling the decision of which model should be used.

Model following

The model following as a reconfigurable control technique was employed by many researchers such as [MO90, SKB82, Cha84, GB95, Bod97]. Aircraft models from F-8 to F-16 and F-18 were used in performing various simulation checks of the schemes suggested. These simulations were considered successful, providing an improved system response in the case when model parameters are varying of impairments can be found in the plant (aircraft).

In essence model following means the adjustment of a constant feedback gain, assuming that this gain is used in the nominal system, so that the reconfigured system approximates the nominal (ideal) one in some sense.

Adaptive reconfiguration via model following involves the redesign of the flight control law by a parameter estimation of the failed system and a re-optimisation. The resulting controller can be either a total new design or an augmented version of the initial controller, depending on the strategy.

Many schemes employ the conventional controller in parallel with the adaptive one, each of them being used in different situations. The nominal one is employed during the flight with the healthy aircraft or when the adaptive controller exhibits failures, as opposed to the adaptive one which is introduced in the algorithm when parameters vary and/or adaptation is needed. There are several problems associated with the on-line redesign of the control law.

The linear dependence between the error signal and the unknown parameters makes possible the use of a least square algorithm where a forgetting factor is introduced in the algorithm when parameters vary and adaptation is needed. The least square algorithm becomes unstable when there is insufficient excitation (ie. the case of a steady level flight). This method is discussed in detail in identification theory. Such situations are tackled by the conventional controller that runs in parallel with the adaptive one or by a persistent excitation of the regressors vector z in order to guarantee convergence of the parameters to the nominal values. Unfortunately the second approach might induce discontinuities and transients in the response of the algorithm. A stabilised version of the least square on-line identification with forgetting factor was developed in [CPM95].

In the case of flight control systems the form of the linear differential equations that reflect the plant model are:

$$\begin{aligned} \dot{x} &= Ax + Bu + d \\ y &= Cx \end{aligned}$$

The usual meaning in state space formulations is given to A, B, C matrices or u, x, y vectors. An input disturbance term d is introduced on purpose to account for the trim values of the input necessary to maintain steady flight at the operating point – a feature that will free the pilot from this task if used in the automation of trim computation.

A reference model with states available for measurement is defined as:

$$\dot{y}_m = A_m y_m + B_m r$$

where $y \in \mathbb{R}^m$ and $r \in \mathbb{R}^m$, A_m, $B_m \in \mathbb{R}^{m \times m}$ square matrices and A_m stable.

Simple solutions in the case of model reference control are obtained when the plant can be made to follow the reference model transfer matrix using an implementable (proper) compensator condition which mathematically boils down to $det(CB) \neq 0$ [1]. Note that $det(CB) \neq 0$ implies that the high frequency roll-off in each I/O direction is no faster than 20dB/decade(*i.e* system is essentially like an integrator at high frequencies), so it

[1] The matrix CB is called the plant high frequency gain matrix, a critical parameter in adaptive algorithms [Bod97].

does not often hold. A more complex reference model is required if the above assumption is not satisfied.

The feedback control law defined on the assumption of measurable states, which is valid in the flight control case, is:

$$u = C_0 r + G_0 x + v$$

where C_0, $G_0 \in \mathbb{R}^{m \times m}$ and $v \in \mathbb{R}^m$ are free controller parameters to be determined by the algorithm.

Considering the closed loop dynamics:

$$\begin{aligned} \dot{x} &= Ax + BG_0 x + BC_0 r + Bv + d \\ y &= Cx \end{aligned}$$

or directly in terms of y:

$$\dot{y} = (CA + CBG_0)x + CBC_0 r + CBv + Cd$$

the above equation will lead to the values of the controller parameter in the nominal case (i.e. when the plant model is considered known):

$$\begin{aligned} C_0^* &= (CB)^{-1} B_m \\ G_0^* &= (CB)^{-1}(A_m C - CA) \\ v^* &= -(CB)^{-1}(Cd) \end{aligned}$$

The implementation of the model following control strategy is given in Figure 7.5.

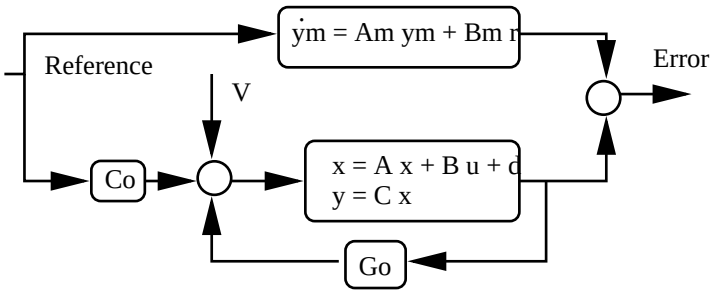


Figure 7.5: RFCS structure using model following

According to one of the most active researchers in this particular field [Bod97] the adaptive model following can be regarded as being direct (implicit) or indirect (explicit). The following sections are aimed at clarifying the difference in between these two.

Direct model following

The defining feature of direct adaptive control is that the algorithm estimates the controller parameters C_0, G_0, v directly from data.

Two main approaches to this estimation problem are considered in the literature [GA92, GB95]:

The output error ($e_0 = y - y_m$):
It is possible to write this error in the following form:

$$e_0 = (sI - A_m)^{-1}CB[(C_0 - C_o^*)r + (G_0 - G_0^*)x + (v - v^*)]$$

or

$$e_0 = (sI - A_m)^{-1}CB[\Phi w]$$

where $\Phi = [(C_0 - C_o^*), (G_0 - G_0^*), (v - v^*)]$ and $w = [r, x, 1]^T$.

The update law $\dot{\Phi} = -Ge_0w^T$ leads to a stable adaptive system in Lyapunov sense, as proved in [Bod97] when $(sI - A_m)^{-1}$ is strictly positive real transfer matrix ($A_m - A_m^T$ is negative definite) and $(CB)^TG^{-1} > 0$ a positive definite matrix. Conditions for having this approach implementable require knowledge of CB and G.

The input error ($e_u = u_m - u$)
Using the following identity:

$$u^* = C_0^*B_m^{-1}(\dot{y} - A_m y) + G_0^*x + v^*$$

the input error is defined as:

$$e_u = C_0B_m^{-1}(\dot{y} - A_m y) + G_0x + v - u$$

or $e_u = \Phi z$ where Φ is defined as before and $z = [B_m^{-1}(\dot{y} - A_m y), x, 1]^T$ is the regressor vector.

This approach avoids the existence of the transfer matrix between the parameter error and the error signal which eliminates conditions more strictly stated above, making possible the use of least square algorithms in a straight forward manner. Least square algorithms are used due to their faster convergence, compared with gradient algorithms.

In general the direct adaptive control methods used for control reconfiguration were proved to perform well the following tasks: the selection of proper trim values for inputs (a failure causing rapid changes in the command trim), the system decoupling of the inputs and outputs during the failure when the symmetry of the aircraft is lost and the tracking of the pilot commands which is ensured despite a major reduction in the control effectiveness.

The implicit model following can be implemented with lower gains [Pat96] which are affected by the error directly, in real time.

Several issues are still open so questions such as: the performance of the algorithm when noise is present or when state variables were not filtered from noise. Another UN-addressed issue is the constraints upon inputs (saturations). Currently these are handled by the aircraft outer loop controller.

Indirect model following

The indirect adaptive control methods are based on a recursive identifier combined with a control algorithm which uses the estimated parameters of the plant or the corresponding model as if they were perfect estimates. Additionally, it uses the estimated uncertainty affecting these parameters or the model.

The error vector using plant estimates $\hat{A}$, $\hat{B}$, $\hat{d}$ is:

$$\begin{aligned} e_i &= \hat{A}x + \hat{B}u + \hat{d} - \dot{x} \\ &= (\hat{A} - A)x + (\hat{B} - B)u + (\hat{d} - d) \end{aligned}$$

Least square algorithms of various types are used to find $\hat{A}$, $\hat{B}$, $\hat{d}$ and filtering via filters having transfer functions like $\frac{1}{1+0.1s}$ to obtain the state derivative $\dot{x}$.

Once the estimates of $\hat{A}$, $\hat{B}$, $\hat{d}$ are obtained the controller parameter are determined via the following expressions:

$$\begin{aligned} C_0^* &= (C\hat{B})^{-1}B_m \\ G_0^* &= (C\hat{B})^{-1}(A_m C - CA) \\ v^* &= -(C\hat{B})^{-1}(C\hat{d}) \end{aligned}$$

This method is less susceptible to parameter variation but model states must be generated and high feedback gains are required for adequate performance. The estimate of $C\hat{B}$ has to have $det(C\hat{B}) \neq 0$ at all times. In [DG96] a method based on pseudo-inverse calculations for $(CB)^{-1}$ is seen as a possible solution that offers feasibility of the indirect approach at all times.

The number of parameters to be identified is bigger than in the direct case (the plant has more parameters than the model $n > m$). Moreover two optimisations are required to produce the solution which slows down significantly the computations.

Such a control scheme is attractive, especially as a replacement of the FDI combined with the multi-model approach, but has to be robust to uncertainties and well behaved in the transient regime. The indirect approach as part of adaptive control schemes is able to provide the control of multivariable plants in the presence of drastic changes in the dynamics such as failures.

The presence of failures raises various problems due to:

- a strong coupling for the original plant
- a flight envelope to be covered by the continuous use of the adaptive algorithm and a system that may be highly unstable leaving little time for reconfiguration
- limitation of the actuators authority
- significant sensor noise

Mixed methods, considered to lie under the umbrella of indirect adaptive reconfigurable control, are based on: neural networks, fuzzy logic or knowledge based systems. They are called mixed due to the identification, that is carried out in general in an unconventional

manner. Sometimes accommodation methods based on statistical approach are restrictive in terms of dynamic changes and failure patterns, extensive on-line tuning being still needed after FDI. All these represent time consuming procedures.

A typical scheme to accommodate failures via intelligent control will involve a low level layer, running at a high sampling rate in order to accommodate hard type of faults such as: actuator failure, structure break or sensor breakdown, stores a set of predictable failure patterns. Then a timely action is produced to prevent the vehicle from being uncontrollable. Afterwards a higher layer accommodates the soft fault, more difficult to detect and isolate because of the induced ambiguities in the configuration management system.

An example of such mixed methods is given in [CMP93] when the model is identified on-line via the on-line optimising networks using a constrained least square algorithm. Then the Hopfield network generates an optimal model following the open loop control law. The work in the area of receding horizon control, predictive control applied to reconfigurable control is mainly due to [CMP93, PCM95, WB95].

The model based predictive control ($MBPC$) approach to reconfiguration is in fact an indirect reconfigurable control system that performs an on-line system identification, sometimes taking advantage about the particular form of the aircraft linear dynamics expressed in terms of stability derivatives, followed by on-line optimisation which is maximising the aircraft tracking performance before and after control surface failure, preventing instability. This method was employed with reasonably success by [CMP93, WB95]. Actually the paper by [WB95] concentrates much more on the identification algorithm, a modified parameter sequential least square identification.

In the case of [PCM95] a combined LQR with LP control strategy is employed such that a suboptimal tracking control law that does not violate the constraints is synthesised. This is not actually a proper $MBPC$ formulation but is the closest algorithm to the ones we are already used to. The LQR controller closes an inner loop, the input constraints being addressed in an outer on-line LP optimisation.

We would like to emphasise the latest contribution in this field provided by [PCS97] where a point-wise linearisation that replace the identification part from their previous approach is employed.

Pseudo-inverse method

As in the case of other intelligent control approaches to control reconfiguration the research on pseudo-inverse reconfigurable control was largely motivated by problems encountered in the aircraft industry - flight control design. In accommodating on-line unanticipated failures the self repairing system [2] based on the pseudo-inverse method (PIM) is considered a key approach. Researchers like [Ost85, Rat85] have used PIM successfully in flight simulations.

The basic idea featured by PIM is an on-line modification of the feedback gain produced such that properties of the nominal system are still retained despite the impairment.

[2] Some authors refer to "reconfigurable" control laws designed *a priori* to accommodate certain anticipated failures as opposed to "restructurable" control that implies an automatic design that accommodates unanticipated impairments.

Theoretically speaking the method relies on the computation of the pseudo-inverse of a constant matrix via least square approximation as described in [ZDG96].

In control system design the first issue to be addressed, before performance, is the one of stability. Unfortunately this is one of the main drawbacks of PIM, because only a few stability guarantees have been given in the literature [GA91] and even those are quite conservative. Most of the works talk about the instability encountered when applying PIM without any safeguard, but do not provide enough insight about when the method works or fails and under what circumstances.

Consider the standard form of the state space representation of the open loop plant:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{7.1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$. This plant has wrapped around it a state feedback $u = Kx$. Note the assumption of measurable states which is generally valid in flight control. The closed loop system is:

$$\begin{aligned} \dot{x} &= (A + BK)x + Bu \\ y &= Cx \end{aligned}$$

The system model containing impairments is expressed as:

$$\begin{aligned} \dot{x}_f &= A_f x_f + B_f u_f \\ y_f &= C_f x_f \end{aligned}$$

where A_f, B_f, C_f have the same dimensions as A, B, C.

The closed loop with the new feedback K_f, to be determined will be:

$$\begin{aligned} \dot{x}_f &= (A_f + B_f K_f)x_f + B_f u_f \\ y_f &= C_f x_f \end{aligned}$$

For the first time, in [Ost85], the idea of finding K_f such that the closed loop matrix which defines the failed system dynamics is similar, in a least square sense, with the one belonging to the healthy system, is proposed:

$$K_f = B_f^{\dagger}(A - A_f + BK)$$

where $B_f^{\dagger}$ denotes the pseudo-inverse of B_f in the least square sense is mentioned.

The row rank of B_f defines the type of problem we are going to solve. If B_f has full row rank then K_f obtained via the pseudo-inverse calculation always satisfies the equation $A + BK = A_f + N_f K_f$ otherwise an approximate solution can be found. Having such an approximate solution means that we do not know how close the behaviour of the reconfigured system is to the nominal one. Hence some guarantees are required.

There is a quick solutions both from the implementation and stability guarantees point of view. This employs the PIM method in the restructurable control sense by having the K_f computed and stored for many anticipated failures. The stability guarantees can be

given by checking off line all the plant plus controller combinations to be identified by the FDI system.

Another solution is the reconfigurable control solution which relies on a modified approach by [GA91]. It is based on the PIM method in conjunction with the theory of robust stability of systems with structured uncertainty. Effectively this method tries to recover the performance of the impaired system as much as possible, retaining the closed loop stability.

In [ZK87, Yed88] stability bounds for linear systems with structured uncertainty are derived. Both methods use a state space perturbation of the form $\dot{x} = (A + E)x$ where $E = \sum_{i=1}^{q} k_i E_i$ with E_i constant matrices and $A \in \mathbb{R}^{n \times n}$ the stable state matrix of an LTI system. The idea presented in [GA91] is to take advantage of these bounds and derive a stable $A_f + B_f K_f$ closed loop state matrix based on them.

