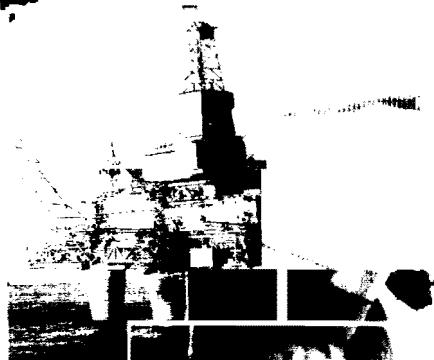


Predictive Control with Constraints



Prentice
Hall

J. M. Maciejowski

Predictive Control

with Constraints

J.M. Maciejowski



An imprint of **Pearson Education**

Harlow, England · London · New York · Reading, Massachusetts · San Francisco · Toronto · Don Mills, Ontario · Sydney
Tokyo · Singapore · Hong Kong · Seoul · Taipei · Cape Town · Madrid · Mexico City · Amsterdam · Munich · Paris · Milan

Contents

Preface	xi
List of Mini-Tutorials	xiv
List of Figures	xv
1 Introduction	1
1.1 Motivation	1
1.2 The ‘receding horizon’ idea	7
1.3 Computing the optimal inputs	9
1.4 A simple <i>MATLAB</i> program	14
1.5 Offset-free tracking	17
1.6 Unstable plant	22
1.7 Early history and terminology	25
1.8 Predictive control in the control hierarchy	26
1.9 General optimal control formulation	28
1.10 What is in this book	31
Exercises	32
2 A basic formulation of predictive control	36
2.1 State-space models	36
2.1.1 Form of the model	36
2.1.2 Linear model, nonlinear plant	37
2.1.3 First-principles models and system identification	40

2.2	A basic formulation	41
2.3	General features of constrained predictive control	47
2.4	Alternative state variable choices	49
2.5	Allowing for computational delay	50
2.6	Prediction	53
2.6.1	No disturbances, full state measurement	54
2.6.2	Constant output disturbance	56
2.6.3	Using an observer	57
2.6.4	Independent and realigned models	62
2.7	Example: Citation aircraft model	64
	Exercises	70
3	Solving predictive control problems	74
3.1	Unconstrained problems	74
3.1.1	Measured state, no disturbances	74
3.1.2	Formulation as a least-squares problem	77
3.1.3	Structure of the unconstrained controller	79
3.1.4	Estimated state	80
3.2	Constrained problems	81
3.2.1	Formulation as a QP problem	81
3.2.2	Controller structure	84
3.3	Solving QP problems	88
3.3.1	Active Set methods	89
3.3.2	Interior point methods	94
3.4	Softening the constraints	97
	Exercises	104
4	Step response and transfer function formulations	108
4.1	Step and pulse response models	108
4.1.1	Step and pulse responses	108
4.1.2	Relations to state-space models	111
4.1.3	Prediction using step response model	113
4.1.4	State-space models from step responses	115
4.2	Transfer function models	121
4.2.1	The basics	121
4.2.2	Prediction using transfer functions	124
4.2.3	Prediction with a disturbance model	127
4.2.4	The GPC model	133
4.2.5	State-space interpretations	134
4.2.6	Multivariable systems	142
	Exercises	143
5	Other formulations of predictive control	145
5.1	Measured disturbances and feedforward	145

5.2	Stabilized predictions	148
5.3	Decomposition of unstable model	150
5.4	Non-quadratic costs	151
5.4.1	Characteristics of LP and QP problems	151
5.4.2	Absolute value formulations	154
5.4.3	Min-max formulations	155
5.5	Zones, funnels and coincidence points	156
5.6	Predictive Functional Control (PFC)	159
5.7	Continuous-time predictive control	163
	Exercises	165
6	Stability	167
6.1	Terminal constraints ensure stability	169
6.2	Infinite horizons	172
6.2.1	Infinite horizons give stability	172
6.2.2	Constraints and infinite horizons — stable plant	176
6.2.3	Constraints and infinite horizons — unstable plant	178
6.3	Fake algebraic Riccati equations	180
6.4	Using the Youla parametrization	183
	Exercises	185
7	Tuning	188
7.1	What are we trying to do?	188
7.2	Some special cases	191
7.2.1	Effect of control weighting	191
7.2.2	Mean-level control	192
7.2.3	Deadbeat control	192
7.2.4	'Perfect' control	193
7.3	Frequency-response analysis	199
7.4	Disturbance models and observer dynamics	201
7.4.1	Disturbance models	201
7.4.2	Observer dynamics	203
7.5	Reference trajectory and pre-filter	211
	Exercises	214
8	Robust predictive control	217
8.1	Formulations of robust control	217
8.1.1	Norm-bounded uncertainty	218
8.1.2	Polytope uncertainty	220
8.2	The tuning procedure of Lee and Yu	221
8.2.1	Simplified disturbance and noise model	221
8.2.2	Tuning procedure	224
8.3	The LQG/LTR tuning procedure	225
8.4	LMI approach	233

8.4.1	Overview	233
8.4.2	Robustness without constraints	234
8.4.3	Robustness with constraints	237
8.5	Robust feasibility	239
8.5.1	Maximal output admissible sets	240
8.5.2	Robust admissible and invariant sets	241
	Exercises	246
9	Two case studies	248
9.1	Shell oil fractionator	248
9.1.1	Process description	248
9.1.2	Control specifications	251
9.1.3	Initial controller design	253
9.1.4	Controller performance and refinement	258
9.1.5	Constraint softening	261
9.1.6	Robustness to model errors	263
9.2	Newell and Lee evaporator	269
	Exercises	274
10	Perspectives	276
10.1	Spare degrees of freedom	276
10.1.1	Ideal resting values	276
10.1.2	Multiobjective formulations	277
10.1.3	Fault tolerance	278
10.2	Constraint management	281
10.3	Nonlinear internal models	284
10.3.1	Motivation and approaches	284
10.3.2	Sequential quadratic programming	285
10.3.3	Neural net models	286
10.3.4	Sub-optimal nonlinear MPC	288
10.4	Moving-horizon estimation	289
10.5	Concluding remarks	290
References		292
A	Some commercial MPC products	305
A.1	Aspentech: <i>DMCPlus</i>	306
A.2	Honeywell: <i>RMPCT</i>	308
A.3	Simulation Sciences: <i>Connoisseur</i>	309
A.4	Adersa: <i>PFC</i> and <i>HIECON</i>	311
A.5	ABB: <i>3dMPC</i>	311
A.6	Pavilion Technologies Inc.: <i>Process Perfecter</i>	311

B MATLAB program basicmpc	313
C The MPC toolbox	318
C.1 General remarks	318
C.2 Functions scmpc2 and scmpcnl2	321
C.3 Functions scmpc3 and scmpc4	322
Author index	323
Subject index	326

Preface

Predictive Control, or Model-Based Predictive Control ('MPC' or 'MBPC') as it is sometimes known, is the only advanced control technique — that is, more advanced than standard PID control — to have had a significant and widespread impact on industrial process control. The main reason for this is that it is

- ➊ The only generic control technology which can deal routinely with equipment and safety constraints.

Operation at or near such constraints is necessary for the most profitable or most efficient operation in many cases. The penetration of predictive control into industrial practice has also been helped by the facts that

- ➋ Its underlying idea is easy to understand,
- ➋ Its basic formulation extends to multivariable plants with almost no modification,
- ➋ It is more powerful than PID control, even for single loops without constraints, without being much more difficult to tune, even on 'difficult' loops such as those containing long time delays.

Predictive control was developed and used in industry for nearly 20 years before attracting much serious attention from the academic control community. This community tended to ignore its potential for dealing with constraints, thus missing its main advantage. In addition, it tended to point to the fact that, when constraints are ignored, predictive control is equivalent to conventional, though generally 'advanced', linear control, and hence apparently nothing new. This is true, but again misses the important point that issues such as tunability and understandability are crucial for the acceptance of a control technology, at least in the process control environment. Fortunately, the academic community has for some years now appreciated that predictive control really does offer something new for control in the presence of constraints, and has provided much analysis, and new ideas, to such an extent that it has gone beyond current industrial

practice, and is preparing the ground for much wider application of predictive control — potentially to almost all control engineering problems. The constant increase in computing speed and power certainly makes that a real prospect.

In this book I have attempted to bring everything together. I have tried to convey the simplicity of the underlying concepts of predictive control, but also to show how it relates to existing control theory, and indeed how much more can be done with it when its use is informed by a knowledge of standard control techniques, such as state estimation, disturbance modelling, and frequency response analysis. Predictive control involves optimization, and I have included details of how to solve at least some of the optimization problems encountered. I have also tried to display the main directions of current research, and give some indication of likely future directions, both of research and of applications.

This book assumes that the reader has some previous exposure to control theory, such as a first course and some contact with state-space models. It will be suitable for graduate students taking systems and control courses, but I hope that it will also prove useful to practising industrial engineers. In order to increase its usefulness to non-students, and also to alleviate the problem of prerequisites for students, I have included a number of *Mini-Tutorials*. These are one-page summaries of topics, such as observers or Lyapunov equations, that are needed in order to understand the material at certain points. I believe that the Mini-Tutorials contain enough explanation to allow the reader to follow the developments in the book, but of course they are not meant to replace in-depth study of these important topics.

It is essential to have access to suitable software in order to solve any non-trivial predictive control problems, and to gain experience of how it actually works. This book assumes that the reader has access to *MATLAB*, together with its *Control System Toolbox* and *Model Predictive Control Toolbox*. Some simple *MATLAB* files have also been written, and some files which augment those available in the *Model Predictive Control Toolbox* — some of these also require the *Optimization Toolbox*. All such additional software is available on this book's Companion Web Site:

<http://www.booksites.net/maciejowski/>

The versions of software used for the preparation of this book were:

<i>MATLAB</i>	5.3.1
<i>Control System Toolbox</i>	4.2.1
<i>Model Predictive Control Toolbox</i>	1.0.4
<i>Optimization Toolbox</i>	2.0

My students Eric Kerrigan and Simon Redhead were responsible for most of the modifications to *Model Predictive Control Toolbox* functions which are available on the web site.

This book originated in a course of lectures given at the Faculty of Aerospace Engineering in Delft, during November and December 1997. I would like to thank Bob Mulder, Hans van der Vaart and Samir Bennani for inviting me to spend a sabbatical term at Delft, for making all the arrangements, for making my stay at Delft both pleasant and interesting, and above all for having enough vision to believe that a course on predictive

control was not out of place in an aerospace department. I would also like to thank Ton van den Boom for giving me feedback on the first few chapters, Rob de Vries (both of TU Delft) for some valuable help with Chapter 4, and Hans van der Vaart for providing me with the linearized model of the Citation aircraft which is used in a number of illustrative examples in the book. The use of an aircraft example may seem quirky, since predictive control has been used almost exclusively in the process industries. There are two reasons for this. Firstly, it reflects my conviction that predictive control has great potential in all application sectors, as a result of continuing increases in real-time computational possibilities. Secondly, most process control examples require considerable explanation of the context to those who are not chemical engineers, whereas most readers will be able to understand the (very simplified) aircraft example from their own experience. Of course there are also examples based on process control in the book, and the major case studies in Chapter 9 are both taken from process control.

I gave a graduate course based on the material in this book to the Centre for Process Control Design at Lund University, and received several very valuable suggestions from both faculty and students who attended that course. Andrew Ogden-Swift of Honeywell Hi-Spec Solutions, Sean Goodhart of Aspentech, David Sandoz of SimSci-Predictive Control, and Jacques Richalet of Adersa have all been very generous of their time, have provided me with details of their companies' products, and have discussed the practicalities of predictive control most valuably and stimulatingly. David Sandoz and David Clarke both provided extremely useful, but also encouraging, feedback at various stages of the development of the manuscript, and Fred Loquasto of UC Santa Barbara read the almost-final version in record time, and picked up several errors and infelicities.

Any errors remain my own responsibility, of course.

J.M. Maciejowski
14 October 2000

A Companion Web Site accompanies *Predictive Control* by Jan Maciejowski.

Visit the *Predictive Control* Companion Web Site at
www.booksites.net/maciejowski

Here you will find valuable teaching and learning material including:

For Students and Lecturers:

- MATLAB files allowing you to model and simulate many of the systems presented in the book
- Links to software needed to run the simulations (commercial Matlab-based software from The Mathworks, and free extensions and other software from the author)
- Links to websites related to predictive control

For Lecturers:

- Downloadable solution's manual



List of Mini-Tutorials

1	Quadratic forms.	42
2	State observers.	58
3	Convex optimization, QP and LP problems.	84
4	The singular value decomposition.	117
5	Diophantine equations.	126
6	Stability and Lyapunov functions.	170
7	Linear quadratic optimal control.	181
8	Interpreting frequency responses.	200
9	The Kalman filter.	226
10	Linear matrix inequalities.	235

List of Figures

1.1	Semi-batch reactor.	3
1.2	Set-point improvement available from the use of predictive control.	4
1.3	Surge level control.	6
1.4	Predictive control: the basic idea.	8
1.5	Results from <code>basicmpc</code> , exact model.	17
1.6	Results from <code>basicmpc</code> with incorrect model gain.	18
1.7	Results from <code>trackmpc</code> with incorrect model gain.	19
1.8	Input and output disturbances and measurement noise.	21
1.9	Simulation of system with input and output disturbances and measurement noise using <code>noisympc</code> .	22
1.10	Control of unstable helicopter using <code>unstampc</code> .	24
1.11	Typical use of predictive control — current practice.	27
1.12	Use of predictive control — future trend?	28
2.1	Paper machine with headbox.	46
2.2	The boundary between controller and plant — a matter of convenience.	49
2.3	Assumed timing of measuring and applying signals.	51
2.4	Output disturbance.	57
2.6	Response of Citation aircraft to 40 m step change in altitude set-point.	65
2.7	Response of Citation aircraft to 400 m step change in altitude set-point.	66
2.8	Response of Citation aircraft to 400 m step change in altitude set-point, with altitude rate constraint.	67
2.9	Response to disturbance with altitude rate constraint.	68
2.10	Response to disturbance with altitude rate set-point.	69
2.11	Response of Citation aircraft to 400 m step change in altitude set-point, with altitude rate constraint and plant-model error.	70

3.1	Structure of controller with no constraints and full state measurement.	79
3.2	Structure of controller with no constraints and state observer.	81
3.3	Structure of controller with constraints and state observer.	85
3.4	A quadratic cost surface and a linear inequality constraint: constraint inactive.	86
3.5	A quadratic cost surface and a linear inequality constraint: constraint active.	87
3.6	Basic interior point method for Example 3.1.	96
3.7	Plant–model error and hard constraints result in infeasibility.	100
3.8	Soft output constraints restore feasibility. 1-norm penalty with $\rho = 10^4$.	101
3.9	Response to disturbance with softened altitude rate constraint, $\rho = 5 \times 10^5$.	102
3.10	Response to disturbance with softened altitude rate constraint, $\rho = 10^4$.	103
4.1	Step response of 3-input, 2-output distillation column. Solid curves: original responses. Dotted curves: responses of 8-state model.	120
4.2	The first 15 singular values of \mathcal{H}_{250} for the example.	121
4.3	Spectral density of crew disturbance, modelled by (4.109) with $\rho = 0.98$.	129
4.4	Generation of $\hat{v}(k k)$ — one interpretation.	130
4.5	GPC applied to unstable helicopter.	142
5.1	Feedforward from measured disturbances.	146
5.2	Decomposition of unstable plant as a feedback interconnection of two stable systems.	150
5.3	Decomposition of unstable plant as a stable model and ‘measured quasi-disturbance’.	151
5.4	Coprime factorization of an unstable system, and its approximation.	151
5.5	Cost contours and constraints for LP (left) and QP (right) problems. The black dot shows the optimal solution in each case.	152
5.6	Using a QP formulation to place the desired operating point just inside the feasible region. <i>Left:</i> x_1 deviations cheaper than x_2 deviations. <i>Right:</i> x_2 deviations cheaper than x_1 deviations.	154
5.7	A <i>zone</i> objective.	156
5.8	A <i>funnel</i> objective.	157
5.9	A <i>funnel</i> objective with a straight line boundary.	158
5.10	A reference trajectory with only a few <i>coincidence points</i> .	158
5.11	A future input built up from a polynomial basis.	159
6.1	Finite and infinite horizons (no disturbances, perfect model).	173
6.2	Finite horizon and deadbeat response.	174
6.3	Youla parametrization of all stable feedback systems (if the plant is stable).	184
7.1	Two degree of freedom feedback system.	190
7.2	Mean-level control: set-point change on r_1 .	195
7.3	Mean-level control: set-point change on r_3 .	196
7.4	Default (DMC) observer: response to step disturbance.	196
7.5	Deadbeat observer: response to step disturbance.	197
7.6	Deadbeat set-point response. Step demand on r_1 .	198

7.7	'Perfect' controller: set-point response. Step demand on r_1 .	198
7.8	'Perfect' controller: set-point response. Step demand on r_3 .	199
7.9	Singular values of $S(z)$ (solid line) and $T(z)$ (broken line) for mean-level control of the headbox.	201
7.10	A 'proportional + integral' (PI) controller.	203
7.11	Predictive controller, unconstrained case.	204
7.12	Predictive controller, simplified.	205
7.13	Predictive controller, simplified again.	205
7.14	Control of water temperature with perfect plant model, and model of air temperature disturbance. $W = 1$, $V = 10^{-6}$. The broken line shows the air temperature.	208
7.15	Control of water temperature with model of air temperature disturbance. Perfect (solid line) and imperfect (broken line) models. $W = 1$, $V = 1$.	209
7.16	Control of water temperature with model of air temperature disturbance, heater power constraints, and imperfect model. $W = 1$, $V = 1$.	210
7.17	Controller structure with reference trajectories: non-anticipated set-point.	212
7.18	Controller structure with reference trajectories: anticipated set-point.	214
8.1	Feedback combination of controller and plant, with additive uncertainty model.	219
8.2	The disturbance and noise model used by Lee and Yu [LY94].	222
8.3	The LQ state feedback regulator.	227
8.4	The Kalman filter as a feedback system.	228
8.5	The closed loop with LQG controller.	228
8.6	Disturbance and noise model used for LQG/LTR .	231
9.1	The 'Shell' heavy oil fractionator.	249
9.2	Open-loop step responses of the 'Shell' heavy oil fractionator.	252
9.3	Hankel singular values of the 'Shell' heavy oil fractionator, with $T_s = 4$ minutes.	256
9.4	Response with initial tuning parameters.	259
9.5	Response with increased weight on z_1 and z_2 tracking errors, and penalized control moves.	260
9.6	Response to measured disturbance.	260
9.7	Response to unmeasured disturbance.	261
9.8	Response to measured and unmeasured disturbances, and soft constraints.	262
9.9	Response to measured and unmeasured disturbances, and hard constraints on $\hat{z}(k+i k)$ for $i > 3$.	263
9.10	Largest singular values of sensitivity $S(z)$ (broken line) and complementary sensitivity $T(z)$ (solid line).	264
9.11	Largest singular value (solid line), and estimates of complex μ (broken line) and real μ (dotted line) of $K(z)S(z)W(z)$.	266
9.12	Response with plant-model mismatch, unmeasured disturbance, and soft constraints.	266
9.13	Response with plant-model mismatch, unmeasured disturbance, soft constraints, and increased control move penalty.	267

9.14	Response with correct model, unmeasured disturbance, and increased control move penalty.	268
9.15	Largest singular value (solid line), complex μ (broken line) and real μ (dotted line) of $K(z)S(z)W(z)$ with the reduced-gain controller.	268
9.16	Forced-circulation evaporator.	269
9.17	<i>Simulink</i> model of the evaporator.	271
9.18	Simulation with single linearized model, obtained at the initial equilibrium condition. (Broken lines show the set-points.)	272
9.19	Simulation with a re-linearized internal model, obtained 70 minutes into the simulation run. (Broken lines show the set-points.)	273
9.20	Simulation with a re-linearized internal model obtained every 10 minutes of simulated time. (Broken lines show the set-points.)	274
10.1	Execution of a step change in yaw angle with a jammed rudder.	279
10.2	LNG liquefaction plant.	280
10.3	LNG plant: response in normal operation.	280
10.4	LNG plant: valve H stuck, controller unaware.	281

Publisher's acknowledgments

We are grateful to the following for permission to reproduce copyright material:

Figure 2.1 from ‘Schematic of Paper Machine Headbox Control Problem’ taken from online User Guide for the MPC Toolbox, pp 3–27, © The Mathworks 1984–1998; Figure 9.1 from *Shell Process Control Workshop*, Butterworths (Morari and Prett, 1987) © Manfred Morari and David Prett.

Whilst every effort has been made to trace the owners of copyright material, in a few cases this has proved impossible and we take this opportunity to offer our apologies to any copyright holders whose rights we may have unwittingly infringed.

Introduction

1.1	Motivation	1
1.2	The ‘receding horizon’ idea	7
1.3	Computing the optimal inputs	9
1.4	A simple <i>MATLAB</i> program	14
1.5	Offset-free tracking	17
1.6	Unstable plant	22
1.7	Early history and terminology	25
1.8	Predictive control in the control hierarchy	26
1.9	General optimal control formulation	28
1.10	What is in this book	31
	Exercises	32

1.1 Motivation

The only advanced control methodology which has made a significant impact on industrial control engineering is *predictive control*. It has so far been applied mainly in the petrochemical industry, but is currently being increasingly applied in other sectors of the process industry. The main reasons for its success in these applications are:

1. It handles multivariable control problems naturally.
2. It can take account of actuator limitations.
3. It allows operation closer to constraints (compared with conventional control), which frequently leads to more profitable operation. Remarkably short pay-back periods have been reported.
4. Control update rates are relatively low in these applications, so that there is plenty of time for the necessary on-line computations.

In addition to the ‘constraint-aware optimizing’ variety of predictive control, there is an ‘easy-to-tune, intuitive’ variety, which puts less emphasis on constraints and optimization, but more emphasis on simplicity and speed of computation, and is particularly suitable for single-input, single-output (‘SISO’) problems. This variety has been applied in high-bandwidth applications such as servomechanisms, as well as to relatively slow processes.

Modern computing hardware is now so fast that it is no longer necessary to distinguish between these two varieties of predictive control. It has been estimated that during the last ten years the increase in the speed of hardware, together with improvements in optimization algorithms, could in principle allow the speed of solution of convex optimization problems — which are central to predictive control — to have increased by a factor of 10^6 [RTV97]. That is, problems which required 10 minutes to solve ten years ago should only require about 600 microseconds now. Of course that is an exaggeration; it considers only the speed of the processor, and ignores factors such as memory access times — a ‘fetch’ from memory can take more than 100 processor clock cycles — but even if it is over-optimistic by a factor of 10^2 , it is compelling evidence that predictive control need no longer be confined to ‘slow’ processes, even with the full paraphernalia of constraints and optimization. Furthermore, this estimate does not take account of efficiencies which are possible due to the structure of the predictive control problem, and which have only recently been discovered [RWR98].

For this reason, this book does not make a big distinction between the two varieties of predictive control identified above, but gives a unified treatment of both of them. In this chapter, the underlying ideas are introduced along the lines of the ‘easy-to-tune, intuitive’ variety — which is already capable of handling very non-trivial problems, as Example 1.8 shows. More details of this approach to predictive control are given later in Section 5.6. But most of the following chapters treat the predictive control problem in its full complexity, and hence in its most useful form: multivariable, and with constraints.

The importance of being able to take account of constraints arises for several reasons. The reason cited most often for the success of predictive control in the process industries is that the most profitable operation is often obtained when a process is running at a constraint, or even at more than one constraint. Often these constraints are associated with direct costs, frequently energy costs; if a product requires heating during its manufacture, for instance, the manufacturing cost can be minimized by keeping the heat supplied as small as possible; nevertheless the heat supplied (and its profile with time) has to be sufficient, and this is a constraint on the manufacturing process. If a product has to be of at least a certain quality in order to be useful, the cost can usually be minimized by making its quality just sufficient; again, this is a constraint which has to be respected. These are examples of constraints on outputs of controlled processes: the quality of the product is either a controlled variable itself, or depends on some combination of controlled variables.

Constraints on the control signals, that is inputs to the process, or manipulated variables, are also present. Most commonly these constraints are in the form of saturation characteristics: valves with a finite range of adjustment, flow rates with maximum values due to fixed pipe diameters, or control surfaces with limited deflection angles. Input constraints also appear in the form of rate constraints: valves and other actuators with

limited slew rates. These constraints, especially of the saturation type, are also often active when a process is running at its most profitable condition. They may limit the production rate, for example.

Example 1.1

Semi-batch reactor

Figure 1.1 shows a semi-batch reactor. A reactant enters the reactor on the left. An exothermic reaction occurs, whose temperature is controlled to a set-point by adjusting the flow of cooling water to the heat exchanger shown on the right. A successful reaction requires the temperature to be held close to the set-point (possibly within specified limits). The reactant flow rate and the cooling water flow rate can both be adjusted between minimum and maximum limits. The economic objective is to finish the batch as quickly as possible, subject to these requirements and constraints. In practice, that is equivalent to keeping the reactant flow rate as large as possible.

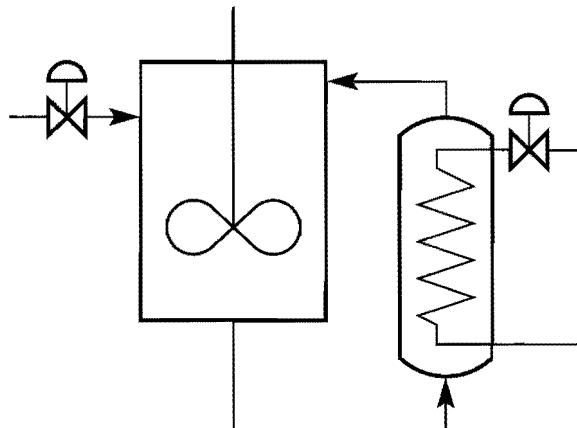


Figure 1.1 Semi-batch reactor.

There are many other control systems in which the performance is limited by the presence of constraints, even if that performance is not naturally expressed in monetary terms. Traditionally, control system performance requirements have not been formulated in a way which reflects this, partly because the available control design and implementation technologies have not been able to support such a formulation. But the feasibility of using predictive control in an increasing range of applications highlights not only the importance of constraints in such applications, but also the natural way in which quite complex control objectives can be stated, once constraints can be explicitly stated as part of the problem formulation.

- Compressors typically operate most efficiently at or close to the ‘surge line’, but can suffer catastrophic failure if the operating point crosses to the wrong side of the line;

that is clearly a hard constraint on any compressor control system.¹

- ➊ Automotive pollution legislation leads to running internal combustion engines at relatively weak fuel/air ratios. This makes the engines liable to stall, and has led to widespread introduction of engine control systems. Currently these systems are not formulated as predictive control systems, but a natural formulation of the control problem in this case is one which contains both the legislated pollution limits and the torque production requirements as constraints.
- ➋ Underwater vehicles, both military (submarines) and commercial (oil platform maintenance etc.), have a requirement to perform manoeuvres as quickly as possible, either to evade pursuers or to reduce operating costs. But they typically have constraints on allowable pitch and roll angles and on rate of change of depth, arising from the needs of internal equipment and of the crew. So again there are scenarios in which the best performance is obtained by operating for an extended period at one or more constraints.

Of course, one does not really want to operate a plant exactly at the real limits of its capabilities. One has to have something in reserve, to deal with unexpected disturbances from various sources. But the better the control system is at dealing with such disturbances, the closer can one operate to the constraints. A classical argument in favour of linear optimal control is that if the disturbances are random, and if one can reduce the variance of the controlled outputs as far as possible, then one can operate as near as possible to the constraints, and hence operate the plant as near as possible to its optimal performance. This is illustrated in Figure 1.2. This shows three hypothetical probability distributions of some controlled output of a plant, and a constraint beyond which the output should not stray. Distribution (a) shows the Gaussian shape and relatively large variance which results from the use of a relatively

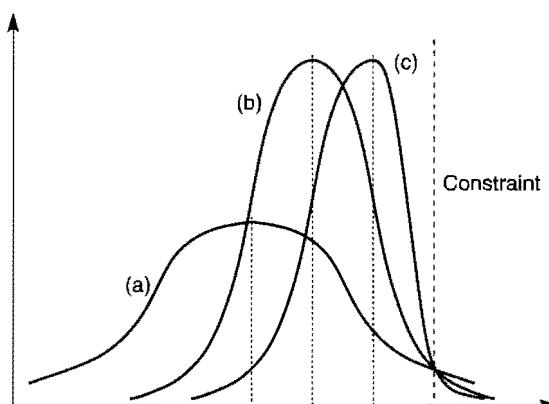


Figure 1.2 Set-point improvement available from the use of predictive control.

¹ The surge line constraint is a *nonlinear* inequality constraint involving the pressure ratio across the compressor and the flow rate through it. Most formulations of predictive control assume *linear* inequality constraints. But the nonlinear constraint can be approximated by one or more linear constraints.

badly tuned linear controller, assuming that the plant behaves approximately linearly, and that the disturbances have a Gaussian distribution. In order to have an acceptably low probability of violating the constraint, the set-point for the output variable has to be set relatively far away from the constraint, and hence the plant operates far away from the optimal point for the great majority of the time. Distribution (b) shows a distribution which might be achieved by the use of linear optimal control. The variance has been reduced, thus allowing the set-point to be significantly closer to the constraint. The distribution remains Gaussian, since the control law is linear.

Distribution (c) shows the effect of using predictive control. The controller is aware of the constraint, and hence reacts very differently in response to a disturbance which pushes the output towards the constraint than it would do in response to a disturbance which pushes it away from it. The controller is therefore nonlinear, and the distribution of the output variable is no longer Gaussian. It becomes significantly unsymmetric, with the result that it is possible to operate the plant with a set-point very close to the constraint, while retaining an acceptably small probability of violating the constraint.²

Another reason for the importance of being able to take account of constraints is the presence of buffer tanks in many processes. Such tanks are often used to store a product between individual units in production processes, and their principal purpose is to absorb the effects of disturbances in one unit, thus preventing the propagation of the effects to downstream units. The level of liquid in such a buffer tank has to be controlled, in order to ensure that it does not become completely empty or full, but between these limits it can be allowed to vary. In fact, the whole point of having such a tank is for the level to vary, since that is the means by which it absorbs the effects of disturbances. There is no point whatever in controlling the level in a buffer tank to a set-point, since that would destroy its purpose. Traditional process control deals with this problem by means of overrides and other *ad hoc* fixes. Predictive control, however, provides a very natural means of handling the control of buffer tanks, within its standard formulation: constraints on the levels of such tanks are defined, but no set-points are specified. (The control objectives for such outputs are sometimes called *zone* objectives — see Section 5.5.) Similar control objectives exist in other contexts, and it has even been suggested [ZAN73] that the majority of process control problems are naturally formulated in this way.

Example 1.2

Surge level control

Figure 1.3 shows a train of fractionators. The liquid heavy fraction in each column drains to the sump at the bottom, and is then passed on to the next fractionator. The flow rate is controlled to keep the level in the sump between minimum and maximum levels. The sump acts as a buffer, and absorbs any surges in flow which might arise from upstream columns. The sump levels are thus naturally controlled by defining

² The figure is supposed to illustrate the qualitative effects of using predictive control. Since the control action becomes nonlinear once constraints become active, it is usually extremely difficult to compute the actual probability distribution achieved.

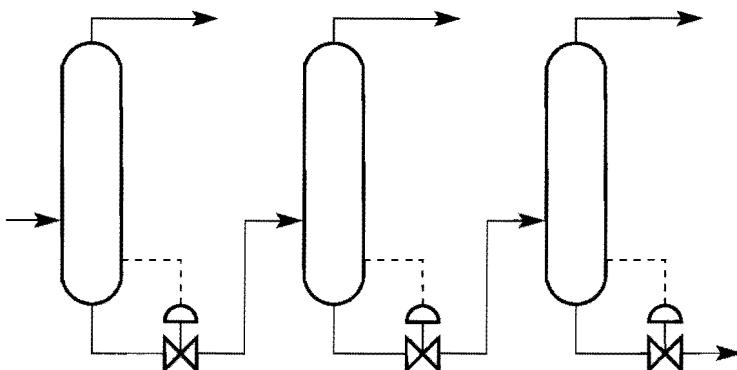


Figure 1.3 Surge level control.

them as zone objectives. The buffering action would be destroyed by tight control to set-points.

There is one more reason why predictive control's ability to handle constraints explicitly is important. The fact that the controller is aware of input constraints, in particular of actuator saturation constraints, and never generates input signals which attempt to violate them, removes the problem of *integrator wind-up*. This problem occurs with conventional controllers if long-duration set-point errors cause integrator outputs to exceed the saturation limits, and can result in large overshoots and even instability [FPEN94]. Standard remedies for the problem are known, but these have to be 'tacked on' to a conventional controller, increasing its complexity considerably. With predictive control, the wind-up problem does not arise.

Another factor which is crucial to the industrial success of predictive control is the natural way in which it can handle multivariable problems. Profit maximization, and more generally performance optimization, requires integrated control of complex plants and processes. This implies monitoring the future behaviour of many output variables, and using the full range of possible control variables to keep a process operating as economically as possible, and safely, over some future prediction horizon. Looking ahead to future applications of predictive control, one can cite as a suitable example the integrated control of advanced aircraft, in which engines, lateral thrusters, vortex separation control and aerodynamic surfaces are all used in a coordinated manner. All the ideas, and most of the algorithms, can be applied without modification to such multivariable problems, and are not significantly different if the number of available control inputs exceeds, is the same as, or is smaller than, the number of controlled outputs.

So far, reasons have been given for the practical success which predictive control has had to date. There is another reason which can be expected to become prominent in the future. Predictive control is *model based* in the sense that it uses an explicit internal model to generate predictions of future plant behaviour. The models currently used are usually 'black box' linear input-output models, obtained from simple plant tests or by applying system identification methods to plant data. But there is an increasing use of

nonlinear 'first-principles' models in the process industries. These are models which are obtained from an understanding of the physical and chemical transformations occurring inside a process. Some of them are extremely detailed and complex. As methodologies and technologies for supporting the development of such models improve, so the cost of developing them can be expected to fall from its current level, which is very high, and their use is therefore likely to become more widespread. There is much potential for future synergy of first-principles models with (model-based) predictive control. This may be realized by increasing use of nonlinear models in order to give more accurate predictions of process behaviour in nonlinear regimes (which leads to difficulties with predictive control — see Chapter 10), or by using nonlinear models to derive linear approximate models without the need for plant tests, or by using them in some other, as yet undeveloped, way.

1.2 The 'receding horizon' idea

Figure 1.4 shows the basic idea of predictive control. In this presentation of the basics, we confine ourselves to discussing the control of a single-input, single-output (SISO) plant. We assume a discrete-time setting, and that the current time is labelled as time step k . At the current time the plant output is $y(k)$, and the figure shows the previous history of the output trajectory. Also shown is a *set-point trajectory*, which is the trajectory that the output should follow, ideally. The value of the set-point trajectory at any time t is denoted by $s(t)$.

Distinct from the set-point trajectory is the *reference trajectory*. This starts at the current output $y(k)$, and defines an ideal trajectory along which the plant should return to the set-point trajectory, for instance after a disturbance occurs. The reference trajectory therefore defines an important aspect of the closed-loop behaviour of the controlled plant. It is not necessary to insist that the plant should be driven back to the set-point trajectory as fast as possible, although that choice remains open. It is frequently assumed that the reference trajectory approaches the set-point exponentially from the current output value, with the 'time constant' of the exponential, which we shall denote T_{ref} , defining the speed of response. That is, if the current error is

$$\epsilon(k) = s(k) - y(k) \quad (1.1)$$

then the reference trajectory is chosen such that the error i steps later, if the output followed it exactly, would be

$$\epsilon(k+i) = e^{-iT_s/T_{ref}} \epsilon(k) \quad (1.2)$$

$$= \lambda^i \epsilon(k) \quad (1.3)$$

where T_s is the sampling interval and $\lambda = e^{-T_s/T_{ref}}$. (Note that $0 < \lambda < 1$.) That is, the reference trajectory is defined to be

$$r(k+i|k) = s(k+i) - \epsilon(k+i) \quad (1.4)$$

$$= s(k+i) - e^{-iT_s/T_{ref}} \epsilon(k) \quad (1.5)$$

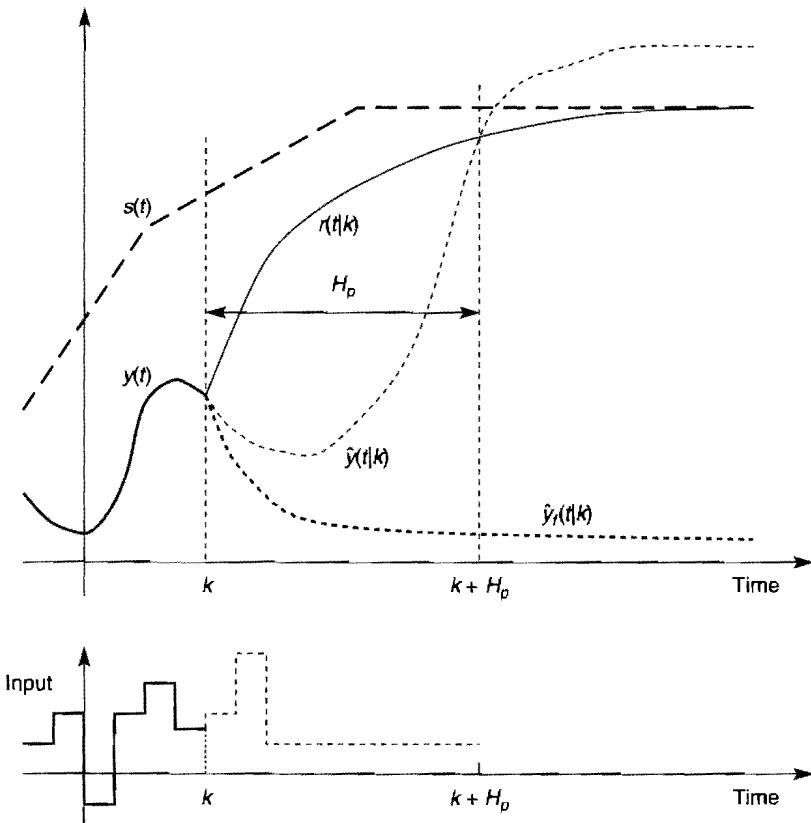


Figure 1.4 Predictive control: the basic idea.

The notation $r(k+i|k)$ indicates that the reference trajectory depends on the conditions at time k , in general. Alternative definitions of the reference trajectory are possible — for example, a straight line from the current output which meets the set-point trajectory after a specified time.

A predictive controller has an *internal model* which is used to predict the behaviour of the plant, starting at the current time, over a future *prediction horizon*. This predicted behaviour depends on the assumed input trajectory $\hat{u}(k+i|k)$ ($i = 0, 1, \dots, H_p - 1$) that is to be applied over the prediction horizon, and the idea is to select that input which promises the best predicted behaviour. We shall assume that the internal model is linear; this makes the calculation of the best input relatively straightforward. The notation \hat{u} rather than u here indicates that at time k we only have a prediction of what the input at time $k+i$ may be; the actual input at that time, $u(k+i)$, will probably be different from $\hat{u}(k+i|k)$. Note that we assume that we have the output measurement $y(k)$ available when deciding the value of the input $u(k)$. This implies that our internal model must be strictly proper, namely that according to the model $y(k)$ depends on past inputs $u(k-1), u(k-2), \dots$, but not on the input $u(k)$.

In the simplest case we can try to choose the input trajectory such as to bring the plant output at the end of the prediction horizon, namely at time $k + H_p$, to the required value $r(k + H_p)$. In this case we say, using the terminology of Richalet [Ric93b], that we have a single *coincidence point* at time $k + H_p$. There are several input trajectories $\{\hat{u}(k|k), \hat{u}(k+1|k), \dots, \hat{u}(k+H_p-1|k)\}$ which achieve this, and we could choose one of them, for example the one which requires the smallest input energy. But it is usually adequate, and in fact preferable, to impose some simple structure on the input trajectory, parametrized by a smaller number of variables. Figure 1.4 shows the input assumed to vary over the first three steps of the prediction horizon, but to remain constant thereafter: $\hat{u}(k+2|k) = \hat{u}(k+3|k) = \dots = \hat{u}(k+H_p-1|k)$, so that there are three ‘parameters’ to choose: $\hat{u}(k|k), \hat{u}(k+1|k), \hat{u}(k+2|k)$. The simplest possible structure is to assume that the input will remain constant over the prediction horizon: $\hat{u}(k|k) = \hat{u}(k+1|k) = \dots = \hat{u}(k+H_p-1|k)$. In this case there is only one parameter, namely $\hat{u}(k|k)$; since there is also only one equation to be satisfied — $\hat{y}(k+H_p|k) = r(k+H_p|k)$ — there is a unique solution.

Once a future input trajectory has been chosen, only the *first element* of that trajectory is applied as the input signal to the plant. That is, we set $u(k) = \hat{u}(k|k)$, where $u(k)$ denotes the actual signal applied. Then the whole cycle of output measurement, prediction, and input trajectory determination is repeated, one sampling interval later: a new output measurement $y(k+1)$ is obtained; a new reference trajectory $r(k+i|k+1)$ ($i = 2, 3, \dots$) is defined; predictions are made over the horizon $k+1+i$, with $i = 1, 2, \dots, H_p$; a new input trajectory $\hat{u}(k+1+i|k+1)$, with $i = 0, 1, \dots, H_p-1$ is chosen; and finally the next input is applied to the plant: $u(k+1) = \hat{u}(k+1|k+1)$. Since the prediction horizon remains of the same length as before, but slides along by one sampling interval at each step, this way of controlling a plant is often called a *receding horizon* strategy.³

13 Computing the optimal inputs

In the simplest case considered above, when there is only one coincidence point and only one parameter to choose for the future input trajectory, there is a unique solution, as has already been stated. More commonly, there are several coincidence points in the prediction horizon — perhaps even all the points $k+1, k+2, \dots, k+H_p$ are coincidence points. Even if there is more than one parameter to be chosen for the future input trajectory, in the usual situation there are more coincidence points than parameters. In this case there are more equations to be satisfied than the number of available variables, and it is in general impossible to solve them exactly. That is, it is impossible to choose the future input trajectory such that the predicted output coincides with the reference input at all the coincidence points. In this case, some kind of approximate solution is necessary. Most commonly, a *least-squares* solution is found, namely one such that the sum of the squares of the errors, $\sum_{i \in P} [r(k+i|k) - \hat{y}(k+i|k)]^2$, is minimized, where P denotes the set of indices i which correspond to coincidence points. As will be seen

³ The *receding horizon* concept corresponds to the usual behaviour of the Earth’s horizon: as one moves towards it, it recedes, remaining a constant distance away from one.

shortly, if the internal model is linear, the least-squares solution is easily found, and in fact yields a linear control law.

Again, let us consider the simplest case first, with one coincidence point $k + H_p$ and one parameter to choose, $\hat{u}(k|k)$. Conceptually, one can proceed as follows. The internal model can first be used to predict the *free response* $\hat{y}_f(k + H_p|k)$ of the plant, namely the response that would be obtained at the coincidence point if the future input trajectory remained at the latest value $u(k - 1)$. The details of how this is obtained will depend on the form of model available, because the ‘initial conditions’ depend on the form of the model. If a step or pulse response is available as the model, then all the available past inputs are needed. For a transfer function or difference equation model, n past inputs and outputs are needed, where n is the order of the transfer function. For a state-space model, the current state is needed, or an estimate of it. Now let $S(H_p)$ be the response of the model to a unit step input, H_p steps after the unit step is applied. The predicted output at time $k + H_p$ is

$$\hat{y}(k + H_p|k) = \hat{y}_f(k + H_p|k) + S(H_p)\Delta\hat{u}(k|k) \quad (1.6)$$

where

$$\Delta\hat{u}(k|k) = \hat{u}(k|k) - u(k - 1) \quad (1.7)$$

is the change from the current input $u(k - 1)$ to the predicted input $\hat{u}(k|k)$. We want to achieve

$$\hat{y}(k + H_p|k) = r(k + H_p|k) \quad (1.8)$$

so the optimal change of input is given by

$$\Delta\hat{u}(k|k) = \frac{r(k + H_p|k) - \hat{y}_f(k + H_p|k)}{S(H_p)} \quad (1.9)$$

Note that we have assumed that the model output and the plant output are the same up to time k . This will not be so in practice, but we shall address that in the next section.

Example 1.3

Suppose that the set-point is constant at value $s(k + i) = 3$, and that $T_{ref} = 9$ sec, the sampling interval is $T_s = 3$ sec, there is a single coincidence point at $H_p = 2$ (steps; that is, 6 sec into the future), and the plant’s z-transform transfer function is

$$G(z) = \frac{2}{z - 0.7} \quad (1.10)$$

The previous and current outputs are $y(k - 1) = y(k) = 2$, and the latest control input is $u(k - 1) = 0.3$. What is the optimal input $\hat{u}(k|k)$, assuming that $\hat{u}(k|k) = \hat{u}(k + 1|k)$?

We have $\epsilon(k) = s(k) - y(k) = 3 - 2 = 1$, and $\lambda = \exp(-T_s/T_{ref}) = 0.7165$. Hence

$$r(k + 2|k) = s(k + 2) - \lambda^2\epsilon(k) = 3 - 0.7165^2 \times 1 = 2.487$$

To get the free response, we put the transfer function into difference equation form:

$$y(k) = 0.7y(k-1) + 2u(k-1) \quad (1.11)$$

then assume that $u(k+1) = u(k) = u(k-1) = 0.3$ to get

$$\hat{y}_f(k+1|k) = 0.7 \times 2 + 2 \times 0.3 = 2.0 \quad (1.12)$$

$$\hat{y}_f(k+2|k) = 0.7 \times 2.0 + 2 \times 0.3 = 2.0 \quad (1.13)$$

Finally we need $S(2)$, the step response $H_p = 2$ time steps after a unit step input is applied. We can get this from (1.11) by assuming $u(k) = u(k+1) = 1$ and $y(k) = y(k-1) = 0$:

$$S(1) = 0.7 \times 0 + 2 \times 1 = 2 \quad (1.14)$$

$$S(2) = 0.7 \times 2 + 2 \times 1 = 3.4 \quad (1.15)$$

We now have everything required to compute the optimal input, using (1.9):

$$\Delta\hat{u}(k|k) = \frac{2.487 - 2.0}{3.4} = 0.1432 \quad (1.16)$$

$$\hat{u}(k|k) = u(k-1) + \Delta\hat{u}(k|k) = 0.4432 \quad (1.17)$$

This is the input signal applied to the plant: $u(k) = \hat{u}(k|k) = 0.4432$. If our model of the plant is perfect, and there are no disturbances, then this would result in the next plant output value being $y(k+1) = 0.7 \times 2 + 2 \times 0.4432 = 2.2864$.

If there is more than one coincidence point, then there are more equations to be satisfied than there are variables, and we therefore have to be satisfied with an approximate solution. Suppose we have c coincidence points, with corresponding values of the reference trajectory $r(k+P_1|k), r(k+P_2|k), \dots, r(k+P_c|k)$, with $P_c \leq H_p$. Since we would like to achieve $\hat{y}(k+P_i|k) = r(k+P_i|k)$ for $i = 1, 2, \dots, c$, we would like to choose $\hat{u}(k|k)$ to solve the following equations:

$$\hat{r}(k+P_1|k) = \hat{y}_f(k+P_1|k) + S(P_1)\Delta\hat{u}(k|k) \quad (1.18)$$

$$\hat{r}(k+P_2|k) = \hat{y}_f(k+P_2|k) + S(P_2)\Delta\hat{u}(k|k) \quad (1.19)$$

⋮

$$\hat{r}(k+P_c|k) = \hat{y}_f(k+P_c|k) + S(P_c)\Delta\hat{u}(k|k) \quad (1.20)$$

Most commonly, this equation is solved in the ‘least-squares’ sense. Some details of least-squares solutions are given in Chapter 3, but for the moment we just note that if we define the vectors

$$\mathcal{T} = \begin{bmatrix} \hat{r}(k+P_1|k) \\ \hat{r}(k+P_2|k) \\ \vdots \\ \hat{r}(k+P_c|k) \end{bmatrix} \quad \mathcal{Y}_f = \begin{bmatrix} \hat{y}_f(k+P_1|k) \\ \hat{y}_f(k+P_2|k) \\ \vdots \\ \hat{y}_f(k+P_c|k) \end{bmatrix} \quad \mathcal{S} = \begin{bmatrix} S(P_1) \\ S(P_2) \\ \vdots \\ S(P_c) \end{bmatrix} \quad (1.21)$$

then in *MATLAB* the least-squares solution is obtained using the ‘backslash’ operator as:

$$\Delta\hat{u}(k|k) = \mathcal{S} \setminus (\mathcal{T} - \mathcal{Y}_f) \quad (1.22)$$

Example 1.4

Consider the same set-up as in Example 1.3, but this time with two coincidence points: $P_1 = 1$, $P_2 = H_p = 2$. We have

$$r(k+1|k) = s(k+1) - \lambda\epsilon(k) = 2.284$$

$$r(k+2|k) = 2.487 \quad \text{as before}$$

So, using results already found in Example 1.3, we have

$$T = \begin{bmatrix} 2.284 \\ 2.487 \end{bmatrix} \quad \mathcal{Y}_f = \begin{bmatrix} 2.0 \\ 2.0 \end{bmatrix} \quad S = \begin{bmatrix} 2.0 \\ 3.4 \end{bmatrix}$$

and hence

$$\Delta\hat{u}(k|k) = S \setminus (T - \mathcal{Y}_f) = 0.1429$$

$$\hat{u}(k|k) = u(k-1) + \Delta\hat{u}(k|k) = 0.4429$$

We can already see at this point how predictive control can easily be given the capability of respecting constraints. If there are constraints on the inputs and/or outputs, then the simple ‘linear least-squares’ solution has to be replaced by a ‘constrained least-squares’ solution. To be sure, it is no longer possible to obtain a closed-form solution and some form of iterative optimization algorithm must be employed, but if the constraints are in the form of linear inequalities then we have a *quadratic programming* problem, which can be solved very reliably and relatively quickly. We will look in detail at solving the constrained predictive control problem in Chapters 3 and 5.

Now suppose that we allow a more complicated future input trajectory. We could assume that the input is allowed to change over the next H_u steps, for example, so that we have to choose $\hat{u}(k|k)$, $\hat{u}(k+1|k)$, ..., $\hat{u}(k+H_u-1|k)$, and assume that $\hat{u}(k+H_u-1|k) = \hat{u}(k+H_u|k) = \dots = \hat{u}(k+H_p-1|k) =$ if $H_u < H_p$. (It would be unusual to choose $H_u > H_p$, but this could be done. Usually H_u is chosen to be considerably smaller than H_p .) Now the predicted output at time $k + P_i$ is given by

$$\begin{aligned} \hat{y}(k+P_i|k) &= \hat{y}_f(k+P_i|k) + H(P_i)\hat{u}(k|k) + H(P_i-1)\hat{u}(k+1|k) + \dots \\ &\quad + H(P_i-H_u+2)\hat{u}(k+H_u-2|k) + \\ &\quad + S(P_i-H_u+1)\hat{u}(k+H_u-1|k) \end{aligned} \quad (1.23)$$

where $H(j) = S(j) - S(j-1)$ is the unit pulse response coefficient of the system after j time steps. The reason why pulse response coefficients appear in this expression, rather than step response coefficients, is that each of the input values $\hat{u}(k|k)$, $\hat{u}(k+1|k)$, ..., $\hat{u}(k+H_u-2|k)$ is assumed to be applied for only one sampling interval. Only the last one, $\hat{u}(k+H_u-1|k)$, remains unchanged until step P_i , and its effect is therefore obtained by multiplying it by the step response coefficient $S(P_i-H_u+1)$. Since we assume that the system is strictly proper, we have $H(0) = 0$ and $S(0) = 0$, and of course $H(j) = 0$ and $S(j) = 0$ if $j < 0$ because of causality. So if $P_i \leq H_u$ then the last non-zero term in this expression is $H(1)\hat{u}(k+P_i-1|k)$.

Since $H(j) = S(j) - S(j-1)$, we can rewrite (1.23) as

$$\begin{aligned} \hat{y}(k+P_i|k) &= \hat{y}_f(k+P_i|k) + S(P_i)\Delta\hat{u}(k|k) + S(P_i-1)\Delta\hat{u}(k+1|k) \\ &\quad + \dots + S(P_i-H_u+1)\Delta\hat{u}(k+H_u-1|k) \end{aligned} \quad (1.24)$$

by regrouping terms. Now we can write a set of equations for the predicted outputs at all the coincidence points in matrix-vector form:

$$\mathcal{Y} = \mathcal{Y}_f + \Theta \Delta \mathcal{U} \quad (1.25)$$

where

$$\mathcal{Y} = \begin{bmatrix} \hat{y}(k + P_1|k) \\ \hat{y}(k + P_2|k) \\ \vdots \\ \hat{y}(k + P_c|k) \end{bmatrix} \quad \Delta \mathcal{U} = \begin{bmatrix} \Delta \hat{u}(k|k) \\ \Delta \hat{u}(k + 1|k) \\ \vdots \\ \Delta \hat{u}(k + H_u - 1|k) \end{bmatrix} \quad (1.26)$$

and

$$\Theta = \begin{bmatrix} S(P_1) & S(P_1 - 1) & \cdots & S(1) & 0 & \cdots & 0 & 0 & \cdots & 0 \\ S(P_2) & S(P_2 - 1) & \cdots & \cdots & \cdots & \cdots & S(1) & 0 & \cdots & 0 \\ \vdots & \vdots \\ S(P_c) & S(P_c - 1) & \cdots & S(P_c - H_u + 1) \end{bmatrix} \quad (1.27)$$

Again, since we want to achieve $\mathcal{Y} = \mathcal{T}$, but usually don't have enough variables to do so exactly, we obtain the least-squares solution:

$$\Delta \mathcal{U} = \Theta \backslash [\mathcal{T} - \mathcal{Y}_f] \quad (1.28)$$

As was said before, we now select the first element of the vector $\Delta \mathcal{U}$, namely $\Delta \hat{u}(k|k)$, use it to form the input applied to the plant:

$$u(k) = \Delta \hat{u}(k|k) + u(k - 1) \quad (1.29)$$

and repeat the whole cycle of calculations at the next time step, starting with the next measurement of the plant output, $y(k + 1)$.

Example 1.5

We return to Example 1.4. This time, in addition to having the two coincidence points $P_1 = 1$ and $P_2 = 2$, we will choose $H_u = 2$, so that we compute optimal values of $\hat{u}(k|k)$ and $\hat{u}(k + 1|k)$ at each step. \mathcal{T} , \mathcal{Y}_f and S remain the same as before. The new entity now is

$$\Theta = \begin{bmatrix} S(1) & 0 \\ S(2) & S(1) \end{bmatrix} = \begin{bmatrix} 2.0 & 0 \\ 3.4 & 2.0 \end{bmatrix}$$

From Example 1.3 we know that $u(k - 1) = 0.3$. Note that in this case the number of variables to be chosen is the same as the number of coincidence points, so that the matrix Θ is square and can be inverted, and the solution is unique. MATLAB's 'backslash' operator still gives the correct solution in this case:

$$\Delta \mathcal{U} = \Theta \backslash [\mathcal{T} - \mathcal{Y}_f]$$

$$= \begin{bmatrix} 0.1420 \\ 0.0021 \end{bmatrix}$$

so that the input to be applied to the plant is

$$u(k) = \Delta\hat{u}(k|k) + u(k-1) = 0.1420 + 0.3 = 0.4420$$

It should be apparent that it is not necessary to have a discrete-time model in order to use predictive control. All that is needed is to know the step response at the coincidence points, and to be able to compute the ‘free response’ at the coincidence points — that is, one must have a simulation model capable of running faster than real time. In practice, however, a discrete-time linear model is easy to obtain, and more convenient than a continuous-time linear model.

1.4 A simple MATLAB program

Clearly, some software is needed not only to implement predictive control, but even to study it. Even for the simple scenarios studied in Section 1.3, with just one or two coincidence points and one or two variables, one needs to repeat the calculations many times in order to see how the resulting control law performs. A simple *MATLAB* program, called `basicmpc`, has been written to simulate basic predictive control, as described in this chapter. A complete listing of it is given in Appendix B, and it is also available on this book’s web site:

<http://www.booksites.net/maciejowski/>

This shows one way of organizing the computations. The reader should be able to use it as a ‘template’ which can be modified to handle other formulations of predictive control problems. For more complicated scenarios, however, such as multivariable problems and/or those involving constraints, and for simulation of nonlinear plants, we will later rely on the use of *MATLAB*’s *Model Predictive Control Toolbox*, which is described in Appendix C.

In this section we shall explain the main features of `basicmpc`, without going through all the details.

`basicmpc` assumes that an exponential reference trajectory is used, and the program starts by defining T_{ref} . The variable `Tref` is used, and as far as possible variables are used which correspond to the notation used in this chapter. Note that the program is in the form of a *MATLAB script file*, not a *function*, so that values of variables such as `Tref` are changed by editing the file, not by supplying arguments. The reader can easily modify the program so that it becomes a function, if desired.⁴ The possibility $T_{ref} = 0$ is allowed for, which represents immediate return of the reference trajectory to the set-point trajectory: $r(k+i|k) = s(k+i)$. Next, the sampling interval `Ts` is defined; by default this is set to $T_{ref}/10$ (or 1 if $T_{ref} = 0$), but this can be changed as required.

Next the plant is defined, as variable `plant`, which is a single-input, single-output discrete-time system in the form of an object of class `lti`, the class used to represent Linear Time-Invariant systems in the *Control System Toolbox*. In the rest of the program, the plant is represented by the numerator and denominator polynomials of its z -transform transfer function representation, `nump` and `demp`, respectively. Consequently,

⁴ Consult the *MATLAB* documentation for details.

however plant has been defined, it is then coerced into transfer function form, and these polynomials are extracted:

```
% Define plant as SISO discrete-time 'lti' object 'plant' in
% transfer function form:
%%%% CHANGE FROM HERE TO DEFINE NEW PLANT %%%
nump=1;
denp=[1,-1.4,0.45];
plant = tf(nump,denp,Ts);
%%%% CHANGE UP TO HERE TO DEFINE NEW PLANT %%%
plant = tf(plant); % Coerce to transfer function form
nump = get(plant,'num'); nump = nump{:};% Get numerator polynomial
denp = get(plant,'den'); denp = denp{:};% Get denominator
% polynomial
```

This allows the plant to be defined in many ways. For example, from a continuous-time state-space description:

```
%%%% CHANGE FROM HERE TO DEFINE NEW PLANT %%%
plant = ss(A,B,C,D); % Continuous-time state-space
plant = c2d(plant,Ts); % Discrete-time equivalent
%%%% CHANGE UP TO HERE TO DEFINE NEW PLANT %%%
```

Next, the model is defined in the same way, as variable `model`. This allows `model` to be different from `plant`, so that the effects of modelling errors can be studied, but as supplied the program sets `model = plant`. Again the numerator and denominator polynomials are extracted, this time with names `numm` and `denm`.

The coincidence points are defined in the vector `P` in terms of the number of sampling intervals, so that $P_1 = 3$, $P_2 = 7$, $P_3 = 12$ would be represented by `P=[3;7;12]`, etc. Later this vector is multiplied by `Ts` to get the corresponding vector of coincidence times, which is needed to compute the values of $\exp(-P_i T_s / T_{ref})$ (in vector `errfac`).

The variable `M` is used to store the control horizon H_u , the change of notation here being made for compatibility with the *Model Predictive Control Toolbox*, which also uses `M` for this purpose.

The set-point trajectory is defined in the vector `setpoint`. Note that its length has to be `nsteps + max(P)` where `nsteps` is the number of steps to be simulated, because the controller needs to have a prediction of the set-point, even when it reaches the end of the simulation.

Next, the *Control System Toolbox* function `step` is used to compute the step response of the model over the required prediction horizon (which is `max(P)`), and form the matrix Θ (variable `theta`) and the vector S , as in (1.27) and (1.21):

```
stepresp = step(model,[0:Ts:max(P)*Ts]);
theta = zeros(length(P),M);
for j=1:length(P),
    theta(j,:) = [stepresp(P(j):-1:max(P(j)-M+1,1))',zeros(1,M-P(j))];
end
S = stepresp(P);
```

Everything described so far needs to be done only once, before the simulation is started. The remaining calculations need to be repeated for every simulation step. This is done in a big ‘for-loop’ (`for k=1:nsteps`). First the reference trajectory has to be computed over the prediction horizon, starting at the current plant output $y_p(k)$; the reference trajectory is stored in the vector `reftraj`.

Next the free response of the model over the prediction horizon is obtained and stored in `ympfree`, by iterating the difference equation which corresponds to the transfer function of the model. At each iteration the previous values of the free response are held in the vector `yfpast`, which is initialized to the previous values of the model output, from previous steps (that is, from lower values of k):

```
% Free response of model over prediction horizon:
yfpast = ympast;
ufpast = umpast;
for kk=1:max(P), % Prediction horizon
    ympfree(kk) = numm*ufpast-denm(2:ndenm+1)*yfpast;
    yfpast=[ymfree(kk);yfpast(1:length(yfpast)-1)];
    ufpast=[ufpast(1);ufpast(1:length(ufpast)-1)];
end
```

Note that `ufpast` is needed to hold past values of the model input, even though the new values applied over the prediction horizon are all the same.

Now everything is ready for computing the optimal value of $\Delta\mathcal{U}$ (variable `dutraj`) and forming the new input to the plant $u(k)$ (variable `uu(k)`):

```
dutraj = theta\ (reftraj-ymfree(P)');
uu(k) = dutraj(1) + uu(k-1);
```

Finally the plant and model are simulated for one step to obtain their next outputs. It is important to note here that, if there is any difference between the plant and the model, then their outputs evolve independently:

```
% Simulate plant:
% Update past plant inputs:
uppast = [uu(k);uppast(1:length(uppast)-1)];
yp(k+1) = -denp(2:ndenp+1)*yppast+nump*uppast; % Simulation
% Update past plant outputs:
yppast = [yp(k+1);yppast(1:length(yppast)-1)];

% Simulate model:
% Update past model inputs:
umpast = [uu(k);umpast(1:length(umpast)-1)];
ym(k+1) = -denm(2:ndenm+1)*ympast+numm*umpast; % Simulation
% Update past model outputs:
ympast = [ym(k+1);ympast(1:length(ympast)-1)];
```

Richalet [Ric93b] has drawn attention to the importance of this *independent model* implementation, because it makes dealing with plant–model errors relatively simple, as

we shall see in the next section. We shall also see, however, that it is not always possible to respect this desideratum.

If the program is modified to simulate the effects of disturbances and/or noise, care should be taken to modify only the simulation of the plant, not of the model. For details, examine the program `noisympc`, which is available on the web site — also see the end of the next section.

When the simulation has been completed, a summary of the parameters is written out to the screen, and the results are presented graphically.

Example 1.6

The program `basicmpc` is supplied with the plant defined by the transfer function

$$\frac{1}{z^2 - 1.4z + 0.45}$$

with $T_{ref} = 6$, $T_s = 0.6$, $H_p = P_1 = 8$, $M = 1$, and with `model = plant`. Figure 1.5 shows the simulation results when the set-point is constant at 1, all initial conditions are zero, and there are no disturbances or noise.

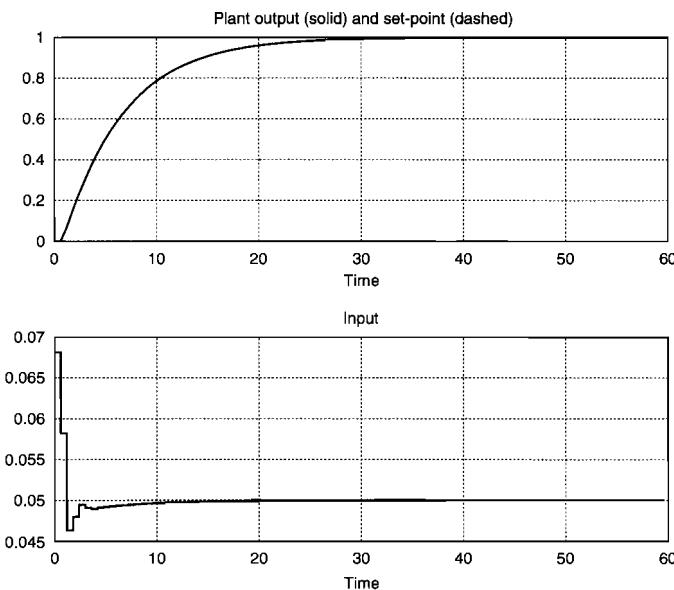


Figure 1.5 Results from `basicmpc`, exact model.

1.5 Offset-free tracking

If the model is not the same as the plant, in particular if the steady-state gain of the model is not correct, then the plant output will reach an incorrect final value. This is



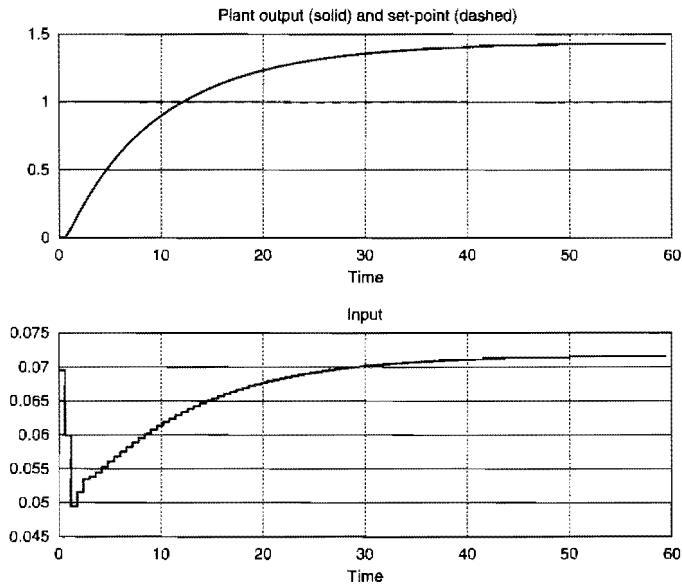


Figure 1.6 Results from `basicmpc` with incorrect model gain.

illustrated in Figure 1.6, which shows the results of running `basicmpc` with the plant having transfer function

$$\frac{1}{z^2 - 1.4z + 0.45}$$

while the model transfer function is

$$\frac{1}{z^2 - 1.4z + 0.46}$$

Here, in addition to changed pole locations, the steady-state gain of the plant is $1/(1 - 1.4 + 0.45) = 20$, while that of the model is $1/(1 - 1.4 + 0.46) = 16.67$. As a result, both the input and the output reach incorrect steady-state values. This is of course an extremely common situation, since the real plant gain is never known perfectly accurately. In fact, since the real plant is invariably nonlinear, its exact steady-state gain is likely to be different at each steady-state condition.

Fortunately, it is very easy to modify the predictive controller slightly, to make it insensitive to errors in the steady-state gain. All that is necessary is to measure the discrepancy between the latest plant output and the latest model output, and then to subtract that discrepancy from the reference trajectory at the coincidence points. That is, if we define

$$d(k) = y(k) - \hat{y}(k|k-1) \quad (1.30)$$

then we replace (1.28) by:

$$\Delta \mathcal{U} = \Theta \backslash [\mathcal{T} - \mathbf{1} d(k) - \mathcal{Y}_f] \quad (1.31)$$

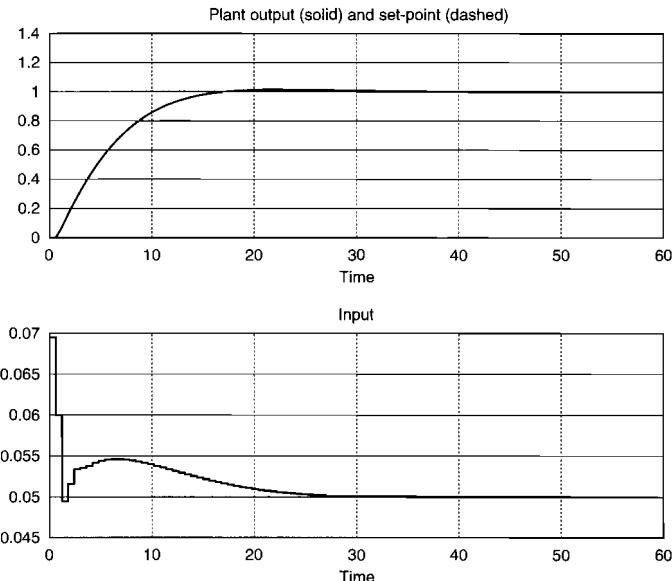


Figure 1.7 Results from `trackmpc` with incorrect model gain.

where $\mathbf{1}$ denotes a vector $[1, 1, \dots, 1]^T$. Note that the step response coefficients in Θ are those of the *model*, not the plant, since the true plant step response is not known to the controller. The corresponding change in the software is to insert the line

$$d = y_p(k) - y_m(k);$$

and to replace

$$\text{dutraj} = \theta \backslash (\text{reftraj} - \text{ymfree}(P)');$$

by

$$\text{dutraj} = \theta \backslash (\text{reftraj} - d - \text{ymfree}(P)');$$

The program `trackmpc` incorporates this modification, and is available on this book's web site. Simulation results, when using this modification with the same plant–model mismatch as in Figure 1.6, are shown in Figure 1.7. This figure shows that the plant output now reaches the set-point correctly.

This works — for asymptotically stable plant and model — for the following reason. Let the steady-state gain of the plant be $S_p(\infty)$, and the steady-state gain of the model be $S_m(\infty)$. Also assume that the closed-loop is asymptotically stable, and that the set-point is a constant $s(k) = s_\infty$. Then the input, and both the plant and model outputs, settle to constant values, say $u(k) \rightarrow u_\infty$, $y_p(k) \rightarrow y_{p\infty}$, and $y_m(k) \rightarrow y_{m\infty}$, where y_p and y_m denote the plant and model outputs, respectively. Then $y_{p\infty} = S_p(\infty)u_\infty$ and $y_{m\infty} = S_m(\infty)u_\infty$, so that

$$d(k) \rightarrow d_\infty = [S_p(\infty) - S_m(\infty)]u_\infty \quad (1.32)$$

In steady-state conditions, since both the set-point trajectory and the plant output are constant, the reference trajectory does not depend on k : $r(k+i|k) = r_i$. Hence \mathcal{T} is a constant vector. Also \mathcal{Y}_f is a constant vector, since the free response is always computed from the same initial conditions, once steady-state has been reached. Now recall from (1.25) that we have

$$\Theta \Delta \mathcal{U} = \mathcal{Y} - \mathcal{Y}_f$$

and that we compute $\Delta \mathcal{U}$ by replacing \mathcal{Y} in this equation by $\mathcal{T} - \mathbf{1}d(k)$. So in steady state we have

$$\Theta \Delta \mathcal{U} = \mathcal{T} - \mathbf{1}d_\infty - \mathcal{Y}_f \quad (1.33)$$

$$= \mathcal{T} - \mathbf{1}y_{p\infty} + (\mathbf{1}y_{m\infty} - \mathcal{Y}_f) \quad (1.34)$$

But note that

$$\mathbf{1}y_{m\infty} - \mathcal{Y}_f = 0 \quad (1.35)$$

The reason for this is a little subtle: the free response starts from a constant level of $y_{m\infty}$, since it is the free response of the model. But we are assuming steady-state conditions, so that the free response remains constant at its initial value. So we conclude that

$$\Theta \Delta \mathcal{U} = \mathcal{T} - \mathbf{1}y_{p\infty} \quad (1.36)$$

But in the steady state $\Delta \mathcal{U} = 0$, and hence $\mathcal{T} = \mathbf{1}y_{p\infty}$, or $r_i = y_{p\infty}$. So the reference trajectory is constant, with value $y_{p\infty}$. But $r_i = s_\infty - \lambda^i(s_\infty - y_{p\infty})$, so we conclude that

$$y_{p\infty} = s_\infty \quad (1.37)$$

which shows that the plant output reaches the set-point value, despite the modelling error.

Note that this analysis has assumed that the closed loop is stable, which is not guaranteed, of course.

This same ‘trick’ also gives offset-free tracking of constant set-points if there is an unknown constant additive disturbance acting at either the input or the output of the plant, since both of these have the same effect as an incorrect steady-state gain of the model. In fact, the use of (1.31) can be seen as explaining any error between the current plant and model outputs as a disturbance acting at the plant output, and assuming that this disturbance will continue to act throughout the prediction horizon. This method of obtaining offset-free tracking is used in every commercial predictive control product known to the author. (See Appendix A for descriptions of some of these products.)

The reader familiar with conventional control may wonder how it is possible to eliminate steady-state errors without the use of ‘integral action’. We shall show later that the scheme just described can in fact be interpreted as incorporating a ‘discrete-time integrator’ in the control law in a particularly simple way.

The use of the disturbance estimate (1.30) is in fact the only way in which *feedback* enters the predictive control law. Using it in (1.31) corresponds to assuming that the

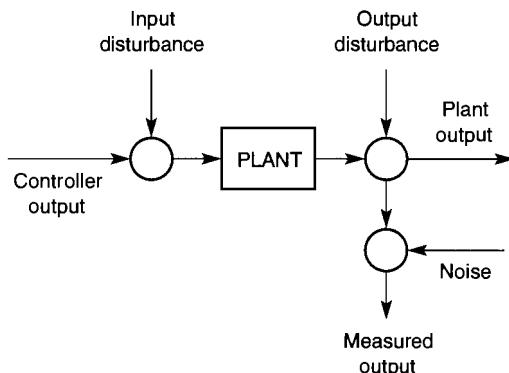


Figure 1.8 Input and output disturbances and measurement noise.

currently estimated disturbance will persist at the same level into the future. This is not the only assumption that can be made; for example, it could be assumed that the disturbance dies away exponentially with a known time constant, though the offset-free tracking property would be lost in that case.

The program `noisympc` has been written, and is available on this book's web site. This program provides offset-free tracking, but in addition simulates the effects of measurement noise, and of input and output disturbances, as shown in Figure 1.8. The vectors `noise`, `udist` and `ydist` contain the measurement noise and the input and output disturbances. The actual plant output, with the output disturbance included (variable `ypd`), and the measured output (variable `ypm`), are formed by the statements

```

ypd(k) = yp(k) + ydist(k);
ypm(k) = ypd(k) + noise(k);

```

One has to be careful to use `ypm` instead of `yp` in the definition of the reference trajectory (`errornow = setpoint(k)-ypm(k)`) and of the output disturbance estimate (`d = ypm(k) - ym(k)`). The model output is left unaltered — because of the ‘independent model’ idea, and because neither the noise nor the disturbances are known directly to the controller. Input disturbances are included by modifying the definition of the vector `uppast`, which holds the past values of the input received by the plant:

```
uppast = [uu(k)+udist(k); uppast(1:length(uppast)-1)];
```

The graphical display is now a little more elaborate: both the real and the measured plant outputs are shown. Also, both the controller output and the effective plant input are plotted — if input disturbances are present these are not the same. As supplied, the program simulates noise of standard deviation 0.1, a constant input disturbance of value 0.1, and an output disturbance which switches from value +0.1 to -0.1 halfway through the simulation.

Example 1.7

Figure 1.9 shows the graphical output from simulating the same system as shown in Figures 1.5 and 1.7, with noise and disturbances present, using the program `noisympc`.

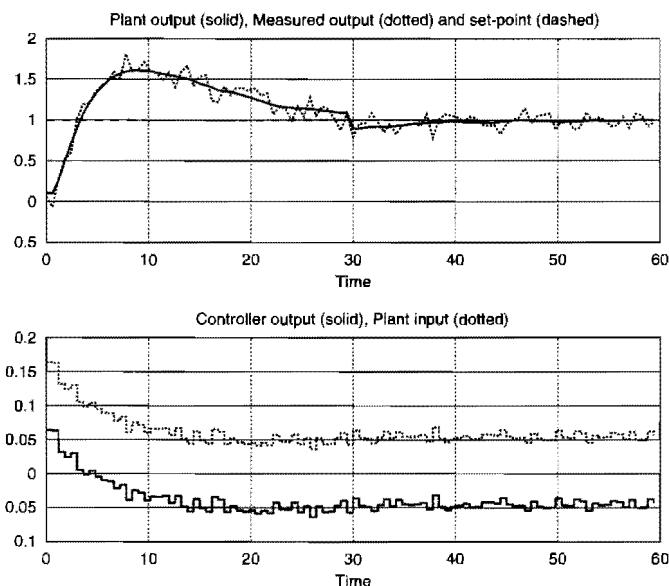


Figure 1.9 Simulation of system with input and output disturbances and measurement noise using `noisympc`.

Note how the plant output reaches the correct set-point, despite the unknown input and output disturbances. The output disturbance switches from $+0.1$ to -0.1 halfway through the simulation, at $t = 30$.

1.6 Unstable plant

Some plants contain integrations, typically if the output is the level of a tank containing liquid, or the position or attitude of an inertial object such as an aircraft. These are no longer asymptotically stable, but the basic predictive controller as developed in the previous two sections can still be applied to such ‘marginally stable’ plants — see Exercise 1.8.

However, if the plant is unstable, in the sense that its impulse or pulse response is unbounded (continuous-time poles in the right half-plane, discrete-time poles outside the unit circle), then the scheme as presented will no longer work. The problem is precisely the use of the ‘independent model’ that was previously advocated as a desirable feature, because this means that the model is running ‘open loop’. If the model is unstable and running open-loop, then its behaviour will rapidly diverge from the real plant behaviour, and numerical errors in the simulation will be rapidly amplified by the instability, so that its value as a predictor of plant behaviour will be completely lost.

The only remedy in this case is to stabilize the model somehow, and that requires the ‘independence’ of the model to be given up. The easiest way of stabilizing the model is to ‘realign’ it on the plant outputs. That is, when running it to generate predictions

of plant behaviour, the initial conditions are taken to be those of the plant, not those resulting from the previous model simulation step. Suppose that the unstable model corresponds to the difference equation

$$y_m(k) = - \sum_{i=1}^n a_i y_m(k-i) + \sum_{i=1}^n b_i u(k-i) \quad (1.38)$$

then the ‘realigned’ model generates predictions according to

$$\hat{y}(k+1|k) = - \sum_{i=1}^n a_i y_p(k+1-i) + b_1 \hat{u}(k|k) + \sum_{i=2}^n b_i u(k+1-i) \quad (1.39)$$

$$\begin{aligned} \hat{y}(k+2|k) = & - a_1 \hat{y}(k+1|k) - \sum_{i=2}^n a_i y_p(k+2-i) + \\ & + b_1 \hat{u}(k+1|k) + b_2 \hat{u}(k|k) + \sum_{i=3}^n b_i u(k+2-i) \end{aligned} \quad (1.40)$$

⋮

That is, actual past plant outputs (and inputs) are used whenever they are available. We shall see later that this always stabilizes the model, and that it can be interpreted as implementing a deadbeat observer for the plant, so the model does not diverge much from the plant. But it does not necessarily lead to a stable closed loop. Note that (1.39) introduces feedback into the predictive controller, in a manner different from the use of (1.30).

The corresponding modification is easily made to basicmpc. Assuming that the plant and model have the same order (degree of denominator), then all that is required is to replace the line

```
ympast = [ym(k+1); ympast(1:length(ympast)-1)];
```

by

```
ympast = yppast;
```

The program unstampc has this modification incorporated.

Now a drawback of the ‘realigned’ model implementation becomes apparent. The simple technique introduced in the previous section does *not* give offset-free tracking. The reason is that (1.35) no longer holds, because the model is now driven not only by the input $u(k)$, but also by the plant output $y_p(k)$, which acts as an input to the model.

It is still possible to get offset-free tracking with a ‘realigned’ model, by explicitly building in integral action in the controller, but this can no longer be done as simply as before. We shall investigate this in more detail later in the book. Other ways of stabilizing the model are possible: Richalet [Ric93b] factors unstable models into stable and unstable parts, then leaves the stable part ‘independent’ and stabilizes only the unstable part — this is described in detail in Section 5.3. A ‘realigned’ model, as described in this section, is assumed in *Generalized Predictive Control* (GPC), one of the best-known variations of predictive control [CMT87].

Example 1.8

The transfer function from the rotor angle to the forward speed of a helicopter in a particular flight condition is given by

$$\frac{9.8(s^2 - 0.5s + 6.3)}{(s + 0.6565)(s^2 - 0.2366s + 0.1493)}$$

This has zeros at $+0.25 \pm 2.5j$ and poles at $+0.118 \pm 0.37j$. It is therefore both non-minimum-phase and unstable — certainly a difficult plant to control. Figure 1.10 shows the response obtained using the program `unstampc` when an exact model is assumed, and parameters $T_{ref} = 6$ and $T_s = 0.6$ are used, with a single coincidence point $P_1 = 8$, and $H_u = 1$. This example shows that predictive control, although simple and intuitive in its formulation, can produce some very sophisticated control action when required. In order to stabilize this unstable plant, classical theory tells us that the Nyquist locus of the loop-gain frequency response must encircle the point -1 twice, and this can only be achieved by supplying sufficient phase lead at the appropriate frequencies. Doing this successfully requires considerable skill on the part of a designer using classical ‘loop-shaping’ techniques. Predictive control achieves this implicitly, given only some ‘reasonable’ design specifications.

On the other hand, this does not mean that classical theory can be forgotten. For this example it tells us, for instance, that the loop gain must increase to values greater than 1 at a frequency no lower than 0.4 rad/sec, approximately, and must decrease again to values smaller than 1 at a frequency no higher than 2.5 rad/sec, approximately [Mac89, Chapter 1]. Its range of possible behaviours is rather restricted, therefore, and this

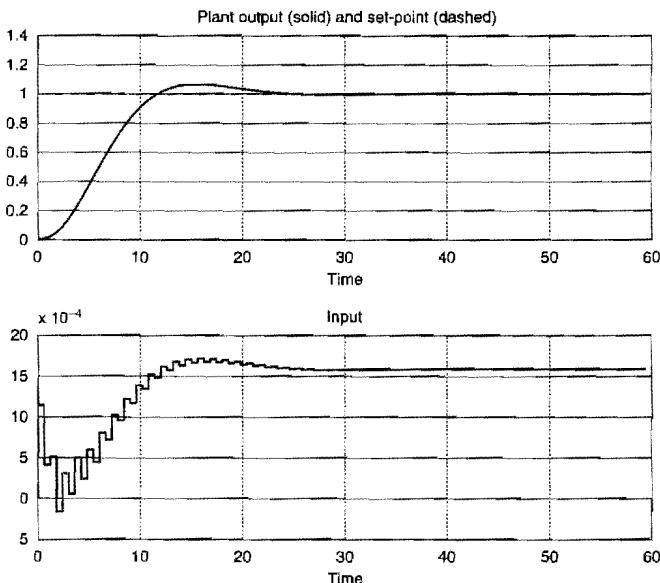


Figure 1.10 Control of unstable helicopter using `unstampc`.

knowledge certainly helps to give the predictive controller a ‘reasonable’ specification to achieve.