The SISO approach has an analytical solution for the problem:
$\min K_f J$ where $J = \| A + BK - A_f + B_f K_f \|_F$. The corresponding algorithm that determines the solution of modified PIM has several steps. A first attempt is made to compute K_f via the classic PIM followed, in the unstable closed loop case, by a more computationally involved algorithm that determines K_f via minimisation of J subject to the constraints stated in [ZK87] or [Yed88]. This in effect picks up from the class of stabilising controllers for the impaired plant the one that gives a closed loop behaviour closed to the nominal one.

For the MIMO case the procedure is similar. In the unstable case effectively K_f is perturbed (i.e. $\hat{K}_f = K_f + \Psi$) with a Ψ based on the bounds derived in [ZK87] or [Yed88].

As an important remark we have to mention that the constraints used in minimisation are sufficient but not necessary conditions to guarantee stability; the constraints tend to be more conservative as q, the number of E_i constant matrices increases [GA91].

In [Pat96] two other serious limitations of the PIM approach to control reconfiguration are mentioned:

1. PIM make no explicit use of the robustness properties of the FDI system relying too much on the supervision system used in determining the fault signature.

2. The state measurement in other plants than aircraft is not always available.

The second alternative of employing model based predictive control (MPBC) as a reconfigurable control system will use this approach, modified at the level of controller synthesis.

As a final conclusion we would like to point out the resemblance of the pseudo-inverse method with model following – see [GA92] – another technique extensively used in aircraft reconfigurable control. In the the first case a nominal design of the closed loop is approximated whereas in the second an ideal model tends to be followed.

7.1.3 Comments on the FDI system

Critical failures in flight control may result from sensor failure, actuator, structure damage or breakdown. The objective of most of RFCS – to permit a graceful degradation of system performance in the presence of failures, damage or malfunctions – require the presence

of a Fault Detection and Isolation (FDI) system. The FDI is required to quickly and reliably determine the faulty components in the presence of noise and modelling errors. Primarily requirements for a succesfull fault detection are: low false alarm, robustness against disturbances and manoeuvres and minimal detection delay. Although developing the FDI is beyond the purpose of these work we give a brief review of main strategies used at the moment. This is due to the possible development and their use in conjunction with high fidelity models.

Involving into the plant model information about the delay of the FDI system in providing information about the failed plant is a difficult task. Before the failure the controller based on the nominal model is used. At the time of the failure until the FDI decides the new characteristics of the plant and hence the model in use we require a certain degree of robustness to be built in the controller. After enough information was acquired about the plant the FDI system will be able to select from a collection of pre-computed controllers the one which provides the best performance and stability. The filters carefully tuned have to avoid the choice of a "bad" model. Of course, robust control design enable us as well to reduce this number up to an implementable one.

An usual FDI system contains two levels:

1. An actuator failure FDI system used to detect and isolate the malfunction of the actuator which could have severe impact on the system response. Typically the first layer consists of devices that detect the failure by one of the following methods: hardware and parity checks and/or detection filters. Fault detection and isolation filters consists of a set of estimators arranged in several cascade layers to test for various failures.

2. A second layer refers the system itself having the main task of providing information regarding structural damage of the aircraft

According with the generally accepted terminology FDI systems perform the following tasks:

- Detection — the indication that something is going wrong in the system behaviour.
- Isolation — the determination of the exact location of the failure.
- Identification — the determination of the failure size.

As a result typical features for FDI systems are:

- Isolability — the ability to isolate certain specific faults provided their size is large enough; this test depend on the system matrices structure being a statistical based algorithm.
- Sensitivity — a qualitative measure defining the size of faults that can be isolated under certain conditions; it depends on the size of system matrices elements and noise properties having closely related the time in which the FDI produces a solution
- Robustness — the property that enables the existence of modelling errors; a serious interference with fault isolation has to be overcome using time history

It is required an enhanced robust stability and performance region achieved with each model and the corresponding controller in order to ensure blending and switching of them. The placement of the threshold, based on the ability of the FDI system and of individual controllers to provide robust closed loops, it is very important. By placing it at the significantly higher level than the detection one it is possible to reduce false alarms without a major decrease in the closed loop failure's sensitivity.

Several classifications of FDI methods were provided in literature but the one used here is based on the type of method used:

- failure sensitive filters
- voting systems
- multiple hypothesis filter detectors
- jump process formulations
- innovation based detection systems

In all these methods *a priori* knowledge of the plant dynamic model is required with more emphasise on this issue in the first case described above. For this case actually methods of detecting and isolating faults accurately are subject of current research.

In fact the research that has been carried out until now has not revealed completely how much further information can be provided by the FDI unit or what are the best mechanisms to perform the integrated design of the FDI and the reconfigurable controller.

Looking at the interaction between the FDI system and the high fidelity model we need to establish, if it is possible, how quick and accurate an update of this model can be such as to reflect the dimension of the failure. Our main concern is that the existence of one structural failure in general is driving towards a series of failures which can hardly be addressed by a system which deals with predetermined situations. For instance in the case quoted in the work of [Sma97] the separation of engines 3 and 4 of a Boeing 747-200F caused several structural and actuator failures:

1. right hand wing leading edge severely damaged
2. right hand wing leading edge flaps partly lost
3. no outboard aileron available due to the outboard trailing edge flaps failure
4. right hand inboard aileron less effective due to disturbed airflow caused by right wing damage and loss of pylon 3
5. the loss of pylons 3 and 4
6. partly lost spoiler system
7. malfunction of the lower rudder

In [Sma97] insight is provided into the flying capabilities of an impaired aircraft. In effect the author was updating off line an high fidelity model based on the data obtained from the Digital Flight Data Recorder (DFDR) such as to account for the right wing multiple engine separations and for the contribution of the right wing leading edge damage. This example represents an important argument to support our fault tolerant strategy assuming the existence of an FDI system which could have done in automatic manner what the author of [Sma97] achieved after certain effort during his research (*i.e.* an accurate update of the high fidelity model). For instance the augmentation of the aircraft aerodynamic surfaces and actuators with a system – an active component of the FDI – which informs the about the possible losses of parts or surfaces in flight can represent a potential alternative.

Going forward we can imagine a solution which involves a high fidelity model embedding all possible failure modes encountered in the past period of time in various accidents combined with an FDI system capable of detecting and isolating these. The FDI system probably will have the high-fidelity model – written as a function of the aerodynamic coefficients susceptible to modification as parameters. Then the identifying these parameters on line the computational burden can be a reduced and in the same time more knowledge about the model is built in the FDI system. The possible drawback of such scheme is the existence of a nonlinear optimisation that has to be employed for the current values of the model parameters. In principle operating such a system in real time will enable us to tackle the difficult problem of the on board implementation and to account for the failure modes encountered during various accidents.

7.2 A new reconfiguration strategy

As we were able to see in the first sections of this chapter several methods were developed to deal with fault tolerant issues. In the following, inspired by these methods, their features and drawbacks, we suggest a new reconfiguration strategy based on four ingredients:

- Fault Detection and Isolation (FDI)
- High fidelity models
- Model approximation and simplifications techniques
- Constrained Model Based Predictive Control ($MBPC$)

Such an approach is motivated by several factors and a few problems encountered with other active approaches surveyed in the previous sections of this chapter.

The four system components shown in Figure 7.6 work together as follows:

1. When a failure occurs the FDI system pinpoints the nature of it. Updated parameters are passed to the high fidelity model. Actuator failure information is passed to the $MBPC$ controller as constraint modifications, if appropriate.

2. Freezing the high fidelity model with respect to the parameter vector yields a linear model which is used by the $MBPC$ as its internal model. An important advantage

for real-time implementation is that this is a computationally inexpensive way of obtaining, in adaptive fashion, a linear internal model, compared with other possibilities such as identification.

3. The constrained $MBPC$ controller provides inputs to the plant. Giving it enough degrees of freedom (a large enough set of control inputs) enables it to keep the plant close to the required trajectory. We are assuming here that any failures are compatible with maintaining this trajectory.

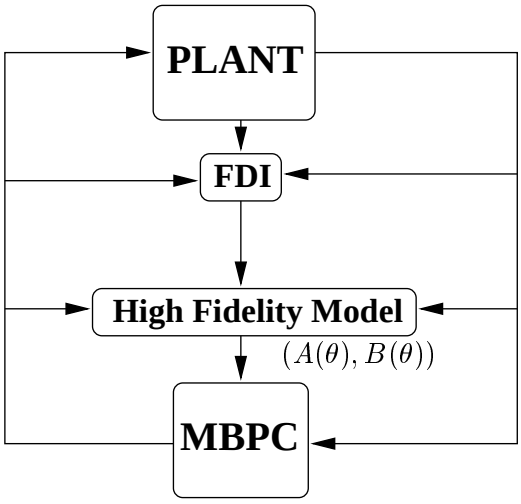


Figure 7.6: The reconfiguration and scheduling $MBPC$ based strategy

All the components performing the functions mentioned above are shown in Figure 7.6. We claim that the system ensures the following successive layers of reconfiguration:

- A simple reconfiguration occurs if there are actuator failures which are consistent with maintenance of the required flight conditions. This can occur even without FDI information [Mac97].
- Small structural failures are reflected in the high-fidelity model, as shown in Section 7.2.1, and hence in the linear internal model used by the $MBPC$ controller. No tuning is necessary for the controller; its original cost function still reflects well the various criteria for which it was designed.
- Major failures will again lead to a change in the linear internal model, but now retuning of the controller will be necessary. The approach which keeps the horizons fixed but adjusts the weights in (2.1) appears to be the most promising at present. This solution is presented in Section 7.2.4.

In answering the question "Why $MBPC$ for reconfigurable control?" we can enumerate the following reasons:

1. the $MBPC$ multi-model formulation (see Section 2.2 and Section 3.3.2), enables us to cover most of the situations addressed by the formulation based on pre-computed laws,

2. the simple form of the cost function leads to a linear time-invariant controller in the unconstrained case which can be parametrised with respect to the elements of output and control increment diagonal weights, and
3. the property of handling constraints in a straightforward manner, and
4. being a challenging problem it can emphasise several features of predictive controllers and push the scheme to its limits.

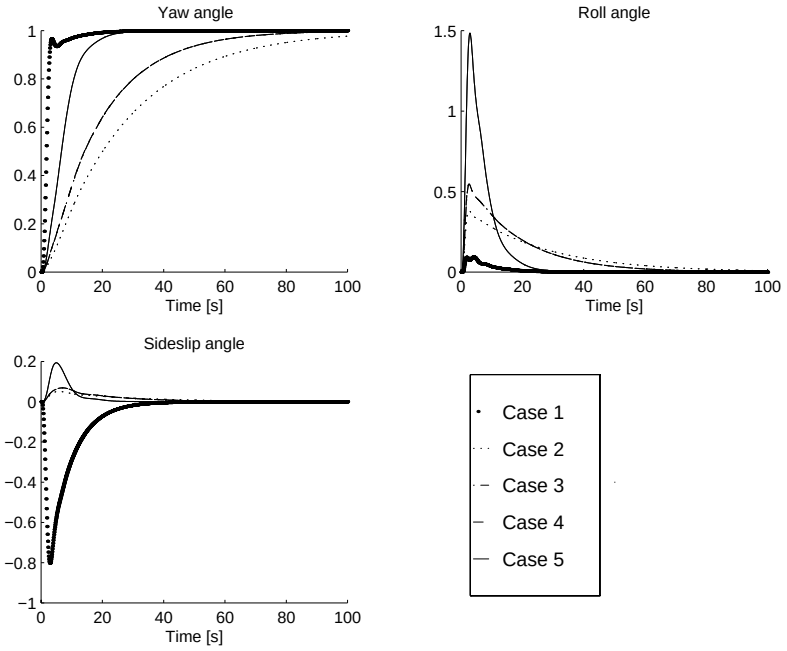


Figure 7.7: Yaw angle step demand with failed rudder: Controlled outputs

For instance, to offer a better insight into the last feature mentioned and its implications in reconfigurable control, we present here an example of reconfiguration performed by an MBPC flight controller, when the rudder of an aircraft jams, and it is required to change the aircraft heading (yaw angle). The same linearised model of the 'RCAM' model is used for this example, as in Chapter 5. This is the internal model used by the MBPC controller.

There are 3 controlled variables: the yaw angle, the roll angle, and the side-slip, and 4 actuators: the rudder, the ailerons, the tail-plane, and the engine thrust. A rudder jam (at the neutral position) is simulated by disconnecting the rudder demand signal (issued by the controller) from the rudder. A step demand is then made on the yaw angle.

The results are shown in Figures 7.7 and 7.8. Each of the sub-figures in these figures shows 5 cases:

Case 1 Normal operation of the rudder. The rudder and aileron positions are constrained to be within ± 2 units on the graphs, which represents $\pm 20^\circ$ in each case.

Case 2 Jammed rudder. No FDI information supplied to the controller, so that the controller is not aware of the failure. The constraints on the rudder position demand are removed, in order to see the effects of actuator constraints (known to the controller) in this scenario. (The rudder demand has been reduced by a factor of 300 in the figure for this case.) Of all the cases, this gives the slowest yaw angle response.

Case 3 Jammed rudder. No FDI information supplied to the controller, but the usual $\pm 20^\circ$ constraints are restored on the rudder position. The yaw response is significantly faster in this case, because the controller stops relying on the rudder sooner, and makes more use of the ailerons, as can be seen from the larger roll angle.

Case 4 Jammed rudder. FDI information supplied — the constraint on the rudder demand has now been tightened to $\pm 0^\circ$, so that the controller knows that it cannot move the rudder. In fact this makes so little difference that the plots for Cases 3 and 4 cannot be distinguished from each other. The reason for this is that in each case the controller moves the rudder demand to its constraint almost immediately, and then uses the ailerons and other actuators. Since the actual rudder position is the same in both cases, the two behaviours are virtually identical.

Case 5 Jammed rudder. FDI information supplied, as in Case 4. But now the weight on roll errors in the cost function has been reduced by a factor of 3, approximately. This leads to a much faster response of the yaw angle. It can be seen that a much larger roll angle develops during the first 10 seconds of the manoeuvre when this weight has been reduced, and the lift then has a larger component in the horizontal plane, which changes the aircraft's heading.

Because the manoeuvre simulated here is rather artificial we can consider a more practical requirement than changing only the yaw angle. This would be to change the aircraft heading, without much concern for what combination of body angles was most appropriate to achieve it.

To continue the comparison with other methods presented in the foregoing sections it is well to note that our approach is one that, like in the model following case, computes on-line the closed loop feedback matrix based on an update of the internal model. For this model no special restrictions apart from its controllability and observability are required. Therefore, we approach reconfigurable control as a multi-model adaptive control problem, namely one in which adaptation occurs by discrete changes to the internal model, rather than continuous tracking of a gradually-changing model.

The classic pseudo-inverse method was extended in the $MBPC$ case such that an automatic tuning procedure for the predictive controller is provided. We have imagined this procedure not only as a tool to provide off-line tuning of the controller having an internal model based on the impaired plant such as to match the performance of the nominal one, but as a potential method to be applied on-line in the event of a major failure.