The reader will find that improving the performance — for example, reducing the overshoot or speeding up the response — by adjusting the parameters of the predictive controller, is not so easy in this case.

1.7

Early history and terminology

Predictive control appears to have been proposed independently by several people, more or less simultaneously. It is notoriously difficult to analyze the history of an idea, so no attempt will be made here to settle the rival claims to originality. Dates of publications give some idea of precedence, of course, but the pioneers were mostly industrial practitioners who implemented predictive control several years before the first publications appeared, so the publication dates do not tell the whole story.

Richalet *et al* [RRTP78], of the French company *Adersa*, proposed predictive control under the name *Model Predictive Heuristic Control*. The emphasis was on a control methodology which could be applied to problems too difficult to be handled by conventional PID control, but which was based on intuitive concepts and offered ease of tuning. Constraint handling and optimality were not the principal objectives.

Cutler and Ramaker [CR80] also proposed predictive control, calling it *Dynamic Matrix Control*, or *DMC*. This emphasized optimal plant operation under constraints, and computed the control signal by repeatedly solving a linear programming (LP) problem. *DMC* went on to become the most well-known of the commercial predictive control products. A patent for it was granted to Prett *et al* in 1982 [PRC82].

The earliest patent, however, appears to be that granted to Martin-Sanchez in 1976 [San76], who called his method simply *Adaptive Predictive Control*. As the name implies, the emphasis here was on exploiting the presence of the internal model to obtain adaptive control, by adapting the model and relying on the optimization to compute the appropriate control signals.

All of these proposals shared the essential features of predictive control: an explicit internal model, the receding horizon idea, and computation of the control signal by optimizing predicted plant behaviour. Early academic publications containing similar ideas include those of Propoi [Pro63], Kleinman [Kle70], Kwon and Pearson [KP75], and Rouhani and Mehra [RM82]. A very early contribution which addressed a rather different problem, but introduced some ideas similar to those now standard in predictive control, was that of Coales and Noton [CN56]. This provided an approximate solution to the problem of synthesizing minimum-time ‘bang-bang’ optimal controls; it did so by generating predictions of plant behaviour, using a fast simulation model and assuming constant control signals, and deciding whether to switch the sign of the control signal on the basis of the predicted behaviour at the end of the prediction horizon. The horizon was not constant, however, so it was not a receding horizon strategy; but it did contain the essential ideas of using predictions generated from an explicit internal model (which was assumed at that time to be an analog computer), and of computing an optimal control signal on-line in real time.⁵

⁵ For an extended discussion of this proposal, and later developments, see [Rya82].

Of course, the idea of using predictions is a very old and common one in control. The well-known Smith predictor [Smi57], for example, which is a controller for plants with large time delays, has at its heart a model of the plant without the time delay, thus providing a prediction of the plant output. Obtaining the prediction is a means of obtaining phase lead, which offsets the large phase lag caused by the delay in the plant. More fundamentally, any controller which uses derivative action, or a phase lead compensator, can be viewed as providing a prediction of some signal. In a basic ‘proportional and derivative’ controller, for instance, the control signal depends on $y + T(dy/dt)$, where y is the plant output. This can be seen as a ‘ T -second-ahead’ prediction of the output, on the assumption that dy/dt will remain constant over this interval. The point of this discussion is to indicate that we will not consider every control law which involves some kind of prediction as being an instance of ‘predictive control’. Only those which include, as a minimum, an explicit model of the plant, the receding horizon idea, and some kind of optimization, will be considered in this book.

There is a plethora of names denoting particular variants of predictive control, usually with corresponding acronyms. Examples of these are:

- ❖ Dynamic Matrix Control (DMC),
- ❖ Extended Prediction Self-Adaptive Control (EPSAC),
- ❖ Generalized Predictive Control (GPC),
- ❖ Model Algorithmic Control (MAC),
- ❖ Predictive Functional Control (PFC),
- ❖ Quadratic Dynamic Matrix Control (QDMC),
- ❖ Sequential Open Loop Optimization (SOLO),

and so on. Generic names which have become widely used to denote the whole area of predictive control are *Model Predictive Control*, or *MPC*, and *Model-Based Predictive Control*, or *MBPC*.

1.8 Predictive control in the control hierarchy

Figure 1.11 shows how predictive control is typically used in the process industries at present. At the top level there is determination of set-points, usually by means of steady-state optimization. This steady-state optimization may itself be performed at more than one level; plant-wide optimization of strategic set-points may be performed once a day, but more detailed optimization, at the unit level, may be performed every hour, say. This optimization is based on economic requirements — so it may produce time-varying set-points, for instance — but it usually does not take any account of the dynamic characteristics of the plant. At the bottom level there are traditional local controllers, individually controlling pressures, flows, temperatures, and so on. These are typically proportional and proportional-integral (PI) controllers, or occasionally three-term PID controllers with derivative action. At the very lowest level, there are control loops associated with individual actuators, such as valve-positioning servos.

In between the static optimization layer and the local controller layer, there is

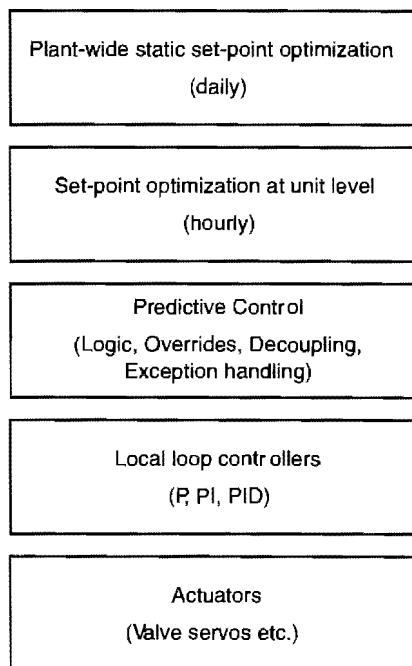


Figure 1.11 Typical use of predictive control — current practice.

traditionally a complex layer of logic, overrides, decoupling networks and exception handling, to deal with all those conditions which cannot be handled by the simple ‘one set-point, one control loop’ paradigm. This layer usually consists of a collection of *ad hoc* solutions to individual problems, and so tends to evolve during the lifetime of a plant, rather than being the product of an integrated design process. Each solution is unlikely to take account of the behaviour of the whole process or plant, and the overall behaviour of this layer is very far from being optimized.

It is in replacing this *ad hoc* layer that predictive control currently has its most successful niche in the process industries. The most commonly used combination, of linear dynamic models and constrained optimization, allows it to handle very many of the ‘exceptional’ (but not rare) conditions previously handled by the layer of *ad hoc* ‘fixes’. Furthermore, because predictive control is an integrated solution to handling all these problems, it can provide dramatically better performance than the technology it is replacing.

The fact that predictive control is usually implemented on top of traditional local controllers has had important implications for its acceptance and development. Firstly, it has allowed operating companies to be relatively brave in introducing this new technology; if a predictive controller starts to misbehave, it is usually possible to disable it, and let the local loop controllers hold the plant at the last set-points they received from higher levels. The great majority of process plant is stable in this condition, so although this may not be the most profitable condition, it is at least a safe one. This

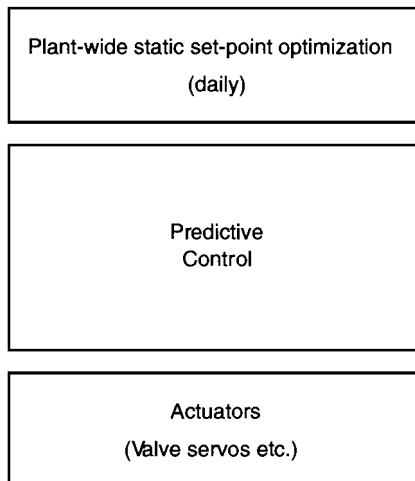


Figure 1.12 Use of predictive control — future trend?

is one of the reasons why there were many predictive controllers installed even before there was a satisfactory stability theory for predictive control, and indeed why many current installations take no account of that stability theory. Secondly, it has resulted in commercial predictive controllers being developed almost exclusively for stable plant. The plant ‘seen’ by the predictive controller is one which is already running under closed-loop control, and is almost invariably stable.

Not all predictive controllers play the role shown in Figure 1.11. There are some which are used as replacements of low-level controllers, for instance in servomechanisms, and in applications where adaptive control is required [CBB94, Cla88, ReADAK87]. It can be expected, especially as predictive control spreads to new application areas, that it will become increasingly common for predictive control to include the lower level currently populated by PI controllers, leaving only the actuator servos below it. This is illustrated in Figure 1.12. In applications such as flight or spacecraft control, for example, there is no scope for individual loops above the actuator level, because holding each set-point (such as climb rate, or 3-axis attitude) requires coordinated action by several actuators. Faster control update rates are needed if lower-level control functions are to be performed, but these are becoming possible as computing hardware becomes faster. The other feature shown in Figure 1.12 is integration of economic (short-term set-point) optimization with the dynamic performance optimization currently implemented by predictive control.

1.9 General optimal control formulation

The idea of posing control problems as problems of constrained optimization is not new. In fact all the ‘classical’ theory of Optimal Control, as developed between, say, 1955 and 1970, was driven by problems of constrained optimization arising out of the needs

of the aerospace industry, particularly by military needs in flight and missile control, and by the problems of launching, guiding and landing space vehicles.

In some ways this theory solved an extremely wide range of problems. Suppose that the ‘plant’ being controlled has an input vector u and a state vector x , and has nonlinear behaviour governed by the vector differential equation

$$\frac{dx}{dt} = f(x, u).$$

Also suppose that the control objective is to minimize a ‘cost function’ (or ‘value function’) which has the form

$$V(x, u, t) = \int_0^T \ell(x(t), u(t), t) dt + F(x(T))$$

where $\ell(x(t), u(t), t)$ is some function which is never negative, and that the control input is constrained to be in some set $u(t) \in U$.

This is a very general formulation, and just about every sensible problem can be represented in this form by using suitable functions f , F and ℓ .

Example 1.9

Suppose the plant is a rocket which is to be transferred from the launch site — state x_0 — to a low-earth orbit — state x_f — using as little fuel as possible, and the vector of inputs includes the fuel flow rate as the first element, u_1 . This problem can be represented by setting

$$\ell(x(t), u(t), t) = |u_1(t)|$$

and

$$F(x(T)) = \begin{cases} \infty & \text{if } x(T) \neq x_f \\ 0 & \text{if } x(T) = x_f \end{cases}$$

The vector function f depends on the dynamics of the rocket. The optimization will be subject to the constraint $0 \leq u_1(t) \leq u_{max}$, where u_{max} is the maximum possible fuel flow rate.

In principle, the solution to such general problems is known: you have to solve the so-called Hamilton–Jacobi–Bellman equation [BH75]:

$$\frac{\partial}{\partial t} V^0(x, t) = \min_{u \in U} H \left(x, u, \frac{\partial}{\partial x} V^0(x, t) \right)$$

where $H(x, u, \lambda) = \ell(x, u) + \lambda f(x, u)$, with the boundary condition $V^0(x, T) = F(x)$. Once you have solved this, you choose the control input signal as the one which minimizes H :

$$u^0(x, t) = \arg \min_{u \in U} H \left(x, u, \frac{\partial}{\partial x} V^0(x, t) \right).$$

Note that this is a *feedback* control law, because it depends, at time t , on $x(t)$, which itself depends on what has happened over the whole interval from the initial time to the current time. (*Note: You do not have to know where all this comes from. The point is to see that it is nasty.*)

Unfortunately, it is virtually impossible to solve this partial differential equation in most cases! A few particular problems have been solved (of the type: ‘Minimize fuel (or time) needed to get from A to B’), but that is all. So it is necessary to consider some more specific problems in order to get solutions in practice.

One way to go is to assume that the plant is *linear*, so that the function $f(x, u)$ takes the special form:

$$\dot{x} = A(t)x(t) + B(t)u(t)$$

and that the functions ℓ and F are *quadratic* functions:

$$\ell(x(t), u(t), t) = x(t)^T Q(t)x(t) + u(t)^T R(t)u(t)$$

$$F(x(T)) = x(T)^T Sx(T)$$

where $Q(t)$, $R(t)$ and S are square, symmetric, positive-definite matrices. In that case the Hamilton–Jacobi–Bellman equation simplifies to an ordinary differential equation (the *Riccati* equation) which can be solved, and which leads to a linear (but time-varying) feedback law:

$$u(t) = F(t)x(t)$$

where $F(t)$ is a matrix (the ‘state feedback’ matrix) which depends on the solution of the Riccati equation [BH75, KR72]. Probably more than 95% of the development of optimal control has been devoted to going in this direction. It leads to a very complete and powerful theory, which has been widely applied — particularly in its ‘dual’ form for the filtering problem, in which case it leads to the theory of Kalman filters. But it has nothing to say about constraints or other nonlinearities.

The other way to go is down the road of Predictive Control. The difficulty of the general optimal control problem arises because it is a *function* optimization problem. Instead of ordinary calculus, it needs the Calculus of Variations to find the optimal input function among all possible ones. (For a very nice account of how this theory developed see [SW97].) The main idea of Predictive Control is to avoid this difficulty by restricting the set of possible inputs to such an extent that the optimization is performed only over a finite set of parameters or ‘decision variables’, rather than over a set of functions. Most commonly this is done as has already been seen: by cutting up time into discrete intervals, optimizing over a finite horizon, and hence over a finite number of intervals, and assuming that the input signal is held constant during each interval, or even during several intervals. The future values of the input signal are then the decision variables. Furthermore, predictive control does not usually attempt to solve the feedback problem; it solves a sequence of open-loop problems, with feedback entering rather indirectly, the latest measurements being used to provide initial conditions for the next open-loop optimization. Other ways of reducing the problem to an ‘ordinary’ optimization problem are possible — see Sections 5.6 or 8.4, for example.

1.10 What is in this book

This chapter has introduced the main concepts involved in predictive control:

- ❖ An *internal model* capable of simulating the plant behaviour faster than real time.
- ❖ A *reference trajectory* which defines the desired closed-loop behaviour.
- ❖ The *receding horizon* principle.
- ❖ Characterizing the assumed future input trajectory by a finite number of ‘moves’, or other parameters.
- ❖ On-line *optimization*, possibly constrained, to determine the future control strategy.

These concepts together provide an intuitively appealing way, and a powerful way, of controlling various systems. The rest of this book elaborates on the use of these concepts in predictive control.

Most of the interest in predictive control comes from its ability to handle constraints, and from the natural way in which it can be applied to the control of multivariable systems, without losing its intuitive aspect. Chapter 2 introduces a standard formulation of predictive control which assumes the presence of constraints, and of a multivariable plant. For this purpose a state-space setting is used — it is the most convenient for multivariable problems, and it includes all other approaches. Some of the questions which have arisen in this chapter, especially the independent model, offset-free tracking and stabilization of an unstable model, are all illuminated by discussing them in the context of state observers. So far we have assumed a small number of coincidence points — only one in all the examples. More commonly, coincidence points are spaced ‘densely’ over the prediction horizon — often there is a coincidence point at each point in the horizon. Similarly, more than one parameter is commonly used to characterize the assumed future control signal ($H_u > 1$).

Chapter 3 then considers how to solve the on-line optimization problems which arise in predictive control when constraints are present. It includes discussion of relatively recent algorithms, such as ‘interior point’ methods. It also discusses the structure of the resulting controller, assuming the standard formulation adopted in Chapter 2. A major problem which arises when constraints are present is the possibility that the optimization problem becomes infeasible; strategies are needed for handling this eventuality. A commonly adopted strategy is to ‘soften’ the constraints, namely to allow their violation as a last resort if no other possibility exists. Chapter 3 includes an explanation of how this can be done.

It was seen in Section 1.3 that the only information really needed to implement a predictive controller is the plant’s step response — so long as the plant is assumed to behave linearly. It is therefore very understandable that the original developments of predictive control assumed a model in the form of a step response, sometimes referred to as a ‘dynamic matrix’.⁶ Some developments, especially the very well-known *Generalized Predictive Control*, or *GPC*, assumed the model to be in the form of a transfer function. Chapter 4 deals with formulations of predictive control based on step-response and transfer function models, and shows how these are related to the state-

⁶ Hence the name of the commercial product *Dynamic Matrix Control*, or *DMC* — see Appendix A.

space setting used in the rest of the book. It tries to persuade the reader that it is best to use state-space models.

Many variations on the basic formulation are possible. If there are measured disturbances, conventional control attempts to compensate for them by using ‘feedforward’. The same is possible with predictive control. It is possible to use other criteria than a quadratic one for finding the optimal future input trajectory. And it is possible to be creative when defining required performance. These and other possibilities are introduced in Chapter 5.

Chapter 6 shows how to adjust the formulation and the tuning parameters so that closed-loop stability is guaranteed. The two principal methods are to introduce ‘terminal constraints’ at the end of the prediction horizon, or to distribute coincidence points evenly along an infinite prediction horizon. Some other methods are also introduced. All the analysis in this chapter is concerned with ‘nominal stability’, namely the stability of the closed loop under the assumption that an exact model of the plant is available. The important problem of ensuring that a predictive controller is robust to inexact modelling is addressed in Chapter 8, which presents some recent research proposals — this problem is still at the research stage.

Predictive controllers are ‘tuned’ by adjusting parameters such as the prediction horizon, the time constant of the reference trajectory, and weights in the criterion being optimized. Satisfactory tuning may not be easy to achieve. Chapter 7 considers this problem. It introduces some ‘classical’ analysis from linear control theory, which is useful because Chapter 3 shows that predictive controllers for the standard formulation actually implement linear control laws, so long as all the constraints are inactive, or if a fixed set of constraints is active.

Chapter 9 gives an extended account of the application of predictive control to the well-known Shell heavy-oil fractionator problem. It also gives a briefer account of its application to the Newell–Lee evaporator, in order to demonstrate the effects of plant nonlinearities.

Finally, Chapter 10 presents some wider perspectives on predictive control: the possibilities of constraint management within the predictive control framework, the use of nonlinear models, and the potential of predictive control as a basis for fault-tolerant control systems are all discussed here.

The appendices provide details of some leading predictive control commercial products, of the software described in this chapter, and of *MATLAB’s Model Predictive Control Toolbox*, the use of which will be assumed in later chapters.

Exercises

- 1.1** Continue Example 1.3 for two more steps to find the optimal values of $\hat{u}(k + 1|k + 1)$ and $\hat{u}(k + 2|k + 2)$. (Note that the steady-state gain of the model is $20/3$, so that the input signal should converge to $3/(20/3) = 0.45$ if the control scheme is stable.)

Verify that the output at time $k + 2$, $y(k + 2)$, is not the same as that predicted at time k , $\hat{y}(k + 2|k)$, even though a perfect model and absence of disturbances are assumed.

1.2 Continue Example 1.4 for two more steps, using *MATLAB* or other suitable software (or a calculator) as necessary.

1.3 Repeat Examples 1.3 and 1.4 using a ‘straight-line’ reference trajectory, with (1.2) replaced by

$$\epsilon(k+i) = \begin{cases} \left(1 - \frac{iT_s}{T_{ref}}\right)\epsilon(k) & \text{if } iT_s < T_{ref} \\ 0 & \text{otherwise} \end{cases}$$

1.4 Verify that (1.24) is equivalent to (1.23).

1.5 A plant has a continuous-time transfer function

$$\frac{2e^{-1.5s}}{1+7s}$$

that is, it has a steady-state gain of 2, a time constant of 7 sec, and a time delay of 1.5 sec. The set-point has a constant value $s(t) = 3$, and the reference trajectory is exponential with $T_{ref} = 5$ sec. A predictive controller operates with a sampling interval $T_s = 0.5$ sec, and coincidence points at 6 sec and 10 sec ahead of the current time. The future control input is assumed to be constant over the prediction horizon ($H_u = 1$).

If the plant output is constant at 1 ($y(k) = 1$, $\dot{y}(k) = 0$), find the optimal input $u(k)$

- (a) by using the continuous-time model directly,
- (b) by first obtaining an equivalent discrete-time model.

(Use *MATLAB*’s *Control System Toolbox* as necessary. In particular, the functions `step` and `lsim` are useful for both parts, and `c2d` for part (b).)

1.6 Explain why a coincidence point nearer than 1.5 sec into the future ($P_1 \times T_s < 1.5$) should not be chosen in Exercise 1.5.

1.7 Show how equation (1.31) and the subsequent analysis of the offset-free tracking property simplifies if there is only one coincidence point, and $H_u = 1$.

1.8 Suppose the plant includes an integrator, so that its z -domain transfer function is $1/(z - 0.5)(z - 1)$. Its gain is modelled incorrectly, so that the model transfer function is $1.1/(z - 0.5)(z - 1)$. Apply the programs `basicmpc` and `trackmpc` in this case. Verify that using `basicmpc` results in a steady-state error when the set-point is constant (despite having an integrator in the plant), whereas using `trackmpc` results in offset-free tracking.

1.9 Explain the following surprising fact. If the basic algorithm contained in program `basicmpc` is used, without the modification to obtain offset-free tracking, and if $T_{ref} = 0$, so that the reference trajectory returns immediately to the set-point trajectory, then the control signal is completely unaffected by measurement noise. Why is this no longer the case when offset-free tracking is included, as in program `trackmpc`?

- 1.10** The discussion of unstable systems emphasizes the problem of having an unstable *model* running open-loop inside the controller. This implies that it may be possible to control an unstable plant using the ‘independent model’ implementation used in `basicmpc` and `trackmpc`, providing that the *model* is stable. This can in fact be done, providing that the plant and model are ‘close enough’, in some sense, for the controller to stabilize the plant despite the modelling error. The reader can check that this occurs for the following example:

$$\begin{aligned}\text{Plant:} \quad & 1/(z - 0.5)(z - 1.01) \\ \text{Model:} \quad & 1/(z - 0.6)(z - 0.99)\end{aligned}$$

by running `trackmpc` with $T_{ref} = 6$, $T_s = 0.6$, a single coincidence point $P_1 = 8$, $H_u = 1$, and a constant set-point. (The transient from zero initial conditions takes some time to complete, so the parameter `tend` should be increased to 200.) Note that in this case not only is the instability unmodelled, but the steady-state gain of the model even has the incorrect sign.

The reader should also verify that stability of the closed loop is easily lost when this kind of mismodelling is present. (For example if the plant has a pole at 1.05, while the model has it at 0.95.)

- 1.11** Show, by using the program `unstampc`, that the parameters used in Example 1.8 give a reasonably robust controller, in the sense that some plant–model mismatch can be tolerated without losing closed-loop stability. Observe that in such cases a steady-state offset exists between the output and the set-point.

(For example, changing the gain factor from 9.8 to 9.9, or the constant term in the second denominator factor from 0.1493 to 0.15 — *in the model only, of course* — leaves the closed loop stable.)

- 1.12** Modify the program `unstampc` by changing the computation of the optimal input trajectory to the form given in (1.31). Verify that offset-free tracking is *not* obtained if the steady-state gain is modelled incorrectly, or if there is a constant input or output disturbance acting on the plant.

- 1.13** Write a version of the program `unstampc` which allows the plant and the model transfer functions to have different orders.

- 1.14** In (1.25)–(1.27) we assumed that the control signal changes at every one of the first H_u sampling intervals, and then it remains constant: $\hat{u}(k + H_u - 1|k) = \hat{u}(k + H_u|k) = \dots$. Suppose that the number of changes remains the same, but the instants at which the changes occur are arbitrarily distributed over the prediction horizon: $\hat{u}(k + i|k) = u_j$ for $M_j \leq i < M_{j+1}$, $j = 1, 2, \dots, H_u$, $M_1 = 0$, $M_{H_u} < H_p$. Describe the corresponding changes that need to be made to $\Delta\mathcal{U}$ and Θ . (Note that their dimensions should remain unchanged.)

Comment: In practice it is often desirable to have more frequent changes of \hat{u} near the beginning of the prediction horizon, and less frequent later; for example, in non-minimum-phase systems which exhibit fast ‘inverse response’ followed by slow settling. In the *Model Predictive Control Toolbox* this is referred to as

‘blocking’, because the input signal is assumed to remain constant over ‘blocks’ of sampling intervals.

- 1.15** Using the programs `basicmpc`, `trackmpc` and `noisympc` (available on this book’s web site), and the system defined in Example 1.6, observe the effects of changing the parameters T_{ref} , T_s , the number and spacing of coincidence points, and the control horizon H_u (variable `M`). Also try non-constant set-point trajectories.

Chapter 2

A basic formulation of predictive control

2.1	State-space models	36
2.2	A basic formulation	41
2.3	General features of constrained predictive control	47
2.4	Alternative state variable choices	49
2.5	Allowing for computational delay	50
2.6	Prediction	53
2.7	Example: Citation aircraft model	64
	Exercises	70

2.1 State-space models

2.1.1 Form of the model

In most of this book we will assume a linearized, discrete-time, state-space model of the plant, in the form

$$x(k+1) = Ax(k) + Bu(k) \quad (2.1)$$

$$y(k) = C_y x(k) \quad (2.2)$$

$$z(k) = C_z x(k) \quad (2.3)$$

where x is an n -dimensional state vector, u is an ℓ -dimensional input vector, y is an m_y -dimensional vector of measured outputs, and z is an m_z -dimensional vector of outputs which are to be controlled, either to particular set-points, or to satisfy some constraints, or both. The variables in y and z will usually overlap to a large extent, and frequently they will be the same — that is, all the controlled outputs will frequently be measured.

We will often assume that $y \equiv z$, and we will then use C to denote both C_y and C_z , and m to denote both m_y and m_z . The index k counts ‘time steps’.

The reason for using this standard form is mainly that it connects well with the standard theory of linear systems and control. We will often generalize this model by including the effects of measured or unmeasured disturbances, and of measurement noise.

We are going to assume that the sequence of actions at time step k is the following:

1. Obtain measurements $y(k)$.
2. Compute the required plant input $u(k)$.
3. Apply $u(k)$ to the plant.

This implies that there is always some delay between measuring $y(k)$ and applying $u(k)$. For this reason there is no ‘direct feed-through’ from $u(k)$ to $y(k)$ in the measured output equation (2.2), so that the model is ‘strictly proper’. Taking computational delay into account is considered in more detail in Section 2.5.

The controlled outputs $z(k)$ could, in principle, depend on $u(k)$. That is, the description

$$z(k) = C_z x(k) + D_z u(k) \quad (2.4)$$

with $D_z \neq 0$ could be appropriate and useful in some cases. This would, however, complicate the computation of the optimal $u(k)$ slightly. This complication can be avoided, without losing anything, by defining a new vector of controlled outputs

$$\tilde{z}(k) = z(k) - D_z u(k) \quad (2.5)$$

which clearly depends on $x(k)$ only, without any direct feed-through from $u(k)$: $\tilde{z}(k) = C_z x(k)$. The corresponding changes to the cost function and constraints are easily made — see Exercises 2.12 and 2.13. We shall therefore assume that the controlled outputs are defined by (2.3).

2.1.2 Linear model, nonlinear plant

The relationship between the linear model (2.1)–(2.3) and the real plant needs careful consideration for predictive control. In most control methodologies the linear model is used off-line, as an aid to analysis and design. In predictive control it is used as part of the control algorithm, and the resulting signals are applied directly to the plant. Therefore careful attention must be paid to appropriate treatment of measurements before using them in the control computation algorithm, and of the computed control signal.

The plant in reality behaves in some complicated nonlinear fashion. Suppose that its state vector X evolves according to some nonlinear differential equation

$$\frac{dX}{dt} = f(X, U, t) \quad (2.6)$$

where U is the vector of inputs. (This is a simplification. Many processes are naturally described by implicit equations of the form

$$f(X, dX/dt, U, t) = 0 \quad (2.7)$$

and the extraction of the explicit form (2.6) may not be easy — but that is not important here.) Suppose the process is in some state $X = X_0$, with input $U = U_0$, and consider the effects of small perturbations $X = X_0 + x$, $U = U_0 + u$, with $\|x\|$ and $\|u\|$ both small:

$$\frac{dX}{dt} = f(X_0 + x, U_0 + u, t) \quad (2.8)$$

$$\approx f(X_0, U_0, t) + \left. \frac{\partial f}{\partial X} \right|_{(X_0, U_0, t)} x + \left. \frac{\partial f}{\partial U} \right|_{(X_0, U_0, t)} u \quad (2.9)$$

where quadratic and higher-order terms in x and u have been neglected. The expressions $\partial f / \partial X|_{(X_0, U_0, t)}$ and $\partial f / \partial U|_{(X_0, U_0, t)}$ denote matrices of partial derivatives, evaluated at (X_0, U_0, t) . Denote these ('Jacobian') matrices by A_c and B_c , respectively. Since $X = X_0 + x$ and X_0 is a particular value of X , we have $dX/dt = dx/dt$. Hence we have the linearized model

$$\frac{dx}{dt} = A_c x + B_c u + f(X_0, U_0, t) \quad (2.10)$$

If (X_0, U_0) is an *equilibrium point* (that is, a possible steady state) at time t , namely if $f(X_0, U_0, t) = 0$, then clearly this model simplifies further, to the familiar form of continuous-time linear state-space model used very widely in systems and control theory, namely

$$\frac{dx}{dt} = A_c x + B_c u \quad (2.11)$$

This is the most common case: linearized models are usually found for the neighbourhood of an equilibrium point.

Sometimes it is useful to linearize at a point (X_0, U_0) which is *not* an equilibrium point. In this case we can define $(\dot{X})_0 = f(X_0, U_0, t)$ ¹ and

$$\dot{x} = \frac{dx}{dt} - (\dot{X})_0 \quad (2.12)$$

to obtain

$$\dot{x} = A_c x + B_c u \quad (2.13)$$

The form of this equation is the same as the one obtained when linearizing about an equilibrium point, and it can be used in much the same way. But it must be remembered that \dot{x} is now a perturbation of the time derivative dx/dt , not the derivative itself. This affects the formation of predictions, for instance. The exact linearization about a non-equilibrium trajectory of a system is a time-varying linear model, since the state X does not remain constant at X_0 . When using such models for predictive control, one does not necessarily re-linearize at each time step, however. Often the same linear model is retained for a number of time steps before re-linearizing, even if the plant is being transferred from one state to another.

¹ Note that this is different from dX_0/dt , which is zero. Non-standard notation is being used here; also our use of \dot{x} is non-standard.

For predictive control we need the difference equation model (2.1), since we use a discrete-time setting. This can be obtained from the linearized differential equation by standard techniques, usually assuming that the input u is constant between sampling intervals [FPEN94]. Section 2.5 considers what to do if the computational delay is too large to allow the standard conversion formulae to be used.

Conceptually, it is also possible to arrive at the linear, discrete-time model (2.1) by linearizing an assumed discrete-time nonlinear model of the general form

$$x(k+1) = \phi(x(k), u(k), k) \quad (2.14)$$

But the function ϕ is usually not one that can be written down as a set of equations; it is the function implemented by solving the continuous-time differential equations between sampling intervals, assuming that the inputs are constant. In other words, it is necessary to run a simulation in order to discover what the value of the function ϕ is for a particular triple $(x(k), u(k), k)$.

The outputs of a real plant are also determined from the state in a nonlinear way:

$$Y = g(X, t) \quad (2.15)$$

where g is some nonlinear function, and we have assumed no explicit dependence on the input U . Proceeding as before, suppose that $Y_0 = g(X_0, t)$ and $Y = Y_0 + y$. Then

$$Y = g(X_0 + x, t) \quad (2.16)$$

$$\approx g(X_0, t) + \left. \frac{\partial g}{\partial X} \right|_{(X_0, t)} x \quad (2.17)$$

$$= Y_0 + C_y x \quad (2.18)$$

which leads to (2.2). The linearized equation (2.3) for the controlled outputs z can obviously be obtained in the same way. Since we assume that the output equations are static equations, their linearizations are the same in both the continuous-time and the discrete-time models. When obtaining measurements from the plant, the ‘baseline’ values Y_0 must be subtracted before using them in the linearized model. When making predictions, and when expressing constraints and objectives, care must be taken to either re-insert Y_0 at the appropriate points, or to express everything relative to Y_0 . Conceptually this is very straightforward, but it requires care and a systematic approach to get it right on a complex plant.

Similar considerations apply when applying the control signal to the plant. The predictive control calculations usually give a value of u , and the ‘baseline’ value U_0 must be added to this before applying it to the plant. However, as we saw in the previous chapter, in predictive control we most often compute the required changes in the control signal from one time step to the next, $\Delta u(k) = u(k) - u(k-1)$. Since $\Delta u = \Delta U$, the required change can be applied directly. But the correct expression of constraints again requires attention to consideration of U_0 , as does the expression of objectives in those cases in which the input values, as well as their changes, are of concern.

2.1.3 First-principles models and system identification

The linearized model (2.1)–(2.3) used in predictive control can be obtained in two ways, essentially. Most commonly, it is obtained by performing tests on the plant, which involve injecting known signals, such as steps, multi-sines, pseudo-random, or others, at the plant inputs, and recording the resulting plant outputs. Linearized models can then be obtained by using the techniques of *system identification*, which range from simple curve-fitting to sophisticated statistically based methods; there is a large literature on this, and we refer the reader to [Lju89, Nor86, vOdM96] as good starting points. It is sometimes claimed that predictive control has special requirements with regard to system identification. This is probably inaccurate, however; but it is certainly true that most applications of predictive control are in the process industries, and these industries have special requirements, largely because of the multivariable cross-couplings between inputs and outputs, and because of the slow dynamics that are typical. Some material on system identification specifically for the process industries can be found in [Ric91, RPG92, SMS92, ZB93, Zhu98].

Models obtained in this way are ‘black box’ models, which represent only the input–output behaviour of the plant, and carry no information about its internal structure. In some cases special-purpose tests may be unnecessary or impossible, but it may be possible to identify a linearized model on the basis of normal operating data; in particular, this is the situation in adaptive control [ÅW89, SR96a, Mos95].

It should be emphasized that the form of linear model that is adopted does not constrain the kind of test that should be applied to the plant. There appears to be quite a widespread misconception that step inputs are required in order to identify a model in step-response form, or that pseudo-random inputs are needed to obtain a transfer-function model form, and so on. This is not true; if one is lucky enough to have some choice of the input signal to use, that choice should depend primarily on the experimental conditions — how much testing is possible, how noisy are the measurements, etc. — and on the inherent requirements of the model — at which operating point should it be valid, what range of signal amplitudes are expected, over what range of frequencies should it be accurate, etc. Once a linear model is obtained in any of the standard forms, it is very straightforward to transform it to any other form. This point will be emphasized again in Chapter 4, and some specific transformation procedures will be discussed there.

Sometimes a *first-principles* nonlinear model of the plant is available. That is, a model in which the equations are obtained from a knowledge of the underlying physical, chemical and thermodynamic processes. For flight control, for example, the equations which accurately describe the nonlinear dynamics of an aircraft are well known, and not very complicated. For complex industrial processes such first-principles dynamic models are much more complex and expensive to develop, but this is being done increasingly commonly. (Typically they are developed for the purposes of operator training or safety certification, but once they have been developed, there is no reason why they should not be used for the purposes of control.) The availability of such a model corresponds to having (2.7) available, though one should be aware that the innocuous-looking notation $f(X, dX/dt, U, t) = 0$ can represent a very large collection of differential-algebraic equations, containing non-smooth elements such as switches (*case* statements) and look-up tables.

Linearized models can be obtained from a first-principles nonlinear model. In relatively simple cases, such as aircraft, this can be done largely ‘by hand’, but in complex cases it has to be done automatically. The standard way of doing this is by applying perturbations to the nonlinear model, and estimating the Jacobian matrices (A_c and B_c) numerically. Another very effective procedure is to use the nonlinear model to generate simulation data for particular conditions, and then to apply system identification techniques to this data, as if it had been obtained from the plant itself — see Chapter 4 for an example.

2.2 A basic formulation

For the basic formulation of predictive control we shall assume that the plant model is linear, that the cost function is quadratic (see Mini-Tutorial 1), and that constraints are in the form of *linear inequalities*. We shall also assume that everything is time-invariant. Furthermore, we shall assume that the cost function does not penalize particular values of the input vector $u(k)$, but only changes of the input vector, $\Delta u(k)$, which are defined as before. This formulation coincides with that used in the majority of the predictive control literature.

To make the formulation useful in the real world, we shall not assume that the state variables can be measured, but that we can obtain an estimate $\hat{x}(k|k)$ of the state $x(k)$, the notation indicating that this estimate is based on measurements up to time k — that is, on measurements of the outputs up to $y(k)$, and on knowledge of the inputs only up to $u(k-1)$, since the next input $u(k)$ has not yet been determined. $\hat{u}(k+i|k)$ will denote a future value (at time $k+i$) of the input u , which is assumed at time k . $\hat{x}(k+i|k)$, $\hat{y}(k+i|k)$ and $\hat{z}(k+i|k)$ will denote the predictions, made at time k , of the variables x , y and z at time $k+i$, on the assumption that some sequence of inputs $\hat{u}(k+j|k)$ ($j = 0, 1, \dots, i-1$) will have occurred. These predictions will be made consistently with the assumed linearized model (2.1)–(2.3). We will usually assume that the real plant is governed by the same equations as the model, although this is not really true. Only in Chapter 8 will we explicitly consider the fact that the model is inevitably inaccurate.

Cost function

The cost function V penalizes deviations of the predicted controlled outputs $\hat{z}(k+i|k)$ from a (vector) reference trajectory $r(k+i|k)$. Again the notation indicates that this reference trajectory may depend on measurements made up to time k ; in particular, its initial point may be the output measurement $y(k)$, as in Chapter 1. But it may also be a fixed set-point, or some other predetermined trajectory. We define the cost function to be

$$V(k) = \sum_{i=H_w}^{H_p} \|\hat{z}(k+i|k) - r(k+i|k)\|_{Q(i)}^2 + \sum_{i=0}^{H_u-1} \|\Delta \hat{u}(k+i|k)\|_{R(i)}^2 \quad (2.19)$$

There are several points to note here. The prediction horizon has length H_p , but we do not necessarily start penalizing deviations of z from r immediately (if $H_w > 1$),

Expressions like $x^T Qx$ and $u^T Ru$, where x, u are vectors and Q, R are symmetric matrices, are called *quadratic forms*, and are often written as $\|x\|_Q^2$ and $\|u\|_R^2$, respectively. They are just compact representations of certain quadratic functions in several variables. This is best illustrated by an example.

Example 2.1

Suppose that

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

and

$$Q = \begin{bmatrix} 10 & 4 \\ 4 & 3 \end{bmatrix}.$$

Then

$$\begin{aligned} x^T Qx &= [x_1 \quad x_2] \begin{bmatrix} 10 & 4 \\ 4 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \\ &= 10x_1^2 + 8x_1x_2 + 3x_2^2. \end{aligned}$$

If $x^T Qx > 0$ for every x except $x = 0$, then this quadratic form is called *positive definite*. It can be shown that a quadratic form is positive definite if and only if all the eigenvalues of the matrix involved in it are positive, in which case we write $Q > 0$. If $x^T Qx \geq 0$ then the quadratic form is called *semi-positive definite* and we write $Q \geq 0$.

The matrix Q in the example has both eigenvalues positive. So the quadratic form is positive-definite. You can try substituting any real values for x_1 and x_2 (even negative ones), and you will see that the value is always positive. Note that this happens even if $x_1 = 0$ or $x_2 = 0$, provided that they are not both zero.

If $Q = I$ then $x^T Qx = x^T x = \|x\|^2$, the squared ‘length’ of the vector, or squared *Euclidean norm*. Even if $Q \neq I$ (but $Q \geq 0$), it is possible to think of a quadratic form as the square of a ‘weighted norm’, since $x^T Qx = \|Q^{1/2}x\|^2$. Hence the notation $\|x\|_Q^2$.

The only other thing we will need to know about a quadratic form is how to find its gradient. Let $V = x^T Qx$, but now consider the general case where x has n elements, $x = [x_1, x_2, \dots, x_n]^T$, and Q is an $n \times n$ matrix. Then the gradient of V is given by:

$$\begin{aligned} \nabla V &= \left[\frac{\partial V}{\partial x_1}, \frac{\partial V}{\partial x_2}, \dots, \frac{\partial V}{\partial x_n} \right] \\ &= 2x^T Q \end{aligned}$$

(We are not going to prove this here.) Note that we have defined the gradient to be a row vector. Sometimes it is defined to be a column vector, in which case it is given by $2Qx$.

Mini-Tutorial 1 Quadratic forms.

because there may be some delay between applying an input and seeing any effect. H_u is the *control horizon*. We will always assume that $H_u \leq H_p$, and that $\Delta\hat{u}(k+i|k) = 0$ for $i \geq H_u$, so that $\hat{u}(k+i|k) = \hat{u}(k+H_u-1|k)$ for all $i \geq H_u$. (Recall Figure 1.4.)

The form of the cost function (2.19) implies that the error vector $\hat{z}(k+i|k) - r(k+i|k)$ is penalized at every point in the prediction horizon, in the range $H_w \leq i \leq H_p$. This

is indeed the most common situation in predictive control. But it is possible to penalize the error at only a few coincidence points, as was done in Chapter 1, by setting $Q(i) = 0$ for most values of i . It is also possible to have different coincidence points for different components of the error vector by setting appropriate elements of the weighting matrices $Q(i)$ to 0. To allow for these possibilities, we do not insist on $Q(i) > 0$, but allow the weaker condition $Q(i) \geq 0$.² (At least this condition is required, to ensure that $V(k) \geq 0$.)

Example 2.2

Suppose that there are two controlled outputs. The prediction horizon is $H_p = 3$. There is only one coincidence point for the first output, at $i = 2$. For the second output there are two coincidence points, at $i = 2$ and $i = 3$. Errors in the second output are penalized more heavily than errors in the first output. This could be represented by

$$Q(1) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \quad Q(2) = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \quad Q(3) = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix} \quad (2.20)$$

with $H_w = 1$. It could also be represented by taking $H_w = 2$, in which case $Q(1)$ would not be defined, while $Q(2)$ and $Q(3)$ would remain as above. Most commonly, the weighting matrices $Q(i)$ and $R(i)$ are diagonal, as in this example.

We also need $R(i) \geq 0$ to ensure that $V(k) \geq 0$. Again, we do not insist on the stronger condition that $R(i) > 0$, because there are cases in which the changes in the control signal are not penalized — for instance, all the examples considered in Chapter 1. The weighting matrices $R(i)$ are sometimes called *move suppression factors*, since increasing them penalizes changes in the input vector more heavily.

The cost function (2.19) only penalizes changes in the input vector, not its value. In some cases an additional term of the form $\sum \|\hat{u}(k+i|k) - u_0\|_S^2$ is added, which penalizes deviations of the input vector from some *ideal resting value*. Usually this is done only if there are more inputs than variables which are to be controlled to set-points. We shall not include such a term in our basic formulation, but it will be discussed in more detail in Section 10.1.

The prediction and control horizons H_p and H_u , the ‘window’ parameter H_w , the weights $Q(i)$ and $R(i)$, and the reference trajectory $r(k+i)$, all affect the behaviour of the closed-loop combination of plant and predictive controller. Some of these parameters, particularly the weights, may be dictated by the economic objectives of the control system, but usually they are in effect *tuning parameters* which are adjusted to give satisfactory dynamic performance. Chapters 6, 7 and 8 all discuss the influence of these parameters (and others).

Constraints

In the following we use $\text{vec}(0)$ to denote a column vector, each element of which is 0. We also use $\text{vec}(a, b, c)$ to denote the vector $[a, b, c]^T$, etc. We assume that constraints

² The notation $Q \geq 0$ usually excludes the extreme possibility that $Q = 0$. But we use it here to include this possibility.

of the following form hold over the control and prediction horizons:

$$E \operatorname{vec}(\Delta\hat{u}(k|k), \dots, \Delta\hat{u}(k+H_u-1|k), 1) \leq \operatorname{vec}(0) \quad (2.21)$$

$$F \operatorname{vec}(\hat{u}(k|k), \dots, \hat{u}(k+H_u-1|k), 1) \leq \operatorname{vec}(0) \quad (2.22)$$

$$G \operatorname{vec}(\hat{z}(k+H_w|k), \dots, \hat{z}(k+H_p|k), 1) \leq \operatorname{vec}(0) \quad (2.23)$$

Here E , F and G are matrices of suitable dimensions. We can use constraints of this form to represent possible actuator slew rates (2.21), actuator ranges (2.22), and constraints on the controlled variables (2.23).

Example 2.3

Suppose we have a plant with 2 input variables and 2 controlled variables. The control horizon is $H_u = 1$ and the prediction horizon is $H_p = 2$ (with $H_w = 1$). We have the following constraints, which are to hold at each time:

$$-2 \leq \Delta u_1 \leq 2 \quad (2.24)$$

$$0 \leq u_2 \leq 3 \quad (2.25)$$

$$z_1 \geq 0 \quad (2.26)$$

$$z_2 \geq 0 \quad (2.27)$$

$$3z_1 + 5z_2 \leq 15 \quad (2.28)$$

To put these into our standard form, we rewrite the Δu_1 constraints as follows.

$$-2 \leq \Delta u_1 \Leftrightarrow -\Delta u_1 \leq 2 \Leftrightarrow -\frac{1}{2}\Delta u_1 - 1 \leq 0 \Leftrightarrow [-1/2, 0, -1] \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ 1 \end{bmatrix} \leq 0$$

$$\Delta u_1 \leq 2 \Leftrightarrow \frac{1}{2}\Delta u_1 - 1 \leq 0 \Leftrightarrow [1/2, 0, -1] \begin{bmatrix} \Delta u_1 \\ \Delta u_2 \\ 1 \end{bmatrix} \leq 0$$

Now we write these two inequalities together, and replace Δu by $\Delta\hat{u}$:

$$\begin{bmatrix} -1/2 & 0 & -1 \\ 1/2 & 0 & -1 \end{bmatrix} \begin{bmatrix} \Delta\hat{u}_1(k|k) \\ \Delta\hat{u}_2(k|k) \\ 1 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

so that we have

$$E = \begin{bmatrix} -1/2 & 0 & -1 \\ 1/2 & 0 & -1 \end{bmatrix}$$

We leave finding the matrix F for this example to the reader (Exercise 2.4), and go on to find G . We proceed in the same way, the main difference being that we have to

express the inequalities across the prediction horizon, which is greater than 1.

$$z_1 \geq 0 \Leftrightarrow -z_1 \leq 0$$

$$z_2 \geq 0 \Leftrightarrow -z_2 \leq 0$$

$$3z_1 + 5z_2 \leq 15 \Leftrightarrow \frac{1}{5}z_1 + \frac{1}{3}z_2 - 1 \leq 0 \Leftrightarrow [1/5, 1/3, -1] \begin{bmatrix} z_1 \\ z_2 \\ 1 \end{bmatrix} \leq 0$$

from which we get

$$\underbrace{\begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 1/5 & 1/3 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1/5 & 1/3 & -1 \end{bmatrix}}_G \begin{bmatrix} \hat{z}_1(k+1|k) \\ \hat{z}_2(k+1|k) \\ \hat{z}_1(k+2|k) \\ \hat{z}_2(k+2|k) \\ 1 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

It will be seen that the dimensions of the matrices E , F and G can become very large very quickly (H_p may be 10 or 20 or more). However, assembling them from the given constraints is simple, and this job can be automated. In fact, the software which we will use — the *Model Predictive Control Toolbox* for MATLAB — does this for the user.

The important thing about the constraints is that they are all in the form of *linear inequalities*. When we come to solve the predictive control optimization problem, all these inequalities will need to be translated into inequalities concerning $\Delta\hat{u}(k+i|k)$. Since we assume a linear model, we will see that these inequalities remain linear, even after this translation (Exercise 2.5). This is going to be crucial for being able to solve the optimization problem reliably and efficiently.

Notice that in the example there are variables $(u_1, \Delta u_2)$ which are not constrained. It is also possible to have the converse situation, of variables which are constrained, but do not appear in the cost function (*zone objectives*). This is only likely to occur with the z variables, and is represented in our standard formulation by having appropriate zero entries in the weighting matrices $Q(i)$.

Example 2.4

Paper machine headbox

This example is based on an example in the *Model Predictive Control Toolbox* User's Guide [MR95], and some of the software-based problems will refer to it.

Part of a paper-making machine is shown in Figure 2.1. The following variables are involved:

Inputs (u)	States (x)	Measured outputs (y)	Controlled variables (z)
Stock flowrate G_P	Feed tank level H_1		
WW flowrate G_w	Headbox level H_2	H_2	H_2
	Feed tank consistency N_1	N_1	
	Headbox consistency N_2	N_2	N_2

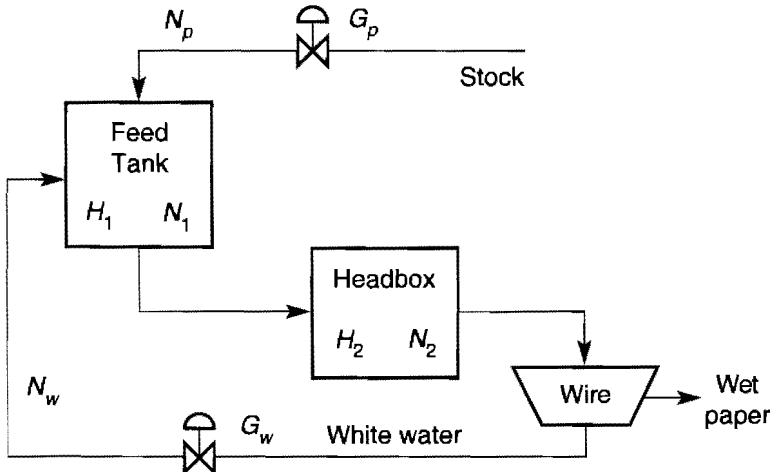


Figure 2.1 Paper machine with headbox.

A linearized model of this machine has the (continuous-time) state-space matrices

$$A_c = \begin{bmatrix} -1.93 & 0 & 0 & 0 \\ 0.394 & -0.426 & 0 & 0 \\ 0 & 0 & -0.63 & 0 \\ 0.82 & -0.784 & 0.413 & -0.426 \end{bmatrix} \quad B_c = \begin{bmatrix} 1.274 & 1.274 \\ 0 & 0 \\ 1.34 & -0.65 \\ 0 & 0 \end{bmatrix}$$

$$C_y = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad C_z = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where time has been measured in minutes. Predictive control is to be applied to this plant, with an update interval of $T_s = 2$ minutes. Discretizing the model with this sample interval (using the MATLAB function `c2d` for example) gives

$$A = \begin{bmatrix} 0.0211 & 0 & 0 & 0 \\ 0.1062 & 0.4266 & 0 & 0 \\ 0 & 0 & 0.2837 & 0 \\ 0.1012 & -0.6688 & 0.2893 & 0.4266 \end{bmatrix} \quad B = \begin{bmatrix} 0.6462 & 0.6462 \\ 0.2800 & 0.2800 \\ 1.5237 & -0.7391 \\ 0.9929 & 0.1507 \end{bmatrix}$$

Note that the matrices C_y and C_z remain the same in the discrete-time model.

The prediction horizon is to be $H_p = 10$ (that is, 20 minutes) and the control horizon is to be $H_u = 3$. Since control of consistency is more important than control of level, errors in consistency (N_2) are penalized more than errors in level (H_2):

$$Q(i) = \begin{bmatrix} 0.2 & 0 \\ 0 & 1 \end{bmatrix}$$

Changes of the two valves are penalized equally, and the same penalties are applied at each step of the prediction and control horizons (that is, for each i):

$$R(i) = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.6 \end{bmatrix}$$

Since there is no delay in the plant (beyond the usual 1-step delay — $u(k)$ does not affect $x(k)$), errors are penalized from the first step, namely $H_w = 1$.

The valves are scaled so that their range of operation is:

$$-10 \leq u_1(k) \leq 10 \quad -10 \leq u_2(k) \leq 10$$

(0 is usually the point at which the plant was linearized, so that the range is not always symmetric about 0), and their slew rate (maximum rate of change) is such that they can move through 10% of the range in one step (2 minutes):

$$-2 \leq \Delta u_1(k) \leq 2 \quad -2 \leq \Delta u_2(k) \leq 2$$

The headbox level (H_2) is constrained, so that the headbox never runs dry or overfills:

$$-3 \leq z_1 \leq 5$$

Note that if an operating point is represented by $u = 0$, then with a linear model we must also have $y = 0$ at that point (unless there is integration in the plant).

From these constraints we can assemble the matrices E , F and G which appear in our standard formulation of the predictive control problem. These matrices will have the following dimensions:

$$E : 4 \times 7 \quad F : 4 \times 7 \quad G : 20 \times 21$$

2.3

General features of constrained predictive control

Since predictive control problems usually include constraints, the resulting control law is usually *nonlinear*. It is easy to understand why this is so. Suppose we have a gas-turbine engine with a ‘hard’ state constraint of the form $x_j(t) < X_j$ — such as ‘Turbine temperature lower than 1000 °C’. If the turbine is running at 995 °C and a disturbance d occurs which has the effect of moving x_j away from X_j — the cooling oil flow rate increases for some reason, say — then the control system will react in a fairly ‘relaxed’ way, and will coordinate bringing x_j back to its set-point with maintaining other objectives — correct turbine speed, power output, fuel economy, etc. Now if a disturbance $-d$ occurs instead, then by definition a linear controller would react in a similarly relaxed way, replacing a control signal $u(t)$ by $-u(t)$.

In contrast, a controller which is aware of the constraint will react in ‘panic mode’. It ‘knows’ that there is no point in worrying about other objectives, because if the constraint is exceeded catastrophic failure is likely to result. It therefore reacts in a very different way to the disturbance $-d$ than it would to disturbance $+d$ — in our example,

perhaps it will cut the fuel flow rate sharply and change the inlet vane angle to get maximum air flow. In a multi-engine aircraft it might even shut the engine down. Such behaviour is of course nonlinear.

We will see later that ‘standard’ predictive controllers in fact behave linearly so long as the plant is operating safely away from constraints, but nonlinearly when constraints are approached too closely. On the other hand, a standard predictive controller would not go so far as to shut down an engine, and it does not have the authority to do so. That kind of decision is usually handled by a separate higher-level ‘supervisor’, which may be a human operator or pilot.

Although constrained predictive controllers are nonlinear, they are usually *time-invariant*. This means that there is a function h , such that the control signal can be expressed as $u = h(x)$, namely it depends only on the current state, and not explicitly on time (that is, it is not necessary to write $u = h(x, t)$). Of course this function ‘exists’ only in a mathematical sense; it would be impossibly difficult to write it down in ‘closed-form’ (as a formula). In order to see what u is, you have to feed x into an optimization algorithm and see what comes out. This time-invariance occurs so long as there is nothing in the problem formulation that depends explicitly on time: the model of the plant, the cost function and the constraints must all be independent of time.

We have to be rather careful about what this means exactly. First of all, our model (2.1)–(2.3) does not depend on the time k , except through x and u . Secondly, the cost function (2.19) *can* depend on i — the weights $Q(i)$ and $R(i)$ can vary over the prediction and control horizons — but *not* on k . So a different weight can be attached to an error predicted to be 3 steps in the future than to an error predicted 1 step ahead. But when the problem is solved at the next k value, it must have the same pattern of weights. As time moves on, so k increases to $k + 1$, but exactly the same problem is solved as was solved at time k . So the cost function does not depend on time.

Thirdly, to get a time-invariant controller we can have constraints depending on i but not on k . So $|\hat{x}_j(k+i|k)| < X_j(i)$ is OK, but $|\hat{x}_j(k+i|k)| < X_j(k)$ is not. More generally,

$$\hat{x}(k+i|k) \in X(i) \quad (2.29)$$

$$\hat{u}(k+i|k) \in U(i) \quad (2.30)$$

is OK, whereas

$$\hat{x}(k+i|k) \in X(k) \quad (2.31)$$

$$\hat{u}(k+i|k) \in U(k) \quad (2.32)$$

is not. This can be a restriction if we want a time-invariant controller. For instance, when landing an aircraft, one may want constraints to become tighter as the aircraft approaches the runway. If one assumes a certain speed profile, the constraint tightening can be ‘scheduled’ as a function of time. This would give a time-varying predictive control law. However, it may make more sense to schedule the constraint tightening as a function of distance from the runway, and make this one of the states. In this case the control law will be time-invariant (though not necessarily simpler!).

There is no necessary advantage to having a time-invariant control law. Having one may make analysis of its behaviour easier in some cases, but in practice changes to the model, the cost and the constraints are all made from time to time.

2.4 Alternative state variable choices

As usual, our plant model (2.1) expresses the plant state x in terms of the values of the input u . But the cost function penalizes changes in the input, Δu , rather than the input values themselves. We shall see in the next chapter that the predictive control algorithm will in fact produce the changes Δu rather than u . It is therefore convenient for many purposes to regard the ‘controller’ as producing the signal Δu , and the ‘plant’ as having this signal as its input. That is, it is often convenient to regard the ‘discrete-time integration’ from Δu to u as being included in the plant dynamics. Figure 2.2 shows the real controller producing the signal u , which is passed to the real plant. It also shows the MPC ‘controller’ producing the signal Δu , which is passed to the MPC ‘plant’.

There are several ways of including this ‘integration’ in a state-space model of the MPC ‘plant’. All of them involve augmenting the state vector. The first way is to define the state vector³

$$\xi(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix}. \quad (2.33)$$

Then, assuming the linear model (2.1) holds for the real plant state, we have

$$\begin{bmatrix} x(k+1) \\ u(k) \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} + \begin{bmatrix} B \\ I \end{bmatrix} \Delta u(k) \quad (2.34)$$

$$y(k) = [C_y, 0] \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \quad (2.35)$$

$$z(k) = [C_z, 0] \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \quad (2.36)$$

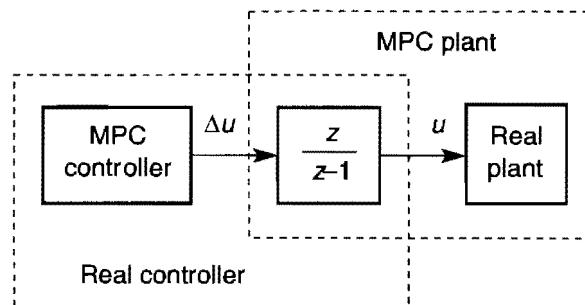


Figure 2.2 The boundary between controller and plant — a matter of convenience.

³ We shall often use ξ as an augmented state, with associated matrices \tilde{A} and \tilde{B} , etc. The meaning will be different, depending on the context, but will always be defined ‘locally’.

A less obvious way is to define the state vector

$$\xi(k) = \begin{bmatrix} \Delta x(k) \\ y(k) \\ z(k) \end{bmatrix} \quad (2.37)$$

where $\Delta x(k) = x(k) - x(k-1)$ is the change in the state at time k . From (2.1) we have

$$x(k+1) = Ax(k) + Bu(k) \quad (2.38)$$

$$x(k) = Ax(k-1) + Bu(k-1) \quad (2.39)$$

Subtracting these gives

$$\Delta x(k+1) = A\Delta x(k) + B\Delta u(k) \quad (2.40)$$

Furthermore, from (2.2) and (2.3) we have

$$\begin{aligned} y(k+1) &= C_y x(k+1) \\ &= C_y [\Delta x(k+1) + x(k)] \\ &= C_y [A\Delta x(k) + B\Delta u(k)] + y(k) \end{aligned} \quad (2.41)$$

$$\begin{aligned} z(k+1) &= C_z x(k+1) \\ &= C_z [\Delta x(k+1) + x(k)] \\ &= C_z [A\Delta x(k) + B\Delta u(k)] + z(k) \end{aligned} \quad (2.42)$$

so that we have the state-space representation

$$\begin{bmatrix} \Delta x(k+1) \\ y(k+1) \\ z(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ C_y A & I & 0 \\ C_z A & 0 & I \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ y(k) \\ z(k) \end{bmatrix} + \begin{bmatrix} B \\ C_y B \\ C_z B \end{bmatrix} \Delta u(k) \quad (2.43)$$

$$\begin{bmatrix} y(k) \\ z(k) \end{bmatrix} = \begin{bmatrix} 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ y(k) \\ z(k) \end{bmatrix} \quad (2.44)$$

Both of these state-space representations have been used in the literature of predictive control. For example, the first representation (2.33) was used in [Ric90], while the second one (2.37) was used in [PG88] and many other publications. The second representation is also used in the *Model Predictive Control Toolbox* for some purposes. These two seem to be the most useful, although other representations are possible — see Exercise 2.8. Note that if the number of inputs is not the same as the number of measured and controlled outputs, then at least one of these representations is not minimal — that is, it must be either uncontrollable or unobservable — since the two representations will then have different state dimensions, but both give the same input–output transfer function.

2.5 Allowing for computational delay

Since predictive control involves on-line optimization, a considerable computational delay may be involved, and this should be taken account of. Figure 2.3 shows the

assumptions we shall make about the timing of measurements made on the plant being controlled, and the resulting control signals being applied.

The measurement interval and control update interval are assumed to be the same, with length T_s . The plant output vector is measured at time kT_s , and this measurement is labelled $y(k)$. If there is a measured disturbance this is assumed to be measured at the same time, and this is labelled $d_m(k)$. There is then a delay of τ , which is the time taken by the predictive controller to complete its computations, after which a new control vector is produced and applied as the plant input signal. This input signal is labelled $u(k)$. The input signal is assumed to be held constant until it is recomputed T_s time units later. This sequence is repeated at time $(k + 1)T_s$, and regularly thereafter.

In practice, process plants may have hundreds of measurements which may be taken, and/or made available, at various times during the measurement interval. If accurate modelling is required, this may have to be taken into account. Also, the computation delay τ may vary in practice, in which case the decision must be taken whether to apply the new control signal to the plant as soon as it becomes available — which probably improves the control but complicates the modelling and analysis — or whether the result should be held up until a standard interval has elapsed before applying it to the

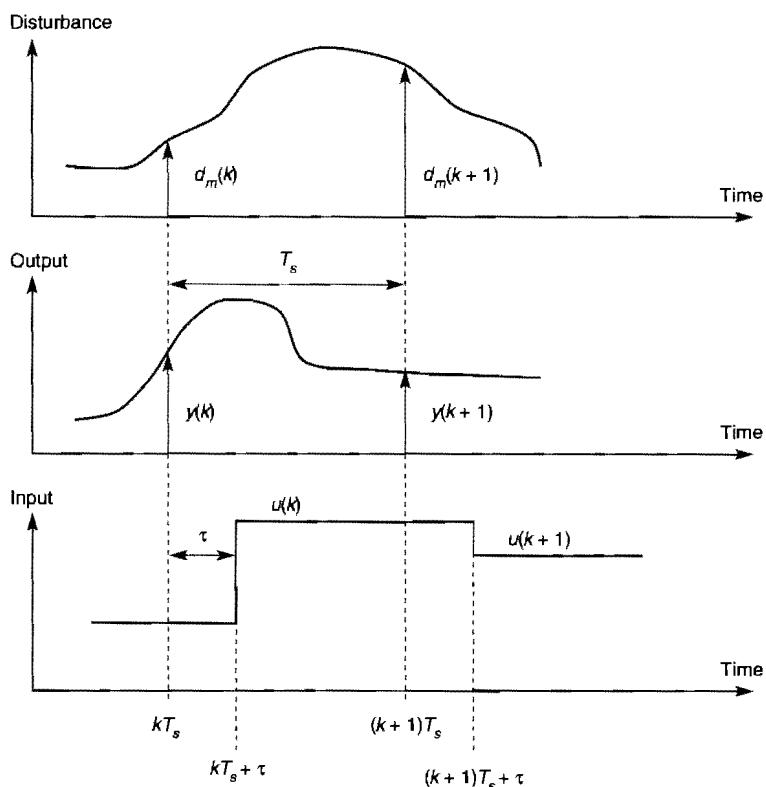


Figure 2.3 Assumed timing of measuring and applying signals.

plant. It would be impossible to deal with all such eventualities here, so we will assume that all the measurements are taken synchronously, as shown in Figure 2.3, and that the computational delay τ is the same at each step.

Now we suppose that we have a linearized model for the continuous-time plant in the state-space form:

$$\dot{x}_c(t) = A_c x_c(t) + B_c u_c(t) \quad (2.45)$$

$$y_c(t) = C_c x_c(t) \quad (2.46)$$

(It is enough to consider the measured outputs y . Controlled outputs z are treated in exactly the same way.) The assumptions shown in Figure 2.3 imply that $y(k) = y_c(kT_s)$, and that

$$u_c(t) = u(k) \quad \text{for } kT_s + \tau \leq t < (k+1)T_s + \tau \quad (2.47)$$

so that we have

$$\dot{x}_c(t) = \begin{cases} A_c x_c(t) + B_c u(k-1) & \text{for } kT_s \leq t < kT_s + \tau \\ A_c x_c(t) + B_c u(k) & \text{for } kT_s + \tau \leq t < (k+1)T_s \end{cases} \quad (2.48)$$

Now the solution of (2.45) is well known [FPEN94, Kai80] to be given by

$$x_c(t) = e^{A_c(t-t_0)} x_c(t_0) + \int_{t_0}^t e^{A_c(t-\theta)} B u_c(\theta) d\theta \quad (2.49)$$

where $t_0 < t$ is some initial time, and $x_c(t_0)$ is the initial state at that time. So, applying this solution over the interval $kT_s \leq t < kT_s + \tau$, and defining $x(k) = x_c(kT_s)$, gives

$$x_c(kT_s + \tau) = e^{A_c \tau} x(k) + \left(\int_{kT_s}^{kT_s + \tau} e^{A_c(kT_s + \tau - \theta)} d\theta \right) B_c u(k-1) \quad (2.50)$$

Now using the change of variable $\eta = kT_s + \tau - \theta$ it is easy to show that

$$\int_{kT_s}^{kT_s + \tau} e^{A_c(kT_s + \tau - \theta)} d\theta = \int_0^\tau e^{A_c \eta} d\eta = \Gamma_1 \quad (2.51)$$

which is a constant matrix, so we have

$$x_c(kT_s + \tau) = e^{A_c \tau} x(k) + \Gamma_1 B_c u(k-1) \quad (2.52)$$

Similarly, applying (2.49) over the interval $kT_s + \tau \leq t < (k+1)T_s$ gives

$$x(k+1) = e^{A_c(T_s - \tau)} x_c(kT_s + \tau) + \left(\int_{kT_s + \tau}^{(k+1)T_s} e^{A_c((k+1)T_s - \theta)} d\theta \right) B_c u(k) \quad (2.53)$$

$$\begin{aligned} &= e^{A_c(T_s - \tau)} [e^{A_c \tau} x(k) + \Gamma_1 B_c u(k-1)] \\ &\quad + \left(\int_0^{T_s - \tau} e^{A_c \eta} d\eta \right) B_c u(k) \end{aligned} \quad (2.54)$$

$$= Ax(k) + B_1 u(k-1) + B_2 u(k) \quad (2.55)$$

where

$$A = e^{A_c T_s} \quad B_1 = e^{A_c(T_s - \tau)} \Gamma_1 B_c \quad B_2 = \int_0^{T_s - \tau} e^{A_c \eta} d\eta B_c \quad (2.56)$$

Now this is not in the standard form of a discrete-time state-space model, because of the dependence of $x(k+1)$ on $u(k-1)$. But we can easily bring it into the standard form by introducing the state

$$\xi(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \quad (2.57)$$

which gives the state equation in standard form:

$$\xi(k+1) = \tilde{A}\xi(k) + \tilde{B}u(k) \quad (2.58)$$

where

$$\tilde{A} = \begin{bmatrix} A & B_1 \\ 0 & 0 \end{bmatrix} \quad \tilde{B} = \begin{bmatrix} B_2 \\ I \end{bmatrix} \quad (2.59)$$

From (2.46) the associated equation for the measured outputs is

$$y(k) = C_c x(k) = \tilde{C}\xi(k) \quad (2.60)$$

where $\tilde{C} = [C_c, 0]$.

Because of this possibility, we can continue to assume that the linearized plant model has the form (2.1)–(2.3), providing that an appropriate definition of the state vector is used. The reason for using this standard form is mainly that it connects well with the standard theory of linear systems and control. It is not necessarily the best form for implementation — for example it may be more efficient to use the form (2.55) in some circumstances.

The reader is warned that most standard software for finding discrete-time equivalents of continuous-time systems, such as the *Control System Toolbox* for *MATLAB* function `c2d`, does not take account of computation delay. It is, however, fairly easy to obtain the required model using such software — see Exercise 2.9.

Note that if the computational delay is negligible ($\tau = 0$) then $B_1 = 0$, and in the other extreme case, namely $\tau = T_s$, we have $B_2 = 0$. Most examples which appear in the predictive control literature neglect the computational delay.

2.6

Prediction

In order to solve the predictive control problem, we must have a way of computing the predicted values of the controlled variables, $\hat{z}(k+i|k)$, from our best estimate of the current state, $\hat{x}(k|k)$, and the assumed future inputs, or equivalently, the latest input, $u(k-1)$, and the assumed future input changes, $\Delta\hat{u}(k+i|k)$.

It may seem that this is really part of the solution, not of the problem formulation, so that it belongs in the next chapter. But it turns out that the way the predictions are

made has a great effect on the performance of the closed-loop system running under predictive control. So the choice of prediction strategy is another ‘tuning parameter’ for predictive control, just as choices of horizons and cost functions are. Furthermore, the prediction strategy follows in a rather systematic way from assumptions made about disturbances acting on the system and measurement errors such as noise. So we can say that, rather than choosing a prediction strategy, we are specifying a model of the environment in which the plant is operating. And that properly belongs here, as part of the problem formulation.

However, prediction can get very complicated. So to avoid being too distracted by it at this stage, we will deal with a few simple cases here — which will already cover much of industrial practice — and come back to more general cases later.

2.6.1 No disturbances, full state measurement

We will start with the simplest situation. Assume that the whole state vector is measured, so that $\hat{x}(k|k) = x(k) = y(k)$ (so $C_y = I$). Also assume that we know nothing about any disturbances or measurement noise. Then all we can do is to predict by iterating the model (2.1)–(2.3). So we get

$$\hat{x}(k+1|k) = Ax(k) + B\hat{u}(k|k) \quad (2.61)$$

$$\hat{x}(k+2|k) = A\hat{x}(k+1|k) + B\hat{u}(k+1|k) \quad (2.62)$$

$$= A^2x(k) + AB\hat{u}(k|k) + B\hat{u}(k+1|k) \quad (2.63)$$

⋮

$$\hat{x}(k+H_p|k) = A\hat{x}(k+H_p-1|k) + B\hat{u}(k+H_p-1|k) \quad (2.64)$$

$$= A^{H_p}x(k) + A^{H_p-1}B\hat{u}(k|k) + \dots + B\hat{u}(k+H_p-1|k) \quad (2.65)$$

In the first line we have used $\hat{u}(k|k)$ rather than $u(k)$, because at the time when we need to compute the predictions we do not yet know what $u(k)$ will be.

Now recall that we have assumed that the input will only change at times $k, k+1, \dots, k+H_u-1$, and will remain constant after that. So we have $\hat{u}(k+i|k) = \hat{u}(k+H_u-1)$ for $H_u \leq i \leq H_p-1$. In fact, we will later want to have the predictions expressed in terms of $\Delta\hat{u}(k+i|k)$ rather than $\hat{u}(k+i|k)$, so let us do that now. Recall that $\Delta\hat{u}(k+i|k) = \hat{u}(k+i|k) - \hat{u}(k+i-1|k)$, and that at time k we already know $u(k-1)$. So we have

$$\hat{u}(k|k) = \Delta\hat{u}(k|k) + u(k-1)$$

$$\hat{u}(k+1|k) = \Delta\hat{u}(k+1|k) + \Delta\hat{u}(k|k) + u(k-1)$$

⋮

$$\hat{u}(k+H_u-1|k) = \Delta\hat{u}(k+H_u-1|k) + \dots + \Delta\hat{u}(k|k) + u(k-1)$$

and hence we get

$$\begin{aligned}
 \hat{x}(k+1|k) &= Ax(k) + B[\Delta\hat{u}(k|k) + u(k-1)] \\
 \hat{x}(k+2|k) &= A^2x(k) + AB[\Delta\hat{u}(k|k) + u(k-1)] \\
 &\quad + B\underbrace{[\Delta\hat{u}(k+1|k) + \Delta\hat{u}(k|k) + u(k-1)]}_{\hat{u}(k+1|k)} \\
 &= A^2x(k) + (A+I)B\Delta\hat{u}(k|k) + B\Delta\hat{u}(k+1|k) + (A+I)Bu(k-1) \\
 &\quad \vdots \\
 \hat{x}(k+H_u|k) &= A^{H_u}x(k) + (A^{H_u-1} + \dots + A+I)B\Delta\hat{u}(k|k) \\
 &\quad + B\Delta\hat{u}(k+H_u-1|k) + (A^{H_u-1} + \dots + A+I)Bu(k-1)
 \end{aligned}$$

(Notice the change at this point)

$$\begin{aligned}
 \hat{x}(k+H_u+1|k) &= A^{H_u+1}x(k) + (A^{H_u} + \dots + A+I)B\Delta\hat{u}(k|k) \\
 &\quad + (A+I)B\Delta\hat{u}(k+H_u-1|k) \\
 &\quad + (A^{H_u} + \dots + A+I)Bu(k-1) \\
 &\quad \vdots \\
 \hat{x}(k+H_p|k) &= A^{H_p}x(k) + (A^{H_p-1} + \dots + A+I)B\Delta\hat{u}(k|k) \\
 &\quad + (A^{H_p-H_u} + \dots + A+I)B\Delta\hat{u}(k+H_u-1|k) \\
 &\quad + (A^{H_p-1} + \dots + A+I)Bu(k-1)
 \end{aligned}$$

Finally we can write this in matrix-vector form:

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \vdots \\ \hat{x}(k+H_u|k) \\ \vdots \\ \hat{x}(k+H_p|k) \end{bmatrix} = \underbrace{\begin{bmatrix} A \\ \vdots \\ A^{H_u} \\ A^{H_u+1} \\ \vdots \\ A^{H_p} \end{bmatrix}}_{\text{past}} x(k) + \underbrace{\begin{bmatrix} B \\ \vdots \\ \sum_{i=0}^{H_u-1} A^i B \\ \vdots \\ \sum_{i=0}^{H_p-1} A^i B \end{bmatrix}}_{\text{past}} u(k-1) + \underbrace{\begin{bmatrix} B & \dots & 0 \\ AB+B & \dots & 0 \\ \vdots & \ddots & \vdots \\ \sum_{i=0}^{H_u-1} A^i B & \dots & B \\ \sum_{i=0}^{H_u} A^i B & \dots & AB+B \\ \vdots & \vdots & \vdots \\ \sum_{i=0}^{H_p-1} A^i B & \dots & \sum_{i=0}^{H_p-H_u} A^i B \end{bmatrix}}_{\text{future}} \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k+H_u-1|k) \end{bmatrix} \quad (2.66)$$

The predictions of z are now obtained simply as

$$\hat{z}(k+1|k) = C_z \hat{x}(k+1|k) \quad (2.67)$$

$$\hat{z}(k+2|k) = C_z \hat{x}(k+2|k) \quad (2.68)$$

⋮

$$\hat{z}(k+H_p|k) = C_z \hat{x}(k+H_p|k) \quad (2.69)$$

or

$$\begin{bmatrix} \hat{z}(k+1|k) \\ \vdots \\ \hat{z}(k+H_p|k) \end{bmatrix} = \begin{bmatrix} C_z & 0 & \cdots & 0 \\ 0 & C_z & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & C_z \end{bmatrix} \begin{bmatrix} \hat{x}(k+1|k) \\ \vdots \\ \hat{x}(k+H_p|k) \end{bmatrix} \quad (2.70)$$

A warning is in order about the prediction equation (2.66). It involves computing A^i , possibly for quite large values of i . This can lead to numerical problems. If the plant is unstable, then some elements in A^i may become extremely large relative to others, and relative to elements in lower powers of A . Since computers work with finite-precision arithmetic, this can sometimes lead to wrong results. Similar problems can occur if the plant is stable; in this case some elements of A^i may become extremely small relative to others. Again wrong answers may result. (Using ‘IEEE Standard’ arithmetic, computers cannot distinguish between 1 and $1 + \epsilon$ if $|\epsilon| < 10^{-16}$, approximately.)

The safest way of computing the predictions is probably to iterate one step at a time. This is in effect what is done if the predictions are not computed explicitly when computing the optimal control signal, but if

$$\hat{x}(k+i|k) = A\hat{x}(k+i-1|k) + B\hat{u}(k+i-1|k) \quad (2.71)$$

is included as an equality constraint during the optimization, which also leads to the most efficient computation [RWR98] — see Section 3.2 for details. Alternatively, problems of predicting with an unstable plant can be alleviated by pre-stabilizing the plant, as described in Section 5.2. Even if explicit predictions are not made for the purposes of optimization, they may be needed for the operator interface. One of the problems associated with running an advanced controller, such as a predictive controller, is that plant operators may lose confidence in the correctness of the actions taken by the controller, if they do not understand what it is doing. Displaying predictions can contribute very effectively to retaining their confidence.

2.6.2 Constant output disturbance

Now we will assume that there are disturbances acting on the plant. The simplest assumption is that the measured and controlled outputs are the same ($z = y$), and that there is an ‘output disturbance’ — see Figure 2.4. At time k we do not know what the disturbance $d(k)$ is, but we can form an estimate of it, $\hat{d}(k|k)$, by comparing the measured output with the predicted one:

$$y(k) = \hat{y}(k|k-1) + \hat{d}(k|k) \quad (2.72)$$

$$= C_y \hat{x}(k|k-1) + \hat{d}(k|k) \quad (2.73)$$

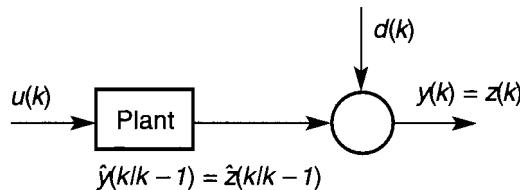


Figure 2.4 Output disturbance.

Another simple assumption is that this disturbance will continue unchanged during the prediction horizon. This means that, although we keep (2.2) and (2.3), we predict y and z using

$$\hat{z}(k+i|k) = \hat{y}(k+i|k) = C_y \hat{x}(k+i|k) + \hat{d}(k+i|k) \quad (2.74)$$

where

$$\hat{d}(k+i|k) = \hat{d}(k|k) \quad (2.75)$$

So at time step k we do the following:

1. Measure the actual plant output $y(k)$.
2. Estimate the disturbance as the difference between the actual and the estimated output.
3. Use that estimate to predict outputs over the prediction horizon.

We will generally follow this scheme, but Step 2 will become more elaborate when we make more elaborate assumptions about the nature of the disturbance. In Step 3 we need to assume future input movements, in order to use (2.1) or (2.66) to predict $\hat{x}(k+i|k)$.

The assumption of a constant output disturbance, and the simplest disturbance estimation scheme of (2.74)–(2.75) is sometimes called the ‘DMC scheme’, because that is what was used in one of the original proprietary predictive control products, *DMC* (= *Dynamic Matrix Control*) [CR80, PRC82]. The same scheme is used in most proprietary predictive control products — see Appendix A — and we have already used it in Chapter 1.

Notice that, even if $C_y = I$, we now have $\hat{y}(k+i|k) \neq \hat{x}(k+i|k)$. In general, this means that we need an *observer* to estimate \hat{x} .

2.6.3 Using an observer

If we cannot measure the full state vector, or if the measured outputs consist of some linear combinations of the states, so that the states cannot be measured directly, then an observer can be used to estimate the state vector. It is instructive to see how the ‘constant output disturbance’ assumption we made in the previous subsection can be handled using observer theory. We can do this by augmenting the model of the plant,

The general structure of a *state observer* is shown in Figure 2.5, for a plant described by the equations

$$x(k+1) = Ax(k) + Bu(k), \quad y(k) = Cx(k) \quad (2.76)$$

It is a copy of the plant, with feedback from the measured plant output, through the gain matrix L , to correct the state estimate \hat{x} .

The equations of the observer are:

$$\dot{\hat{x}}(k|k) = \hat{x}(k|k-1) + L[y(k) - \hat{y}(k|k-1)] \quad (2.77)$$

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) \quad (2.78)$$

$$\hat{y}(k|k-1) = C\hat{x}(k|k-1) \quad (2.79)$$

Substituting the third equation into the first, and eliminating $\dot{\hat{x}}(k|k)$, we get

$$\dot{\hat{x}}(k+1|k) = A(I - L'C)\hat{x}(k|k-1) + Bu(k) + AL'y(k) \quad (2.80)$$

$$= (A - LC)\hat{x}(k|k-1) + Bu(k) + Ly(k) \quad (2.81)$$

where $L = AL'$. This is a stable system if the eigenvalues of $A - LC$ lie inside the unit disk. Furthermore if we define the state estimation error as $e(k) = x(k) - \hat{x}(k|k-1)$, then using (2.76) we have

$$e(k+1) = (A - LC)e(k)$$

which shows that the state estimation error converges to zero if the observer is stable, at a rate determined by the eigenvalues of $A - LC$.

If the pair (A, C) is observable, then given an arbitrary set of locations in the complex plane, a gain matrix L exists which places the observer eigenvalues at these locations. The problem of finding L is the dual of the state-feedback pole-placement problem, and can be solved using the same algorithm. (See functions `place`, `acker`, or `kalman` in MATLAB's *Control System Toolbox*.)

If the state and output equations of the plant are assumed to be subjected to white noise disturbances with known covariance matrices, then L can be chosen such that the mean square state estimation error is the smallest possible. The observer is then known as a *Kalman Filter*.