As shown in the survey section Patcher et. al. [PCM95] were the first to view a simplified version of constrained predictive control as a promising tool for reconfiguration. Their scheme was an indirect adaptive approach with the system identification module

employing a mixed identification method (optimisation involving flight mechanics information) and the controller being split into two loops (an LQR inner loop and an outer linear programme on-line optimisation to enforce constraints a step ahead).

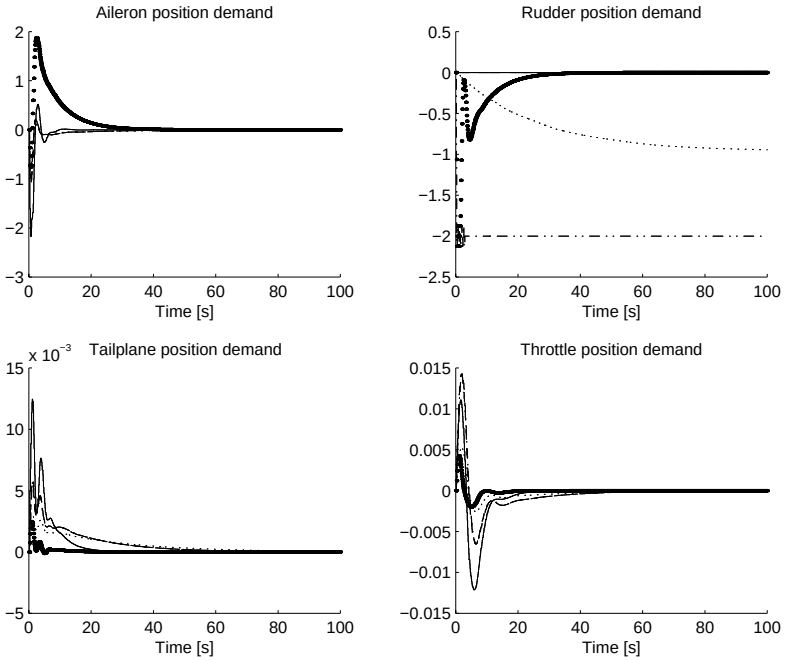


Figure 7.8: Yaw angle step demand with failed rudder: Actuator demands. (Key as for previous figure.)

In our scheme the on-line identification is replaced with a combination of high-fidelity models and approximation/simplification techniques that will lead to a linear representation of the plant to be used as the $MBPC$ internal model. We therefore do not use an identification module, at least for ordinary 'scheduling'. For reconfiguration in case of failures some fault detection module is required; for the purpose of this work we assume the existence of such a module. In fact two techniques were developed: one of them involves a quasi-LPV form of the nonlinear model versus the other approach which is based on point-wise linearisations around current value of the input and state pair along the plant trajectory.

7.2.1 High fidelity models written in the quasi-LPV form

High fidelity dynamic models are increasingly built for complex plants. This has been the case in the aerospace industry for many years. This section will firstly introduce quasi-LPV models. Next it shows how these can be integrated to give a strategy for scheduling and reconfiguration. The solution will be illustrated via a well known missile example.

In [SC93] a quasi-LPV model that embeds the plant nonlinearities without interpolating between point-wise (Jacobian) linearisations is presented. The main characteristic

of these models, compared with the usual LPV way of representing systems, is that the scheduling variable is a state of the model.

The quasi-LPV approach is mostly suited for systems exhibiting output nonlinearities (*e.g.* aerospace applications). Such nonlinearities enable us to write the system in form of equations (7.2). A principal requirement for a nonlinear system to be transformed into a quasi-LPV system is that the number of available equations has to be equal to the number of states plus the number of outputs minus the number of scheduling variables. When it is impossible to embed all the system nonlinearities in the output then the transformations used in producing the quasi-LPV model, see equation (7.4), have to be approximated up to first order terms in all the states except the scheduling parameters.

To develop the quasi-LPV model we start with a nonlinear missile model with the states α (angle of attack) and q (pitch rate), and the command δ (horizontal fin angle) of which complete details are given in [SC93].

Then we write the nonlinear in model such a form that the nonlinearities depend only on the scheduling variable α:

$$\frac{d}{dt}\begin{bmatrix}\alpha\\ q\end{bmatrix} = f(\alpha) + \begin{bmatrix}A_{11}(\alpha) & A_{12}(\alpha)\\ A_{21}(\alpha) & A_{22}(\alpha)\end{bmatrix}\begin{bmatrix}\alpha\\ q\end{bmatrix} + \begin{bmatrix}B_{11}(\alpha)\\ B_{21}(\alpha)\end{bmatrix}\delta \tag{7.2}$$

A family of equilibrium states, parametrised by the scheduling variable α, is obtained by setting the state derivatives to zero:

$$0 = f(\alpha) + A(\alpha)\begin{bmatrix}\alpha\\ q_{eq}(\alpha)\end{bmatrix} + B(\alpha)\delta_{eq}(\alpha)$$

Providing that there exist continuously differentiable functions $q_{eq}(\alpha)$ and $\delta_{eq}(\alpha)$, we are able to write the system (7.2) in the following form [SC93]:

$$\begin{array}{l}\frac{d}{dt}\begin{bmatrix}\alpha\\ q - q_{eq}(\alpha)\end{bmatrix} = \\ \begin{bmatrix}0 & A_{12}(\alpha)\\ 0 & A_{22} - \frac{d}{d\alpha}q_{eq}(\alpha)A_{12}(\alpha)\end{bmatrix}\begin{bmatrix}\alpha\\ q - q_{eq}(\alpha)\end{bmatrix} + \\ \begin{bmatrix}B_{11}(\alpha)\\ B_{21}(\alpha) - \frac{d}{d\alpha}q_{eq}(\alpha)B_{11}(\alpha)\end{bmatrix}(\delta - \delta_{eq}(\alpha))\end{array} \tag{7.3}$$

This form gives a different α-dependent family than would be obtained by point-wise linearisation.

In order to use (7.3), the function $\delta_{eq}(\alpha)$ must be known. This can be estimated by using an 'inner loop' [SC93] but, because of model uncertainty, this can reduce the robustness of the main control loop in a way which is difficult to predict at the design stage. Like in [SC93] we avoid the problem generated by the existence of an inner loop required to compute $\delta_{eq}(\alpha)$ by adding an integrator at the plant input. As a result we

have the quasi-LPV form for the system dynamics:

$$\frac{d}{dt}\begin{bmatrix} \alpha \\ q - q_{eq}(\alpha) \\ \delta - \delta_{eq}(\alpha) \end{bmatrix} = \begin{bmatrix} 0 & A_{12}(\alpha) & B_{11}(\alpha) \\ 0 & A_{22} - \frac{d}{d\alpha}q_{eq}(\alpha)A_{12}(\alpha) & B_{21}(\alpha) - \frac{d}{d\alpha}[q_{eq}(\alpha)]B_{11}(\alpha) \\ 0 & -\frac{d}{d\alpha}[\delta_{eq}(\alpha)]A_{12}(\alpha) & -\frac{d}{d\alpha}\delta_{eq}(\alpha)B_{11}(\alpha) \end{bmatrix} \times \begin{bmatrix} \alpha \\ q - q_{eq}(\alpha) \\ \delta - \delta_{eq}(\alpha) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \nu \tag{7.4}$$

This final form is actually the representation used for control purposes. If α remains constant, the $MBPC$ controller stabilising this model will drive the states to zero which means that the δ input to the plant will be set at the true δ_{eq} trim value. The $MBPC$ internal model will be the corresponding LTI system obtained at each value of the scheduling parameter α.

The controlled output of the plant is the normal acceleration (n_Z), which is a nonlinear function of α and $\delta - \delta_{eq}(\alpha)$. In the internal model a linearised approximation of n_Z with respect to α is used, as in [SC93].

The normal force (C_Z) and pitch moment (C_m) aerodynamic coefficients are embedded in the expressions for $q_{eq}(\alpha)$, $\delta_{eq}(\alpha)$, $A(\alpha)$ and $B(\alpha)$. The representation of C_Z and C_m is approximated by the following expressions:

$$\begin{aligned} C_Z &= \Phi_Z(\alpha) + b_Z\delta \\ C_m &= \Phi_m(\alpha) + b_m\delta \end{aligned}$$

where the expressions for b_Z and b_m are given in [SC93], and $\Phi_Z(\alpha)$, $\Phi_m(\alpha)$ are polynomials in α. Let k_Z and k_m, respectively, be the vectors of coefficients of these polynomials. We assume that k_Z and k_m will be the outputs of an FDI module, as shown in Figure 7.9. These expressions for C_Z and C_m explicitly contain contributions from all the missile's surfaces, and are based on data obtained from wind tunnel tests. Such data may be available in the form of a database [Hoa78].

The quasi-LPV model of the plant gives the clarity required by industry and eases certification since it retains a physical meaning for the elements of the model LTV matrices. It is advisable to have the scheduling variable as a system output rather than an estimate.

7.2.2 The HIRM nonlinear model and its on-line point-wise linearisation

In this section we restrict our attention to the longitudinal channel of the High Incidence Research Model. We assume that the side slip angle and the roll and yaw rates are zero.

The model is based on the following equations of motion [MBT97]:

$$\begin{aligned}
\dot{\gamma} &= -\left(\frac{g}{V}\right)\cos(\gamma) + \left(\frac{L}{mV}\right) + \left(\frac{F}{mV}\right)\sin(\alpha) \\
\dot{\alpha} &= q - \dot{\gamma} \\
\dot{q} &= M\left(\frac{\bar{c}}{J_y}\right) - Z_{ATP}F \\
\dot{V} &= -g\sin(\gamma) - \left(\frac{D}{m}\right) + \left(\frac{F}{m}\right) \\
\dot{H} &= V\sin(\gamma)
\end{aligned}$$

In these equations γ, α, q, V, H denote the vertical flight path angle, angle of attack, pitch rate, airspeed, and altitude, respectively. The variables m, $\bar{c}$, J_y are constants denoting, respectively, the aircraft mass, the reference mean aerodynamic chord, and pitch moment of inertia.

The aerodynamic drag (D), lift (L) and pitch moment (M), expressed in body axes, are:

$$\begin{aligned}
D &= \bar{q}S\left[C_{X_{\delta_{TS}}}(\alpha,\delta_{TS}) + C_{X_{\delta_{CS}}}(\alpha,\delta_{TS})\delta_{CS}\right] \\
L &= \bar{q}S\left[C_{Z_{\delta_{TS}}}(\alpha,\delta_{TS}) + C_{Z_{\delta_{CS}}}(\alpha,\delta_{TS})\delta_{CS} + \right. \\
&\quad \left. + C_{Z_q}(\alpha,\delta_{CS})\frac{q\bar{c}}{2V}\right] \\
M &= \bar{q}S\bar{c}\left[C_{M_{\delta_{TS}}}(\alpha,\delta_{TS}) + C_{M_{\delta_{CS}}}(\alpha,\delta_{TS})\delta_{CS} + \right. \\
&\quad \left. + C_{M_q}(\alpha,\delta_{CS})\frac{q\bar{c}}{2V}\right]
\end{aligned}$$

where $\bar{q} = \frac{1}{2}\rho(H)V^2$ is the dynamic pressure and $\rho(H)$ the air density which varies as a function of altitude.

In the above equations the meaning of F, δ_{TS}, δ_{CS} is thrust magnitude, symmetrical taileron deflection, and symmetrical canard deflection, respectively.

The non dimensional coefficients in these equations $C_{X_{\delta_{TS}}}(\alpha,\delta_{TS})$, $C_{X_{\delta_{CS}}}(\alpha,\delta_{CS})$, $C_{Z_{\delta_{TS}}}(\alpha,\delta_{TS})$, $C_{Z_{\delta_{CS}}}(\alpha,\delta_{CS})$, $C_{Z_q}(\alpha,\delta_{CS})$, $C_{M_{\delta_{TS}}}(\alpha,\delta_{TS})$, $C_{M_{\delta_{CS}}}(\alpha,\delta_{CS})$, $C_{M_q}(\alpha,\delta_{CS})$ are given in look-up tables obtained as a result of wind tunnel tests [MBT97, Wil97]. In the high fidelity model their values are computed by linearly interpolating between the values stored as functions of the variable α and the corresponding surface deflection.

We propose to obtain an on-line linearisation of the system by using Taylor series to derive the analytical expressions for the linear equations. This is done around the current input and state pair, which are not, in general, equilibrium points (that is, the linearisation is not performed about a "trimmed" condition of the aircraft). The values of the derivatives of the aerodynamic coefficients are stored in look-up tables.

As an example, we show here the expression for $\Delta V = V - V_c$, where V is the total

airspeed and V_c is the current airspeed value at which the linearisation is performed:

$$\begin{aligned}\dot{V}c + \dot{\Delta V} &= -g\cos(\gamma_c)(\gamma-\gamma_c) + \frac{1}{m}(F-F_c) - \\ &-\frac{\bar{q}(H)S}{m}\left[\left(\left(\frac{\partial C_{X_{\delta_{TS}}}(\alpha,\delta_{TS})}{\partial\alpha}\right)_{(\alpha_c,\delta_{TS_c})}(\alpha-\alpha_c)+\right.\right. \\ &+\left.\left(\frac{\partial C_{X_{\delta_{TS}}}(\alpha,\delta_{TS})}{\partial\delta_{TS}}\right)_{(\alpha_c,\delta_{TS_c})}(\delta_{TS}-\delta_{TS_c})\right)+ \\ &\left(\left(\frac{\partial C_{X_{\delta_{CS}}}(\alpha,\delta_{TS})}{\partial\alpha}\right)_{(\alpha_c,\delta_{TS_c})}\delta_{CS_c}(\alpha-\alpha_c)\ +\right. \\ &+\left.\left(\frac{\partial C_{X_{\delta_{CS}}}(\alpha,\delta_{TS})}{\partial\delta_{TS}}\right)_{(\alpha_c,\delta_{TS_c})}\delta_{CS_c}(\delta_{TS}-\delta_{TS_c})\ +\right. \\ &+\left.\left.C_{X_{\delta_{CS}}}(\alpha_c,\delta_{TS_c})(\delta_{CS}-\delta_{CS_c})\right)\right]+\mathcal{C}\end{aligned}$$

where the subscript c denotes the current value of a variable and $\mathcal{C}$ collects all the values determined at the current state and input from the Taylor series expansion. In trim $\dot{V}_c$ and $\mathcal{C}$ are zero.

This procedure allows linearised models to be obtained quickly enough for real-time use in the case of $MBPC$ because of the way the internal model is written in Δx and Δu. Of course this assumes that the FDI system can update the look-up tables quickly enough.

The reason of having two alternative approaches used in getting the $MBPC$ internal model (*i.e.* the quasi-LPV and point-wise linearisation) is the impossibility of writing the nonlinear model in a quasi-LPV form when we want to leave un-blended the redundant actuators. Such a case occurs when we would like to have the optimiser deciding upon this issue.