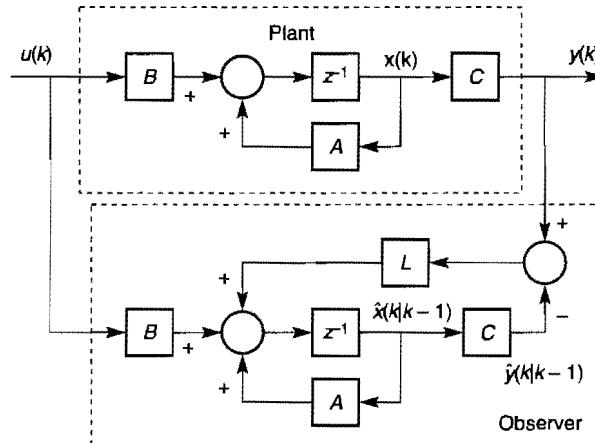


Figure 2.5 A state observer.

so that it also includes a model of the output disturbance. We do this by defining a new augmented state:

$$\xi(k) = \begin{bmatrix} x(k) \\ d(k) \end{bmatrix}$$

Since we assume that the output disturbance is constant, the new state and output equations are

$$\begin{bmatrix} x(k+1) \\ d(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} x(k) \\ d(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) \quad (2.82)$$

$$y(k) = [C_y \quad I] \begin{bmatrix} x(k) \\ d(k) \end{bmatrix} \quad (2.83)$$

Now if we partition the observer gain matrix L :

$$L = \begin{bmatrix} L_x \\ L_d \end{bmatrix}$$

then applying the standard observer equation (see Mini-Tutorial 2) gives the following estimates:

$$\begin{bmatrix} \hat{x}(k+1|k) \\ \hat{d}(k+1|k) \end{bmatrix} = \left(\begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} - \begin{bmatrix} L_x \\ L_d \end{bmatrix} [C_y \quad I] \right) \begin{bmatrix} \hat{x}(k|k-1) \\ \hat{d}(k|k-1) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} L_x \\ L_d \end{bmatrix} y(k) \quad (2.84)$$

But in the previous subsection we had

$$\hat{d}(k|k) = -C_y \hat{x}(k|k-1) + y(k)$$

and since we assume that $\hat{d}(k+1|k) = \hat{d}(k|k)$ we have

$$\hat{d}(k+1|k) = -C_y \hat{x}(k|k-1) + y(k) \quad (2.85)$$

We can see that the estimate obtained from the observer is the same as this if the observer gain matrix L is

$$L = \begin{bmatrix} L_x \\ L_d \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix}$$

This gives

$$\left(\begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} - \begin{bmatrix} L_x \\ L_d \end{bmatrix} [C_y \quad I] \right) = \begin{bmatrix} A & 0 \\ -C_y & 0 \end{bmatrix}$$

The block-triangular structure of this matrix shows that the observer's eigenvalues are those of the plant (the matrix A) and the remaining ones are zeros. The zeros indicate that 'deadbeat' estimation of the disturbance is obtained, namely that the disturbance estimate is exact after a finite number of steps — and we can see from (2.85) that it is

in fact exact after only 1 step (if the real disturbance behaves exactly according to our model).

The fact that the observer's eigenvalues include those of A shows that this simple disturbance estimation scheme can be used only with stable plants — otherwise the estimate $\hat{x}(k+1|k)$ will get worse and worse as k increases. So we see that, although the 'DMC scheme' for estimating disturbances is simple and intuitive, it has some limitations. We also see that it is easy to overcome these limitations: we can keep the same disturbance model, but use a different observer gain matrix L . Even with unstable plants, it is possible to find L such that all the observer's eigenvalues lie inside the unit disk (providing that the pair (A, C_y) is observable).

Note that if an observer is used, then the form of the state prediction equation (2.66) remains the same, but $x(k)$ has to be replaced by $\hat{x}(k|k)$. We will look at the details of this later.

Suppose now that we use the alternative state-space representation (2.37), and assume again that y and z are the same. Since we are now assuming the presence of a disturbance on the output, we must distinguish between the actual measured output $y(k)$, and the output $\eta(k)$ that would have been obtained in the absence of any disturbance. We now take the state vector to be

$$\xi(k) = \begin{bmatrix} \Delta x(k) \\ \eta(k) \end{bmatrix} \quad (2.86)$$

which gives the plant model:

$$\xi(k+1) = \begin{bmatrix} A & 0 \\ C_y A & I \end{bmatrix} \xi(k) + \begin{bmatrix} B \\ C_y B \end{bmatrix} \Delta u(k) \quad (2.87)$$

$$y(k) = [0 \quad I] \xi(k) + d(k) \quad (2.88)$$

so the standard observer equation is

$$\begin{aligned} \hat{\xi}(k+1) &= \left(\begin{bmatrix} A & 0 \\ C_y A & I \end{bmatrix} - \begin{bmatrix} L_{\Delta x} \\ L_{\eta} \end{bmatrix} [0 \quad I] \right) \hat{\xi}(k) \\ &\quad + \begin{bmatrix} B \\ C_y B \end{bmatrix} \Delta u(k) + \begin{bmatrix} L_{\Delta x} \\ L_{\eta} \end{bmatrix} y(k) \end{aligned} \quad (2.89)$$

Reading along the second row of this equation gives

$$\begin{aligned} \hat{\eta}(k+1) &= C_y A \Delta \hat{x}(k) + (I - L_{\eta}) \hat{\eta}(k) + C_y B \Delta u(k) + L_{\eta} y(k) \\ &= [C_y A \Delta \hat{x}(k) + \hat{\eta}(k) + C_y B \Delta u(k)] + L_{\eta} [y(k) - \hat{\eta}(k)] \end{aligned} \quad (2.90)$$

The first bracketed term is what we would expect the output to be at time $k+1$ in the absence of any disturbance at time $k+1$, and this is corrected by the second term, which depends on the error in the previous prediction.

Now if we have $L_{\eta} = I$, and if we write $\hat{d}(k|k) = y(k) - \hat{\eta}(k)$, then we get precisely the DMC prediction of the next plant output. That is, the difference between the actual and the predicted outputs is taken as the estimate of an output disturbance, and the same estimate is assumed to affect the next output. So we see that, in both state-space

representations, the DMC estimate is obtained by the same observer gain matrix:

$$\begin{bmatrix} L_x \\ L_d \end{bmatrix} = \begin{bmatrix} 0 \\ I \end{bmatrix} = \begin{bmatrix} L_{\Delta x} \\ L_\eta \end{bmatrix} \quad (2.91)$$

(Note that the partitions in these two observer gain matrices have the same dimensions, since d and η are vectors with the same number of elements.) From (2.89) we see that in this case the observer's state transition matrix is

$$\left(\begin{bmatrix} A & 0 \\ C_y A & I \end{bmatrix} - \begin{bmatrix} L_{\Delta x} \\ L_\eta \end{bmatrix} [0 \quad I] \right) = \begin{bmatrix} A & 0 \\ C_y A & 0 \end{bmatrix} \quad (2.92)$$

so again the observer's eigenvalues are located at the plant's pole locations and at zero.

An important observation is that, with either representation, if we have $L_d = L_\eta = I$ then at each step the prediction is corrected by the prediction error at the previous step, whatever the value of L_x or $L_{\Delta x}$ — that is, *even if* $L_x \neq 0$ or $L_{\Delta x} \neq 0$. The significance of this is that most often it is not possible to obtain truly 'optimal' estimates of the state, because not enough is known about the statistical characteristics of disturbances and measurement noises, and because one has limited confidence in the accuracy of one's model. In these circumstances the observer gain becomes, in effect, another 'tuning parameter' which affects the behaviour of the predictive controller — particularly its response to disturbances, its stability margins, etc.: see Sections 7.4.2, 8.2 and 8.3 for further details. The state estimates produced by the observer can therefore bear little relation to any physical variables. But the key feature of predictive control is the ability to respect constraints. Clearly, it will only be possible to keep accurately to output constraints if accurate estimates of the outputs are available. The choice $L_d = I$ or $L_\eta = I$ ensures that the predicted output never diverges too far from the actual output — prediction errors are not allowed to accumulate over more than one step. This will therefore often be a constraint upon the observer design.

On the other hand, if some measurements are known to be very noisy, then there is no point in having the estimate of the real (i.e. meaningful) output jump about, following the noise at every step, which will not only cause unnecessary moves of the plant inputs, but might also cause random activation and de-activation of output constraints. In that case, at least we should be sure that the estimated output follows the real output 'on average'. In particular, if the measured output is constant, the estimated output should converge to the same constant value. Since the observer should be designed to be asymptotically stable, we know that, if the plant input and output measurements are constant ($\Delta u(k) = 0$ and $\Delta x(k) = 0$), then the state estimate will converge to a constant value. Suppose that $\hat{\eta}(k)$ converges to η_0 . From (2.90) we see that, if the measured plant output has constant value $y(k) = y_0$, then

$$\eta_0 = \eta_0 + L_\eta(y_0 - \eta_0) \quad (2.93)$$

and hence $\eta_0 = y_0$ *whatever the value of L_η* (providing that $\det L_\eta \neq 0$). So, 'on average' the observer will always estimate the measured plant output correctly, so long as it is asymptotically stable. If there is no bias (no systematic error) in the measurement, then it will also estimate the real plant output correctly 'on average'. The same conclusion follows if we use the representation (2.82)–(2.83) — see Exercise 2.16.

2.6.4 Independent and realigned models

In Chapter 1 we noted the idea of having an *independent model*, namely one whose predictions are influenced only by the inputs which are applied to the plant, but not by the actual plant response. We can now see that this corresponds precisely to choosing an observer gain such that $L_x = 0$ (or $L_{\Delta x} = 0$ if the alternative representation is used). That part of the model's state vector which is supposed to correspond to the plant's state vector is not affected by output measurements in this case, as can be seen from equation (2.84) or (2.89).

As was already stated in Chapter 1, an independent model can only be used if it is stable. Attempting to use an independent unstable model would lead to rapid divergence of the state estimates from the plant states, which in turn would lead to very inaccurate predictions. With an unstable model, it is necessary to use $L_x \neq 0$ (or $L_{\Delta x} \neq 0$) in order to stabilize the observer.

One widely used procedure with difference equation (transfer function) models is to *realign* the model on the measurements, namely to use actual measurements of past outputs instead of past model outputs, when predicting future outputs. We have already introduced this in Section 1.6. We shall now show that this can be interpreted as using an observer, with a particular way of stabilizing it.

Consider the case when no attempt is made to model any disturbance. This corresponds to the use of equation (1.39). Suppose that a difference equation model is given in the form

$$y(k) + \sum_{i=1}^r A_i y(k-i) = \sum_{i=1}^r B_i u(k-i) \quad (2.94)$$

Several equivalent state-space forms can be obtained [Kai80]. One simple way of getting a state-space form of this model is to define the state to be made up of past inputs and outputs:

$$\begin{aligned} x(k) = & [y(k)^T, y(k-1)^T, \dots, y(k-r+1)^T, u(k-1)^T, u(k-2)^T, \dots, \\ & u(k-r+1)^T]^T \end{aligned} \quad (2.95)$$

Then a state-space model equivalent to (2.94) has its (A, B, C) matrices defined as:

$$A = \left[\begin{array}{ccccccccc} -A_1 & -A_2 & \dots & -A_{r-1} & -A_r & B_2 & \dots & B_{r-1} & B_r \\ I_m & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & I_m & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & I_m & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & \dots & 0 & 0 & I_\ell & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & 0 & 0 & \dots & I_\ell & 0 \end{array} \right] \quad (2.96)$$

$$B = [B_1^T \ 0 \ 0 \ \dots \ 0 \ I_\ell \ 0 \ \dots \ 0]^T \quad (2.97)$$

$$C = [I_m \ 0 \ \dots \ 0 \ 0 \ 0 \ \dots \ 0 \ 0] \quad (2.98)$$

Now suppose that an observer is used with gain matrix

$$L' = [I_m \ 0 \ \dots \ 0]^T \quad (2.99)$$

Then

$$I - L'C = \begin{bmatrix} 0_{m,m} & 0 \\ 0 & I \end{bmatrix} \quad (2.100)$$

so the first element of $\hat{x}(k|k)$ becomes $y(k)$. Furthermore, the form of A ensures that $y(k)$ will become the second (vector) element of $\hat{x}(k+1|k)$ and of $\hat{x}(k+1|k+1)$, the third element of $\hat{x}(k+2|k+1)$ and of $\hat{x}(k+2|k+2)$, and so on. Also $u(k)$ will enter $\hat{x}(k+1|k)$ and $\hat{x}(k+1|k+1)$, and again the form of A will lead to it propagating ‘downwards’ in $\hat{x}(k+2|k+1)$, $\hat{x}(k+3|k+2)$, etc. Thus, after at most r steps of running the observer, it will be true that $\hat{x}(k|k) = x(k)$. That is, the state estimate obtained from the observer will contain present and past values of the input and output, and predictions obtained on the basis of this estimate will be precisely the same as predictions obtained using the ‘realigned’ difference equation model.

The dynamics of the observer are determined by the eigenvalues of $A(I - L'C) = A - LC$, which has the block-triangular form:

$$A(I - L'C) = A - LC = \begin{bmatrix} 0_{m,m} & X \\ 0 & J \end{bmatrix} \quad (2.101)$$

where $X = [-A_2, -A_3, \dots, B_r]$, and

$$J = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ I_m & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & I_m & 0 \end{bmatrix} \quad (2.102)$$

Since this matrix is block-triangular, the top-left block is zero, and all the eigenvalues of J are zero, it is clear that all the eigenvalues of $A - LC$ are zero. That is, this observer is always stable, and in fact ‘deadbeat’; note that no assumptions have been made about the stability of the plant.

This section, then, has shown several things:

1. ‘Realigning’ the prediction model on past data is a way of stabilizing the observer, and explains why it can be used with unstable plant models.
2. It is not the only way of dealing with unstable plant models. There are many other observer gain matrices which give a stable observer, and may sometimes be more suitable.
3. However, the advantage of ‘realigning’ the model, namely of choosing the state as in (2.95), is that the past inputs and outputs are already known, and there is no real need for an observer — the analysis of this section is just an interpretation in terms of observers.

2.7 Example: Citation aircraft model

The following is a constant-speed approximation of some of the linearized dynamics of a Cessna Citation 500 aircraft, when it is cruising at an altitude of 5000 m and a speed of 128.2 m/sec. The elevator angle (rad) is the only input, and the pitch angle (rad), altitude (m), and altitude rate (m/s) are outputs.

$$\dot{x} = Ax + Bu \quad y = Cx + Du$$

where

$$A = \begin{bmatrix} -1.2822 & 0 & 0.98 & 0 \\ 0 & 0 & 1 & 0 \\ -5.4293 & 0 & -1.8366 & 0 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} -0.3 \\ 0 \\ -17 \\ 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -128.2 & 128.2 & 0 & 0 \end{bmatrix} \quad D = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

For the purposes of this example we will pretend that this is an accurate representation of the aircraft dynamics.

The elevator angle is limited to $\pm 15^\circ$ (± 0.262 rad), and the elevator slew rate is limited to $\pm 30^\circ/\text{sec}$ (± 0.524 rad/sec). These are limits imposed by the equipment design, and cannot be exceeded. For passenger comfort the pitch angle is limited to $\pm 20^\circ$ (± 0.349 rad).

Figure 2.6 shows the response to a step change of 40 m in the altitude set-point at time 0, with set-points for pitch angle and altitude rate held at 0, and the reference trajectory equal to the set-point: $r(k + i|k) = [0, 40, 0]^T$ for $k \geq 0$. The sampling interval is $T_s = 0.5$ sec, the prediction horizon is $H_p = 10$ (5 sec), and the control horizon is $H_u = 3$ (1.5 sec). Tracking errors are penalized over the whole prediction horizon ($H_w = 1$). The weights on tracking errors are $Q(i) = I_3$ (the same at each point in the prediction horizon) and the weights on control moves are $R(i) = 1$ (the same at each point in the control horizon). For this magnitude of change none of the constraints are active at any point of the transient, and a linear time-invariant controller results — as will be shown in detail in Chapter 3.

Note that the altitude tracking error dominates the pitch angle and altitude rate errors during most of the transient, so that the pitch angle and altitude rate depart from their set-points in order to allow the altitude error to be reduced. As the required altitude is acquired, all three outputs settle rapidly to their set-points. This behaviour is entirely a result of the numerical values of the errors which arise, the altitude error being sufficiently larger, numerically, than the other errors for its contribution to the cost function to be the dominant one. This will not be the case in general, and — as in all multivariable control problems — problem-dependent scaling must be employed to obtain the desired performance. (For example, if the pitch angle had been represented in degrees rather than radians in the predictive control algorithm then the pitch set-point would have been much more significant, and would have impeded the acquisition of the required altitude.)

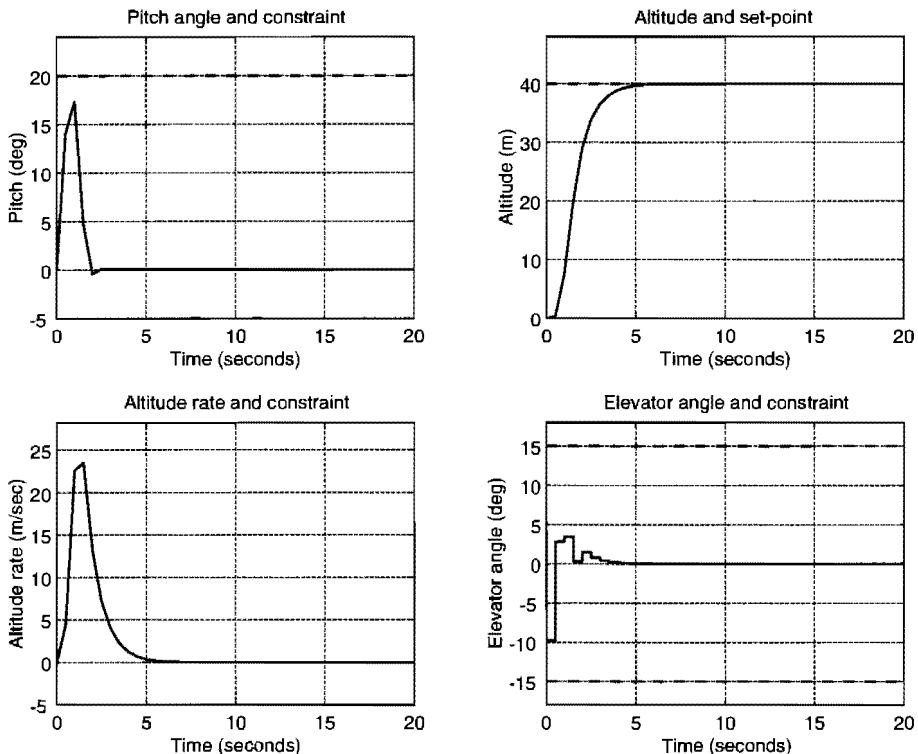


Figure 2.6 Response of Citation aircraft to 40 m step change in altitude set-point.

Other specifications are possible for this manoeuvre. One obvious possibility is to allow the pitch angle and altitude rate to be free until the altitude error has been reduced to some small value. This could be done by reducing the weights on the corresponding errors, possibly for a fixed time, based on the anticipated time required to complete the manoeuvre:

$$Q_1(k+i|k) = Q_3(k+i|k) = \begin{cases} 0 & \text{if } k+i < 8 \\ 1 & \text{if } k+i \geq 8 \end{cases} \quad (2.103)$$

But this would need to be adjusted for each amplitude of required altitude change, and would be prone to error if the manoeuvre was not performed exactly as planned; a much better alternative would be to make the weights state-dependent:⁴

$$Q_1(k+i|k) = Q_3(k+i|k) = \begin{cases} 0 & \text{if } |r_2(k+i|k) - \hat{y}_2(k+i|k)| > 5 \\ 1 & \text{if } |r_2(k+i|k) - \hat{y}_2(k+i|k)| \leq 5 \end{cases} \quad (2.104)$$

Suppose that a larger change in altitude is required: 400 m instead of 40 m. Figure 2.7 shows the response in this case. It can be seen that the pitch angle constraint becomes

⁴ Simulation with state-dependent weights is not possible in the *Model Predictive Control Toolbox*. However, the modified functions `smpc2` and `smpcn12` available on this book's web site make this possible, albeit inefficiently.

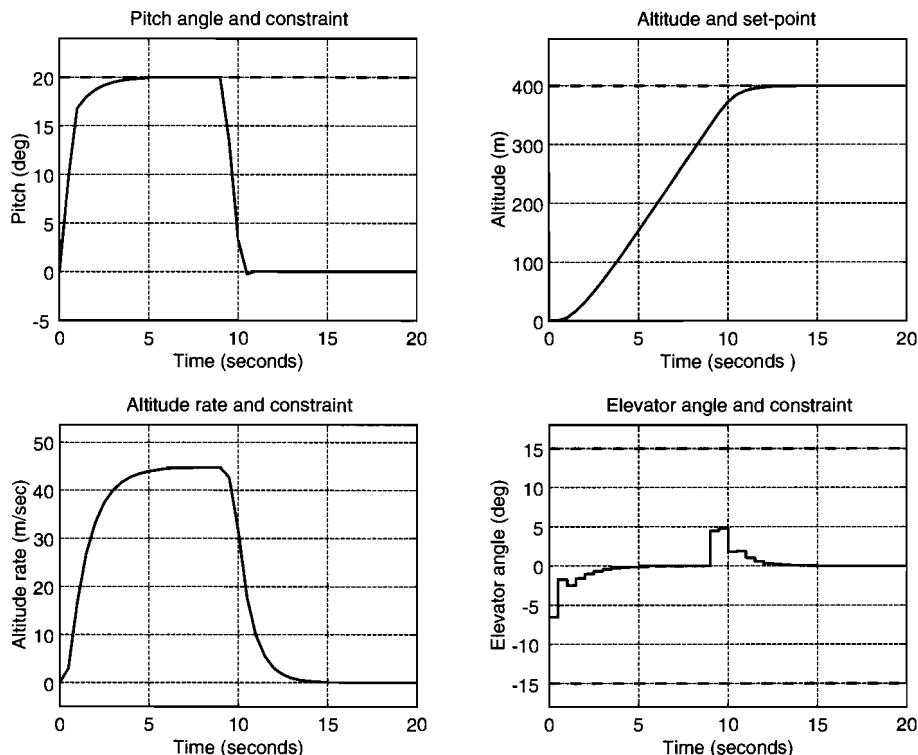


Figure 2.7 Response of Citation aircraft to 400 m step change in altitude set-point.

active for much of the transient, with the result that the altitude rate becomes constant for part of the transient. The control action is now nonlinear.

In Figures 2.6 and 2.7 the rate of change of altitude was unconstrained. Suppose now that it is constrained to 30 m/sec, and an altitude change of 400 m is again required. Figure 2.8 shows the resulting response. It can be seen that the pitch angle constraint is briefly active near the beginning of the transient, but that the altitude rate constraint is active for most of the time. The resulting (nonlinear) behaviour is close to that obtained from a conventional autopilot during an altitude change: a required rate of climb is held until the required altitude is nearly acquired, whereupon the required altitude is acquired and held.

This example demonstrates some of the flexibility that results from the ability to specify and respect constraints. Quite complicated behaviours can be obtained by relatively simple specifications, without any need to introduce mode-switching logic explicitly.⁵ It should be emphasized that the intention here is to demonstrate just how

⁵ There is an analogy here with the difference between ‘procedural’ programming, in which software is used to tell a computer *how* to perform a required computation, and ‘declarative’ programming, in which it is used to define *what* is to be computed. Mode-switching logic specifies *how* the controller is to achieve the required behaviour, whereas the use of constraints specifies *what* behaviour it must achieve.

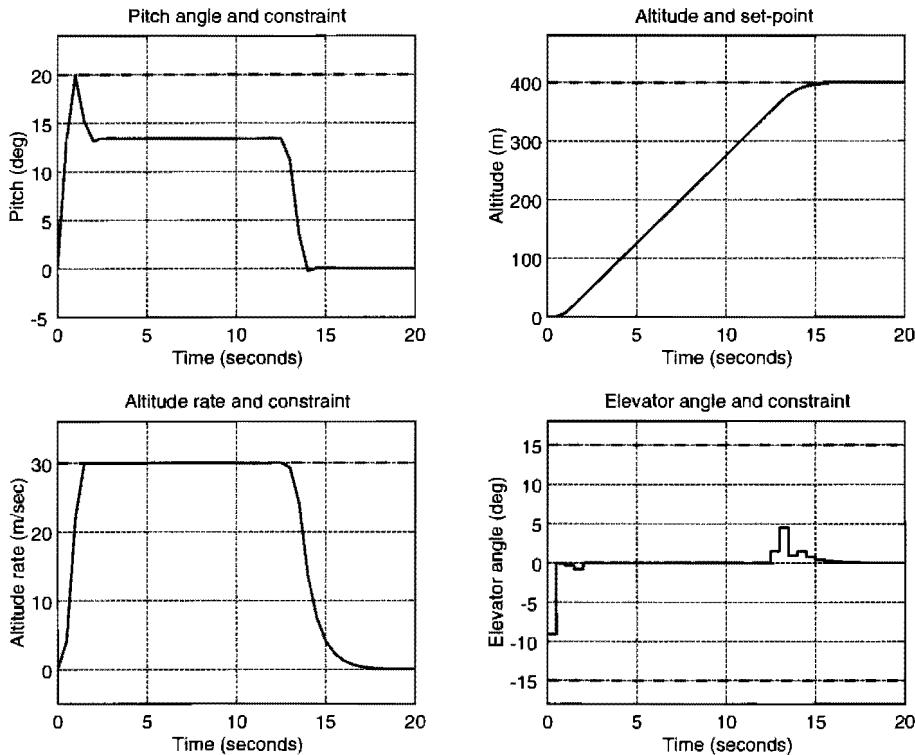


Figure 2.8 Response of Citation aircraft to 400 m step change in altitude set-point, with altitude rate constraint.

much flexibility there is. It is *not* being proposed that constraints are usually a good substitute for set-points. In this example it may well be preferable to have a set-point for the altitude rate, either simultaneously with a constant set-point for the altitude (which may require changing the weights during the manoeuvre, as in (2.103) or (2.104)), or to ramp the altitude reference trajectory from the latest altitude measurement:

$$r(k+i|k) = \begin{bmatrix} 0 \\ y_2(k) + 30T_s i \\ 30 \end{bmatrix} \quad \text{if } \hat{y}_2(k+i|k) < 395 \quad (2.105)$$

$$r(k+i|k) = \begin{bmatrix} 0 \\ 400 \\ 0 \end{bmatrix} \quad \text{if } \hat{y}_2(k+i|k) \geq 395 \quad (2.106)$$

Different ways of specifying such manoeuvres will result in different behaviours in response to disturbances. For example, turbulence which results in the altitude rate exceeding 30 m/sec will result in more aggressive control action to reduce this rate if it is specified by a constraint rather than as a set-point. Figures 2.9 and 2.10 show the responses when turbulence appears as a disturbance pulse on the altitude rate of 5 m/sec, lasting for 5 sec, which starts 5 sec into the transient, in the two cases. Figure 2.9

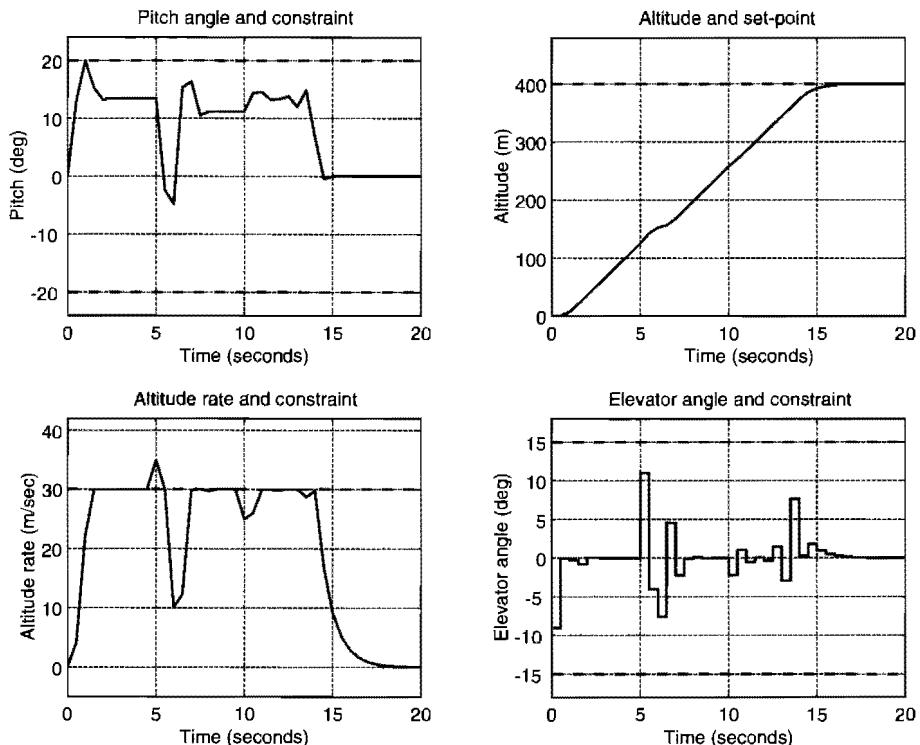


Figure 2.9 Response to disturbance with altitude rate constraint.

corresponds to the constraint of 30 m/sec being imposed on the altitude rate. This constraint is active when the disturbance occurs, causing a violation of the constraint. The controller immediately brings the altitude rate down below the constraint by moving the elevator to $+11^\circ$ for one sample period, which causes the pitch angle to change suddenly from about 13° to -5° , before recovering to a steady value of $+11^\circ$. This value of the pitch angle holds the altitude rate at its constraint while the disturbance is present. The disturbance ceases suddenly 10 seconds into the manoeuvre, which has the immediate effect of reducing the altitude rate. Since there is now no danger of the altitude rate constraint being violated, the controller reacts to this in a much less aggressive manner than it did to the onset of the disturbance. This nicely illustrates the nonlinear behaviour of the controller.

Figure 2.10, on the other hand, shows what happens when there is a set-point of 30 m/sec for the altitude rate instead of a constraint. The set-points for the pitch angle and altitude are constant, as before, at 0° and 400 m, respectively. In order to give the altitude rate set-point sufficient weight, compared with the altitude set-point, the tracking error weights $Q(i)$ were chosen to be $Q(i) = \text{diag}[1, 1, 100]$ for $i = 1, \dots, 5$, and $Q(i) = I_3$ for $i = 6, \dots, 10$ — recall that $H_p = 10$. It can be seen that the response to the disturbance is now much milder. Although the altitude rate remains above 30 m/sec for a little longer than in the previous case, overall it remains closer

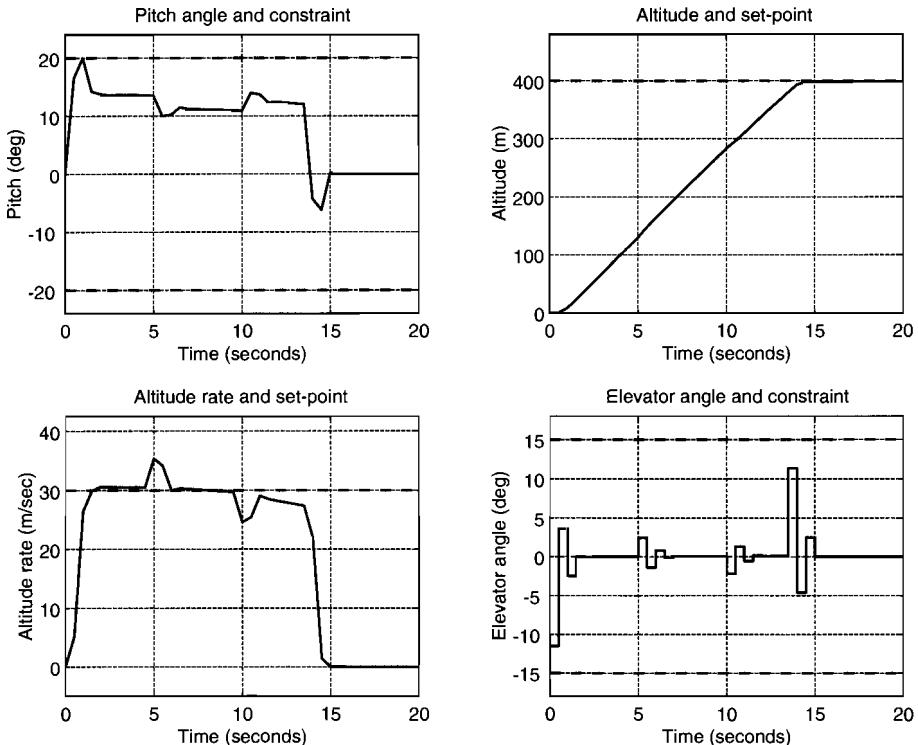


Figure 2.10 Response to disturbance with altitude rate set-point.

to this value than in the previous case, because there is no big dip below 30 m/sec, since the elevator action is now much smaller. It can also be seen that the responses to the start and end of the disturbance pulse are now much more symmetric, as a result of the fact that the controller is now operating in linear mode. The fact that at the end of the manoeuvre there is a more violent transient than in the previous case is an unimportant artefact of the choice of weights and set-point specifications, and could be removed by more careful tuning.

For further discussion of possible specifications of closed-loop behaviours see Section 5.5 and Chapter 7.

Now suppose that the gain from the elevator angle to the altitude rate is mis-modelled, so that the model used by the predictive controller underestimates this gain by 10%. (That is, elements $A(4, 1)$, $A(4, 2)$, $C(3, 1)$, $C(3, 2)$ are 10% smaller in the model than in the plant.) Figure 2.11 shows the response to an altitude set-point change of 400 m, when the altitude rate is constrained to 30 m/sec, and when the constant output disturbance assumption is made on each output. Comparing the response with Figure 2.8 shows that the response is now a little more erratic, with more elevator activity. The altitude rate constraint is violated briefly about 1.5 sec into the transient (by about 0.5 m/sec), but after that it is not violated again. The required altitude is acquired and held without any error in the steady state.

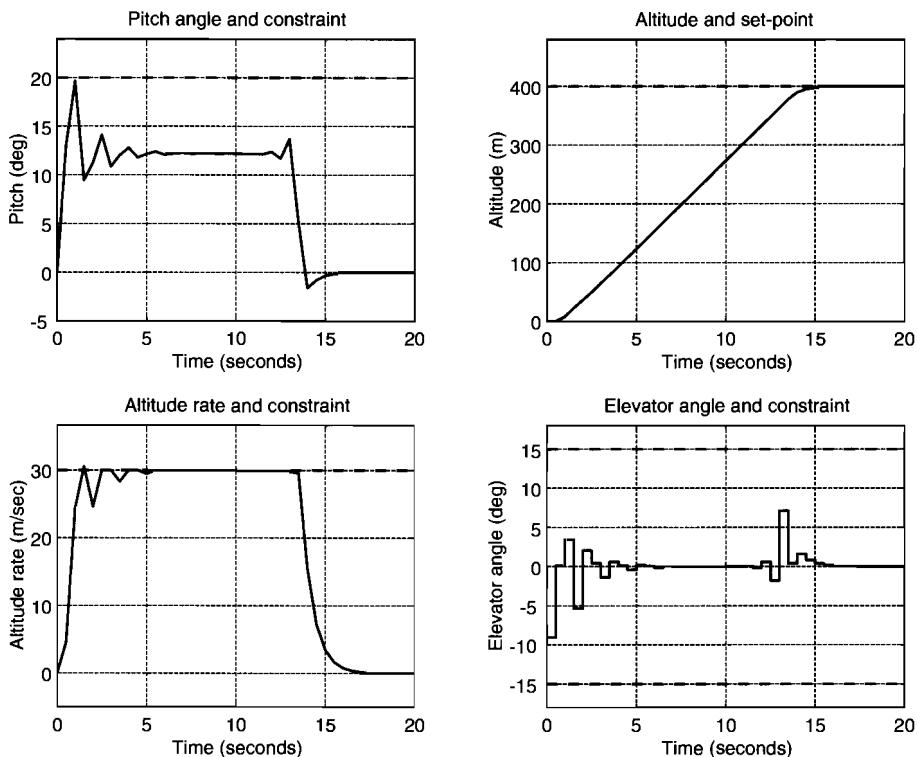


Figure 2.11 Response of Citation aircraft to 400 m step change in altitude set-point, with altitude rate constraint and plant-model error.

Exercises

The first three exercises can be skipped by those familiar with quadratic forms, ∇V , etc.

- 2.1** Use *MATLAB* to find the eigenvalues of the matrix Q which appears in Example 2.1. Check that they are both positive.
- 2.2** (a) Suppose $V(x) = 9x_1^2 + 25x_2^2 + 16x_3^2$. Find Q such that $V(x) = x^T Q x$.
 (b) Suppose $V(x, u) = (5x_1^2 + 2x_2^2 + x_3^2) + (100u_1^2 + 4u_2^2)$. Find Q and R such that $V(x) = x^T Q x + u^T R u$.
 (c) Are the functions V which appear in this problem positive-definite?
- 2.3** Check that the formula (2.19) for the gradient is correct for Example 2.1, by working out the partial derivatives in the elementary way, and comparing the results.
- 2.4** Find the matrix F corresponding to the constraints given in Example 2.4.

2.5 Suppose that we have a 1-state, 1-input model with 1 controlled variable:

$$\begin{aligned}\hat{x}(k+1|k) &= 2\hat{x}(k|k-1) + u(k) \\ \hat{z}(k|k) &= 3\hat{x}(k|k)\end{aligned}$$

and we have the constraint

$$-1 \leq z(k) \leq 2 \quad \text{for all } k$$

If $\hat{x}(k|k) = 3$ and $u(k-1) = -1$, show that the corresponding constraint on $\Delta u(k)$ is

$$-\frac{16}{3} \leq \Delta u(k) \leq -\frac{13}{3}$$

2.6 As practice at using *MATLAB*, and a little revision of discrete-time systems, check the discretization done in Example 2.4. You can create a state-space system object by a command of the form: `cont_sys=ss(Ac,Bc,Cc,Dc)`; where A_c etc. are the continuous time state-space matrices. You can discretize it using `disc_sys=c2d(cont_sys,Ts)`; where T_s is the sample interval. Matrices can be extracted from these system objects by commands of the form `Ad=get(disc_sys,'a')`. (Use `help lti` for more information.)

Is the system stable? With the default discretization method used by `c2d` each eigenvalue λ of A_c should be mapped to $\exp(\lambda T_s)$. Check this (using the functions `eig` and `exp`).

2.7 Suppose that in Example 2.4 the feed tank level H_1 is constrained, to ensure that the feed tank neither runs dry nor overfills (in addition to the other constraints).

- (a) Explain why the matrix C_z in the basic formulation must be changed to represent this. What is the new value of this matrix?
- (b) What are the new dimensions of the matrices E , F and G in this case?
- (c) Do you anticipate any additional difficulty with this case? (*Hint:* Compare the matrices C_y and C_z .)

2.8 Referring to Section 2.4, show that a third possible state-space representation is obtained by taking the augmented state vector to be

$$\xi(k) = \begin{bmatrix} \Delta x(k) \\ y(k-1) \\ z(k-1) \end{bmatrix} \quad (2.107)$$

in which case the plant-model equations become

$$\xi(k+1) = \begin{bmatrix} A & 0 & 0 \\ C_y & I & 0 \\ C_z & 0 & I \end{bmatrix} \xi(k) + \begin{bmatrix} B \\ 0 \\ 0 \end{bmatrix} \Delta u(k) \quad (2.108)$$

$$\begin{bmatrix} y(k) \\ z(k) \end{bmatrix} = \begin{bmatrix} C_y & I & 0 \\ C_z & 0 & I \end{bmatrix} \xi(k) \quad (2.109)$$

2.9 Write a *MATLAB* function ‘c2dd’ to obtain the discrete-time model in the form (2.58) from a continuous-time model in the form (2.45) when there is a computational delay, using the *Control System Toolbox* function c2d.

2.10 For the linearized model of the Citation aircraft defined in Section 2.7:

- (a) Obtain discrete-time models of the aircraft, assuming a sampling interval of 0.1 second, with
 - (i) no computational delay,
 - (ii) a computational delay of 0.02 second (using the function c2dd referred to in Exercise 2.9).
- (b) Compare the step responses from the elevator angle to the three outputs for the two models.
- (c) Compare the frequency responses from the elevator angle to the three outputs for the two models. Comment on the implications, for control, of neglecting the computational delay.

(The function `makecita`, available on this book’s web site, creates the continuous-time Citation model.)

2.11 Using equations (2.67)–(2.69), or (2.70), write the predictions $\hat{z}(k + i|k)$ in matrix-vector form (analogously to (2.66)).

2.12 Suppose that $D_z \neq 0$ in (2.4). Introduce the new vector of controlled variables $\tilde{z}(k) = z(k) - D_z u(k)$. Show that if the cost function and linear inequality constraints are rewritten in terms of \tilde{z} instead of z then the predictive control problem remains in the standard form, namely the cost function remains quadratic in $\Delta\hat{u}$ and the constraints remain as linear inequalities involving $\Delta\hat{u}$.

2.13 Suppose that $D_z \neq 0$ in (2.4), but that the change of variable $\tilde{z}(k) = z(k) - D_z u(k)$ is not made. How does the computation of $\hat{z}(k + i|k)$ need to be modified from (2.67)–(2.69)? Can you find an expression for the vector $[\hat{z}(k + 1|k), \dots, \hat{z}(k + H_p|k)]^T$ in matrix-vector form? (Note that the predictions \hat{z} remain linear in the \hat{x} and $\Delta\hat{u}$ variables.)

2.14 Show that the pair

$$\begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix}, \quad [C_y, \quad I]$$

is observable if and only if the pair (A, C_y) is observable.

(The significance is that if this pair is observable then an observer gain matrix L can always be found which places the eigenvalues of the ‘constant output’ observer (2.84) at arbitrary locations.)

2.15 A plant with 1 input, 1 output and 1 state has the model

$$x(k+1) = 0.9x(k) + 0.5u(k), \quad y(k) = x(k)$$

Predictive control is to be used with $Q(i) = 1$, $R(i) = 0$, $H_p = 30$, and $H_u = 2$. A constant step disturbance at the plant input is assumed (that is, $u(k)$ is replaced by $u(k) + d(k)$).

- (a) Show how the plant model can be augmented to incorporate this disturbance model.
 - (b) Design a state observer with a ‘deadbeat’ response for the plant with this disturbance model. (The easy way to do this is to use the *Model Predictive Control Toolbox* function `smpcest(imod,Q,R)`, where Q and R are fictitious state and output noise covariances, respectively, with R very small. The observer approaches a deadbeat observer as R approaches 0.)
 - (c) Simulate the response of the predictive controller to an input step disturbance when your observer is used. Compare the response to that obtained when the default ‘DMC’ observer is used. (Assume there are no constraints.)
 - (d) How does the observer design affect the set-point response?
 - (e) Work out the details of how to use the *Control System Toolbox* function `place` to design observers for use with the *Model Predictive Control Toolbox*. (You will need to pay attention to the model representations used by the *Model Predictive Control Toolbox*.)
- 2.16** Using the representation (2.82)–(2.83) show that the plant output estimated by any asymptotically stable observer converges to the measured plant output, if that is constant (assuming that the plant input and state are both constant).

Solving predictive control problems

3.1	Unconstrained problems	74
3.2	Constrained problems	81
3.3	Solving QP problems	88
3.4	Softening the constraints	97
	Exercises	104

This chapter explains how the standard predictive control problem can be solved. It also discusses the structure of the controller which results. *Active set* and *interior point* methods of solving the optimization problems which arise in predictive control are introduced, and the very important topic of *constraint softening* is treated.

3.1 Unconstrained problems

3.1.1 Measured state, no disturbances

Recall that the cost function which we must minimize is

$$V(k) = \sum_{i=H_w}^{H_p} \|\hat{z}(k+i|k) - r(k+i)\|_{Q(i)}^2 + \sum_{i=0}^{H_u-1} \|\Delta\hat{u}(k+i|k)\|_{R(i)}^2 \quad (3.1)$$

We can rewrite this as

$$V(k) = \|\mathcal{Z}(k) - \mathcal{T}(k)\|_{\mathcal{Q}}^2 + \|\Delta\mathcal{U}(k)\|_{\mathcal{R}}^2 \quad (3.2)$$

where

$$\mathcal{Z}(k) = \begin{bmatrix} \hat{z}(k + H_w|k) \\ \vdots \\ \hat{z}(k + H_p|k) \end{bmatrix} \quad \mathcal{T}(k) = \begin{bmatrix} \hat{r}(k + H_w|k) \\ \vdots \\ \hat{r}(k + H_p|k) \end{bmatrix}$$

$$\Delta\mathcal{U}(k) = \begin{bmatrix} \Delta\hat{u}(k|k) \\ \vdots \\ \Delta\hat{u}(k + H_u - 1|k) \end{bmatrix}$$

and the weighting matrices \mathcal{Q} and \mathcal{R} are given by

$$\mathcal{Q} = \begin{bmatrix} Q(H_w) & 0 & \cdots & 0 \\ 0 & Q(H_w + 1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q(H_p) \end{bmatrix} \quad (3.3)$$

$$\mathcal{R} = \begin{bmatrix} R(0) & 0 & \cdots & 0 \\ 0 & R(1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R(H_u - 1) \end{bmatrix} \quad (3.4)$$

Also recall from (2.66) and (2.70) — and from Exercises 2.11 and 2.13 of Chapter 2 — that \mathcal{Z} has the form

$$\mathcal{Z}(k) = \Psi x(k) + \Upsilon u(k - 1) + \Theta \Delta\mathcal{U}(k) \quad (3.5)$$

for suitable matrices Ψ , Υ and Θ . Define

$$\mathcal{E}(k) = \mathcal{T}(k) - \Psi x(k) - \Upsilon u(k - 1) \quad (3.6)$$

This can be thought of as a ‘tracking error’, in the sense that it is the difference between the future target trajectory and the ‘free response’ of the system, namely the response that would occur over the prediction horizon if no input changes were made — that is, if we set $\Delta\mathcal{U}(k) = 0$. And if $\mathcal{E}(k)$ really were 0, then it would indeed be correct to set $\Delta\mathcal{U}(k) = 0$. Now we can write

$$V(k) = \|\Theta \Delta\mathcal{U}(k) - \mathcal{E}(k)\|_{\mathcal{Q}}^2 + \|\Delta\mathcal{U}(k)\|_{\mathcal{R}}^2 \quad (3.7)$$

$$= [\Delta\mathcal{U}(k)^T \Theta^T - \mathcal{E}(k)^T] \mathcal{Q} [\Theta \Delta\mathcal{U}(k) - \mathcal{E}(k)] + \Delta\mathcal{U}(k)^T \mathcal{R} \Delta\mathcal{U}(k) \quad (3.8)$$

$$= \mathcal{E}(k)^T \mathcal{Q} \mathcal{E}(k) - 2\Delta\mathcal{U}(k)^T \Theta^T \mathcal{Q} \mathcal{E}(k) + \Delta\mathcal{U}(k)^T [\Theta^T \mathcal{Q} \Theta + \mathcal{R}] \Delta\mathcal{U}(k) \quad (3.9)$$

But this has the form

$$V(k) = \text{const} - \Delta\mathcal{U}(k)^T \mathcal{G} + \Delta\mathcal{U}(k)^T \mathcal{H} \Delta\mathcal{U}(k) \quad (3.10)$$

where

$$\mathcal{G} = 2\Theta^T \mathcal{Q} \mathcal{E}(k) \quad (3.11)$$

and

$$\mathcal{H} = \Theta^T \mathcal{Q} \Theta + \mathcal{R}, \quad (3.12)$$

and neither \mathcal{G} nor \mathcal{H} depends on $\Delta\mathcal{U}(k)$.

To find the optimal $\Delta\mathcal{U}(k)$ we can now find the gradient of $V(k)$ and set it to zero. From (3.10) we find

$$\nabla_{\Delta\mathcal{U}(k)} V = -\mathcal{G} + 2\mathcal{H}\Delta\mathcal{U}(k) \quad (3.13)$$

so the optimal set of future input moves is

$$\Delta\mathcal{U}(k)_{opt} = \frac{1}{2} \mathcal{H}^{-1} \mathcal{G}$$

(3.14)

Remember that we use only the part of this solution corresponding to the first step, in accordance with the receding horizon strategy. So if the number of plant inputs is ℓ then we just use the first ℓ rows of the vector $\Delta\mathcal{U}(k)_{opt}$. We can represent this as

$$\Delta u(k)_{opt} = [I_\ell, \underbrace{0_\ell, \dots, 0_\ell}_{(H_u-1) \text{ times}}] \Delta\mathcal{U}(k)_{opt} \quad (3.15)$$

where I_ℓ is the $\ell \times \ell$ identity matrix, and 0_ℓ is the $\ell \times \ell$ zero matrix.

Note that we can write $\Delta u(k)_{opt}$ here, rather than $\hat{u}(k|k)_{opt}$, because we have now found the solution and this is the input that is *really* applied to the plant at time k .

Does $\Delta\mathcal{U}(k)_{opt}$ really give a minimum of the cost function V ? It certainly gives a stationary point, but that is not enough to guarantee a minimum. Differentiating the gradient $\nabla_{\Delta\mathcal{U}(k)} V$ (3.13) again with respect to $\Delta\mathcal{U}(k)$ gives the matrix of second derivatives, or *Hessian*, of V :

$$\frac{\partial^2 V}{\partial \Delta\mathcal{U}(k)^2} = 2\mathcal{H} = 2(\Theta^T \mathcal{Q} \Theta + \mathcal{R}). \quad (3.16)$$

We have assumed that $\mathcal{Q}(i) \geq 0$ for each i , and this ensures that $\Theta^T \mathcal{Q} \Theta \geq 0$. So if $\mathcal{R} > 0$ then the Hessian is certainly positive-definite, which is enough to guarantee that we have a minimum. And this will be the case if $R(i) > 0$ for each i .

But sometimes we may want to have no penalty on the input moves, which would lead to $\mathcal{R} = 0$. Or we may want to leave the moves of some inputs unpenalized, or the moves at some points in the control horizon unpenalized. In these cases we will have $\mathcal{R} \geq 0$, but not $\mathcal{R} > 0$. When $\mathcal{R} = 0$ we need $\Theta^T \mathcal{Q} \Theta > 0$ in order to have a minimum — and of course, to ensure that \mathcal{H}^{-1} exists. In intermediate cases, when $\mathcal{R} \geq 0$, we need to ensure that $\Theta^T \mathcal{Q} \Theta + \mathcal{R} > 0$.

Let us check the dimensions of the various matrices introduced in this section. Θ has as many rows as there are elements in \mathcal{Z} . If we have p controlled outputs, and we predict between steps H_w and H_p , then there are $p(H_p - H_w + 1)$ of these elements. The number of columns in Θ is the same as the number of elements in $\Delta\mathcal{U}(k)$, which is ℓH_u . $\mathcal{Q}(i)$ has m rows and columns, so \mathcal{Q} has $m(H_p - H_w + 1)$ of each. Thus $\mathcal{H} = \Theta^T \mathcal{Q} \Theta$ is square, with ℓH_u rows and columns. This agrees (as it must!) with the number of rows and columns in \mathcal{R} . Similar checks on the remaining matrices lead to Table 3.1.

Table 3.1 Dimensions of matrices and vectors involved in computing the optimal input moves.
(The plant has ℓ inputs, n states, and m controlled outputs.)

Matrix	Dimensions
\mathcal{Q}	$m(H_p - H_w + 1) \times m(H_p - H_w + 1)$
\mathcal{R}	$\ell H_u \times \ell H_u$
Ψ	$m(H_p - H_w + 1) \times n$
Υ	$m(H_p - H_w + 1) \times \ell$
Θ	$m(H_p - H_w + 1) \times \ell H_u$
\mathcal{E}	$m(H_p - H_w + 1) \times 1$
\mathcal{G}	$\ell H_u \times 1$
\mathcal{H}	$\ell H_u \times \ell H_u$

3.1.2 Formulation as a least-squares problem

The optimal solution, as expressed in (3.14), should *never* be obtained by computing the inverse of \mathcal{H} . The matrix Θ is often ill-conditioned, which can result in \mathcal{H} being ill-conditioned. It is therefore imperative to pay attention to the numerical algorithms involved in finding the optimal solution.

The best way of computing the solution is by solving it as a ‘least-squares’ problem. It is also a way which gives some additional insight.

Since $\mathcal{Q} \geq 0$ and $\mathcal{R} \geq 0$, we can find matrices $S_{\mathcal{Q}}$ and $S_{\mathcal{R}}$ which are their ‘square-roots’:

$$S_{\mathcal{Q}}^T S_{\mathcal{Q}} = \mathcal{Q} \quad S_{\mathcal{R}}^T S_{\mathcal{R}} = \mathcal{R}$$

If \mathcal{Q} and \mathcal{R} are diagonal, it is trivial to do this — just take the square-root of each diagonal element. If they are not diagonal, square-roots can be obtained by using the ‘Cholesky’ algorithm (function `chol` in *MATLAB*) for positive-definite matrices, or by using other algorithms, such as singular value decomposition (`svd`) for semi-definite matrices.

Now consider the vector

$$\begin{bmatrix} S_{\mathcal{Q}}\{\Theta\Delta\mathcal{U}(k) - \mathcal{E}(k)\} \\ S_{\mathcal{R}}\Delta\mathcal{U}(k) \end{bmatrix}$$

We shall show that the squared ‘length’ of this vector, or equivalently, the sum of squares of its elements, is the same as the cost function $V(k)$, so that $\Delta\mathcal{U}(k)_{opt}$ is the value of $\Delta\mathcal{U}(k)$ which minimizes this length.

$$\left\| \begin{bmatrix} S_{\mathcal{Q}}\{\Theta\Delta\mathcal{U}(k) - \mathcal{E}(k)\} \\ S_{\mathcal{R}}\Delta\mathcal{U}(k) \end{bmatrix} \right\|^2 = \left\| \begin{bmatrix} S_{\mathcal{Q}}\{\mathcal{Z}(k) - \mathcal{T}(k)\} \\ S_{\mathcal{R}}\Delta\mathcal{U}(k) \end{bmatrix} \right\|^2 \quad (3.17)$$

$$= [\mathcal{Z}(k) - \mathcal{T}(k)]^T S_{\mathcal{Q}}^T S_{\mathcal{Q}} [\mathcal{Z}(k) - \mathcal{T}(k)] + \Delta\mathcal{U}(k)^T S_{\mathcal{R}}^T S_{\mathcal{R}} \Delta\mathcal{U}(k) \quad (3.18)$$

$$= \|\mathcal{Z}(k) - \mathcal{T}(k)\|_{\mathcal{Q}}^2 + \|\Delta\mathcal{U}(k)\|_{\mathcal{R}}^2 \quad (3.19)$$

$$= V(k) \quad (3.20)$$

So $\Delta\mathcal{U}(k)_{opt}$ is the ‘least-squares’ solution of the equation

$$\begin{bmatrix} S_Q \{\Theta \Delta\mathcal{U}(k) - \mathcal{E}(k)\} \\ S_R \Delta\mathcal{U}(k) \end{bmatrix} = 0 \quad (3.21)$$

or, equivalently, of:

$$\begin{bmatrix} S_Q \Theta \\ S_R \end{bmatrix} \Delta\mathcal{U}(k) = \begin{bmatrix} S_Q \mathcal{E}(k) \\ 0 \end{bmatrix} \quad (3.22)$$

Equations of the form $A\theta = b$ can be solved in a least-squares sense using the ‘QR’ algorithm. In MATLAB this solution is obtained as $\theta_{opt} = A \setminus b$. Although formally this solution is the same as $\theta_{opt} = (A^T A)^{-1} A^T b$ (which gives (3.14)), this algorithm avoids ‘squaring up’ A , and never forms the product $A^T A$, or computes its inverse. If A is ill-conditioned and/or large this is crucial, since it avoids unnecessary loss of precision. For more details see any book on numerical linear algebra, such as [GL89], or Chapter 8 of [Mac89] (or the MATLAB documentation).

Hence we have, using MATLAB notation:

$$\Delta\mathcal{U}(k)_{opt} = \begin{bmatrix} S_Q \Theta \\ S_R \end{bmatrix} \setminus \begin{bmatrix} S_Q \mathcal{E}(k) \\ 0 \end{bmatrix} \quad (3.23)$$

Equation (3.22) is almost always over-determined: there are not enough degrees of freedom to get an exact solution. It contains $m(H_p - H_w + 1) + \ell H_u$ scalar equations, but there are only ℓH_u variables to be solved for. Even in the special case of $R = 0$, when there is no penalty on input moves, and the equation simplifies to

$$S_Q \Theta \Delta\mathcal{U}(k) = S_Q \mathcal{E}(k) \quad (3.24)$$

there are usually more scalar equations than variables. A unique exact solution exists only if the matrix $S_Q \Theta$ is square and non-singular, which requires $m(H_p - H_w + 1) = \ell H_u$. However, we usually have $m(H_p - H_w + 1) > \ell H_u$. Note that in the case of $R = 0$ we cannot remove S_Q from each side of the equation — doing so would give a different solution, since it would change the weighting used when minimizing the sum of squares of the error.

Why do we expect ill-conditioning in the predictive control problem? Typically it arises in the following way. Suppose we had two controlled variables which behaved exactly the same as each other. Then the corresponding pairs of rows of Ψ would be identical. Consequently the rank of Θ (the number of linearly independent rows) would be smaller than expected from its dimensions by $H_p - H_w + 1$ — that is how many pairs of identical rows there would be. If it were sufficiently smaller, the rank could become smaller than ℓH_u , at which point $\Theta^T Q \Theta$ would become singular. If we also had $R = 0$, then \mathcal{H} would be singular. In practice variables do not behave exactly the same as each other. But sometimes they behave very similarly. Consider a distillation column with 40 trays, for example. Adjacent trays do not behave identically, but obviously if you can manipulate only a few temperatures in the column, adjacent trays are going to react very similarly. In such cases one can have many rows of Θ being ‘nearly linearly dependent’ on each other, and then \mathcal{H} can be nearly singular. Even without this cause,

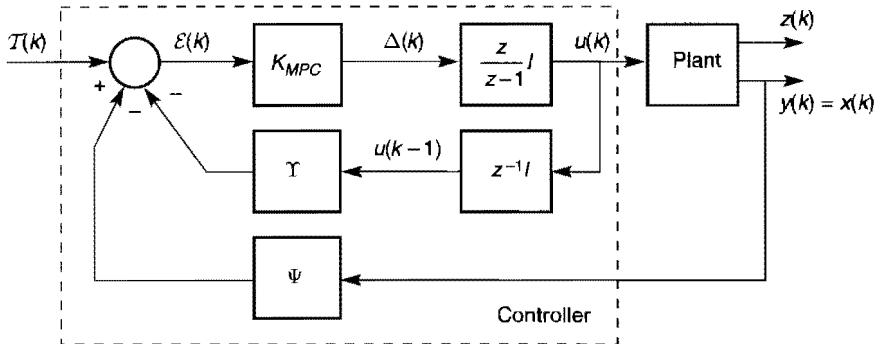


Figure 3.1 Structure of controller with no constraints and full state measurement.

the rows near the bottom of Θ are often ‘nearly linearly dependent’ if $H_p - H_w \gg H_u$. One remedy for this is to increase the diagonal elements of \mathcal{R} — but that may distort the problem away from the real one [QB96].

3.1.3 Structure of the unconstrained controller

Recall from (3.15) and (3.6) that

$$\Delta u(k)_{opt} = [I_\ell, 0_\ell, \dots, 0_\ell] \mathcal{H}^{-1} \Theta^T \mathcal{Q} \mathcal{E}(k) \quad (3.25)$$

and

$$\mathcal{E}(k) = \mathcal{T}(k) - \Psi x(k) - \Upsilon u(k-1). \quad (3.26)$$

The only part of this solution which changes from step to step is the ‘tracking error’ $\mathcal{E}(k)$. Consequently the predictive controller, for the unconstrained problem and with full state measurement, can be drawn as in Figure 3.1.

The block labelled K_{MPC} is defined by

$$K_{MPC} = [I_\ell, 0_\ell, \dots, 0_\ell] \mathcal{H}^{-1} \Theta^T \mathcal{Q} \quad (3.27)$$

We point out that the ‘correct’ way of computing K_{MPC} is (again using *MATLAB* notation, including the ‘:’ operator to pick out the first ℓ rows of the solution):

$$K_{full} = \begin{bmatrix} S_Q \Theta \\ S_R \end{bmatrix} \backslash \begin{bmatrix} S_Q \\ 0 \end{bmatrix} \quad (3.28)$$

$$K_{MPC} = K_{full}(1 : \ell, :) \quad (3.29)$$

(This works for the following reason: If we had $\mathcal{E}(k) = [1, 0, \dots, 0]^T$ then we would effectively have only the first column of S_Q on the right-hand side of (3.23). $\mathcal{E}(k) = [0, 1, 0, \dots, 0]^T$ would effectively give the second column, etc. Since $\mathcal{E}(k)$ enters the solution linearly, it is only necessary to solve (3.23) with these columns on the right-hand side. The ‘\’ operator in *MATLAB* is smart enough to solve for all the columns

simultaneously — and this is very efficient, since most of the work involved is in computing the QR decomposition of the left-hand side, which needs to be done only once.)

It is clear from the figure that the controller is a linear time-invariant system in this case. So it is possible to compute its frequency response, stability margins etc. We shall do that kind of analysis in a later chapter. Note that the controller is, in general, dynamic. It is not just ‘state feedback’, except under very special circumstances.

The matrix K_s computed by the *Model Predictive Control Toolbox* function `smpccon` is related to K_{MPC} as follows:

$$K_s = K_{MPC} \left[\begin{bmatrix} I \\ I \\ \vdots \\ I \end{bmatrix}, -\Psi, -\Upsilon \right] \quad (3.30)$$

so that

$$\Delta u(k)_{opt} = K_s \begin{bmatrix} T(k) \\ x(k) \\ u(k-1) \end{bmatrix} \quad (3.31)$$

3.1.4 Estimated state

Now we need to address the more realistic case, when we do not have measurements of the whole state vector, and must use an observer. We will see that the solution is very similar to the solution in the previous case. In fact, we will see that all the gain matrices involved are exactly the same as in the previous case. The only difference is that we need to use an observer, and use the state estimate $\hat{x}(k|k)$ to replace the measured state $x(k)$. The controller structure in this case is shown in Figure 3.2. This is again a linear time-invariant system, but now with more dynamic complexity, because its state vector includes the observer state.

To obtain the vector of predicted controlled outputs, $\mathcal{Z}(k)$, it is reasonable to go back to equation (3.5), and simply replace $x(k)$ by the best estimate of it available to us, namely $\hat{x}(k|k)$. We say ‘reasonable’ because it is not self-evident that this is the ‘optimal’ thing to do in any sense. The *Separation Principle* or *Certainty Equivalence Principle* says that it *is* the optimal thing to do if one is solving a stochastic linear quadratic problem, and one has Gaussian noises acting on the states and outputs, and the observer gain L' is obtained using Kalman filtering theory [AM79, BH75, BGW90]. But in general it is just a heuristic, albeit one which is very widely used in control engineering. After all, no other obvious alternative presents itself.

So now we define

$$\mathcal{Z}(k) = \Psi \hat{x}(k|k) + \Upsilon u(k-1) + \Theta \Delta u(k) \quad (3.32)$$

and we make the corresponding change in the definition of the ‘tracking error’ $\mathcal{E}(k)$ (compare with equation (3.6)):

$$\mathcal{E}(k) = T(k) - \Psi \hat{x}(k|k) - \Upsilon u(k-1) \quad (3.33)$$

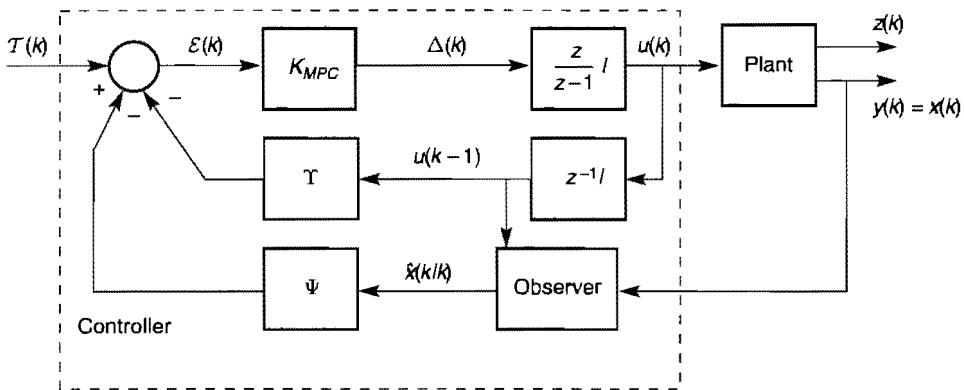


Figure 3.2 Structure of controller with no constraints and state observer.

Once these changes have been made, the derivation of the optimal control $\Delta u(k)_{opt}$ is exactly the same as in earlier sections. Hence we arrive at the controller structure shown in Figure 3.2.

3.2 Constrained problems

3.2.1 Formulation as a QP problem

Now we deal with the case when constraints are present. Recall that these are in the form:

$$E \begin{bmatrix} \Delta \mathcal{U}(k) \\ 1 \end{bmatrix} \leq 0 \quad (3.34)$$

$$F \begin{bmatrix} \mathcal{U}(k) \\ 1 \end{bmatrix} \leq 0 \quad (3.35)$$

$$G \begin{bmatrix} \mathcal{Z}(k) \\ 1 \end{bmatrix} \leq 0 \quad (3.36)$$

where $\mathcal{U}(k) = [\hat{u}(k|k)^T, \dots, \hat{u}(k+H_u-1|k)^T]^T$ is defined analogously to $\Delta \mathcal{U}(k)$. We have to express all of these as constraints on $\Delta \mathcal{U}(k)$.

Suppose F has the form

$$F = [F_1, F_2, \dots, F_{H_u}, f]$$

where each F_i is of size $q \times m$, and f has size $q \times 1$, so that (3.35) can be written as

$$\sum_{i=1}^{H_u} F_i \hat{u}(k+i-1|k) + f \leq 0.$$

Since

$$\hat{u}(k+i-1|k) = u(k-1) + \sum_{j=0}^{i-1} \Delta\hat{u}(k+j|k)$$

we can write (3.35) as

$$\begin{aligned} & \sum_{j=1}^{H_u} F_j \Delta\hat{u}(k|k) + \sum_{j=2}^{H_u} F_j \Delta\hat{u}(k+1|k) + \dots + F_{H_u} \Delta\hat{u}(k+H_u-1|k) \\ & + \sum_{j=1}^{H_u} F_j u(k-1) + f \leq 0 \end{aligned}$$

Now define $F_i = \sum_{j=i}^{H_u} F_j$ and $F = [F_1, \dots, F_{H_u}]$. Then (3.35) can be written as

$$F \Delta\mathcal{U}(k) \leq -F_1 u(k-1) - f \quad (3.37)$$

where the right-hand side of the inequality is a vector, which is known at time k . So we have converted (3.35) into a linear inequality constraint on $\Delta\mathcal{U}(k)$.

Note that if we have simple range constraints on the inputs, of the form

$$u_{low}(k+i) \leq \hat{u}(k+i|k) \leq u_{high}(k+i) \quad (3.38)$$

then this inequality takes quite a simple form. (See Exercise 3.6.)

Now we have to do a similar thing for (3.36). Fortunately we have already done most of the work needed in this case. Assuming full state measurements, we can use (3.5) to write (3.36) as

$$G \begin{bmatrix} \Psi x(k) + \Upsilon u(k-1) + \Theta \Delta\mathcal{U}(k) \\ 1 \end{bmatrix} \leq 0$$

Now letting $G = [\Gamma, g]$, where g is the last column of G , this is the same as

$$\Gamma[\Psi x(k) + \Upsilon u(k-1)] + \Gamma\Theta \Delta\mathcal{U}(k) + g \leq 0$$

or

$$\Gamma\Theta \Delta\mathcal{U}(k) \leq -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - g \quad (3.39)$$

which is in the required form.

If we have only state estimates available, then we replace $x(k)$ by $\hat{x}(k|k)$, just as we did in Section 3.1.4.

It only remains to put inequality (3.34) into the form

$$W \Delta\mathcal{U}(k) \leq w \quad (3.40)$$

(see Exercise 3.6). Then we can assemble inequalities (3.37), (3.39), and (3.40) into the single inequality

$$\begin{bmatrix} F \\ \Gamma\Theta \\ W \end{bmatrix} \Delta\mathcal{U}(k) \leq \begin{bmatrix} -F_1 u(k-1) - f \\ -\Gamma[\Psi x(k) + \Upsilon u(k-1)] - g \\ w \end{bmatrix} \quad (3.41)$$

(Replace $x(k)$ by $\hat{x}(k|k)$ if an observer is used.)

Now the cost function $V(k)$ which we have to minimize is still the same as in the unconstrained case. So, from (3.10), we see that we have to solve the following constrained optimization problem:

$$\text{minimize } \Delta\mathcal{U}(k)^T \mathcal{H} \Delta\mathcal{U}(k) - \mathcal{G}^T \Delta\mathcal{U}(k) \quad (3.42)$$

subject to the inequality constraint (3.41). But this has the form

$$\min_{\theta} \frac{1}{2} \theta^T \Phi \theta + \phi^T \theta \quad (3.43)$$

subject to

$$\Omega\theta \leq \omega \quad (3.44)$$

which is a standard optimization problem known as the *Quadratic Programming* (or *QP*) problem, and standard algorithms are available for its solution.

Similarly to solving the unconstrained problem, it is better to pass the *QP* problem to a solution algorithm in ‘square root’ form, namely in the form

$$\min_{\Delta\mathcal{U}(k)} \left\| \begin{bmatrix} S_Q \{\Theta \Delta\mathcal{U}(k) - \mathcal{E}(k)\} \\ S_R \Delta\mathcal{U}(k) \end{bmatrix} \right\|^2 \quad \text{subject to (3.41).} \quad (3.45)$$

Since $\mathcal{H} \geq 0$, the *QP* problem which we have to solve is *convex*. This is extremely good news — see Mini-Tutorial 3. Because of the convexity we can guarantee termination of the optimization problem, and because of the additional structure of the *QP* problem, we can estimate how long it will take to solve. This is an extremely desirable property for an algorithm which has to be used on-line, and to keep up with the real-time operation of the plant.

A major problem which can occur with constrained optimization is that the problem may be infeasible. Standard *QP* solvers just stop in such cases, their only output being a message such as **Problem Infeasible**, or perhaps some diagnostics in certain cases. This is obviously unacceptable as a substitute for a control signal which must be provided to the plant. So when implementing predictive control it is essential to take steps either to avoid posing an infeasible problem, or to have a ‘back-up’ method of computing the control signal. Various approaches to this have been suggested, including:

- ✿ Avoid ‘hard’ constraints on z .
- ✿ Actively manage the constraint definition at each k .
- ✿ Actively manage the horizons at each k .
- ✿ Use non-standard solution algorithms.

We shall examine these more closely later in the book.

In general optimization problems are solved numerically by ‘going downhill’ — assuming a minimization problem — until one reaches a minimum. The big problem with this is that in general problems there are many ‘local’ minima, and the algorithm is very likely to be stuck in such a local minimum, unaware that the true ‘global’ minimum is elsewhere.

A *convex* optimization problem is one in which this problem does not occur. Because of the convexity of the objective function, there is only one minimum — or possibly a connected set of equally good minima, as on a flat valley floor. When solving a convex problem, one is guaranteed that a global minimum will eventually be reached if one keeps ‘going downhill’.

For a ‘smooth’ problem the property of convexity can be established from the **Hessian** of the objective function — it has to be positive semi-definite everywhere. But constrained problems are usually not smooth, or at least not everywhere. In this case convexity can be characterized without using derivatives, as follows. A function $V(\theta)$ is convex if, for every pair of points θ_1 and θ_2 , and any λ such that $0 \leq \lambda \leq 1$, it is always true that

$$\lambda V(\theta_1) + (1 - \lambda)V(\theta_2) \geq V(\lambda\theta_1 + [1 - \lambda]\theta_2) \quad (3.46)$$

The straight line joining any two points on the cost surface is never below the surface.

A *Quadratic Program* is an optimization problem of the form

$$\min_{\theta} \frac{1}{2}\theta^T \Phi\theta + \phi^T \theta \quad \text{subject to} \quad \Omega\theta \leq \omega$$

Here the objective function is $V(\theta) = \frac{1}{2}\theta^T \Phi\theta + \phi^T \theta$ and its Hessian is Φ . If there are no constraints this is clearly convex if $\Phi \geq 0$. (Without this condition there might be no minimum, since arbitrarily large negative values of V might be attainable.) Since the constraints are linear inequalities, the surfaces on which they are active are hyperplanes. So the constrained objective function can be visualized as a convex quadratic surface, parts of which have been cut off by a number of flat ‘faces’. It is intuitively clear that this constrained surface remains convex, although we shall not give a formal proof of this.

A *Linear Program* (or *LP*) is the special case of a *QP* when $\Phi = 0$, so that the objective function is linear rather than quadratic. It is also convex when Ω and ϕ are such that a minimum exists; in this case the minimum always occurs at a vertex (or possibly an edge). The constrained objective surface can be visualized as a convex object with flat faces. Such an object is called a *simplex*. There are standard algorithms for solving large *LP* problems, including the famous ‘simplex method’.

The literature of convex optimization, *LP* and *QP* problems is enormous. Good general books on optimization which include relevant material are [Fle87, GMW81]. A whole book on using convex optimization for control design (but which does not deal with predictive control) is [BB91].

Mini-Tutorial 3 Convex optimization, QP and LP problems.

3.2.2 Controller structure

So long as all the constraints are inactive, the solution of the predictive controller is exactly the same as in the unconstrained case. But if constraints become active then the controller becomes nonlinear and the structure shown in Figures 3.1 and 3.2 is lost. Figure 3.3 shows the controller structure in this case. The controller is nonlinear because the box labelled ‘Optimizer’ computes a nonlinear function of its inputs.

We can say a little more about the controller structure. Suppose a particular set of constraints is active. That is, in the *QP* problem (3.43)–(3.44), suppose that

$$\Omega_a\theta = \omega_a \quad (3.47)$$

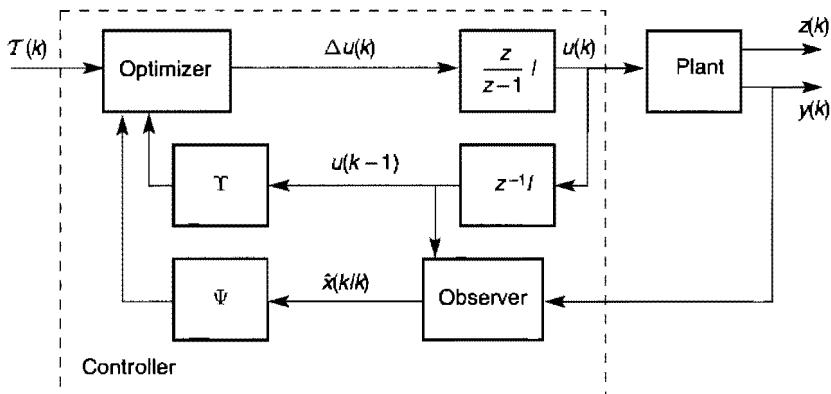


Figure 3.3 Structure of controller with constraints and state observer.

where Ω_a is made up of those rows of Ω which relate to the active constraints, and ω_a is made up of the corresponding elements of ω . If we knew before solving the problem that these would be the active constraints, and were therefore equality rather than inequality constraints, then we could (only in principle!) pose the optimization problem

$$\min_{\theta} \frac{1}{2} \theta^T \Phi \theta + \phi^T \theta \quad \text{subject to} \quad \Omega_a \theta = \omega_a \quad (3.48)$$

which could, by the theory of Lagrange multipliers, be solved by solving the problem

$$\min_{\theta, \lambda} L(\theta, \lambda) \quad (3.49)$$

where

$$L(\theta, \lambda) = \frac{1}{2} \theta^T \Phi \theta + \phi^T \theta + \lambda (\Omega_a \theta - \omega_a) \quad (3.50)$$

Now

$$\nabla_{\theta} L(\theta, \lambda) = \Phi \theta + \phi + \Omega_a^T \lambda \quad (3.51)$$

$$\nabla_{\lambda} L(\theta, \lambda) = \Omega_a \theta - \omega_a \quad (3.52)$$

or

$$\nabla L(\theta, \lambda) = \begin{bmatrix} \Phi & \Omega_a^T \\ \Omega_a & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \lambda \end{bmatrix} - \begin{bmatrix} -\phi \\ \omega_a \end{bmatrix} \quad (3.53)$$

Consequently the optimal solution would be obtained, by setting $\nabla L(\theta, \lambda) = 0$, as

$$\begin{bmatrix} \theta \\ \lambda \end{bmatrix}_{opt} = \begin{bmatrix} \Phi & \Omega_a^T \\ \Omega_a & 0 \end{bmatrix}^{-1} \begin{bmatrix} -\phi \\ \omega_a \end{bmatrix} \quad (3.54)$$

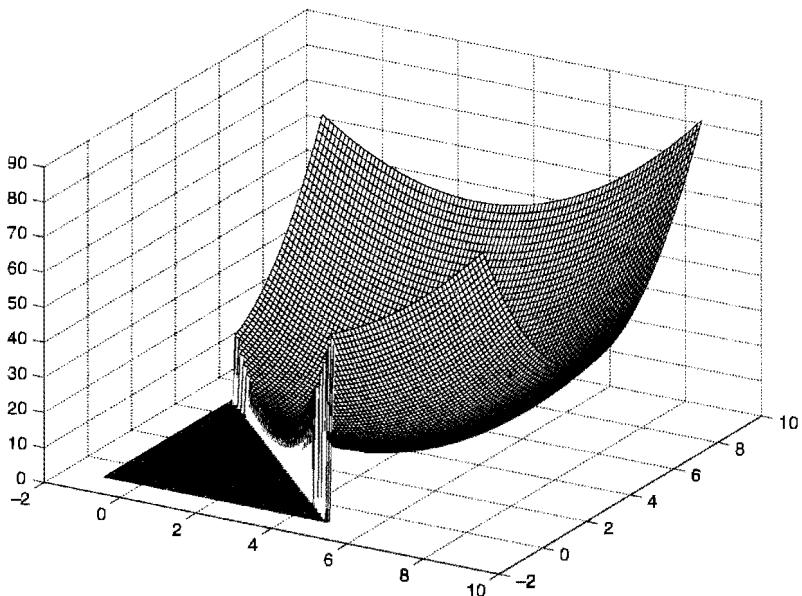


Figure 3.4 A quadratic cost surface and a linear inequality constraint: constraint inactive.

Now recall that

$$\Omega = \begin{bmatrix} F \\ \Gamma\Theta \\ W \end{bmatrix}$$

and $\Phi = \mathcal{H}/2$. Looking back at the definitions of F , Γ , Θ , W and \mathcal{H} , it will be seen that none of these depend on signals at time k . So the matrix which is being inverted here is fixed, so long as a fixed set of constraints is active. On the other hand, $\phi = -\mathcal{G} = -2\Theta^T Q \mathcal{E}(k)$ clearly does depend on the signals present at time k , and so does ω_a .

We can therefore conclude that the constrained predictive control law is a linear time-invariant control law, *so long as the set of active constraints is fixed*. A more intuitive explanation can be given on the basis of Figures 3.4 and 3.5. These show a quadratic cost function as a surface in the case when there are only two decision variables. In Figure 3.4 a linear inequality constraint restricts the decision variables so that only part of the cost surface is ‘attainable’, but the unconstrained minimum is in the feasible region, so the optimal solution is at the unconstrained minimum. In Figure 3.5 the constraint is active, so that the optimal solution is on the intersection of the cost surface and the inequality constraint. It can be seen that the optimal solution is still the global minimum of a quadratic surface, but one of lower dimension — in this case, the minimum of a quadratic curve. It is the fact that a quadratic problem is still being solved that leads to the controller being linear.

In practice the set of active constraints changes, so we have the picture of the control law as consisting of a number of linear controllers, each with a structure similar to that

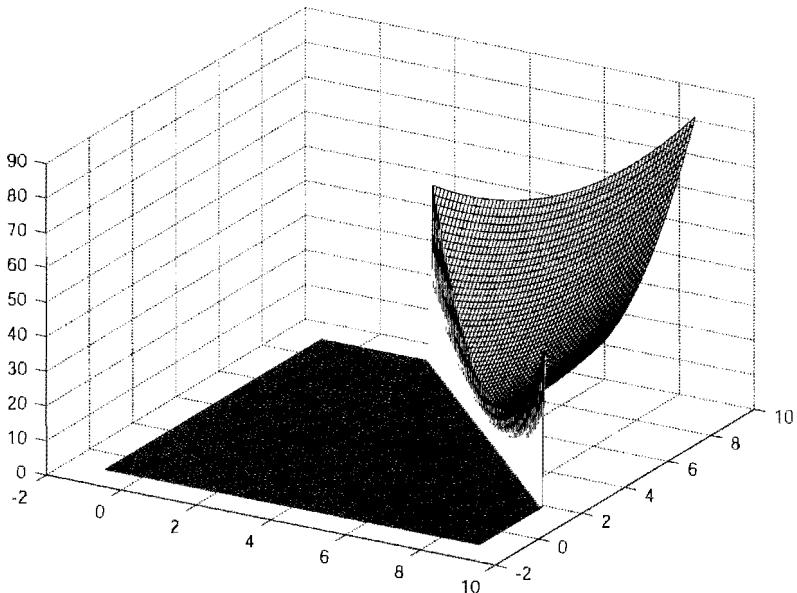


Figure 3.5 A quadratic cost surface and a linear inequality constraint: constraint active.

shown in Figure 3.2, and switching between them. If we could be sure that changes of active constraint sets occurred rarely enough — and there is no reason to suppose that this would be the case in general — we might be able to exploit this structure to perform some analysis of the constrained controller.

In [BMDP, BMDP99] a promising attempt is made not only to use the piecewise-linear nature of the controller as a basis for analysis, but also as a way of computing the optimal solution efficiently for small problems. In this context ‘small’ means that the number of constraints is such that the number of possible active constraint sets is manageable, since a separate solution is pre-computed explicitly for each such set. Note that the number of possible active constraint sets can be extremely large. If there are q constraints in the problem formulation (Ω has q rows) then there are 2^q possible sets of active constraints. Analysis of specific problems can often reduce the size of this set to some extent — for example, lower and upper limits on variable values cannot be active simultaneously.

The approach taken by Bemporad *et al* in [BMDP] and [BMDP99] is based on the observation that in the MPC problem, the inequality constraint (3.44) takes the form:

$$\Omega\theta \leq \omega(\hat{x}) \quad (3.55)$$

The vector on the right-hand side of the inequality depends on the (estimate of the) current state, which can be thought of as a set of parameters of the QP problem. Techniques of *multiparametric programming* are then employed to analyze the set of solutions. One new result which is obtained by this means is that the control law is *continuous* in \hat{x} . Algorithms are also given for determining efficiently the boundaries (in the state-space) at which changes from one piecewise-linear control law to another

occur. The idea, then, is that for small problems, in which the state-space is divided up into a manageable small number of (convex) pieces, one could pre-compute (off-line) the control law that should be applied in each piece, and then the MPC algorithm would consist simply of reading the appropriate gain matrix from a table look-up, depending on the current state estimate. This should be particularly useful in high-bandwidth applications, when high control update rates are required.

Although this idea is not feasible for applications in which the number of constraints is large ($q > 10$, say), it may be possible even in large problems to compute the solution by conventional means (as described in the next subsection) when a region of the state space is entered for the first time, but then to store it in a growing database, as the state space is explored, so that if the region is revisited later the solution does not need to be recomputed. Many variations of this can be easily imagined.

3.3 Solving QP problems

In this section we shall consider the general QP problem:

$$\min_{\theta} \frac{1}{2} \theta^T \Phi \theta + \phi^T \theta \quad (\Phi = \Phi^T \geq 0) \quad (3.56)$$

subject to

$$H\theta = h \quad (3.57)$$

and

$$\Omega\theta \leq \omega \quad (3.58)$$

This is the same as the QP problem introduced earlier, except that the equality constraints $H\theta = h$ have been identified explicitly. We shall assume that it is not feasible to pre-compute the solutions in the manner described at the end of the previous subsection.