The way the aerodynamic forces, together with the engine thrust, act upon the 6-DOF nonlinear simulation model is defined based on wind tunnel data and simple actuator models (*i.e.* first order models plus delay). At the moment the model is programmed in C using the Matlab Simulink S-function template and run on a SPARCstation 20 as compiled code.

For the simulations shown in Section 7.2.5 we have employed numerical linearisation instead (the Simulink function linmod), for convenience. In principle this gives the same linearised model because of the way the A, B, C, D matrices are computed in the case of a trim or non-trim pair of inputs and states.

7.2.3 Designing the $MBPC$ controller

The autopilot structure

In Figure 7.9 all the essential components of the system which perform the functions referred in the Section 7.2 for the missile case are depicted.

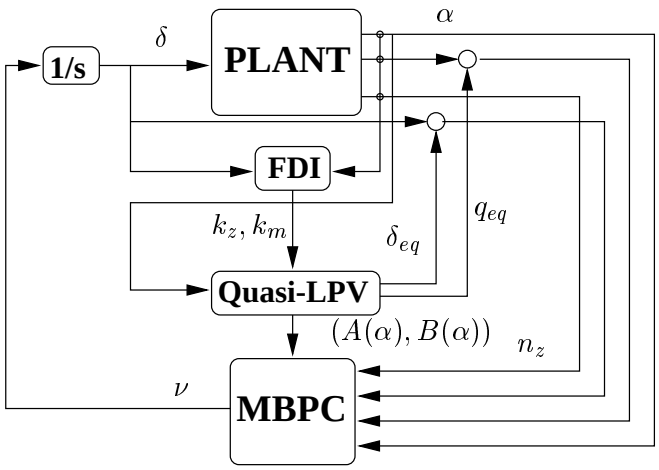


Figure 7.9: The missile autopilot having reconfiguration and scheduling features

The quasi-LPV model runs in parallel with the nonlinear plant providing the states equilibrium values $q_{eq}(\alpha)$, $\delta_{eq}(\alpha)$ together with the $A(\alpha)$, $B(\alpha)$, C, D matrices depending on the current value of the parameter α.

The structure of the $MBPC$ controller consists of standard modules. States, including α, are assumed measurable, since they are usually available in flight control. We use a state-space LTI model for the plant, different at each time step and corresponding to the parameter value α.

The other alternative, of on-line point-wise linearisations which was used in the HIRM case is depicted in Figure 7.6. Here the strategy is the same as in the missile case with the exception of the way the $MBPC$ internal model is produced.

Prediction strategies

The main question raised while implementing the adaptive mechanism was which model should be used to provide predictions. Two strategies have been considered for use during manoeuvres which cause significant α variations:

- No *a priori* trajectory information. A single model was used across the whole prediction horizon (*ie* for $l = 1, \ldots, N_y$) but changed at each current time step (*ie* different for each k) in accordance with the measured α value.

- *A priori* trajectory information available. This allows the internal model to vary over the prediction horizon, but one needs to predict α over the prediction horizon in order to do this. It is important to base this prediction on the commanded trajectory, rather than the achieved one; this is more approximate, but it retains the QP structure of the optimisation problem. Basing it on the achieved trajectory would lose even the convexity property, so should be avoided if at all possible [CKC96]. For the missile it is possible to obtain a prediction for α, based on the commanded trajectory for n_Z.

Note that these two strategies become the same in the special case when $N_y = N_u = 1$. In our evaluations we have simulated only the first prediction strategy.

It is advisable to take one of the above suggested paths. Otherwise, obtaining the models through the plant future dynamics, the resulting cost function is no longer quadratic in the command increment. This results in the standard $MBPC$ quadratic optimisation being transformed into a nonlinear constrained optimisation for which it is hard to guarantee a global solution [CKC96].

Designing the missile controller for the unfailed condition

The controller design followed the conventional procedure used for $MBPC$ design (see Chapter 4). This involved tuning the $MBPC$ controller for a plant model obtained by freezing the healthy quasi-LPV model at $\alpha = 0$. Note that this model is unstable. The plant model used for control had the following continuous time transfer function:

$$\frac{n_Z}{\nu} = \frac{(s - 31.58)(s + 31.30)}{s(s + 6.0029)(s - 5.3984)}$$

A sampling time of $T_s = 0.01s$ will provide a good approximation over the system bandwidth.

Horizon dimensions were defined according with the general guide lines used in tuning $MBPC$ controllers [HM97c, HM96b] and Chapter 4. The tuning was performed in the unconstrained case.

The control horizon was chosen to be the same as the number of states ($N_u = 3$ in this case) which will ensure stability even in the worst case, namely when all the modes are unstable, should that ever occur [RM93].

The small sampling time has determined a long prediction horizon of $N_y = 30$ in order to predict $0.3s$ ahead. This was enough to ensure the performance criteria required [SC93].

The $MBPC$ control weighting matrix had to be made relatively small $R = 0.005$ due to the small effectiveness of the fin in this particular model. The constraint predictive controller tracking weighting matrix $Q = 1$ reflects the requirements on the time response characteristics (rise time, settling time and overshoot) for the closed loop system. These were obtained after some trial and error, starting with unit values for both weighting matrices.

The next step in design, after checking the unconstrained closed loop behaviour, was to enforce the constraints and check the constrained $MBPC$ performance. Once these steps were passed the controller was checked when performing adaptation with respect to α.

Designing the HIRM controller for the unfailed condition

As in the missile case the controller design followed the general procedure which we have developed for conventional $MBPC$ control [MBT97, HM96b]. This involved tuning the $MBPC$ controller for a plant model obtained by linearising the healthy high-fidelity model for an equilibrium point corresponding to a level flight with Mach number $M = 0.30$ at a hight of $H = 5000\ ft$ with parametric uncertainties in M of 3%.

A sampling time of $T_s = 0.05s$ provided for the discrete time model a good approximation of the continuous time one over the whole open-loop bandwidth (the smallest

constant which appears in the model is on the transfer function from symmetrical taileron (δ_{TS}) to pitch rate (q) and it is defined by a cross-over of about $10rad/s$).

The HIRM longitudinal model contains actuator models for which the bandwidths are at $60rad/s$, $10rad/s$ and $1rad/s$ for canards, taileron and engines, respectively.

In order to speed up the process the tuning was performed first for the unconstrained case. The control horizon was chosen to be the same as the number of states $N_u = 11$ but the small sampling time led to a long prediction horizon of $N_2 = 60$ in order to predict 3 s ahead. This was enough to ensure the achievement of the performance criteria.

The $MBPC$ control weighting matrix had to be made relatively big
$R = \begin{bmatrix} 9 \times 10^4 & 3 \times 10^4 & 1 \times 10^4 \end{bmatrix}$, a typical choice for longitudinal channel of the aircraft.

The $MBPC$ tracking weighting matrix $Q = \begin{bmatrix} 700 & 1 \end{bmatrix}$ reflects the requirements on the time response characteristics (rise time, settling time and overshoot) for the closed loop system. These were obtained after some trial and error, starting with unit values for both weighting matrices.

7.2.4 Automatic Tuning for $MBPC$ During Reconfiguration

In [Mac97, HM98a, HM98c] and the previous section a particular combination of high fidelity models, fault detection and isolation (FDI), model approximation/simplification and constrained predictive control which enables a generic solution to reconfiguration and adaptation was described.

As will be shown in Section 7.2.5 it is not too simplistic to believe that it is enough to change the internal model and the constraints limits in order to reflect a fault for which the $MBPC$ controller has to provide satisfactory control inputs. But, in the most general case the controller needs to be re-tuned in order to achieve satisfactory performance. "Tuning" here means the adjustment of the $MBPC$ cost function weights and possibly the relaxation of some of the constraints in order to make the problem feasible.

Therefore, in this section, on-line optimisation to provide the tuning parameters that will enable the reconfigured controller to provide good performance is investigated [HM98b]. In Section 7.2.5 a practical problem of reconfiguring a missile on-line in the presence of major impairments is solved.

Matching the reference closed loop eigenvalues via on-line optimisation

Our main goal is to have the same responses for the impaired system as the ones achieved by the healthy plant and its controller. Before discussing the optimisation problem involved in solving the matching of the two closed loops, we would like to point out two assumptions made for both Section 7.2.4 and 7.2.4.

First one assumes the horizons as fixed parameters for both the nominal and the impaired system cost functions. In fact horizon dimensions are defined according with the general guide lines [Soe92] used in tuning $MBPC$ controllers. The control horizon was chosen to be the same as the number of model states which will ensure stability even in the worst case when all the plant modes are unstable [RM93]. The prediction horizon is determined by the sampling time and the performance to be reached with the closed loop.

The second assumption refers to the order of the impaired system. Because in our case the impaired model is obtained employing the high fidelity model written in the quasi-LPV form [HM98a] the number of states is considered to be the same in spite of major structural failures.

Authors like [GA91] applying pseudo-inverse reconfiguration methods, translate the matching of the healthy and impaired system responses into the goal of keeping the closed loop eigenvalues of the impaired system in the same region as in the nominal case, if possible.

Initially, we matched the $MBPC$ controller by performing a Frobenius norm optimisation of the form:

$$\min_{\Theta} \|K_f(\Theta) - K_n\|_F$$

The results obtained drove us to the conclusion that a matching of the closed-loop eigenvalues with the ones corresponding to the nominal closed-loop will be more appropriate. As result the following optimisation was employed:

$$\min_{\Theta} \|\bar{A}_f(\Theta) - \bar{A}_n\|_F$$

The minimisation was carried out using the `fminu` function from the Matlab Optimisation Toolbox, a nonlinear optimisation algorithm based on quasi-Newton method using a mixed quadratic and cubic search procedure. This was employed due to the particular nonlinear dependency of A_f on Θ.

The form of the cost function (2.10) depending on Q and R explains the existence of a non-unique solution. In other words, for the SISO case, the non-uniqueness of the solution can be explained observing that the actual parameter in the optimisation problem is the ratio between the weights Q and R.

The reason of choosing the Frobenius norm as a measure for the distance between the two closed loop matrices is not only related to the optimisation feasibility but as well to stability guarantees that we provide when the reconfigured controller is employed. In the case of a small perturbation upon the eigenvalues of the closed loop (A_f) from the nominal closed loop (A_n) the Bauer-Fike theorem gives a condition for the location of the eigenvalues of the reconfigured closed loop. We include the proof because in [GL96] the theorem was stated and proved only for p-norms. The stability guarantees are restricted to the assumption that A_n is non defective in the sense that it can be reduced to a diagonal form by a similarity transformation.

Theorem 7.2.1 *If μ is an eigenvalue of the matrix $A_f \in \mathbb{R}^{q\times q}$, $X^{-1}A_nX = $ $= diag([\lambda_1, \ldots, \lambda_q]) = D$ where X is the matrix of the A_n eigenvectors, $k_2(X)$ is the X matrix condition number and λ is an eigenvalue of A_n ($\lambda \in \lambda(A_n)$) then:*

$$\min_{\lambda\in\lambda(M_n)} |\lambda - \mu| \leq k_2(X) \parallel E \parallel_F \tag{7.5}$$

where $E = A_f - A_n$

Proof It is enough to consider the case when μ is not in $\lambda(A_n)$. The matrix $X^{-1}(A_f - \mu I)X$ is singular, then applying the matrix inversion lemma so is $I + (D - \mu I)^{-1}X^{-1}(A_f -$

$A_n)X$. Note that if $I + M$ is singular at least one of the eigenvalues of M is -1 which means that $\|M\|_F \geq \|M\|_2 \geq \bar{\sigma}(M) \geq |-1|$. Thus, using the modified sub multiplicative property for the Frobenius norm written in its two different forms $\|AB\| \leq \|A\|_2\|B\|_F$ and $\|AB\| \leq \|B\|_2\|A\|_F$ we have:

$$\begin{aligned} 1 &\leq \|(X^{-1}A_nX - \mu I)^{-1}(X^{-1}EX)\|_F \\ &\leq \|(X^{-1}A_nX - \mu I)^{-1}\|_2\|(X^{-1}EX)\|_F \\ &\leq \min_{\lambda\in\lambda(A_n)} \frac{1}{|\lambda-\mu|}\|X^{-1}\|_2\|EX\|_F \\ &\leq \min_{\lambda\in\lambda(A_n)} \frac{1}{|\lambda-\mu|}\|X^{-1}\|_2\|X\|_2\|E\|_F \\ &\leq \min_{\lambda\in\lambda(A_n)} \frac{1}{|\lambda-\mu|}k_2(X)\|E\|_F \end{aligned}$$

□

Basically the theorem defines a disk, centred on the old eigenvalue, in which a new eigenvalue will be located once the optimisation performed. This bound is given by the condition number of the eigenvector matrix of A_n and the value of the cost function.

The eigenvalues of A_f lie in the union of disks $|\lambda - \mu|$ having a radius $k_2(X) \parallel E \parallel_F$. We recommend a nominal design that will place the nominal closed loop eigenvalues centred within the open unit disk ensuring an increased level of robust stability to the system. Using the above procedure, for a good conditioning of eigenvectors of the nominal A_n, the minimisation performed provides a closed loop behaviour of the reconfigured system close to the nominal one. The increased control effort used, that requires enough actuator authority, might be considered as a disadvantage of the current design procedure which made us to consider the solution presented in the next section.

If $k_2(X)$ is large then small changes in A_f can induce large changes in the closed loop eigenvalues. In the case of a practical problem a poor conditioning of the matrix of eigenvectors of A_n (X) is possible. This is a problem located at the level of the nominal design. Further research has to be carried out in order to provide a good conditioning for the $k_2(X)$ number when designing the $MBPC$ controller for the nominal plant.

Despite the stability guarantees and a relatively simple optimisation procedure the above approach suffers from the fact that only the closed-loop poles are matched but not the zeros, which are known to have as well a big effect on the system response.

Control reconfiguration automatic design procedure

The natural evolution from the previous way of addressing the problem of reconfiguration was to solve a minimisation which accounts for both closed-loop poles and zeros matching the system four-block transfer matrix.

The automatic tuning procedure employed as part of the overall control mechanism follows the following steps:

1. A nominal controller is designed for the given performance specifications having the nominal realisation $\left[\begin{array}{c|c} A_n & B_n \\ \hline C_n & D_n \end{array}\right]$ as the internal model.

2. Given a plant model $\left[\begin{array}{c|c} A_f & B_f \\ \hline C_f & D_f \end{array}\right]$ reflecting an impairment a closed loop realisation is produced depending on the elements of the cost function weighting matrices Q and R which we denote by the parameter vector Θ.

3. A minimisation procedure is performed with respect to Θ of the ∞-norm of the difference between the transfer matrix defining the nominal reference closed loop $\hat{M}_n\left(\left[\begin{array}{c|c} A_n & B_n \\ \hline C_n & D_n \end{array}\right]\right)$ and the transfer matrix $\hat{M}_f\left(\left[\begin{array}{c|c} A_f & B_f \\ \hline C_f & D_f \end{array}\right], \Theta\right)$ which is defined on the basis of the impaired model controlled by the $MBPC$, depending on the parameter Θ.

The procedure can be frequency weighted via W_i and W_o, as common in model matching setups, leading to the following optimisation problem with respect to the parameter Θ:

$$\min_{\Theta} \left\| W_o \left[\hat{M}_f \left(\left[\begin{array}{c|c} A_f & B_f \\ \hline C_f & D_f \end{array}\right], \Theta \right) - \hat{M}_n \left(\left[\begin{array}{c|c} A_n & B_n \\ \hline C_n & D_n \end{array}\right] \right) \right] W_i \right\|_{\infty} \tag{7.6}$$

The algorithm chosen to perform this minimisation is based on a fairly brute force procedure but in exchange it provides feasibility together with good control of the level of matching. This procedure proved successful when severe impairments were encountered.

Therefore here we include the steps of the algorithm which produces a solution for Θ that gives the best matching:

1. Perform a gridding of the parameter space Θ with an appropriate step.

2. Compute for the current value of the parameter the value of the cost function from (7.6).

3. Store the value of Θ and the corresponding norm if smaller than figures previously obtained.

4. Iterate Steps 2 and 3 for all the grid vertices.

7.2.5 Evaluation results

Evaluating the scheduling and reconfiguration capabilities of the missile controller

The two different capabilities of the controller – scheduling on α and reconfiguration – are investigated in the following simulations.

Firstly we show the behaviour of the closed loop with the different controllers when the system is subjected to a step of 0.1 in the non-dimensional controlled variable n_Z.

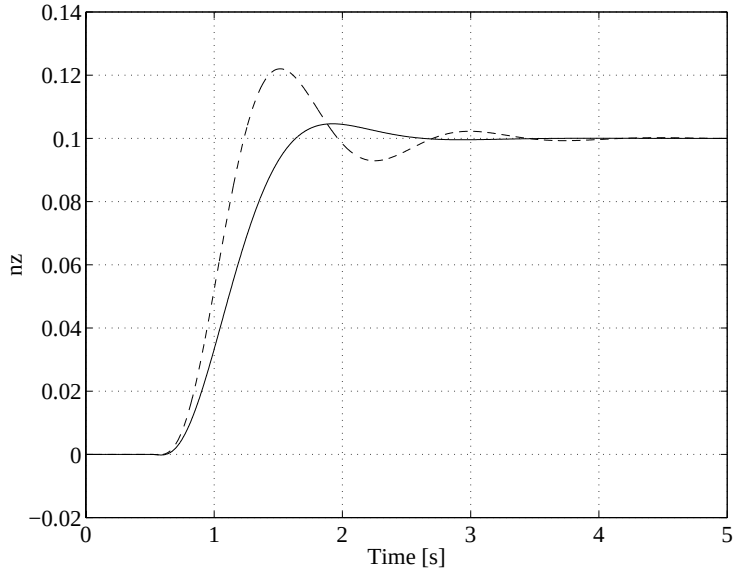


Figure 7.10: Comparison between the fixed and the adaptive $MBPC$ (adaptive–solid, fixed–broken)

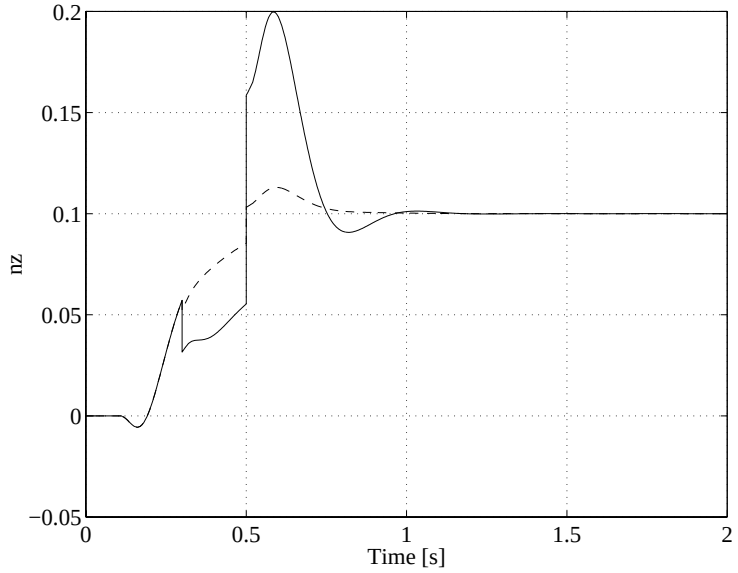


Figure 7.11: The reconfiguration capabilities provided by the new approach (update internal model–broken, nominal internal model–solid)

The first controller is represented by the nominal design for which a fixed internal model, defined at $\alpha = 0$ is used, even though α varies during the manoeuvre.

In the second case the controller uses the fixed values for the cost function tuning parameters (N_y, N_u, Q, R), but the internal model is updated to follow the evolution of

α, as described in Section 7.2.1

For each such case we have ensured stability by an adequate choice of the control horizon (N_u greater than the system order). In this first case no failures are assumed for the controlled system.

The response of the system controlled by the adaptive $MBPC$ provides less overshoot and a shorter settling time as demonstrated in Figure 7.10.

In the second simulation we show the reconfiguration capability of the new approach. Successive structural failures of the missile were simulated: a failure on the body at $t =$ 0.3s, and one on the fin at $t = 0.5$s. These failures modified the aerodynamic characteristics by 10% and 50%, respectively. Figure 7.11 shows how the controller coped with each failure.

The failure was simulated by manipulating the nonlinear model dynamics. The impairment was represented in the high fidelity model by changing the coefficients relating the contributions of various aerodynamic surfaces to C_Z and C_m.

In both situations the new controller behaved well, providing scheduling and reconfiguration as required.

Simulation results for several failure scenarios in the HIRM case

Actuator failures

One of the most frequent failures encountered in real life is actuator failure. The reduction of control authority leads to reduced performance, or even instability.

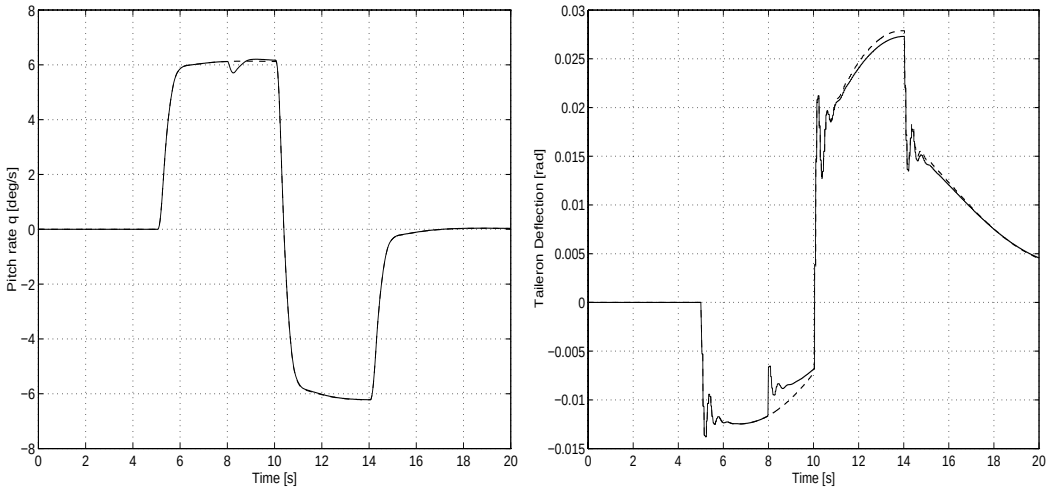


Figure 7.12: The reconfiguration capabilities provided by the new approach in the case of actuator failures (nominal – solid, stuck canard – broken)

The simulation result shown in Figure 7.12 shows the system (HIRM) response when the canard actuator failed during a pitch rate demand of 6 deg/s. The step in pitch rate demand occurred at time 5 s followed by the failure, a stuck symmetrical canard, which occurred at time 8 s.

This failure, was represented by tightening the constraints on the allowed canard deflection in the $MBPC$ optimisation. Due to the $MBPC$ feature of handling constraints explicitly no other change was necessary in the structure of the controller.

The solid line in Figure 7.12 shows the response when no information about the failure is passed to the controller.

By looking at the performance criteria stated in the GARTEUR Design Challenge [MBT97], for this manoeuvre, we consider the controller performance in the presence of the failure to be satisfactory.

Structural failures

For a fighter aircraft, we can assume that one of the typical failures that its controller has to cope with is battle damage which, in general, falls in the class of structural failures.

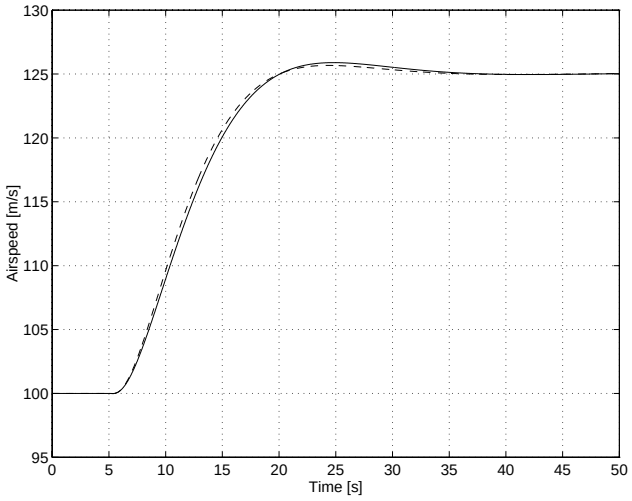


Figure 7.13: The reconfiguration capabilities provided by the new approach in the case of structural failures (nominal – broken, 20 % change in aerodynamic coefficients – solid)

The simplest way to simulate structural failures in the aircraft high fidelity model is by means of the non-dimensional aerodynamic coefficients, which are modified accordingly. For instance a missing part of an aerodynamic surface translates into more drag and a reduced lift, which in their turn are translated into new values of the aerodynamic coefficients. This is according with a study [Hoa78] which relates various types of structural failures to the aerodynamic model.

In our scenario we assume that such a structural failure, which changed the values of the aerodynamic coefficients by 20 %, occurred at time 8 s while a change of 25 m/s in speed set-point was demanded at time 5 s, see Figure 7.13. A fairly large set-point change was chosen as part of the scenario because we want to show the controller fault tolerant features. During this manoeuvre the actuator limits were not reached.

The performance of the reconfigurable flight controller, shown in Figure 7.13, complies with the criteria mentioned in GARTEUR Design Challenge document [MBT97].

Combined failures

Now we consider a simultaneous actuator and structural failure. This is typical of battle damage or of fatigue fracture.

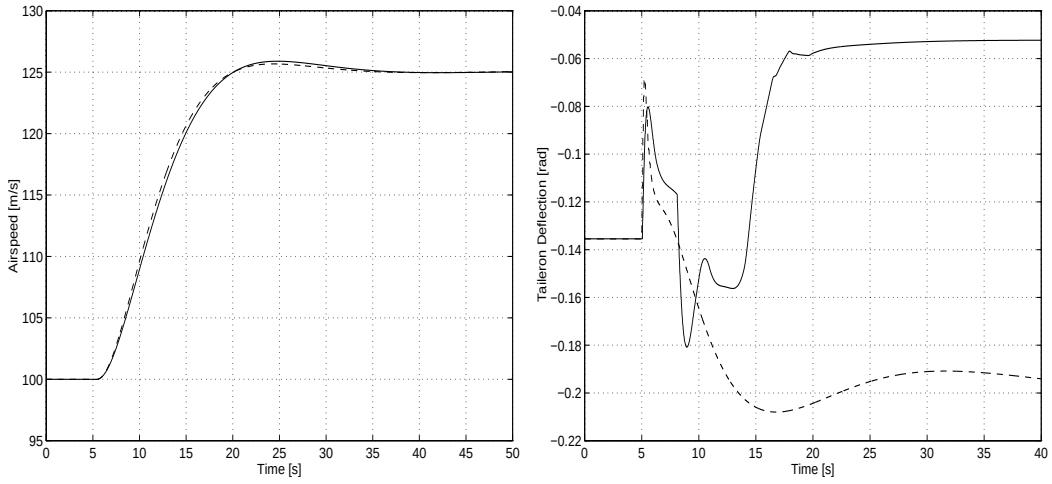


Figure 7.14: The reconfiguration capabilities provided by the new approach in the case of a combined failure (nominal – broken, combined structural and actuator failure – solid)

In Figure 7.14 the performance of the controller is investigated when at time 5 s a change of 25 m/s in the air speed was demanded, followed at time 8 s by an actuator failure and at time 15 s by a structural failure similar to the ones described in Sections 7.2.5 and 7.2.5.

The model chosen to illustrate the algorithm was subjected to several failure scenarios which showed the ability of the controller to meet robust performance criteria mentioned in the GARTEUR [MBT97] and DERA design challenges.

Checking the automatic tuning procedure with the missile model

This work represent a natural extension of the idea described in [HM98a]. In this section we concentrate on the same missile model of which complete details are given in [SC93]. In flight scheduling for the nominal loop and reconfiguration were performed using quasi-LPV high-fidelity models and $MBPC$ control as described in the previous section.

The $MBPC$ internal model is based on the plant having the approximated discrete time transfer function (7.7) obtained for a sampling time of $T_s = 0.01$s. This model was defined at $\alpha = 0$, even though α varies during the manoeuvre, is the same one as in the previous section described now in discrete time:

$$\frac{n_Z}{\nu} = \frac{(z - 1.3732)(z - 0.073)}{(z - 1)(z - 0.9417)(z - 1.055)} \tag{7.7}$$

The structural failure was simulated by manipulating the nonlinear model dynamics. The failure model (7.8) was obtained for values of the parameters k_Z and k_m, see [HM98a],

reflecting a failure that modified the aerodynamic characteristics. The resulting plant model is:

$$\frac{n_Z}{\nu} = \frac{(z-0.77)(z-1.42)}{(z-1)(z-1.12)(z-0.975)} \tag{7.8}$$

As seen from (7.7) and (7.8) major failures lead to a change in the linear internal model. The severity of these impairments requires controller re-tuning.