Two approaches to solving the QP problem appear to offer the best performance: *Active Set* methods [Fle87], and the more recent *Interior Point* methods [NN94, Wri97, RTV97]. It is tempting to assume that the more recent methods should offer the best performance, but Wright [Wri96] argues that for predictive control this issue is still open. He points out that the predictive control problem has a lot of special structure, and that the main performance gains are to be obtained by exploiting this structure, whichever approach is used. The special structure is primarily due to two features:

1. With a particular ordering of the variables, the QP problem which arises in predictive control is ‘sparse’.
2. Most of the time — in the absence of large disturbances, or of set-point changes, etc. — a very good initial guess of the solution is available, from the solution computed at the previous step.

In this section, both the Active Set and the Interior Point methods will be described very briefly, and some of the points at which the structure of the predictive control problem can be exploited will be identified. The presentation is based on [Fle87] and [Wri96].

As a result of the QP problem being convex, the necessary and sufficient conditions for θ to be the global optimum are given by the *Karush–Kuhn–Tucker* (or *KKT*) conditions [Fle87, Wri97]: there must exist vectors (Lagrange multipliers) $\lambda \geq 0$ and ζ , and a vector $t \geq 0$, such that

$$\Phi\theta + H^T \zeta + \Omega^T \lambda = -\phi \quad (3.59)$$

$$-H\theta = -h \quad (3.60)$$

$$-\Omega\theta - t = -\omega \quad (3.61)$$

$$t^T \lambda = 0 \quad (3.62)$$

3.3.1 Active Set methods

The *Active Set* method assumes that a feasible solution is available. (We shall consider later how that might be found.) For such a solution, the equality constraints (3.57) are of course satisfied, and some subset of the inequality constraints (3.58) is active, namely the constraints hold with equality for this subset. This subset is called the *active set*; it is possible for it to be empty, if none of the inequality constraints is active at the current feasible solution. Using the same notation as earlier, we use the subscript a to denote those rows of (3.58) which are in the active set, so that

$$\Omega_a \theta = \omega_a$$

Suppose that at the r th iteration we have the feasible solution θ_r . The Active Set method then finds an improved solution $\theta_r + \Delta\theta$ which minimizes the cost (3.56) while satisfying the equality constraints $H\theta = h$ and $\Omega_a \theta_r = \omega_a$, without worrying about the inactive inequality constraints. If this new solution is feasible, that is if $\Omega(\theta_r + \Delta\theta) \leq \omega$, then it is accepted as the next iteration: $\theta_{r+1} = \theta_r + \Delta\theta$. If it is not feasible, then a line-search is made in the direction $\Delta\theta$ to locate the point at which feasibility is lost — namely the point at which one of the inactive inequality constraints becomes active. The solution at this point is accepted as the next iteration: $\theta_{r+1} = \theta_r + \alpha_r \Delta\theta$, where $0 < \alpha_r < 1$, and the newly active constraint is added to the active set.

It remains to decide whether this new solution is already the global optimum of the QP problem, or whether further improvement can be obtained. This decision can be made by checking whether the *KKT* conditions (3.59)–(3.62) are satisfied at θ_{r+1} . Note that the *complementarity condition* (3.62) implies that those elements of λ corresponding to inactive constraints are zero, since the corresponding elements of t are positive. So only those elements of λ corresponding to active constraints need to be evaluated; if they are all nonnegative then the global solution has been found, because of the condition $\lambda \geq 0$; otherwise further iteration is needed. So suppose that $\lambda_q < 0$. Since λ_q is a Lagrange multiplier corresponding to the q th inequality constraint, which is active (by assumption), its negative value indicates that the cost function could be reduced by making that constraint inactive, namely moving to a solution θ for which $\Omega_q \theta < \omega_q$. Thus the q th constraint is removed from the active set. (If more than one element of

λ is negative, then the constraint corresponding to the most negative one is removed from the active set.) So now a new active set has been selected, and the whole process is repeated, replacing θ_r by θ_{r+1} .

Note that $V(\theta_{r+1}) < V(\theta_r)$, where $V(\theta) = \frac{1}{2}\theta^T\Phi\theta + \phi^T\theta$ is the cost function, so that this iterative process is guaranteed to terminate at the global optimum, because of the convexity of the QP problem. A potential advantage of the Active Set method over other methods, for predictive control, is that the iterations remain feasible once an initial feasible solution has been found. In many predictive control problems it is the feasibility of the solution, rather than exact optimality, that is most important. So if the QP solver has failed to terminate by the time the next control move is required, using the latest available iteration may be an adequate solution, and in many cases it is likely to be nearly as effective as the true optimal solution. This may arise, for example, when an unusually large disturbance occurs, so that the most important thing is to keep the plant within a safe operating region — that is, find a feasible solution if the constraints delineate the boundary of this region; in these circumstances, a good initial guess of the solution is not likely to be available from earlier steps.

Now we consider in more detail the minimization of the cost function at the r th iteration. The new cost is

$$V(\theta_r + \Delta\theta) = \frac{1}{2}(\theta_r + \Delta\theta)^T\Phi(\theta_r + \Delta\theta) + \phi^T(\theta_r + \Delta\theta) \quad (3.63)$$

$$= V(\theta_r) + \frac{1}{2}\Delta\theta^T\Phi\Delta\theta + (\phi^T + \theta_r^T\Phi)\Delta\theta \quad (3.64)$$

so the minimization problem to be solved can also be stated as

$$\min_{\Delta\theta} \frac{1}{2}\Delta\theta^T\Phi\Delta\theta + \phi_r^T\Delta\theta \quad (3.65)$$

subject to

$$H\Delta\theta = 0 \quad \text{and} \quad \Omega_a\Delta\theta = 0 \quad (3.66)$$

where $\phi_r = \phi + \Phi\theta_r$. This is a convex QP problem, but with equality constraints only.

One way of solving such an equality-constrained problem is by the method of Lagrange multipliers — that is, by using the KKT conditions for this sub-problem — but since there are only equality constraints, only (3.59) and (3.60) are needed from the KKT conditions. However, it is convenient to separate out the vector ζ into those components corresponding to $H\Delta\theta = 0$ and those corresponding to $\Omega_a\Delta\theta = 0$. We shall call these $\Delta\zeta$ and $\Delta\lambda$, respectively. Thus the KKT conditions for this sub-problem are: there must exist vectors $\Delta\zeta$ and $\Delta\lambda$ (unrestricted as to sign), such that

$$\Phi\Delta\theta + H^T\Delta\zeta + \Omega_a^T\Delta\lambda = -\phi_r \quad (3.67)$$

$$-H\Delta\theta = 0 \quad (3.68)$$

$$-\Omega_a\Delta\theta = 0 \quad (3.69)$$

These equations can be assembled into the matrix equation:

$$\begin{bmatrix} \Phi & H^T & \Omega_a^T \\ H & 0 & 0 \\ \Omega_a & 0 & 0 \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta\zeta \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} -\phi_r \\ 0 \\ 0 \end{bmatrix} \quad (3.70)$$

This equation is solved by some version of Gaussian elimination, which involves factorizing the matrix on the left-hand side into the product of a lower-triangular and an upper-triangular matrix:

$$\begin{bmatrix} \Phi & H^T & \Omega_a^T \\ H & 0 & 0 \\ \Omega_a & 0 & 0 \end{bmatrix} = LU \quad (3.71)$$

Once this factorization has been obtained, the solution is obtained very quickly, since it requires the solution (for η) of

$$L\eta = \begin{bmatrix} -\phi_r \\ 0 \\ 0 \end{bmatrix}$$

which is done by forward substitution, followed by the solution of

$$U \begin{bmatrix} \Delta\theta \\ \Delta\xi \\ \Delta\lambda \end{bmatrix} = \eta$$

which is done by back-substitution. So the speed of the Active Set method is dominated by the speed with which this factorization can be performed. At first sight it seems obvious that the symmetric structure of the matrix should be exploited, and there are algorithms for efficient LU factorization of symmetric matrices [GL89]. But as pointed out in [Wri96], more advantage may be obtained by rearranging the variables in $[\Delta\theta^T, \Delta\xi^T, \Delta\lambda^T]^T$, so that the matrix becomes *banded*, if that is possible. The structure of the predictive control problem allows this to be done, though at the cost of introducing many more variables into the problem.

Taking the predictive control QP problem to be defined by (3.41) and (3.42), we see that the corresponding values for Φ and Ω are

$$\Phi = \mathcal{H} = \Theta^T \mathcal{Q} \Theta \quad (3.72)$$

$$\Omega = \begin{bmatrix} F \\ \Gamma\Theta \\ W \end{bmatrix} \quad (3.73)$$

and H does not exist. Although \mathcal{Q} is block-diagonal, Θ is nearly full, so that Φ does not have any particular structure which can be exploited, apart from symmetry.

But an alternative formulation is obtained by not eliminating the predicted states $\hat{x}(k+j|k)$ from the problem, and leaving them as variables to be found by the QP solver. That is, we still minimize the cost function (3.2), but instead of replacing $\mathcal{Z}(k)$ by using (3.5), we use (2.70) and impose the *equality* constraints

$$\hat{x}(k+j+1|k) = A\hat{x}(k+j|k) + B\hat{u}(k+j|k) \quad \text{for } j = 0, \dots, H_p - 1 \quad (3.74)$$

and

$$\hat{u}(k+j+1|k) = \hat{u}(k+j|k) + \Delta\hat{u}(k+j+1|k) \quad \text{for } j = 0, \dots, H_u - 1 \quad (3.75)$$

(with $\hat{u}(k-1|k) = u(k-1)$). The minimization is then performed over the variables $\Delta\hat{u}(k+j|k)$ and $\hat{u}(k+j|k)$ for $(j = 0, \dots, H_u - 1)$, and $\hat{x}(k+j|k)$ for $j = 1, \dots, H_p$. This introduces $\ell H_u + n H_p$ additional variables into the problem (where ℓ is the number of inputs and n is the state dimension), which looks like a bad idea. But the potential benefit is that now the matrix which has to be factorized in the Active Set method can be banded. If the variables are ordered in the following way:

$$\theta = \begin{bmatrix} \hat{u}(k|k) \\ \Delta\hat{u}(k|k) \\ \hat{x}(k+1|k) \\ \hat{u}(k+1|k) \\ \Delta\hat{u}(k+1|k) \\ \hat{x}(k+2|k) \\ \vdots \\ \hat{u}(k+H_u-1|k) \\ \Delta\hat{u}(k+H_u-1|k) \\ \hat{x}(k+H_u|k) \\ \hat{x}(k+H_u+1|k) \\ \vdots \\ \hat{x}(k+H_p|k) \end{bmatrix} \quad (3.76)$$

then we have (assuming $H_w = 1$ and $H_u < H_p$ for simplicity):

$$\Phi = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & R(0) & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \bar{Q}(1) & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & R(1) & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \bar{Q}(2) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \bar{Q}(H_p-1) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \bar{Q}(H_p) \end{bmatrix} \quad (3.77)$$

$$H = \begin{bmatrix} B & 0 & -I & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ I & -I & 0 & -I & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & A & B & 0 & -I & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & I & -I & 0 & -I & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & A & -I \end{bmatrix} \quad (3.78)$$

where $\bar{Q}(i) = C_z^T Q(i) C_z$. The point here is that both Φ and H have a banded structure (i.e. all their non-zero entries are relatively close to the principal diagonal), as does the matrix Ω associated with the inequality constraints. Now this does not itself make the matrix to be factorized in (3.71) banded, but this can also be achieved by reordering the order of the variables. The key is to keep all the variables associated with predicted time

$k + j$ grouped together, including the relevant elements of $\Delta\zeta$ and $\Delta\lambda$. The bandedness can be exploited to speed up the factorization. Furthermore, the matrix to be factorized at the $(r + 1)$ th iteration of the active set method is only a little different from the one factorized at the r th iteration. Wright [Wri96] proposes a method which takes advantage of both of these features.

Is it worthwhile obtaining bandedness in this way, since it involves introducing additional variables into the QP problem? According to [Wri96], the time required to factor a banded matrix with half-bandwidth b and total dimension $N(b + 1)$ is $O(N(b + 1)^3)$, compared with $O(N^3(b + 1)^3)$ for a dense matrix of the same size. If we assume that $H_u = H_p$ (so that the bandwidth is the same along the whole diagonal of the matrix) and that half of the inequality constraints are active, then the half-bandwidth in the scheme just outlined is approximately $2n + 3\ell + v/2$, where v is the number of inequality constraints (on inputs, input changes, and outputs), and the total size of the matrix to be factorized is $H_p(2n + 3\ell + v/2)$. On the other hand, for the original scheme, using (3.41) and (3.42), the matrix is dense, and its size is approximately $(\ell + v/2)H_p$. So, roughly speaking, it is worth considering the banded scheme if

$$H_p^2 \left(\ell + \frac{v}{2}\right)^3 > \left(2n + 3\ell + \frac{v}{2}\right)^3 \quad (3.79)$$

For some typical numbers this works out as follows:

Typical process application $n = 200$, $\ell = 20$, $v = 40$

$$\ell + v/2 = 40, 2n + 3\ell + v/2 = 480$$

'Banded' scheme looks worthwhile if $H_p > 41$.

Typical aerospace application $n = 12$, $\ell = 3$, $v = 4$

$$\ell + v/2 = 5, 2n + 3\ell + v/2 = 35$$

'Banded' scheme looks worthwhile if $H_p > 18$.

These examples suggest that the banded scheme is likely to be worthwhile in many cases. As we will see in Chapters 6 and 8, long prediction horizons help to ensure closed-loop stability, and to minimize the risk of driving the plant into 'dead-ends' from which no feasible solution is possible; thus there are good reasons for making H_p as large as computation speed will allow. Note that if $H_u \ll H_p$ then the original 'dense' scheme will be more favoured over the 'banded' scheme than inequality (3.79) suggests. The comparison should be done more carefully for any particular application, since it will be affected by factors such as how many constraints are likely to be active typically (which it may be possible to predict for particular applications). Also, the comparison as made above disregards the difference in the difficulty of finding an initial feasible solution. In practice, actual performance comparisons of the two schemes may be required in order to make the correct decision.

One more point remains to be dealt with: how to find an initial feasible solution. This is needed in order to get the iterations of the active set method started. The basic idea is the following. Suppose that we have an infeasible solution θ_0 , which satisfies the equality constraints $H\theta_0 = h$, and that a subset of the inequality constraints is satisfied, so that $\Omega_s\theta_0 \leq \omega_s$, and the remaining inequality constraints are not satisfied:

$\Omega_u \theta_0 > \omega_u$. Then a solution θ is sought for the *linear programming* (LP) problem:

$$\min_{\theta} \sum_j (\omega_u^j - \Omega_u^j \theta) \quad (3.80)$$

subject to

$$H\theta = h \quad (3.81)$$

and

$$\Omega_s \theta \leq \omega_s \quad (3.82)$$

where Ω_u^j denotes the j th row of Ω_u and ω_u^j denotes the j th element of ω_u . The usual simplex algorithm for solving LP problems can be used, which searches among the vertices of the feasible region. However, the problem formulation is changed at every new vertex visited, because the selection of rows in Ω_s and Ω_u (and of corresponding elements in ω_s and ω_u) changes at every vertex. Eventually either all the inequality constraints are satisfied, in which case a feasible solution has been found, or the cost (3.80) is still positive and the Lagrange multipliers indicate that it cannot be improved, in which case the problem is infeasible. Problems with this basic idea are:

1. The feasible solution which is found is arbitrary, and can be very far from the optimal solution of the QP problem.
2. The simplex algorithm always requires as many active constraints as the dimension of θ . So it may not be possible to use the solution of the QP problem solved at the previous step as an initial guess (*hot starting*), because that may have a different number of active constraints.

These problems can be overcome by adding a small multiple of the QP cost to the cost function (3.80). This ‘biases’ the initial feasible solution towards the optimal solution of the QP problem, hopefully leading to fewer iterations of the Active Set method, and it also changes the initial feasibility problem from an LP problem into a QP problem, which can be started with any number of active constraints.

3.3.2 Interior point methods

In the last twenty years or so, a rival family of algorithms for solving convex optimization problems has emerged. These are the so-called *interior point* methods [NN94, Wri97, RTV97]. They first came to prominence with Kamarkar’s algorithm for solving LP problems, the first serious rival to the simplex algorithm, and one which was potentially dramatically faster for large problems. Much development since then has led to versions of interior point methods which are particularly effective for solving QP problems. The attraction of these new methods is that their computational complexity (number of iterations, time to complete, . . .) is no worse than some polynomial function of parameters such as the number of constraints or the number of variables, whereas

the complexity of other known approaches, including Active Set methods, can be exponential in these parameters in the worst case. We remark that the availability of interior point methods for more general convex optimization problems has led to the intense current interest in *linear matrix inequalities (LMIs)* for solving control and other ‘systems’ problems [BGFB94] — we shall make use of them in Section 8.4.

In this subsection we shall give a brief indication of how interior point methods work. As with active set methods, one can apply interior point methods to the predictive control problem ‘naively’, that is regarding it as just a standard QP problem, or one can exploit the particular structure inherent in predictive control problems. The second alternative has been investigated in detail by Wright *et al* [Wri96, RWR98].

Whereas the main iterations of the active set methods search among points on the boundary of the feasible region, early interior point methods searched among points in the ‘interior’ of this region; in other words, the iterates were always feasible solutions of the optimization problem being solved. It turned out that this was not a particularly efficient strategy, among other things requiring an initial feasible point to be found. The iterates in more recent versions are not feasible — until the end of the search, typically — but they are still away from the boundary of the feasible region, and in that sense ‘interior’ (and they are in the interior of a certain positive orthant).

Suppose that a function $V(x)$ is to be minimized over x , subject to the constraint $Ax \leq b$. One way of looking at interior point methods is to consider the minimization of the function

$$f(x, \gamma) = \gamma V(x) - \underbrace{\sum_i \log(b_i - a_i^T x)}_{\text{barrier function}} \quad (3.83)$$

where a_i^T is the i th row of A , b_i is the i th element of the vector b , and γ is some positive scalar. If the minimum of $f(x, \gamma)$ is being sought by some search strategy, starting in the feasible region, then the logarithmic barrier function prevents the search leaving the feasible region, since it becomes infinite on the boundary of the region.¹ Let x_γ be the minimizer of $f(x, \gamma)$, and let x^* be the solution of the original problem, namely the minimizer of $V(x)$ which satisfies the constraints. x_0 is known as the *analytic centre* of the constraints; if the feasible region is not empty then x_0 lies in its interior, and does not depend on the objective function $V(x)$ at all. On the other hand, $x_\gamma \rightarrow x^*$ as $\gamma \rightarrow \infty$. The underlying idea is that if an initial (feasible) solution is found in the vicinity of x_0 , then it can be continuously improved by increasing γ and minimizing $f(x, \gamma)$, until $\|x^* - x_\gamma\|$ is sufficiently small. The path traced out by x_γ is known as the *central path*.

Example 3.1

Consider the following QP problem, taken from Fletcher [Fle87, Problem 10.4]:

$$\min_{\theta} V(\theta) = \theta_1^2 - \theta_1\theta_2 + \theta_2^2 - 3\theta_1$$

¹ Other barrier functions are possible.

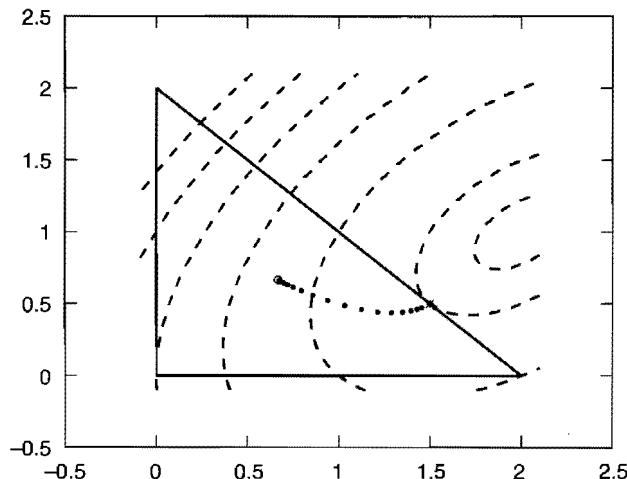


Figure 3.6 Basic interior point method for Example 3.1.

subject to

$$\theta_1 \geq 0$$

$$\theta_2 \geq 0$$

$$\theta_1 + \theta_2 \leq 2$$

Figure 3.6 shows the feasible region — the interior of the triangle — and some contours of $V(\theta)$. It is clear that the optimal solution is at $(3/2, 1/2)$, where the $V(\theta) = -11/4$ contour is tangential to the constraint $\theta_1 + \theta_2 \leq 2$. The small circle in the middle of the triangle shows the analytic centre, which is at $(2/3, 2/3)$, and the dots show the central path traced out as γ increases from 0 to 100. To get the sequence of solutions shown, the γ values were spaced logarithmically between 0.01 and 100, using 20 values (*MATLAB* command: `logspace(-2, 2, 20)`).

Examining the neighbourhood of the optimal solution in Figure 3.6 reveals a drawback of naive implementations of the interior point approach: one of the solutions seems to be at the constraint, but a little way away from the optimum — in fact, it is at $(1.5266, 0.4742)$, which is just infeasible. This is the solution obtained with the last few values of γ , and it is clearly incorrect. The reason is that with these values of γ the solution is very close to the constraint boundary, where the logarithmic barrier function is increasing extremely quickly, and as a result the optimization problem becomes very ill-conditioned. The solution which is obtained then depends strongly on the particular algorithm used; the solution shown in the figure was obtained using the very unsophisticated ‘direct-search’ approach implemented by *MATLAB*’s *Optimization Toolbox* function `fmins`. The major reason why good interior point algorithms are relatively complicated is that they take elaborate precautions to deal with this ill-conditioning.

At present the most effective interior-point algorithms are the so-called *primal-dual* methods [Wri97]. These find solutions to convex optimization problems and their dual problems simultaneously. Recall the KKT conditions (3.59)–(3.62) for the QP problem. If the complementarity condition (3.62) is relaxed to

$$t^T \lambda = \frac{1}{\gamma} > 0 \quad (3.84)$$

then the equations (3.59), (3.60), (3.61), (3.84) define a central path which converges to the optimal solution of the QP problem, and simultaneously of its dual, as $\gamma \rightarrow \infty$. The basic idea of primal-dual methods is to alternately take steps towards the central path (γ fixed), and ‘parallel’ to the central path (γ increased). For both kinds of step the search direction is found by solving an equation of the form

$$\begin{bmatrix} \Phi & H^T & \Omega^T \\ H & 0 & 0 \\ \Omega & 0 & \Lambda^{-1}T \end{bmatrix} \begin{bmatrix} \Delta\theta \\ \Delta\zeta \\ \Delta\lambda \end{bmatrix} = \begin{bmatrix} r_\phi \\ r_H \\ r_\Omega \end{bmatrix} \quad (3.85)$$

where $\Lambda = \text{diag}\{\lambda_i\}$ and $T = \text{diag}\{t_i\}$. The vector on the right-hand side of (3.85) varies, depending on which kind of step is being performed, and on which particular variant of primal-dual algorithms is being implemented. It will be seen that the matrix on the left-hand side of this equation has the same form as the matrix which appears in (3.70). As in the Active Set method, the speed of solution is dominated by the speed with which equation (3.85) can be solved. Therefore much the same considerations apply as were discussed in the previous section. Once again, it is advantageous to order the variables so that the matrix in (3.85) is banded. For further details see [RWR98, Wri97].

3.4 Softening the constraints

A serious problem which can occur with the predictive control problem as we have formulated it so far is that the optimizer may be faced with an infeasible problem. This can happen because an unexpectedly large disturbance has occurred, so there is really no way in which the plant can be kept within the specified constraints. Or it can happen because the real plant behaves differently from the internal model; the predictive controller may attribute differences between the plant and the model behaviours to large disturbances, and if these keep growing then it can eventually decide — erroneously — that it does not have enough control authority to keep the plant within the constraints. There are many ways in which the predictive control problem can become infeasible, and most of them are difficult to anticipate.

Because of this, it is essential to have a strategy for dealing with the possibility of infeasibility. Standard algorithms for solving QP problems just give up, and output a message such as ‘Infeasible problem’. Clearly this is unacceptable behaviour for an on-line controller. Various possibilities exist, ranging from *ad hoc* measures such as outputting the same control signal as in the previous step, or (better) the control signal computed as $\hat{u}(k+2|k)$ in the previous step, to sophisticated strategies of ‘constraint management’, in which one tries to relax the least-important constraints in an attempt to regain feasibility — see Section 10.2.

One systematic strategy for dealing with infeasibility is to ‘soften’ the constraints. That is, rather than regard the constraints as ‘hard’ boundaries which can never be crossed, to allow them to be crossed occasionally, but only if really necessary.

There is an important distinction between input and output constraints. Usually input constraints really are ‘hard’ constraints, and there is no way in which they can be softened; valves, control surfaces, and other actuators have limited ranges of action, and limited slew rates. Once these have been reached there is no way of exceeding them, except for fitting more powerful actuators. Therefore input constraints are usually not softened.

A straightforward way of softening output constraints is to add new variables, so-called ‘slack variables’, which are defined in such a way that they are non-zero only if the constraints are violated. Then their non-zero values are very heavily penalized in the cost function, so that the optimizer has a strong incentive to keep them at zero if possible.

A first way of doing this is to add a quadratic penalty for constraint violations [OB94]. The optimization problem (3.43)–(3.44) is modified to

$$\min_{\theta, \epsilon} \frac{1}{2} \theta^T \Phi \theta + \phi^T \theta + \rho \|\epsilon\|^2 \quad (3.86)$$

subject to

$$\begin{cases} \Omega \theta \leq \omega + \epsilon \\ \epsilon \geq 0 \end{cases} \quad (3.87)$$

where ϵ is a nonnegative vector of the same dimension as ω , and ρ is a nonnegative scalar. This is still a QP problem, though with a larger number of variables. It is clear that a similar modification can be made if some of the constraints are to be retained as hard constraints.

If $\rho = 0$ then one has the completely unconstrained problem, and as $\rho \rightarrow \infty$ so one recovers the solution to the hard-constrained problem. But a drawback of a quadratic penalty on constraint violations is that, if the constraints are active, then for all finite values of ρ this formulation will result in them being violated to some extent (decreasing as ρ increases), even if such violation is not necessary.

Alternatives are to penalize the 1-norm (sum of violations) or the ∞ -norm (maximum violation) of the constraint violations:

$$\min_{\theta, \epsilon} \left(\frac{1}{2} \theta^T \Phi \theta + \phi^T \theta + \rho \|\epsilon\|_1 \right) = \min_{\theta, \epsilon} \left(\frac{1}{2} \theta^T \Phi \theta + \phi^T \theta + \rho \sum_j \epsilon_j \right) \quad (3.88)$$

subject to

$$\begin{cases} \Omega \theta \leq \omega + \epsilon \\ \epsilon \geq 0 \end{cases}$$

or

$$\min_{\theta, \epsilon} \left(\frac{1}{2} \theta^T \Phi \theta + \phi^T \theta + \rho \|\epsilon\|_\infty \right) = \min_{\theta, \epsilon} \left(\frac{1}{2} \theta^T \Phi \theta + \phi^T \theta + \rho \max_j \epsilon_j \right) \quad (3.89)$$

subject to

$$\begin{cases} \Omega\theta \leq \omega + \epsilon \\ \epsilon \geq 0 \end{cases}$$

Note that the second alternative can be posed more efficiently as

$$\min_{\theta, \epsilon} \left(\frac{1}{2} \theta^T \Phi \theta + \phi^T \theta + \rho \epsilon \right) \quad (3.90)$$

subject to

$$\begin{cases} \Omega\theta \leq \omega + \epsilon \mathbf{1} \\ \epsilon \geq 0 \end{cases} \quad (3.91)$$

where ϵ is a scalar, and $\mathbf{1}$ is a vector, each element of which is 1. In this form only one slack variable is introduced, which gives, in general, a much faster algorithm. Penalizing the 1-norm of constraint violations requires a separate slack variable for every constraint, at every point of the prediction horizon for which the constraint is enforced. The number of slack variables can (greatly) exceed the number of decision variables in the original ‘hard-constrained’ problem. Both of these ‘soft’ optimization problems are again equivalent to QP problems.

It can be shown that, with ρ large enough, using either the 1-norm or the ∞ -norm penalty on constraint violations gives an ‘exact penalty’ method, which means that constraint violations will not occur unless there is no feasible solution to the original ‘hard’ problem. That is, the same solution will be obtained as with a ‘hard’ formulation, if a feasible solution exists. The reason for this difference in behaviour between these formulations and the quadratic penalty formulation is that, starting from the true constrained solution θ^* , there exists a move to an infeasible solution $\theta^* + \epsilon d$, for some vector d , which gives a reduction in the cost function of $o(\epsilon)$. Since the penalty for the violation is $o(\epsilon^2)$ then the violation gives a smaller value of the ‘softened’ cost function (3.86), for small enough ϵ . But in the cost functions (3.88) and (3.90) the penalty is also $o(\epsilon)$, so if the coefficient ρ is large enough then such a move results in an increase in the cost function. ‘Large enough’ means ‘larger than $\|d(\frac{1}{2}\theta^{*T}\Phi\theta^* + \phi^T\theta^*)/d\omega\|_\infty$ at the optimal solution’; this can be related to the Lagrange multipliers of the original ‘hard’ problem at the optimal solution — for details see [Fle87]. Unfortunately, in the context of predictive control, the Lagrange multipliers will depend on the current state, so it is not easy to decide just how large ρ should be [OB94, SR96b]. A contribution towards solving this problem is made in [KM00b]. In [SR96b] a mixture of linear and quadratic penalties on constraint violations is proposed.

The following example shows that it can be difficult to predict the circumstances in which infeasibility is encountered, and hence the virtual necessity of softening constraints. In this example the infeasibility arises as a result of modelling error.

Example 3.2

Consider again the linearized Citation aircraft, as introduced in Section 2.7. In that section it has already been shown that some error in the model can be tolerated without

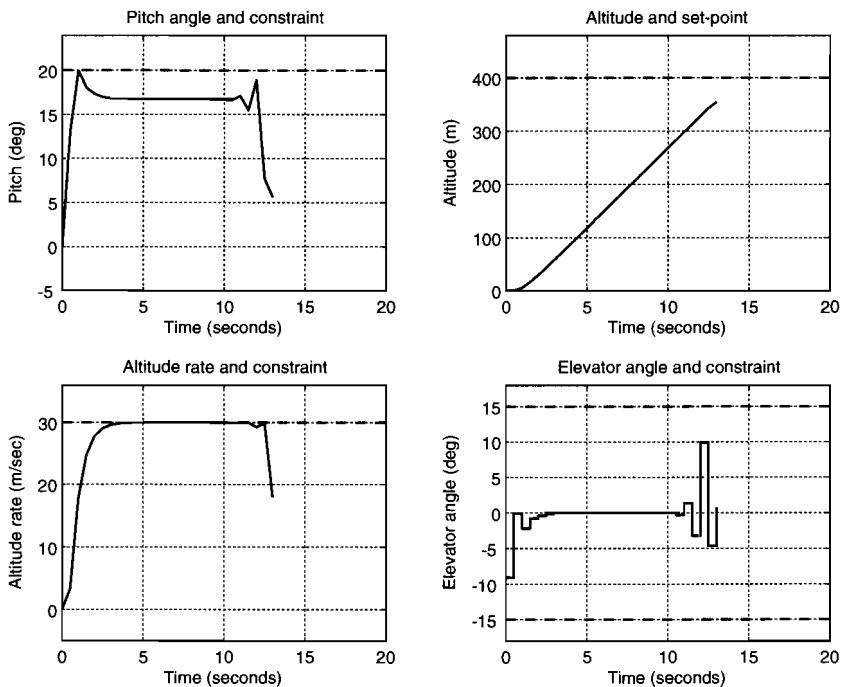


Figure 3.7 Plant–model error and hard constraints result in infeasibility.

serious difficulty. But this is not always true. Suppose that the gain from elevator to altitude rate is *over*-estimated by 20%, and that a constraint is imposed on the altitude, in an attempt to ensure a very small overshoot. If the altitude is to be increased by 400 m, as in Figure 2.8, the constraint

$$\hat{y}_2(k+i|k) < 405 \quad (3.92)$$

is imposed for all k and i , and all other parameters are the same as in Section 2.7, Figure 2.8, then the result is shown in Figure 3.7. Thirteen seconds into the manoeuvre, the predictive controller is faced with an infeasible problem, and stops working.

Figure 3.8 shows the results when the output constraints are softened, using a 1-norm penalty function with weight $\rho = 10^4$. Note that the pitch angle constraint is violated briefly in the early part of the manoeuvre, although this did not happen with hard output constraints — see Figure 3.7; this suggests that the penalty function is not exact, despite the large value of ρ . Another notable feature is that the altitude constraint of 400 m is not violated — in fact there is no overshoot, and the altitude never exceeds 400 m. This may seem inconsistent with the fact that the hard-constrained problem became infeasible after 13 seconds. The explanation is that the figure shows the actual altitude of the aircraft, whereas the infeasibility arose from the controller's predictions of the altitude, which were based on an inaccurate model. This phenomenon, of the controller predicting a constraint violation which does not actually occur, is quite common. The converse phenomenon can be seen in Figure 2.11; there the altitude rate constraint is

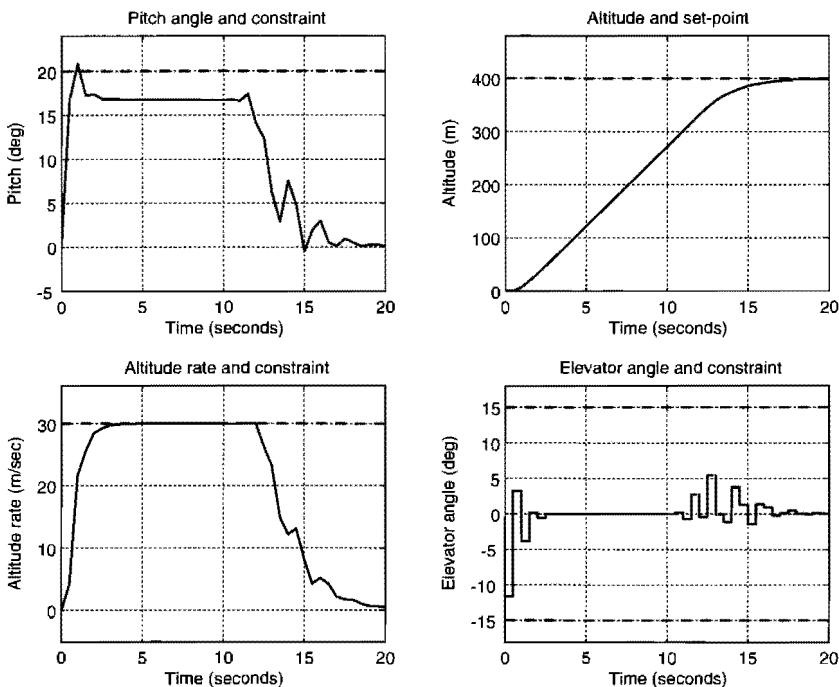


Figure 3.8 Soft output constraints restore feasibility. 1-norm penalty with $\rho = 10^4$.

violated briefly, yet no infeasibility is detected by the controller. Again, the explanation is that the controller's predictions were inaccurate and did not indicate an inevitable constraint violation.

In this example, the ‘hard-constrained’ QP problem to be solved at each step has three decision variables, namely $\hat{u}(k+i|k)$ for $i = 0, 1, 2$, since $H_u = 3$ and there is only one control input. When the output constraints are softened using a 1-norm penalty, four slack variables are introduced at each point in the prediction horizon — one for each constraint: maximum pitch angle, minimum pitch angle, maximum altitude rate, and maximum altitude. Since the constraints are enforced at each point of the prediction horizon, which has length $H_p = 10$, there are 40 slack variables altogether. So the total number of decision variables has increased from 3 to 43, as a result of softening the constraints in this way. The average time required to solve and simulate one step on a 333 MHz Pentium II processor was 0.81 sec. (Most of this time is taken by the optimization, since simulation of a linear model is very fast.) Solving the same problem using an ∞ -norm penalty, again with $\rho = 10^4$, gave virtually identical results, but required only 0.15 sec per step. The average time required per step for the hard-constrained problem was 0.04 sec.² These solution times are summarized in Table 3.2.

² The simulation was performed using the *Model Predictive Control Toolbox* function `scmpc` for the hard-constrained case, and its modified version `scmpc2` for the soft-constrained cases — see Appendix C. The latter uses a different QP solver, so the solution speeds are not directly comparable. It should also be borne in mind that the quoted solution times are very far from the minimum achievable times.

Table 3.2 Relative solution times for different formulations of altitude change manoeuvre.

Problem formulation	Time
Soft-constrained, 1-norm penalty	0.81 s
Soft-constrained, ∞ -norm penalty	0.15 s
Hard-constrained	0.04 s

A more typical reason for encountering infeasibility is the presence of disturbances. The next example illustrates this, and again shows the efficacy of softening constraints. It also shows that using inexact penalty functions can give a useful compromise between the harshness of constraints and the ‘sponginess’ of set-point specifications.

Example 3.3

In Section 2.7, Figure 2.9, the response was shown to a disturbance of 5 m/sec in the altitude rate, arising from turbulence acting on the Citation aircraft. If, however, the disturbance is slightly greater, 6 m/sec instead of 5 m/sec, then the controller finds that it cannot return the altitude rate to below 30 m/sec within one sample period — because of the elevator angle being limited to 15° — and infeasibility occurs. Note that this is true even if the model is perfect.

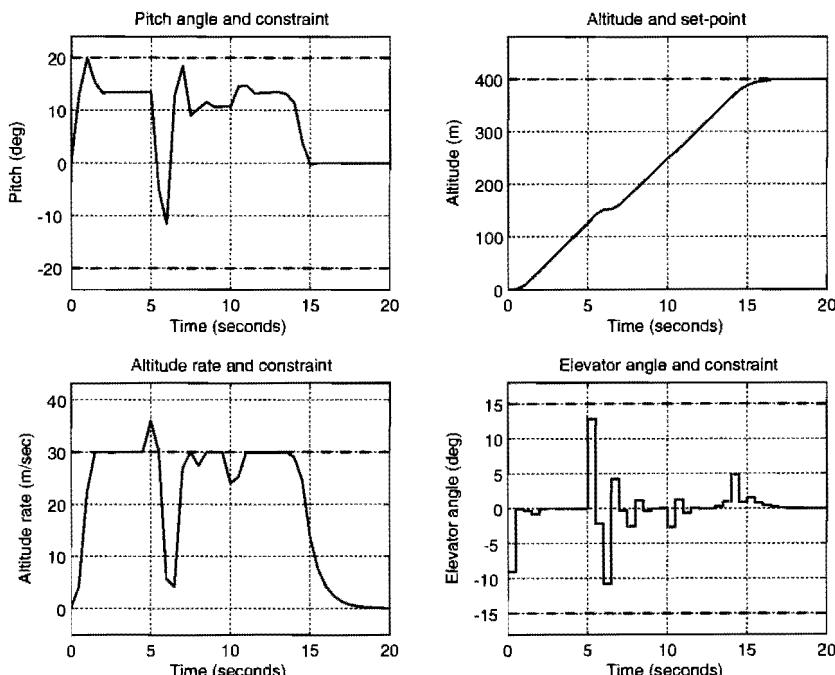


Figure 3.9 Response to disturbance with softened altitude rate constraint, $\rho = 5 \times 10^5$.

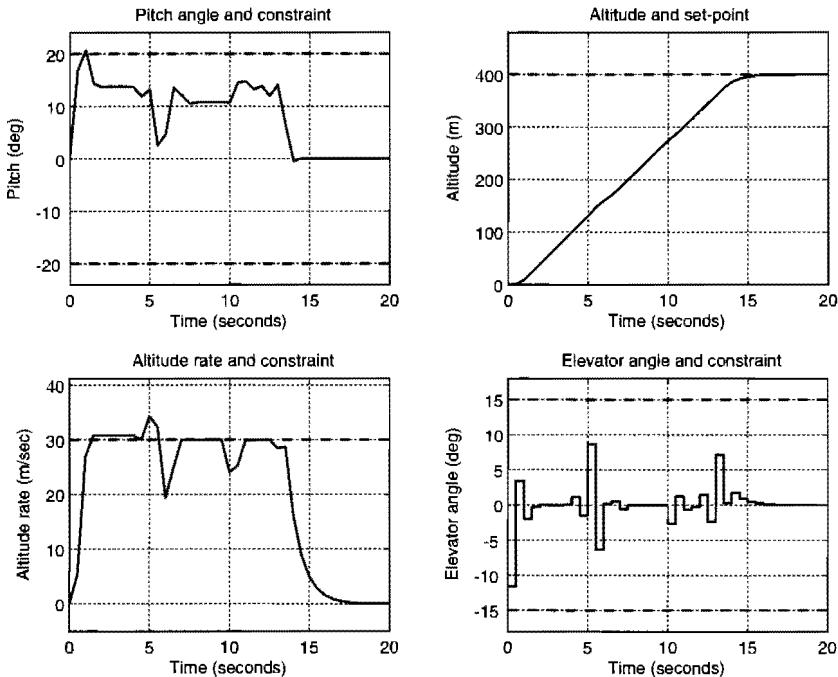


Figure 3.10 Response to disturbance with softened altitude rate constraint, $\rho = 10^4$.

If, however, the constraints are softened, then the disturbance can be dealt with. Figure 3.9 shows the response when an ∞ -norm penalty function is used with $\rho = 5 \times 10^5$. Such a large value of ρ is needed to get an exact penalty function in this case; the peak value of all the Lagrange multipliers obtained during the simulation shown in Figure 2.9 was approximately 4×10^5 . This response is very similar to that shown in Figure 2.9; thus, although the manoeuvre can be completed successfully despite the large disturbance, the disadvantage of a violent response to avoid violation of the constraints is still present.

If, however, the penalty coefficient is reduced to $\rho = 10^4$, so that the penalty function is no longer exact, then the response shown in Figure 3.10 is obtained. The response to the onset of the disturbance is now significantly less aggressive than before, although still more aggressive than the response to the disappearance of the disturbance 5 seconds later. The inexactness of the penalty function reveals itself by the slight violation of the altitude rate constraint before the disturbance occurs.

Using a 1-norm penalty function (instead of ∞ -norm) gives very similar results for this example, if the same values of the penalty coefficient ρ are used.

In [RK93] an unconventional way of solving the constrained QP problem is introduced. This solves the problem as an iteration of weighted least-squares problems, the weights being changed at each iteration in such a way that the greatest penalty is attached to the most-violated constraint. This can be shown to find the correct solution to the QP problem if it exists, but has the advantage that it produces a solution even

if the problem is infeasible, so that it enforces soft constraints, in effect. Furthermore, the solution that is produced is one which tries to minimize the worst-case constraint violation, in the sense of minimizing the peak value of the violation.

It may happen, however, that minimizing the magnitude of constraint violations leads to the violations persisting over an unnecessarily long time, and this may not be the preferred trade-off. In [SR99] several formulations of soft constraints are discussed, which allow various magnitude/duration trade-offs to be imposed.

Exercises

3.1 Work through the unconstrained predictive control example described in the section ‘Unconstrained MPC using state-space models’ of the *MPC Toolbox User’s Guide*. Don’t worry about how the *Toolbox* does the conversion from transfer function to state-space form. Also don’t worry about how the observer (= ‘estimator’) gain is designed (p. 55). The emphasis for the moment is on the ‘mechanical’ skills of running the *Toolbox*.

(You should read the detailed description of the function `smpccon` before/while doing this.)

3.2 Work through the constrained predictive control example described in the section ‘State-space MPC with constraints’ of the *MPC Toolbox User’s Guide*.

(You should read the detailed description of the function `scmpc` before or while doing this.)

3.3 The water temperature in a heated swimming pool, θ , is related to the heater input power, q , and the ambient air temperature, θ_a , according to the equation

$$T \frac{d\theta}{dt} = kq + \theta_a - \theta \quad (3.93)$$

where $T = 1$ hour and $k = 0.2 \text{ } ^\circ\text{C/kW}$. (It is assumed that the water is perfectly mixed, so that it has a uniform temperature.) Predictive control is to be applied to keep the water at a desired temperature, and a sampling interval $T_s = 0.25$ hour is to be used. The control update interval is to be the same as T_s .

(a) Use *MATLAB*’s *Control System Toolbox* to show that the corresponding discrete-time model is

$$\theta(k+1) = 0.7788\theta(k) + 0.0442q(k) + 0.2212\theta_a(k) \quad (3.94)$$

Form the corresponding model in the *Model Predictive Control Toolbox*’s ‘MOD’ format, assuming that q is the control input and that θ_a is an unmeasured disturbance.

(b) Verify that if the weights $Q = 1$ and $R = 0$, and the horizons $H_p = 10$ and $H_u = 3$ (and $H_w = 1$) are used, then

$$K_s = K_{MPC}[1, -\Psi, -\Upsilon] = [22.604, -17.604, -22.604] \quad (3.95)$$

where K_s is the unconstrained controller gain matrix K_s produced by *Model Predictive Control Toolbox* function `smpccon`, and that stable closed-loop behaviour is obtained.

- (c) If the air temperature θ_a is constant at 15 °C verify that a set-point for θ of 20 °C is attained without error. Verify that this remains the case even if the pool parameters change to $T = 1.25$ hour, $k = 0.3$ °C/kW but the predictive controller's internal model remains unchanged.
- (d) Suppose that the air temperature follows a sinusoidal diurnal variation with amplitude 10 °C:

$$\theta_a(t) = 15 + 10 \sin\left(\frac{2\pi}{24}t\right) \quad (3.96)$$

(where t is measured in hours). Verify that in the steady state the mean water temperature reaches the set-point exactly, but that θ has a small residual oscillation of amplitude approximately 0.5 °C.

- (e) Now suppose that the input power is constrained:

$$0 \leq q \leq 40 \text{ kW} \quad (3.97)$$

and that the air temperature varies sinusoidally, as above. Investigate the behaviour of the predictive control system — use *Model Predictive Control Toolbox* function `smpc`.

(You are advised to save any models created, and to keep a record of *MATLAB* commands you use — using `diary` for example — as this swimming pool example will reappear in later examples and problems. See Exercises 5.1 and 5.6, Example 7.4 and Exercise 7.6.)

3.4 If *blocking* is used, as in the *Model Predictive Control Toolbox*, how should the details of equation (2.23) be changed? (Also see Exercise 1.14.)

3.5 Suppose that the cost function used in the predictive control formulation was changed to include the penalty term

$$\sum_{i=0}^{H_u-1} ||\hat{u}(k+i|k) - u_{ref}(k+i)||_{S(i)}^2$$

in addition to the terms penalizing tracking errors and control moves, where u_{ref} denotes some prescribed future trajectory for the plant inputs.

- (a) Describe in detail the changes that would need to be made to the computations of predictions and the optimal solution.
- (b) Briefly list some reasons for and against making such a modification.

3.6 (a) If inequality (3.35) arises from the simple range constraints (3.38), show that

\mathcal{F} takes the simple form

$$\mathcal{F} = \begin{bmatrix} I & 0 & \cdots & 0 \\ -I & 0 & \cdots & 0 \\ I & I & \cdots & 0 \\ -I & -I & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ I & I & \cdots & 0 \\ -I & -I & \cdots & 0 \\ I & I & \cdots & I \\ -I & -I & \cdots & -I \end{bmatrix}$$

What form does the vector on the right-hand side of inequality (3.37) take in this case? (Note that it is possible to reorder the inequalities so that \mathcal{F} takes the form of two lower-triangular matrices stacked on top of each other in this case, each one being made up of identity matrices. This is commonly done in the literature [PG88].)

- (b) Put inequality (3.34) into the form $W\Delta\mathcal{U}(k) \leq w$, where W is a matrix and w is a vector. If (3.34) arises from simple rate constraints of the form $-B(k+i) \leq \Delta\hat{u}(k+i|k) \leq B(k+i)$, what form do W and w take?
- 3.7** If you look at the code for function `smpcccon` in the *MPC Toolbox* (using type `smpcccon`), you will see how the weights are forced to be diagonal before solving for the unconstrained gain matrix K_{MPC} . Copy the function to your own directory, change its name, and edit it so that you can specify arbitrary weights. Then solve Example 6.1 and Exercise 6.2 using your new function. Simulate the designs using `smpcsim`.
- 3.8** Put the QP problem stated in Example 3.1 into the standard form (3.56)–(3.58). Check that $\Phi \geq 0$, and hence that the problem is convex. Check that the point $(\theta_1 = 3/2, \theta_2 = 1/2)$ satisfies the KKT conditions (3.59)–(3.62), and hence that it is the optimal solution, as claimed in the example.
- 3.9** (a) Check that the optimization problem (3.86)–(3.87) is a standard QP problem, as claimed in the text.
 (b) Modify the problem (3.86)–(3.87) to deal with the case that a subset of the constraints $\Omega_1\theta \leq \omega_1$ is to be ‘hard’, while the remainder, $\Omega_2\theta \leq \omega_2$, is to be ‘soft’, and again check that this is a QP problem.
- 3.10** Check that the optimization problems (3.88)–(3.4) and (3.90)–(3.91) are standard QP problems.
- 3.11** Suppose that the gain from the elevator angle to the altitude rate of the Citation aircraft model is underestimated by 20%, rather than overestimated, as in Example 3.2, but everything else is the same as in Example 3.2. Show that infeasibility is not encountered in this case, but the response is relatively oscillatory — as one would expect with conventional control if the plant gain were higher than expected.

(The function `mismatch2`, available from this book's web site, makes it easy to run the simulation. Use parameter values: `percent=-20` to simulate the 20% gain undermodelling, and `pencoeff=inf`, `normtype='inf-norm'` to enforce hard constraints.)

- 3.12** Verify that similar results to those shown in Example 3.3 are obtained if a 1-norm penalty function is used instead of an ∞ -norm penalty function.

(The function `disturb2`, available from this book's web site, makes it easy to run the simulation. The parameter `pencoeff` corresponds to the penalty coefficient ρ , and can either be a scalar (the same coefficient for each constraint violation) or a vector, one element corresponding to each constraint. Set `normtype='1-norm'` to select the 1-norm penalty function.)

Step response and transfer function formulations

4.1 Step and pulse response models	108
4.2 Transfer function models	121
Exercises	143

The original formulations of predictive control, such as DMC [CR80, PG88], used step or pulse response models instead of state-space models. Some commercial products still use these representations. GPC [CMT87, Soe92] popularized the use of transfer function, or difference equation, models in predictive control; these models are the most widely used in the academic literature of predictive control, and are also used in some commercial products. This chapter shows the details of using these models in predictive control. Wherever possible, everything is related to what has already been done with state-space models.

4.1 Step and pulse response models

4.1.1 Step and pulse responses

The idea behind step response models is that one can apply a step input to each input ('manipulated variable' or actuator) of the plant, and record the open-loop response of each output variable, until all the output variables have settled to constant values. Because of the assumed linearity of the plant, knowing the (multivariable) step response allows one to deduce the response to any other input signal (vector). This is an easy and intuitive concept; however, it does have some drawbacks:

- ➊ It can only be used with asymptotically stable plants.
- ➋ It is frequently impractical to apply step inputs — they can be too disruptive to normal operations.
- ➌ Estimating models from step responses gives a lot of emphasis to low frequencies. For feedback control other frequencies are usually more important. In particular, step response tests estimate steady-state gains very well. But steady-state gains are relatively unimportant for feedback design — the feedback changes them anyway. (However, steady-state gains are important for certain purposes, such as sizing actuators and steady-state optimization.)
- ➍ Step response models are only adequate if all the controlled variables are measured outputs. (Otherwise one needs some other kind of model to predict how the unmeasured outputs will behave.)
- ➎ Storing multivariable step responses is an inefficient form of model representation, in terms of storage requirements.

There are some myths about step response models, particularly in the process industries, which are simply not true:

- ➏ Step response models are necessary to capture complicated patterns in the response. This is not true; one can routinely build a state-space model which can reproduce *exactly* any given step response, even a multivariable one. This will be shown in this chapter.
- ➐ Step response models are necessary to represent delays in the plant. This is also not true; since we are working in discrete time, one can simply introduce states into state-space models to represent as many delays as necessary.

Mathematically, a more fundamental concept than the step response is the pulse response. Suppose that the plant is at steady state, with all inputs and outputs initially at 0. We apply a unit pulse at time 0 on input j :

$$u_j(0) = 1 \quad u_j(k) = 0 \text{ for } k > 0$$

Let the response sequence of output i be

$$(h_{ij}(0), h_{ij}(1), \dots)$$

so that the vector of responses of all the outputs at time t is

$$[h_{1j}(t), h_{2j}(t), \dots, h_{pj}(t)]^T$$

We can arrange a matrix which shows how each output responds to a unit pulse on each input:

$$H(t) = \begin{bmatrix} h_{11}(t) & h_{12}(t) & \dots & h_{1m}(t) \\ h_{21}(t) & h_{22}(t) & \dots & h_{2m}(t) \\ \vdots & \vdots & \ddots & \vdots \\ h_{p1}(t) & h_{p2}(t) & \dots & h_{pm}(t) \end{bmatrix} \quad (4.1)$$

Now the response $y(t)$ to an arbitrary input signal vector $\{u(0), u(1), \dots\}$ is — because of linearity — given by the *convolution sum*:

$$y(t) = \sum_{k=0}^t H(t-k)u(k) \quad (4.2)$$

(This is just the discrete-time equivalent of the convolution integral which occurs in continuous-time linear systems theory.)

In principle, the sequence of pulse response matrices $\{H(0), H(1), \dots, H(N)\}$ can be obtained from pulse response tests. However, it is very rarely practical to perform such tests, because unacceptably large pulse amplitudes are usually needed in order to excite the plant sufficiently to get useful results. The sequence must be long enough so that $H(N) \approx 0$. If one really performed such a test, the initial steady state would not be at zero, and the sequence would really provide information about the differences $y(t) - y(-1)$ resulting from pulse inputs on u .

Now let us return to the concept of the step response of the plant. Consider a *unit step* on input j : $\{u_j(t)\} = (1, 1, 1, \dots)$. Using (4.2) the response of output i is:

$$\begin{aligned} y_i(t) &= \sum_{k=0}^t h_{ij}(t-k)u_j(k) \\ &= \sum_{k=0}^t h_{ij}(t-k) \\ &= \sum_{k=0}^t h_{ij}(k) \end{aligned}$$

so that we can define the step response matrix as

$$S(t) = \sum_{k=0}^t H(k) \quad (4.3)$$

This matrix (or perhaps the whole sequence $(S(0), S(1), \dots)$ — it is not clear) is sometimes called the *Dynamic Matrix* of the plant, which gives rise to the name *Dynamic Matrix Control*, or *DMC*. The sequence $(S(0), S(1), \dots, S(N))$ can be used as a model of the plant, if N is sufficiently large that $S(N+1) \approx S(N)$.

Recall that in the standard formulation we use changes of the input $\Delta u(t) = u(t) - u(t-1)$, rather than the input itself. We can express the output which results from an arbitrary input sequence by using the step response matrices and the changes of the input, instead of pulse response matrices and the actual input as we did in (4.2):

$$y(t) = \sum_{k=0}^t H(t-k)u(k) \quad (4.4)$$

$$= \sum_{k=0}^t H(t-k) \sum_{i=0}^k \Delta u(i) \quad (\text{assuming } u(0) = 0) \quad (4.5)$$

$$= \sum_{k=0}^t H(k)\Delta u(0) + \sum_{k=0}^{t-1} H(k)\Delta u(1) + \dots + H(0)\Delta u(t) \quad (4.6)$$

$$= \sum_{k=0}^t S(t-k)\Delta u(k) \quad (4.7)$$

Notice the similarity in form of this expression and of (4.2). Again, in practice we should replace $y(t)$ by $y(t) - y(-1)$.

It is useful to note that the arguments $t - k$ and k can be interchanged in (4.7), as usual in convolutions: let $\ell = t - k$, then

$$y(t) = \sum_{k=0}^t S(t-k)\Delta u(k) \quad (4.8)$$

$$= \sum_{\ell=t}^0 S(\ell)\Delta u(t-\ell) \quad (4.9)$$

$$= \sum_{k=0}^t S(k)\Delta u(t-k) \quad (4.10)$$

4.1.2 Relations to state-space models

Suppose the plant has the state-space model

$$x(k+1) = Ax(k) + Bu(k) \quad (4.11)$$

$$y(k) = C_y x(k) + D_y u(k) \quad (4.12)$$

and suppose that $x(0) = 0$. Now apply a pulse input vector, so that $u(0) \neq 0$, but $u(k) = 0$ for all $k > 0$. We get the following sequence of states and outputs:

$$x(0) = 0 \quad y(0) = D_y u(0) \quad (4.13)$$

$$x(1) = Bu_0 \quad y(1) = C_y Bu(0) \quad (4.14)$$

$$x(2) = ABu(0) \quad y(2) = C_y ABu(0) \quad (4.15)$$

$$\vdots \quad \vdots$$

$$x(k) = A^{k-1} Bu(0) \quad y(k) = C_y A^{k-1} Bu(0) \quad (4.16)$$

$$\vdots \quad \vdots$$

It is clear from this sequence that the pulse response matrix sequence is given by

$$H(0) = D_y \quad (\text{which is often } 0) \quad (4.17)$$

$$H(1) = C_y B \quad (4.18)$$

$$H(2) = C_y AB \quad (4.19)$$

$$\vdots \quad \vdots$$

$$H(k) = C_y A^{k-1} B \quad (4.20)$$

$$\vdots \quad \vdots$$

The matrix $C_y A^{k-1} B$ is called the k th *Markov parameter* of the state-space model.

From this we immediately get the step response sequence, using (4.3), as

$$S(0) = D_y \quad (\text{which is often } 0) \quad (4.21)$$

$$S(1) = C_y B + D_y \quad (4.22)$$

$$S(2) = C_y A B + C_y B + D_y \quad (4.23)$$

⋮

$$S(k) = \sum_{i=0}^{k-1} C_y A^i B + D_y \quad (4.24)$$

$$= C_y \left(\sum_{i=0}^{k-1} A^i \right) B + D_y \quad (4.25)$$

⋮

$$(4.26)$$

Looking back at equations (2.66) and (2.70), and at Exercises 2.11 and 2.12 — and remembering that here we must assume that $y = z$ — one can see that most of the expressions appearing in the matrices required for computing the predictions needed for predictive control in fact involve the step response matrices. In particular, recalling (3.5):

$$\mathcal{Z}(k) = \Psi x(k) + \Upsilon u(k-1) + \Theta \Delta \mathcal{U}(k)$$

and the definitions of the matrices Υ and Θ , we see that

$$\Upsilon = \begin{bmatrix} S(H_u) \\ S(H_u + 1) \\ \vdots \\ S(H_p) \end{bmatrix} \quad (4.27)$$

and

$$\Theta = \begin{bmatrix} S(H_u) & S(H_u - 1) & \dots & 0 \\ S(H_u + 1) & S(H_u) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ S(H_u) & S(H_u - 1) & \dots & S(1) \\ S(H_u + 1) & S(H_u) & \dots & S(2) \\ \vdots & \vdots & \ddots & \vdots \\ S(H_p) & S(H_p - 1) & \dots & S(H_p - H_u + 1) \end{bmatrix} \quad (4.28)$$

So we can immediately use the results from Chapter 2 for prediction, and those from Chapter 3 for obtaining a solution, if we have a step response model of the plant. Well, not quite immediately, because the predictions in Chapter 2 made use of the state $x(k)$. With a step response model we do not seem to have a state, so we need to replace $x(k)$ with something else. The state of a system summarizes its ‘past history’ — if you know

$x(k)$ you don't need to know anything that happened before time k in order to predict its future behaviour. But now we do not have a state accessible, so it is reasonable to expect that we will have to use 'old' information, about what happened to the system before time k — in principle we need to look infinitely far back into the past.

4.1.3 Prediction using step response model

In order to see what exactly we should do, it is easiest to start from first principles. Given that the only model that we have of plant behaviour is the step response (matrix) sequence $\{S(0), \dots, S(N)\}$ (by assumption) — or, equivalently, the pulse response sequence $\{H(0), \dots, H(N)\}$, and no disturbance model for the time being, we can form predictions of future outputs as:

$$\hat{z}(k+j|k) = \hat{y}(k+j|k) = \sum_{i=1}^{\infty} H(i)\tilde{u}(k+j-i) \quad (4.29)$$

where the input sequence $\{\tilde{u}\}$ consists of known inputs from the past and predicted inputs in the future:

$$\tilde{u}(i) = \begin{cases} u(i) & \text{if } i < k, \\ \hat{u}(i|k) & \text{if } i \geq k. \end{cases} \quad (4.30)$$

Since we assume that $H(i) \approx 0$ for $i > N$ we approximate this prediction by

$$\hat{z}(k+j|k) = \sum_{i=1}^N H(i)\tilde{u}(k+j-i) \quad (4.31)$$

$$= \sum_{i=j+1}^N H(i)u(k+j-i) + \sum_{i=1}^j H(i)\hat{u}(k+j-i|k) \quad (4.32)$$

$$\begin{aligned} &= \sum_{i=j+1}^N H(i)u(k+j-i) \\ &\quad + \sum_{i=1}^j H(i) \left[u(k-1) + \sum_{\ell=0}^{j-i} \Delta\hat{u}(k+j-i-\ell|k) \right] \quad (4.33) \end{aligned}$$

$$\begin{aligned} &= \sum_{i=j+1}^N H(i)u(k+j-i) + \sum_{i=1}^j H(i)u(k-1) \\ &\quad + H(1)\Delta\hat{u}(k+j-1|k) + [H(1) + H(2)]\Delta\hat{u}(k+j-2|k) + \dots \\ &\quad + [H(1) + H(2) + \dots + H(j)]\Delta\hat{u}(k|k) \quad (4.34) \end{aligned}$$

Now, recalling that $S(j) = \sum_{i=1}^j H(i)$ (if we assume $H(0) = 0$), and that

$$\sum_{i=1}^j H(i)u(j-i) = \sum_{i=1}^j S(i)\Delta u(j-i) \quad (4.35)$$

we can write the prediction as

$$\hat{z}(k+j|k) = \sum_{i=j+1}^N S(i)\Delta u(k+j-i) + S(j)u(k-1) + \sum_{i=1}^j S(i)\Delta \hat{u}(k+j-i|k) \quad (4.36)$$

Now from equation (2.66) we had (using the relations derived in the previous subsection, and remembering that $z = y$ in this section):

$$\hat{z}(k+j|k) = C_y A^j x(k) + S(j)u(k-1) + \sum_{i=1}^j S(i)\Delta \hat{u}(k+j-i|k) \quad (4.37)$$

The second and third terms are the same in both cases. So we see that the term

$$\Psi x(k) = \begin{bmatrix} C_y A^{H_w} \\ \vdots \\ C_y A^{H_p} \end{bmatrix} x(k) \quad (4.38)$$

which appears in the prediction when a state-space model is used must be replaced by the term

$$\begin{bmatrix} S(H_w + 1) & S(H_w + 2) & \cdots & \cdots & \cdots & S(N - 1) & S(N) \\ S(H_w + 2) & S(H_w + 3) & \cdots & \cdots & \cdots & S(N) & S(N) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ S(H_p + 1) & S(H_p + 2) & \cdots & S(N) & \cdots & S(N) & S(N) \end{bmatrix} \times \begin{bmatrix} \Delta u(k-1) \\ \Delta u(k-2) \\ \vdots \\ \Delta u(k+H_w-N) \end{bmatrix} \quad (4.39)$$

Not surprisingly, one must have $N > H_p$. In addition, one must store the last $N - H_w$ input moves (or input values). This gives a striking illustration of the ‘efficiency’ of state-space models at representing the past history of a system.

A matrix such as this, with blocks along ‘anti-diagonals’ being the same, is called a *block-Hankel* matrix. Such matrices appear frequently in systems theory, usually when relating ‘past inputs’ to ‘future outputs’, as here.

The only disturbance model commonly used with step response models is the ‘constant output disturbance’ model that was treated in Section 2.3.2, and the disturbance is estimated using the ‘DMC scheme’ described there. The disturbance is simply estimated as the difference between the measured and the predicted outputs:

$$\hat{d}(k|k) = y(k) - \hat{y}(k|k-1), \quad (4.40)$$

it is assumed that it will remain the same over the prediction horizon:

$$\hat{d}(k+j|k) = \hat{d}(k|k) \quad (4.41)$$

and this estimate is just added on to the prediction obtained previously:

$$\begin{aligned}\hat{z}(k+j|k) &= \sum_{i=j+1}^N S(i)\Delta u(k+j-i) + S(j)u(k-1) \\ &\quad + \sum_{i=1}^j S(i)\Delta \hat{u}(k+j-i|k) + \hat{d}(k+j|k)\end{aligned}\quad (4.42)$$

Since we are now using an input-output model, there is no alternative to assuming a disturbance which acts directly on the inputs or outputs. But it is not essential to assume that such a disturbance remains constant. For example, one could assume an exponentially decaying output disturbance:

$$\hat{d}(k+j|k) = \alpha^j \hat{d}(k|k) \quad \text{with } 0 < \alpha < 1 \quad (4.43)$$

The main reason why a constant disturbance is commonly assumed is that this leads to integral action in the controller, as we shall see later in the book. More complicated disturbance models can be handled much more systematically when one uses state-space or transfer function models.

4.1.4 State-space models from step responses

There is really no good reason for working with step (or pulse) response models. If one has such a model, one can easily obtain a state-space model which reproduces the step response *exactly*, without losing anything. But this model will be of large state dimension — $N \times \min(m, p)$ if the plant has m inputs, p outputs, and you want to match N step response matrices. Usually it is possible to get an approximate model of *much* lower state dimension, which matches the step response matrices very closely. This subsection will describe one way in which this can be done.

First we start with a very simple method of getting state-space models, best described by an example.

Example 4.1

Consider an example which is simple enough to work through by hand. Suppose we have a 1-input, 1-output (SISO) system with step responses $S(0) = 0$, $S(1) = 0$, $S(2) = -1$, $S(3) = +2$, $S(k) = S(3)$ for $k > 3$. Note that $S(1) = 0$ shows that this system has a delay between the input and output. The corresponding pulse responses are $H(0) = 0$, $H(1) = 0$, $H(2) = -1$, $H(3) = +3$, $H(k) = 0$ for $k > 3$. Since the pulse response becomes 0 after 3 steps and we have only 1 input and output, we need at most 3 states. In order to get a ‘finite impulse response’ or ‘FIR’ model, all the eigenvalues of the ‘A’ matrix will have to be at 0. So a suitable (but not the only) choice is

$$A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

This does not determine B and C uniquely, but try

$$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

(This combination of A and B is known as the *controller form* of a state-space model [Kai80].) Now $C = [c_1, c_2, c_3]$ is determined uniquely from the equations:

$$CB = c_3 = 0$$

$$CAB = c_2 = -1$$

$$CA^2B = c_1 = 3$$

namely $C = [3, -1, 0]$. And $D = 0$ since $H(0) = 0$.

A state-space model for any single-input, single-output ('SISO') system can be obtained from step response data in this way. Even for a multivariable system it is possible to use this method for one input at a time, and then combine the 1-input models. However, for large N , and with more than one input and output, numerical problems begin to arise. And models obtained in this form (the 'controller form') tend to have numerical problems.

A better practical algorithm is the following. Recall from Section 4.1.2 that the pulse response matrices are related to the state-space matrices by

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

Hence we have the following relationship:

$$\begin{bmatrix} H(1) & H(2) & \cdots \\ H(2) & H(3) & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} = \begin{bmatrix} CB & CAB & \cdots \\ CAB & CA^2B & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad (4.44)$$

$$= \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \end{bmatrix} [B, AB, A^2B, \dots] \quad (4.45)$$

That is, the block-Hankel matrix built up from the pulse response matrices factorizes into the product of the (extended) observability and controllability matrices. Since each of these has rank n for a minimal state-space system of state dimension n , the block-Hankel matrix must also have rank n .

Given a set of step response matrices $S(0), \dots, S(N)$, we obtain a corresponding set of pulse response matrices $H(0), \dots, H(N)$, and assemble them into a block-Hankel matrix, as shown in (4.44). The rank of this matrix will usually be large. But it is usually 'close' to a matrix of much lower rank. The idea now is to approximate it by a lower-rank matrix, factorize it into two low-rank matrices, and compute A , B , and C from

Let M be any matrix of dimensions $p \times m$. It can always be factorized as

$$M = U \Sigma V^T \quad (4.46)$$

where $UU^T = I_p$, $VV^T = I_m$, and Σ is a rectangular matrix of dimensions $p \times m$ which has non-zero entries only on its leading diagonal. For the case $p < m$ it looks like:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_N & 0 & \cdots & 0 \end{bmatrix}$$

and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_N \geq 0$. This factorization is called the *Singular Value Decomposition*. The σ s are called the *Singular Values* of M .

The number of positive (non-zero) σ s is the rank of the matrix. If one replaces Σ by Σ_n , in which $\sigma_{n+1}, \sigma_{n+2}, \dots, \sigma_N$ are replaced by zeros (where $n < N$), then the matrix

$$M_n = U \Sigma_n V^T$$

has rank n . Furthermore the matrix M_n is the *best rank-n approximation* to M , in the sense that the error $\|M - M_n\|_F = \sigma_{n+1}$ is the smallest possible among all rank- n matrices. ($\|X\|_F$ denotes the *Frobenius norm*, which is just the square root of the sum of the squares of all the elements of X : $\|X\|_F = \sqrt{\text{trace}(X^T X)}$.)

The *SVD* can be computed extremely reliably, even for very large matrices, although it does involve some heavy computation. The *flop count* increases as $(\min(m, p))^3$. In *MATLAB* it can be obtained by the function `svd: [U,S,V] = svd(M)`

The *SVD* is of great importance in numerical linear algebra [GL89], and in multivariable robust control theory [Mac89, ZDG96].

The Moore–Penrose pseudo-inverse of M is obtained from its *SVD* as

$$M^\dagger = V \Sigma^\dagger U^T \quad (4.47)$$

where Σ^\dagger is obtained from Σ by replacing each positive σ_i by $1/\sigma_i$.

The largest singular value, σ_1 , is an *induced norm* of M :

$$\sigma_1 = \sup_{x \neq 0} \frac{\|Mx\|}{\|x\|} \quad (4.48)$$

This is the origin of the importance of the *SVD* in robust control theory.

Mini-Tutorial 4 The singular value decomposition.

these. The essential tool for doing this is the *Singular Value Decomposition* or *SVD* — see Mini-Tutorial 4.

Suppose that the available pulse response matrices are assembled into an ‘anti-upper triangular’ block-Hankel matrix as follows:

$$\mathcal{H}_N = \begin{bmatrix} H(1) & H(2) & \cdots & H(N) \\ H(2) & H(3) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ H(N) & 0 & \cdots & 0 \end{bmatrix} \quad (4.49)$$



Now obtain an *SVD* of this matrix:

$$\mathcal{H}_N = U \Sigma V^T. \quad (4.50)$$

This has rank $N \times \min(m, p)$. But the singular values usually approach zero very quickly, so that \mathcal{H}_N is usually close to a matrix of lower rank. Somehow choose a value $n < N \times \min(m, p)$. This step is not rigorous, but for example one can choose n such that $\sigma_n < \sigma_1/100$, or some similar (heuristic) criterion. It is sometimes suggested that one should choose n such that there is a significant ‘gap’ between σ_n and σ_{n+1} , but with real data no such gap is usually evident. Let Σ_n be the result of replacing $\sigma_{n+1}, \sigma_{n+2}, \dots$ in Σ by zeros.

Now we have

$$\mathcal{H}_n = U \Sigma_n V^T \quad (4.51)$$

as a rank- n approximation to \mathcal{H}_N , and we can factorize it as follows: define

$$\Omega_n = U \Sigma_n^{1/2} \begin{bmatrix} I_n \\ 0 \end{bmatrix} \quad (4.52)$$

and

$$\Gamma_n = [I_n, 0] \Sigma_n^{1/2} V^T \quad (4.53)$$

where $\Sigma_n^{1/2}$ is the same as Σ , but with each σ_i replaced by $\sqrt{\sigma_i}$. Note that Ω_n has n columns, Γ_n has n rows, and

$$\mathcal{H}_n = \Omega_n \Gamma_n \quad (4.54)$$

Now the idea is to find the A, B and C matrices of an n -state model, which have Ω_n and Γ_n as their observability and controllability matrices, respectively.

Finding B and C is easy: just take B as the first m columns of Γ_n , and take C as the top p rows of Ω_n . Finding A is almost as easy. Notice that if an (exact) observability matrix Ω is ‘shifted upwards’ by p rows to form the matrix Ω^\uparrow , then $\Omega^\uparrow = \Omega A$. So if we obtain Ω_n^\uparrow from Ω_n by shifting it upwards by p rows, then we can estimate a suitable A by solving the equation

$$\Omega_n^\uparrow = \Omega_n A \quad (4.55)$$

This equation is usually over-determined, so there is no exact solution, but it can be solved in a least-squares sense. It can be shown that the least-squares solution is given by

$$A = \Sigma_n^{-1/2} U^T U^\uparrow \Sigma_n^{1/2} \quad (4.56)$$

where U^\uparrow is the same as U , but shifted upwards by p rows.

The algorithm described here is due to Kung [Kun78] and to Zeiger and McEwen [ZM74]. It can be shown that if n is taken as the largest possible value, namely $n = N \times \min(m, p)$, then the state-space model obtained reproduces the pulse response (and hence the step response) exactly, and all the eigenvalues of A are then at zero, so that it is a ‘deadbeat’ or ‘finite impulse response’ model.

Example 4.2

Applying this method to the problem of Example 4.1, with $n = N = 3$, gives

$$A = \begin{bmatrix} -0.7021 & 0.5117 & -0.0405 \\ -0.5117 & 0.0023 & 0.4807 \\ -0.0405 & -0.4807 & 0.6998 \end{bmatrix} \quad B = \begin{bmatrix} 1.0430 \\ -1.2499 \\ 0.6887 \end{bmatrix}$$

$$C = [1.0430, 1.2499, 0.6887] \quad D = 0$$

The reader can check that it reproduces the original step response exactly.

An interesting property of this algorithm is that, once one has a model for a given value of n , it is possible to obtain the model for a smaller value, say q , by ‘truncating’ the matrices: $A_q = A_n(1 : q, 1 : q)$, $B_q = B_n(1 : q, :)$, and $C_q = C_n(:, 1 : q)$, using MATLAB notation. So the heavy computation involved in finding the SVD has to be done only once, and a range of approximate linear realizations of various state dimensions is then easily obtained. For more details of this, and the associated topic of approximation by *balanced truncation* or *balanced model reduction*, see [Mac89, ZDG96].

Example 4.3

Figure 4.1 shows the step responses of a high-fidelity, first-principles, nonlinear simulation of a 3-input, 2-output distillation column. The solid curves show the original responses, while the dotted curves show the approximate responses achieved by a model obtained using the algorithm described above, with *only 8 states*. Note that some of the responses have quite intricate shapes; yet a model with such a low number of states was able to reproduce these quite accurately.

The original step response data was sampled at 10 second intervals, and 250 samples were available ($N = 250$). A model of largest possible state dimension was first obtained, namely one with $\min(2, 3) \times 250 = 500$ states. This matched the original step responses exactly, as expected.

Figure 4.2 shows the first 15 singular values (of 500) of the block-Hankel matrix \mathcal{H}_N . Assessing these ‘by eye’ indicates that about 8 states should be enough for a good approximation. The 8-state model was obtained by following the algorithm described above, with $n = 8$.

The data for this example is available in the *DAISY* database of data used for various system identification exercises, which is available on the World-Wide Web.¹ A brief description of the data, and relevant references, are available in *DAISY*.

The state of the state-space model constructed above does not have any direct physical interpretation. It is possible to obtain state-space models from step responses in such a way that the elements of the state vector *do* have physical interpretations, so long as one does not approximate the model by one with a smaller state vector. For example, Lee *et al* give a way of constructing a state-space model in which the state variables can be interpreted as future outputs [LMG94]. In [CC95, CC99] an alternative is given, in which the state vector contains past values of the plant outputs and control moves.

¹ URL: <http://www.esat.kuleuven.ac.be/sista/daisy/>

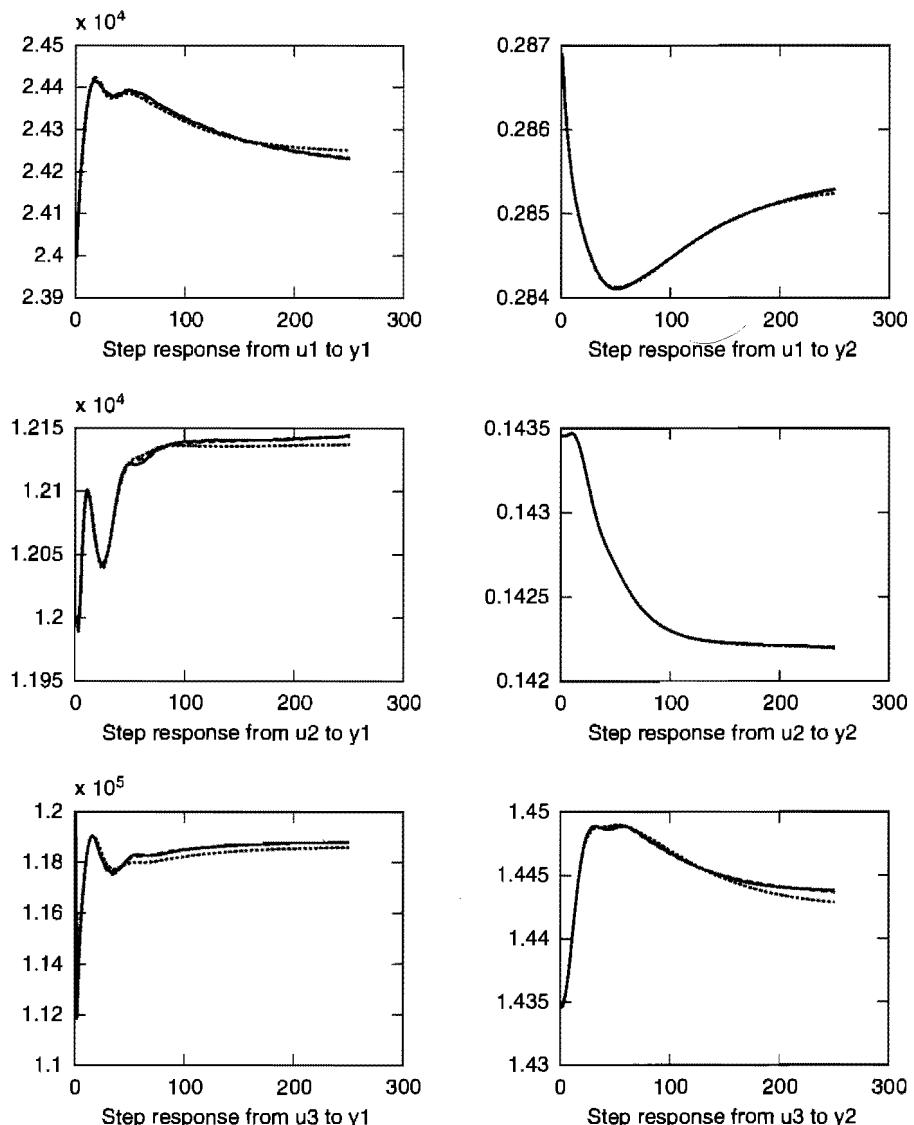


Figure 4.1 Step response of 3-input, 2-output distillation column. Solid curves: original responses. Dotted curves: responses of 8-state model.

Note that the procedure:

step tests \rightarrow step responses \rightarrow state-space model

is not a particularly good one. In general it is better to go directly from step or other tests (or possibly normal operating data) to a state-space model, using methods of *System Identification* which have good statistical properties [Lju89, Nor86]. Recently, *subspace*

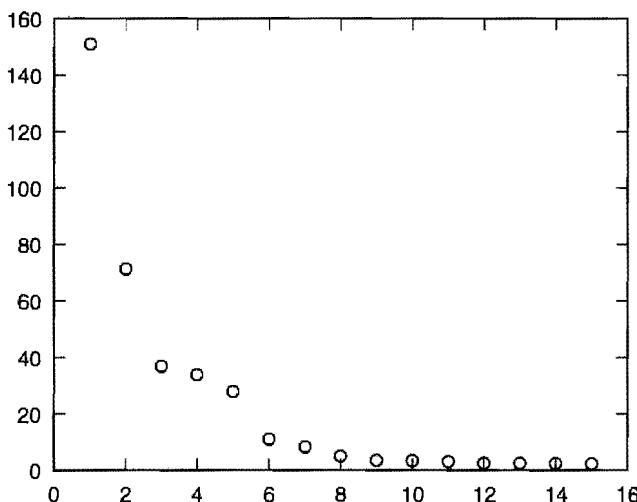


Figure 4.2 The first 15 singular values of \mathcal{H}_{250} for the example.

methods have been developed which are particularly effective for multivariable systems [vOdM96]; these are in some ways similar to the algorithm described in this section, but they can be used with arbitrary input–output data, not just with step or pulse responses.

4.2 Transfer function models

4.2.1 The basics

We will use z to denote both the ‘time advance’ operator and the complex variable used with z transforms. Since z^{-1} is the transfer function of a one-step time delay, there is no inconsistency between writing $\bar{w}(z) = z^{-1}\bar{y}(z)$ as the z transform of a delayed version of the signal $\{y(k)\}$, and writing $w(k) = z^{-1}y(k)$ to denote that $w(k) = y(k - 1)$. For most purposes the interpretation of z^{-1} as a time delay operator will be the one required. (Many texts use q^{-1} to denote the time delay operator, perhaps to emphasize this interpretation rather than the transform one.)

We will take the plant to be described by the input–output difference equation

$$A(z^{-1})y(k) = z^{-d}B(z^{-1})u(k). \quad (4.57)$$

In the case of a single-input, single-output plant $A(z^{-1})$ and $B(z^{-1})$ are the polynomials:

$$A(z^{-1}) = 1 + a_1z^{-1} + \cdots + a_nz^{-n} \quad (4.58)$$

$$B(z^{-1}) = b_0 + b_1z^{-1} + \cdots + b_nz^{-n} \quad (4.59)$$

so that (4.57) can also be written as the difference equation

$$\begin{aligned} y(k) + a_1y(k-1) + \cdots + a_ny(k-n) = \\ b_0u(k-d) + b_1u(k-d-1) + \cdots + b_nu(k-d-n) \end{aligned} \quad (4.60)$$

The input-output delay d has been ‘pulled out’ of the polynomial $B(z^{-1})$. This is convenient, but not essential.

We can also define the polynomials $\tilde{A}(z)$ and $\tilde{B}(z)$ as

$$\tilde{A}(z) = z^n A(z^{-1}) = z^n + a_1z^{n-1} + \cdots + a_n \quad (4.61)$$

$$\tilde{B}(z) = z^n B(z^{-1}) = b_0z^n + b_1z^{n-1} + \cdots + b_n \quad (4.62)$$

If we take z transforms of the input and output sequences, then we obtain the transfer function description

$$\bar{y}(z) = P(z)\bar{u}(z) = z^{-d} \frac{B(z^{-1})}{A(z^{-1})} \bar{u}(z) = z^{-d} \frac{\tilde{B}(z)}{\tilde{A}(z)} \bar{u}(z) \quad (4.63)$$

For multivariable systems, $A(z^{-1})$ and $B(z^{-1})$ are the polynomial matrices:

$$A(z^{-1}) = I_p + A_1z^{-1} + \cdots + A_nz^{-n} \quad (4.64)$$

$$B(z^{-1}) = B_0 + B_1z^{-1} + \cdots + B_nz^{-n} \quad (4.65)$$

where each A_i is a matrix of dimensions $p \times p$, and each B_i is a matrix of dimensions $p \times m$. Polynomial matrices $\tilde{A}(z)$ and $\tilde{B}(z)$ can be defined as

$$\tilde{A}(z) = z^n A(z^{-1}) \quad (4.66)$$

$$\tilde{B}(z) = z^n B(z^{-1}) \quad (4.67)$$

and we now have a *transfer function matrix* description

$$\bar{y}(z) = P(z)\bar{u}(z) = z^{-d} A(z^{-1})^{-1} B(z^{-1}) \bar{u}(z) = z^{-d} \tilde{A}(z)^{-1} \tilde{B}(z) \bar{u}(z) \quad (4.68)$$

Note that pulling out the delay d is not very useful in the MIMO case, because in general each input-output channel may have a different delay. However, setting $d = 1$ represents the assumption that the input $u(k)$ does not affect $y(k)$ — the model is ‘strictly proper’. These polynomial matrix and transfer function matrix descriptions are much less useful and convenient than in the SISO case, although in principle nearly everything can be carried over to the multivariable case. We will therefore confine ourselves almost entirely to the SISO case in this chapter.

The transfer function matrix (4.68) with $d = 1$ corresponds to the multivariable (vector) difference equation

$$\begin{aligned} y(k+1) = -A_1y(k) - A_2y(k-1) - \cdots - A_ny(k-n+1) + \\ B_1u(k) + B_2u(k-1) + \cdots + B_nu(k-n) \end{aligned} \quad (4.69)$$

where the matrices A_i and B_i are the same as those in (4.64) and (4.65). This form is quite convenient for computing predictions, even in the multivariable case. Several MPC products allow plant descriptions to be defined in this form. If a (vector) ‘white noise’ term is added to the right-hand side of (4.69) then this model corresponds to the form known in time-series analysis as *AutoRegressive with eXogenous inputs* or an ARX model. Some of the MPC products use this terminology in their documentation — see Appendix A.

It is important to be able to move between transfer-function and state-space descriptions. Suppose we have a standard state-space model:

$$x(k+1) = Ax(k) + Bu(k) \quad y(k) = Cx(k) + Du(k). \quad (4.70)$$

Take z transforms of these:

$$z\bar{x}(z) - x(0) = A\bar{x}(z) + B\bar{u}(z) \quad \bar{y}(z) = C\bar{x}(z) + D\bar{u}(z) \quad (4.71)$$

from which we have, assuming $x(0) = 0$:

$$\bar{x}(z) = (zI - A)^{-1}B\bar{u}(z) \quad (4.72)$$

and hence

$$\bar{y}(z) = [C(zI - A)^{-1}B + D]\bar{u}(z) \quad (4.73)$$

so that we have

$$P(z) = C(zI - A)^{-1}B + D \quad (4.74)$$

This holds for both SISO and multivariable systems. It can be seen that it is easy to obtain a transfer function matrix, given a state-space model. However, it is not easy to obtain the polynomial matrices $A(z^{-1})$ and $B(z^{-1})$ in the multivariable case. Going in the opposite direction, from a transfer function to a state-space model, is not so straightforward, but there are algorithms for doing so (see [Kai80], for example), which are quite reliable for SISO systems. MATLAB’s *Control System Toolbox* has the functions `ss2tf` and `tf2ss` for performing these conversions.

We can also move between transfer function models and step response or pulse response models. In fact, the transfer function is *defined* as the z transform of the pulse response, so that:

$$P(z) = \sum_{k=0}^{\infty} z^{-k} H(k) \quad (4.75)$$

$$= H(0) + z^{-1}H(1) + z^{-2}H(2) + \dots \quad (4.76)$$

$$= D + z^{-1}CB + z^{-2}CAB + \dots \quad (4.77)$$

(where the last line applies if we have a state-space model). This means that, at least in the SISO case, we can obtain the pulse response from the transfer function by ‘long division’.

Example 4.4

Consider the transfer function

$$P(z) = z^{-1} \frac{1 - 0.7z^{-1}}{1 - 1.6z^{-1} + 0.6z^{-2}} \quad (4.78)$$

$$= \frac{z - 0.7}{z^2 - 1.6z + 0.6} \quad (4.79)$$

By long division we can expand $P(z)$ as a series in z^{-1} :

$$\begin{aligned} & \frac{0(z^2 - 1.6z + 0.6) + z - 0.7}{z^2 - 1.6z + 0.6} \rightarrow 0 + \frac{z - 0.7}{z^2 - 1.6z + 0.6} \\ & \frac{z^{-1}(z^2 - 1.6z + 0.6) + 0.9 - 0.6z^{-1}}{z^2 - 1.6z + 0.6} \rightarrow 0 + 1z^{-1} + \frac{0.9 - 0.6z^{-1}}{z^2 - 1.6z + 0.6} \\ & \frac{0.9z^{-2}(z^2 - 1.6z + 0.6) + 0.84z^{-1} - 0.54z^{-2}}{z^2 - 1.6z + 0.6} \rightarrow 0 + 1z^{-1} + 0.9z^{-2} + \dots \end{aligned}$$

and so on. Hence we have the pulse response $H(0) = 0$, $H(1) = 1$, $H(2) = 0.9$,

We can also see from (4.75) that a finite pulse response corresponds to a transfer function with all its poles at 0.

Example 4.5

If $H(0) = 3$, $H(1) = -2$, $H(2) = 1$, $H(k) = 0$ for $k > 2$, then

$$P(z) = 3 - 2z^{-1} + 1z^{-2} = \frac{3z^2 - 2z + 1}{z^2}$$

4.2.2 Prediction using transfer functions

We can rewrite (4.60) as

$$y(k) = -a_1 y(k-1) - \dots - a_n y(k-n) + b_0 u(k-d) + \dots + b_n u(k-d-n) \quad (4.80)$$

and we can use this as the basis for prediction, if $d \geq 1$. (Recall that we assume $u(k)$ is not yet known when we predict $\hat{y}(k+i|k)$.)

The obvious way of predicting the output is as follows:

$$\hat{y}(k+1|k) = -\sum_{j=1}^n a_j y(k+1-j) + \sum_{j=0}^n b_j \tilde{u}(k-d-j) \quad (4.81)$$

$$\begin{aligned} \hat{y}(k+2|k) &= -a_1 \hat{y}(k+1|k) - \sum_{j=2}^n a_j y(k+2-j) + \sum_{j=0}^n b_j \tilde{u}(k+1-d-j) \\ &\vdots \end{aligned} \quad (4.82)$$

$$(4.83)$$

or, in general,

$$\tilde{y}(k+i|k) = -\sum_{j=1}^n a_j \tilde{y}(k-j) + \sum_{j=0}^n b_j \tilde{u}(k-d-j) \quad (4.84)$$

or

$$A(z^{-1})\tilde{y}(k+i) = z^{-d}B(z^{-1})\tilde{u}(k+i) \quad (4.85)$$

where

$$\tilde{u}(\ell) = \begin{cases} u(\ell) & \text{if } \ell < k \\ \hat{u}(\ell|k) & \text{if } \ell \geq k \end{cases} \quad (4.86)$$

and

$$\tilde{y}(\ell) = \begin{cases} y(\ell) & \text{if } \ell \leq k \\ \hat{y}(\ell|k) & \text{if } \ell > k \end{cases} \quad (4.87)$$

(We referred to this earlier as ‘realigning’ the model — see Sections 1.6 and 2.6.4.) Since the prediction $\hat{y}(k+i|k)$ depends on some other predicted outputs, which are themselves obtained from measurements of past outputs, it is possible to find an expression for $\hat{y}(k+i|k)$ which depends only on the measured outputs $y(k), y(k-1), \dots$ (and of course, on actual past inputs and predicted future inputs). This saves on computation, which is particularly important in applications to adaptive control, where the predictors — as well as the predictions themselves — have to be computed on-line. It also gives some insight into the way the predictor design affects the predictive controller.

Suppose we have a polynomial $E_i(z^{-1})$, of degree no greater than $i-1$ (i is a positive integer), and a polynomial $F_i(z^{-1})$, of degree $n-1$, such that

$$\frac{1}{A(z^{-1})} = E_i(z^{-1}) + z^{-i} \frac{F_i(z^{-1})}{A(z^{-1})} \quad (4.88)$$

or

$$E_i(z^{-1})A(z^{-1}) = 1 - z^{-i}F_i(z^{-1}) \quad (4.89)$$

Multiplying (4.85) by $E_i(z^{-1})$ gives

$$[1 - z^{-i}F_i(z^{-1})]\tilde{y}(k+i) = z^{-d}E_i(z^{-1})B(z^{-1})\tilde{u}(k+i) \quad (4.90)$$

or

$$\tilde{y}(k+i) = z^{-i}F_i(z^{-1})\tilde{y}(k+i) + z^{-d}E_i(z^{-1})B(z^{-1})\tilde{u}(k+i) \quad (4.91)$$

Now notice that $z^{-i}\tilde{y}(k+i)$ is just $y(k)$, and hence $z^{-i}F_i(z^{-1})\tilde{y}(k+i)$ involves only measured values of past outputs. So we can write the predicted output as

$$\tilde{y}(k+i|k) = F_i(z^{-1})y(k) + z^{-d}E_i(z^{-1})B(z^{-1})\tilde{u}(k+i) \quad (4.92)$$

A Diophantine equation is one of the form

$$A(z^{-1})X(z^{-1}) + B(z^{-1})Y(z^{-1}) = C(z^{-1}) \quad (4.93)$$

$$\text{or} \quad X(z^{-1})A(z^{-1}) + Y(z^{-1})B(z^{-1}) = C(z^{-1}) \quad (4.94)$$

where all the variables are either polynomials or polynomial matrices. We will consider only the scalar case, when they are all polynomials, in which case these two equations are the same. In these equations, $A(z^{-1})$, $B(z^{-1})$ and $C(z^{-1})$ are assumed to be known, while $X(z^{-1})$ and $Y(z^{-1})$ are unknown polynomials.

If X_0 and Y_0 is a pair of solutions of the Diophantine equation, then it is clear that $X = X_0 - B_2 P$ and $Y = Y_0 + A_2 P$ is also a pair of solutions, if A_2 and B_2 are polynomials such that $B_2/A_2 = B/A$, and P is any polynomial. This follows since

$$A(X_0 - B_2 P) + B(Y_0 + A_2 P) = AX_0 + BY_0 + (BA_2 - AB_2)P = C$$

So it is clear that the solution (pair) of a Diophantine equation is not unique. However, a solution in which either X or Y has the smallest possible degree is unique, and can be obtained as follows. Find polynomials Q and Γ such that $Y_0 = A_2 Q + \Gamma$. (This is itself a Diophantine equation, but simpler than the general one.) These can be obtained by ‘long division’: Q is the quotient, and Γ the remainder, of the division Y_0/A_2 . The degree of Γ is necessarily smaller than the degree of A_2 . Hence we have $Y = Y_0 + A_2 P = A_2(Q + P) + \Gamma$. Now take $P = -Q$ to make the degree of Y as small as possible. Then we have the unique solution pair $X = X_0 + B_2 Q$, $Y = \Gamma$. Similarly, a solution pair in which either X or Y has a specified degree is unique.

An important special case of Diophantine equations arises if $C(z^{-1}) = 1$, in which case the equation is also known as the *Bezout identity*. This has a solution if and only if A and B are coprime — have no common factors. For suppose that they did have a common factor D , so that $A = PD$ and $B = QD$. Then the Bezout identity would be $D(PX + QY) = 1$, which is impossible unless $D = 1$. Because of this, the Bezout identity plays a large role in the theory of feedback stability, and in system theory.

When used for predicting stochastic processes we usually have $B(z^{-1}) = z^{-i}$ when finding an i -step-ahead prediction. Rewriting the Diophantine equation as

$$X_i(z^{-1}) + \frac{z^{-i}}{A(z^{-1})} Y_i(z^{-1}) = \frac{C(z^{-1})}{A(z^{-1})} \quad (4.95)$$

and imposing the requirement that X_i be of degree $i - 1$, shows that the coefficients in X_i are the first i terms in the pulse response of the transfer function C/A . If both A and C have degree n , then Y_i must have degree $n - 1$. For more details see [ÅW84]. There is an efficient recursive algorithm for computing the coefficients in X_i and Y_i for a sequence of values of i [Soe92].

Mini-Tutorial 5 Diophantine equations.

which does not involve any predicted outputs on the right-hand side of the equation. All this depends on having solutions $E_i(z^{-1})$ and $F_i(z^{-1})$ to equation (4.89). This is an example of a *Diophantine equation*, and much theory exists about solving such equations — see Mini-Tutorial 5.

There is an interesting interpretation that can be made of equation (4.92). Multiply both sides of (4.88) by $B(z^{-1})$:

$$E_i(z^{-1})B(z^{-1}) = \frac{B(z^{-1})}{A(z^{-1})} - z^{-i} \frac{F_i(z^{-1})B(z^{-1})}{A(z^{-1})} \quad (4.96)$$

Substituting this into (4.92) we get

$$\hat{y}(k+i|k) = F_i(z^{-1})y(k) + z^{-d} \left[\frac{B(z^{-1})}{A(z^{-1})} - z^{-i} \frac{F_i(z^{-1})B(z^{-1})}{A(z^{-1})} \right] \tilde{u}(k+i) \quad (4.97)$$

$$= z^{-d} \frac{B(z^{-1})}{A(z^{-1})} \tilde{u}(k+i) + F_i(z^{-1}) \left[y(k) - z^{-d} \frac{B(z^{-1})}{A(z^{-1})} u(k) \right] \quad (4.98)$$

This shows that the prediction has a ‘predictor–corrector’ structure. This is different from the predictor–corrector structure used for the observer in Chapter 2, however. Here the predictions $z^{-d}(B/A)\tilde{u}(k+i)$ and $z^{-d}(B/A)u(k)$ are ‘long-term’ predictions made on the basis of input signals only, not corrected by output measurements at any time. In Chapter 2 the observer predicts only one step ahead, on the basis of output measurement $y(k-1)$, before it is corrected by the next measurement, $y(k)$. Nevertheless, we will be able to relate this prediction to predictions obtained by a state observer in a later subsection.

4.2.3 Prediction with a disturbance model

In this section we will assume that the plant has an unmeasured output disturbance, which is modelled as some other signal passed through a filter with transfer function $C(z^{-1})/D(z^{-1})$:

$$y(k) = z^{-d} \frac{B(z^{-1})}{A(z^{-1})} u(k) + d(k) \quad (4.99)$$

$$d(k) = \frac{C(z^{-1})}{D(z^{-1})} v(k) \quad (4.100)$$

in which we assume that

$$C(z^{-1}) = 1 + c_1 z^{-1} + \dots + c_v z^v \quad (4.101)$$

$$D(z^{-1}) = 1 + d_1 z^{-1} + \dots + d_v z^v \quad (4.102)$$

Note that we can take both $C(z^{-1})$ and $D(z^{-1})$ to be *monic* (their leading coefficients are 1), because the magnitude of $v(k)$ can be adjusted to compensate for this if necessary.

This model is general enough to allow both deterministic and stochastic disturbances to be modelled, as well as mixtures of these. Deterministic disturbances are usually modelled by taking $C(z^{-1}) = 1$ ($c_i = 0$) and making only the first few values of $v(k)$ non-zero.

Example 4.6

Constant output disturbance

If we take $C(z^{-1}) = 1$, $D(z) = 1 - z^{-1}$, and $v(0) = v_0$, $v(k) = 0$ for $k > 0$, then we get

$$d(k) - d(k-1) = v(k) \quad (4.103)$$

and hence $d(0) = v_0, d(1) = v_0, d(2) = v_0, \dots$. That is, we have modelled a constant output disturbance of unspecified magnitude v_0 . So this would be equivalent to the disturbance model assumed by the *DMC scheme*.

Example 4.7

Sinusoidal disturbance

To model a sinusoidal output disturbance of known frequency ω_0 , but unknown amplitude and phase, we can take $C(z^{-1}) = 1$ and

$$D(z^{-1}) = (1 - z^{-1}e^{j\omega_0 T_s})(1 - z^{-1}e^{-j\omega_0 T_s}) \quad (4.104)$$

$$= 1 - 2 \cos(\omega_0 T_s)z^{-1} + z^{-2} \quad (4.105)$$

where T_s is the sampling interval, and $v(0) = v_0, v(1) = v_1$. Then the z transform of $d(k)$ is given by

$$\bar{d}(z) = \frac{v_0 + v_1 z^{-1}}{1 - 2 \cos(\omega_0 T_s)z^{-1} + z^{-2}} \quad (4.106)$$

which is the z transform of a signal of the form

$$d(k) = A \cos(\omega_0 T_s k + \phi) \quad (4.107)$$

To model a stationary, zero-mean stochastic disturbance we can take $v(k)$ to be a ‘white noise’ process, namely $E\{v(k)^2\} = \sigma^2, E\{v(k)v(k-\ell)\} = 0$ for $\ell \neq 0$, the probability distribution of $v(k)$ being the same for all k , and each $v(k)$ being independent of $v(\ell)$ if $\ell \neq k$. Then, if $C(z^{-1})/D(z^{-1})$ is an asymptotically stable transfer function, $d(k)$ will be a stationary process with spectral density

$$\Phi_{dd}(\omega) = \sigma^2 \frac{|C(e^{-j\omega T_s})|^2}{|D(e^{-j\omega T_s})|^2}. \quad (4.108)$$

Note that since $|C(e^{-j\omega T_s})|^2 = C(e^{-j\omega T_s})C(e^{+j\omega T_s})$, it is always possible to choose $C(z^{-1})$ such that all its roots lie inside the unit disc — that is, without restricting the spectral densities which can be modelled in this way. Also, for the same reason, factors of z^{-j} in $C(z^{-1})$ do not affect the spectral density.

Example 4.8

Disturbances due to crew movements on board a spacecraft can be modelled approximately as a stochastic process with a spectral density which has a peak value at $\omega_0 = 0.12$ rad/sec, and becomes zero at low frequencies. With a sampling interval $T_s = 0.6$ sec, we have $\omega_0 T_s = 0.072$ rad. Hence choose

$$\frac{C(z^{-1})}{D(z^{-1})} = \frac{1 - z^{-1}}{(1 - \rho e^{-j0.072}z^{-1})(1 - \rho e^{+j0.072}z^{-1})} \quad (4.109)$$

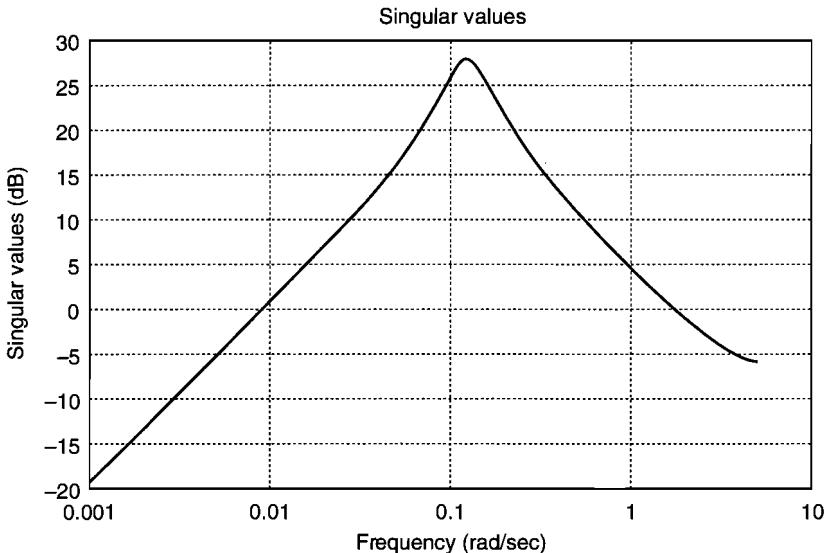


Figure 4.3 Spectral density of crew disturbance, modelled by (4.109) with $\rho = 0.98$.

with $\rho < 1$ determined by the sharpness of the peak. Figure 4.3 shows the spectral density for $\rho = 0.98$ and $\sigma = 1$.

(Note: This model of crew disturbances is probably more appropriate as a model of input disturbances (e.g. torques) than of output disturbances — but it illustrates the point. The details may not be correct for real crew disturbance spectra.)

When a disturbance model of the form (4.99) — (4.100) is present, solutions of a Diophantine equation can again be used to obtain the predicted outputs. Now we assume that we have polynomials $E'_i(z^{-1})$ and $F'_i(z^{-1})$ which solve the equation

$$\frac{C(z^{-1})}{D(z^{-1})} = E'_i(z^{-1}) + z^{-i} \frac{F'_i(z^{-1})}{D(z^{-1})} \quad (4.110)$$

with $E'_i(z^{-1})$ of degree at most $i - 1$, and $F'_i(z^{-1})$ of degree at most $v - 1$. That is, these polynomials solve the Diophantine equation

$$E'_i(z^{-1})D(z^{-1}) = C(z^{-1}) - z^{-i}F'_i(z^{-1}). \quad (4.111)$$

Note that in this case the first term on the right-hand side is the polynomial $C(z^{-1})$, whereas in (4.89) it was 1. This difference arises because the plant input u is known, whereas the disturbance v is not. Using (4.110) in (4.100) we get

$$\hat{d}(k+i|k) = \left[E'_i(z^{-1}) + z^{-i} \frac{F'_i(z^{-1})}{D(z^{-1})} \right] \hat{v}(k+i|k) \quad (4.112)$$

$$= E'_i(z^{-1})\hat{v}(k+i|k) + \frac{F'_i(z^{-1})}{D(z^{-1})}\hat{v}(k|k) \quad (4.113)$$

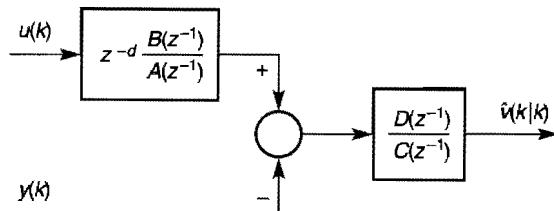


Figure 4.4 Generation of $\hat{v}(k|k)$ — one interpretation.

Now the difference from our previous use of the Diophantine equation is that although we have split the prediction up into ‘future’ and ‘past’ terms, in this case we do not know the ‘past’ process $\{v(k)\}$. This has to be estimated somehow.

From (4.99) and (4.100) we have

$$y(k) = z^{-d} \frac{B(z^{-1})}{A(z^{-1})} u(k) + \frac{C(z^{-1})}{D(z^{-1})} v(k) \quad (4.114)$$

Consequently we can estimate $v(k)$ as

$$\hat{v}(k|k) = \frac{D(z^{-1})}{C(z^{-1})} \left[y(k) - z^{-d} \frac{B(z^{-1})}{A(z^{-1})} u(k) \right] \quad (4.115)$$

$$= \frac{D(z^{-1})}{C(z^{-1})} [y(k) - \hat{y}(k)] \quad (4.116)$$

where $\hat{y}(k)$ is the output prediction obtained by filtering the input u through the model of the plant. Here we see the importance of ensuring that $C(z^{-1})$ has all its roots inside the unit disc — if it did not, this estimator would be unstable.

There is an ambiguity in the interpretation of (4.115). We can interpret it as meaning that $\hat{v}(k|k)$ should be generated as shown in Figure 4.4. In this case $\hat{y}(k)$ is a ‘long-term’ prediction of $y(k)$, obtained without correction by actual measured past values. This is corrected at only one point in time, namely k , by the measurement $y(k)$, and the corrected value is filtered further by D/C .

An alternative interpretation is obtained by multiplying through by $C(z^{-1})A(z^{-1})$, which gives

$$A(z^{-1})C(z^{-1})\hat{v}(k|k) = A(z^{-1})D(z^{-1})y(k) - z^{-d}B(z^{-1})D(z^{-1})u(k) \quad (4.117)$$

This can be solved as a ‘realigned’ difference equation, with the interpretation that $z^{-i}\hat{v}(k|k) = \hat{v}(k-i|k-i)$. (Which is not the only possible one — for example one could use the interpretation $z^{-i}\hat{v}(k|k) = \hat{v}(k-i|k)$, using some ‘smoothed’ estimate of $v(k-i)$.) These two interpretations are not the same; different estimates are obtained. The interpretation shown in Figure 4.4 seems to be more ‘correct’, while that of (4.117) seems more sensible, since more information is being used, because past measurements $y(k), y(k-1), \dots$ are being used at time k , rather than $y(k)$ only. In practice the difference equation (realigned model) interpretation is the one used, and that is what we shall assume.

Now we have for the ‘ i -step-ahead’ predicted output:

$$\hat{y}(k+i|k) = z^{-d} \frac{B(z^{-1})}{A(z^{-1})} \tilde{u}(k+i) + E'_i(z^{-1}) \hat{v}(k+i|k) + \frac{F'_i(z^{-1})}{D(z^{-1})} \hat{v}(k|k) \quad (4.118)$$

$$\begin{aligned} &= z^{-d} \frac{B(z^{-1})}{A(z^{-1})} \tilde{u}(k+i) + E'_i(z^{-1}) \hat{v}(k+i|k) \\ &\quad + \frac{F'_i(z^{-1})}{C(z^{-1})} \left[y(k) - z^{-d} \frac{B(z^{-1})}{A(z^{-1})} u(k) \right] \end{aligned} \quad (4.119)$$

where we have used (4.117) to get the second equality.

Before getting too alarmed at this expression, the reader should note that usually some simplifications are introduced, as for example in the next subsection.

This prediction includes the term $E'_i(z^{-1}) \hat{v}(k+i|k)$, which contains only predictions of future values of $v(k)$. The best way of making these predictions should depend on the assumptions being made about the nature of the disturbance. However, it is not sensible to assume that $v(k)$ is a signal which is predictable in any way — because if that was the case, then such a belief should already have been reflected in the choice of the C and D polynomials. If we believe that the disturbance $d(k)$ is deterministic, then that usually corresponds to the belief that $v(k) = 0$, as we saw in the earlier examples. In such a case the estimate $\hat{v}(k|k)$ serves as an ‘initial condition’ (together with the previous one or two estimates, perhaps), but it is then appropriate to assume that $\hat{v}(k+i|k) = 0$ for $i > 0$.

On the other hand, if the disturbance is assumed to be stochastic, then one can consider predicting $v(k+i)$ in such a way as to optimize the prediction $\hat{y}(k+i|k)$ in some way. The usual choice at this point is to go for a *minimum variance* prediction. The idea is to form the prediction in such a way that the variance of the error between the predicted and the actual values, $E\{|y(k+i) - \hat{y}(k+i|k)|^2\}$ is as small as possible. But there is a theorem in probability theory which says that the minimum variance prediction is obtained by choosing $\hat{y}(k+i|k) = E\{y(k+i)|k\}$, namely the conditional mean value, the conditioning being on all the information available at time k . (See [Mos95, ÅW84] for more details.) Now the conditional mean of a sum of terms is the sum of the conditional means, so we should choose $\hat{v}(k+i) = E\{v(k+i)|k\}$. But in the stochastic case $\{v(k)\}$ is assumed to be a zero-mean white-noise process, so its conditional mean is the same as its unconditional mean, namely 0, so we set $\hat{v}(k+i) = 0$. Thus finally we have the *minimum variance predictor*:

$$\begin{aligned} \hat{y}(k+i|k) &= z^{-d} \frac{B(z^{-1})}{A(z^{-1})} \tilde{u}(k+i) \\ &\quad + \frac{F'_i(z^{-1})}{C(z^{-1})} \left[y(k) - z^{-d} \frac{B(z^{-1})}{A(z^{-1})} u(k) \right] \end{aligned} \quad (4.120)$$

For the purpose of solving the predictive control problem, it is necessary to separate out the ‘free response’ part of this — the predicted response which would occur if $\Delta\hat{u}(k+i|k) = 0$ — from the ‘forced response’, namely that part which depends on $\Delta\hat{u}(k+i|k)$. In other words, it is necessary to separate out the part which depends on

the past from that which depends on future inputs. This can be done by solving another Diophantine equation.

Since the only signal which crosses over the ‘past–future’ boundary in the prediction (4.120) is \tilde{u} , and that is filtered by the transfer function $z^{-d}B/A$, we need to extract the first $i - d + 1$ terms of the pulse response of B/A when finding $\hat{y}(k + i|k)$. (Only $i - d + 1$ because the signal is delayed by d steps, so we do not need to consider inputs which occur later than $k + i - d$. Our ‘past–future’ boundary is between $k - 1$ and k .) Thus we need to find the solution pair $(E_i(z^{-1}), F_i(z^{-1}))$ of the Diophantine equation

$$\frac{B(z^{-1})}{A(z^{-1})} = E_i(z^{-1}) + z^{-(i-d)} \frac{F_i(z^{-1})}{A(z^{-1})} \quad (4.121)$$

with the degree of E_i no bigger than $i - d$. (Note that this is not quite the same as (4.88).) Substituting this into (4.120) we get

$$\begin{aligned} \hat{y}(k + i|k) &= z^{-d} \left[E_i(z^{-1}) + z^{-(i-d)} \frac{F_i(z^{-1})}{A(z^{-1})} \right] \tilde{u}(k + i) + \\ &\quad \frac{F'_i(z^{-1})}{C(z^{-1})} \left[y(k) - z^{-d} \frac{B(z^{-1})}{A(z^{-1})} u(k) \right] \\ &= \underbrace{E_i(z^{-1}) \hat{u}(k + i - d)}_{\text{future}} + \end{aligned} \quad (4.122)$$

$$\underbrace{\frac{F_i(z^{-1})}{A(z^{-1})} u(k - 1) + \frac{F'_i(z^{-1})}{C(z^{-1})} \left[y(k) - z^{-d} \frac{B(z^{-1})}{A(z^{-1})} u(k) \right]}_{\text{past}} \quad (4.123)$$

$$\begin{aligned} &= \underbrace{E_i(z^{-1}) \hat{u}(k + i - d)}_{\text{future}} + \\ &\quad \underbrace{\frac{F'_i(z^{-1})}{C(z^{-1})} y(k) + \frac{z^{-1} F_i(z^{-1}) C(z^{-1}) - z^{-d} F'_i(z^{-1}) B(z^{-1})}{A(z^{-1}) C(z^{-1})} u(k)}_{\text{past}} \end{aligned} \quad (4.124)$$

In order to express the set of output predictions $\hat{y}(k + i|k)$ for $i = 1, \dots, H_p$ in a matrix–vector notation similar to that of (2.23), it is necessary to compute the filtered signals $(1/C)y$ and $(1/AC)u$, which can be done by computing one new term of each signal at each step. Since each prediction (that is, for each i) depends on polynomial operations on these filtered signals, it is a linear combination of past values of them, and hence the whole set of predictions can be given a matrix–vector representation. For details see [Soe92, BGW90, Mos95, CC95, CC99]. There are several ways of arranging these computations.

Note that using the minimum variance predictions does not necessarily lead to a minimum mean value of the cost function (2.9), in the stochastic case. But that is what is generally done; as in the state-space case, this is in general a heuristic application of the ‘separation’ or ‘certainty equivalence’ principle.

Example 4.9

Suppose that

$$\begin{aligned} A(z^{-1}) &= 1 - 0.9z^{-1}, & B(z^{-1}) &= 0.5 \\ C(z^{-1}) &= 1 + 0.5z^{-1}, & D(z^{-1}) &= 1 - z^{-1} \\ d &= 1 \end{aligned}$$

Solution of (4.110): $E'_1(z^{-1})$ is to be of degree 0, namely a constant. We have

$$\frac{C(z^{-1})}{D(z^{-1})} = \frac{1 - z^{-1} + 1.5z^{-1}}{1 - z^{-1}} = 1 + \frac{1.5z^{-1}}{1 - z^{-1}}$$

so

$$E'_1 = 1, \quad F'_1 = 1.5$$

Solution of (4.121):

$$\frac{B(z^{-1})}{A(z^{-1})} = \frac{0.5(1 - 0.9z^{-1}) + 0.45z^{-1}}{1 - 0.9z^{-1}} = 0.5 + \frac{0.45z^{-1}}{1 - 0.9z^{-1}}$$

so

$$E_1 = 0.5, \quad F_1 = 0.45$$

4.2.4 The GPC model

In the case of *Generalized Predictive Control*, or *GPC*, the disturbance is usually assumed to be stochastic, and the denominator polynomial $D(z^{-1})$ which appears in (4.100) is always

$$D(z^{-1}) = (1 - z^{-1})A(z^{-1}) \quad (4.125)$$

Note that although a transfer function description of the plant is being used, difficulties will arise if the stochastic interpretation is adopted for the disturbance and the plant is unstable. Even with a stable plant, the disturbance is not stationary if this model is used, because $D(z^{-1})$ has a root at 1 — this implies that, even if $v(k)$ is white noise, the disturbance $d(k)$ will be a ‘random walk’. A justification for this choice of disturbance model is provided by the following, non-stochastic, argument. If disturbances enter ‘inside’ the plant, rather than at its output — think of disturbances entering the state equations, if we had any — then they will appear at the output of the plant as if they had been filtered through a transfer function of the form $C(z^{-1})/A(z^{-1})$. Since most realistic disturbances do in fact arise in this way, it seems reasonable to include $A(z^{-1})$ as a factor of $D(z^{-1})$. Including the factor $(1 - z^{-1})$ in $D(z^{-1})$ allows constant disturbances to be represented, which is a rather common and important case. Furthermore, allowing $v(k)$ to consist of pulses occurring at random times leads to a disturbance which is piecewise-constant and jumps at random times. This is often a realistic representation of phenomena such as load disturbances in a plant — for example, a sudden increase in

load torque, which reduces the effective torque available from a controller. Also, and very importantly, including the factor $(1 - z^{-1})$ leads to the controller having ‘integral action’. One could make the disturbance model more elaborate, but using (4.125) leads to some nice interpretations and keeps things relatively simple.

Returning to (4.117), substituting (4.100), and cancelling $A(z^{-1})$ from each term gives

$$C(z^{-1})\hat{v}(k|k) = A(z^{-1})(1 - z^{-1})y(k) - z^{-d}B(z^{-1})(1 - z^{-1})u(k) \quad (4.126)$$

and using the notation $\Delta y(k) = (1 - z^{-1})y(k)$ — analogously to our use of $\Delta u(k)$ — gives

$$C(z^{-1})\hat{v}(k|k) = A(z^{-1})\Delta y(k) - z^{-d}B(z^{-1})\Delta u(k) \quad (4.127)$$

from which $\hat{v}(k|k)$ is computed. Then the ‘minimum variance’ output prediction is obtained as

$$\hat{y}(k+i|k) = z^{-d} \frac{B(z^{-1})}{A(z^{-1})} \tilde{u}(k+i) + \frac{C(z^{-1})}{A(z^{-1})(1 - z^{-1})} \hat{v}(k+i|k) \quad (4.128)$$

$$= z^{-d} \frac{B(z^{-1})}{A(z^{-1})} \tilde{u}(k+i) + \frac{F'_i(z^{-1})}{A(z^{-1})(1 - z^{-1})} \hat{v}(k|k) \quad (4.129)$$

where we have put $\hat{v}(k+i|k) = 0$ for $i > 0$ to get the second equality. Substituting in for $\hat{v}(k|k)$ from (4.127) gives the same equation as (4.120).

4.2.5 State-space interpretations

Recall that in Chapter 2 we had the following equation, which describes the evolution of the estimated state in an observer:

$$\hat{x}(k+1|k) = (A - LC_y)\hat{x}(k|k-1) + Bu(k|k) + Ly(k) \quad (4.130)$$

Taking z transforms, we get

$$\bar{\hat{x}}(z) = [zI - (A - LC_y)]^{-1}[L\bar{y}(z) + B\bar{u}(z)] \quad (4.131)$$

All further predictions are based on $\hat{x}(k+1|k)$ — for example, $\hat{x}(k+i|k) = A^{i-1}\hat{x}(k+1|k) + \dots$. So we see that the output measurements $y(k)$ get filtered by $[zI - (A - LC_y)]^{-1}L$ before being used to generate predictions. But in the transfer function approach we can see, for example from equation (4.120) or (4.124), that the output measurements get filtered by $1/C(z^{-1})$ before being used to generate predictions. Recalling that the inverse of a matrix is given by the formula $X^{-1} = \text{adj}X / \det X$, we see that we have the following correspondence between the state-space and the transfer function formulations:

$$z^v C(z^{-1}) \leftrightarrow \det[zI_v - (A - LC_y)] \quad (4.132)$$

(Recall that factors of powers of z in $C(z^{-1})$ have no effect on the spectral density of the disturbance.)

We have the following consequences of this:

- The polynomial $C(z^{-1})$ (or $z^v C(z^{-1})$) in the transfer function formulation is often called the *observer polynomial*. (This is true also in the ‘pole-placement’ literature.)
- Using the state-space formulation we can augment the model with additional states to represent the effects of disturbances. The dynamics of these states correspond to the polynomial $D(z^{-1})$ in the transfer function formulation, and we can choose the observer gain L such that the *observer dynamics* match any desired numerator polynomial $C(z^{-1})$.
- We do not have to believe that there is a real stochastic disturbance. In either approach we can regard the polynomial $C(z^{-1})$ or the observer gain L as ‘tuning parameters’, which are chosen to give the predictive controller desirable performance characteristics. Actually, the same is true of the denominator polynomial $D(z^{-1})$, as is made evident by the *GPC* model, for instance.
- When dealing with one signal at a time, it is often easier to think in terms of transfer functions than in terms of state-space models. Even in multivariable systems, it is often enough to think of disturbances on each output being independent of each other. In such cases some designers seem to find it easier to think in terms of ‘ $C(z^{-1})$ and $D(z^{-1})$ ’, but to compute and implement using state-space methods.

Example 4.10

Returning to the crew disturbance example (Example 4.8), there we had

$$C(z^{-1}) = 1 - z^{-1} \quad D(z^{-1}) = (1 - \rho e^{-j\omega_0 T_s})(1 - \rho e^{+j\omega_0 T_s})$$

A state-space model equivalent to this is:

$$x_d(k+1) = A_d x_d(k) + B_d v(k) \quad d(k) = C_d x_d(k)$$

where

$$A_d = \begin{bmatrix} 0 & 1 \\ -\rho^2 & 2\rho \cos(\omega_0 T_s) \end{bmatrix} \quad B_d = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad C_d = [-1, 1]$$

(This actually gives $C(z^{-1}) = z^{-1}(1 - z^{-1})$.) We can combine this with a plant model

$$x_p(k+1) = A_p x_p(k) + B_p u(k) \quad y(k) = C_p x_p(k)$$

as follows:

$$\begin{bmatrix} x_p(k+1) \\ x_d(k+1) \end{bmatrix} = \begin{bmatrix} A_p & 0 \\ 0 & A_d \end{bmatrix} \begin{bmatrix} x_p(k) \\ x_d(k) \end{bmatrix} + \begin{bmatrix} B_p \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ B_d \end{bmatrix} v(k) \quad (4.133)$$

$$y(k) = [C_p, C_d] \begin{bmatrix} x_p(k) \\ x_d(k) \end{bmatrix} \quad (4.134)$$

Partitioning the observer gain matrix as $L = [L_p^T, L_d^T]^T$, the complete observer gain matrix becomes

$$\mathcal{A} - LC = \begin{bmatrix} A_p - L_p C_p & -L_p C_d \\ -L_d C_p & A_d - L_d C_d \end{bmatrix} \quad (4.135)$$

Assuming the plant is stable, a possible choice is $L_p = 0$ — that is, leave the open-loop plant dynamics as part of the observer dynamics, in which case we get

$$\mathcal{A} - LC = \begin{bmatrix} A_p & 0 \\ -L_d C_y & A_d - L_d C_d \end{bmatrix} \quad (4.136)$$

and we can choose L_d to get any eigenvalues we like for $A_d - L_d C_d$, providing that the pair (A_d, C_d) is observable. It is easy to see that we can place the eigenvalues of $A_d - L_d C_d$ at 1 and 0 — the roots of $z^{-1}C(z^{-1})$ — by choosing $L_d = [1, 2\rho \cos(\omega_0 T_s)]^T$ for the crew disturbance example, because in this case we have

$$A_d - L_d C_d = \begin{bmatrix} 1 & 0 \\ -\rho^2 + 2\rho \cos(\omega_0 T_s) & 0 \end{bmatrix}$$

Why did we set $L_p = 0$? Because if the transfer function model is written in the common denominator form

$$y(k) = \frac{z^{-d} D(z^{-1}) B(z^{-1}) u(k) + A(z^{-1}) C(z^{-1}) v(k)}{A(z^{-1}) D(z^{-1})}$$

then it is clear that the open-loop plant pole polynomial should be included in the observer dynamics.

This shows how we can recover the disturbance model using the state-space formulation. It also shows, however, that there is more design freedom with the state-space formulation. For example, there is no need to choose $L_p = 0$; other choices are possible. In particular, with the state-space formulation it is possible to choose the observer dynamics independently of the numerator polynomial of the disturbance model. Whether this additional freedom is actually useful for the designer is debatable. If one has a physically based disturbance model, such as one for crew motions, then it probably is useful, because $C(z^{-1})$ represents known physical characteristics. This is particularly true if one also has a statistical description of measurement noise, because then the observer gain matrix L can be chosen optimally, using Kalman filter theory.

There is another connection that can be made between the transfer function and state-space formulations, through Kalman filter theory. It is known that whatever combination of state and output noises act on a plant, a disturbance with the same spectral density at the output can be obtained from the model

$$x_d(k+1) = A_d x(k) + L_d v(k) \quad (4.137)$$

$$y(k) = C_d x(k) + v(k) \quad (4.138)$$

which is known as the ‘innovations’ representation of the disturbance [Mos95]. In this model, L_d is the Kalman filter gain that should be used in an observer, to estimate the state optimally. Applying the usual formula (4.74) for obtaining a transfer function from a state-space model, we see that

$$\bar{y}(z) = [C_d(zI - A_d)^{-1} L_d + I] \bar{v}(z) \quad (4.139)$$

from which we make the association $z^v D(z^{-1}) \rightarrow \det(zI - A_d)$, and we see that $C(z^{-1})$ is the polynomial whose roots are the same as the roots of $\det[C_d(zI - A_d)^{-1} L_d + I]$. In

the single-output case this is just the polynomial $[C_d(zI - A_d)^{-1}L_d + 1]\det(zI - A_d)$. This seems to be different from the association obtained in (4.132) above. But, using the *matrix inversion lemma*,² it can be shown that

$$[zI - (A_d - L_d C_d)]^{-1} = (zI - A_d)^{-1}[L_d C_d(zI - A_d)^{-1} + I]^{-1} \quad (4.140)$$

from which it follows easily (using $(I + XY)^{-1}X = X(I + YX)^{-1}$) that

$$[zI - (A_d - L_d C_d)]^{-1}L_d = (zI - A_d)^{-1}L_d[C_d(zI - A_d)^{-1}L_d + I]^{-1} \quad (4.141)$$

The denominators on the two sides of the equation are $\det[zI - (A_d - L_d C_d)]$ on the left, and $\det[C_d(zI - A_d)^{-1}L_d + I]$ on the right ($\det(zI - A_d)$ cancels on the right with the numerator). Thus we see that there is no contradiction in the two associations we have made for $C(z^{-1})$.

There are usually several ways of modelling a disturbance in the state-space formulation, which are all equivalent to a single output disturbance. If the GPC disturbance model is adopted ($D(z^{-1}) = (1 - z^{-1})A(z^{-1})$), most of its dynamics are shared with those of the plant. It is only necessary to add m states, therefore (where m is the number of outputs), in order to model the additional disturbance poles at $+1$:

$$x_p(k+1) = A_p x_p(k) + x_d(k) + B_p u(k) \quad (4.142)$$

$$x_d(k+1) = x_d(k) + B_d v_2(k) \quad (4.143)$$

$$y(k) = [C_p \quad C_d] \begin{bmatrix} x_p(k) \\ x_d(k) \end{bmatrix} \quad (4.144)$$

We will now show how a state-space model and observer can be defined which correspond precisely to the GPC model, and to the use of a ‘realigned’ model. That is, we shall extend the development of Section 2.6.4 to the case when there is a disturbance which is modelled as in GPC. Using similar notation to that used in equation (4.127) we can write the GPC model in the form

$$A(z^{-1})\Delta y(k) = B(z^{-1})\Delta u(k) + C(z^{-1})v(k) \quad (4.145)$$

(where we assume $d = 1$). Assume that the polynomial matrices $A(z^{-1})$, $B(z^{-1})$, $C(z^{-1})$ — we can do everything for the multivariable case here — are given by

$$A(z^{-1}) = I - A_1 z^{-1} - \cdots - A_n z^{-n} \quad (4.146)$$

$$B(z^{-1}) = B_1 z^{-1} + \cdots + B_p z^{-p} \quad (4.147)$$

$$C(z^{-1}) = I z^{-1} + C_2 z^{-2} + \cdots + C_q z^{-q} \quad (4.148)$$

Notice that we can take the leading coefficient matrices of $A(z^{-1})$ and $C(z^{-1})$ to be the identity I , without loss of generality. The model can be written in difference equation form as

$$\begin{aligned} y(k) - y(k-1) &= A_1[y(k-1) - y(k-2)] + \cdots + A_n[y(k-n) - y(k-n-1)] \\ &\quad + B_1[u(k-1) - u(k-2)] + \cdots + B_p[u(k-p) - u(k-p-1)] \\ &\quad + v(k-1) + C_2 v(k-2) + \cdots + C_q v(k-q) \end{aligned} \quad (4.149)$$

² $(W+XYZ)^{-1} = W^{-1} - W^{-1}X(ZW^{-1}X+Y^{-1})^{-1}ZW^{-1}$. In this case we apply it with $W = (zI - A_d)$, $X = L_d$, $Y = I$, $Z = C_d$.

Now define the state vector to be

$$\begin{aligned} \mathbf{x}(k) = & [\mathbf{y}(k)^T, \mathbf{y}(k-1)^T, \dots, \mathbf{y}(k-n)^T, \mathbf{u}(k-1)^T, \dots, \mathbf{u}(k-p)^T, \\ & \mathbf{v}(k-1)^T, \dots, \mathbf{v}(k-q+1)^T]^T \end{aligned} \quad (4.150)$$

then (4.149) is equivalent to the (non-minimal) state-space model

$$\mathbf{x}(k+1) = \mathcal{A}\mathbf{x}(k) + \mathcal{B}_u\mathbf{u}(k) + \mathcal{B}_v\mathbf{v}(k) \quad (4.151)$$

$$\mathbf{y}(k) = \mathcal{C}\mathbf{x}(k) \quad (4.152)$$

where

$$\mathcal{A} = \begin{bmatrix} \mathcal{A}_{11} & \mathcal{A}_{12} & \mathcal{A}_{13} \\ 0 & \mathcal{A}_{22} & 0 \\ 0 & 0 & \mathcal{A}_{33} \end{bmatrix} \quad (4.153)$$

and the sub-matrices are given by:

$$\mathcal{A}_{11} = \begin{bmatrix} I + A_1 & A_2 - A_1 & \cdots & A_n - A_{n-1} & -A_n \\ I & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix} \quad (4.154)$$

$$\mathcal{A}_{12} = \begin{bmatrix} B_2 - B_1 & \cdots & B_p - B_{p-1} & -B_p \\ 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 \end{bmatrix} \quad (4.155)$$

$$\mathcal{A}_{13} = \begin{bmatrix} C_2 & \cdots & C_q \\ 0 & \cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & 0 \end{bmatrix} \quad (4.156)$$

$$\mathcal{A}_{22} = \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ I & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix}}_{p \text{ blocks}} \quad (4.157)$$

$$\mathcal{A}_{33} = \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ I & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & I & 0 \end{bmatrix}}_{q-1 \text{ blocks}} \quad (4.158)$$

The matrices \mathcal{B}_u and \mathcal{B}_v are given by

$$\mathcal{B}_u = [B_1^T, \underbrace{0, \dots, 0}_{n \text{ blocks}}, I, \underbrace{0, \dots, 0}_{p-1 \text{ blocks}}, \underbrace{0, \dots, 0}_{q-1 \text{ blocks}}]^T \quad (4.159)$$

and

$$\mathcal{B}_v = [I, \underbrace{0, \dots, 0}_{n \text{ blocks}}, \underbrace{0, \dots, 0}_{p \text{ blocks}}, I, \underbrace{0, \dots, 0}_{q-2 \text{ blocks}}]^T \quad (4.160)$$

The output matrix \mathcal{C} takes the form

$$\mathcal{C} = [I, 0, \dots, 0] \quad (4.161)$$

Note that discrete-time ‘integration’ is built in to this model; \mathcal{A} has m eigenvalues at +1, which can be seen as follows. Suppose that $y(k) = y(k-1) = \dots = y(k-n) = y_\infty$, that $u(k-1) = \dots = u(k-p) = 0$, and that $v(k-1) = \dots = v(k-q+1) = 0$, and let x_∞ be the state vector which corresponds to these values:

$$x_\infty = [\underbrace{y_\infty^T, \dots, y_\infty^T}_{n+1 \text{ times}}, \underbrace{0, \dots, 0}_{p \text{ times}}, \underbrace{0, \dots, 0}_{q-1 \text{ times}}]^T \quad (4.162)$$

Then it is easily verified that

$$\mathcal{A}x_\infty = x_\infty \quad (4.163)$$

which shows that any x_∞ of this form is an eigenvector of \mathcal{A} , with eigenvalue +1. This is another way of saying that any set of constant outputs is an equilibrium point for this system, when the inputs and disturbances are all zero.

Now consider the observer gain

$$L' = [I, \underbrace{0, \dots, 0}_{n+p \text{ times}}, I, 0, \dots, 0]^T \quad (4.164)$$

The first identity matrix in L' gives $\hat{y}(k|k) = y(k)$. As in Section 2.6.4, the structure of \mathcal{A} and \mathcal{B}_u then ensures that the top $n + p + 1$ elements of $\hat{x}(k|k)$ contain measured outputs and inputs after $\max(n+1, p)$ steps of running the observer. The second identity matrix in L' gives the disturbance estimate $\hat{v}(k|k) = y(k) - \hat{y}(k|k-1)$, where, as usual, $\hat{y}(k|k-1) = \mathcal{C}\hat{x}(k|k-1)$, and $\hat{x}(k|k-1) = \mathcal{A}\hat{x}(k-1|k-1) + \mathcal{B}_u u(k-1)$. This is a reasonable estimate of the disturbance, given the model. Is this observer stable? In Section 2.6.4, with no disturbances, we found that the observer was deadbeat, with all the eigenvalues of $A - LC = A(I - L'C)$ being zero. The situation now is a little more complicated. The matrix $A - LC = \mathcal{A}(I - L'C)$ is the same as the matrix \mathcal{A} , except that the first block column (that is, the first m columns, where m is the number of outputs) is replaced by

$$[-C_2^T, \underbrace{0, \dots, 0}_{n+p \text{ times}}, 0, -I, 0, \dots, 0]^T \quad (4.165)$$

This matrix has its eigenvalues at zero, and at the roots of the polynomial $C(z^{-1})$. The observer is therefore stable, providing that $C(z^{-1})$ has all of its roots inside the unit disk. Recall that this is the same as the condition that was required when estimating

the disturbance from equation (4.116). The zero eigenvalues reflect the fact that those elements of the state vector which correspond to past values of inputs and outputs are estimated exactly after a finite number of steps.

Note that, as in Section 2.6.4, this observer can be used even with unstable plants.

For further examples of state-space interpretations of the GPC model, see [OC93].

Note on use of Model Predictive Control Toolbox. The *Model Predictive Control Toolbox* uses the alternative state-space representation (2.37), with the assumption that y and z are the same (which is also assumed by GPC). The observer gain that must be supplied to *Model Predictive Control Toolbox* functions must be compatible with this choice of state vector. In this case using the observer gain

$$\mathcal{L}' = \begin{bmatrix} L' \\ I \end{bmatrix} \quad (4.166)$$

where L' is given by (4.164), gives the GPC model — see Exercise 4.11. L' is the matrix that must be supplied as the argument `Kest` in *Model Predictive Control Toolbox* functions such as `smpc`.

Example 4.11

Consider the unstable helicopter of Example 1.8. We saw in Chapter 1 that the simple ‘DMC’ disturbance model is not adequate to give offset-free tracking when the plant is unstable, because an independent model cannot be used — see Exercise 1.11. In order to overcome this problem the GPC model (4.145) can be used.

With a sampling interval of 0.6 sec, the discrete-time transfer function of the helicopter is

$$\frac{6.472z^2 - 2.476z + 7.794}{z^3 - 2.769z^2 + 2.565z - 0.7773} \quad (4.167)$$

Dividing the numerator and denominator by z^{-3} gives

$$A(z^{-1}) = 1 - 2.769z^{-1} + 2.565z^{-2} - 0.7773z^{-3} \quad (4.168)$$

$$B(z^{-1}) = 6.472z^{-1} - 2.476z^{-2} + 7.794z^{-3} \quad (4.169)$$

so that $n = p = 3$. We need to choose the polynomial $C(z^{-1})$, ensuring that its roots are inside the unit disk. Let us choose the polynomial ($q = 2$)

$$C(z^{-1}) = z^{-1} - 0.8z^{-2} = z^{-1}(1 - 0.8z^{-1}) \quad (4.170)$$

which has one root at $z = 0.8$ — see Section 7.4.2 for some motivation for this choice. Choosing the state vector as in (4.150) gives

$$x(k) = [y(k), y(k-1), \dots, y(k-3), u(k-1), \dots, u(k-3), v(k-1)]^T \quad (4.171)$$

and

$$\mathcal{A} = \begin{bmatrix} 1 + A_1 & A_2 - A_1 & A_3 - A_2 & -A_3 & B_2 - B_1 & B_3 - B_2 & -B_3 & C_2 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 3.769 & -5.334 & 3.3423 & -0.7773 & -8.948 & 10.27 & -7.794 & -0.8 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathcal{B}_u = [6.472, 0, 0, 0, 1, 0, 0, 0]^T$$

$$\mathcal{B}_v = [-0.8, 0, 0, 0, 0, 0, 0, 1]^T$$

Choosing the observer gain matrix L' as in (4.164), namely

$$L' = [1, 0, 0, 0, 0, 0, 0, 1]^T \quad (4.172)$$

gives the observer state-transition matrix

$$\mathcal{A} - L\mathcal{C} = \begin{bmatrix} 0.8 & -5.334 & 3.3423 & -0.7773 & -8.948 & 10.27 & -7.794 & -0.8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.173)$$

which has, as expected, seven eigenvalues at 0 and one at 0.8.

The solid lines in Figure 4.5 show the response when GPC is applied, without constraints, with $H_p = 5$, $H_u = 2$, $Q = 1$, and $R = 0$, assuming a perfect model. The set-point is $s(k) = 1$. The plant and internal model both have zero initial state, and there are no disturbances for the first 10 seconds. Then an unmeasured step disturbance of -0.1 is added to the input. It can be seen that, after a transient lasting about 10 seconds, the output returns to its set-point. The input increases to compensate for the disturbance.

The dashed lines in Figure 4.5 show the response when a deadbeat observer is used — namely, when the observer pole is placed at $z = 0$ instead of $z = 0.8$. It is seen that the

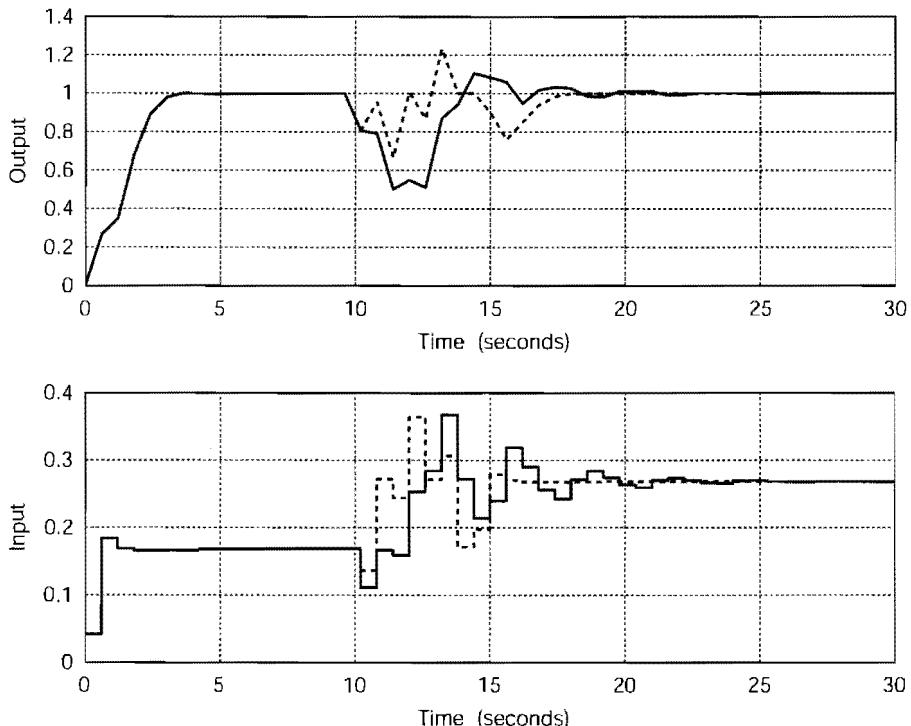


Figure 4.5 GPC applied to unstable helicopter.

peak error from the set-point is reduced, compared with the previous response, but the control actions are more aggressive, and the transient is not finished any more quickly. Note that the two responses are identical for the first 10 seconds. This is because in this idealized situation the state of the internal model tracks the state of the plant exactly, and hence the observer has no effect, until the disturbance occurs.

4.2.6 Multivariable systems

Since everything that can be done with a transfer function formulation of predictive control can also be done with a state-space formulation, and more, there seems to be no reason for using transfer functions. This is particularly true for the multivariable case. In the SISO case there are at least some nice interpretations which are not seen so clearly with state-space models. But this is no longer true in the multivariable case.

It is sometimes claimed that the transfer function formulation is more suitable for adaptive applications. This claim does not appear to have been justified in detail. Plant parameter estimation is no more difficult for state-space models than it is for general transfer function models — in the SISO case it is equally difficult, whereas in the multivariable case it is probably easier with state-space models — see [CM97], for example. Arguably, estimation of ARX models is easier, because it is sufficient to use

linear regression. But that involves restricting the class of models for the sake of a simpler algorithm, which may not be a worthwhile trade-off. Of course, even if one estimates a multivariable ARX model, it can easily be transformed into an equivalent state-space model, using standard algorithms.

For detailed treatments of the transfer function approach applied to multivariable systems, see [Mos95] or [CC95, CC99].

Exercises

4.1 Suppose you are given the sequence of step response matrices $S(0), S(1), \dots, S(N)$ for a plant. Show how to obtain the sequence of pulse response matrices $H(0), H(1), \dots$. What is the last one you can obtain?

4.2 (a) Write a *MATLAB* function `ss2step` which computes the step response matrices $S(M), S(M+1), \dots, S(N)$ of a state-space model, and stores them in a three-dimensional array, so that $S_{ij}(t)$ is stored in `S(i,j,t)`. It should have the interface specification:

$$S = \text{ss2step}(A, B, C, D, M, N)$$

(b) Write *MATLAB* functions `step2ups` and `step2the` for computing the matrices Υ ('upsilon') and Θ ('theta') from the step response sequence. The input data should be the three-dimensional array created in part (a), holding $S(1), \dots, S(N)$, where $N \geq H_p, H_w, H_p$ and H_u should be input arguments. The interface specifications should be:

$$\text{upsilon} = \text{step2ups}(S, H_w, H_p)$$

$$\text{theta} = \text{step2the}(S, H_w, H_p, H_u)$$

4.3 Consider the case of a ‘square’ plant: $\ell = m$, horizons $H_p = H_w$, and $H_u = 1$, and no penalty on input moves: $\mathcal{R} = 0$. Using (4.27) and (4.28), and the results of Chapter 3, show that in the unconstrained case we have $K_{MPC} = S(H_w)^{-1}$, and hence that, with full state measurements, the controller has no dynamics in this case — that is, it consists of gains only.

4.4 A 2-input, 1-output system has the step response sequence $S(0) = [0, 0], S(1) = [0, 2], S(2) = [1, 3], S(k) = S(2)$ for $k > 2$. Find a state-space model which reproduces this step response exactly. *Note:* You need four states.

4.5 Show that, as an alternative to equation (4.55), one could estimate A from the equation

$$\Gamma_n^\leftarrow = A\Gamma_n$$

with a suitable definition of Γ_n^\leftarrow .

4.6 Solve the two Diophantine equations (4.110) and (4.121) for Example 4.9, for the case when the 2-step-ahead predictor is required.

4.7 Show that an output disturbance which is a ramp (straight line) of unknown slope can be modelled by taking $D(z^{-1}) = (1 - z^{-1})^2$ in Section 4.2.3. How should $C(z^{-1})$ and $v(k)$ be chosen?

- 4.8** Suppose we have a state-space model, with a disturbance $w(k)$ entering the state equations:

$$x(k+1) = Ax(k) + Bu(k) + Ww(k) \quad (4.174)$$

$$y(k) = C_y x(k) \quad (4.175)$$

Show that the transfer functions from the vector $w(k)$ to the output $y(k)$ all have the same denominator as the transfer function from the input $u(k)$ to $y(k)$. (You may prefer to do this for a specific 2 or 3-state example rather than in general.)

- 4.9** Verify that the model (4.142)–(4.144) corresponds to the GPC disturbance model.

- 4.10** Verify that the state-space model (4.151)–(4.160) corresponds to the difference equation (4.149).

- 4.11** Verify that the observer gain \mathcal{L}' given by (4.166) corresponds to the GPC model when the state vector is chosen as

$$\xi(k) = [\Delta x(k)^T, \eta(k)^T]^T \quad (4.176)$$

(that is, as in the *Model Predictive Control Toolbox*) and when the ‘A’ and ‘C’ matrices of the state-space model are chosen to be

$$\begin{bmatrix} A & 0 \\ CA & I \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & I \end{bmatrix} \quad (4.177)$$

by taking the following steps:

First observe that if

$$\hat{x}(k|k) = (I - L'C)\hat{x}(k|k-1) + L'y(k)$$

then

$$\Delta\hat{x}(k|k) = (I - L'C)\Delta\hat{x}(k|k-1) + L'[y(k) - y(k-1)] \quad (4.178)$$

Then show that use of the observer gain

$$\mathcal{L}' = \begin{bmatrix} L_{\Delta x} \\ L_\eta \end{bmatrix}$$

gives $\hat{\eta}(k|k) = y(k)$ if $L_\eta = I$ and hence that

$$\Delta\hat{x}(k|k) = (I - L_{\Delta x}C)\Delta\hat{x}(k|k-1) + L_{\Delta x}[y(k) - y(k-1)] \quad (4.179)$$

from which the result follows.

- 4.12** Use the *Model Predictive Control Toolbox* function `smpc` to reproduce the results shown in Figure 4.5 for Example 4.11. (The input disturbance used to generate the figure changed from 0 to -0.1 in 3 increments: $d_u(k) = (-0.03, -0.07, -0.1)$ for $k = 17, 18, 19$ — recall that k counts time increments, and the sampling interval is 0.6 second. This is obtained by passing the following argument `wu` to function `smpc`: `wu=[zeros(16,1);-0.03;-0.07;-0.1]`. Also note that the observer gain argument `Kest` must be passed to the function `smpc` in the form shown in (4.166).)

Investigate the responses obtained for other values of the observer pole, such as 0.3 and 0.99.

Other formulations of predictive control

5.1	Measured disturbances and feedforward	145
5.2	Stabilized predictions	148
5.3	Decomposition of unstable model	150
5.4	Non-quadratic costs	151
5.5	Zones, funnels and coincidence points	156
5.6	Predictive Functional Control (PFC)	159
5.7	Continuous-time predictive control	163
	Exercises	165

5.1 Measured disturbances and feedforward

It is frequently the case that the effects of some disturbances can be anticipated and approximately cancelled out by suitable control actions. When this is done it is called *feedforward* control. It can be more effective than feedback control, because the latter has to wait until the effect of the disturbance becomes apparent before taking corrective action. In order to anticipate the effect of the disturbance, it is necessary to have some measurement which indicates that it is on its way. Examples of measured disturbances which can be corrected in this way are:

- Upstream changes in raw material composition.
- Feed rate from an upstream process.
- Heat duty requirements of upstream and downstream oil fractionator columns. (This assumes that each column is controlled individually, which is not the best solution [PG88].)

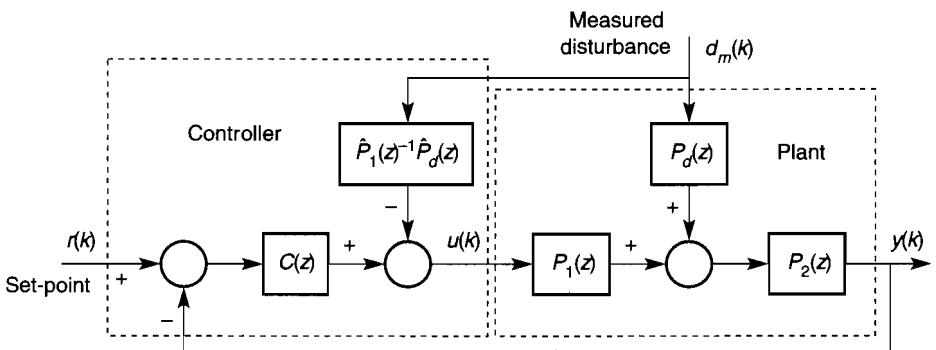


Figure 5.1 Feedforward from measured disturbances.

✿ Windshear, measured by a look-ahead Doppler radar (in an aircraft).

Exact cancellation would require an exact model of the ‘disturbance to output’ transfer function, and an exact inversion of this model. Since neither of these is possible, feed-forward control has to be used in combination with feedback control; the feedforward control removes most of the effect of the measured disturbance, and the feedback control removes the rest — as well as dealing with unmeasured disturbances, of course.

Conceptually, the control structure can be visualized as shown in Figure 5.1. The plant is shown as a linear model, which is split up into several blocks, to allow for the fact that often the control inputs and the disturbances pass through common dynamics — in this case the transfer function matrix $P_2(z)$. The measured disturbances are shown as passing first through the transfer function matrix $P_d(z)$. Thus their effect could be completely cancelled by adding the signal $-P_1(z)^{-1} P_d(z) \bar{d}(z)$ to the control input. This is shown in the figure, except that the notation $\hat{P}_1(z)$ and $\hat{P}_d(z)$ is used to emphasize that the transfer functions are never known exactly, and that it is usually impossible to realize inverses exactly — $P_1(z)$ may not even be a square matrix, so it may be impossible to find an inverse even in principle.

Feedforward is easily incorporated into predictive control. All that has to be done is to include the effects of the measured disturbances in the predictions of future outputs. The optimizer will then take these into account when computing the control signal. Looking back to Chapter 3, it will be seen that the only change that is required is to include the effects of the measured disturbances in the vector $\mathcal{Z}(k)$ of predicted outputs.

We shall modify the model (2.1)–(2.3) to include the measured disturbance vector $d_m(k)$:

Plant model with measured disturbance:

$$\text{States: } \hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) + B_d d_m(k) \quad (5.1)$$

$$\text{Measured outputs: } \hat{y}(k|k) = C_y \hat{x}(k|k) \quad (5.2)$$

$$\text{Controlled outputs: } \hat{z}(k|k) = C_z \hat{x}(k|k) \quad (5.3)$$

where we assume that the disturbance has no effect on the outputs until some time after it has been measured.

Example 5.1

Suppose that the transfer functions $P_1(z)$, $P_2(z)$ and $P_d(z)$, which appear in Figure 5.1, have state-space realizations (A_1, B_1, C_1, D_1) , (A_2, B_2, C_2, D_2) and $(A_d, B_d, C_d, 0)$, with corresponding state vectors x_1 , x_2 , and x_d . Assume that $D_2 D_1 = 0$ so that there is no direct feed-through from $u(k)$ to $y(k)$. Then a complete state-space model is given by

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_d(k+1) \end{bmatrix} = \begin{bmatrix} A_1 & 0 & 0 \\ B_2 C_1 & A_2 & B_2 C_d \\ 0 & 0 & A_d \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_d(k) \end{bmatrix} + \begin{bmatrix} B_1 \\ B_2 D_1 \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ 0 \\ B_d \end{bmatrix} d_m(k) \quad (5.4)$$

$$z(k) = [D_2 C_1 \quad C_2 \quad D_2 C_d] \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_d(k) \end{bmatrix} \quad (5.5)$$

The state estimate $\hat{x}(k|k)$ is still given by (2.77) — with L' designed using the model (5.1)–(5.3) of course — but (2.78) has to be replaced by

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) + B_d d_m(k) \quad (5.6)$$

To obtain $\mathcal{Z}(k)$ we must now replace (3.32) by

$$\mathcal{Z}(k) = \Psi \hat{x}(k|k) + \Upsilon u(k-1) + \Theta \Delta \mathcal{U}(k) + \Xi \mathcal{D}_m(k) \quad (5.7)$$

where

$$\mathcal{D}_m(k) = \begin{bmatrix} d_m(k) \\ \hat{d}_m(k+1|k) \\ \vdots \\ \hat{d}_m(k+H_p-1|k) \end{bmatrix} \quad (5.8)$$

and

$$\Xi = \begin{bmatrix} C_z B_d & 0 & \cdots & 0 \\ C_z A B_d & C_z B_d & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_z A^{H_p-1} B_d & C_z A^{H_p-2} B_d & \cdots & C_z B_d \end{bmatrix} \quad (5.9)$$

The matrices Ψ , Υ and Θ remain unchanged. We have assumed here that the measurement $d_m(k)$ is made at the same time as the measurement $y(k)$, and can be used for the calculation of $u(k)$.

The output predictions will clearly be influenced by what one assumes about the future behaviour of the measured disturbance, as represented by $\hat{d}_m(k+1|k), \dots, \hat{d}_m(k+H_p-1|k)$. Common practice is to assume that it will remain constant at the last measured

value, namely $d_m(k) = \hat{d}_m(k+1|k) = \dots = \hat{d}_m(k+H_p-1|k)$. (This is the assumption made by the *Model Predictive Control Toolbox*, for instance.) Other assumptions may be more appropriate in particular cases, if a better model of these disturbances is available.

The only remaining change required is to redefine the ‘tracking error’ $\mathcal{E}(k)$ — see (3.33) — as

$$\mathcal{E}(k) = \mathcal{T}(k) - \Psi\hat{x}(k|k) - \Upsilon u(k-1) - \Xi\mathcal{D}_m(k) \quad (5.10)$$

Once this change has been made, the optimal solutions in the unconstrained and constrained cases can be found as detailed in Chapter 3.

5.2 Stabilized predictions

In Chapter 2 we remarked that care should be taken when forming predictions, because of numerical problems that may arise. In particular, if the plant is unstable then problems are very likely to arise. An effective way of reducing this problem is to ‘stabilize’ the predictions. What is involved is a kind of re-parametrization of the plant’s input signal.

So far we have always computed predictions on the assumption that the plant is running ‘open-loop’, and we have computed the control signals as changes from a ‘do-nothing’ policy. Furthermore, we have assumed that after the end of the control horizon the input signal will remain constant. But this is not the only possibility.

Suppose that we assume that the ‘do-nothing’ policy will be to apply the linear state-feedback control

$$u(k) = -K\hat{x}(k|k) \quad (5.11)$$

for the indefinite future. And of course we will assume that the state-feedback gain K is one which stabilizes the linear model, namely that all the eigenvalues of $A - BK$ lie inside the unit disc. Then the ‘do-nothing’ predictions are formed as follows:

$$\begin{aligned} \hat{x}(k+1|k) &= A\hat{x}(k|k) + B\hat{u}(k|k) \\ &= (A - BK)\hat{x}(k|k) \end{aligned} \quad (5.12)$$

$$\begin{aligned} \hat{x}(k+2|k) &= A\hat{x}(k+1|k) + B\hat{u}(k+1|k) \\ &= (A - BK)\hat{x}(k+1|k) \\ &= (A - BK)^2\hat{x}(k|k) \end{aligned} \quad (5.13)$$

⋮

$$\hat{x}(k+H_p|k) = (A - BK)^{H_p}\hat{x}(k|k) \quad (5.14)$$

Now we can use optimization over a control horizon H_u to modify these baseline ‘do-nothing’ predictions, so as to minimize our cost function, and to ensure that the predicted behaviour stays within the constraints. This means that the prediction of the future input value $\hat{u}(k+i|k)$ will no longer be given by $-K\hat{x}(k+i|k)$, but by

$$\hat{u}(k+i|k) = -K\hat{x}(k+i|k) + q(i) \quad (5.15)$$

with $q(i) = 0$ for $i \geq H_u$, where the values $q(i)$ ($i = 0, 1, \dots, H_u - 1$) are chosen by the optimizer. Now we have

$$\hat{x}(k+1|k) = (A - BK)\hat{x}(k|k) + Bq(1) \quad (5.16)$$

$$\begin{aligned} \hat{x}(k+2|k) &= (A - BK)\hat{x}(k+1|k) + Bq(2) \\ &= (A - BK)^2\hat{x}(k|k) + (A - BK)Bq(1) + Bq(2) \end{aligned} \quad (5.17)$$

\vdots

$$\begin{aligned} \hat{x}(k+H_u-1|k) &= (A - BK)^{H_u-1}\hat{x}(k|k) + (A - BK)^{H_u-2}Bq(1) \\ &\quad + \cdots + Bq(H_u - 1) \end{aligned} \quad (5.18)$$

$$\begin{aligned} \hat{x}(k+H_u|k) &= (A - BK)^{H_u}\hat{x}(k|k) + (A - BK)^{H_u-1}Bq(1) \\ &\quad + \cdots + (A - BK)Bq(H_u - 1) \end{aligned} \quad (5.19)$$

\vdots

$$\begin{aligned} \hat{x}(k+H_p|k) &= (A - BK)^{H_p}\hat{x}(k|k) + (A - BK)^{H_p-1}Bq(1) \\ &\quad + \cdots + (A - BK)^{H_p-H_u+1}Bq(H_u - 1) \end{aligned} \quad (5.20)$$

The important point here is that instead of high powers of A , which we had in the expressions for the ‘open-loop’ predictions in Chapter 2, here we have high powers of $A - BK$. And because K is chosen to be stabilizing, we have $(A - BK)^i \rightarrow 0$ as $i \rightarrow \infty$, so the likelihood of numerical problems due to finite-precision arithmetic is greatly reduced.

In this formulation the ‘decision variables’ of the optimization problem are the $q(i)$ s, in place of the old $\Delta\hat{u}(k+i)$ s. We need to check whether we still have a QP problem. Equation (5.15) shows that the predicted plant input $\hat{u}(k+i|k)$ is related linearly to $q(i)$. ($\hat{x}(k+i|k)$ is related linearly to $q(j)$, $j < i$.) And clearly

$$\begin{aligned} \Delta\hat{u}(k+i|k) &= [-K\hat{x}(k+i|k) + q(i)] - [-K\hat{x}(k+i-1|k) + q(i-1)] \\ &= -K\Delta\hat{x}(k+i|k) + \Delta q(i) \end{aligned} \quad (5.21)$$

This makes it clear that the $\Delta q(i)$ s, and hence the $q(i)$ s themselves, are linearly related to the $\Delta\hat{u}(k+i|k)$ s. So the linear inequalities in the variables $\Delta\hat{u}(k+i|k)$ remain as linear inequalities in the variables $q(i)$. So we still have a QP problem to solve.

The use of stabilized predictions was first introduced in [KR92, RK93]. The emphasis in these papers was on the use of feedback to obtain deadbeat, or finite impulse response FIR, behaviour prior to the use of optimization, as a stratagem to prove closed-loop stability — see Section 6.2. The numerical benefits for prediction were noticed later [RRK98]. In [MR93b, MR93a, RM93, SR96b] a parametrization of the form $u(k) = -K\hat{x}(k|k)$ was assumed *after the end of the control horizon*, and was an essential step in showing how constrained predictive control problems could be solved over infinite horizons. (We shall return to this in Section 6.2.) In [RRK98] it was pointed out that such parametrizations could be assumed over the whole horizon, with consequent benefits for numerical conditioning of the predictions.

The remaining question is, which stabilizing state-feedback gain K should be used? In principle there are many possibilities to choose from, obtained for example by pole-placement techniques. But in practice, with state dimensions bigger than about 5, it is

very difficult to get pole placement to work (because of the difficulty of knowing what closed-loop pole locations are reasonably attainable), and the only practical alternatives are to obtain K by solving a ‘Linear Quadratic’, or possibly an ‘ H_∞ ’ problem. In the context of predictive control with a quadratic cost function, the natural choice is to obtain K as the solution to the infinite-horizon linear quadratic problem, with the same cost function as that used in the formulation of the predictive control problem. One proposal for obtaining ‘robust’ predictive control is to make K the ‘decision variable’, namely to have K chosen (for each k) by an optimizer — see Section 8.4.

Using stabilized predictions has another benefit. It allows a predictive controller to be more optimistic about the possibility of remaining feasible over the duration of the prediction horizon, if there are disturbances which are not known accurately or if there is some uncertainty about the true behaviour of the plant. With open-loop predictions the controller may be too pessimistic, and wrongly conclude that there is no way of ensuring a feasible solution [BM99b]. We shall examine this further in Chapter 8.

5.3 Decomposition of unstable model

Richalet’s *PFC* approach uses a very different approach to dealing with unstable systems from that taken by GPC [Ric93b]. Since *PFC* insists on an ‘independent model’, namely one in which the plant outputs are not used to correct the internal model, the poles of the internal model are necessarily cancelled by the controller. But this is unacceptable if the internal model is unstable, since it leads to loss of internal stability — for example, the transfer function from the plant input to the plant output is unstable when the feedback loop is closed (see Chapter 6 for details).

Richalet therefore represents the unstable plant by a feedback interconnection of two *stable* systems, as shown in Figure 5.2. Such a representation, or ‘decomposition’ as Richalet calls it, can always be found, although it is not unique. The original system $P(z)$ and the two stable systems $P_1(z)$, $P_2(z)$ are related by

$$P(z) = P_1(z)[I - P_2(z)P_1(z)]^{-1} \quad (5.22)$$

or, more simply in the SISO case, by:

$$P(z) = \frac{P_1(z)}{1 - P_2(z)P_1(z)} \quad (5.23)$$

This representation is now redrawn as in Figure 5.3. This shows $P_1(z)$ driven by the input, but the feedback loop is replaced by the open-loop block $P_2(z)P_1(z)$ driven

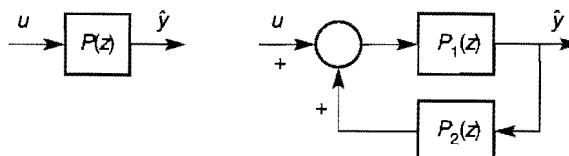


Figure 5.2 Decomposition of unstable plant as a feedback interconnection of two stable systems.

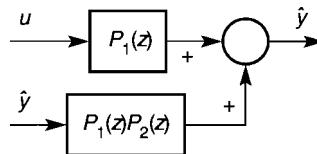


Figure 5.3 Decomposition of unstable plant as a stable model and ‘measured quasi-disturbance’.

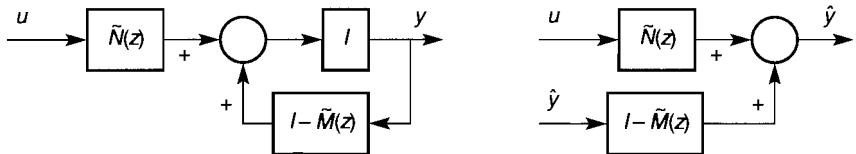


Figure 5.4 Coprime factorization of an unstable system, and its approximation.

by the *output* of the model. The ingenious trick which Richalet uses at this point is to take $P_1(z)$ as the internal model, and to regard the effect of the model output, passing through $P_2(z)$, as a measured disturbance. Of course, some approximation enters at this point, because the future output has to be estimated, in order to obtain predictions of itself. Richalet suggests replacing the model output \hat{y} at the input of $P_2(z)$ by the actual plant output y , and deriving predictions from this by making plausible assumptions. For example, if the set-point is constant one could assume that future output values will be equal to the latest measured values.

This idea can be generalized as follows. It is always possible to find a coprime factorization of a transfer function (matrix):

$$P(z) = \tilde{M}(z)^{-1} \tilde{N}(z) \quad (5.24)$$

such that $\tilde{M}(z)$ and $\tilde{N}(z)$ are stable, and effective algorithms for doing this exist [Mac89, ZDG96]. (‘Coprime’ means that there are no unstable pole-zero cancellations between the factors.) A block diagram which corresponds to this factorization is shown on the left of Figure 5.4. Note that the system $I - \tilde{M}(z)$ is stable, since $\tilde{M}(z)$ is stable. On the right of Figure 5.4 is shown an approximation of the coprime factorization, using an estimate \hat{y} of the output as the input to the feedback block.

More details of PFC are given in Section 5.6.

5.4 Non-quadratic costs

5.4.1 Characteristics of LP and QP problems

It is possible to modify the cost function used in predictive control, so that absolute values of errors are penalized, rather than squared values:

$$V(k) = \sum_{i=1}^{H_p} \sum_{j=1}^m |\hat{z}_j(k+i|k) - r_j(k+i)| q_j \quad (5.25)$$

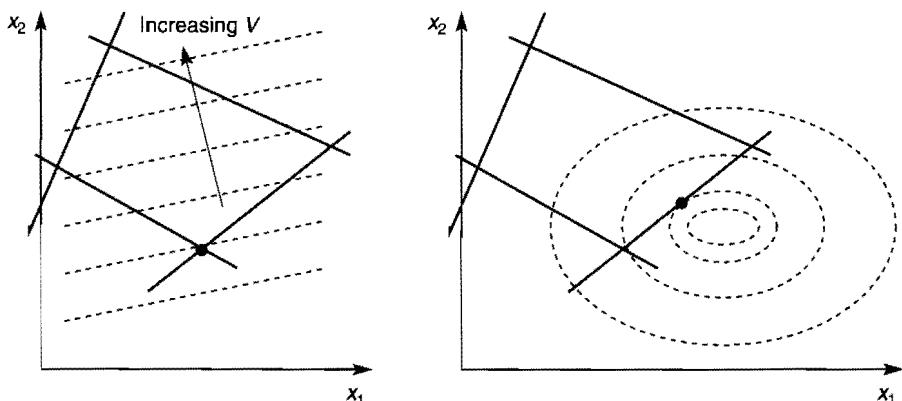


Figure 5.5 Cost contours and constraints for LP (left) and QP (right) problems. The black dot shows the optimal solution in each case.

subject to the usual constraints on $\hat{u}(k + i|k)$, $\Delta\hat{u}(k + i|k)$, and $\hat{z}(k + i|k)$. (The q_j s are nonnegative weights.) This cost function was used in [CS83]. Such problems are *Linear Programming* (LP) rather than *Quadratic Programming* (QP) problems,¹ as we shall show shortly.