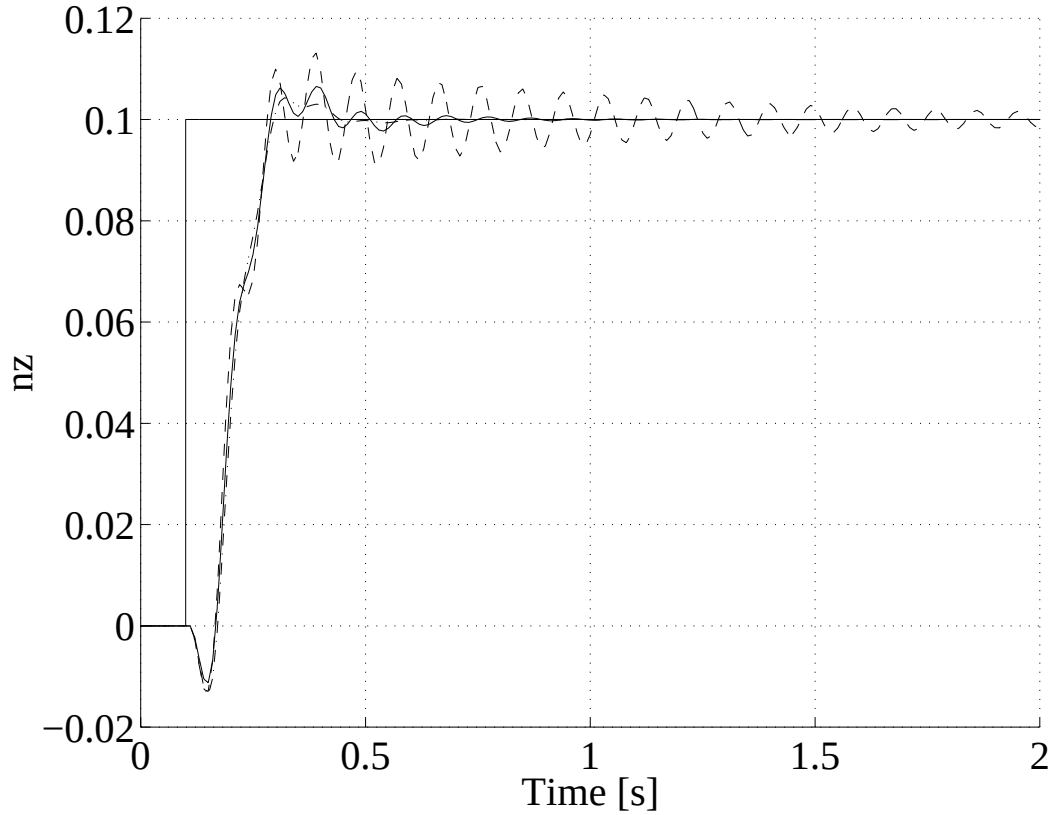


Figure 7.15: Evaluation of the $MBPC$ automatic tuning

The automatic tuning procedure keeps the horizons fixed but adjusts the weights in (2.10) via the optimisation (7.6) performed with respect to Θ. The control horizon was chosen to be the same as the number of states ($N_u = 3$ in this case) which will ensure stability even in the worst case, namely when all the modes are unstable, should that ever occur [RM93]. The small sampling time has determined a long prediction horizon of $N_y = 30$ in order to predict 0.3s ahead. This was enough to ensure the performance criteria required in [SC93]. The nominal values chosen for Q and R were 0.005 and 1, respectively.

The new weights $R = 0.035$ and $Q = 1.12$ obtained from the optimal Θ were computed using the procedure described in the Section 7.2.4. The value for the grid step was in this

case 0.01 but for other cases exhibiting a large ratio between Q and R it has to be much finer. Of course the grid step influences the solution speed.

We show in Figure 7.15 the behaviour of the closed loop with the different controllers when the system is subjected to a step of 0.1 in the non-dimensional controlled variable n_Z.

In Figure 7.15 the dash-dotted line depicts the behaviour of the nominal plant with an $MBPC$ controller based on a nominal internal model and nominal tuning. The broken line shows the impaired plant controlled by the $MBPC$ having information about the failure via the internal model but still using the cost function nominal tuning. The solid line shows the behaviour of the impaired plant this time controlled by the re-tuned $MBPC$ using the newly developed automatic procedure.

7.3 Conclusions

In our case the objective was to find a systematic way of reconfiguring control systems in the event of actuator and structural failures. Therefore we consider that the particular combination of high fidelity models, fault detection and isolation (FDI), model approximation/simplification and constrained predictive control enables a generic solution to reconfiguration and adaptation [Mac97].

Assuming reliable FDI, the proposed approach gives:

- A systematic and compact in some cases way of representing high-fidelity models,
- Automatic updating of models following failure,
- Real-time production of internal models for $MBPC$ control,
- An autopilot structure which integrates the components shown in Figure 7.9.

The exploitation of the quasi-LPV models fits naturally in the $MBPC$ framework. Since no linearisation is involved these models are not limited to standard gain-scheduled designs. Moreover, there is a wide class of systems exhibiting output nonlinearities. They are particularly common in aerospace, but there are also process models which can be expressed in a similar form.

There has been some progress in analysing the stability of certain multi-model control schemes [NB97, Mor93], although it is too early to know whether these approaches will be helpful for our proposal.

The automatic tuning procedure represents an extension to a general reconfiguration strategy described in [HM98a]. Depending on the bounds defined by the ∞-norm computation we can state for a practical problem how big the failure can be to have the method coping with it.

The examples tackled and the failure scenarios suggested are common enough to enable the reader to have a documented opinion about the performance of the fault tolerant method suggested. Of course more things have to be done in this direction by setting up a benchmark in terms of failure scenarios but this initial attempts opened already the path on which we are going to pursue our future research. For a while we are going to

concentrate on the HIRM model but a possible move towards civil aviation aircraft can be foreseen. As a validation scheme the control reconfiguration concepts will use the full nonlinear simulation of the aircraft, the main performance goal being to land the aircraft safely whenever possible.

Although our approach is quite complex, we believe that the complexity is worthwhile if effective scheduling and reconfiguration can be achieved. The results of Section 7.2.5 are encouraging, and papers such as [PCS97] indicate that others also take this view. Addressing the problem in such a manner will have a serious economical impact upon the sales of the aircraft producing companies. Therefore embedding such a fault tolerant system into the aircraft's flight controller will result in less incidents or accidents which will enable open funding of future research.

Chapter 8

Conclusions

8.1 Most significant contributions of the thesis

We will highlight here the contributions of the thesis which we consider most significant to research and development in the field of Model Based Predictive Control ($MBPC$) applied to aerospace problems:

A personal view upon the field of $MBPC$ control Using the experience accumulated during the years of research in Cambridge we were able to have a global view upon the $MBPC$ technique and present various issues connected with. A comparison between several formulations was provided with two aims: first one is to make the reader aware about the equivalence of them and point out their advantages and drawbacks and the second one was concerned with providing the necessary information to make possible the understanding of the features addressed by the other contributions included in the thesis.

General guidelines for $MBPC$ tuning illustrated on various examples Tuning the $MPBC$ controllers is a relatively difficult task that combines theoretical proved rules with the experience of the designer materialised in several rules of thumb. We have tried to deliver this knowledge in an structured manner using several examples, SISO and MIMO systems. The aim of including this part in the thesis was to give the reader the possibility to concentrate on the originality of the ideas presented in the next chapters, assuming the controllers tuning as a secondary task.

A Development Space for $MBPC$ control This part of the thesis answers the question "Why an MBPC development space?". A new software was produced in order to enable us to show in simulation the features of the $MBPC$ controllers. The Matlab – Simulink Graphical User Interface (GUI) enable the user to integrate besides predictive control various other modules such as: aircraft or missile nonlinear models, various scenarios for landing or failures and control reconfiguration features.

A SAS for the RCAM model using $MBPC$ control This contribution consists of applying Model Based Predictive Control (MBPC) to the GARTEUR - RCAM design

challenge. Separate controllers are proposed for the longitudinal and lateral channels, each of these having MBPC in the inner loop and a conventional controller in the outer loop. Emphasis is placed on describing the controller structure and the design process. Main conclusion of the research is that MBPC is not recommended for routine use in flight control, but has good potential for higher level control functions such as: on-board flight management and reconfiguration of controllers in event of damage of the aircraft structure or actuators.

A combined $MBPC/H_\infty$ autopilot for a civil aircraft This development motivates the combined use of an H_∞ loop-shaping controller and $MBPC$ as a method of designing automatic pilots for civil aircraft. The H_∞ loop-shaping controller will provide stability augmentation and guidance. The $MBPC$ controller will act as a flight manager. The design procedure developed was tested by designing an autopilot for the Research Civil Aircraft Model (RCAM) used in the GARTEUR design challenge. The resulting controller was subjected to a standard evaluation procedure. Satisfactory results were achieved.

Reconfiguration and scheduling using high-fidelity models and $MBPC$ Advances towards reconfiguration and scheduling in flight control systems using high fidelity models, Fault Detection and Isolation (FDI), model approximation/simplification and constrained Model Based Predictive Control ($MBPC$) were made. When major failures are encountered in a missile or the HIRM aircraft model controlled by a nominal $MBPC$ controller these are reflected in the internal model via a relatively sophisticated mechanism involving high fidelity models. At this stage an optimisation based on the nominal tuning but updated internal model will provide the required performance from the impaired system. As an additional feature we have introduced the on-line computation of the tuning parameters – the controller cost function weights– which will enable the controller to match, on-line, a reference closed loop. Of course this is an initial attempt affected by problems such as the inability to ensure a real time implementation. This algorithm is applied to provide the $MBPC$ tuning only in the case of major structural impairments.

8.2 Recommendation for future research

- The Development Space represents an alternative to the Matlab MPC toolbox which can be used in various situations from demonstrating the features of $MBPC$ to the development of the algorithm involving new on-line optimisation and computation tools. Making it to include: exponential weighting of the cost function, an identification module that will use subspace methods, infinite horizon formulation with constraints, variable step times into the future, nonlinear optimisation will complete this software.

- In the case of the combined $MBPC/H_\infty$ autopilot the optimality of the solution has to be investigated. As long as in practice the lack of interference between the two controllers was shown now a theorem stating in the general case this property for

the multi-rate system considered is necessary in order to offer a complete solution to the problem.

- The reconfiguration and scheduling in flight using quasi-LPV high-fidelity models and $MBPC$ control has to be tested on several systems. Once this stage will be passed the set including the systems for which reconfiguration can be addressed in this way can be defined.

- Regarding the automatic procedure suggested, it would be useful to know its limitations and ways of tuning it for best performance. The value of the cost function minimised is an indicator of how close the controlled impaired system to the nominal one is. It is worth while investigating a different method of minimisation such as to account a priori for such a measure. This is because we cannot expect the impaired plant to provide the same performance as the nominal closed loop.

- The use of the automatic tuning feature can be also directed towards a tuning of the $MBPC$ controller such as to approximate an ideal model or other closed loops including different controllers.

Appendix A

A.1 setup.m

```
function [error_function] = setup(Ts_NNu_NN2,mml_mmu_rrl_rru,nnl_nnu,
R_small,Q_small,pr);

global ll mm nn mu_sys ns_sys po_sys NNu NN2 Uo Xo Yo X_pred A_new
 B_new D_new E_new Ts J parameter fl precision setpoint A_matrix
B_matrix C_matrix D_matrix;

% Reading procedure of the values precised in the graphic interface

Ts  = Ts_NNu_NN2(1);
NNu = Ts_NNu_NN2(2);
NN2 = Ts_NNu_NN2(3);
parameter =Ts_NNu_NN2(4);
fl =Ts_NNu_NN2(5);

mml = mml_mmu_rrl_rru(1,:)';
mmu = mml_mmu_rrl_rru(2,:)';

rrl = mml_mmu_rrl_rru(3,:)';
rru = mml_mmu_rrl_rru(4,:)';

nnl = nnl_nnu(1,:)';
nnu = nnl_nnu(2,:)';

precision = pr;

load discrete_model;

% Computation of orginal system dimensions
mu_ini = size(BB,2);
ns_ini = size(AA,1);
po_ini = size(CC,1);

% Feasibiliy of the algorithm
stop=0;

if NNu>NN2
  error('??? We cannot operate MBPC algorithm with such settings
for horizons !!');
end
if NN2-NNu+1<3
  error ('??? We cannot operate MBPC algorithm with such settings
for horizons !!');
end
if NNu-1<1
```

```
  error('??? We cannot operate MBPC algorithm with such settings
for horizons !!');
end
if NN2<1
  error('??? We cannot operate MBPC algorithm with such settings
for horizons !!');
end

% 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,po_ini)];

B = [    BB;
      CC*BB];

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

D = DD;

A_matrix = A;
B_matrix = B;
C_matrix = C;
D_matrix = D;

% Computation of augmented system dimensions
mu_sys = size(B,2);
ns_sys = size(A,1);
po_sys = size(C,1);

if stop == 0
% Constraints values
% mm = constraints upon inputs
mm1 = [-mml  ; -mml];
for k = 3:NNu
  mm1 = [mm1 ; -mml];
end;
mm = mm1;
mm1 = [+mmu  ;+mmu];
for k = 3:NNu
  mm1 = [mm1 ; +mmu];
end;
mm = [mm ; mm1];

% ll = constraints upon rate of inputs change
rr1 = [-rrl  ; -rrl];
for k = 3:NNu
  rr1 = [rr1 ; -rrl];
end;
ll = rr1;
rr1 = [+rru  ; +rru];
for k = 3:NNu
  rr1 = [rr1 ; +rru];
end;
ll = [ll ; rr1];

% nn = constraints upon states
nn1 = [-nnl  ; -nnl];
for k = 3:NN2
  nn1 = [nn1 ; -nnl];
end;
nn = nn1;
```

```
nn1 = [+nnu ; +nnu];
for k = 3:NN2
  nn1 = [nn1 ; +nnu];
end;
nn = [nn ; nn1];

% Kalman Filter matrix computation
% Kalman filter with estimated states
% KK = dlqe(A,0.001*eye(ns_sys,mu_sys),C,10000*eye(mu_sys),10*eye(po_sys));
% Kalman filter with measurable states
KK = zeros(ns_sys,po_sys);

% Computaion of matrices involved in the MBPC algorithm
[A_new,B_new,D_new,E_new] =
algorithm(ll,mm,nn,A,B,C,D,KK,NNu,NN2,mu_sys,ns_sys,po_sys,R_small,Q_small);

A_new = sparse(A_new);
B_new = sparse(B_new);
D_new = sparse(D_new);
E_new = sparse(E_new);
end
error_function = [];
```