One of the historical reasons for adopting such a formulation is that LP problems can be solved more quickly than QP problems, and that there is more experience (particularly in industrial and commercial environments) of solving LP problems, with excellent and well-proven software available. The importance of this reason is decreasing rapidly, because of the great strides being made in the development of algorithms and software for solving QP and other convex problems — see Section 3.3.

But another reason — and a better one — for adopting this alternative formulation of the predictive control problem is that the resulting behaviour is different. The solutions of LP problems are always at the intersection of constraints — occasionally on a single constraint — whereas solutions of QP problems can be off constraints, on a single constraint, or (relatively rarely) at an intersection of constraints. An attempt to illustrate this behaviour is made in Figure 5.5 which shows a problem with only two ‘decision variables’ x_1 and x_2 . The thick solid lines represent linear inequality constraints, with the feasible region being the interior of the region bounded by these lines. The dashed lines represent constant cost contours, those for an LP problem being shown on the left — these are straight lines, or hyperplanes in general — and those for a QP problem being shown on the right — these are ellipses, or hyper-ellipsoids in general. For the LP problem the cost increases as one moves upwards in the figure, and for the QP problem the cost increases as one moves out from the centre of the ellipse.

It can be seen from the figure on the left, that the optimal solution to an LP problem must always lie on at least one constraint. Also, it can be expected to lie at the intersection

¹ Some authors have started to adopt the phrase *Linear Optimization*, because the meaning of ‘programming’ has changed since the 1940s. While this is undoubtedly correct and logical, we retain the traditional terminology for the sake of easy recognition.

of more than one constraint, the only exception being if the slope of the cost contours is the same as the slope of at least one constraint. Furthermore, the solution is relatively robust with respect to the cost function: the slope of the cost contours can vary quite a lot before the optimal solution changes — but when it does change, it jumps suddenly to one of the other corners of the feasible region. On the other hand, the optimal solution is sensitive to the constraints, and changes continuously (mostly) as these are changed.

The figure on the right shows that the optimal solution to a QP problem behaves very differently. As shown, the optimal solution is on one of the constraints. But it is easy to imagine a situation in which the centre of the ellipse is in the feasible region, and the solution will then not be on any of the constraints. It is also easy to see that it would be very unusual for the solution to be at the intersection of more than one constraint. (In problems with particular structure it may not be so unusual. For example, in the predictive control context, a big disturbance may make an output constraint active over several steps of the prediction horizon. Seen from the viewpoint of the QP algorithm this would mean that the solution is on the intersection of several constraints.) In this case the optimal solution is completely insensitive to changes in the constraints if it is in the interior of the feasible region, but depends continuously on the cost function. If it is on the boundary of this region — that is, some constraints are active — then it depends continuously on both the cost function and the constraints.

Each of these behaviours has its merits. For example, if it is known that most profitable operation is obtained with the plant at the intersection of several constraints, and if the constraints are known precisely, and the model is reasonably accurate, whereas the ‘true’ (economic) cost function is not known so precisely, then the robust behaviour of the LP solution is appropriate — it will try to keep the plant at the true optimal operating point, despite the uncertainty about the true cost function. In practice, however, one would not risk trying to hold the plant at the real constraints, so these would be replaced by synthetic constraints, which allowed some margin for unavoidable constraint violations. Such violations will arise because of unexpected disturbances, errors in the model of the plant, errors in measurements, and errors in the constraint specifications themselves. So in practice the LP solution would try to hold the plant robustly at a fixed operating point, but one which was not the true optimum. In this case the QP solution looks more attractive.

In the situation just outlined, a QP formulation would try to hold the plant at a point just inside the feasible region, close to the optimal operating point, as shown in Figure 5.6. Because of the awareness that the predictive control formulation has of constraints, and the resulting nonlinear action if the constraints are approached, this operating point can be much closer to the constraints than could be contemplated if a linear controller were used. The QP formulation now offers a lot of flexibility, all of which can potentially be exercised on-line by plant operators:

1. The desired operating point, namely the centre of the iso-cost ellipsoids, can be moved around within the feasible region in a natural way. (In an LP formulation this can only be done by redefining the constraints.)
2. The relative weights in the cost function can be altered to change the unconstrained behaviour. For example the right half of Figure 5.6 shows the same operating point

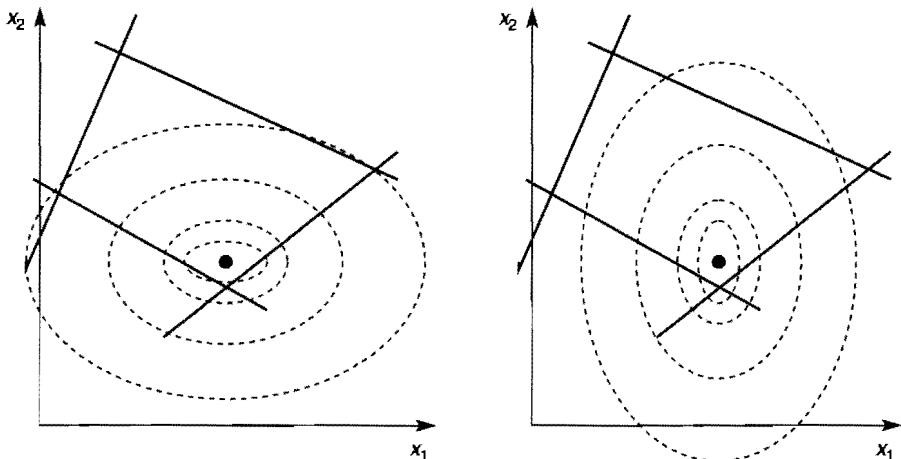


Figure 5.6 Using a QP formulation to place the desired operating point just inside the feasible region. *Left:* x_1 deviations cheaper than x_2 deviations. *Right:* x_2 deviations cheaper than x_1 deviations.

defined, but a lower cost indicated for deviations of the x_2 variable rather than the x_1 variable.

3. If constraints are persistently active, as in Figure 5.5, then the location of the operating point on the boundary of the feasible region can be altered by altering the weights. (The optimal solution can be moved up and down the solid line in Figure 5.5.)

As regards analysis of controller behaviour, a QP formulation has the very big advantage that it gives linear behaviour so long as the constraints are inactive, or a fixed set of constraints is active. All the tools of linear control theory can be applied to analyze the controller in these circumstances. This is not so with LP formulations.

5.4.2 Absolute value formulations

We return now to the cost function (5.25). We want to demonstrate that minimizing this, subject to the usual constraints on input changes, inputs and outputs, is indeed an LP problem. The standard form of an LP problem is:

$$\min_{\theta} c^T \theta \quad \text{subject to} \quad A\theta \geq b \text{ and } \theta \geq 0 \quad (5.26)$$

where θ is the vector of decision variables, b and c are vectors, and A is a matrix. A problem of the form

$$\min_{\theta} |c^T \theta| \quad \text{subject to} \quad A\theta \geq b \text{ and } \theta \geq 0 \quad (5.27)$$

can be shown to be an LP problem as follows. It is equivalent to the problem

$$\min_{\gamma, \theta} \gamma \quad \text{subject to} \quad \begin{cases} \gamma \geq c^T \theta \\ \gamma \geq -c^T \theta \\ A\theta \geq b \\ \theta \geq 0 \end{cases} \quad (5.28)$$

where γ is a new scalar variable, and hence to

$$\min_{\gamma, \theta} [1, 0] \begin{bmatrix} \gamma \\ \theta \end{bmatrix} \quad \text{subject to} \quad \begin{bmatrix} 1 & -c^T \\ 1 & c^T \\ 0 & A \end{bmatrix} \begin{bmatrix} \gamma \\ \theta \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ b \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \gamma \\ \theta \end{bmatrix} \geq 0 \quad (5.29)$$

which is in the standard form (5.26). (The constraint $\gamma \geq 0$ is in fact redundant.)

In a similar way, the ‘1-norm’ problem

$$\min_{\theta} \|C\theta - d\|_1 = \min_{\theta} \sum_j |c_j^T \theta - d_j| \quad \text{subject to} \quad A\theta \geq b \quad (5.30)$$

where c_j^T is the j th row of the matrix C , is equivalent to the LP problem

$$\min_{\gamma, \theta} \sum_j \gamma_j \quad \text{subject to} \quad \begin{bmatrix} I & -C \\ I & C \\ 0 & A \end{bmatrix} \begin{bmatrix} \gamma \\ \theta \end{bmatrix} \geq \begin{bmatrix} -d \\ d \\ b \end{bmatrix} \quad (5.31)$$

where γ is now a vector. This is now very close to the predictive control problem we want to solve.

The cost function (5.25) can be written as

$$V(k) = \|\mathcal{Q}[\Theta \Delta \mathcal{U}(k) - \mathcal{E}(k)]\|_1 \quad (5.32)$$

where Θ , $\mathcal{E}(k)$ and $\Delta \mathcal{U}(k)$ are as defined in Section 3.1.1 and \mathcal{Q} consists of H_u copies of Q stacked on top of each other, where $Q = \text{diag}\{q_1, \dots, q_p\}$. This has to be minimized subject to inequality (3.41). This problem is in the same form as (5.30), so we see that this is an LP problem.

5.4.3 Min-max formulations

In [CM86] it was proposed to solve the predictive control problem by minimizing the ‘ ∞ -norm’ cost function:

$$V(k) = \|\mathcal{Q}[\Theta \Delta \mathcal{U}(k) - \mathcal{E}(k)]\|_\infty = \max_i \max_j |\hat{z}_j(k+i|k) - r_j(k+i)|q_j \quad (5.33)$$

subject to the usual constraints. This attempts to minimize the peak deviation from the set-points, over all the controlled outputs and over the prediction horizon. Using the ‘1-norm’ as in (5.25) or the ‘2-norm’ as in (3.45) minimizes something like the

average deviation, loosely speaking, but can permit occasional large deviations. Using the ' ∞ -norm' is an attempt to overcome this, if it is a problem.

This is again an LP problem, which can be seen as follows. The problem is of the form

$$\min_{\theta} \|C\theta - d\|_{\infty} \quad \text{subject to} \quad A\theta \geq b \quad (5.34)$$

which is the same as

$$\min_{\theta} \max_j |c_j^T \theta - d_j| \quad \text{subject to} \quad A\theta \geq b \quad (5.35)$$

This is equivalent to

$$\min_{\gamma, \theta} \gamma \quad \text{subject to} \quad \begin{bmatrix} \mathbf{1} & -C \\ \mathbf{1} & C \\ 0 & A \end{bmatrix} \begin{bmatrix} \gamma \\ \theta \end{bmatrix} \geq \begin{bmatrix} -d \\ d \\ b \end{bmatrix} \quad (5.36)$$

where γ is now a scalar, and $\mathbf{1}$ denotes a vector, each element of which is 1. Clearly this is an LP problem.

5.5 Zones, funnels and coincidence points

We have represented the control objectives as a reference trajectory for the controlled outputs to follow. In some applications the exact values of the controlled outputs are not important, so long as they remain within specified boundaries, or 'zones', as shown in Figure 5.7. This is easily accommodated within our 'standard problem' by

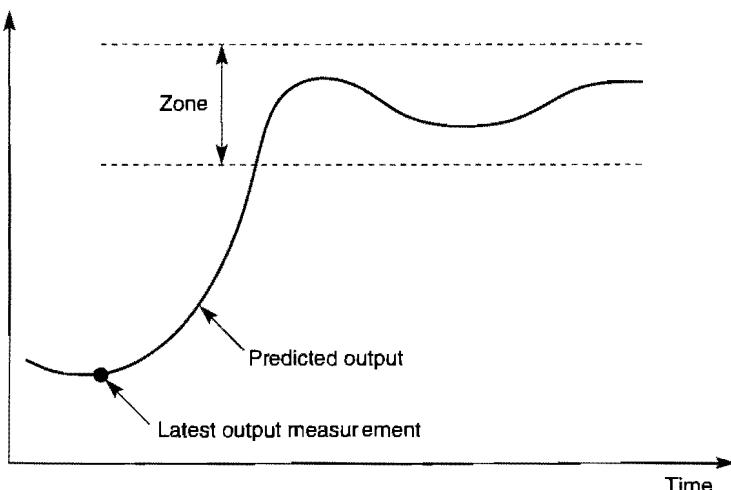


Figure 5.7 A zone objective.

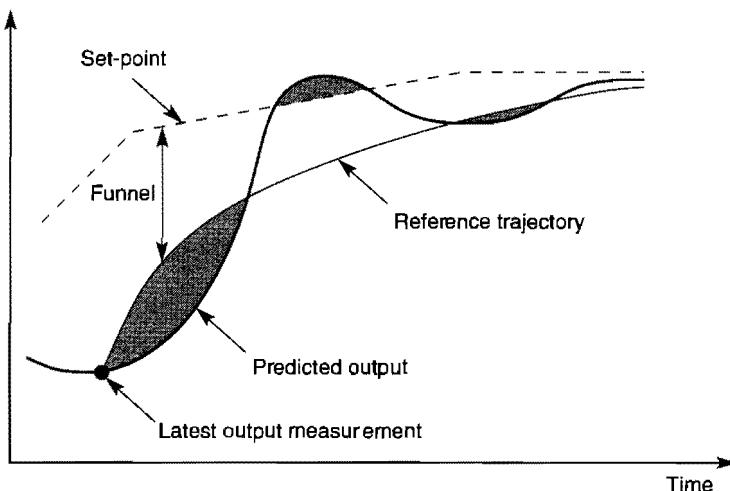


Figure 5.8 A funnel objective.

setting $Q(i) = 0$, and leaving only constraints on the controlled outputs to define the performance objectives. If $R(i) = 0$ also, then something has to be done to ensure that the optimization remains well-defined. The optimization problem typically simplifies from a QP to an LP problem, if all the objectives are ‘zone’ objectives.

A variation of this is to represent the objectives by ‘funnels’, which are the same as ‘zones’, except that they become narrower over the prediction horizon. If moving towards a set-point s_j from a lower value $z_j(k)$, for example, one can define a ‘funnel’ whose upper boundary is s_j , and whose lower boundary is a first-order ‘trajectory’:

$$(1 - \alpha^i)s_j + \alpha^i z(k) \leq \hat{z}_j(k + i|k) \leq s_j \quad (0 < \alpha < 1) \quad (5.37)$$

as illustrated in Figure 5.8. The shaded areas in the figure show that the predicted trajectory is penalized only when it leaves the funnel. The advantage of this, over defining a reference trajectory

$$r(k + i) = (1 - \alpha^i)s_j + \alpha^i z(k)$$

is that, in the multivariable case, the reference trajectories for different outputs can be inconsistent with each other. Following one reference trajectory might cause another output to return to its set-point more quickly than its reference trajectory specifies. The use of funnels in this situation allows the controller to take advantage of this benign turn of events, rather than impeding the return of both outputs to their set-points unnecessarily.

A variation on this is to make the lower boundary of the funnel a straight line, as shown in Figure 5.9. This is what is done in Honeywell’s RMPCT product.

According to [QB96], commercial products enforce funnel constraints as ‘soft’ constraints, by switching in large penalty weights when the funnel boundaries are crossed or approached.

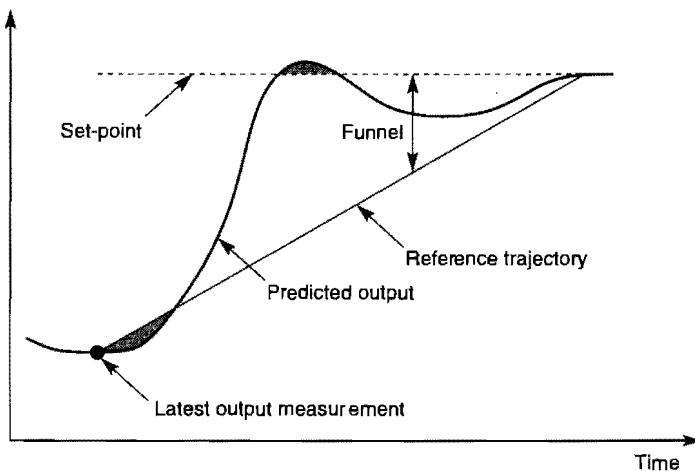


Figure 5.9 A *funnel* objective with a straight line boundary.

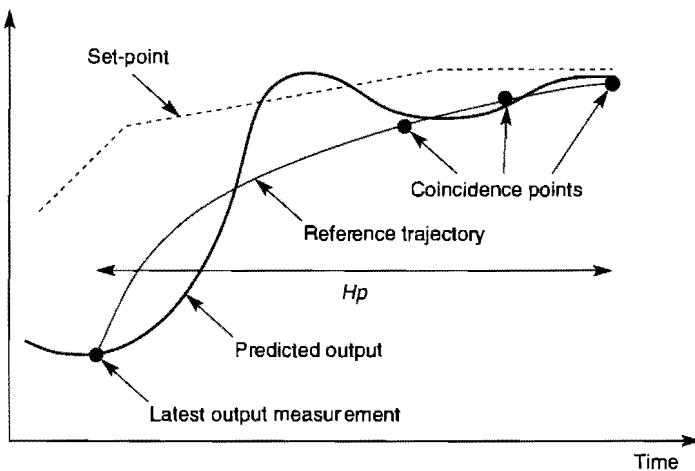


Figure 5.10 A reference trajectory with only a few *coincidence points*.

A third alternative is to use only a few *coincidence points*, rather than penalizing errors between $r(k+i)$ and $\hat{z}(k+i|k)$ over the whole prediction horizon, as we did in the simple examples of Chapter 1. This is illustrated in Figure 5.10. This is in some ways similar to the ‘zone’ idea; if the coincidence points are at the end of the prediction horizon, this corresponds to expressing indifference about the precise trajectory of the controlled outputs, so long as they are reasonably close to their set-points eventually. If different controlled outputs respond with different time-scales, then the coincidence points for some variables can be made different from those for others. This idea is easily accommodated within our standard problem formulation: just set most elements of $Q(i)$ to zero for most i .

5.6 Predictive Functional Control (PFC)

Predictive Functional Control, frequently abbreviated to *PFC*, is a form of predictive control which has been pioneered by Richalet [RRTP78, Ric93b]. In principle it is no different to predictive control as we have presented it so far, but it places emphasis on other features than those which have been emphasized in this book. Two of these features, namely the use of relatively few coincidence points, and the use of a reference trajectory which is distinct from the set-point trajectory, have already been introduced in Chapter 1.

Perhaps the most distinctive feature of *PFC* is that the future input is assumed to be a linear combination of a few simple *basis functions*. In principle these could be any appropriate functions, but in practice a polynomial basis is usually adopted (Figure 5.11). In this case the future input is assumed to be of the form

$$\hat{u}(k+i|k) = u_0(k) + u_1(k)i + u_2(k)i^2 + \cdots + u_c(k)i^c \quad (5.38)$$

Thus the future input profile is parametrized by the $c+1$ coefficients $u_0(k), u_1(k), \dots, u_c(k)$. In practice, low values of c are used, such as 1 or 2, so that the optimization is performed over only two or three parameters, for single-input systems. The notion of ‘control horizon’ therefore does not exist, but is replaced by the number of basis functions used. The same idea extends to multi-input systems, by taking $u_0(k), \dots, u_c(k)$ to

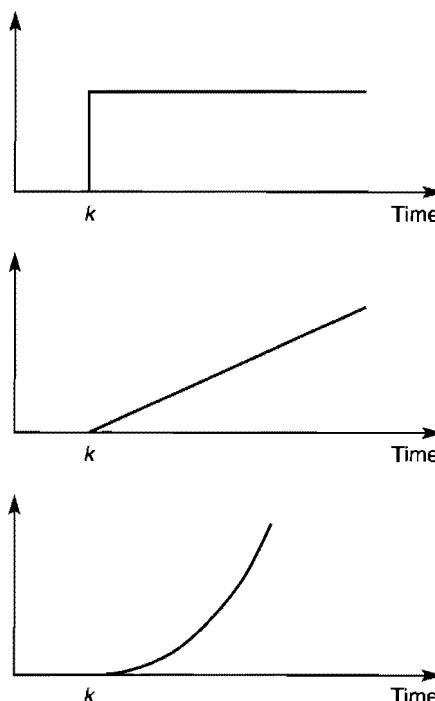


Figure 5.11 A future input built up from a polynomial basis.

be vectors of coefficients. The coefficients are shown here indexed by the time argument ($u_0(k)$ etc.) in order to emphasize that the coefficients are chosen to be optimal at each k , and so are different at each step.

An alternative parametrization of the input used in *PFC* is by means of delayed step functions — but that is exactly equivalent to the parametrization by means of increments $\Delta u(k)$ which we have been using since Chapter 2, so we shall not consider it further here.

The time window over which the cost function is minimized, namely $i = H_w$ to $i = H_p$ (called the *coincidence horizon* in the *PFC* literature) usually contains only a few points, and often a single point, so that the cost function is frequently of the form

$$V(k) = \|r(k + H_p) - \hat{y}(k + H_p | k)\|_Q^2 \quad (5.39)$$

The number of coincidence points in the coincidence horizon is taken at least as large as the number of basis functions used to parametrize the predicted input in (5.38). It will be noticed that no penalty on input moves has been imposed here, for reasons which will become apparent shortly. Such a penalty is sometimes imposed in *PFC*, and if it is, then, as usual, care is taken not to destroy steady-state accuracy, and to allow offset-free tracking: moves $\Delta \hat{u}(k + i | k)$ are penalized if the steady-state input is expected to be a constant, second differences $\Delta^2 \hat{u}(k + i | k)$ are penalized if the steady-state input is expected to be a ramp, and so on.

In the case of a SISO system, with the number of coincidence points equal to the number of parameters in (5.38), no optimization is necessary, since one can solve analytically for the input which makes $V(k) = 0$ — at least, if constraints are ignored. We have already seen examples of this in Chapter 1.

Correction for plant–model mismatches in steady-state gains, and for persistent disturbances, is achieved by ‘auto-compensation’, which is equivalent to modelling output disturbances. For example, if the plant is asymptotically stable, then steady-state gain errors in the model can be compensated for by forming an estimate of the actual plant output, once it is in steady state. This can be done, for example, by averaging the last few measured values. It is then assumed that the error between this estimated plant output and the previously predicted model output will continue into the future. This is almost equivalent to the method of obtaining offset-free tracking which was introduced in Chapter 1, and to the ‘DMC model’ of output disturbances introduced in Chapter 2. The only difference is that there the ‘trend’ is taken to be the last measured output, whereas *PFC* allows a more general estimate of the actual plant output.

This can be generalized in the following way: if the set-point is a polynomial of degree v , then it can be tracked without error, even in the presence of plant–model mismatch, providing that the input is structured as a polynomial of degree v (take $c = v$ in (5.38)). ‘Auto-compensation’ is achieved by fitting a polynomial of degree v to the plant output, then correcting the model prediction by extrapolating this polynomial. In practice, $v = 1$ or $v = 2$ is used, for following ramp or parabolic set-point trajectories. *PFC* places emphasis on this feature because it has been applied in servomechanisms, where such set-points are common [RRTP78, ReADAK87, Ric93a]. Let us suppose that the set-point trajectory is

$$s(k) = s_0 + s_1 k + \cdots + s_v k^v \quad (5.40)$$

and assume, for simplicity, that the reference trajectory coincides with the set-point trajectory: $r(k) = s(k)$. If the plant is asymptotically stable, and has reached a ‘steady-state’ in which its output is following a polynomial of degree v , then the input must also be such a polynomial:

$$u(k) = v_0 + v_1 k + \cdots + v_v k^v \quad (5.41)$$

Since this input is applied to both the model and the plant, the error between the model output and the plant output will, at worst, be a polynomial of degree v . Hence the predicted plant–model error is assumed to have the form

$$\hat{e}(k+i|k) = e(k|k) + \sum_{j=1}^v e_j i^j \quad (5.42)$$

where $e(k|k) = y(k) - \hat{y}(k|k-1)$ is the current error between the plant output and the model output. The coefficients e_1, \dots, e_v are obtained by fitting a trend to previous errors in some way — for example, a least-squares fit. The model predictions are corrected by this plant–model error, so that the future input trajectory is chosen to minimize the cost

$$V(k) = \sum_{i=1}^{v+1} \|s(k+P_i) - \hat{y}(k+P_i|k) - e(k+P_i|k)\|^2 \quad (5.43)$$

where $v+1$ coincidence points have been chosen, at times $k+P_1, \dots, k+P_{v+1}$. Since the only term in the cost that depends on the future input trajectory is $\hat{y}(k+P_i|k)$, the cost can be made zero by choosing the input trajectory such that

$$\hat{y}(k+P_i|k) = s(k+P_i) - e(k+P_i|k) \quad (i = 1, \dots, v+1) \quad (5.44)$$

and this can be done if the number of inputs is at least as great as the number of outputs. Now, by assumption, $y(k+P_i) = \hat{y}(k+P_i|k) + e(k+P_i|k)$, so we get $y(k+P_i) = s(k+P_i)$. (Otherwise our original assumption that the plant–model error can be extrapolated by a polynomial would not be valid.) Note that this argument does not depend on the model which produces $\hat{y}(k+P_i|k)$ being accurate in any respect, except for the feature that a polynomial input should result in the output being a polynomial of the same degree.

An important point is that the polynomial parametrization of the input is not necessary to obtain error-free tracking. The essentials are:

1. To have ‘auto-compensation’ of sufficiently high degree (at least v) — that is, to have an ‘internal model’ of the persistent plant–model mismatch.
2. To have enough coincidence points so that matching the reference trajectory at these points implies matching it everywhere else — that is, at least $v+1$ coincidence points.
3. To have enough degrees of freedom in the choice of future control actions to enable the reference trajectory to be matched at the coincidence points.

Thus other forms of predictive control, which do not use such an input parametrization, can achieve error-free tracking if they employ a suitably rich internal model of the plant–model mismatch, if there are sufficiently many coincidence points, and if the control horizon is long enough.

PFC generally uses a reference trajectory which is *re-anchored at the latest output measurement*, as already described in Chapter 1. Recall that, if the current set-point is $s(k)$, then the current error is $\varepsilon(k) = s(k) - y(k)$, and the reference trajectory $r(k+i|k)$ is defined so that $s(k+i) - r(k+i|k) = \varepsilon(k+i)$ decreases from $\varepsilon(k)$ to zero gradually as i increases. Most commonly, the reduction is exponential: $\varepsilon(k+i) = \lambda^i \varepsilon(k)$, where $0 \leq \lambda < 1$.

λ is an important parameter for *PFC*, since it defines the aggressiveness of the controller in recovering from disturbances. $\lambda = 0$ gives a fast, aggressive response, while $\lambda \approx 1$ gives a slow, gentle response. If several outputs are being controlled, it is possible to define a different value of λ for each output. But in a multi-output system it may be difficult to define an appropriate reference trajectory for each output, because different disturbances may result in different ‘natural’ or desirable combinations of responses of the various outputs. In this case the idea of a reference trajectory can be very usefully combined with the *funnel* idea which was described in Section 5.5. A value of λ can be defined for each output which defines, say, the slowest reduction of the error which one would normally like to see on that output. But only errors due to outputs being ‘on the wrong side’ of the reference trajectories are penalized. This idea is implemented in Honeywell’s product *RMPCT*, which is described in Appendix A. In this product, however, the reference trajectory is a straight line from the current output to some point on the future set-point trajectory, rather than an exponential curve. Some further discussion of how a reference trajectory modifies the feedback loop is given in Section 7.5.

There are two reasons why the cost function used in *PFC* often omits any penalty on control moves. One is that the required performance is largely determined by the reference trajectory, so that it is not necessary to use the weights Q and R in the cost function as tuning parameters to obtain the required performance. The other is that the polynomial structure of the future control signal results in relatively few degrees of freedom being available to the optimizer — because low degree polynomials are usually used — and this usually results in relatively smooth input trajectories being obtained. It is therefore not necessary to use a control move penalty to reduce the ‘roughness’ of the input signal.

The results obtained above for error-free tracking of polynomial set-point trajectories continue to hold if such reference trajectories are used, essentially because the reference trajectory coincides (asymptotically) with the set-point trajectory if steady disturbance-free conditions are assumed. For details see [Ric93b].

PFC deals with constraints in a heuristic manner — that is, in a way which does not yield the true constrained optimal solution in general. This approach originates in the application of *PFC* to high-bandwidth servomechanism systems, for which computing the true optimal solution could not be done in real time — especially in the 1970s, when this version of predictive control was originally developed. It is claimed, however, that in many applications the heuristic solution is adequate, particularly with SISO plant.

Constraints on input levels and rates of change are dealt with in *PFC* simply by

'clipping'. Consider a single-input plant for simplicity. Suppose that the constraints on the input level are $u_{min} < u(k) < u_{max}$ and on the input move are $\Delta u_{min} < \Delta u(k) < \Delta u_{max}$, and that the solution algorithm computes an input move $\Delta u(k)^*$, without taking any constraints into account. Then the move actually applied to the plant is (assuming $u_{min} < 0$ and $u_{max} > 0$)

$$\Delta u(k) = \begin{cases} \min(\Delta u(k)^*, \Delta u_{max}, u_{max} - u(k-1)) & \text{if } \Delta u(k)^* > 0 \\ \max(\Delta u(k)^*, \Delta u_{min}, u_{min} - u(k-1)) & \text{if } \Delta u(k)^* < 0 \end{cases} \quad (5.45)$$

Of course the same move must be applied to the internal model as to the plant. This not only helps to keep the predictions accurate, but also avoids problems analogous to 'integrator wind-up'. This heuristic approach will not give the true optimal solution in general, but it often gives very similar results. (See Exercise 5.6 for an example.)

PFC does not have a clear strategy for dealing with state and output constraints. The commercial product based on *PFC*, in common with several other MPC products, adopts heuristics to find a sub-optimal solution when such constraints are active.

5.7

Continuous-time predictive control

It was already remarked in Section 1.3 that predictive control does not require a discrete-time model; it is enough to know the values of the step response at the coincidence points. But it is possible to go further, and formulate the whole predictive control problem, and its solution, in continuous time. This has been proposed by Gawthrop *et al* [DG91, DG92, GDSA98].

They propose a quadratic cost function of the form

$$V(t) = \int_{\tau_1}^{\tau_2} \|r(t+\tau) - \hat{y}(t+\tau|t)\|^2 d\tau + \lambda \int_{\tau_1}^{\tau_2} \|\hat{u}(t+\tau|t)\|^2 d\tau \quad (5.46)$$

and the use of a continuous-time model. Predictions of the output are formed by using a truncated Taylor series expansion:

$$\hat{y}(t+\tau|t) = y(t) + \tau \dot{y}(t) + \frac{\tau^2}{2!} \ddot{y}(t) + \cdots + \frac{\tau^n}{n!} y^{(n)}(t) \quad (5.47)$$

A Taylor series is also used to parametrize the future input trajectory, the parameters being the current input (to be applied to the plant) and its first m time derivatives:

$$\hat{u}(t+\tau|t) = u(t) + \tau \dot{u}(t) + \frac{\tau^2}{2!} \ddot{u}(t) + \cdots + \frac{\tau^m}{m!} u^{(m)}(t) \quad (5.48)$$

m plays a role analogous to the control horizon H_u in the discrete-time formulation; it restricts the complexity of the assumed future input trajectory, in the sense that only the first m derivatives of $\hat{u}(t+\tau|t)$ (*with respect to* τ) are non-zero. (Note the conceptual similarity of this to the polynomial basis used in *PFC*.)

If a linear model is used then the optimal parameters $u(t), \dot{u}(t), \dots, u^{(m)}(t)$ can be found by solving a least-squares problem, if there are no constraints. Suppose that a

state-space model is used:

$$\dot{x}(t) = Ax(t) + Bu(t), \quad y(t) = Cx(t) \quad (5.49)$$

Then

$$\begin{aligned} \mathcal{Y}(t) &= \begin{bmatrix} y(t) \\ \dot{y}(t) \\ \vdots \\ y^{(n)}(t) \end{bmatrix} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^n \end{bmatrix} x(t) \\ &+ \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 \\ CB & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ CA^{n-1}B & CA^{n-2}B & \cdots & CB & 0 \end{bmatrix} \begin{bmatrix} u(t) \\ \dot{u}(t) \\ \vdots \\ u^{(n-1)}(t) \end{bmatrix} \end{aligned} \quad (5.50)$$

so the prediction (5.47) can be obtained if it is assumed that the current state is known, and it is linear in $u(t), \dot{u}(t), \dots, u^{(m)}(t)$. Now if (5.47) is written in the form

$$\hat{y}(t + \tau | t) = \left[1, \tau, \dots, \frac{\tau^n}{n!} \right] \mathcal{Y}(t) \quad (5.51)$$

then, assuming for simplicity that there is only one output, so that y and r are scalars,

$$\begin{aligned} \int_{\tau_1}^{\tau_2} \|r(t + \tau) - \hat{y}(t + \tau | t)\|^2 d\tau &= \mathcal{Y}(t)^T \Phi(\tau_1, \tau_2) \mathcal{Y}(t) - 2\phi(\tau_1, \tau_2) \mathcal{Y}(t) \\ &+ \int_{\tau_1}^{\tau_2} r(t + \tau)^2 d\tau \end{aligned} \quad (5.52)$$

where

$$\Phi(\tau_1, \tau_2) = \int_{\tau_1}^{\tau_2} \begin{bmatrix} 1 \\ \tau \\ \dots \\ \frac{\tau^n}{n!} \end{bmatrix} \left[1, \tau, \dots, \frac{\tau^n}{n!} \right] d\tau \quad (5.53)$$

and

$$\phi(\tau_1, \tau_2) = \int_{\tau_1}^{\tau_2} \left[1, \tau, \dots, \frac{\tau^n}{n!} \right] r(t + \tau) d\tau \quad (5.54)$$

One can find a similar expression for $\int_{\tau_1}^{\tau_2} \|\hat{u}(t + \tau | t)\|^2 d\tau$ using (5.48).

Thus the cost function $V(t)$ is a quadratic function of $\mathcal{Y}(t)$ and of the input and its derivatives, and $\mathcal{Y}(t)$ is itself linear in the input and its derivatives, so $V(t)$ is a quadratic function of the parameters $u(t), \dot{u}(t), \dots, u^{(m)}(t)$. So minimizing it requires the solution of a least-squares problem, as was claimed earlier. Although it is not explored in the work of Gawthrop *et al*, the addition of linear inequality constraints on the inputs and outputs, and possibly on their derivatives, would give rise to a quadratic programming problem, just as in the discrete-time case.

While this formulation nicely parallels the discrete-time formulation, the proposal is not complete, in that it does not specify how the solution should be implemented. Conceptually, the optimal $u(t)$ is computed and applied continuously — the derivatives found as part of the solution are not used, which is analogous to discarding all but the initial part of the optimal input trajectory in the discrete-time formulation. In practice this would not be feasible, and presumably some kind of frequent update would need to be implemented, with care being taken to impose sufficient smoothness on the input trajectory actually applied to the plant, for it to be consistent with the use of (5.50). The use of Taylor series for prediction is very vulnerable to measurement errors, since it relies on knowing high-order time derivatives. In [DG91, DG92, GDSA98] there is some discussion of how the robustness of the continuous-time scheme can be increased.

Exercises

- 5.1** Consider the swimming pool from Exercise 3.3, with a sinusoidal diurnal variation of the ambient air temperature θ_a defined as in (3.96). Suppose that the air temperature is now measured and used for feedforward. Assume that weights and horizons remain the same as earlier.
- Show by simulation that if the model is perfect and there are no constraints on the input power then the control system compensates perfectly for the variation in air temperature.
 - If there is a modelling error, as in Exercise 3.3(c), show that there is a residual variation of approximately $0.2\text{ }^{\circ}\text{C}$ in the water temperature. (Recall that when the air temperature is an unmeasured disturbance and the power is unconstrained, the residual variation in θ is approximately $0.5\text{ }^{\circ}\text{C}$, with both the perfect and the imperfect models.)
 - If the input power is constrained to $0 \leq q \leq 40\text{ kW}$, show that the set-point is held much more accurately when the air temperature is measured and used for feedforward, but only in those periods when the input constraints are not active. In those periods when the input power is constrained the water temperature variations are almost the same, regardless of whether the air temperature is measured or not.

- 5.2** Suppose that the ‘1-norm’ cost function (5.25) is augmented by penalties on input moves:

$$V(k) = \sum_{i=1}^{H_p} \sum_{j=1}^p |\hat{z}_j(k+i|k) - r_j(k+i)|q_j + \sum_{i=1}^{H_u} \sum_{j=1}^m |\Delta \hat{u}_j(k+i|k)|r_j \quad (5.55)$$

where the r_j s are nonnegative weights. Show that minimizing this, subject to the usual constraints, is an LP problem.

- 5.3** Consider the unstable helicopter model of Example 4.11, with transfer function

$$P(z) = \frac{6.472z^2 - 2.476z + 7.794}{(z - 0.675)(z - 1.047 + 0.2352i)(z - 1.047 - 0.2352i)}$$

Find a pair of coprime factors $\tilde{N}(z)$, $\tilde{M}(z)$, such that $P(z) = \tilde{N}(z)/\tilde{M}(z)$ and both $\tilde{N}(z)$ and $\tilde{M}(z)$ are stable. Ensure that each factor is proper (numerator degree does not exceed denominator degree), so that each could be realized as a state-space system. (Note: there are many possible solutions.)

Note that finding a ‘decomposition’ of the form (5.22) is harder.

- 5.4** Using the notation of Sections 1.3 and 5.6, show that the predicted *model* output resulting from the input $u_0(k) + u_1(k)i$ is

$$\begin{aligned}\hat{y}(k+H_p|k) &= S(H_p)u_0(k) + [S(H_p - 1) + \dots \\ &\quad + S(1)]u_1(k) + \hat{y}_f(k+H_p|k)\end{aligned}$$

(where \hat{y}_f denotes the predicted free response).

- 5.5** Suppose that an asymptotically stable plant has one controlled output, which is to follow a ramp set-point without error, using PFC.

- (a) What is the smallest number of coincidence points that must be used?
- (b) If there is only one input, and a polynomial basis is used to structure the assumed future input trajectory, what degree of polynomial should be used?
- (c) Suppose two inputs are available. Discuss the possibilities for structuring the future input trajectories.
- (d) What other feature must be included in the control scheme, if error-free tracking of the ramp is to be obtained?

- 5.6** Consider the swimming pool from Exercise 3.3, with a sinusoidal diurnal variation of the ambient air temperature θ_a defined as in (3.96), and with the input power constraint $0 \leq q \leq 40$ kW.

- (a) Show that in this case the behaviour obtained by ‘clipping’ the unconstrained solution is the same as that obtained by solving the QP problem.
- (b) If there is also a slew rate constraint of $|\dot{q}| < 20$ kW/hour, show that the QP and clipped solutions are no longer the same, although they are still very close.

Note: use the *Model Predictive Control Toolbox* function `smpcsim` to get the clipped solution.

- 5.7** Verify that the vector $\mathcal{Y}(t)$, containing $y(t)$ and its time derivatives, is given by (5.50).

- 5.8** Define a vector $\mathcal{U}(t)$, analogous to $\mathcal{Y}(t)$, which contains $u(t)$ and its time derivatives. Express the continuous-time cost $V(t)$, defined by (5.46), as a quadratic function of $\mathcal{U}(t)$.

Chapter 6

Stability

6.1	Terminal constraints ensure stability	169
6.2	Infinite horizons	172
6.3	Fake algebraic Riccati equations	180
6.4	Using the Youla parametrization	183
	Exercises	185

If one is designing a predictive controller off-line, then nominal stability of the closed loop is hardly an issue. It is quite easy to obtain stability by tuning the parameters in the problem formulation, and very easy to check that the designed system is stable (assuming the correct plant model). This at least is the typical situation with current applications of predictive control. Current applications are invariably implemented on stable plants, and the dynamic performance demanded from predictive controllers is typically not very challenging. As predictive control is applied more widely, with tight dynamic specifications, and especially if it is applied to unstable plants, then this situation may change.

Closed-loop stability is much more of an issue if predictive control is being used in a way which requires any on-line redesign. This includes adaptive control in the traditional set-up, where re-estimation of the plant model and consequent redesign of the controller is assumed to occur continuously. It also includes situations such as occasional re-estimation followed by redesign, for example following a failure or a change in the plant configuration.

In this chapter several ways of guaranteeing nominal stability are investigated. It will be seen that there are now several ways of ensuring that the nominal closed loop, at least — that is, with the assumption that the model is perfect — is stable.

Predictive control, using the receding horizon idea, is a feedback control policy. There is therefore a risk that the resulting closed loop might be unstable. Even though the performance of the plant is being optimized over the prediction horizon, and even though the optimization keeps being repeated, each optimization ‘does not care’ about what

happens beyond the prediction horizon, and so could be putting the plant into such a state that it will eventually be impossible to stabilize. This is particularly likely to occur when there are constraints on the possible control input signals. It is easy to construct an example where this happens, even without constraints:

Example 6.1

Consider a discrete-time system consisting of two delays in series:

$$x_1(k+1) = u(k)$$

$$x_2(k+1) = x_1(k)$$

or, using vector-matrix notation:

$$x(k+1) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} x(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k)$$

Suppose the prediction horizon is $H_p = 1$, and the cost to be minimized is

$$\begin{aligned} V(k) &= \hat{x}_1(k+1|k)^2 + 6\hat{x}_2(k+1|k)^2 + 4\hat{x}_1(k+1|k)\hat{x}_2(k+1|k) \\ &= \hat{x}(k+1|k)^T \begin{bmatrix} 1 & 2 \\ 2 & 6 \end{bmatrix} \hat{x}(k+1|k) \end{aligned}$$

(which is positive-definite). The predictive control problem is to find $u(k)$ which will minimize $V(k)$.

Using the model, the predictions are $\hat{x}_1(k+1|k) = u(k)$ and $\hat{x}_2(k+1|k) = x_1(k)$. Substituting these into $V(k)$ gives

$$V(k) = u(k)^2 + 6x_1(k)^2 + 4u(k)x_1(k).$$

To find the minimum of this, differentiate and set the derivative to zero:

$$\begin{aligned} \frac{\partial V}{\partial u(k)} &= 2u(k) + 4x_1(k) \\ &= 0 \text{ if } u(k) = -2x_1(k). \end{aligned}$$

Substituting this back into the model gives the closed-loop equations:

$$x_1(k+1) = -2x_1(k)$$

$$x_2(k+1) = x_1(k)$$

or

$$x(k+1) = \begin{bmatrix} -2 & 0 \\ 1 & 0 \end{bmatrix} x(k).$$

Clearly this is unstable, since the closed-loop transition matrix has an eigenvalue at -2 , outside the unit circle.

One might guess that this problem arises in the example because the prediction horizon is too short, so that the control is too ‘short-sighted’. This is correct, and it turns out that stability can usually be ensured by making the prediction horizon long enough, or even infinite. We will see that later in this chapter.

6.1 Terminal constraints ensure stability

Another way of ensuring stability is to have any length of horizon, but to add a ‘terminal constraint’ which forces the state to take a particular value at the end of the prediction horizon. And it is surprisingly easy to prove this, even in a very general case, by using a Lyapunov function, as was first shown by Keerthi and Gilbert [KG88]. For the purpose of proving stability it is enough to consider the case when the state is to be driven to the origin, from some initial condition.

Theorem 6.1

Suppose that predictive control is obtained for the plant

$$x(k+1) = f(x(k), u(k))$$

by minimizing the cost function

$$V(k) = \sum_{i=1}^{H_p} \ell(\hat{x}(k+i|k), \hat{u}(k+i-1|k))$$

where $\ell(x, u) \geq 0$ and $\ell(x, u) = 0$ only if $x = 0$ and $u = 0$, and ℓ is ‘decreasing’, subject to the terminal constraint

$$\hat{x}(k + H_p|k) = 0$$

The minimization is over the input signals $\{\hat{u}(k+i|k) : i = 0, 1, \dots, H_u - 1\}$, with $H_u = H_p$ (for simplicity), subject to the constraints $\hat{u}(k+i|k) \in U$ and $\hat{x}(k+i|k) \in X$, where U and X are some sets. We assume that $x = 0, u = 0$ is an equilibrium condition for the plant: $0 = f(0, 0)$. The receding horizon method is applied, with only $u^0(k|k)$ being used from the optimal solution $\{u^0(k+i|k) : i = 0, \dots, H_u - 1\}$.

Then the equilibrium point $x = 0, u = 0$ is stable, providing that the optimization problem is feasible and is solved at each step.

Proof 6.1

The proof makes use of concepts from Mini-Tutorial 6. Let $V^0(t)$ be the optimal value of V which corresponds to the optimal input signal u^0 , as evaluated at time t . Clearly $V^0(t) \geq 0$, and $V^0(t) = 0$ only if $x(t) = 0$ — because if $x(t) = 0$ then the optimal thing to do is to set $u(t+i) = 0$ for each i . We are going to show that $V^0(t+1) \leq V^0(t)$, and hence that $V^0(t)$ is a Lyapunov function for the closed-loop system.

As usual in stability proofs, we will assume that the plant model is perfect, so that the predicted and real state trajectories coincide: $x(t+i) = \hat{x}(t+i|t)$ if $u(t+i) = \hat{u}(t+i|t)$.

Equilibrium: The equation $x(k+1) = f(x(k), u(k))$ has an equilibrium (or ‘fixed point’) at state x_0 and input u_0 if $x_0 = f(x_0, u_0)$.

Note that we can always introduce a change of coordinates $z(k) = x(k) - x_0$, $v(k) = u(k) - u_0$, so that the equilibrium is at $(0, 0)$ in the new coordinates:

$$\begin{aligned} z(k+1) &= x(k+1) - x_0 = f(z(k) + x_0, v(k) + u_0) - f(x_0, u_0) \\ &= g(z(k), v(k)) \\ &= g(0, 0) \end{aligned}$$

So we can always assume that an equilibrium is at $(0, 0)$.

Stability: For nonlinear systems one has to consider the stability of a particular equilibrium point, rather than the system. (Some equilibria may be stable, others unstable.)

An equilibrium point $(0, 0)$ is *stable* (in the sense of Lyapunov) if a small perturbation of the state or input results in a ‘continuous’ perturbation of the subsequent state and input trajectories. More precisely, given any $\epsilon > 0$, there is a $\delta > 0$ (which depends on $\epsilon > 0$ and the system) such that if $\|[(x(0)^T, u(0)^T)]\| < \epsilon$ then $\|[(x(k)^T, u(k)^T)]\| < \delta$ for all $k > 0$.

It is *asymptotically stable* if in addition $\|[(x(k)^T, u(k)^T)]\| \rightarrow 0$ as $k \rightarrow \infty$.

For closed-loop systems $u(k)$ itself depends on $x(k)$, so we could have written $x(k+1) = f(x(k))$, where f now represents the closed-loop state equation.

Lyapunov’s Theorem: If there is a function $V(x, u)$ which is positive-definite, namely such that $V(x, u) \geq 0$ and $V(x, u) = 0$ only if $(x, u) = (0, 0)$, and has the (‘decreasing’) property that

$$\|[(x_1^T, u_1^T)]\| > \|[(x_2^T, u_2^T)]\| \Rightarrow V(x_1, u_1) \geq V(x_2, u_2)$$

and if, along any trajectory of the system $x(k+1) = f(x(k), u(k))$ in some neighbourhood of $(0, 0)$ the property

$$V(x(t+1), u(t+1)) \leq V(x(t), u(t))$$

holds, then $(0, 0)$ is a stable equilibrium point.

If, in addition, $V(x(t), u(t)) \rightarrow 0$ as $t \rightarrow \infty$ then it is asymptotically stable.

Such a function V is called a *Lyapunov function*.

Mini-Tutorial 6 Stability and Lyapunov functions.

With this assumption we have

$$\begin{aligned} V^0(t+1) &= \min_u \sum_{i=1}^{H_p} \ell(x(t+1+i), u(t+i)) \\ &= \min_u \left\{ \sum_{i=1}^{H_p} \ell(x(t+i), u(t-1+i)) - \ell(x(t+1), u(t)) \right. \\ &\quad \left. + \ell(x(t+1+H_p), u(t+H_p)) \right\} \\ &\leq -\ell(x(t+1), u^0(t)) + V^0(t) \\ &\quad + \min_u \{\ell(x(t+1+H_p), u(t+H_p))\} \end{aligned}$$

since the optimum is certainly no worse than keeping the optimal solution found at time t , which will take us up to time $t + H_p$, then doing the best we can for the final step. But we have assumed that the constraint $x(t + H_p) = 0$ is satisfied, since the optimization problem was assumed to be feasible, so we can make $u(t + H_p) = 0$ and stay at $x = 0$, which gives

$$\min_u \{ \ell(x(t + 1 + H_p), u(t + H_p)) \} = 0.$$

Since $\ell(x(t), u^0(t)) \geq 0$, we can conclude that $V^0(t + 1) \leq V^0(t)$. So $V^0(t)$ is a Lyapunov function, and we conclude by Lyapunov's stability theorem that the equilibrium $x = 0, u = 0$ is stable.

This sounds too good to be true. It seems too easy to guarantee stability. The problem is with the assumption that the optimization problem has a solution at each step, and that the global optimum can be found at each step. General constrained optimization problems can be extremely difficult to solve, and just adding a terminal constraint may not be feasible. But we will use the idea of this proof several times later, in more realistic situations.

Example 6.2

If we look again at Example 6.1 and try to get stability by adding the terminal constraint $x(k + 1) = 0$, we get an infeasible problem, because that could only be achieved if $x_1(k) = 0$, which will not be true in general. (In general you need at least two steps to drive a 2-state linear system to the origin.) Exercise 6.2 shows that for this example stability is obtained by increasing the prediction horizon to $H_p = 2$, even without a terminal constraint.

The same basic idea can be applied in various ways. Two particular ways, associated with GPC, have become quite well known and have been incorporated into particular versions of GPC: ‘Stable input–output receding horizon control’, or *SIORHC* (see [Mos95, Chapter 5.8]), and ‘Constrained receding horizon predictive control’, or *CRHPC* (see [CC95, Chapter 6.5]).

However, since we shall see in the following sections that stability can be achieved without imposing conditions as severe as single-point terminal constraints, we shall not pursue this approach further.

A very important generalization, and relaxation, of the idea of terminal constraints is to specify a terminal constraint set X_0 , which contains the origin, rather than a single point. Usually, the idea is to use predictive control to drive the state into such a set, and then to switch to some other control law, which can be guaranteed to stabilize the system for initial conditions within X_0 , assuring that the state will be driven to the origin, providing that it has first been driven into X_0 . It is assumed that all constraints are inactive within X_0 , so that conventional control (usually but not necessarily linear) is adequate when the state is within this terminal set. The idea of using a terminal constraint set was introduced in [MM93], as part of a scheme for obtaining robustly stable predictive control of nonlinear systems. Some more remarks about it can be found in Sections 8.5.2 and 10.3. Schemes which involve first using predictive control to drive

the state into a terminal constraint set, then switching to another control law, are often called *dual-mode* predictive control schemes.

It has recently been argued [MRRS00] that all MPC schemes which have been proposed to date and which guarantee closed-loop stability, have a terminal constraint set built into them (possibly implicitly, and possibly consisting of a single point). Furthermore, they contain a control law defined on this terminal set which respects all the constraints, and which keeps the state in the set, once the state trajectory has entered it. Finally, they all contain (possibly implicitly) a terminal *cost*, that is, a cost penalizing non-zero states at the end of the prediction horizon; furthermore, the terminal cost function is such that it is a Lyapunov function when confined to the terminal constraint set. In many of the published proposals one or more of these elements are trivial — for instance, the terminal constraint set may consist of only the origin, the control law on this set (point) may be the prescription $u \equiv 0$, and the terminal cost function may be $F(x) = \infty$ if $x \neq 0$, $F(x) = 0$ if $x = 0$. Nevertheless, the explicit identification of these elements may be valuable for future proposals; in particular the recognition that both a terminal constraint set and a terminal cost can usefully be part of a successful ‘prescription’ is a surprising revelation.

6.2

Infinite horizons

The specific approach described in this section was originated in [RM93], and elaborated upon in [MR93b, MR93a], although the underlying ideas are similar to those in [KG88], where it was shown that infinite-horizon constrained control problems could be approximated by finite-receding-horizon problems with a terminal constraint. It had been known for some time (e.g. see [BGW90]) that making the horizons infinite in predictive control leads to guaranteed stability, but it was not known how to handle constraints with infinite horizons. The work of Rawlings and Muske made a breakthrough in this respect. The same ideas have also been applied to the transfer-function formulation [Sco97] — not surprisingly, the whole development can be repeated in that setting.

The key idea is to reparametrize the predictive control problem with infinite horizons in terms of a finite number of parameters (instead of the infinite number of control decisions $\Delta u(k|k)$, $\Delta u(k+1|k)$, ...), so that the optimization can still be performed over a finite-dimensional space — in fact, it remains a QP problem. Since the original work of Rawlings and Muske, several other proposals have been made, which can be interpreted as different reparametrizations of the infinite horizon problem [RK93, SR96b].

In [Mor94] the opinion has been expressed that there is now no longer any reason to use finite horizons — at least with linear models.

Let us begin by examining why the use of an infinite costing horizon guarantees stability.

6.2.1 Infinite horizons give stability

Figure 6.1 shows an essential difference between a finite, receding horizon formulation of predictive control, and an infinite horizon formulation. The top half of the figure

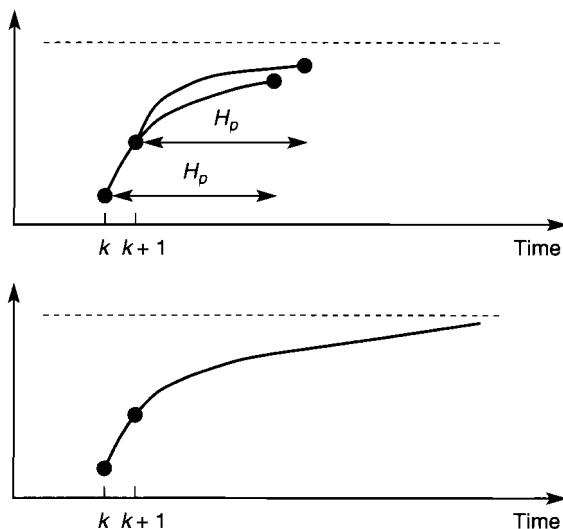


Figure 6.1 Finite and infinite horizons (no disturbances, perfect model).

depicts the finite-horizon formulation, with the plant output heading towards a constant set-point. At time k a particular trajectory is optimal over the prediction horizon of length H_p . In the absence of disturbances, and with a perfect model, the plant at time $k+1$ is precisely in the state which was predicted at the previous time step. One might expect, therefore, that the initial portion of the optimal trajectory over the prediction horizon from time $k+1$ to time $k+1+H_p$ would coincide with the previously computed optimal trajectory. But a new time interval now enters the picture which had not been considered when the earlier optimization had been performed — the interval between time $k+H_p$ and time $k+1+H_p$. The presence of this new time interval may lead to an optimal trajectory completely different from the one computed at the earlier step.

The lower half of Figure 6.1 shows the situation with an infinite horizon. At time k an optimal trajectory over the whole infinite horizon is determined (implicitly). At time $k+1$ no new information enters the optimization problem, so the optimal trajectory from this time on is the same as the ‘tail’ of the previously computed trajectory. (This is Bellman’s celebrated *principle of optimality*, which states that the tail of any optimal trajectory is itself the optimal trajectory from its starting point. It does not apply in the finite-horizon case, because there a different optimization problem arises at each step.) This leads to the cost function $V(k)$ decreasing as k increases (under the no-disturbance, perfect-model assumption), which allows it to be used as a Lyapunov function, hence establishing stability.

Figure 6.2 shows a particular case for which closed-loop stability can be established despite the use of a finite horizon. This is when deadbeat behaviour is enforced, with settling to the set-point occurring within the prediction horizon. In this case the situation is similar to the infinite-horizon case, in that at each step the optimal trajectory is the ‘tail’ of the previously computed one. (It can also be regarded as a finite-horizon case

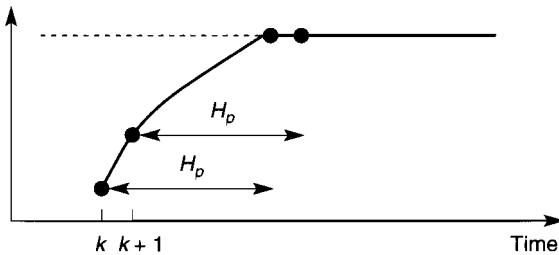


Figure 6.2 Finite horizon and deadbeat response.

with terminal constraint.) So in this case $V(k)$ is also decreasing, and hence stability can be established. This strategy for ensuring closed-loop stability was used in [KR92].

Let us now consider the details of the infinite-horizon formulation. We modify our standard set-up a little. First, we keep the weights constant across the prediction horizon, so that $Q(i) = Q$ and $R(i) = R$. Secondly, we penalize control *levels* as well as moves over an infinite horizon. Thirdly, we assume a ‘regulator’ formulation, namely that we want to drive the output to zero:

$$\dot{V}(k) = \sum_{i=1}^{\infty} \left\{ \|\hat{z}(k+i|k)\|_Q^2 + \|\Delta\hat{u}(k+i-1|k)\|_R^2 + \|\hat{u}(k+i-1|k)\|_S^2 \right\} \quad (6.1)$$

although we still assume that only the first H_u control moves are non-zero:

$$\Delta\hat{u}(k+i-1|k) = 0 \quad \text{for } i > H_u \quad (6.2)$$

We assume that $S > 0$ and $Q \geq 0$, $R \geq 0$. (In [RM93] it was assumed that the control levels were penalized, not the control moves. In [MR93b] both control levels and control moves were penalized. A proof which holds in the general case is given in [MR93a]. Unfortunately the weights R and S in our notation have the opposite meanings to those used in [RM93, MR93b, MR93a].)

Suppose initially that we have full state measurement, so that $x(k) = y(k)$, and that the plant is stable. Then the following argument shows that closed-loop stability is obtained for any $H_u > 0$, providing that the optimization problem is, and remains, feasible, and that the global optimum is found at each step.

Let $V^0(k)$ be the optimal value of the cost function at time k , let u^0 denote the computed optimal input levels, and let z^0 denote values of the controlled output obtained as a result of applying u^0 . Note that, since $u^0(k+i) = u^0(k+H_u-1)$ for all $i \geq H_u$, and since the steady-state value of u^0 needed to keep z at 0 is 0 (according to the optimizer’s internal model), the optimizer will certainly put $u^0(k+i) = 0$ for $i \geq H_u$, since an infinite cost would otherwise result. Thus we have

$$V^0(k) = \sum_{i=1}^{\infty} \|\hat{z}^0(k+i|k)\|_Q^2 + \sum_{i=1}^{H_u} \left\{ \|\Delta\hat{u}^0(k+i-1|k)\|_R^2 + \|\hat{u}^0(k+i-1|k)\|_S^2 \right\} \quad (6.3)$$

Now if we assume, as usual in proofs of nominal stability, that our model is exact and that there are no disturbances, then $z^0(k+1) = \hat{z}^0(k+1|k)$. Thus the value of V , evaluated at time $k+1$, but maintaining the same control sequence $\hat{u}^0(k+i|k)$ as computed at time k , would be

$$V(k+1) = V^0(k) - \|z^0(k+1)\|_Q^2 - \|\Delta\hat{u}^0(k|k)\|_R^2 - \|\hat{u}^0(k|k)\|_S^2 \quad (6.4)$$

But at time $k+1$ the new optimization problem, with initial condition $z^0(k+1) = C_z x^0(k+1)$ — which is known, since we assume full state measurement — is solved, so that

$$V^0(k+1) \leq V(k+1) \quad (6.5)$$

$$= V^0(k) - \|z^0(k+1)\|_Q^2 - \|\Delta\hat{u}^0(k|k)\|_R^2 - \|\hat{u}^0(k|k)\|_S^2 \quad (6.6)$$

$$< V^0(k) \quad (6.7)$$

the strict inequality being due to the assumption that $S > 0$.

To infer stability we must now show that $V^0(k+1) < V^0(k)$ implies that $\|x(k)\|$ is decreasing. Since we have assumed that $S > 0$, it is clear that $u^0(k)$ is decreasing (since $u^0(k) = \hat{u}^0(k|k)$), which implies that $x^0(k)$ is eventually decreasing, since we have assumed a stable plant. But we can relax the $S > 0$ condition, if we replace it by something else. For example, assuming that $Q > 0$ and observability of the state from z implies that $\|x^0\|$ must eventually be decreasing, even if $S = 0$, as in our standard formulation. So there are a number of possible assumptions which ensure that decreasing V^0 implies decreasing $\|x^0\|$, and that is enough to show that V^0 is a Lyapunov function for the closed loop, which shows, in turn, that the closed-loop system is stable.

Now we must consider the case of an unstable plant. The essential difference in this case is that the unstable modes must be driven to 0 within H_u steps. Otherwise these modes, which are uncontrolled for $i \geq H_u$, would become unbounded, giving an infinite cost value. But it is known that in general it is not possible to drive the states of a system to zero in less than n steps, where n is the state dimension. Since only the unstable modes need to be driven to zero, only n_u steps are actually needed, if n_u is the number of unstable modes. Thus we have to take $H_u \geq n_u$ for an unstable plant. We also need to ensure that the unstable modes *can* be driven to zero by some input. Technically, we have to assume that the pair (A, B) is *stabilizable* — namely that the unstable modes are controllable from the input; the stable modes do not have to be controllable, since they decay to zero anyway.

With these provisos, stability follows by the same argument as for stable plants, providing that feasibility is maintained.

When full state measurement is not available ($y(k) \neq x(k)$) an observer must be used to obtain the state estimate $\hat{x}(k|k)$. Subject to a few technical conditions, and assuming a perfect model as usual, $\hat{x}(k|k) \rightarrow x(k)$ as $k \rightarrow \infty$ if the observer dynamics are stable. Consequently, as k increases, the value of the optimal cost function $V^0(k)$ will approach the same value as in the case of $y(k) = x(k)$, and stability can again be inferred.

In order to be able to pose problems which are feasible over the whole infinite horizon, it is necessary to introduce two new parameters C_w and C_p , which are analogous to the

horizons H_w and H_p , but which refer to the times over which output constraints are enforced. To be precise, suppose that the output constraints are of the form

$$z_{i,min} \leq \hat{z}_i(k+j) \leq z_{i,max} \quad \text{for } j = C_w, C_w + 1, \dots, C_p. \quad (6.8)$$

In order to apply the stability theorems, both C_w and C_p must be chosen large enough. C_w must be chosen large enough that the problem is feasible at time k . C_p must be chosen large enough that if the problem is feasible over the finite horizon up to time $k + C_p$, then it will remain feasible over the rest of the infinite horizon. In [RM93] it is shown that finite values of C_w and C_p always exist — although for unstable plants they depend on $x(k)$, in general.

6.2.2 Constraints and infinite horizons — stable plant

Now we need to consider how constrained predictive control problems can be solved, when the horizons are infinite. As we have already seen, for the regulator problem the computed input signal $\hat{u}(k+i-1|k)$ is necessarily zero for $i \geq H_u$. The cost function can therefore be written as

$$\begin{aligned} V(k) = & \sum_{i=H_u+1}^{\infty} \|\hat{z}(k+i-1|k)\|_Q^2 + \sum_{i=1}^{H_u} \left\{ \|\hat{z}(k+i-1|k)\|_Q^2 \right. \\ & \left. + \|\Delta\hat{u}(k+i-1|k)\|_R^2 + \|\hat{u}(k+i-1|k)\|_S^2 \right\} \end{aligned} \quad (6.9)$$

Consider the first term in this expression. Since $\hat{u}(k+i-1|k) = 0$ for $i \geq H_u$, we have

$$\hat{z}(k+H_u|k) = C_z \hat{x}(k+H_u|k) \quad (6.10)$$

$$\hat{z}(k+H_u+1|k) = C_z A \hat{x}(k+H_u|k) \quad (6.11)$$

⋮

$$\hat{z}(k+H_u+j|k) = C_z A^j \hat{x}(k+H_u|k) \quad (6.12)$$

so that

$$\sum_{i=H_u+1}^{\infty} \|\hat{z}(k+i|k)\|_Q^2 = \hat{x}(k+H_u|k)^T \left[\sum_{i=0}^{\infty} (A^T)^i C_z^T Q C_z A^i \right] \hat{x}(k+H_u|k) \quad (6.13)$$

Now let

$$\bar{Q} = \sum_{i=0}^{\infty} (A^T)^i C_z^T Q C_z A^i \quad (6.14)$$

(the series converges if the plant is stable) then

$$A^T \bar{Q} A = \sum_{i=1}^{\infty} (A^T)^i C_z^T Q C_z A^i \quad (6.15)$$

$$= \bar{Q} - C_z^T Q C_z \quad (6.16)$$

This equation is known as the *matrix Lyapunov equation*, and it can be solved for \bar{Q} , given A , C_z and Q . (In MATLAB it can be solved using the function `dlyap` from the *Control System Toolbox*.) Furthermore, it is well known that $\bar{Q} \geq 0$ if $Q \geq 0$ and A has all its eigenvalues inside the unit disc, namely if the plant is stable. Thus the cost function can be written as

$$V(k) = \hat{x}(k + H_u|k)^T \bar{Q} \hat{x}(k + H_u|k) + \sum_{i=1}^{H_u} \left\{ \|\hat{z}(k + i - 1|k)\|_Q^2 + \|\Delta \hat{u}(k + i - 1|k)\|_R^2 + \|\hat{u}(k + i - 1|k)\|_S^2 \right\} \quad (6.17)$$

This now looks like a predictive control problem formulated over a finite horizon of length H_u , with a terminal cost penalty — which shows, incidentally, that imposing a terminal equality constraint to obtain stability is unnecessarily restrictive.

If the controlled output is to follow a set-point with a non-zero final value, then it is necessary to replace the term $\|\hat{u}(k + i - 1|k)\|_S^2$ by $\|\hat{u}(k + i - 1|k) - u_s\|_S^2$ in the cost function in order to get a bounded cost, where u_s is a steady input value which is predicted to bring the controlled output to the required steady value r_s :

$$r_s = P(1)u_s = C_z(I - A)^{-1}Bu_s \quad (6.18)$$

But this is unnecessary if $S = 0$, as in our standard formulation. If a future target trajectory is known over a horizon H_p longer than H_u , then the approach outlined above is easily modified to handle this. It is simply necessary to rewrite the problem as a finite-horizon problem with horizon H_p rather than H_u , with a terminal cost of the form $\hat{x}(k + H_p|k)^T \bar{Q} \hat{x}(k + H_p|k)$.

In order to formulate this predictive control problem as a standard QP problem, we proceed as in Chapters 2 and 3. First we form the vector of predicted states $[\hat{x}(k + 1|k)^T, \dots, \hat{x}(k + H_u|k)^T]^T$ exactly as in equation (2.23) — but noting that the number of predictions needed now is H_u . The only other change that is needed is that Q in (3.3) should be replaced by

$$Q = \underbrace{\begin{bmatrix} Q & 0 & \cdots & 0 & 0 \\ 0 & Q & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & Q & 0 \\ 0 & 0 & \cdots & 0 & \bar{Q} \end{bmatrix}}_{H_u \text{ blocks}} \quad (6.19)$$

(Note that the last block is different from the others.)

Example 6.3

In [BGW90] the following plant is used as an example of one which is difficult to control:

$$P(z) = \frac{-0.0539z^{-1} + 0.5775z^{-2} + 0.5188z^{-3}}{1 - 0.6543z^{-1} + 0.5013z^{-2} - 0.2865z^{-3}} \quad (6.20)$$

One state-space realization of this has

$$A = \begin{bmatrix} 0.6543 & -0.5013 & 0.2865 \\ 1.0000 & 0 & 0 \\ 0 & 1.0000 & 0 \end{bmatrix} \quad C_z = [-0.0536, 0.5775, 0.5188] \quad (6.21)$$

Suppose that $Q = I_3$, then

$$\bar{Q} = \begin{bmatrix} 1.2431 & -0.1011 & 0.1763 \\ -0.1011 & 0.8404 & 0.1716 \\ 0.1763 & 0.1716 & 0.3712 \end{bmatrix} \quad (6.22)$$

6.2.3 Constraints and infinite horizons — unstable plant

With an unstable plant there are two reasons why we cannot proceed exactly as before:

- We need to impose the constraint that the unstable modes must be at zero at the end of the control horizon.
- The infinite series in (6.14) does not converge.

We proceed by decomposing the plant into its stable and unstable parts, by means of an eigenvalue–eigenvector (Jordan) decomposition:¹

$$A = W J W^{-1} \quad (6.23)$$

$$= [W_u, W_s] \begin{bmatrix} J_u & 0 \\ 0 & J_s \end{bmatrix} \begin{bmatrix} \tilde{W}_u \\ \tilde{W}_s \end{bmatrix} \quad (6.24)$$

The unstable and stable modes can be obtained from this as

$$\begin{bmatrix} \xi_u(k) \\ \xi_s(k) \end{bmatrix} = \begin{bmatrix} \tilde{W}_u \\ \tilde{W}_s \end{bmatrix} x(k) \quad (6.25)$$

and they evolve according to

$$\begin{bmatrix} \xi_u(k+1) \\ \xi_s(k+1) \end{bmatrix} = \begin{bmatrix} J_u & 0 \\ 0 & J_s \end{bmatrix} \begin{bmatrix} \xi_u(k) \\ \xi_s(k) \end{bmatrix} + \begin{bmatrix} \tilde{W}_u \\ \tilde{W}_s \end{bmatrix} B u(k) \quad (6.26)$$

The terminal equality constraint on the unstable modes becomes

$$\xi_u(k+H_u) = \tilde{W}_u x(k+H_u) = 0 \quad (6.27)$$

¹ There is no numerically stable algorithm for performing the Jordan decomposition. But there are numerically stable algorithms for finding the subspaces of the state space spanned by the unstable and stable modes, respectively [Mac89, Chapter 8]. Since it would be enough to find these spaces, and the corresponding projection matrices, it would be advisable to do this in an implementation.

Because of this constraint, only the stable modes contribute to the infinite sum in (6.9), which therefore has a finite value:

$$\sum_{i=H_u+1}^{\infty} \|\hat{z}(k+i-1|k)\|_Q^2 = \sum_{i=H_u+1}^{\infty} \|C_z \tilde{W}_s \hat{x}(k+i-1|k)\|_Q^2 \quad (6.28)$$

$$= \hat{x}(k+H_u|k)^T \bar{Q} \hat{x}(k+H_u|k) \quad (6.29)$$

where, proceeding as before, we find that

$$\bar{Q} = \tilde{W}_s^T \Pi \tilde{W}_s \quad (6.30)$$

and Π is the solution of the matrix Lyapunov equation:

$$\Pi = W_s^T C_z^T Q C_z W_s + J_s^T \Pi J_s \quad (6.31)$$

Again Q should be as in (6.19), with \bar{Q} being obtained from (6.30). Predictions are obtained as before.

The significant difference now is the appearance of the equality constraint (6.27). This can be simply added to the QP formulation as an equality constraint. But, as pointed out by Scokaert in the GPC context [Sco97], this equality constraint removes some degrees of freedom from the optimization (as many as there are unstable modes), and it is possible to reparametrize the problem so that the optimization is over a smaller number of decision variables.

In the state-space setting this can be done as follows. From equation (2.66) we see that

$$\hat{\xi}_u(k+H_u|k) = \tilde{W}_u \hat{x}(k+H_u|k) \quad (6.32)$$

$$= \tilde{W}_u \left\{ A^{H_u} \hat{x}(k|k) + \sum_{i=0}^{H_u-1} A^i B u(k-i) + \left[\sum_{i=0}^{H_u-1} A^i B \cdots B \right] \Delta \mathcal{U}(k) \right\} \quad (6.33)$$

This represents n_u equations, if there are n_u unstable modes. Now we can use these equations to express any n_u of the variables in the vector $\Delta \mathcal{U}(k)$ in terms of the remaining ones, and thus eliminate these variables from the optimization. Scokaert [Sco97] considers single-input systems, and expresses the last n_u elements in $\Delta \mathcal{U}(k)$, namely the last n_u elements of the computed input signal, in terms of the other. The same can be done with multi-input systems, but other possibilities may be equally sensible, such as selecting the last n_u elements of the j th input signal $\Delta \hat{u}_j(k+i|k)$ for $i = H_u - n_u, H_u - n_u + 1, \dots, H_u - 1$. But it is not clear whether it matters how these n_u elements are selected, even in the single-input case. The computational saving is small — typically one has only one or two unstable modes — so it may be as well to let the optimizer decide how to ‘use’ the equality constraints.²

² With multivariable unstable models a state-space formulation is almost essential. It is almost impossible to recover the correct number of unstable modes from a transfer function matrix, because of numerical sensitivities.

6.3 Fake algebraic Riccati equations

In [BGW90, Mos95] it is shown that stability can sometimes be guaranteed with finite horizons, even when there is no explicit terminal constraint. The finite horizon predictive control problem is associated with a time-varying Riccati difference equation which is intimately related to the optimal value of the cost function. The Fake Algebraic Riccati Technique replaces this equation with an algebraic (time-invariant) Riccati equation which looks like the one that occurs in infinite-horizon LQ problems. If this equation has properties analogous to those which occur for the (real) algebraic Riccati equation of infinite-horizon control, then stability can be inferred. In order to develop the details we shall need to refer to Mini-Tutorial 7, on linear-quadratic optimal control.

We need to note some differences between the finite-horizon problem treated in the Mini-Tutorial and our standard predictive control problem. First, note that the indices on Q and R ‘run backwards’ in the Mini-Tutorial, in the sense that $x(t)$ is weighted by Q_{N-1} , whereas $x(t+N-1)$ is weighted by Q_0 . Similarly, the state-feedback gain applied at time t is K_{N-1} , while that applied at time $t+N-1$ is K_0 .

Secondly, the LQ cost function weights $\|x(k)\|$ and $\|u(k)\|$, whereas in our standard problem we weight $\|z(k)\|$ and $\|\Delta u(k)\|$. We can put our standard problem into the LQ framework as follows. Introduce the augmented state ξ , and the new matrices \tilde{A} and \tilde{B} :

$$\xi(k) = \begin{bmatrix} x(k) \\ u(k-1) \end{bmatrix} \quad \tilde{A} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \quad \tilde{B} = \begin{bmatrix} B \\ I \end{bmatrix} \quad (6.34)$$

then the two models

$$x(k+1) = Ax(k) + Bu(k) \quad \text{and} \quad \xi(k+1) = \tilde{A}\xi(k) + \tilde{B}\Delta u(k) \quad (6.35)$$

are equivalent (with our usual definition $\Delta u(k) = u(k) - u(k-1)$). Furthermore, $\|z(k)\|_Q = \|\xi(k)\|_{\tilde{Q}}$, where

$$\tilde{Q} = \begin{bmatrix} C_z^T Q C_z & 0 \\ 0 & 0 \end{bmatrix} \quad (6.36)$$

(where $z(k) = C_z x(k)$ — recall (2.3)). So our standard predictive control problem is the same as the LQ problem, if we replace A and B in the plant model by \tilde{A} and \tilde{B} , and the weight Q_{N-j-1} by \tilde{Q}_{N-j-1} in the cost function. (The weight R_{N-j-1} stays unchanged in the cost function, but of course it is now interpreted as the weight which penalizes $\Delta u(k)$.) The optimal control is obtained as $\Delta u(t+N-j) = -\tilde{K}_{j-1}\xi(t+N-j)$, where \tilde{K} is the state-feedback gain matrix obtained from (6.39) when these substitutions have been made.

Thirdly, a single horizon N is used in the LQ cost function, whereas we use the two horizons H_p and H_u . This is no problem so long as $H_u \leq H_p$, because we can always set $R_{N-j-1} = \infty$ for $j \geq H_u$, which will effectively reduce the control horizon to H_u .³

³ Note that the Riccati difference equation (6.38) becomes the Lyapunov equation $P_{j+1} = A^T P_j A + Q_j$ when $R_j = \infty$.

Finite horizon:

We consider the problem of having an initial state $x(t)$ at time t , and finding a control sequence which will minimize the finite-horizon cost function

$$V_N(x(t)) = \sum_{j=0}^{N-1} \{ \|x(t+j)\|_{Q_{N-j-1}}^2 + \|u(t+j)\|_{R_{N-j-1}}^2 \} + \|x(t+N)\|_{P_0}^2 \quad (6.37)$$

where $Q_j \geq 0$, $R_j \geq 0$ and $P_0 \geq 0$. The optimal control is found as follows [BH75, KR72, Mos95, BGW90]: Iterate the *Riccati Difference Equation*

$$P_{j+1} = A^T P_j A - A^T P_j B (B^T P_j B + R_j)^{-1} B^T P_j A + Q_j \quad (6.38)$$

and form the state-feedback gain matrix

$$K_{j-1} = (B^T P_{j-1} B + R_{j-1})^{-1} B^T P_{j-1} A. \quad (6.39)$$

The optimal control sequence is given by

$$u(t+N-j) = -K_{j-1} x(t+N-j) \quad (6.40)$$

and the optimal value of the cost (6.37) obtained in this way is $V_N(x(t))^0 = \|x(t)\|_{P_N}^2$.

Infinite horizon:

Now suppose that the horizon becomes infinite ($N \rightarrow \infty$), we consider the infinite-horizon cost

$$V_\infty(x(t)) = \lim_{N \rightarrow \infty} V_N(x(t)) \quad (6.41)$$

and the weighting matrices are constant: $Q_j = Q$, $R_j = R$. If $R > 0$, the pair (A, B) is stabilizable, and the pair $(A, Q^{1/2})$ is detectable, then $P_j \rightarrow P_\infty \geq 0$ (as $j \rightarrow \infty$), and (6.38) is replaced by the *Algebraic Riccati Equation*:

$$P_\infty = A^T P_\infty A - A^T P_\infty B (B^T P_\infty B + R)^{-1} B^T P_\infty A + Q. \quad (6.42)$$

Hence

$$K_j \rightarrow K_\infty = (B^T P_\infty B + R)^{-1} B^T P_\infty A \quad (6.43)$$

and the optimal control becomes the constant state-feedback law:

$$u(t+j) = -K_\infty x(t+j). \quad (6.44)$$

It can be proved that this feedback law is stabilizing — otherwise the cost $V_\infty(x(t))$ would be infinite — so that all the eigenvalues of the closed-loop state-transition matrix $A - BK_\infty$ lie strictly within the unit circle (if the stated conditions hold). The optimal cost is now $V_\infty(x(t))^0 = \|x(t)\|_{P_\infty}^2$.

Mini-Tutorial 7 Linear quadratic optimal control.

When we apply the receding horizon control strategy, we always apply the first part of the finite-horizon control strategy, which means that we apply the constant state-feedback law

$$\Delta u(t) = -\tilde{K}_{N-1} \xi(t) \quad (6.45)$$

(assuming no constraints for the present discussion). The big question now is, when can this law be guaranteed to be stabilizing? That is, when will all the eigenvalues of $\tilde{A} - \tilde{B}\tilde{K}_{N-1}$ be guaranteed to lie inside the unit circle?

We now revert to using the notation which appears in the LQ problem, namely we use A , Q , K etc., with the understanding that these are replaced by \tilde{A} , \tilde{Q} , \tilde{K} etc. when necessary (see Exercise 6.6.). From the *infinite-horizon* LQ problem, it is apparent (subject to the technical conditions: $Q \geq 0$, (A, B) stabilizable, $(A, Q^{1/2})$ detectable) that so long as the algebraic Riccati equation (6.42) has a positive semi-definite solution ($P \geq 0$), and a constant state feedback gain matrix is obtained from this solution by the formula (6.43), then the resulting feedback law will be stable. In the finite horizon problem we have the Riccati difference equation (6.38) instead of (6.42). Let us suppose that we use constant weighting matrices: $Q_{N-j-1} = Q$ and $R_{N-j-1} = R$, so that (6.38) becomes

$$P_{j+1} = A^T P_j A - A^T P_j B (B^T P_j B + R)^{-1} B^T P_j A + Q. \quad (6.46)$$

Now introduce the matrix

$$Q_j = Q - (P_{j+1} - P_j) \quad (6.47)$$

then, substituting for P_{j+1} in (6.46) gives

$$P_j = A^T P_j A - A^T P_j B (B^T P_j B + R)^{-1} B^T P_j A + Q_j. \quad (6.48)$$

But this is an *algebraic* Riccati equation (because the same matrix P_j appears on both left- and right-hand sides). It has been called the *Fake Algebraic Riccati Equation* (or *FARE*) because it does not arise directly from an infinite-horizon LQ problem. If this equation has a solution $P_j \geq 0$, and if $Q_j \geq 0$, then applying the *constant* state-feedback gain K_j (i.e. apply the same gain K_j at each time) will give a stable feedback law, where K_j is obtained from P_j using (6.43). So, one way of ensuring a stable (unconstrained) predictive control law is to ensure that the fake algebraic Riccati equation

$$P_{N-1} = A^T P_{N-1} A - A^T P_{N-1} B (B^T P_{N-1} B + R)^{-1} B^T P_{N-1} A + Q_{N-1} \quad (6.49)$$

(with A replaced by \tilde{A} etc.) with $Q_{N-1} \geq 0$ has a solution $P_{N-1} \geq 0$. Note that standard algorithms are available for solving the algebraic Riccati equation; in *MATLAB's Control System Toolbox* it can be solved using the function `dare`.

Since we start with $Q \geq 0$, we will have $Q_{N-1} \geq Q \geq 0$ providing that $P_N \leq P_{N-1}$. In [BGW90] the following monotonicity property is proved: if $P_{t+1} \leq P_t$, then $P_{t+j+1} \leq P_{t+j}$ for all $j \geq 0$. This shows that stability can be ensured by introducing a suitable P_0 into the predictive control problem, which is the same as introducing a suitable terminal cost (see (6.37)). In fact, $Q_0 \geq Q$ is sufficient to give stability. In [BGW90] it is shown that it is not enough just to make P_0 ‘very large’, although there are some specific ways of making P_0 ‘large’ which have the desired effect. One particular way of doing this is to set $W_0 = P_0^{-1} = 0$, and propagating $W_j = P_j^{-1}$ instead of P_j .⁴ Note that this is a particular way of making P_0 infinite, and hence is equivalent to imposing a terminal constraint, which provides an alternative explanation of why this choice of P_0 guarantees stability.

⁴ As is done in ‘information filter’ versions of the Kalman filter, and in recursive least-squares.

There is one further technicality involved: the pair $(A, Q_{N-1}^{1/2})$ should be detectable. But it is shown in [BGW90] that, if $(A, Q^{1/2})$ is detectable and $Q_{N-1} \geq Q$, then $(A, Q_{N-1}^{1/2})$ is detectable. So there is no problem. There are several other related and interesting results in [BGW90]. In particular, the Fake Algebraic Riccati Equation (FARE) approach can be interpreted as producing a predictive control law which is the same as the law which would result from posing a particular infinite-horizon problem, which ‘explains’ why the approach is effective. Also the monotonicity property mentioned above is related to a similar monotonicity property of the optimal cost, which implies that the optimal cost can serve as a Lyapunov function, and hence gives another explanation of the efficacy of this approach.

It should be emphasized that the FARE approach gives sufficient but not necessary conditions for stability of predictive control laws. In particular, if the *GPC* formulation of predictive control is used, in which $P(0)$ is fixed as $P(0) = Q = C_y^T C_y$ and $R = \lambda I$, and if the prediction and control horizons are made the same, $H_p = H_u$, then P_j can be shown to be monotonically nondecreasing, rather than nonincreasing [BGW90]. Nevertheless, *GPC* laws can be stabilizing, even with this choice of horizons.

We have examined three ways of ensuring stability: imposing terminal constraints, using infinite horizons, and using the FARE approach. We have seen that they can all be regarded as modifying the weight on the terminal control error in some way. In that sense the three approaches are not so different from each other. But we can note that the first and most obvious way of achieving stability, namely imposing terminal constraints, is in general unnecessarily severe — one does not have to make the terminal weight infinitely large in order to obtain stability. We also mention again the survey paper [MRRS00], in which alternative common elements are identified for these and other ways of ensuring stability while using predictive control.

The authors of [BGW90] conclude by suggesting the use of infinite horizons, so perhaps the FARE method is only of historical interest now. However, recently the method has been generalized to a *Fake Hamilton–Jacobi–Bellman Equation* method for obtaining stability with nonlinear internal models in unconstrained problems [MS97a].

6.4

Using the Youla parametrization

If the plant is stable, then it is known that the feedback system shown in Figure 6.3 is internally stable if and only if the block labelled *Youla parameter* is stable, providing that the internal model is an exact model of the plant [Mac89, ZDG96]. This holds under rather general conditions; all the blocks shown in the figure can be nonlinear operators, for example. Furthermore, for linear systems it is known that *every* stable feedback system can be represented in this way (if the plant is stable) — so nothing is lost by assuming this form. (The term ‘parameter’ here comes from the fact that as the Youla parameter ranges over all stable systems, so it generates all possible stabilizing controllers for the given plant. But it is not just a number!)

Let the Youla parameter shown in Figure 6.3 have transfer function (matrix) $Q(z)$, and let both the plant and the internal model have transfer function $P(z)$. We have

$$u(z) = Q(z)[y(z) - P(z)u(z)] \quad (6.50)$$

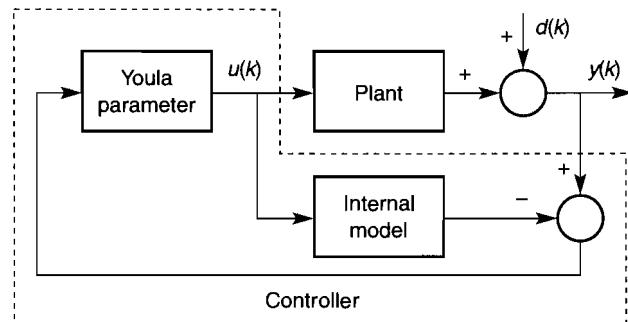


Figure 6.3 Youla parametrization of all stable feedback systems (if the plant is stable).

so that

$$[I + Q(z)P(z)]u(z) = Q(z)y(z) \quad (6.51)$$

and hence

$$u(z) = [I + Q(z)P(z)]^{-1}Q(z)y(z) \quad (6.52)$$

Comparing this with the more conventional representation of a feedback system shown in Figure 7.1, in which we have $u(z) = -K(z)H(z)y(z)$ (if we ignore the reference signal r and the noise n), we see that the following correspondence holds:

$$K(z)H(z) = -[I + Q(z)P(z)]^{-1}Q(z) \quad (6.53)$$

From this, it is very easy to obtain the correspondence in the other direction:

$$Q(z) = -K(z)H(z)[I + P(z)K(z)H(z)]^{-1} \quad (6.54)$$

If the plant is unstable (and linear) then closed-loop stability can still be characterized by the stability of the Youla parameter, but its definition in terms of the controller becomes more complicated [Mac89, ZDG96]. Figure 6.3 has to be replaced by a more complicated block diagram.

Mathematically, the set of all stable Youla parameters is an infinite-dimensional space, so optimizing over it can only be done for particular problems (such as unconstrained LQG or H_∞ problems). But it is also possible to parametrize some subset of all stable systems by a finite number of parameters, which converts the optimization into a finite-dimensional one. Furthermore, if the parametrization is linear and the cost function is quadratic then the problem remains convex. This is due to the remarkable property of the Youla parameter, that all the closed-loop transfer functions which are of interest are *affine* in this parameter — that is, they all have the form $XQY + Z$, where Q is the Youla parameter, and X , Y and Z are some (fixed) transfer function matrices.⁵ There have been some proposals to solve the constrained predictive control problem with this approach [RK93, vdBdV95].

⁵ One consequence of this is that the minimization of any induced norm of a closed-loop transfer function, $\|XQY + Z\|$, is a convex optimization problem. The book [BB91] is devoted to exploiting this.

Example 6.4

All SISO FIR systems of order n are parametrized linearly by the n terms of their pulse responses $H(0), H(1), \dots, H(n - 1)$.

From Figure 6.3 it is apparent that $u(z) = Q(z)d(z)$, where d is the output disturbance. But if $Q(z)$ is an FIR filter of order n then $Q(z) = H(0) + z^{-1}H(1) + \dots + z^{-n}H(n)$. Hence the control signal can be computed as

$$\hat{u}(k + j|k) = \sum_{i=0}^n H(i)\hat{d}(k + j - i|k) \quad (6.55)$$

or, equivalently,

$$\Delta\hat{u}(k + j|k) = \sum_{i=0}^n H(i)[\hat{d}(k + j - i|k) - \hat{d}(k + j - i - 1|k)] \quad (6.56)$$

Note that $\Delta\hat{u}(k + j|k)$ is linear in the pulse response coefficients $H(0), \dots, H(n)$. Furthermore, the disturbance estimates $\hat{d}(k + j - i|k)$ do not depend on future inputs $\hat{u}(k + j|k)$. So substituting (6.56) into (2.66), for instance, makes the state predictions depend linearly on the pulse response coefficients, and hence the cost function (2.19) is quadratic in these coefficients. So the problem remains quadratic.

Note, however, that the constraints on the input levels and the input moves are now related to the ‘decision variables’, namely to the pulse response coefficients, through transfer functions.

Exercises

- 6.1** Simulate the model and control law from Example 6.1 using *MATLAB*, and check that the closed loop is unstable.

Recommended procedure: create the state-space matrices a, b, c, d corresponding to the *closed-loop* system, then create a Matlab object representing this closed-loop system using:

```
clsys=ss(a,b,c,d,1);
```

(Use `help ss` to see what this does.) Then get the response from initial condition $x_0=[1;0]$ — or some other initial condition — by typing

```
initial(clsys,x0);
```

- 6.2** Consider the plant given in Example 6.1 again. But now suppose that the prediction horizon is $H_p = 2$, with the cost being

$$V(k) = \hat{x}(k + 1|k)^T \begin{bmatrix} 1 & 2 \\ 2 & 6 \end{bmatrix} \hat{x}(k + 1|k) + \hat{x}(k + 2|k)^T \begin{bmatrix} 1 & 2 \\ 2 & 6 \end{bmatrix} \hat{x}(k + 2|k)$$

- (a) Keep $H_u = 1$, so that only $u(k)$ is to be optimized, with the assumption that $u(k+1) = u(k)$. Show that the predictive control law is $u^0(k) = -\frac{1}{6}x_1(k)$, and hence that the closed loop is stable.
- (b) Now let $H_u = 2$, so that $u(k)$ and $u(k+1)$ have to be optimized. By setting both derivatives $(\partial V/\partial u(k))$ and $(\partial V/\partial u(k+1))$ to zero (or $\nabla_u V = 0$ if you are happy with that notation) show that the predictive control law is $u^0(k) = -\frac{2}{3}x_1(k)$, and hence that the closed loop is stable.
- (c) Simulate the closed-loop behaviours for these two cases, using *MATLAB*.

(Note that the *Model Predictive Control Toolbox* does not allow non-diagonal weights.)

- 6.3** Consider Example 6.1 again, and Exercise 6.2. Solve and simulate them using the *Model Predictive Control Toolbox*. However, the *Toolbox* only allows diagonal weighting matrices to be specified. If you have solved Exercise 3.7, then you can keep the original non-diagonal weight. Otherwise change the weight to $\text{diag}(1, 6)$ (that is, just keep the diagonal elements). Unfortunately this will not give the same results, but it is still a good exercise to see how the results differ for the cases

- ✿ $H_p = H_u = 1$
- ✿ $H_p = 2, H_u = 1$
- ✿ $H_p = H_u = 2$

- 6.4** Verify that \bar{Q} is given by (6.30) and (6.31) in the case of an unstable plant.

- 6.5** For the plant

$$x(k+1) = 0.9x(k) + 0.5u(k), \quad y(k) = x(k)$$

suppose that the infinite-horizon case ($H_p = \infty$) is to be solved.

- (a) Formulate an equivalent finite-horizon problem if $Q(i) = 1$, $R(i) = 0$, and $H_u = 2$.
- (b) Find the predictive controller and simulate the set-point response.
- (c) What feature of the *Model Predictive Control Toolbox* would prevent you from following this procedure in general?

- 6.6** The infinite-horizon LQ problem requires the assumptions that (A, B) is a stabilizable pair and that $(A, Q^{1/2})$ is a detectable pair. To put the standard predictive control problem into the LQ framework, we need to use the matrices \tilde{A} , \tilde{B} , and \tilde{Q} , as defined in (6.34) and (6.36). Show that, if (\tilde{A}, \tilde{B}) is stabilizable then so is the pair (A, B) , and that if $(\tilde{A}, \tilde{Q}^{1/2})$ is detectable then so is the pair $(A, Q^{1/2}C)$. (Note that you can take $\tilde{Q}^{1/2} = [Q^{1/2}C, 0]$.)

6.7 Consider the finite-horizon LQ problem for a plant with

$$A = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (6.57)$$

Verify that if

$$P_{N-1} = \begin{bmatrix} 16 & 0 \\ 0 & 16 \end{bmatrix} \quad (6.58)$$

and $R = 1$ then the corresponding state-feedback gain matrix is $K_{N-1} = [0, 1.8824]$ and that this gives a stable feedback law. Show, nevertheless, that this solution corresponds to a value of Q_{N-1} which is not positive-definite. (This underlines the fact that the FARE approach gives a sufficient, but not necessary, condition for stability.)

Chapter 7

Tuning

7.1	What are we trying to do?	188
7.2	Some special cases	191
7.3	Frequency-response analysis	199
7.4	Disturbance models and observer dynamics	201
7.5	Reference trajectory and pre-filter	211
	Exercises	214

There are many adjustable parameters in predictive control, even in the standard formulation:

- ❖ Weights
- ❖ Horizons
- ❖ Disturbance model and observer dynamics
- ❖ Reference trajectory

In this chapter we shall examine the effects of these, try to obtain some insight into the effects of these parameters, and some systematic methods of adjusting them.

It will be seen that tuning can be based on a few theorems, but it is mostly a matter of ‘rules of thumb’, based largely on experience gained from simulations of ‘typical’ problems.

More can be said in the case of single-input, single-output (SISO) plant, than in the (more important) case of multivariable plant. In particular, Soeterboek [Soe92] gives more details for the SISO case than we give here.

7.1 What are we trying to do?

It should always be borne in mind that

- ➊ Feedback is dangerous.
- ➋ The *only* purpose of using feedback is to reduce the effects of uncertainty.

Feedback is dangerous because it can, among other things, destabilize systems which are themselves quite stable. The reason for using feedback is that it can be a very effective strategy for reducing the effects of unexpected and unmeasurable disturbances, and for reducing the effects of uncertainty about the behaviour of a system. It is important to realize that this is the only justifiable reason for using feedback. (One important application of feedback is to stabilize unstable plant. But it can be argued that this is also necessary because of some uncertainty: if the model and initial conditions were known perfectly, then it would be possible to stabilize an unstable plant using an open-loop strategy.)

It follows that simulating the response to a step set-point change is not a meaningful way of assessing a feedback system, except under very special circumstances. If we really wanted to follow a set-point, and there were no uncertainty, then the best way to do it would be by using an open-loop pre-computed control signal. Looking at set-point step responses is meaningful only under the following circumstances:

- ➊ If the set-point step response allows us to deduce something about the robustness — insensitivity — of the closed-loop behaviour to modelling errors.
This is typically the case with linear control systems.
- ➋ If the set-point step response can be easily related to the way in which the feedback loop deals with disturbances.
This is true for ‘one degree of freedom’ linear feedback systems, in which the set-point response completely determines the disturbance response. This is typically not true for predictive control systems.
- ➌ If the step response allows us to deduce how the system would respond to other signals.
This is true for linear systems. It is therefore true for most predictive control systems so long as the constraints are inactive, but not otherwise.

Figure 7.1 shows a ‘two degree of freedom’ feedback system. It shows the set-point signal being filtered by the pre-filter $F(z)$ before becoming the ‘reference’ input to the feedback loop. It also shows the part of the controller which is inside the feedback loop split into two transfer functions, $K(z)$ in the ‘forward path’, and $H(z)$ in the feedback path. This is done to have a similar structure to that shown for the unconstrained predictive controller in Figure 3.2. We will assume, for simplicity, that the measured and controlled outputs are the same: $z(k) = y(k)$.

Apart from the set-point signal s , the figure shows an output disturbance d and measurement noise n entering the system. We need to find the transfer functions relating these input signals to the plant input and output signals u and y . We do this for the general case, with each signal being a vector, and so each transfer function being a matrix. We write $\bar{y}(z)$ for the z -transform of $y(k)$, and similarly for other signals.

$$\bar{y}(z) = \bar{d}(z) + P(z)K(z)\bar{e}(z) \quad (7.1)$$

$$\bar{e}(z) = F(z)\bar{s}(z) - H(z)[\bar{n}(z) + \bar{y}(z)] \quad (7.2)$$

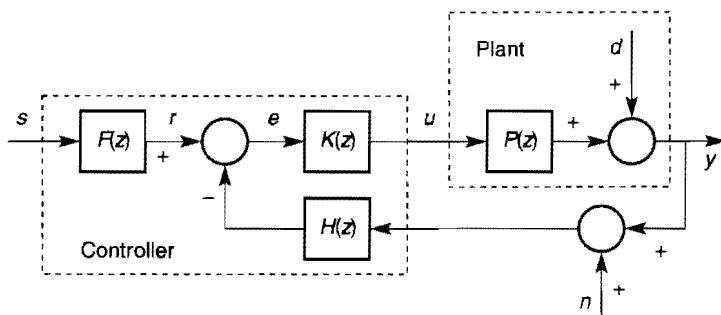


Figure 7.1 Two degree of freedom feedback system.

so

$$\bar{y}(z) = \bar{d}(z) + P(z)K(z)\{F(z)\bar{s}(z) - H(z)[\bar{n}(z) + \bar{y}(z)]\} \quad (7.3)$$

and hence

$$[I + P(z)K(z)H(z)]\bar{y}(z) = \bar{d}(z) + P(z)K(z)F(z)\bar{s}(z) - P(z)K(z)H(z)\bar{n}(z) \quad (7.4)$$

Finally, we have

$$\bar{y}(z) = S(z)\bar{d}(z) + S(z)P(z)K(z)F(z)\bar{s}(z) - T(z)\bar{n}(z) \quad (7.5)$$

where

$$S(z) = [I + P(z)K(z)H(z)]^{-1} \quad (7.6)$$

$$T(z) = [I + P(z)K(z)H(z)]^{-1}P(z)K(z)H(z) \quad (7.7)$$

The last two transfer functions are fundamental in feedback theory:

* $S(z)$ — The *Sensitivity function*.

* $T(z)$ — The *Complementary Sensitivity function*.

Roughly speaking, the ‘smaller’ the sensitivity function, the better the feedback action, in the sense that the effect of the output disturbance d is kept small. It can also be shown that the sensitivity of the closed-loop performance to open-loop changes depends on $S(z)$, and is small if $S(z)$ is ‘small’ in some sense. On the other hand, the ‘smaller’ the complementary sensitivity $T(z)$, the smaller is the effect of measurement noise. It can also be shown that the ‘gain’ of $T(z)$ is a kind of stability margin: if $T(z)$ is ‘large’ in some sense, then a small plant–model mismatch may cause the feedback loop to be unstable. Note that $S(z)$ and $T(z)$ are completely determined by each other, because

$$S(z) + T(z) = I \quad (7.8)$$

It is therefore not possible to have both $S(z)$ and $T(z)$ very ‘small’ (near 0) simultaneously. However, if measured by their frequency responses, it is possible to have both

$S(z)$ and $T(z)$ very large simultaneously (because the frequency responses are complex quantities); so it is certainly possible to have very bad feedback designs, although there are limits to how good they can be.

Note that the response of the output to the set-point can be designed independently of the response to disturbances or noise, by means of the pre-filter $F(z)$. Actually, even without $F(z)$ it can be designed independently if $K(z)$ and $H(z)$ can be designed independently. However, in the predictive controller it is not easy to influence these two independently of each other. But it should be noted that, even without a pre-filter ($F(z) = I$), the response of the output to set-point changes — transfer function $S(z)P(z)K(z)$ — is significantly different from the response to output disturbances — transfer function $S(z)$.

The formulation of the predictive control problem emphasizes tracking the reference signal r , subject to constraints. This is supposed to be a nice, intuitive formulation which can be easily related to real-world requirements. But we can see that this is only partly true. It is easily related to tracking requirements and to real constraints, but not at all easily related to good feedback properties.

Recall that the main reason for using predictive control is its ability to handle constraints. This means that when tuning a predictive controller, we should consider its performance — including its feedback properties — when constraints are active. Unfortunately, we have few theoretical tools to help us do this — or even to formulate appropriate measures of performance under these conditions. We have to rely on simulation to assess performance with active constraints, because we have little else for the time being. In this chapter we shall devote most space to the case when constraints are not active, because then the controller is linear and we can do some analysis. But the reader should be aware that this is a rather distorted emphasis, forced by necessity.

7.2

Some special cases

We can look at some special choices of parameters, and deduce heuristically how the predictive controller will then behave.

7.2.1 Effect of control weighting

Note first that increasing the weights $R(i)$ on the control moves relative to the weights $Q(i)$ on the tracking errors has the effect of reducing the control activity — recall the cost function (2.19). (In some MPC products the elements of $R(i)$ are called *move suppression factors* for this reason.) Increasing these weights indefinitely will reduce the control activity to zero, which ‘switches off’ the feedback action. If the plant is stable, this will result in a stable system, but not otherwise. Thus with a stable plant, we can expect to obtain a stable closed loop by increasing the control weighting sufficiently. The penalty of doing this will be slow response to disturbances, since only small control actions will result. With an unstable plant we can expect an unstable feedback loop, if the $R(i)$ s are increased too much.

We have already seen in Chapter 6 that there are better ways of ensuring closed-loop stability than using heavy penalty weights $R(i)$.

7.2.2 Mean-level control

Suppose that the plant is stable. Choose $H_u = 1$ and $R(i) = 0$. Suppose that the reference trajectory changes to a new constant value: $r(k+i) = r_1$. So the predictive controller tries to minimize

$$\sum_{i=H_w}^{H_p} \|\hat{y}(k+i|k) - r_1\|_{Q(i)}^2 \quad (7.9)$$

using only one control move $\Delta u(k)$. If H_w is fixed and $H_p \rightarrow \infty$, clearly the optimal thing to do is to move the control to that level which will give $y = r_1$ in the steady state. So in the absence of any disturbances the control signal will be a step (vector). The transient response at the plant output will therefore just be the open-loop response of the plant. That is, we will have, assuming a ‘square’ plant:

$$\bar{y}(z) = P(1)^{-1} P(z) \frac{r_1}{1 - z^{-1}} \quad (7.10)$$

where the ‘zero-frequency plant gain’ $P(1)$ has been introduced to adjust the steady-state level correctly. Comparing this with (7.5) we see that

$$S(z)P(z)K(z) = P(1)^{-1}P(z) \quad (7.11)$$

If the plant is SISO then we can deduce that

$$S(z) = \frac{1}{P(1)K(z)} \quad (7.12)$$

which is the transfer function that governs the response to output disturbances. It turns out that making the observer deadbeat (placing all the eigenvalues of $A - LC$ at zero) leads to deadbeat response to a disturbance, in the shortest possible time. A detailed discussion of the ‘mean-level controller’ is available in [CM89].

Note that the discussion here is consistent with Exercise 4.3, which shows that the controller is non-dynamic if the parameters are chosen as for mean-level control, and with full state measurement.

It has been argued that mean-level control is suitable for many applications: it allows the plant to follow set-point changes at its ‘natural’ rate, while being able to react to disturbances very quickly — at a rate determined by the observer design (or the choice of the polynomial $C(z^{-1})$ in the transfer-function formulation).

7.2.3 Deadbeat control

Now suppose that $H_w = H_u = n$, the number of states in the plant, together with any additional states needed for the disturbance model. Again take $R(i) = 0$, assume that $r(k+i) = r_1$, and this time choose $H_p \geq 2n$. So now the cost function being minimized is

$$\sum_{i=n}^{2n} \|\hat{y}(k+i|k) - r_1\|_{Q(i)}^2 \quad (7.13)$$

Now the idea is that the controller has enough time to drive the output to r_1 and leave it there thereafter — in general, at most n steps are needed to do this. Since $H_w = n$, errors do not start entering the cost function until the set-point has been achieved exactly. Consequently this strategy achieves zero cost, and hence it is the optimal one.

The reason for making $H_p \geq 2n$ is to have a long enough costing interval. This ensures that the output remains at zero for at least n steps, which is enough to ensure that there are no ‘delayed modes’ inside the controller which might emerge in the future.

This behaviour can only be explained by a ‘deadbeat’ closed loop. That is, all the closed-loop poles are at zero. So we have, from (7.5):

$$S(z)P(z)K(z) = z^{-n}N(z) \quad (7.14)$$

where $N(z)$ is a *polynomial* matrix in z (just numerator polynomial in the SISO case) such that $N(1) = I$. In the SISO case we can deduce that

$$S(z) = \frac{N(z)}{z^n P(z) K(z)} \quad (7.15)$$

which shows that $S(z)$ (and hence $T(z)$) would contain the zeros of the plant among its poles if they were not cancelled by zeros of $N(z)$. This would lead to instability if $P(z)$ had zeros outside the unit disc, and to very resonant behaviour if $P(z)$ had zeros close to the unit disc. (Plants with zeros outside the unit disc usually exhibit ‘inverse response’, and are often called *non-minimum-phase* plants.) In [CM89] it is shown that the zeros of $N(z)$ in fact include the zeros of $P(z)$, so that the closed-loop system is stable.

If the sampling interval is small then the control signals needed to achieve deadbeat control may be too aggressive to be usable, in which case the expected behaviour will not be obtained because input constraints will be active. But in a sense ‘mean-level’ control and ‘deadbeat’ control are two extremes obtained with $R(i) = 0$. The main ‘parameter’ for moving between them is the combination of control and window horizons, H_u and H_w . (Note that the deadbeat behaviour is not obtained if $H_w < n$.)

7.2.4 ‘Perfect’ control

Now we examine the case which the *Model Predictive Control Toolbox User’s Guide* calls the ‘perfect’ controller. This is obtained by choosing $H_u = H_p = 1$ and $R(i) = 0$. In this case the cost being minimized is $\|\hat{y}(k+1|k) - r(k+1)\|_{Q(i)}^2$. Clearly the ‘optimal’ thing to do in this case is to choose the input signal so that the next output matches the reference as closely as possible, without taking any account of future consequences. The easiest way to see the result of this is to consider the scalar case:

$$\bar{y}(z) = P(z)\bar{u}(z) = \frac{B(z^{-1})}{A(z^{-1})}\bar{u}(z) \quad (7.16)$$

and write this, using the difference equation interpretation, as

$$b_0 u(k) = y(k+1) + \dots + a_n y(k+1-n) - b_1 u(k-1) - \dots - b_n u(k-n) \quad (7.17)$$

Clearly we can choose $u(k)$ such as to make $y(k+1) = r(k+1)$ by setting

$$\begin{aligned} b_0 u(k) &= r(k+1) + a_1 y(k) + \dots \\ &\quad + a_n y(k+1-n) - b_1 u(k-1) - \dots - b_n u(k-n) \end{aligned} \quad (7.18)$$

But if this is done at each step, then we will eventually have — assuming no disturbances — $y(k) = r(k)$, $y(k-1) = r(k-1)$, etc., so that eventually we will have

$$\begin{aligned} b_0 u(k) &= r(k+1) + a_1 r(k) + \dots \\ &\quad + a_n r(k+1-n) - b_1 u(k-1) - \dots - b_n u(k-n) \end{aligned} \quad (7.19)$$

or

$$B(z^{-1})u(k) = A(z^{-1})r(k+1) \quad (7.20)$$

which shows that the controller is essentially the inverse of the plant in this case.

This will lead to an unstable feedback loop if the plant has zeros outside the unit disc, and unacceptable performance if it has zeros close to the unit disc. The zeros of $B(z^{-1})$ will not show up in the poles of $S(z)$ or $T(z)$, because they will be cancelled when forming the product of the controller and the plant transfer functions $P(z)K(z)H(z)$. But there will be ‘internal instability’ due to the cancellation of unstable poles with unstable zeros, which will show up as an unstable transfer function between the disturbance d and the plant input u .

Example 7.1

We apply predictive control, without constraints, to the Paper Machine Headbox described in Example 2.2. Recall that there are two inputs:

- The stock flowrate
- The white water flowrate

and three measured outputs:

- The headbox level
- The feed tank consistency
- The headbox consistency.

But we will control only outputs 1 and 3, which is achieved in the *Model Predictive Control Toolbox* by setting the output weight to $ywt=[1, 0, 1]$ (as in the *Toolbox* demo file `pmlin.m`).

Throughout this example we set $R(i) = 0$ — no penalty on control moves Δu . First a little analysis of the open-loop plant:

Poles: 0.0211, 0.2837, 0.4266, 0.4266

Zeros: -0.4937, -0.2241

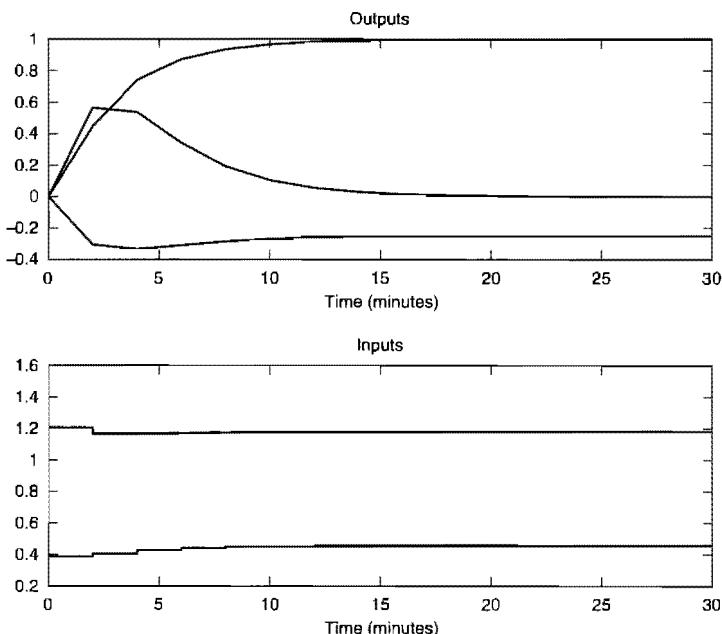


Figure 7.2 Mean-level control: set-point change on r_1 .

So the plant is stable, with real negative zeros. (Using variable names as in the demo file `pmlin.m`, the poles are computed using `eig(PHI)`, while the zeros are computed using

```
tzero(PHI, GAM, C([1, 3], :), D([1, 3], :))
```

where only the first and third rows of C and D are used, since we are only controlling outputs 1 and 3.)

Mean-level control. $H_u = 1$, $H_p = 10$ ($M=1$, $P=10$). Recall that $H_p \rightarrow \infty$ for mean-level control; but the responses do not change significantly for $H_p > 10$.

Responses to step set-point changes are shown in Figures 7.2 and 7.3. These show responses consistent with what one expects from the open-loop poles — the slowest poles, at 0.4266, taking four steps to reduce to below 5% of initial values ($0.4266^4 < 0.05$).

Figures 7.4 and 7.5 show the responses to a unit step unmeasured disturbance acting on the states. Of these the first shows the responses with the default (DMC) observer gain of $[0, I]^T$, while the second shows the response with a deadbeat observer. In the latter case the oscillatory nature of the input signals shows the presence of negative real observer poles, which originate from the negative real zeros of the plant. The disturbance is reduced more effectively, however. In each case the response to set-point changes is exactly the same — the observer design does not affect it.

Observer design in the *Model Predictive Control Toolbox* needs a few tricks, because the *Toolbox* uses the augmented state vector $[\Delta x^T, y^T]^T$, as described in Section 2.4.

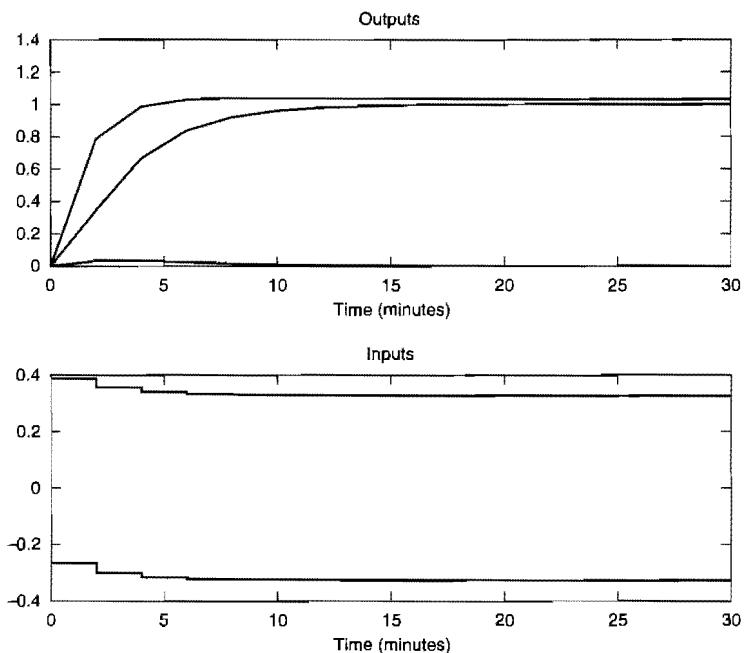


Figure 7.3 Mean-level control: set-point change on r_3 .

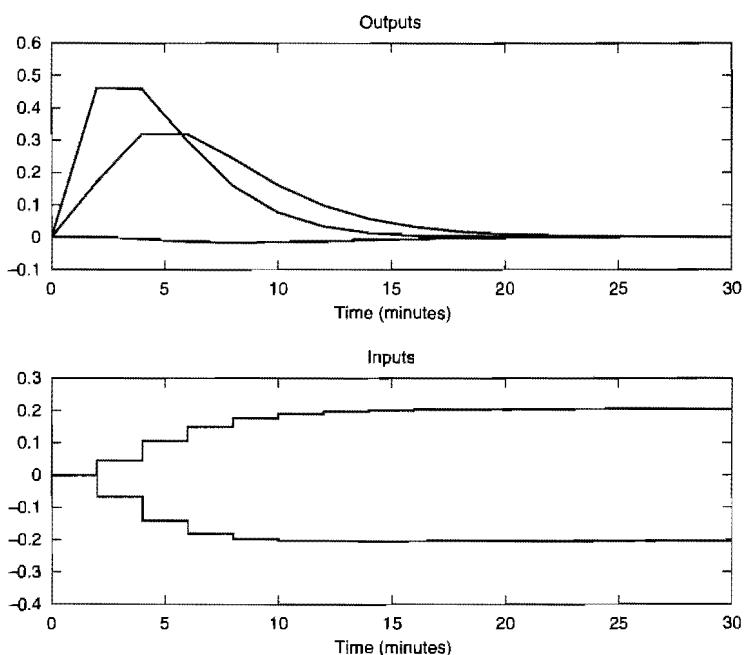


Figure 7.4 Default (DMC) observer: response to step disturbance.

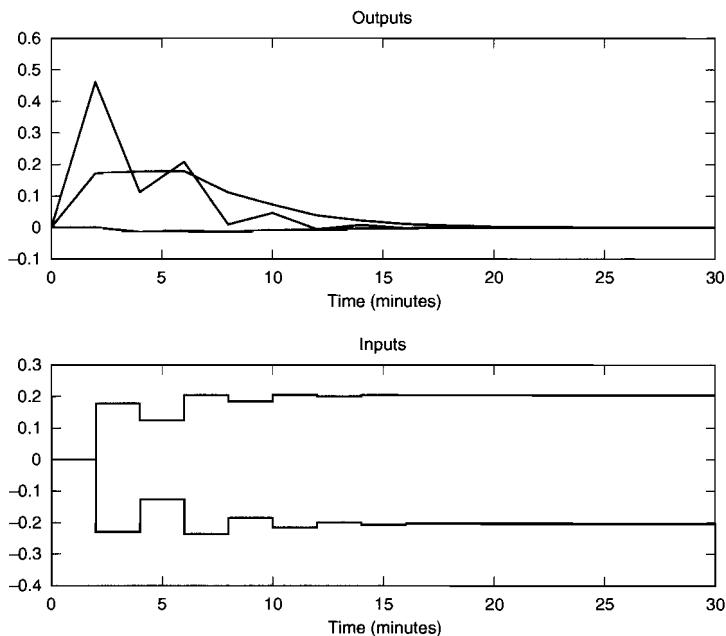


Figure 7.5 Deadbeat observer: response to step disturbance.

This model is obtained from the ordinary one using function `mpcaugss`. The deadbeat observer was designed as follows:

```
[phia,gama,ca] = mpcaugss(PHI,GAM,C); % Augmented model
% Dual pole placement:
Kest = place(phia',ca',[0,0,0,0.01,0.01,0.01,0.005]);
Kest = Kest'; % Transpose because dual problem
```

Here the function `place` is used to place the observer poles at the locations specified in the vector. (The augmented system has seven states.) Ideally these would all be at 0, but `place` allows the multiplicity of each pole to be no greater than the number of outputs, which is three in this case. So 3 poles were placed at 0, 3 at 0.01, and 1 at 0.005. This should result in behaviour rather close to deadbeat.

Deadbeat control. Figure 7.6 shows the set-point response with $H_w = H_u = 4$, $H_p = 8$. Since the plant has four states this should give deadbeat response. It can be seen that this is achieved.

$H_w = 4$ was represented in the *Toolbox* by defining the output weights as follows, which gives zero weight for the first four steps:

```
ywt = [ 0 0 0
        0 0 0
        0 0 0
        0 0 0
        1 0 1 ]
```

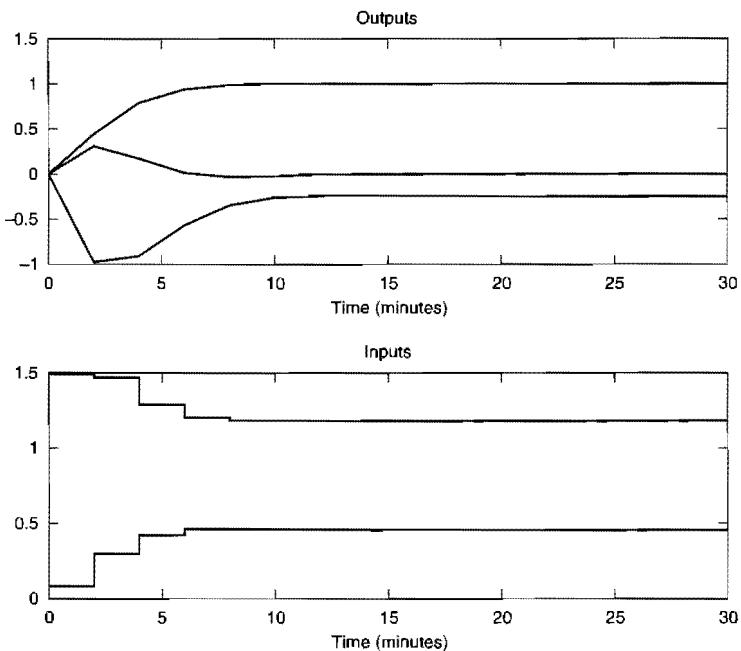


Figure 7.6 Deadbeat set-point response. Step demand on r_1 .

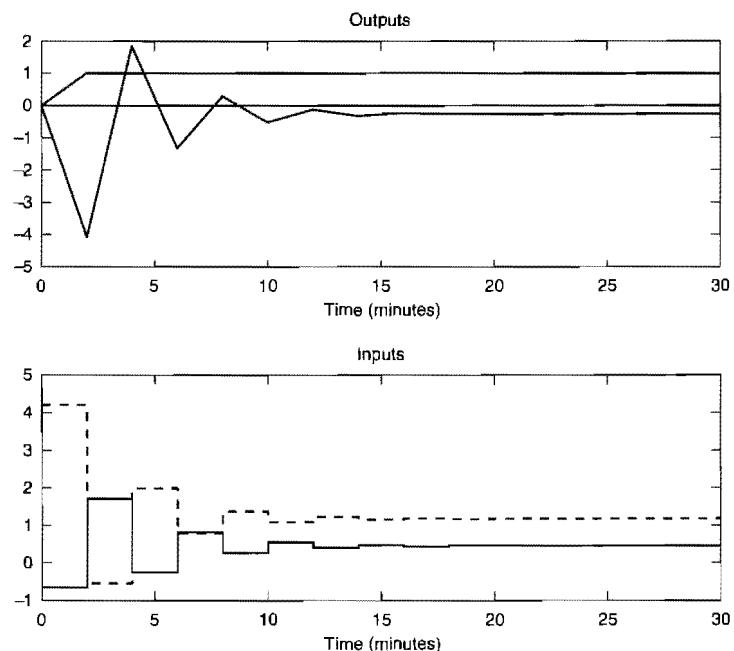


Figure 7.7 'Perfect' controller: set-point response. Step demand on r_1 .

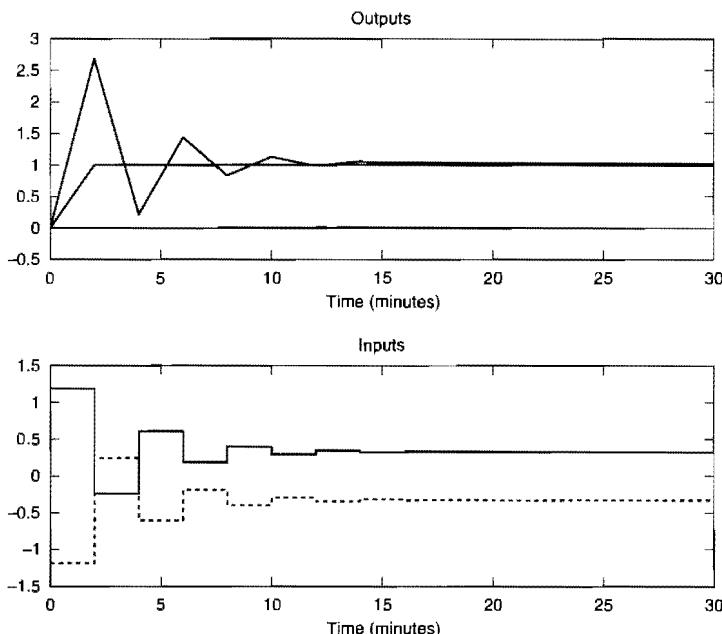


Figure 7.8 ‘Perfect’ controller: set-point response. Step demand on r_3 .

Perfect control. Finally, Figures 7.7 and 7.8 show the set-point step responses with $H_u = H_p = 1$. This gives the so-called ‘perfect’ controller; as expected, the negative real plant zeros now appear as poles of the closed-loop set-point response, which is seen by the oscillatory nature (‘ringing’) of both the control inputs, and one of the plant outputs.

7.3 Frequency-response analysis

Examining time responses gives only part of the story about feedback performance. Frequency response analysis helps to complete the picture — see Mini-Tutorial 8 for an introduction. Frequency responses can be generated directly from a state-space model, without computing a transfer function first. For example, using (4.74) the frequency response of $P(z)$ can be obtained by evaluating

$$P(e^{j\omega T_s}) = C(e^{j\omega T_s} I - A)^{-1} B + D \quad (7.21)$$

for a set of values of ω . Since we can write all the equations governing the closed loop plant-controller combination in state-space form, it is straightforward to compute any frequency response we like, connected with a predictive control system — so long as constraints are not active.¹

¹ Since we showed in Chapter 3 that a predictive controller is linear and time-invariant so long as the set of active constraints remains fixed, it is also possible to perform a frequency response analysis for a given fixed set of active constraints.

SISO feedback design is often performed using open-loop frequency responses, using Nyquist's theorem to ensure stability of the closed loop, and judging stability margins by the 'closeness' of the Nyquist locus to the point -1 . With multivariable systems it is not so useful to examine open-loop frequency loci (although Nyquist's theorem still holds, in a generalized form [Mac89]). But if SISO design rules are interpreted in terms of the closed-loop transfer functions $S(z)$ and $T(z)$, then it is possible to generalize them to the multivariable case quite easily.

With reference to Figure 7.1, if we let $C(z) = K(z)H(z)$ then $S(z) = [I + P(z)C(z)]^{-1}$, or in the SISO case, $S(z) = 1/[1 + P(z)C(z)]$. So in the SISO case, the reciprocal of the smallest distance between the point -1 and the Nyquist locus of $P(z)C(z)$ is the peak value of $|S(e^{j\omega T_s})|$. So the bigger this peak, the smaller the stability margin. This generalizes to the multivariable case as follows.

The 'gain' of a multivariable system depends on the 'direction' of the input signal. If a finite-energy signal is applied at the input of a stable system, the output is also a finite-energy signal. At a given frequency ω , we can define the 'gain' of a multivariable system as the largest amplification (*output energy/input energy*) for any possible input direction. It turns out that this can be measured as the *largest singular value of the frequency response matrix* evaluated at the given frequency. This is a consequence of the fact that the largest singular value is an induced norm for matrices — see Mini-Tutorial 4:

$$\|S(e^{j\omega T_s})\| = \bar{\sigma}[S(e^{j\omega T_s})] \quad (7.22)$$

where $\bar{\sigma}$ is used to denote the largest singular value. So in the multivariable case one can display $\bar{\sigma}[S(e^{j\omega T_s})]$ on a Bode plot, and take the peak value as an indication of the stability margin. (Rigorous interpretations in terms of robustness to plant modelling error can be given [Mac89, ZDG96].)

Another criterion traditionally used for SISO design is the 'peak M -value', and is another measure of how closely the open-loop Nyquist locus approaches the point -1 . But this is nothing other than the peak value of $|T(e^{j\omega T_s})|$. Again this can be generalized to the multivariable case: plot $\bar{\sigma}[T(e^{j\omega T_s})]$ on a Bode plot, and read off the peak value. Again this value can be interpreted rigorously as a robustness measure. Note that large peak values of $\bar{\sigma}[S(e^{j\omega T_s})]$ or $\bar{\sigma}[T(e^{j\omega T_s})]$ can only occur as a result of 'resonant' closed-loop poles lying very close to the unit circle. So not only do they indicate a danger of losing stability, but also the fact that some signals in the loop are likely to exhibit 'ringing' (lightly damped oscillations).

Apart from the peak values, the plots of $\bar{\sigma}[S(e^{j\omega T_s})]$ and $\bar{\sigma}[T(e^{j\omega T_s})]$ can be used to see the frequencies at which the feedback loop rejects disturbances (small $\bar{\sigma}[S(e^{j\omega T_s})]$ is good), and how well it filters out measurement noise (small $\bar{\sigma}[T(e^{j\omega T_s})]$ is good). These two objectives conflict with each other.

Singular values of other closed-loop transfer functions can also be plotted if appropriate. Modern multivariable robust control theory emphasizes the importance of examining the transfer functions $S(z)P(z)$ and $C(z)S(z)$ in addition to $S(z)$ and $T(z)$.

Note that the singular value plots do not indicate whether the closed loop is stable. That has to be checked independently (e.g. by computing the closed-loop poles). If it is stable, they can be used to indicate whether the stability is 'robust' or 'fragile'.

Mini-Tutorial 8 Interpreting frequency responses.

Note: in the *Model Predictive Control Toolbox* it should be easy to compute the frequency responses of the sensitivity and complementary sensitivity functions $S(z)$ and $T(z)$ using functions `mod2frsp` and `smpcc1`, and using `plotfrsp` or `svdfrsp` to see the corresponding Bode plots. However, the closed-loop model formed by `smpcc1` has additional inputs and outputs which must be removed in order to get $T(z)$ — you have to select the first ℓ inputs and the first m outputs, using `mod2ss` to get at the individual state-space matrices, and `ss2mod` to put them together again.

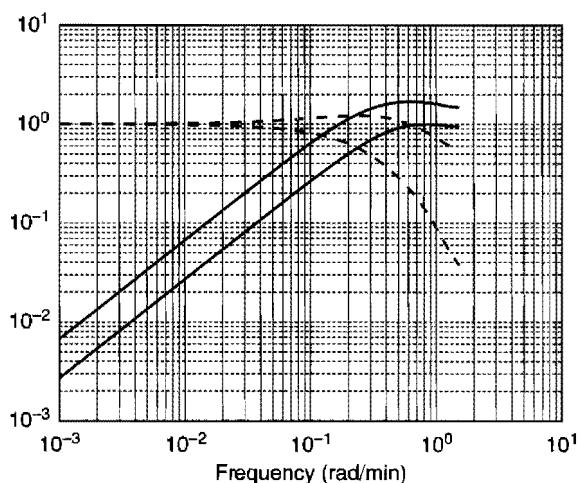


Figure 7.9 Singular values of $S(z)$ (solid line) and $T(z)$ (broken line) for mean-level control of the headbox.

Example 7.2

Figure 7.9 shows the singular values of the sensitivity $S(z)$ and the complementary sensitivity $T(z)$ for the mean-level control of the paper machine headbox, using the default (DMC) observer. This shows that $\bar{\sigma}[T(z)]$ reaches a peak value of almost 2. This is a little higher than is usually considered acceptable for a good feedback design. ($\sqrt{2}$ is often taken as a ‘good’ peak value, but this is application-dependent to some extent.)

Note that there are two singular values of each of S and T at each frequency, because only two outputs have been retained — the headbox level and the headbox consistency — so that $S(z)$ and $T(z)$ are 2×2 transfer function matrices. The plots are shown up to the Nyquist frequency $\pi/T_s = \pi/2 \text{ rad min}^{-1}$ (recall that the sampling interval is $T_s = 2$ minutes for the headbox).

This example shows that the mean-level controller for the paper machine headbox is not very good as a feedback controller. The design parameters can be altered to try to improve its performance in this respect. But at present there is little knowledge of how to change them systematically, so as to improve performance. So we have good analysis tools, but the design is still to a large extent a matter of trial and error. Some progress towards a systematic approach has been reported in [LY94] — see Section 8.2.

7.4 Disturbance models and observer dynamics

7.4.1 Disturbance models

Two very important tuning ‘parameters’ in predictive control are the choice of disturbance model, and the choice of observer dynamics. In the transfer function formulation this corresponds to the choice of the $D(z^{-1})$ and $C(z^{-1})$ polynomials, respectively. In

GPC $D(z^{-1}) = (1 - z^{-1})A(z^{-1})$ is fixed, so only the choice of $C(z^{-1})$ remains. In *DMC* both are fixed.

The essential fact about the disturbance model is that its poles become poles of the (open-loop) controller. So both the *DMC* and the *GPC* disturbance models (poles at $+1$) result in the controller having integral action, and hence offset-free tracking (zero steady-state error) in the presence of constant output disturbances. This is an example of a more general phenomenon, known as the *Internal Model Principle* [Won74].² Suppose that a ‘persistent’ deterministic disturbance acts on a system — that is, one which does not decay to zero. Examples encountered in practice can often be approximated by constant, ramp or sinusoidal signals, but in principle the discussion applies to any signals which have poles on or outside the unit circle. Then a feedback controller can compensate perfectly for that disturbance — asymptotically with time — only if the poles of the disturbance signal appear among the poles of the controller. (The detailed requirements are a little stronger: the controller must have an ‘internal model’ whose state-space structure is the same as that of the disturbance.) When we augment the internal model of a predictive controller by a disturbance model, we are responding to the requirements of this principle.

Example 7.3

Using the *DMC/GPC* disturbance model gives integral action in the controller. Consequently constant disturbances are rejected perfectly (asymptotically with time). Therefore the steady-state gain from output disturbances to controlled outputs should be zero. But this is the zero-frequency response of the transfer function relating d to y in Figure 7.1. So, from (7.5), we expect $S(1) = 0$. (Remember that $\omega = 0$ corresponds to $z = 1$.) Looking at the frequency response plots for Example 7.2, we see that indeed $\tilde{\sigma}[S(e^{j\omega T_s})]$ approaches zero as $\omega \rightarrow 0$. Furthermore, the slope of the graph is 20 dB per decade of frequency, as expected from a single integrator in the controller — there are actually two integrators in this controller, one for each output.

Since $T(z) + S(z) = I$, we also expect $T(1) = I$. This is confirmed by the fact that both singular values of $T(z)$ become 1 (0dB) at low frequencies.

Augmenting the plant model with a disturbance model certainly makes it possible for the controller to exhibit the disturbance poles, but does not seem to be sufficient to ensure that this occurs. After all, the poles of the plant do not usually appear as poles of the controller, even though the plant model is made available to the controller in a similar way.

It is possible to understand how the *DMC* model leads to integral action without a technical derivation. One explanation was already given in Section 1.5; here we give another one. The essential property of a controller with integral action is that it is impossible for the closed-loop system to be at an equilibrium unless the error e , as shown in Figure 7.10, is zero. (The figure shows a ‘PI’ controller for illustration.) Because if it is non-zero, then the output of the integrator will change, and so the plant input u will not remain constant. Now consider a predictive controller with the *DMC*

² The Internal Model Principle, due to Wonham, should not be confused with *Internal Model Control*, a way of looking at feedback systems due to Morari [GM82, MZ88], and originally intended as a way of understanding predictive control.

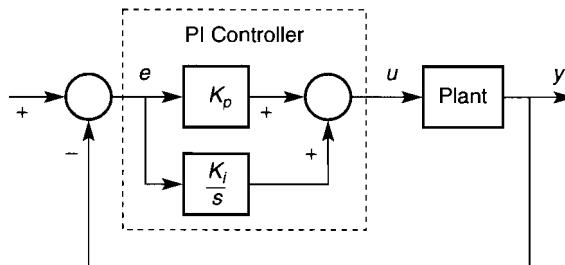


Figure 7.10 A ‘proportional + integral’ (PI) controller.

disturbance model. Suppose, for simplicity, that all set-points are constant. At each step the controller estimates the magnitude of an output disturbance, and produces a control signal to counteract it. Typically the estimate will be incorrect, so that the plant outputs will move to different values than those predicted. The predictive controller attributes this difference to a changed disturbance ($\hat{d}(k|k) = y(k) - \hat{y}(k|k-1)$), and revises its estimate, thus producing a new control signal. If the disturbance is actually constant — and the closed-loop is stable — then the controller iteratively ‘learns’ the magnitude of the disturbance in this way, and eventually compensates for it exactly. It is therefore impossible for the closed-loop system to be at an equilibrium unless the outputs are at their set-points, which is precisely the key property of integral action. (This discussion assumes that no constraints are active. Similarly for the PI controller, the discussion assumes that the integrator is not saturated.)

In general the ‘internal model principle’ works for persistent disturbances, such as ramps and sinusoids, because such disturbances can be predicted perfectly, and hence corrected for, once the initial conditions which specify a particular ramp or sinusoid have been estimated. The principle demands that the dynamic equation governing such a disturbance should be included in the controller, and a (stable) observer should then be used to estimate the initial conditions. The disturbance dynamics in the augmented model are not controllable from the plant input u . Recall that a typical augmented model is of the form:

$$\begin{bmatrix} x_p(k+1) \\ x_d(k+1) \end{bmatrix} = \begin{bmatrix} A_p & X \\ 0 & A_d \end{bmatrix} \begin{bmatrix} x_p(k) \\ x_d(k) \end{bmatrix} + \begin{bmatrix} B_p \\ 0 \end{bmatrix} u(k) \quad (7.23)$$

These dynamics therefore remain unchanged as eigenvalues of the closed-loop, because in effect there is no feedback loop around them. They do not, however, appear in any of the closed-loop transfer functions, since only modes which are both controllable and observable appear as poles of transfer functions.

7.4.2 Observer dynamics

The remaining question to be considered in this section is that of the observer dynamics. In this case we can give a detailed derivation of the transfer function $H(z)$ which appears in Figure 7.1, which will clarify the significance of this question. Figure 7.11 shows the

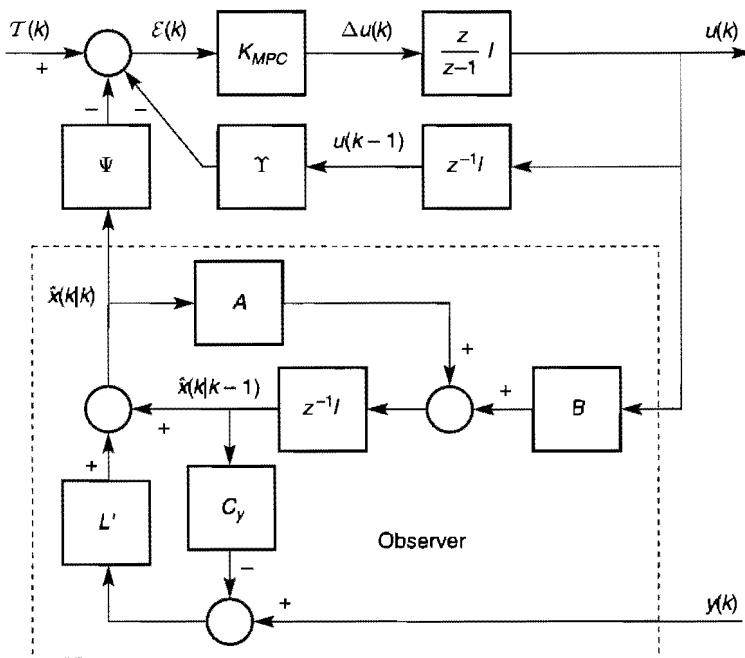


Figure 7.11 Predictive controller, unconstrained case.

controller in block-diagram form, for the unconstrained case. This diagram is obtained from Figure 3.2 by including the details of the observer.

This block-diagram can be simplified slightly to the one shown in Figure 7.12, and then again to the one shown in Figure 7.13.

Comparing Figures 7.1 and 7.13 we can see that

$$H(z) = \Psi \left[I - z^{-1}A(I - L'C_y) \right]^{-1} L' \quad (7.24)$$

$$= z\Psi \left[zI - (A - LC_y) \right]^{-1} L' \quad (7.25)$$

This confirms that the measured outputs are filtered before affecting anything in the controller, and that the poles of the filter are the observer eigenvalues. This filtering will clearly affect the response of the controller to disturbances — since it detects them only through the measured outputs — and the stability of the closed loop.

One could go on to obtain an expression for the forward path transfer function $K(z)$, but that would give a rather complicated expression which would not be very illuminating.

Note that if constraints are active, the measured outputs $y(k)$ are still filtered by a filter with the same poles, in order to form the state estimate $\hat{x}(k|k)$. The nonlinear effects arise afterwards, in the optimization.

So how should one choose the observer dynamics? Choosing deadbeat dynamics gives a fast response to disturbances. But if the magnitudes of typical disturbances

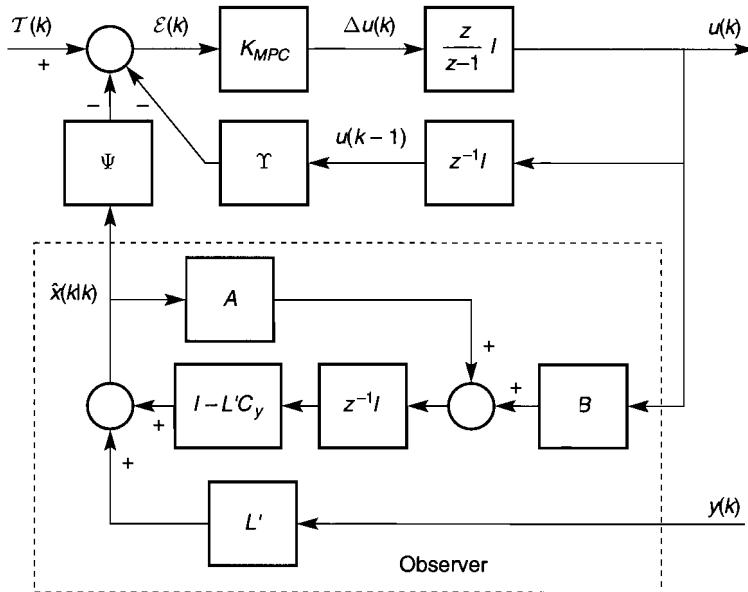


Figure 7.12 Predictive controller, simplified.

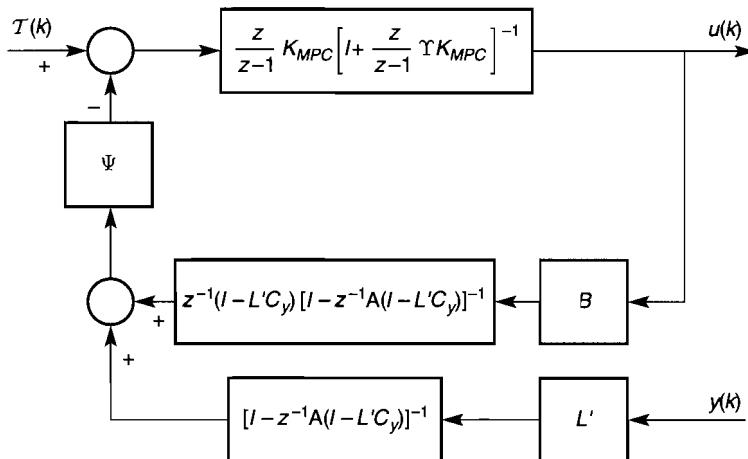


Figure 7.13 Predictive controller, simplified again.

are such that this leads to frequent saturation of actuators, then it may be better to accept a slower response to disturbances, reserving actuator saturation for dealing with exceptional large disturbances. Also, if the observer dynamics are made fast, then the system will react relatively strongly to high-frequency variations in the measured output signal. If there is an appreciable amount of measurement noise in this signal, it may be better to have slower observer dynamics, in order to have some low-pass filtering

of the noise. (This is the usual feedback design trade-off between small $S(z)$ and small $T(z)$.) If statistical information is available about the relative strengths of disturbances and noise, then the optimal trade-off for state estimation can be made by using Kalman filter theory to design the observer. If such information is not available, Clarke [Cla94] advocates using the observer polynomial $C(z^{-1}) = A(z^{-1})(1 - \beta z^{-1})^{H_w}$, noting that $\beta = 0.8$ seems to be always satisfactory. (This assumes the *GPC* disturbance model, so that $D(z^{-1}) = (1 - z^{-1})A(z^{-1})$.)

Example 7.4

Swimming pool

Consider the swimming pool of Exercise 3.3. That problem showed that a predictive controller with the standard ‘DMC’ disturbance model does not compensate perfectly for a sinusoidal diurnal variation of the air temperature if the air temperature is not measured. With the particular choice of parameters used there and a 10 °C amplitude of the air temperature, the water temperature oscillates with an amplitude of about 0.5 °C if there are no constraints on the heater power. Exercise 5.1 then showed that if the air temperature *is* measured and used for feedforward control, then the air temperature is perfectly compensated, *providing* that the model is perfect. If there is some modelling error then a residual oscillation of the water temperature remains, although it typically has smaller amplitude than when the air temperature is not measured.

Such a small oscillation would be acceptable in a swimming pool, but it may be important to remove it completely in some industrial process. This can be done by modelling the sinusoidal disturbance and designing a suitable observer, even if the air temperature is not measured. Furthermore it can be done *robustly* — the oscillation can be removed completely, even if the model is not perfect. This can be done as follows.

First, note that it is reasonable to assume that we know that the air temperature variation is sinusoidal, and its period, from our knowledge of the physical situation. The amplitude and phase of the variation, however, are not known in advance. Thus we can model the air temperature by the differential equation

$$\ddot{\theta}_a + \left(\frac{2\pi}{24}\right)^2 \theta_a = 0 \quad (7.26)$$

with unknown initial conditions on θ and $\dot{\theta}$, which will determine the amplitude and phase. Choosing state variables θ_a and $\dot{\theta}_a$ gives the state-space model

$$\frac{d}{dt} \begin{bmatrix} \theta_a \\ \dot{\theta}_a \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\left(\frac{2\pi}{24}\right)^2 & 0 \end{bmatrix} \begin{bmatrix} \theta_a \\ \dot{\theta}_a \end{bmatrix} \quad (7.27)$$

The complete state-space model is thus:

$$\frac{d}{dt} \begin{bmatrix} \theta \\ \theta_a \\ \dot{\theta}_a \end{bmatrix} = \begin{bmatrix} -1/T & 1/T & 0 \\ 0 & 0 & 1 \\ 0 & -\left(\frac{2\pi}{24}\right)^2 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ \theta_a \\ \dot{\theta}_a \end{bmatrix} + \begin{bmatrix} k/T \\ 0 \\ 0 \end{bmatrix} q \quad (7.28)$$

The corresponding discrete-time model for the nominal parameters $T = 1$ hour and $k = 0.2^\circ\text{C}/\text{kW}$, when the sampling interval is $T_s = 0.25$ hour, is

$$\begin{bmatrix} \theta \\ \theta_a \\ \dot{\theta}_a \end{bmatrix} (k+1) = \begin{bmatrix} 0.7788 & 0.2210 & 0.0288 \\ 0 & 0.9979 & 0.2498 \\ 0 & -0.0171 & 0.9979 \end{bmatrix} \begin{bmatrix} \theta \\ \theta_a \\ \dot{\theta}_a \end{bmatrix} (k) + \begin{bmatrix} 0.0442 \\ 0 \\ 0 \end{bmatrix} q(k) \quad (7.29)$$

which has two eigenvalues on the unit circle, as expected.

If this plant model is used for predictive control, together with the standard ‘DMC’ estimator, then no improvement of performance is obtained — a residual oscillation of the water temperature remains, as before. The reason is that the ‘DMC’ estimator yields an observer whose poles are the plant’s poles and a pole at 0 — see the analysis in Section 2.6.3. But the plant poles include the two disturbance poles on the unit circle, so the observer is not asymptotically stable. The result is that any initial error in estimating the initial conditions $\theta(0)$, $\dot{\theta}(0)$ results in an oscillating, but not reducing, error as measurements of θ become available. The remedy for this is to use an alternative observer gain, which will make the observer asymptotically stable.

In Example 7.1 we used ‘pole placement’ to obtain the observer gain. Here we will pretend that the model of the air temperature disturbance is subject to random disturbances, and that measurements of the water temperature are subject to random noise. We will then use Kalman filter theory to obtain the observer gain. When the state vector has only three elements, as here, both methods will work; but when the state dimension becomes larger than about 5, only the Kalman filter method remains useful in practice. In order to do this, the model (7.29) has to be modified to admit another input w , which is the random disturbance on $\dot{\theta}_a$:

$$\begin{bmatrix} \theta \\ \theta_a \\ \dot{\theta}_a \end{bmatrix} (k+1) = \begin{bmatrix} 0.7788 & 0.2210 & 0.0288 \\ 0 & 0.9979 & 0.2498 \\ 0 & -0.0171 & 0.9979 \end{bmatrix} \begin{bmatrix} \theta \\ \theta_a \\ \dot{\theta}_a \end{bmatrix} (k) + \begin{bmatrix} 0.0442 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} q(k) \\ w(k) \end{bmatrix} \quad (7.30)$$

and the output equation has to be modified to admit measurement noise v :

$$\theta(k) = [1 \ 0 \ 0] \begin{bmatrix} \theta \\ \theta_a \\ \dot{\theta}_a \end{bmatrix} (k) + v(k) \quad (7.31)$$

Now making various assumptions about the variances W and V of w and v , respectively, gives various observer gains, and hence various observer pole locations. Note that this modification of the model is needed only for computing the observer gain; it is not needed for the internal model in the controller, which should be (7.29).

Since the *Model Predictive Control Toolbox* uses the augmented state vector (2.37) described in Section 2.4, there are four state variables when working through this example with the *Model Predictive Control Toolbox*, and hence four observer poles.

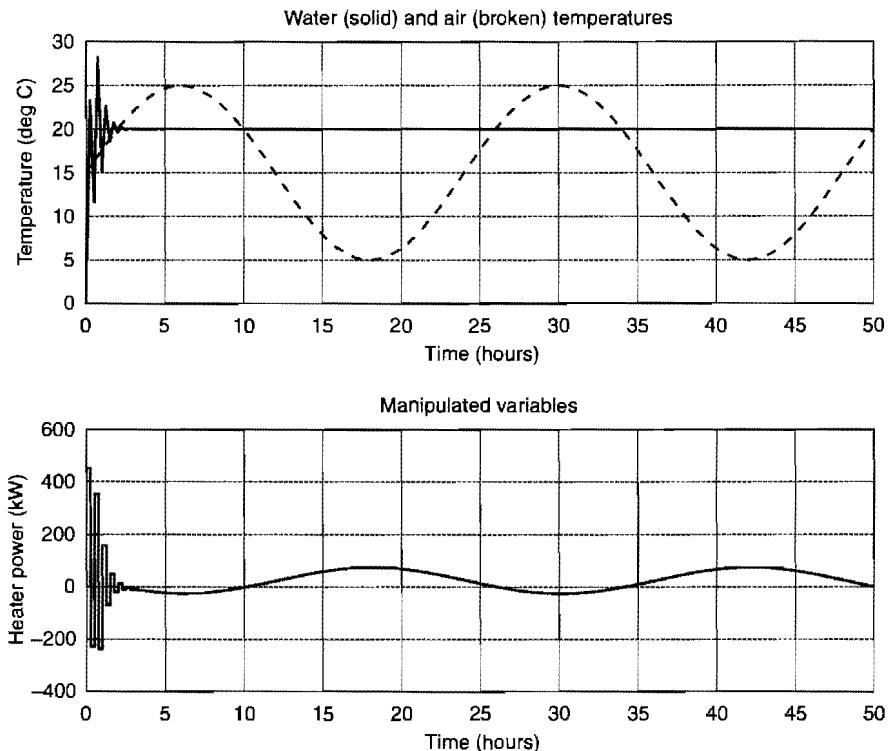


Figure 7.14 Control of water temperature with perfect plant model, and model of air temperature disturbance. $W = 1$, $V = 10^{-6}$. The broken line shows the air temperature.

The choice $W = 1$, $V = 10^{-6}$ gives observer poles (eigenvalues of $A(I - L'C)$ in (7.24)) at -0.5402 , -0.1783 , and $0.0562 \pm 0.0618i$. These are all within the unit circle, so the observer is asymptotically stable, as expected. Since V is very small relative to W , the measurements are considered to be very accurate, and hence the estimation of $\theta_a(0)$ and $\dot{\theta}_a(0)$ by the observer is very fast, as shown by the two poles very close to the origin. (Deadbeat estimation — poles at zero — would be expected with $V = 0$).³ This leads to relatively aggressive control action. Figure 7.14 shows the resulting performance of the predictive controller. This figure was produced using the *Model Predictive Control Toolbox* function `smpc`, which sets the initial states of both the plant and the internal model to zero at the beginning of the simulation. There is therefore an initial transient, lasting about 10 samples, or 2.5 hours, during which the air temperature disturbance is estimated and the water temperature is brought up to its set-point of 20°C . Once this transient has been completed, the set-point is held very accurately, with perfect compensation for the sinusoidal air temperature variation. Note that, in addition to compensating for the sinusoidal variation, which is due to the inclusion of the model

³ The observer gain was computed using the *Model Predictive Control Toolbox* function `smpcest`. This function requires V to be positive definite.

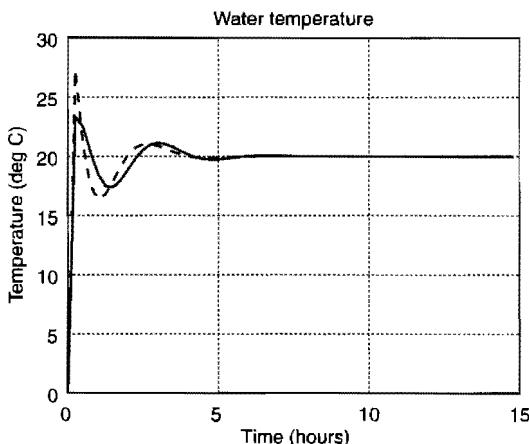


Figure 7.15 Control of water temperature with model of air temperature disturbance. Perfect (solid line) and imperfect (broken line) models. $W = 1$, $V = 1$.

(7.27), the controller also compensates for the mean air temperature (15°C), even though it does not know the value of k/T accurately; this is due to the use of the augmented state, which results in the DMC constant-disturbance model being included implicitly.

The results shown in Figure 7.14 were obtained with the controller having a perfect model of the plant — that is, correct estimates of the parameters T and k (1 hour and $0.2^{\circ}\text{C}/\text{kW}$, respectively). If the same controller is applied when the parameters change to $T = 1.25$ hour and $k = 0.3^{\circ}\text{C}/\text{kW}$ then the closed loop is unstable. If the observer is redesigned on the assumption that $W = 1$ and $V = 1$ then the observer poles become $0.7397 \pm 0.3578i$ and $0.6118 \pm 0.1136i$. Since the measurements are now assumed to be much more noisy, the observer estimates the disturbance more slowly, but remains asymptotically stable. The control is now less aggressive, as is shown by the solid line in Figure 7.15. The initial transient lasts longer than before: about 20 samples, or 5 hours. The benefit of this ‘de-tuning’ is that the control is now much more robust. If the same controller is applied to the pool when the parameters change to $T = 1.25$ hour and $k = 0.3^{\circ}\text{C}/\text{kW}$ then the closed loop remains stable, and the response is almost unchanged — this is shown by the broken line in Figure 7.15. The most important thing to notice here is that the compensation of the sinusoidal air temperature variation remains perfect, although the controller now has the incorrect values of the parameters T and k . This illustrates that modelling the disturbance correctly gives robust disturbance compensation, even when the disturbance is not measured.

The responses shown in Figures 7.14 and 7.15 were obtained with no constraints on the heater power. In particular, the heater power was allowed to be negative at those times when the air temperature exceeded the set-point temperature. If the heater power is limited to

$$0 \leq q \leq 60 \quad \text{kW}$$

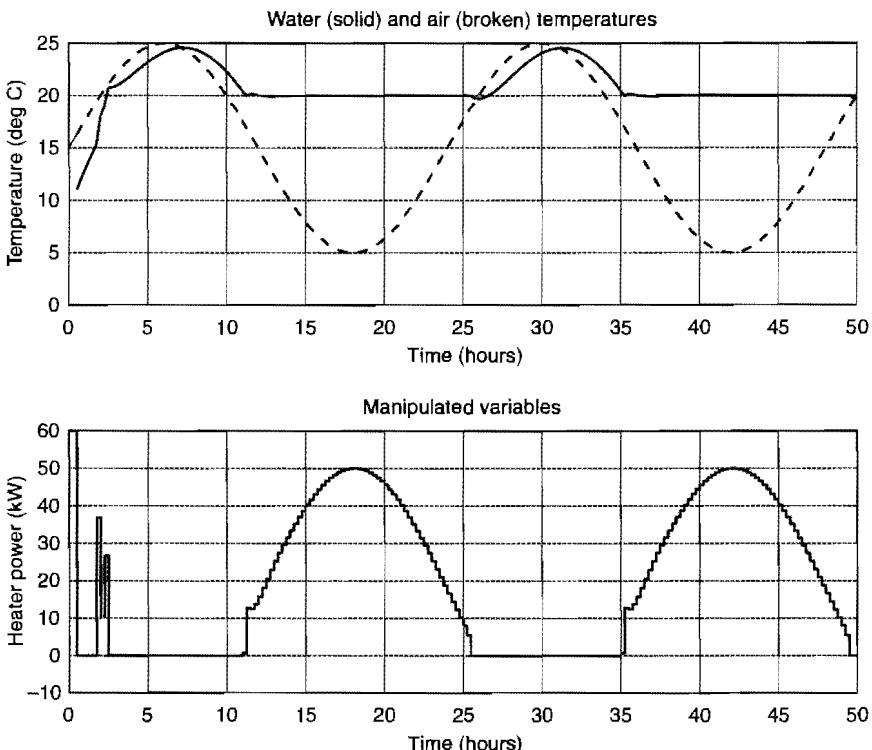


Figure 7.16 Control of water temperature with model of air temperature disturbance, heater power constraints, and imperfect model. $W = 1$, $V = 1$.

then the responses shown in Figure 7.16 are obtained when the same controller as used in Figure 7.15 is used, with an imperfect model. The controller now cannot correct for the air temperature rising above the set-point, but the notable feature is that it still compensates perfectly at other times. It is able to do this only because it is aware of the actual power that the heater is providing, so that its estimate of the air temperature remains accurate, even when it cannot provide effective control.

The model of an unmeasured persistent disturbance does not have to be perfect; even a mild improvement on the ‘constant disturbance’ assumption can lead to a big improvement in performance. Exercise 7.6 demonstrates that a big improvement can be obtained by assuming that the air temperature varies as a ramp, even if it actually varies sinusoidally. The explanation of this is that the observer is constantly re-estimating the disturbance as a ramp, and this allows the controller to predict the sinusoidal disturbance quite well over a short interval.

All the responses shown for this example were computed and displayed using the *MATLAB* function *swimpool*, which is available on this book’s web site. This function enables the reader to experiment with the disturbance model, the observer design, the plant/model error, and the heater constraints.

7.5 Reference trajectory and pre-filter

Another tuning parameter in predictive control is the choice of reference trajectories for each controlled output. Reference trajectories, in the sense introduced by Richalet [Ric93b], were already discussed in Sections 1.2 and 5.6, where the effects of various reference trajectories were presented. As Soeterboek points out [Soe92], initializing the reference trajectory at the latest output measurement introduces another feedback loop. Although one expects, intuitively, that a slower reference trajectory should lead to less aggressive feedback action, and hence greater robustness to modelling errors, this need not always be so. However, only a few particular cases seem to have been analyzed; there is scope for more complete analysis of the effects of reference trajectory specifications.

To see how the use of a reference trajectory modifies the feedback loop, we can proceed as follows.⁴ Assume first that future set-point changes are not known. A set of exponential reference trajectories, one for each controlled output, can be represented by

$$r(k + i_p | k) = (I - \Lambda^{i_p})s(k) + \Lambda^{i_p}z(k) \quad (7.32)$$

where $s(k)$ is the current set-point and $z(k)$ the current value of the controlled output, $\Lambda = \text{diag}(e^{-T_s/T_{ref,1}}, e^{-T_s/T_{ref,2}}, \dots, e^{-T_s/T_{ref,n}})$, and $T_{ref,j}$ is the time constant applicable to the j th output z_j . $T(k)$ in equation (3.2) becomes

$$T(k) = \begin{bmatrix} I - \Lambda^{H_w} \\ I - \Lambda^{H_w+1} \\ \vdots \\ I - \Lambda^{H_p} \end{bmatrix} s(k) + \begin{bmatrix} \Lambda^{H_w} \\ \Lambda^{H_w+1} \\ \vdots \\ \Lambda^{H_p} \end{bmatrix} z(k) \quad (7.33)$$

This modification of $T(k)$ is shown in block-diagram form in Figure 7.17 for the (oversimplified) case when the controlled output $z(k)$, the measured output $y(k)$, and the state $x(k)$, are all the same. This figure clearly exhibits the additional feedback loop, compared with Figure 3.1.

If the controller is aware of future set-point changes, then this information can be exploited in the definition of the reference trajectory. The easiest way to generate exponential reference trajectories in this case is with an iterative loop, in which y_{traj} is a ‘dummy’ variable:

```

let  $y_{traj} = y(k)$ 
for  $i = 1 : p$ 
   $y_{traj} = s(k + i) - \Lambda \{s(k + i) - y_{traj}\}$ 
   $r(k + i | k) = y_{traj}$ 
end
```

⁴ The development in this section, and the figures, are due to my student Simon Redhead.

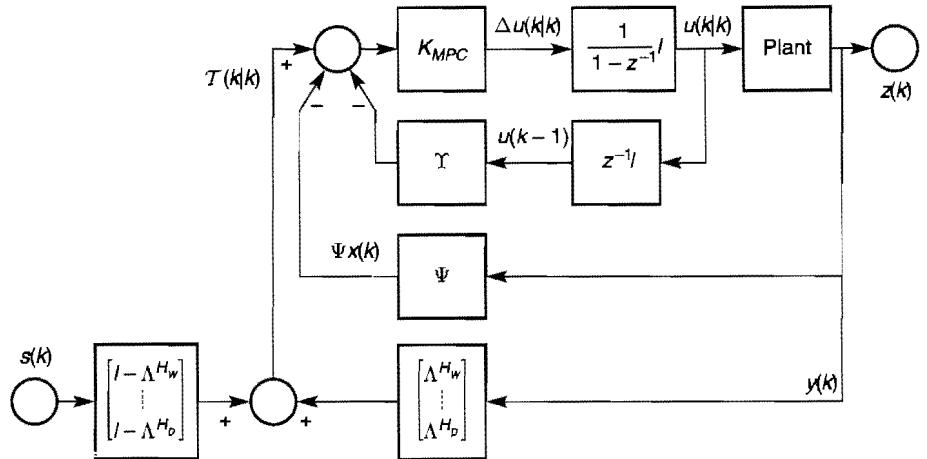


Figure 7.17 Controller structure with reference trajectories: non-anticipated set-point.

which is equivalent to

$$T(k) = \begin{bmatrix} (I - \Lambda)s(k + H_w) \\ (I - \Lambda)s(k + H_w + 1) \\ \vdots \\ (I - \Lambda)s(k + H_p) \end{bmatrix} + \begin{bmatrix} \Lambda r(k + H_w - 1|k) \\ \Lambda r(k + H_w|k) \\ \vdots \\ \Lambda r(k + H_p - 1|k) \end{bmatrix} \quad (7.34)$$

If $H_w = 1$ then

$$T(k) = \begin{bmatrix} r(k + 1|k) \\ r(k + 2|k) \\ \vdots \\ r(k + H_p|k) \end{bmatrix} = \begin{bmatrix} (I - \Lambda)s(k + 1) \\ (I - \Lambda)s(k + 2) \\ \vdots \\ (I - \Lambda)s(k + H_p) \end{bmatrix} + \begin{bmatrix} \Lambda r(k|k) \\ \Lambda r(k + 1|k) \\ \vdots \\ \Lambda r(k + H_p - 1|k) \end{bmatrix} \quad (7.35)$$

where $r(k|k)$ is initialized to the current controlled output: $r(k|k) = z(k)$. Using z to also denote the forward shift operator, equation (7.35) can then be rewritten as:

$$T(k) = \begin{bmatrix} (I - \Lambda)z \\ (I - \Lambda)z^2 \\ \vdots \\ (I - \Lambda)z^{H_p} \end{bmatrix} s(k) + \begin{bmatrix} \Lambda \\ 0 \\ \vdots \\ 0 \end{bmatrix} z(k) + \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ \Lambda & 0 & \dots & 0 & 0 \\ 0 & \Lambda & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & 0 & \dots & \Lambda & 0 \end{bmatrix} T(k) \quad (7.36)$$

or, equivalently,

$$\mathcal{T}(k) = \begin{bmatrix} I & 0 & \dots & 0 & 0 \\ -\Lambda & I & & 0 & 0 \\ 0 & -\Lambda & \ddots & & \vdots \\ \vdots & & \ddots & I & 0 \\ 0 & 0 & \dots & -\Lambda & I \end{bmatrix}^{-1} \left\{ \begin{bmatrix} (I - \Lambda)z \\ (I - \Lambda)z^2 \\ \vdots \\ (I - \Lambda)z^{H_p} \end{bmatrix} s(k) + \begin{bmatrix} \Lambda \\ 0 \\ \vdots \\ 0 \end{bmatrix} z(k) \right\} \quad (7.37)$$

It transpires that this is equivalent to

$$\mathcal{T}(k) = \begin{bmatrix} I & 0 & 0 & \dots & 0 & 0 \\ \Lambda & I & 0 & \dots & 0 & 0 \\ \Lambda^2 & \Lambda & I & & 0 & 0 \\ \Lambda^3 & \Lambda^2 & \Lambda & \ddots & & \vdots \\ \vdots & \vdots & \vdots & \ddots & I & 0 \\ \Lambda^{H_p-1} & \Lambda^{H_p-2} & \Lambda^{H_p-3} & \dots & \Lambda & I \end{bmatrix} \left\{ \begin{bmatrix} (I - \Lambda)z \\ (I - \Lambda)z^2 \\ \vdots \\ (I - \Lambda)z^{H_p} \end{bmatrix} s(k) + \begin{bmatrix} \Lambda \\ 0 \\ \vdots \\ 0 \end{bmatrix} z(k) \right\} \quad (7.38)$$

Figure 7.18 shows this in block diagram form. This is just a particular filter with inputs $z(k)$ and $s(k + i|k)$ ($i = 1, \dots, H_p$), and output $\mathcal{T}(k)$. More complicated reference trajectories can be represented by replacing it by other filters. Note that in Figure 7.18 $\Xi(z)$ represents the vector of transfer functions

$$\Xi(z) = \begin{bmatrix} (I - \Lambda)z \\ (I - \Lambda)z^2 \\ \vdots \\ (I - \Lambda)z^{H_p} \end{bmatrix}$$

The *Model Predictive Control Toolbox* function `scmpc` has been modified to allow reference trajectories to be defined, both without and with anticipation of the set-point. The modified functions, called `scmpc3` and `scmpc4`, respectively, are available on this book's web site.