A.2 algorithm.m

```
function [K_mbpc, L_mbpc, R_mbpc] =
algorithm(Z1,Z2,Z3,Z4,Z5,Z6,Z7,Z8,Z9,Z10,Z11,Z12,Z13,Z14,Z15,Z16)

inargs =
'(ll_,mm_,nn_,A_,B_,C_,D_,KK_,NN_u,NN_2,mu_sys_,ns_sys_,po_sys_,
R_small_,Q_small_,para)';
eval(mkargs(inargs,nargin,'ss'));

% 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 j = 3:NN_2
  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 j = 3:NN_2
  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 j = 3:(NN_2-NN_u+1)
  G_aux_1 = A_*G_aux_1;
  G_col_aux = [G_col_aux;
              G_aux_1  ];
end

G_col = G_col_aux;

for j = 1:(NN_u-1)
  G_col = [Z    ;
    G_col];
end

G_matrix = [G_col G_matrix];

for q = 2:(NN_u-1)

  G_aux_1 = A_*G_aux_1;
  G_col_aux = [G_col_aux;
    G_aux_1  ];
  G_col = G_col_aux;
  for j = 1:(NN_u-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 = 2:NN_2
  Z_l = zeros(po_sys_,(ns_sys_*(q-1)));
  Z_u = zeros((po_sys_*(q-1)),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:NN_u

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:NN_u-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
Q1 = Q_small_;
for k = 2:NN_2
  Q1 = [Q1;Q_small_];
end
Q = diag(Q1);

% R_new computation
R1 = R_small_;
for k = 2:NN_u
  R1 = [R1;R_small_];
end
R = diag(R1);

% 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(NN_2*po_sys_)]'*Q*C_new*G_new;

% L_new computation
L_new = [-eye(NN_u*mu_sys_);
          eye(NN_u*mu_sys_)];

% M_new computation
M_new = L_new;

% N_new computation
N_new = [-eye(NN_2*ns_sys_);
          eye(NN_2*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*NN_u*mu_sys_,c1) , zeros(2*NN_u*mu_sys_,c2) ,
zeros(2*NN_u*mu_sys_,c3) , eye(2*NN_u*mu_sys_,sl) ,
zeros(2*NN_u*mu_sys_,sm) , zeros(2*NN_u*mu_sys_,sn);
         (-M_new*I_new) ,    zeros(2*NN_u*mu_sys_,c2) ,
zeros(2*NN_u*mu_sys_,c3) , zeros(2*NN_u*mu_sys_,sl) ,
eye(2*NN_u*mu_sys_,sm)   , zeros(2*NN_u*mu_sys_,sn);
         zeros(2*NN_2*ns_sys_,c1) , (-N_new*F_new) ,
(-N_new*H_new) , zeros(2*NN_2*ns_sys_,sl) ,
zeros(2*NN_2*ns_sys_,sm)    , eye(2*NN_2*ns_sys_,sn)];

% Generation of selection matrix
Sel = eye(mu_sys_);
Sel_aux = zeros(mu_sys_);

for i = 1:NN_u-1
Sel = [Sel, Sel_aux];
end

% Additional computations
A_inv = inv(A_new_);
D_t = D_new_';

% Sigma2 computation
S_new_1 = [  eye(ns_sys_,ns_sys_), zeros(ns_sys_,po_sys_),
 zeros(ns_sys_,po_sys_)];
S_new_2 = [zeros(po_sys_,ns_sys_),   eye(po_sys_,po_sys_),
 zeros(po_sys_,po_sys_)];
S_aux_3 = [zeros(po_sys_,ns_sys_), zeros(po_sys_,po_sys_),
 eye(po_sys_,po_sys_)];
S_new_3 = [];

for i = 1:NN_2
S_new_3 = [S_new_3; S_aux_3];
end

S_new = [S_new_1; S_new_2; S_new_3];

K_mbpc = Sel*A_inv*(Q*C_new*G_new)';
L_mbpc = [C_new*F_new C_new*H_new];
S_new_mod = [];
for i = 1:NN_2
S_new_mod = [S_new_mod; eye(po_sys_,po_sys_)];
end
R_mbpc = S_new_mod;
```

A.3 optimiser.m

```
function [sys,x0] =
optimiser(t, x, u, flag, ll, mm, nn, mu_sys, ns_sys, po_sys, NNu, NN2,
Uo, Xo, Yo, X_pred, A_new, B_new, D_new, E_new, Ts, parameter, fl)

global ll mm nn mu_sys ns_sys po_sys NNu NN2 Uo Xo Yo X_pred A_new
B_new D_new E_new Ts J parameter fl precision setpoint;

% Sample and offset times
offset = 0;
ts = Ts;
```

```
if abs(flag) == 1 % return derivatives
  sys = [];

elseif abs(flag) == 2 % return discrete states in sys
  sys = [];

elseif flag ==3 % return outputs

  % Is it a sample hit (within a tolerance of 1e-8) ?
  if abs(round((t-offset)/ts)-(t-offset)/ts)<precision
    time = u(1);
    U = u(2:(mu_sys+1));
    Y = u((mu_sys+2):(mu_sys+po_sys+1));
    X_pred1 = u((mu_sys+po_sys+2):(mu_sys+ns_sys+po_sys+1));
    comm = u((mu_sys+ns_sys+po_sys+2):(mu_sys+ns_sys+po_sys+po_sys+1));
    if t == 0
      J = [];
      U = Uo;
      Y = Yo;
      X_pred1 = X_pred;
      comm = Yo;
    end

    S = setpoint(((t*po_sys)+1):((t*po_sys)+(NN2*po_sys)));
    % QP initialization
    ccc = [X_pred1' Y' S']*B_new;
    vvv = B_new'*[X_pred1; Y; S];
    bbb = E_new*[U;X_pred1;Y;ll;mm;nn];
    AAA = D_new;
    QQQ = A_new;

  if fl == 0

      % Least square approximation (unconstrained approximation)
      % The analytic solution in unconstrained case is computed at every step
      % Delta_U_0 = -0.5*(1/parameter)*inv(A_new)*B_new'*[X_pred1' Y' S']';
      Delta_U_0 = -0.5*inv(A_new)*vvv;
      J1 = Delta_U_0'*QQQ*Delta_U_0+ccc*Delta_U_0;
      Delta_U = Delta_U_0(1:mu_sys);

    elseif fl == 1

      % Matlab QP solver, constrained case
      Delta_U_0 = qp(2*QQQ,vvv,AAA,bbb);
      Delta_U_QP = Delta_U_0;
      % Delta_U_0 = -0.5*inv(A_new)*vvv;
      % Delta_U_0 = qp(2*QQQ,vvv,[],[],[],[],Delta_U_0);
      J1 = Delta_U_0'*QQQ*Delta_U_0+ccc*Delta_U_0;
      Delta_U = Delta_U_0(1:mu_sys);

    elseif fl == 2

      % C Ansi QP solver, constrained case
      Delta_U_0 = lqp2(2*sparse(QQQ),vvv,sparse(-AAA),-bbb,[],[],[],0);
      J1 = Delta_U_0'*QQQ*Delta_U_0+ccc*Delta_U_0;
      Delta_U = Delta_U_0(1:mu_sys);

    elseif fl == 3

      % MWLS solver, constrained case
      V = eye(NNu*mu_sys);
      gamma = 4*NNu*mu_sys+2*NN2*ns_sys;
      W = eye(gamma);
```

```
      big_x = ones(gamma,1);
      zet = [U; X_pred1; Y; ll; mm; nn];
      Delta_u_opt = -0.5*inv(A_new)*vvv;
      ei = D_new*Delta_u_opt-E_new*zet+big_x;
      eii = ei;

      No_Optimization = 0;
      if No_Optimization == 0
        if norm(ei,'inf') <= 1
          while norm(eii,'inf') > 1
            den=0;
            for i = 1:gamma
              den=den + W(i,i)*abs(ei(i));
            end;
            for i = 1:gamma
              W(i,i) = W(i,i)*abs(ei(i))/den;
            end;
            V = V/den;
            H = (A_new^0.5)'*V*(A_new^0.5)+(D_new)'*W*(D_new);
            b = -2*Delta_u_opt'*(A_new^0.5)'*V*(A_new^0.5)+2*(-E_new*zet+big_x)'*W*D_new;
            Delta_u_opt = -0.5*inv(H)*b';
            eii = D_new*Delta_u_opt-E_new*zet+big_x;
            ei= eii;
          end;
        end;
      end;
      Delta_U_0 = Delta_u_opt;
      J1 = Delta_U_0'*QQQ*Delta_U_0+ccc*Delta_U_0;
      Delta_U = Delta_U_0(1:mu_sys);
    else

    end
    J = [J;J1];
    sys = Delta_U;

    if abs(sys) < 1e-14*ones(mu_sys,1)
     sys = zeros(1,mu_sys);
   end

 else

    sys = zeros(1,mu_sys);

  end

elseif flag == 4 %                      retun next sample hit
  % ns stores the number of samples
  ns = (t-offset)/ts;
  % this is the time of nextsample hit
  sys = offset+(1+floor(ns+1e-13*(1+ns)))*ts;

elseif flag == 0,    %  return initial conditions and size informations
  sys = zeros(6,1);
  sys(1) = 0;                            % number of continous state
  sys(2) = 0;                            % number discrete state
  sys(3) = mu_sys;                       % number of system outputs
  sys(4) = 1+mu_sys+po_sys+ns_sys+po_sys;% number of system inputs
  sys(5) = 0;                            % no roots
  sys(6) = 0;                            % no direct feedthrough
  x0 = [];
else
  sys = [];
end
```

Bibliography

[All94] 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, 1994.

[AM71] B.D.O. Anderson and J.B. Moore. *Linear Optimal Control*. Enlewoods Cliffs, Prentice Hall, 1971.

[ANA96] G. Obinata A. Naganawa and K. Aida. A Design Method of Generalized Predictive Control System Based on Parametrization of Two-degree-of-freedom Integral Controllers. In 35^th *Conference on Decision and Control*. Kobe, Japan, 1996.

[AP92] J.C. Allwright and G.C. Papavasiliou. On linear programming and robust model predictive control using impulse-responses. *Systems and Control Letters*, 18:159–164, 1992.

[Bad97] T.A. Badgwell. Robust Model Predictive Control of Stable Linear Systems. *International Journal of Control*, 1997. To Appear.

[BGFB94] S. Boyd, L.El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities*. SIAM, 1994.

[BGW90] R.R. Bitmead, M. Gevers, and V. Wertz. *Adaptive Optimal Control: The Thinking Man's GPC*. Prentice-Hall, 1990.

[BKC92] 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, 1992.

[Bla95] M. Blanke. Aims and Means in the Evolution of Fault-tolerant Control. In *European Science Foundation Workshop COSY*, September 1995.

[Bod97] M. Bodson. Multivariable Adaptive Algorithms for Reconfigurable Control. *IEEE Transactions on Control Systems Technology*, 7(2):217–227, Mar. 1997.

[BS93] P. Bendotti and M.M. Saad. A skid-to-turn missile autopilot design: the generalised predictive adaptive control approach. *International Journal of Adaptive Control and Signal Processing*, 7:13–31, 1993.

[BT78] C.B. Borosilow and T.M. Tong. The structure and dynamics of inferential control systems. *AIChE*, 24:492–500, 1978.

[BV84] P.M. Bruijn and H.B. Verbrunggen. Model Algorithmic Control (MAC) using impulse respeonse models. *Automatica*, 25(2):69–74, 1984.

[CG97] M.W. Cantoni and K. Glover. H_∞ Sample-data Synthesis and Related Numerical Issues. *Automatica*, 33(12):2233–2241, 1997.

[Cha84] P.R. Chandler. Self repairing flight control system reliability and maintanaibility program executive overview. In *Proceedings to National Aerospace Electronics Conference*, pages 586–590, Dayton, OH, 1984.

[CKC96] C.M. Chow, A.G. Kuznetsov, and D.W. Clarke. Using multiple model predictive control. In *Proceedings of the spring school "Adaptive and Predictive Control"*, Oxford, 21–22 Mar. 1996. Oxford University.

[Cla93] D. Clarke. *Advances in Model-Based Predictive Control*, pages 3–21. Oxford University Press, 1993.

[CM89a] D.W Clarke and C. Mohtadi. Properties of Generalized Predictive Control. *Automatica*, 25(6):859–875, 6 1989.

[CM89b] D.W. Clarke and C. Mohtadi. Properties of generalized predictive control. *Automatica*, 25(6):859–875, 6 1989.

[CMP93] P. Chandler, M. Mears, and M. Patcher. On-Line Optimising Networks for Reconfigurable Control. In *Proceedings of the 32nd Conference on Decision and Control*, pages 2272–2277. IEEE, 1993.

[CMT87] D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalized Predictive Control – Part I. The Basic Algorithm. *Automatica*, 23(2):137–148, 2 1987.

[CPM95] P.R. Chandler, M. Patcher, and M. Mears. System Identification for Adaptive and Reconfigurable Control. *Journal of Guidance Control and Dynamics*, 18(3):516–524, May–Jun. 1995.

[CS91] D.W. Clarke and R. Scattolini. Constrained receding-horizon predictive control. In *Proceedings of the Institution of Electrical Engineers*, volume 138, pages 347–354, 1991.

[CZ91] L. Chisci and G. Zappa. A systolic architecture for iterative LQ optimisation. *Automatica*, 27:799–810, 1991.

[DC93] H. Dermircioglu and D.W. Clarke. Generalised Predictive Control with endpoint weighting. In *IEE Proceedings Part D*, volume 140/4, pages 275–282. IEE, 1993.

[DCT87] C. Mohtadi D.W. Clarke and P.S. Tuffs. Generalized Predictive Control I and II. *Automatica*, 23(2):137–160, 1987.

[DG96] N. Dhayagunde and Z. Gao. Novel Approach to Reconfigurable Control Systems Design. *Journal of Guidance Control and Dynamics*, 19(4):963–966, 1996.

[Fle91] R. Fletcher. *Practical Optimization*. John Wiley and Sons, first edition, 1991.

[FMB90] M.P. Ford, J.M. Maciejowski, and J.M. Boyle. Multivariable Frequency Domain Toolbox. *The Mathworks Partner Series, GEC Engineering Research Centre, Cambridge Control Ltd.*, 1990.

[FR94] W.A. Fisher and H.E. Rauch. Augmentation of an Extended Kalman Filter with a Neural Network. In *Proceedings of IEEE International Conference on Neural Networks*. IEEE, June 1994.

[GA91] Z. Gao and P.J. Antsaklis. Stability of the pseudo-inverse method for reconfigurable control systems. *International Journal of Control*, 53(3):717–729, 1991.

[GA92] Z. Gao and P.J. Antsaklis. Reconfigurable control systems design via perfect model following. *International Journal of Control*, 56(4):783–798, 1992.

[GB95] J.E. Groszkiewicz and M. Bodson. Flight Control Reconfiguration Using Adaptive Methods. In *Proceedings of the 34th Conference on Decision and Control*, pages 1159–1164, New Orleans, LA, Dec. 1995. IEEE.

[GE91] P. Guerchet and J.L. Estival. Line of sight guidance law by predictive functional control for high-velocity short-range tactical missile. In *Proceedings First European Control Conference, Grenoble*, 1991.

[GL95] M. Green and D.J.N. Limebeer. *Linear Robust Control*. Prentice Hall, 1995.

[GL96] G.H. Golub and C.F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[GPM89] C.E. Garcia, D.M. Prett, and M. Morari. Model predictive control: Theory and practice – a survey. *Automatica*, 25:335–348, 1989.

[Hei94] S.A. Heise. *Multivariable Constrained Model Predictive Control*. PhD thesis, University of Cambridge, 1994.

[HM94] S.A. Heise and J.M. Maciejowski. *Stability of constrained MBPC using an internal model control structure*, pages 230–244. Oxford University Press, 1994.

[HM96a] M. Huzmezan and J.M. Maciejowski. Constrained Predictive Control Methods - A State Space Formulation. *Revue Roumaine de Science et Matematique Appliquee*, November 1996.

[HM96b] M. Huzmezan and J.M. Maciejowski. RCAM Design Challenge Presentation Document: The Model Based Predictive Control Approach. Technical Report FM(AG08) TP-088-20, Group for Aeronautical Research and Technology in Europe (GARTEUR), 1996.

[HM97a] M. Huzmezan and J.M. Maciejowski. A Development Space for Model Based Predictive Control. In *7th Symposium on Computer Aided Control System Design, Ghent,Belgium*. IFAC, 1997.

[HM97b] M. Huzmezan and J.M. Maciejowski. Notes on Filtering, Robust Tracking and Disturbance Rejection Used in Model Based Predictive Control Schemes. In *Proceedings of* 2^{nd} *Conference on System Structure and Control*, Bucharest, October 23-25 1997. IFAC.

[HM97c] M. Huzmezan and J.M. Maciejowski. *Robust Flight Control: A Design Challenge, GARTEUR*, volume 224 of *Lecture Notes in Control and Information Sciences*, chapter 12 – Predictive Control and 25 – Flight Management Using Predictive Control, pages 125–134 and 379–396. Springer-Verlag, 1997.

[HM98a] M. Huzmezan and J.M. Maciejowski. Reconfiguration and Scheduling in Flight Using Quasi-LPV High-Fidelity Models and $MBPC$ Control. In *Proceedings of American Control Conference, Philadelphia*, June 1998.

[HM98b] M. Huzmezan and J.M. Maciejowski. Automatic Tuning for Model Based Predictive Control During Reconfiguration. In *Proceedings of AERO'98, Seoul, Korea*. IFAC, 1998.

[HM98c] M. Huzmezan and J.M. Maciejowski. Reconfigurable Flight Control of a High Incidence Research Model Using Predictive Control. In *IEE-Control'98*, September 1998.

[HMW80] T.J. Harris, J.F. MacGregor, and J.D. Wright. Optimal sensor location with an application to a packed-bed tubular reactor. *AIChE*, 26:910–916, 1980.

[Hoa78] D.E. Hoak. USAF Stability and Control DATCOM. Technical Report AFWAL-TR-83-3048, Air Force Wright Aeronautical Lab., Wright-Patterson AFB, 1978.

[KBM96] M.V. Kothare, V. Balakrishnan, and M. Morari. Robust Constrained Model Predictive Control using Linear Matrix Inequalities. *Automatica*, pages 1361–1379, 1996.

[KP75] W.H. Kwon and A.E. Pearson. On the stabilization of a discrete constant linear system. *IEEE Transactions on Automatic Control*, 20:800–801, 1975.

[KP78] W.H. Kwon and A.E. Pearson. On feedback stabilisation of time varying discrete linear systems. *IEEE Transactions on Automatic Control*, 23:479–481, 1978.

[KP89] W.H. Kwon and A.E. Pearson. Receding horizon tracking control as a predictive control and its stability properties. *International Journal of Control*, 50:1807–1824, 1989.

[KR93] B. Kouvaritakis and J.A. Rossiter. Multivariable Stable Generalized Predictive Control. *IEE Procedings-D*, 140(5):364–372, 1993.

[KRC92] B. Kouvaritakis, J.A. Rossiter, and A.O.T. Chang. Stable Generalized Predictive Control: An Algorithm with Guaranteed Stability. *ProceedingsIEE – Part D*, 139(4):349–362, 1992.

[Lam83a] A.A. Lambreghts. Operational Aspects of the Integrated Vertical Flight Path and Speed Control System. In *Proceedings of the* 2^{nd} *Aerospace Behavioral Engineering Technology Coonference*, pages 53–65, Long-Beach, CA, 1983. Society of Automotive Engineers.

[Lam83b] A.A. Lambreghts. Vertical Flight Path and Speed Autopilot Design Using Total Energy Principles. *AIAA*, pages 2239–2250, August 1983.

[LBMP95] J.H. Lee, R.D. Braatz, M. Morari, and A. Packard. Screening Tools for Robust Control Structure Selection. *Automatica*, 31(2):229–235, 1995.

[Lee96] J.H. Lee. Recent advances in model predictive control and other related areas. In *Chemical Process Control*, Tahoe City, California, 1996. CACHE.

[LGM92] J.H. Lee, M.S. Gelormino, and M. Morari. Model Predictive Control of Multi-Rate Sampled-Data Systems. *International Journal of Control*, 55(1):153–191, 1992.

[LH74] C.L. Lawson and R.J. Hanson. *Solving least square problems.* Prentice Hall, 1974.

[LM91a] J.H. Lee and M. Morari. Robust measurement selection. *Automatica*, 27:519–527, 1991.

[LM91b] J.H Lee and M. Morari. Robust model predictive control: an identification-relevant algorithm. In *AIChE Anual Meeting*, Los Angeles, California, 1991.

[LP93] M. Labarrere and R.J. Patton. *Concise Encyclopedia of Aeronautics and Space Systems.* Pergamon Press, 1993.

[LY94] J.H. Lee and Z.H. Yu. Tuning of model predictive controllers for robust performance. *Computers in Chemical Engineering*, 18(1):15–37, 1994.

[Mac89] J.M. Maciejowski. *Multivariable Feedback Design.* Addison Wesley, first edition, 1989.

[Mac97] J.M. Maciejowski. Reconfigurable Control Using Constrained Optimisation. In *Proceedings of ECC'97*, Bruxelles, Belgium, 1997.

[Mac98] J.M. Maciejowski. Predictive Control. A lecture course given in the Aerospace Engineering Faculty TU Delft, October – December 1998.

[Mat95] The MathWorks. SIMULINK: A program for simulating dynamic systems. *The Mathworks Series*, 1995.

[MBT97] JF. Magni, S. Bennani, and J. Terlouw. *Robust Flight Control: A Design Challenge, GARTEUR*, volume 224 of *Lecture Notes in Control and Information Sciences*. Springer-Verlag, 1997.

[MGHM88] R.H. Middleton, G.C. Goodwin, D.J. Hill, and D.Q. Mayne. Design issues in adaptive control. *IEEE Transactions on Automatic Control*, 33:50–58, 1988.

[MH93] 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.

[MHBC89] D.D. Moerder, N. Halyo, J.R. Broussard, and A.K. Caglayan. Applications of pre-computed control laws in a reconfigurable aircraft flight control system. *Journal of Guidance Control and Dynamics*, 12(3):325–333, 1989.

[MO90] W.D. Morse and K.A. Ossman. Model following reconfigurable flight control system for the AFTI/F–16. *Journal of Guidance Control and Dynamics*, 13(6):969–976, 1990.

[Mor93] A.S. Morse. Supervisory control of families of linear set point controllers. In *Proceedings of the 32nd Conference on Decision and Control*, San Antonio, Texas, Dec 1993. IEEE.

[Mor94] 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, 1994.

[MR95] M. Morari and N.L. Ricker. Model Predictive Control Toolbox. *The Mathworks Partner Series*, 1995.

[MS91] P.S. Maybeck and R.D. Stevens. Reconfigurable Flight Control Via Multiple Adaptive Control Methods. *IEEE Transactions on Aerospace and Electronic Systems*, 27(3):470–479, 1991.

[MZ92] E. Mosca and J. Zhang. Stable redesign of predictive control. *Automatica*, 28(6):1229–1233, 1992.

[MZM84] E. Mosca, G. Zappa, and C. Manfredi. Multistep horizon self-tuning controllers: the MUSMAR approach . In *The* 9^th *World Congress*, Budapest, Hungary, 1984. IFAC.

[NB97] K.S. Narendra and J. Balakrishnan. Adaptive Control Using Multiple Models. *IEEE Transactions on Automatic Control*, 42(2):171–187, 1997.

[NBG97] G.De Nicolao, R.R. Bitmead, and M. Gevers. *To appear in: Digital and Control Systems, Techniques and Their Applications, Control and Dynamic Systems series (C.T. Leondes ed.* Academic Press, San Diego, 1997.

[Ost85] A.J. Ostroff. Techniques for accommodating control effector failures on a mildly statically unstable aircraft. In *Proceedings of the American Control Conference*, pages 903–906, 1985.

[Pat96] R.J. Patton. Where Are We in Fault-Tolerant Control? In E. Garcia and P. Albertos, editors, *European Science Foundation Workshop COSY*, October 1996.

[PCM95] M. Patcher, P.R. Chandler, and M. Mears. Reconfigurable Tracking Control with Saturation. *Journal of Guidance Control and Dynamics*, 18(5):1016–1022, Sep.–Oct. 1995.

[PCS97] M. Patcher, P.R. Chandler, and L. Smith. Maneuvering Flight Control. In *Proceedings of the American Control Conference*, July 1997.

[Pet84] V. Peterka. Predictor-based self-tuning control. *Automatica*, 20(1):39–50, 1984.

[PG88] D.M. Prett and C.E. Garcia. *Fundamental Process Control.* Butterworths Series in Chemical Engineering, first edition, 1988.

[PGC82] D.M. Prett, G.E. Garcia, and C.R. Cutler. The dynamic matrix control method. *US Patent 4349869*, 9 1982.

[PGS+97] G. Papageorgiou, K. Glover, A. Smerlas, I. Postlethwaite, and R.A. Hyde. *Robust Flight Control: A Design Challenge, GARTEUR*, volume 224 of *Lecture Notes in Control and Information Sciences*, chapter 7 – H_∞ Loop-shaping and 29 – The H_∞ Loop-shaping approach, pages 64–80 and 464–483. Springer-Verlag, 1997.

[PHGM97] G. Papageorgiou, M. Huzmezan, K. Glover, and J. Maciejowski. Combined MBPC/H_∞ Autopilot for a Civil Aircraft. *Proceedings of American Control Conference, New Mexico, USA*, 1997.

[Rat85] K.S. Rattan. Evaluation of control mixer concept for reconfiguration for flight control system. In *Proceedings on the 1985 IEEE National Aerospace and Electronics Conference*, volume 3, pages 560–569, 1985.

[Rau95] H.E. Rauch. Autonomous Control Reconfiguration. *IEEE Control Systems Magazine*, 15(6):37–48, 1995.

[RC91] B.D. Robinson and D.W. Clarke. Robustness effects of a prefilter in generalised predictive control. *IEE Procedings-D*, 138(1):2–8, 1991.

[RG96] J.A. Rossiter and B.G. Grinnell. A two-degree of freedom GPC algorithm with better tracking. In *Proceedings, 13^{t}h Triennial World Congress, San Francisco, USA*. IFAC, 1996.

[RK93] J.A. Rossiter and B. Kouvaritakis. Constrained Stable Generalized Predictive Control. *IEE Procedings-D*, 140(4):243–254, 1993.

[RKG95a] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner. Feasibility and Stability Results for Constrained Stable Generalised Predictive Control. *Automatica*, 31(6):863–877, 1995.

[RKG95b] J.A. Rossiter, B. Kouvaritakis, and J.R. Gossner. Mixed objective constrained stable generalised predictive control. *IEE Procedings-D*, 142(4):286–294, 1995.

[RM82] R. Rouhani and R.K. Mehra. Model Algorithmic Control (MAC): basic theoretical properties. *Automatica*, 18(4):401–414, 1982.

[RM93] J. Rawlings and K. Muske. The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 38:1512–1516, 1993.

[RRTP78] J. Richalet, A. Rault, J. Testud, and J. Papon. Model predictive heuristic control: application to industrial processes. *Automatica*, 14(5):413–428, 1978.

[SA90] J.S. Shamma and M. Athans. Analysisi of gain scheduled control for nolinear plants. *IEEE Transactions on Automatic Control*, 11(1):79–84, 1990.

[SC93] J.S. Shamma and J.R. Cloutier. Gain-Scheduled Missile Autopilot Design Using Linear Parameter Varying Transformations. *Journal of Guidance Control and Dynamics*, 16(2):256–263, Mar.–Apr. 1993.

[SCC95] P.O.M. Scokaert, C.M. Chow, and D.W. Clarke. Simulink Predictive Adaptive Control Environment - Users Guide. *Report No. OUEL 2057/95*, 1995.

[Sco97] P.O.M. Scokaert. Infinite horizon generalized predictive control. *International Journal of Control*, 66(1):161–175, 1997.

[SKB82] K.M. Soebel, H. Kaufmann, and M. Balas. Implicit adaptive control for a class of MIMO systems. *IEEE Transactions on Aerospace and Electronic Systems*, 18(5):576–589, 1982.

[Sma97] M.H. Smaili. Flight Data Reconstruction and Simulation of EL AL Flight 1862. Technical report, Delft University of Technology, Faculty of Aerospace Engineering, November 1997.

[Soe92] A.R.M. Soeterboek. *Predictive Control - A unified approach.* Prentice Hall, New York, first edition, 1992.

[SP96] S. Skogestad and J. Postelthwaite. *Multivariable Feedback Control Analysis and Design.* John Wiley and Sons, 1996.

[SPVvL91] A.R.M. Soeterboek, A.F. Pels, H.B. Verbruggen, and G.C.A. van Langen. A predictive controller for the Mach number in a transonic wind tunnel. *IEEE Control System Magazine*, 11(1):63–72, 1991.

[SSD95] S.N. Singh, M. Steinberg, and R.D. DiGirolamo. Nonlinear Predictive Control of Feedback Linearizable Systems and Flight Control System Design. *Journal of Guidance Control and Dynamics*, 18(5):1023–1028, 1995.

[VGN95] P. Vuthandam, H. Ghenceli, and M. Nikolaou. Performance bounds for robust quadratic Dynamic Matrix Control with end conditions. *AIChE Journal*, 1995.

[Vin93] G. Vinnicombe. *Measuring Robustness of Feedback Systems.* PhD thesis, Department of Engineering, University of Cambridge, 1993.

[WB95] D.G. Ward and R.L. Barron. A self-designing receding horizon optimal flight controller. In *Proceedings of American Control Conference*, 1995.

[Wes77] T.F. Westermeier. Redundancy management of digital FBW systems. In *Joint Automatic Control Conference*, pages 272–282, 1977.

[WG94] S.F. Wu and S.F. Guo. Optimum Flight Trajectory Guidance Based on Total Energy Control of Aircraft. *Journal of Guidance Control and Dynamics*, 17(2):291–296, March-April 1994.

[Wil76] A.S. Willsky. A survey of design methods for fault detection in dynamic systems. *Automatica*, 12:601–611, 1976.

[Wil97] S.W. Willcox. Private communication, DERA. Technical report, 1997.

[Woo95] G.D. Wood. *Control of Parameter–Dependent Mechanical Systems.* PhD thesis, University of Cambridge, 1995.

[YC93] T.W. Yoon and D.W. Clarke. Receding-horizon predictive control with exponential weighting. *International Journal of System Science*, 24(9):1745–1757, 1993.

[YC95] T.W. Yoon and D.W. Clarke. Observer design in receding-horizon predictive control. *International Journal of Control*, 61(1):171–191, 1995.

[Yed88] R.K. Yedavalli. Stability robustness measures under dependent uncertainty. In *Proceedings of the American Control Conference*, pages 820–823, 1988.

[Zaf90] E. Zafiriou. Robust Model Predictive Control of processes with Hard Constraints. *Computers Chemical Engineering*, 14(4/5):359–371, 1990.

[ZDG96] K. Zhou, J.C. Doyle, and K. Glover. *Robust and Optimal Control.* Prentice Hall, 1996.

[ZK87] K. Zhou and P.P. Khargonekar. Stability robustness bounds for linear state-space models with structured uncertainty. *IEEE Transactions on Automatic Control*, 32:621–623, 1987.