Reference trajectories are often confused with set-point pre-filtering. A pre-filter is located outside the feedback loop, and hence does not affect the stability or robustness properties of the loop, and has no effect on the response to disturbances. The block with transfer function $F(z)$ in Figure 7.1 shows the location of a set-point pre-filter. The primary effect of the pre-filter is that it defines the ideal plant response to set-point changes. It can be thought of as a 'reference model' — if the plant controlled output followed it exactly, the 'tracking error' vector in the predictive control problem formulation would be zero. A similar effect can be achieved by varying the weights $Q(i)$ over the prediction horizon. But it is much easier to design an appropriate filter than to choose the corresponding pattern of weights — and, as shown in [HM97], it is also computationally more efficient to use a filter. It should be said, however, that if

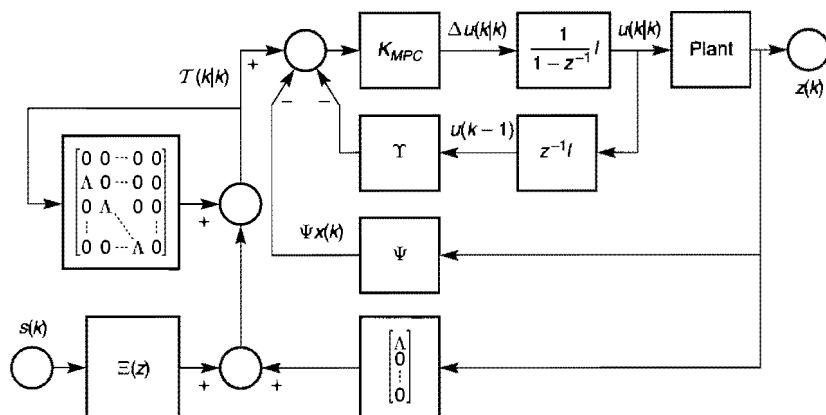


Figure 7.18 Controller structure with reference trajectories: anticipated set-point.

the plant output does not follow the reference signal $r(k)$ closely, then the effects of changing the pre-filter may not be obvious.

In traditional controllers a pre-filter is often used to reduce actuator saturation. The idea is that if operators make ‘step’ changes in the set-point, and if the typical or maximum amplitudes of such changes are known, then passing this change through a low-pass filter before forming the error can be used to limit the error amplitude to such a magnitude that the actuators do not become saturated. With predictive control there seems to be less reason to do this, since we have a more systematic way of handling constraint violations, so there is no particular reason to avoid activating constraints. But there are still good reasons for avoiding saturation:

- ❖ The behaviour of the closed loop is not understood so well when constraints are active.
- ❖ It seems a good idea to keep some control authority ‘in reserve’ for dealing with unexpected disturbances.

So it seems sensible to use pre-filtering to avoid actuator saturation in response to set-points, while still relying on input constraints to handle fast recovery from disturbances — for example, with a deadbeat observer, getting as fast a recovery as the constraints will allow. A sophisticated development of the idea of set-point pre-filtering is that of a *reference governor* [GKT95, BCM97, MKGW00].

In the GPC literature a polynomial, usually denoted by $P(z)$, is used to represent the effects of specifying a reference trajectory in some papers, and of set-point pre-filtering in others. Authors do not always clearly distinguish between these two.

Exercises

7.1 Prove equation (7.8): $S(z) + T(z) = I$.

7.2 Consider a SISO plant with transfer function $P(z) = B(z)/A(z)$, and suppose that $K(z)H(z) = X(z)/B(z)$, as in the case of ‘perfect’ control, where $X(z)$ is

some transfer function. Show that the transfer function from the disturbance d to the plant input u has $B(z)$ as a factor of its denominator. (Hence the feedback loop is internally unstable if $B(z)$ has any zeros outside the unit disc.)

- 7.3** (a) Design a mean-level controller for the plant used in the `mpctutss` demo in the *Model Predictive Control Toolbox*. (An easy way to load the data is to run the demo first. You can then save it to file using the `save` command and retrieve it with the `load` command.)
- (b) With horizons set as for ‘perfect’ control, investigate whether you can achieve similar responses to those achieved with mean-level control, by using non-zero weights $R(i)$ on the input moves. (Consider both set-point and disturbance responses.)
- (c) Investigate the sensitivity and complementary sensitivity functions for some of your designs.
- 7.4** (a) For the paper machine headbox (demo `pmlin` in the *Model Predictive Control Toolbox*), find (by trial and error) the best combination of horizons and weights that you can, with the default (DMC) observer. (Ignore constraints.)
- (b) Try improving your best design so far by changing the observer design.

7.5 A SISO plant with transfer function

$$\frac{-10s + 1}{(s + 1)(s + 30)}$$

is to be controlled using predictive control with a sampling interval of $T_s = 0.2$ sec.

- (a) Design and simulate a ‘perfect’ controller for this plant, and explain the results.
- (b) Explain why increasing H_p will eventually give closed-loop stability (assuming no active constraints). Find the smallest H_p which gives stability, if $H_u = 1$ and $R(i) = 0$. What are the closed-loop poles in this case? Roughly what is the time constant of the set-point response in this case?
- (c) Simulating this system with predictive control, with $H_u = 2$, $H_p = 30$, $Q(i) = 1$ and $R(i) = 0.1$ shows that stability is obtained, but the response to set-point step changes shows that the output moves in the ‘wrong’ direction initially. Show (by simulation) that this effect can be reduced by limiting the input moves to $|\Delta u(k)| < 0.1$ for each k .

7.6 Consider the swimming pool of Exercise 3.3 and Example 7.4.

- (a) Verify, using the *Model Predictive Control Toolbox*, that if the plant model (7.29) is used with the default (DMC) observer, then an oscillating water temperature will result when the air temperature varies sinusoidally as in (3.96).
- (Note: The model (7.29) should be used only for the controller’s internal model, namely the argument `imod` of function `smpc`. The argument `pmod`, which represents the plant, should remain the same as in Exercise 3.3.)

- (b) Modify the function `swimpool` (which is available on this book's web site) so that the air temperature is modelled as a ramp rather than a sinusoidal disturbance, although the actual disturbance remains a sinusoid with period 24 hours. Show, by simulation, that a residual sinusoidal variation of the water temperature remains, but that it is much smaller than the one obtained by assuming a constant disturbance. (For example, with $V = 1$, $T = 1.25$, and $k = 0.3$, a residual variation with amplitude approximately $0.01\text{ }^{\circ}\text{C}$ is obtained. In Exercise 3.3 an amplitude of approximately $0.5\text{ }^{\circ}\text{C}$ is obtained when the disturbance is assumed to be constant.)
- (c) Investigate the sensitivity functions obtained when the air temperature disturbance is modelled as a sinusoid, and as a ramp, respectively, assuming that all constraints are inactive. Verify that in each case the frequency response of the sensitivity becomes zero at the frequency of the assumed disturbance, namely $2\pi/24$ if the sinusoid is assumed, and 0 if the ramp is assumed. Comment on the rate at which the sensitivity approaches zero as $\omega \rightarrow 0$. (*Note:* Use the *Model Predictive Control Toolbox* function `smpccon` to obtain the linear controller for the unconstrained case.)
- 7.7** (a) Verify that, if $\Lambda = 0$ in (7.36), then the controller anticipates set-point changes, and the reference trajectory coincides with the future set-point trajectory.
- (b) Suppose a second-order reference trajectory is preferred to a first-order exponential trajectory:

$$r(k+j|k) = s(k+j) - \frac{e^{\alpha j}}{2}[s(k+j) - z(k)] - \frac{e^{\beta j}}{2}[s(k+j) - z(k)]$$

where α and β are either negative real numbers, or complex conjugates of each other (with negative real parts). Make the corresponding modifications to the development of Section 7.5.

Chapter 8

Robust predictive control

8.1	Formulations of robust control	217
8.2	The tuning procedure of Lee and Yu	221
8.3	The LQG/LTR tuning procedure	225
8.4	LMI approach	233
8.5	Robust feasibility	239
	Exercises	246

8.1 Formulations of robust control

Most formulations of robust control problems take the following general form. The plant is assumed to be known only approximately; this is represented by assuming that the plant lies in a set which can be characterized in some quantitative way. The objective of robust control design is then to ensure that some performance specification is met by the designed feedback system, so long as the plant remains in the specified set.

Most robust control theory assumes that the controller is linear. But, as we have seen, predictive controllers are nonlinear if constraints become active. Nevertheless, this theory remains useful even for predictive control, because many predictive controllers operate in their unconstrained mode most of the time, or at a fixed set of active constraints for long periods of time. In either case, predictive controllers formulated with quadratic costs, linear models and linear constraints behave linearly in these conditions.

In this chapter we begin by examining two approaches to tuning for robust control which assume a linear controller, then go on to look at approaches which acknowledge the presence of constraints.

An interesting but untypical early result on the robustness of unconstrained MPC was



given by Åström in [Åst80]. The problem considered was, in our terms, that of MPC of a SISO system with $H_p = H_u = 1$ and $R = 0$, for an asymptotically stable plant with model

$$y(k+1) = bu(k) \quad (b > 0) \quad (8.1)$$

It was shown that closed-loop stability is obtained if this MPC controller is applied to *any* linear SISO plant whose continuous-time step response $S_c(t)$ is monotonic and positive ($t_1 > t_2 \Rightarrow S_c(t_1) > S_c(t_2)$), providing that the sampling interval T_s is chosen such that

$$S_c(T_s) > \frac{S_c(\infty)}{2} \quad (8.2)$$

and that the model gain is assumed to be large enough:

$$b > \frac{S_c(\infty)}{2} \quad (8.3)$$

This good robustness is obtained at the cost of performance; the closed-loop settling time is longer than the settling time of the open-loop plant if T_s and b are chosen in this way. Although this is a very special result, it shows one rather general feature: stability is frequently obtained with MPC if the prediction horizon is made large enough — in this case, the prediction horizon is the same as the sampling interval T_s , since $H_p = 1$.

8.1.1 Norm-bounded uncertainty

The most common specification of plant uncertainty is ‘norm-bounded’. It is assumed that one has a nominal model of the plant, say a transfer function $P_0(z)$, but that the real plant $P(z)$ is given by a description such as:

$$P(z) = P_0(z) + \Delta \quad (8.4)$$

$$P(z) = P_0(z)[I + \Delta] \quad (8.5)$$

$$P(z) = [M_0(z) + \Delta_M]^{-1}[N_0(z) + \Delta_N] \quad \text{where } P_0(z) = M_0(z)^{-1}N_0(z) \quad (8.6)$$

where, in each case, Δ is a stable bounded operator, and $P_0(z)$ is often normalized in such a way that $\|\Delta\| \leq 1$. Of course one does not know exactly what Δ is. Various assumptions can be made about the nature of Δ : nonlinear, linear time-varying, linear parameter-varying and linear time-invariant being the most common ones. Also various norms can be used; the most commonly used one is the ‘H-infinity’ norm $\|\Delta\|_\infty$ which is defined even for nonlinear systems, as the worst-case ‘energy gain’ of an operator.

A linear feedback controller $K(z)$ designed for the nominal plant $P_0(z)$ will certainly give a stable closed loop if the plant is exactly the same as the model $P_0(z)$. But it will not necessarily be stable when used with the real plant $P(z)$. It is therefore important to have some way of testing whether it will remain stable for all possible plants allowed by the uncertainty description. Suppose that the uncertainty about the plant is represented by the additive model (8.4). Then the feedback combination of the real plant with the controller can be drawn as shown in Figure 8.1. The uncertainty Δ is in parallel with the

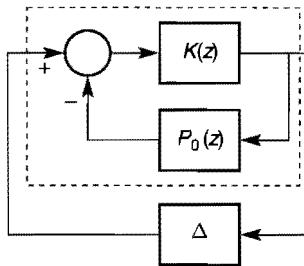


Figure 8.1 Feedback combination of controller and plant, with additive uncertainty model.

nominal plant model P_0 , but the figure has been drawn to show it as being in feedback around the closed loop formed by the controller K and P_0 (which is enclosed by the broken line). The transfer function (matrix) of the block within the broken line can be shown to be $K(z)S(z)$, where $S(z) = [I + P_0(z)K(z)]^{-1}$ is the sensitivity function introduced in Section 7.1, and this transfer function is of course stable, by design. Thus the ‘loop gain’ transfer function is $K S \Delta$. It then follows from the *small-gain theorem*¹ that the feedback combination of this system with the uncertainty block Δ will remain stable if

$$\bar{\sigma}[K(e^{j\omega T_s})S(e^{j\omega T_s})]\|\Delta\|_\infty < 1 \quad (8.7)$$

where $\bar{\sigma}[\cdot]$ denotes the largest singular value — see Mini-Tutorial 8 and Section 7.3.

Inequality (8.7) can be tested for a given design, to check whether robust stability is guaranteed for any plant within the set allowed by the uncertainty description. When tuning a controller, one can try to influence the frequency response properties in such a way as to make (8.7) hold. Note that (8.7) is only a sufficient condition for robust stability; if it is not satisfied, robust stability may nevertheless have been obtained.

Other uncertainty descriptions lead to other tests for robust stability. For example, the ‘input multiplicative’ uncertainty model (8.5) leads to the following sufficient condition for robust stability:

$$\bar{\sigma}[K(e^{j\omega T_s})P_0(e^{j\omega T_s})(I + K(e^{j\omega T_s})P_0(e^{j\omega T_s}))^{-1}]\|\Delta\|_\infty < 1 \quad (8.8)$$

(Note that for multivariable systems, $KP_0 \neq P_0K$, so that the transfer function which appears in (8.8) is not the same as the complementary sensitivity introduced in Section 7.1.) And the ‘coprime factor’ uncertainty model (8.6) leads to the condition:

$$\bar{\sigma}\left\{\begin{bmatrix} K(e^{j\omega T_s}) \\ I \end{bmatrix}[I + P_0(e^{j\omega T_s})K(e^{j\omega T_s})]^{-1}M_0(e^{j\omega T_s})^{-1}\right\}\|[\Delta_M, \Delta_N]\|_\infty < 1 \quad (8.9)$$

Descriptions such as those given in (8.4)–(8.6), in which the only specification on the uncertainty Δ is a norm-bound, are known as *unstructured* uncertainty descriptions. In practice much more specific information is available about the way in which the plant

¹ Alternatively, from a generalization of the classical Nyquist stability theorem

behaviour can vary. For example it may be known that specific physical parameters, such as mass or resistance, can vary within known limits. In such cases a norm-bounded description can still be used, but the operator Δ then is restricted to have a fixed structure. It can usually be arranged that this structure is that of a block-diagonal transfer function, with specified sizes of the blocks. The advantage of doing this is that robust stability tests can then be used which are less conservative than singular-value tests such as (8.7) or (8.8). This is due to the fact that the singular-value tests check whether *any* perturbation Δ of the permitted ‘size’ (as measured by the norm) could destabilize the closed loop, even a perturbation whose structure does not correspond to any possible parameter variation. A more refined test, which checks only perturbations of the permitted size *and* structure, is the *structured singular value*, also commonly referred to as μ . Even this gives sufficient conditions only for robust stability, because it checks for plant perturbations which may be complex-valued (such as phase perturbations in frequency-response properties), even if only real-valued perturbations are known to occur in the plant. In principle, a necessary and sufficient test of robust stability can be obtained by a further refinement, the *real structured singular value*, or ‘real- μ ’. In practice, it is known that ‘real- μ ’ can be extremely difficult to compute (its computation is known to be ‘NP-hard’), and one has to be content with upper and lower bounds for it. Consequently, there is usually some conservativeness in the robust stability robustness test, even if a ‘real- μ ’ test is employed.²

One of the important attributes of the structured singular value is that it can be used to assess not only whether stability is robustly preserved, but a similar test can be performed for robust preservation of performance, if this can be expressed in terms of norm-bounds.

For further details of the norm-bounded approach to modelling plant uncertainty, and corresponding tests for robustness, see [Mac89, ZDG96]. In Section 9.1.6 this approach is illustrated in a case-study. In [PG88] it is argued that structured singular values, together with predictive control, can be made the basis of a systematic approach to process control.

8.1.2 Polytope uncertainty

Another approach to specifying the uncertainty of a plant model is to define regions in which the parameters defining the model must lie. The difficulty is to do this in such a way that some analysis is possible on the resulting description.

Early approaches to robust predictive control assumed that the model was defined as an FIR system of fixed order, and the uncertainty was in the form of bounds on the pulse response coefficients [CM87]. This had the advantage that the uncertainty appeared linearly in the model description. But unfortunately it was later shown that design methods based on such descriptions did not guarantee robustness [ZM95b].

A more recent approach is to assume that a number of ‘corner’ points are known:

$$x(k+1) = A_i x(k) + B_i u(k) \quad i = 1, 2, \dots, L \quad (8.10)$$

² Singular values can be computed using the *Model Predictive Control Toolbox* function `svdfrsp`. μ and ‘real- μ ’ can be estimated using the function `mu` in the *Mu-Analysis and Synthesis Toolbox* for *MATLAB*.

and that the real plant lies in the polytope³ which is the convex hull of these corners:

$$[A, B] = \sum_{i=1}^L \lambda_i [A_i, B_i], \quad \sum_{i=1}^L \lambda_i = 1 \quad (8.11)$$

The real plant does not need to be fixed. It can vary with time, so long as it remains within this polytope [KBM96]. We shall use this uncertainty description in Section 8.4.

8.2

The tuning procedure of Lee and Yu

8.2.1 Simplified disturbance and noise model

In this section we shall outline a tuning procedure proposed by Lee and Yu in [LY94]. They adopt the disturbance and noise model shown in Figure 8.2, which was introduced in [LMG94]. Only output disturbances are assumed to act on the plant, as in the *DMC* model, but they are not assumed to be step disturbances. Instead the disturbances are assumed to be generated by independent stochastic ‘white noise’ processes (shown as the vector $\Delta w(k)$ in the figure), which are integrated to form the vector of stochastic processes $w(k)$, and these are then filtered by first-order low-pass filters to form the stochastic output disturbance vector $d(k)$. The disturbance on the i th output is passed through a filter with a pole at α_i , which corresponds to a time constant of $-T_s / \ln \alpha_i \approx T_s / (1 - \alpha_i)$ if α_i is close to 1, where T_s is the update interval — note that $0 \leq \alpha_i \leq 1$ for stability of the filter. The plant output vector is then assumed to be corrupted by the white measurement noise vector $v(k)$, each element of which is independent of the others.

This model generalizes the *DMC* disturbance model, which is obtained if each $\alpha_i = 0$ and the covariance of $v(k)$ is zero. The use of integrated white noise ensures offset-free performance in the face of constant disturbances and plant/model steady-state gain mismatches. The filters allow the designer to specify that disturbances on various outputs appear with different ‘speeds’, and hence that the disturbance rejection can be performed with a different speed of response on each output. This allows the disturbance rejection to be less aggressive than it would be with the *DMC* model (assuming the same cost function is used), and thus provides a useful additional set of tuning parameters. These parameters can be used to enhance the robustness of the controller: the intuition that control has to be ‘de-tuned’ in order to increase robustness is usually sound,⁴ and one can perform rigorous analysis of the robustness for various choices of the α_i parameters, at least for the case of inactive constraints. In addition to the filter time constants, one can adjust the covariances of the processes $\Delta w(k)$ and $v(k)$, namely the relative strengths of the disturbance and the measurement noise on each output. This

³ A *polytope* is a finite region of n -dimensional space bounded by a finite number of hyperplanes. It is not necessarily convex, though the particular polytope referred to here is convex. The set of solutions to the vector inequality $Ax \leq b$, where x has n elements, is a convex n -dimensional polytope, if it is bounded. A three-dimensional polytope is usually referred to as a *polyhedron*.

⁴ ‘De-tuned’ here is used in the traditional sense, referring to less aggressive control action. Of course if one’s objectives include increasing robustness then the tuning is actually being improved by ‘de-tuning’.

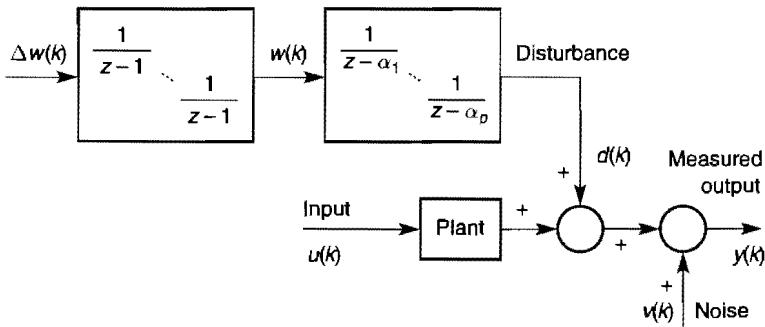


Figure 8.2 The disturbance and noise model used by Lee and Yu [LY94].

again influences the response of the controller to apparent disturbances — if an output measurement is specified as being very noisy, then the estimator, if designed properly, should regard most variations as being due to noise, and so ‘tell’ the predictor — and hence the controller — that there is not much disturbance on that output, and hence not much response is needed. (Note that this is a frequency-independent effect, whereas adjusting the filter time constants ‘shapes’ the frequency response.)

It seems plausible that the filter parameters α_i and the relative strengths of disturbances and noises could be made available to plant operators as on-line tuning parameters, since their meanings can be easily understood. But that depends on the extent to which the effects of adjusting these parameters corresponds to intuition. Lee *et al* [LMG94] and Lee and Yu [LY94] have performed a detailed investigation of these effects. We will give only a summary of their analysis and findings.

An appropriate estimator and predictor, which corresponds to the assumptions on the disturbances and noise, can be obtained by using Kalman filter theory. In order to use this theory, we have to put our plant and assumptions into the standard form (omitting measured disturbances, since we do not need them here):

$$\xi(k+1) = \tilde{A}\xi(k) + \tilde{B}u(k) + \Gamma\Delta w(k) \quad (8.12)$$

$$y(k) = \tilde{C}\xi(k) + v(k) \quad (8.13)$$

where $\xi(k)$ is a suitable state vector, and we can use $\Delta w(k)$ and $v(k)$ because they are already assumed to be white, as required in the standard form. Now, using our usual plant model, our assumptions correspond to:

$$x(k+1) = Ax(k) + Bu(k) \quad (8.14)$$

$$y(k) = Cx(k) + d(k) + v(k) \quad (8.15)$$

where $x(k)$ is the state of the plant itself, and we need a state-space model of the disturbance process $d(k)$. It is convenient to obtain this in two steps: first a model of $d(k)$ obtained from $w(k)$, then a model of $w(k)$ obtained from $\Delta w(k)$. For the first step we have:

$$x_w(k+1) = A_w x_w(k) + w(k) \quad (8.16)$$

$$d(k) = x_w(k) \quad (8.17)$$

where w , x_w and d all have the same dimensions, and $A_w = \text{diag}\{\alpha_i\}$. Combining this with (8.14) and (8.15) gives

$$\begin{bmatrix} x(k+1) \\ x_w(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & A_w \end{bmatrix} \begin{bmatrix} x(k) \\ x_w(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} 0 \\ I \end{bmatrix} w(k) \quad (8.18)$$

$$y(k) = [C \quad I] \begin{bmatrix} x(k) \\ x_w(k) \end{bmatrix} + v(k) \quad (8.19)$$

Since $\Delta w(k) = w(k) - w(k-1)$ we can take the second step by using the ‘differenced’ or ‘velocity’ form of the model as developed in Section 2.4. By analogy with that development, we have:

$$\begin{bmatrix} \Delta x(k+1) \\ \Delta x_w(k+1) \\ \eta(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 & 0 \\ 0 & A_w & 0 \\ CA & A_w & I \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ \Delta x_w(k) \\ \eta(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ CB \end{bmatrix} \Delta u(k) + \begin{bmatrix} 0 \\ I \\ I \end{bmatrix} \Delta w(k) \quad (8.20)$$

$$y(k) = [0 \quad 0 \quad I] \begin{bmatrix} \Delta x(k) \\ \Delta x_w(k) \\ \eta(k) \end{bmatrix} + v(k) \quad (8.21)$$

This is now in the form of the standard model (8.12)–(8.13), with u replaced by Δu . Note that the stochastic disturbance $\Delta w(k)$ does not excite the plant state $\Delta x(k)$, which means that any unstable modes in the plant model will not be stabilized by the Kalman filter (estimator) gain. So, just like the DMC model, this disturbance model can only be used with asymptotically stable plants. This is true for any model which treats all disturbances as being output disturbances which do not affect the plant state.

To complete the model, the covariance matrices of $\Delta w(k)$ and $v(k)$ must be specified. By assumption these are both diagonal: $W = \text{diag}\{\rho_i\}$, $V = \text{diag}\{\sigma_i\}$, with $\rho_i \geq 0$ and $\sigma_i \geq 0$.

The Kalman filter gain can now be computed by solving the appropriate discrete-time Riccati equation — see (8.31). (In [LMG94] the plant state vector is chosen to be made up of predicted values of the plant output. It is pointed out that this choice reduces the dimension of the Riccati equation which must be solved. But this is now a minor issue, given current computing power, and considering that this equation needs only be solved off-line.⁵) It is also shown in [LMG94] that the DMC estimator/predictor is obtained if $\alpha_i = 0$ and $V = 0$.

Note that the *Model Predictive Control Toolbox* function `smpcest` has an option in which this disturbance/noise model is assumed, and only the α_i , ρ_i and σ_i parameters need to be specified.

⁵ Computing the Kalman filter gain for a random system with 100 states and 20 outputs, using the *MATLAB Control System Toolbox*'s function `d1qe` takes about 30 seconds on a Pentium II processor running at 333 MHz with 128 Mbyte of main memory.

It is easy to verify that the Riccati equation (8.31) for the model (8.20)–(8.21) has a solution with the structure

$$P = \begin{bmatrix} 0 & 0 & 0 \\ 0 & P_{22} & P_{23} \\ 0 & P_{23}^T & P_{33} \end{bmatrix} \quad (8.22)$$

(assuming an asymptotically stable plant, of course) where the dimensions of the blocks correspond to the dimensions of the vectors x , x_w and y . Consequently the Kalman filter gain matrix is given by

$$L = \begin{bmatrix} 0 \\ P_{23} \\ P_{33} \end{bmatrix} [P_{33} + V]^{-1} \quad (8.23)$$

Furthermore, since A_w , W and V are all diagonal, the blocks P_{22} , P_{23} and P_{33} are all diagonal (and square). Consequently L has the form

$$L = \begin{bmatrix} 0 \\ L_{\Delta w} \\ L_\eta \end{bmatrix} = \begin{bmatrix} 0 \\ \text{diag}\{\phi_i\} \\ \text{diag}\{\psi_i\} \end{bmatrix} \quad (8.24)$$

In [LY94] it is shown that

$$\phi_i = \frac{\psi_i^2}{1 + \alpha_i - \alpha_i \psi_i} \quad (8.25)$$

and that $\psi_i \rightarrow 0$ as $\rho_i/\sigma_i \rightarrow 0$. Note that $\psi_i \rightarrow 1$ as $\sigma_i \rightarrow 0$. It can be shown that $0 < \psi_i \leq 1$ for all values of ρ_i/σ_i .

In [LY94] the tuning procedure is expressed in terms of adjusting the gain ψ_i directly, rather than adjusting the variances ρ_i and σ_i .

8.2.2 Tuning procedure

Lee and Yu [LY94] suggest a two-step tuning procedure. The first step is concerned with choosing cost-function weights and horizons such as to obtain nominal stability (i.e. closed-loop stability, assuming that the linear plant model is perfect). They therefore suggest choosing the control penalty weighting matrix as $R = 0$, making the control horizon as large as practically possible, and the prediction horizon at least as large as the control horizon, possibly infinite. The reason for making the horizons as large as possible is to make the ‘regulator’ part of the controller behave as closely as possible to an infinite-horizon linear quadratic state-feedback controller, since such a controller usually exhibits very good feedback properties. (More will be said about this in Section 8.3.) In the light of Section 6.2 a better suggestion now is to make both horizons infinite, so that the exact ‘linear quadratic’ behaviour is obtained, so long as all the constraints are inactive. The reason for choosing $R = 0$ is so as not to limit the controller action unnecessarily at this stage — but a further reason will be seen in Section 8.3.

The second step consists of ‘de-tuning’ the controller so as to obtain robustness with respect to plant/model mismatch. This is done by adjusting the disturbance and noise parameters α_i , ρ_i and σ_i , or (almost) equivalently the parameters α_i and ψ_i .

If there is only one output being controlled then there is only one filter parameter, α_1 , and one gain, ψ_1 , to adjust. The situation is then identical to adopting a second-order ‘observer polynomial’ $C(z)$ in GPC — see Sections 4.2.4 and 4.2.5 and Exercise 8.2. Broadly speaking, increasing ψ_1 (or ρ_i/σ_i) increases the bandwidth of the closed loop with the whole predictive controller in place. More precisely, it increases the frequency at which the gain of the sensitivity function $|S(e^{j\omega T_s})|$ becomes close to 1, and at which the gain of the complementary sensitivity $|T(e^{j\omega T_s})|$ begins to deviate significantly from 1. Increasing α_1 has the effect of improving (i.e. reducing) the sensitivity at low frequencies, but at the cost of increasing both the sensitivity and the complementary sensitivity at high frequencies, and hence reducing robustness to modelling errors and leading to underdamped (‘ringing’) behaviour. The appropriate performance/robustness trade-off has to be decided by examining the Bode plots of these functions for specific cases.

For the multivariable case, Lee and Yu [LY94] give an extensive discussion of how the tuning should be related to what is known about the uncertainty structure of the plant, as referred to either its inputs or outputs. They suggest that the disturbance/noise parameters should be used to tune for robustness in the face of output uncertainty, but that the control penalty matrix R should be used to tune for robustness against input uncertainty. However, in most cases information about the uncertainty structure is not available sufficiently precisely to allow one to follow this advice. In such cases one is reduced to adjusting the α_i and ψ_i (or ρ_i , σ_i) parameters, using one’s understanding of their effects in the single-output case, monitoring the singular values of the sensitivity and complementary sensitivity functions, and accepting that their effects interact with each other.

Disadvantages of increasing R from its initial value of 0 are that its effects are sometimes counter-intuitive, and that they can be completely lost if output constraints become active, since the need to respect constraints then takes precedence over minimization of the cost. On the other hand, the functioning of the state estimator is not affected by the constraints, so some of the benefits achieved by tuning the estimator may remain in place even if output constraints become active. This is a very heuristic argument, but there is some practical experience which indicates that it has some validity.

8.3 The LQG/LTR tuning procedure

‘Loop Transfer Recovery’, or *LTR*, is a tuning technique developed for use with the ‘Linear Quadratic Gaussian’, or *LQG*, design approach. *LQG* theory says that if we have a Linear plant model, Quadratic penalty cost, and disturbance and noise signals with Gaussian probability distributions, then the optimal control problem separates into two sub-problems, which can be solved independently:

- 1. Estimate:** Use a Kalman filter to obtain the optimal estimate $\hat{x}(k|k)$ of the plant state (from plant input and output signals). See Mini-Tutorial 9 for details.

Consider a plant model

$$\dot{x}(k+1) = Ax(k) + Bu(k) + \Gamma w(k) \quad y(k) = Cx(k) + v(k) \quad (8.26)$$

in which w and v are white noise processes with covariance matrices $E\{ww^T\} = W \geq 0$, $E\{vv^T\} = V > 0$, and $E\{wv^T\} = 0$ (i.e. w and v are uncorrelated). w is supposed to be a disturbance which affects the plant behaviour (and must therefore be ‘tracked’ by the filter in some sense), whereas v is supposed to be a measurement noise (which must therefore be ‘rejected’ by the filter in some sense). An optimal (minimum error variance) estimate of the state x can be obtained by iterating the following equations which define the Kalman filter [AM79, BGW90, BH75, KR72]:

$$\text{Correction: } \hat{x}(k|k) = \hat{x}(k|k-1) + L'(k)[y(k) - C\hat{x}(k|k-1)] \quad (8.27)$$

$$\text{Prediction: } \hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) \quad (8.28)$$

where the Kalman filter gain $L'(k)$ is given by

$$L'(k) = P(k)C^T \left[CP(k)C^T + V \right]^{-1} \quad (8.29)$$

and

$$P(k+1) = AP(k)A^T - AP(k)C^T \left[CP(k)C^T + V \right]^{-1} CP(k)A^T + \Gamma W \Gamma^T \quad (8.30)$$

The model and covariances can be time-varying, but if they are all constant then the Riccati equation (8.30) converges to $P_\infty \geq 0$, which is a solution to the (filtering) *algebraic Riccati equation*:

$$P_\infty = AP_\infty A^T - AP_\infty C^T \left[CP_\infty C^T + V \right]^{-1} CP_\infty A^T + \Gamma W \Gamma^T \quad (8.31)$$

and from which the ‘stationary’ Kalman filter gain is obtained as

$$L'_\infty = P_\infty C^T \left[CP_\infty C^T + V \right]^{-1} \quad (8.32)$$

Note that the Kalman filter equations have the form of an observer, with a special choice of observer gain. If the pair $(A, \Gamma W^{1/2})$ is stabilizable and the pair (C, A) is detectable, then the closed-loop observer state-transition matrix $A(I - L'_\infty C)$ has all its eigenvalues within the unit disc, so that the observer is stable. Also, note that $V \geq 0$ or even $V = 0$ is allowed, providing that $CP_\infty C^T > 0$.

Optimal *predictions* are obtained by assuming that future values of the disturbance and noise will be equal to their mean values (i.e. 0) and hence are obtained from the optimal filtered state estimate $\hat{x}(k|k)$ by using the recursion

$$\hat{x}(k+1|k) = A\hat{x}(k|k) + Bu(k) \quad (8.33)$$

$$\hat{x}(k+l|k) = A\hat{x}(k+l-1|k) + B\hat{u}(k+l-1) \quad \text{for } l > 1. \quad (8.34)$$

Mini-Tutorial 9 The Kalman filter.

2. Control: Find the optimal state feedback gain matrix K_{LQ} for the deterministic (i.e. no disturbance or noise) control problem with the same cost function for the same plant. (See Mini-Tutorial 7 for details.) Then use the optimal state estimate as if it were a measurement of the true state: apply the feedback law $u(k) = -K_{LQ}\hat{x}(k|k)$.

We have already used the *LQG* approach in the previous section. The proposal to use the *LQG/LTR* tuning procedure is very close to the procedure proposed in the previous section, namely the Lee and Yu procedure. It generalizes it by using a more

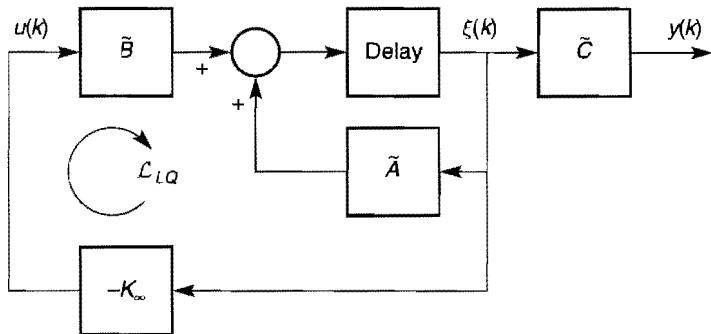


Figure 8.3 The LQ state feedback regulator.

general disturbance model. This allows more freedom to shape the frequency response, if required, and it removes the restriction to stable plants. Also, it allows the disturbance model to include correlated disturbances, which is an advantage in those applications in which strong correlation is present, and is known precisely enough to be worth modelling — for example in the control of ships, in which state disturbances are highly correlated because they are due to common causes such as waves. On the other hand, the complexity is not increased very much — if at all — because the weights in the cost function are fixed in advance, and only the noise model is adjusted. Also, the tuning can be pursued by monitoring an *open-loop* frequency response (that of the Kalman filter), which makes it easier to take inherent plant characteristics into account while tuning.

The authors of [BGW90] advocated using *LQG/LTR* to design adaptive predictive controllers. Since the approach requires the use of infinite horizons, it did not seem possible, at that time, to apply this idea to constrained predictive control. Now, however, since we know how to solve constrained predictive control problems with infinite horizons — especially since the work of [SR96b], in which both the prediction and the control horizons are infinite — it is possible to use *LTR* methods for these problems. In [LMG94] it is also proposed that the *LQG/LTR* idea can be combined with the use of the simplified disturbance model which was described in the previous section.

At the heart of the *LQG/LTR* approach is the recognition that both the (infinite horizon) 'LQ state-feedback' scheme and the (steady-state) Kalman filter are themselves feedback systems. This is shown in Figures 8.3 and 8.4. Furthermore, each of them usually exhibits excellent 'feedback properties'. In continuous time, it is known that each of the feedback loops identified by \mathcal{L}_{LQ} and \mathcal{L}_{KF} in these figures has infinite gain margin against gain increases in individual input channels, a gain margin of at least 2 against gain decreases in individual input channels, and at least 60° phase margin against unmodelled phase lags in individual input channels, etc. [Kal64, SA77]. In discrete time, these properties are not achieved exactly, and cannot be guaranteed. Nevertheless, providing that reasonable demands are made of these feedback systems — usually, that the loop gains have fallen to small values well before the Nyquist frequency π/T_s is approached — then they usually exhibit comparable properties.

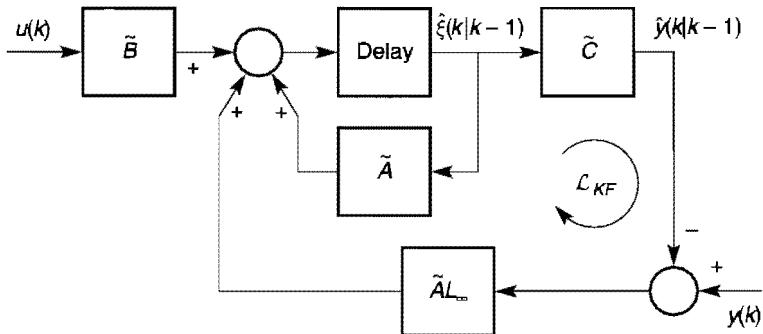


Figure 8.4 The Kalman filter as a feedback system.

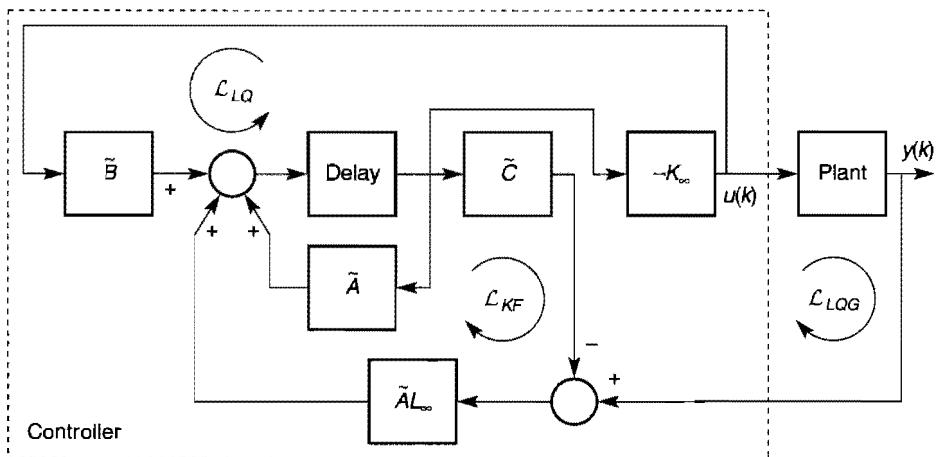


Figure 8.5 The closed loop with LQG controller.

The fact that the complete closed-loop system containing the plant and an *LQG* controller is made up of these two feedback systems, each of which has such good feedback properties, was often held to imply that the complete system would have good feedback properties. In other words, that the loop identified by \mathcal{L}_{LQG} in Figure 8.5 would inherit these good properties. Rosenbrock [Ros71] pointed out that this was not necessarily the case, and Doyle [Doy78] showed that *LQG* loops could exhibit arbitrarily bad feedback behaviour — in particular, arbitrarily small stability margins. These and similar observations motivated the development of ‘ H_∞ control theory’; but in the meantime the *LTR* tuning technique was developed for use with the *LQG* approach [DS81, Mac85, SA87].

LQG/LTR theory shows that, under certain circumstances, if the state feedback gain K_∞ is obtained by solving the *LQ* problem with $Q = C^T C$ and $R \rightarrow 0$, then $\mathcal{L}_{LQG} \rightarrow \mathcal{L}_{KF}$. To be more precise, the return-ratio, the sensitivity function and the complementary sensitivity function, evaluated at the output of the plant, converge to the corresponding functions for the loop \mathcal{L}_{KF} , evaluated at the output of the plant model

(assuming that the plant model is exact). The convergence is ‘pointwise-in-frequency’, which means that the frequency responses get closer at each frequency as R is reduced towards 0 — although convergence at some frequencies may be slower than at others. So the Kalman filter feedback properties are ‘recovered’ at the plant output by this procedure.

Here we will concentrate on how to use this procedure for tuning predictive controllers, and not derive this convergence result. For the technical details the reader is referred to [DS81] for the original development, in continuous time, and to [Mac85] and [BGW90] for the development in discrete time (which is surprisingly different). The latter reference discusses how to apply *LQG/LTR* to unconstrained predictive control. Extensive discussions of the use of *LQG/LTR* in continuous time design are available in [SA87, Mac89, SCS93]. But a few points about the details are in order:

1. The convergence theorem assumes that the linear plant model is square — has equal numbers of inputs and outputs. The technique can still be used if there are more outputs than inputs, by adding ‘dummy inputs’.
2. The convergence theorem assumes that the linear plant model is minimum-phase — that is, that there are no zeros outside the unit disc, and that the product CB is non-singular. If this does not hold, convergence is still usually obtained in practice, at least for frequencies below those at which the effects of the non-minimum-phase zeros become significant. This is not usually a significant restriction, because such zeros in any case limit the achievable bandwidth obtained using any feedback control. In particular, any non-minimum-phase zeros introduced by discretization (i.e. which do not arise from right half-plane zeros in the continuous-time system) place no practical restriction on the technique.
3. In continuous time there is a complete duality between the Kalman filter and the LQ state feedback controller. Consequently, one can follow a dual procedure, namely tuning the LQ state-feedback loop, then recovering its properties at the *input* of the plant by choosing the disturbance and noise covariances according to a formula ($W = BB^T$, $V \rightarrow 0$). This allows, among other things, the use of the *LTR* technique when there are more inputs than outputs. In discrete time the duality is not so complete, and hence the corresponding theorem does not hold. But again, in practice, enough recovery occurs in practice to be useful, providing that the sampling frequency is high enough, relative to the demanded bandwidth. (Basically, if the sampling frequency is high enough, then all the continuous-time results hold approximately enough.)

Instead of the model (8.14)–(8.15) which we used in the previous section, we will now use the model:

$$x(k+1) = Ax(k) + Bu(k) + \Gamma_x d(k) \quad (8.35)$$

$$y(k) = Cx(k) + v(k) \quad (8.36)$$

The disturbance d now acts on the state rather than the output. It can be generated by the model

$$x_w(k+1) = A_w x_w(k) + \Gamma_w w(k) \quad (8.37)$$

$$d(k) = C_w x_w(k) + D_w w(k) \quad (8.38)$$

where now the dimensions of x_w and d are not fixed, and A_w need not be diagonal. Now (8.18)–(8.19) is replaced by

$$\begin{bmatrix} x(k+1) \\ x_w(k+1) \end{bmatrix} = \begin{bmatrix} A & \Gamma_x C_w \\ 0 & A_w \end{bmatrix} \begin{bmatrix} x(k) \\ x_w(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} \Gamma_x D_w \\ \Gamma_w \end{bmatrix} w(k) \quad (8.39)$$

$$y(k) = [C \quad 0] \begin{bmatrix} x(k) \\ x_w(k) \end{bmatrix} + v(k) \quad (8.40)$$

One could take $w(k)$ to be white noise and $D_w = I$, which would allow any desired spectral density to be modelled for $d(k)$ by (8.37)–(8.38). But since we usually want to make the spectrum of d unbounded at frequency $\omega = 0$, in order to obtain offset-free tracking, it is more convenient to consider $w(k)$ to be integrated white noise, as we did in the previous section. That is, we assume $\Delta w(k)$ to be white noise. In this case we get (by analogy with the development in Section 2.4) the ‘differenced’ form

$$\begin{bmatrix} \Delta x(k+1) \\ \Delta x_w(k+1) \\ \eta(k+1) \end{bmatrix} = \begin{bmatrix} A & \Gamma_x C_w & 0 \\ 0 & A_w & 0 \\ CA & C\Gamma_x C_w & I \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ \Delta x_w(k) \\ \eta(k) \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ CB \end{bmatrix} \Delta u(k) + \begin{bmatrix} \Gamma_x D_w \\ \Gamma_w \\ C\Gamma_x D_w \end{bmatrix} \Delta w(k) \quad (8.41)$$

$$y(k) = [0 \quad 0 \quad I] \begin{bmatrix} \Delta x(k) \\ \Delta x_w(k) \\ \eta(k) \end{bmatrix} + v(k) \quad (8.42)$$

In many cases the full complexity of this model is not needed. The return-ratio which is recovered by the *LTR* procedure is that of the Kalman filter shown in Figure 8.4, with the loop broken at the output of the plant model, namely $\tilde{C}[zI - \tilde{A}]^{-1}\tilde{A}L'_\infty$. Suppose that the disturbance $d(k)$ is just taken to be integrated white noise. In that case the state x_w is not needed and we can take $D_w = I$ (so that $d(k) = w(k)$), so the model simplifies to

$$\begin{bmatrix} \Delta x(k+1) \\ \eta(k+1) \end{bmatrix} = \begin{bmatrix} A & 0 \\ CA & I \end{bmatrix} \begin{bmatrix} \Delta x(k) \\ \eta(k) \end{bmatrix} + \begin{bmatrix} B \\ CB \end{bmatrix} \Delta u(k) + \begin{bmatrix} \Gamma_x \\ C\Gamma_x \end{bmatrix} \Delta w(k) \quad (8.43)$$

$$y(k) = [0 \quad I] \begin{bmatrix} \Delta x(k) \\ \eta(k) \end{bmatrix} + v(k) \quad (8.44)$$

Hence the Kalman filter return-ratio is

$$\tilde{C}[zI - \tilde{A}]^{-1}\tilde{A}L_\infty = [0 \quad I] \begin{bmatrix} zI - A & 0 \\ -CA & (z-1)I \end{bmatrix}^{-1} \begin{bmatrix} L_{\Delta x} \\ L_\eta \end{bmatrix} \quad (8.45)$$

$$= [0 \quad I] \begin{bmatrix} (zI - A)^{-1} & 0 \\ \frac{CA(zI - A)^{-1}}{z-1} & \frac{I}{z-1} \end{bmatrix} \begin{bmatrix} L_{\Delta x} \\ L_\eta \end{bmatrix} \quad (8.46)$$

$$= \frac{CA(zI - A)^{-1}L_{\Delta x} + L_\eta}{z-1} \quad (8.47)$$

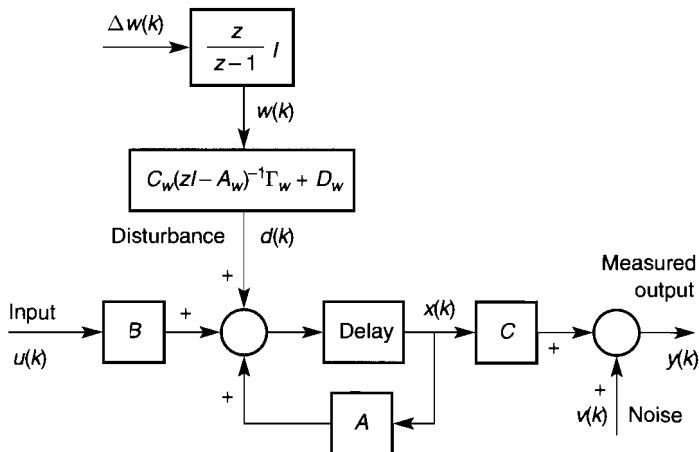


Figure 8.6 Disturbance and noise model used for LQG/LTR.

This shows that the return-ratio has integral action (poles at +1), as expected because of the ‘integrated white noise’ disturbance model,⁶ and that there are enough degrees of freedom in the Kalman filter gains $L_{\Delta x}$ and L_η to adjust the gains of the return-ratio (and hence of the closed-loop bandwidth). There is also some freedom to adjust the ‘loop shape’, namely the behaviour of the return-ratio with frequency, by adjusting the relative magnitudes of these filter gains. Of course, $L_{\Delta x}$ and L_η cannot be adjusted directly, but by means of adjustments to the parameters of the disturbance/noise model, namely Γ_x and the covariance matrices W and V . But these parameters have some intuitive meaning, whereas $L_{\Delta x}$ and L_η do not. If the outputs are ‘not too noisy’ then one can set $V = 0$, which gives $L_\eta = I$. This gives ‘deadbeat tracking’ of the measured outputs — see Section 2.6.3 — and still leaves considerable degrees of freedom to adjust the return-ratio. So this simplification still leaves enough tuning freedom for many applications. Note that, in contrast to the tuning procedure presented in the previous section, it is not limited to stable plants.

If more ‘shaping’ of the return-ratio with frequency is needed — for example to boost the gain in a frequency region in which disturbance power is known to be high — then the full model (8.41)–(8.42) must be used. A block diagram of the disturbance/noise model used in this section is shown in Figure 8.6. This should be compared with Figure 8.2.

Once the disturbance model has been adjusted so that the Kalman filter return-ratio, and the associated sensitivity and complementary sensitivity functions, have ‘good’ frequency characteristics across the whole frequency range, then the ‘recovery step’ is taken. In relation to the standard LQG system shown in Figure 8.5, this is done by using

⁶ The number of integrators is given by the rank of the residue matrix of the return-ratio at $z = 1$, namely by $\lim_{z \rightarrow 1} (z - 1) \tilde{C}[zI - \tilde{A}]^{-1} \tilde{A}L'_\infty = CA(I - A)^{-1}L_{\Delta x} + L_\eta$. Usually the rank of this matrix is the same as the number of outputs, so that there is an integrator for each output.

the cost function

$$\begin{aligned}
 V(k) &= \sum_{l=1}^{\infty} \left\{ ||\hat{\xi}(k+l|k)||_{\tilde{C}^T \tilde{C}}^2 + \rho ||\Delta \hat{u}(k+l-1|k)||^2 \right\} \\
 &= \sum_{l=1}^{\infty} \left\{ ||\hat{\eta}(k+l|k)||^2 + \rho ||\Delta \hat{u}(k+l-1|k)||^2 \right\} \\
 &= \sum_{l=1}^{\infty} \left\{ ||\hat{y}(k+l|k)||^2 + \rho ||\Delta \hat{u}(k+l-1|k)||^2 \right\}
 \end{aligned} \tag{8.48}$$

where the last equality follows because the mean value of future measurement noise is always zero, so $\hat{v}(k+l|k) = 0$. Recovery of the Kalman filter characteristics is then obtained at the plant output as $\rho \rightarrow 0$ (under the conditions specified earlier). But this is in the standard form of a predictive control cost function, with infinite horizons and fixed weights (except for the scalar parameter ρ). So all the tuning has been transferred to the design of the disturbance model, and a fixed, but sensible, control cost function remains.

How far should one reduce ρ ? It is possible to set $\rho = 0$ (in fact, for discrete-time *LQG/LTR* the main convergence theorem relates to this extreme case), but it is usually not a good idea. The continuous-time Kalman filter (and LQ state-feedback) has excellent feedback properties only because it behaves like a set of first-order lags at high frequencies. In the single-output case, the phase lag never exceeds 90° . This has an associated penalty, which is that the loop gain decreases relatively slowly at high frequencies — slope -1 on log-log scales, or -20 decibels per decade of frequency. Since all real plants exhibit larger high-frequency phase lags, and steeper attenuation of gain with frequency, it would only be possible to recover these characteristics perfectly by making the controller have ‘high-pass’ characteristics at high frequencies — that is to have increasing gain at high frequencies. But this is always very undesirable, because there is usually no useful frequency content at high frequencies, and the result would only be to amplify measurement noise, with consequent rapid actuator movements and increased risk of hitting actuator constraints. Now as ρ is reduced the usual effect is for the transfer functions evaluated at the plant output to converge to the Kalman filter transfer functions over an increasing range of frequencies. The convergence is usually most rapid at low frequencies. Therefore one should monitor the return-ratio, and the sensitivity $S(z)$ and complementary sensitivity $T(z)$ as ρ is reduced, and stop reducing ρ once the feedback properties have become adequately good over a wide enough range of frequencies. ‘Wide enough’ means to at least the desired closed-loop bandwidth, and a little way beyond. This whole procedure is based on ‘continuous-time thinking’, so it cannot be expected to work if the sampling interval T_s is too long, relative to the desired closed-loop bandwidth. The Nyquist frequency π/T_s should be (at least) a factor of 10, say, larger than the required bandwidth.

In [RM00] it is shown how MPC can be tuned to replicate the behaviour of an H_∞ design, rather than that of an LQG design, when the predictive controller is operating in its linear mode — that is, when all constraints are inactive. This is done while retaining a quadratic cost function.

8.4 LMI approach

8.4.1 Overview

In [KBM96] a proposal is made to address the issues of maintaining stability as well as respecting constraints despite the uncertainty in the model description. That is, robust stability and robust constraint satisfaction are addressed. This work exploits the very powerful methods which have recently become available for solving *Linear Matrix Inequalities* (or *LMIs*), and the connections which exist between *LMIs* and control theory [BGFB94]. The approach can be applied to both norm-bounded structured uncertainty descriptions, and to polytope descriptions.

Although the formulation no longer leads to a *QP* problem, it does lead to an *LMI* optimization problem which is a convex problem. The algorithms available for solving *LMI* optimization problems appear to be very fast, so that the whole proposal seems to be a plausible candidate for on-line use.

The formulation of [KBM96] looks rather different from the one that we have been considering. The cost function and the constraints are similar to the ones we have used, the difference being that $\|\boldsymbol{u}\|$ is penalized rather than $\|\Delta\boldsymbol{u}\|$ — but this difference can be overcome by reformulating the problem as in Section 6.3. (If this is not done then properties such as integral action will not be obtained.) The bigger difference is that the optimization is performed over *state feedback matrices* K_k , such that $\hat{\boldsymbol{u}}(k+i|k) = -K_k \boldsymbol{x}(k)$ for all $i > 0$. (This is similar to the approach taken in [SR96b].) At each step it is assumed that the same state feedback matrix will be used over an infinite prediction horizon, and this guarantees that this matrix is stabilizing. But in addition to stabilizing the nominal plant, it is guaranteed that any plant in the uncertainty set will be stabilized. This is achieved by obtaining K_k as a solution to an *LMI* problem.

Satisfaction of constraints is also guaranteed by solving an *LMI* problem. If it is assumed that the current state satisfies $\boldsymbol{x}(k)^T X \boldsymbol{x}(k) < 1$ for some $X > 0$, the problem of finding a K_k which ensures that $\boldsymbol{x}(k+i)^T X \boldsymbol{x}(k+i) < 1$ for all $i > 0$ can be posed as an *LMI* problem. This *LMI* problem can be combined with the previous one, which gives robust stability, resulting in a bigger *LMI* problem.⁷

Further *LMI* problems can be posed to handle input constraints, and certain aspects of performance requirements (such as speed of response). These can all be combined with the previous *LMIs*.

An important property of the proposed approach is that if feasibility is obtained at time k , then it is maintained over the whole infinite prediction horizon.

There are potential problems with this formulation. Firstly, it is not clear whether it all extends to the case when an observer is used to estimate the state. Secondly, the *LMI* optimization problems always minimize an upper bound for the quantity which is really of interest. It is possible that such upper bounds are very conservative in some cases, which could seriously reduce the practicality of the approach. However, in [KBM96] it is claimed that typically the bounds are not too conservative. And thirdly, although algorithms for solving *LMI* problems are fast, the problem size becomes very substantial for real applications, so the use of this approach in real time will be restricted to applications with relatively slow dynamics.

⁷ A basic property of *LMIs* is that a set of *LMIs* can be expressed as a single *LMI* — see Mini-Tutorial 10.

8.4.2 Robustness without constraints

It is assumed that the system is modelled by a time-varying linear model:

$$x(k+1) = A(k)x(k) + B(k)u(k) \quad (8.49)$$

$$z(k) = C_z(k)x(k) \quad (8.50)$$

in which the notation $A(k)$ etc., means that the matrices in this model can change at each step, but it is not meant to imply that their variation is known ahead of time (as a function of k). Because their precise variation is not known, the model behaviour is uncertain. The ‘amount’ of uncertainty is prescribed by the assumption that

$$[A(k), B(k)] \in \Omega \quad (8.51)$$

where Ω is a convex polytope derived from either a ‘norm-bounded’ uncertainty description or from a ‘polytope’ uncertainty description.

The cost function is defined to be

$$V_\infty(k) = \sum_{j=0}^{\infty} \{ ||\hat{x}(k+j|k)||_{Q_1}^2 + ||\hat{u}(k+j|k)||_R^2 \} \quad (8.52)$$

Robustness against the effects of uncertainty is obtained by choosing the input to solve the following ‘min-max’ problem:

$$\min_{\mathcal{U}(k)} \max_{[A(k+j), B(k+j)] \in \Omega} V_\infty(k) \quad (8.53)$$

where $\mathcal{U}(k) = [\hat{u}(k|k)^T, \dots, \hat{u}(k+H_u-1|k)^T]^T$, and the maximization is over all $j \geq 0$.

Now suppose that a function $V(x) = x^T P x$ exists, with $P > 0$, such that for any pair $\hat{x}(k+j|k), \hat{u}(k+j|k)$ corresponding to the uncertain model (8.49),

$$V(\hat{x}(k+j+1|k)) - V(\hat{x}(k+j|k)) \leq -||\hat{x}(k+j|k)||_{Q_1}^2 - ||\hat{u}(k+j|k)||_R^2 \quad (8.54)$$

for each $j \geq 0$. Then such a function is an upper bound for the cost $V_\infty(k)$. This can be shown as follows. Assume that $V_\infty(k) < \infty$, then $\hat{x}(\infty|k) = 0$, and hence $V(\hat{x}(\infty|k)) = 0$. Now sum both sides of (8.54) from $j = 0$ to $j = \infty$, which gives

$$-V(x(k|k)) \leq -V_\infty(k) \quad (8.55)$$

Since (8.54) was assumed to hold for any model in the assumed uncertainty set, we have that

$$\max_{[A(k+j), B(k+j)] \in \Omega} V_\infty(k) \leq V(x(k|k)) \quad (8.56)$$

Consequently we can replace the nasty problem (8.53) by the nicer problem:

$$\min_{\mathcal{U}(k)} V(x(k|k)) \quad (8.57)$$

A *Linear Matrix Inequality*, or *LMI*, is a matrix inequality of the form

$$F(v) = F_0 + \sum_{i=1}^l v_i F_i \geq 0 \quad (8.58)$$

where each F_i is a symmetric matrix, and the scalar variables v_i appear in the inequality affinely. It is clear that a set of *LMIs*, $F_1(v) \geq 0, \dots, F_m(v) \geq 0$ is the same as one large *LMI*, obtained by assembling all the matrices $F_i(v)$ into one large block-diagonal matrix.

The importance of *LMIs* is due to the fact that optimization problems of the kind

$$\min_v c^T v \quad \text{subject to} \quad F(v) \geq 0 \quad (8.59)$$

where $F(v) \geq 0$ is an *LMI*, are convex problems, that very many convex problems can be written in this form [BGFB94], and that there are very efficient algorithms for solving problems of this kind [Mat].

One of the major reasons why *LMIs* can be used to represent such a wide range of problems is that convex quadratic inequalities are equivalent to *LMIs*. In particular, if $Q(v) = Q(v)^T$ and $R(v) = R(v)^T$, and $Q(v)$, $R(v)$ and $S(v)$ are all affine in v , then the pair of inequalities

$$Q(v) > 0 \quad (8.60)$$

$$R(v) - S(v)^T Q(v)^{-1} S(v) \geq 0 \quad (8.61)$$

is equivalent to the *LMI*

$$\begin{bmatrix} Q(v) & S(v) \\ S(v)^T & R(v) \end{bmatrix} \geq 0 \quad (8.62)$$

Example: Consider the convex QP problem ($Q \geq 0$):

$$\min_x x^T Qx + q^T x + r \quad \text{subject to} \quad Ax \geq b \quad (8.63)$$

This is equivalent to

$$\min_{\gamma, x} \gamma \quad \text{subject to} \quad \begin{cases} \gamma - x^T Qx - q^T x - r \geq 0 \\ Ax \geq b \end{cases} \quad (8.64)$$

which is equivalent to

$$\min_{\gamma, x} \gamma \quad \text{subject to} \quad \begin{bmatrix} I & Q^{1/2}x \\ x^T Q^{1/2} & \gamma - r - q^T x \end{bmatrix} \geq 0 \quad \text{and} \quad \begin{bmatrix} (Ax - b)_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & (Ax - b)_m \end{bmatrix} \geq 0 \quad (8.65)$$

where $(Ax - b)_i$ denotes the i th element of $Ax - b$, which is in the form of (8.59) --- see Exercise 8.5.

Mini-Tutorial 10 Linear matrix inequalities.

subject to the constraints that (8.54) holds and $P > 0$. Some conservativeness enters the problem formulation here, because $V(x(k|k))$ is only an upper bound for the cost that we really want to minimize. But it is stated in [KBM96] that this is not a source of much conservativeness in practice. The reason that (8.57) is a relatively nice optimization problem is that it is convex, whereas (8.53) is not. Problem (8.57) can be solved with the aid of *LMIs*.

Solving problem (8.57) is equivalent to solving the problem

$$\min_{\gamma, P} \gamma \quad \text{subject to} \quad x(k|k)^T P x(k|k) \leq \gamma \quad (8.66)$$

where we understand that the solution P depends on $\mathcal{U}(k)$ in some way, which will become clear shortly. But $P > 0$, so we can define $Q = \gamma P^{-1} > 0$, which makes the inequality constraint equivalent to $1 - x(k|k)^T Q^{-1} x(k|k) \geq 0$. From the equivalence between the pair of conditions (8.60) and (8.61), and (8.62), we see that this constraint is equivalent to the *LMI*

$$\begin{bmatrix} Q & x(k|k) \\ x(k|k)^T & 1 \end{bmatrix} \geq 0 \quad (8.67)$$

Now we assume that the control signal will be determined by the state feedback law $\hat{u}(k+j|k) = K_k \hat{x}(k+j|k)$, for some matrix K_k , and we substitute for $\hat{u}(k+j|k)$ in inequality (8.54). This gives the inequality

$$\hat{x}(k+j|k)^T \left[\{A(k+j) + B(k+j)K_k\}^T P \{A(k+j) + B(k+j)K_k\} \right. \\ \left. - P + K_k^T R K_k + Q_1 \right] \hat{x}(k+j|k) \leq 0 \quad (8.68)$$

which will be true for all $j \geq 0$ if

$$\{A(k+j) + B(k+j)K_k\}^T P \{A(k+j) + B(k+j)K_k\} - P + K_k^T R K_k + Q_1 \leq 0 \quad (8.69)$$

This inequality turns out to be another *LMI*. It can be pre- and post-multiplied by Q , since $Q > 0$, and then the substitution $P = \gamma Q^{-1}$ made. Defining $Y = K_k Q$ then gives the inequality

$$Q - \{A(k+j)Q + B(k+j)Y\}^T Q^{-1} \{A(k+j)Q + B(k+j)Y\} \\ - \frac{1}{\gamma} Q Q_1 Q - \frac{1}{\gamma} Y^T R Y \geq 0 \quad (8.70)$$

which, using (8.61), is the same as the *LMI* (with variables γ, Q, Y):

$$\begin{bmatrix} Q & 0 & 0 & A(k+j)Q + B(k+j)Y \\ 0 & \gamma I & 0 & Q^{1/2}Q \\ 0 & 0 & \gamma I & R^{1/2}Y \\ QA(k+j)^T + Y^T B(k+j)^T & QQ_1^{T/2} & Y^T R^{T/2} & Q \end{bmatrix} \geq 0 \quad (8.71)$$

Now suppose that $[A(k+j), B(k+j)]$ belongs to the convex polytope $\Omega = [A, B]$ defined by (8.11). Since the *LMI* (8.71) is affine in $A(k+j)$ and $B(k+j)$, it is satisfied for every $[A(k+j), B(k+j)] \in \Omega$ if and only if it is satisfied at its ‘corners’, namely for every pair (A_i, B_i) . So we need the following set of *LMIs* to be satisfied:

$$\begin{bmatrix} Q & 0 & 0 & A_i Q + B_i Y \\ 0 & \gamma I & 0 & Q^{1/2}Q \\ 0 & 0 & \gamma I & R^{1/2}Y \\ QA_i^T + Y^T B_i^T & QQ_1^{T/2} & Y^T R^{T/2} & Q \end{bmatrix} \geq 0 \quad \text{for every } i = 1, \dots, L \quad (8.72)$$

So finally we see that we obtain a robust solution to the predictive control problem without constraints by solving the convex problem:

$$\min_{\gamma, Q, Y} \gamma \quad \text{subject to (8.67) and (8.72)} \quad (8.73)$$

and, if it has a solution, setting $K_k = YQ^{-1}$ and $\hat{u}(k+j|k) = K_k \hat{x}(k+j|k)$. As remarked earlier, this solution assumes that the state vector is measured, because $x(k|k) = x(k)$ appears as data in (8.67).

Note that the state-feedback matrix K_k may change with k , since it depends on $x(k)$, even though we have so far considered the unconstrained problem. This is a distinct difference from the ‘standard’ formulation, in which the model uncertainty is not considered. In that case, of course, we get a time-invariant feedback law.

If the uncertainty description arises from using a norm-bounded Δ model, then a robust feedback law can be found by solving a slightly different *LMI* problem [KBM96].

8.4.3 Robustness with constraints

Input and output constraints can be included in the robust predictive control formulation, by adding further *LMIs*. Thus the nature of the solution algorithm does not change, although the complexity of the problem increases, of course. In this section we again assume that $[A(k+j), B(k+j)]$ belongs to the convex polytope $\Omega = [A, B]$ defined by (8.11). Similar results for a norm-bounded uncertainty model are available in [KBM96].

If problem (8.73) is solved and the corresponding control law is assumed to be applied, then (8.54) holds, and hence, if $\|\hat{x}(k+j|k)\|_P \leq \rho$ then $\|\hat{x}(k+j+1|k)\|_P \leq \rho$. Hence if the current state $x(k|k)$ satisfies $\|x(k|k)\|_P \leq \rho$ then the ‘currently predicted’ state $\hat{x}(k+j|k)$ satisfies $\|\hat{x}(k+j|k)\|_P \leq \rho$. So the ellipsoid $\mathcal{E}_\rho = \{z : \|z\|_P \leq \rho\}$, where $\rho = \|x(k|k)\|_P$, is an invariant set for the predicted state. Note that this holds for all predicted states which might arise from the uncertain plant model.

Now suppose that we have symmetric constraints on the levels of the input signals:

$$|\hat{u}_i(k+j|k)| \leq u_i \quad (8.74)$$

We can get a (probably conservative) sufficient condition for this as follows:

$$\max_j |\hat{u}_i(k+j|k)|^2 = \max_j \left| \left(YQ^{-1}\hat{x}(k+j|k) \right)_i \right|^2 \quad (8.75)$$

$$\leq \max_{v \in \mathcal{E}_\rho} \left| \left(YQ^{-1}v \right)_i \right|^2 \quad (8.76)$$

$$\leq \rho^2 \left\| \left(YQ^{-1/2} \right)_i \right\|_2^2 \quad (8.77)$$

$$= \rho^2 \left(YQ^{-1}Y^T \right)_{ii} \quad (8.78)$$

where $(YQ^{-1/2})_i$ denotes the i th row of $YQ^{-1/2}$, and (8.77) is a consequence of the Cauchy–Schwarz inequality.⁸ Now, using the fact that $A \geq B \Rightarrow A_{ii} \geq B_{ii}$, we can be

⁸ For real vectors, $|a^T b| \leq \|a\|_2 \|b\|_2$.

sure that this is satisfied if there exists $X = X^T$, such that

$$\begin{bmatrix} X & \rho Y \\ \rho Y^T & Q \end{bmatrix} \geq 0 \quad \text{with} \quad X_{ii} \leq u_i^2, \quad i = 1, \dots, m. \quad (8.79)$$

This condition is an *LMI* in the variables X , Y and Q .

Suppose that we have symmetric output constraints:

$$|\hat{z}_i(k+j|k)| \leq z_i \quad (8.80)$$

Again we can get a (probably conservative) condition for this:

$$|\hat{z}_i(k+j|k)| \leq \max_{j \geq 0} |C_i \hat{x}(k+j|k)| \quad (8.81)$$

where we use C_i here to denote the i th row of C_z . Hence

$$|\hat{z}_i(k+j|k)| \leq \max_{v \in \mathcal{E}_\rho} |C_i[A(k+j) + B(k+j)K_k]v| \quad (8.82)$$

$$= \rho \sqrt{\|C_i[A(k+j) + B(k+j)K_k]Q^{1/2}\|_2^2} \quad (8.83)$$

So the constraint $|\hat{z}_i(k+j|k)| \leq z_i$ will be satisfied if $\hat{x}(k|k) \in \mathcal{E}_\rho$ and

$$\rho \sqrt{\|C_i[A(k+j) + B(k+j)K_k]Q^{1/2}\|_2^2} \leq z_i. \quad (8.84)$$

But this is equivalent to

$$\begin{bmatrix} Q & \rho[A(k+j)Q + B(k+j)Y]^T C_i^T \\ \rho C_i[A(k+j)Q + B(k+j)Y] & z_i^2 \end{bmatrix} \geq 0 \quad \text{for each } j. \quad (8.85)$$

Since this is affine in $A(k+j)$ and $B(k+j)$, this inequality will be satisfied for every $[A(k+j), B(k+j)] \in \Omega$ if

$$\begin{bmatrix} Q & \rho[A_j Q + B_j Y]^T C_i^T \\ \rho C_i[A_j Q + B_j Y] & z_i^2 \end{bmatrix} \geq 0 \quad \text{for } j = 1, \dots, L. \quad (8.86)$$

Again, this is a set of *LMIs*.

Of course simultaneous satisfaction of all the input and output constraints is ensured by simultaneous satisfaction of the *LMIs* (8.79) and (8.86).

In [KBM96] it is also shown that ‘energy’ constraints on $\|\hat{u}(k+j|k)\|_2$ and $\|\hat{z}(k+j|k)\|_2$ can be satisfied by the satisfaction of similar *LMIs*.

The feedback control law obtained by minimizing γ over the variables γ , Q , X , Y while satisfying the *LMIs* (8.67), (8.72), (8.79) and (8.86) is stable (over all plants in the specified uncertainty set Ω), providing that it is feasible. The argument is essentially the same as that used in Section 6.2.1. Let $V(x(k|k)) = x(k|k)^T P_k x(k|k)$, where $P_k = \gamma_k Q_k^{-1}$ and γ_k , Q_k are the values of γ and Q obtained by solving the *LMI*

problem at time k . If we assume, as usual in stability proofs, that no new disturbances occur after time k , then

$$\max_{[A,B] \in \Omega} x(k+1|k+1)^T P_k x(k+1|k+1) = \max_{[A,B] \in \Omega} \hat{x}(k+1|k)^T P_k \hat{x}(k+1|k).$$

But $\hat{x}(k+1|k) \in \mathcal{E}_{V(x(k|k))}$, so

$$\max_{[A,B] \in \Omega} x(k+1|k+1)^T P_k x(k+1|k+1) \leq V(x(k|k)).$$

Furthermore, the solution obtained at time k remains feasible at time $k+1$ — see Exercise 8.8. Therefore the solution obtained at time $k+1$ will be at least as good as that obtained at time k , and hence

$$\begin{aligned} V(x(k+1|k+1)) &= x(k+1|k+1)^T P_{k+1} x(k+1|k+1) \leq \\ &x(k+1|k+1)^T P_k x(k+1|k+1) \leq V(x(k|k)). \end{aligned} \quad (8.87)$$

Thus the function $V(x(k|k))$ is a Lyapunov function for the uncertain system, and hence the closed loop is robustly stable.

8.5

Robust feasibility

The use of soft constraints greatly ameliorates the problem of unexpectedly running into infeasibility, but the problem of avoiding constraint violations remains a very important one. If a system is frequently violating the specified constraints then it is clearly not working as intended, even though constraint softening may be keeping the control algorithm running. Constraints cannot always be softened. Input constraints, for instance, usually cannot be softened. Also, recall that in Section 6.2.3 there was a constraint that the unstable modes of the plant should be brought to equilibrium by the end of the control horizon; this constraint cannot be softened if that particular predictive control algorithm is used in order to guarantee stability. So the possibility of softening constraints certainly does not remove the necessity of considering the question of feasibility.

Although infeasibility can occur even if there is no uncertainty about the plant model or about disturbances, it arises most commonly as a result of such uncertainty, or of mismodelling — for example, a mismatch between the linearized internal model and a nonlinear plant. Therefore the essential problem facing the user of predictive control is that of *robust feasibility*: designing the predictive controller so that it maintains feasibility despite uncertainty about the plant being controlled, and its environment. It bears repeating that the question of robust feasibility is also closely connected with the question of *robust stability* when constraints are present. The reason for this is that most stability proofs for predictive control rely on the assumption that the on-line optimization problem is solved correctly at each step, and this implicitly assumes that the problem is feasible at each step. In fact, a variation of predictive control in which the horizon is not fixed, but is one of the variables chosen by the optimizer, was proposed by Michalska and Mayne [MM93]; the robust stability of this scheme relies only on

having a feasible solution at each step, rather than an optimal solution [May96]. (Also see Section 10.3 and [SMR99].)

8.5.1 Maximal output admissible sets

In Section 8.4 one approach to ensuring robust feasibility was described. A major problem with that approach is that the state is forced to remain within an invariant set whose nature is chosen to fit with the mathematical framework being used, rather than matching the real constraints. For example, in Section 8.4.3 the state was confined to the ellipsoid $\mathcal{E}_\rho = \{z : \|z\|_P \leq \rho\}$, which could be a much smaller set than that implied by the real constraints. This makes the LMI approach very conservative; indeed, it may lead one to conclude wrongly that no robust solution exists.

The key to reducing such conservativeness is to estimate the set within which the state must be confined as accurately as possible. A very important contribution to this was made by Gilbert and Tan [GT91], who introduced the notion of *maximal output admissible sets* and showed how they could be computed. If an autonomous (no input) time-invariant linear system $x(k+1) = Fx(k)$, $y(k) = Hx(k)$ has to satisfy an output constraint $y(k) \in Y$ for each k , then a set \mathcal{O} is *output admissible* if $x(0) \in \mathcal{O}$ implies that $y(k) \in Y$ for all $k > 0$, which is equivalent to $HF^k x(0) \in Y$ for all $k > 0$. The maximal output admissible set \mathcal{O}_∞ is the largest such set, namely the set of all initial conditions $x(0)$ for which $HF^k x(0) \in Y$ holds for all $k > 0$. Note that the set Y is assumed to be constant, which is not always the case in predictive control; the definitions could be generalized to allow for Y changing with k . A set X is *positively invariant* if $x(0) \in X$ implies that $x(k) \in X$ for all $k > 0$. In general, output admissible sets are not necessarily positively invariant, but the *maximal* output admissible set is always positively invariant.

As in Section 8.4, we shall assume that the objective is to drive the system to the origin, so that $0 \in Y$ for consistency. In fact we shall assume for simplicity that 0 is in the interior of Y (rather than on the boundary), so that a steady state at the origin corresponds to all constraints being inactive. So far we have not mentioned inputs; that is because the definitions are going to be applied to closed-loop systems, in which the input depends on the state. For example, suppose that linear state feedback $u(k) = -Kx(k)$ is applied to the system $x(k+1) = Ax(k) + Bu(k)$, $z(k) = Cx(k)$, with constraints $u(k) \in U$ and $z(k) \in Z$. Then we have

$$x(k+1) = (A - BK)x(k) = Fx(k) \quad (8.88)$$

and the constraints $-Kx(k) \in U$, $Cx(k) \in Z$, which can be written as

$$y(k) = Hx(k) = \begin{bmatrix} -K \\ C \end{bmatrix} x(k) \in Y \quad (8.89)$$

In predictive control the constraints are usually defined by linear inequalities, and these include bounds on input and output amplitudes. This results in Y being a convex polytope. In [GT91] it is shown that this results in \mathcal{O}_∞ being convex, bounded (provided that (F, H) is an observable pair), and containing the origin in its interior (provided that F is stable, possibly including simple eigenvalues on the unit circle, such as those

due to integrations). Furthermore, \mathcal{O}_∞ scales with Y : if Y is scaled by a factor α , then \mathcal{O}_∞ is scaled by the same factor, without changing shape.

The set \mathcal{O}_t is defined as the set of initial conditions $x(0)$ such that the output satisfies the constraints for t steps, namely such that $H F^k x(0) \in Y$ for $k = 1, 2, \dots, t$. Clearly $\mathcal{O}_{t_2} \subset \mathcal{O}_{t_1}$ if $t_2 > t_1$. Now it may happen that the sequence of sets \mathcal{O}_t ($t = 1, 2, \dots$) stops getting smaller as t increases. That is, there may be a value t^* , such that $\mathcal{O}_t = \mathcal{O}_{t^*}$ for all $t > t^*$. In that case it is clear that $\mathcal{O}_\infty = \mathcal{O}_{t^*}$, and \mathcal{O}_∞ is said to be *finitely determined*. The maximal output admissible set \mathcal{O}_∞ can be computed relatively easily if and only if it is finitely determined. An important theorem states that \mathcal{O}_∞ is finitely determined if and only if $\mathcal{O}_t = \mathcal{O}_{t+1}$ for some t . In [GT91] it is shown that if F is asymptotically stable and (F, H) is an observable pair, and if Y is bounded with 0 in its interior, then \mathcal{O}_∞ is finitely determined. It is also shown that if the stability assumption is relaxed to allow simple eigenvalues on the unit circle then \mathcal{O}_∞ can be approximated arbitrarily closely, and an algorithm for doing so is given.

If Y is a convex polytope, then the following is an algorithm for computing \mathcal{O}_∞ . Suppose Y is defined by the linear inequalities $Py \leq p$, with the matrix P having s rows. Fix $t > 0$. For each $i = 1, 2, \dots, s$ maximize $J_i(x) = P_i H F^{t+1} x$ over x , subject to $P_j H F^k x \leq p_j$ for $j \neq i$ and $k = 0, 1, \dots, t$, where P_i denotes the i th row of P and p_i denotes the i th element of the vector p . Let J_i^* denote the maximized value of $J_i(x)$. If $J_i^* \leq p_i$ for each i then stop. Otherwise increase t and repeat. If \mathcal{O}_∞ is finitely determined then this algorithm will terminate for some value $t = t^*$, and

$$\mathcal{O}_\infty = \{x : PHF^t x \leq p, \quad 0 \leq t \leq t^*\} \quad (8.90)$$

The maximizations required in this algorithm are linear programming problems, so they can be solved efficiently and without difficulty.⁹

8.5.2 Robust admissible and invariant sets

In order to apply these ideas to predictive control, three further ideas have to be introduced. Firstly, suppose that an unmeasured disturbance w acts on the system, so that

$$x(k+1) = Fx(k) + w(k) \quad (8.91)$$

If it is known that $w(k) \in W$, where W is some bounded set, then one can define the robust maximal output admissible set as the set of all current states $x(k)$, such that $Hx(k+j) \in Y$ for all $j > 0$ for all possible $w(k+i)$, $i = 0, 1, \dots, j-1$. This is the same as the set of states $x(k)$ such that

$$HF^j x(k) + H \sum_{i=1}^j F^{i-1} w(k+j-i) \in Y \quad (8.92)$$

If Y and W are convex polytopes then this robust maximal output admissible set can be computed using similar methods to those outlined above [DH99, GK95].

⁹ If Y is not a polytope then a similar algorithm, with Y defined by more general inequalities, works conceptually, but finding the global maximum of $J_i(x)$ may not be possible, even if Y is convex.

Secondly, in the predictive control context F is not fixed. If linear state feedback is used then $F = A - BK$, and there is some freedom in choosing K . More generally, the control sequence $u(k+i)$ can be chosen, not necessarily according to a linear law. For this reason, the notion of a maximal (A, B) -invariant admissible set was introduced in [DH99]. This is the set of current states $x(k)$, such that a sequence of future inputs $u(k+j)$ exists for which $y(k+j) \in Y$ for all $j > 0$ (and for all $w(k+j) \in W$, if there are disturbances). Given a current state $x(k)$ and a disturbance set W , one does not know, in general, whether $x(k)$ is in this maximal (A, B) -invariant set. Several authors [CM87, GN93, ZM95b] have suggested that in such problems the future inputs $u(k+j)$ should be determined by solving a min-max problem, of the type

$$\min_{u(k+j) \in U} \max_{w(k+j) \in W} \sum_{j=1}^N L(x(k+j), u(k+j-1)) \quad (8.93)$$

subject to

$$Hx(k+j) \in Y$$

where $L(., .)$ denotes some suitable cost function — usually quadratic, but various proposals have been made. If this problem is solved successfully then one not only discovers that $x(k)$ is in the maximal (A, B) -invariant set, but one also discovers a suitable future input trajectory which maintains $y(k+j) \in Y$.

This is essentially an ‘open-loop’ approach, in which the disturbance-attenuating effects of future feedback action are not taken account of. It therefore tends to be excessively conservative: the optimization may fail, even if $x(k)$ is in the maximal (A, B) -invariant set. For this reason, proposals for closed-loop min-max strategies have been advanced by Lee and Yu [LY97] and Scokaert and Mayne [SM98]. Adopting the notation of [SM98], let the set of possible future disturbance trajectories be

$$w_{k+j|k}^\ell \in W, \quad j > 0 \quad (8.94)$$

Then the optimization (8.93) is replaced by

$$\min_{u_{k+j|k}^\ell \in U} \max_\ell \sum_{j=1}^N L(x_{k+j|k}^\ell, u_{k+j-1|k}^\ell) \quad (8.95)$$

subject to

$$Hx_{k+j|k}^\ell \in Y \quad (8.96)$$

$$x_{k+N|k}^\ell \in X_0 \quad (8.97)$$

$$x_{k+j|k}^{\ell_1} = x_{k+j|k}^{\ell_2} \Rightarrow u_{k+j|k}^{\ell_1} = u_{k+j|k}^{\ell_2} \quad (8.98)$$

The main point here is that the optimization is performed over a whole set of possible input trajectories, rather than selecting a single trajectory. $x_{k+j|k}^\ell$ and $u_{k+j|k}^\ell$ denote the state and input trajectories which correspond to the disturbance trajectory $w_{k+j|k}^\ell$, and (8.98) is a *causality constraint* which prevents the optimization from ‘cheating’.

by assuming knowledge of future disturbances — that is, the value of the disturbance w_{k+j+i} which occurs at time $k + j + i$, for $i > 0$, is not allowed to influence the value of $u_{k+j|k}$.

The mathematical notation here can disguise the radical difference between the closed-loop, or feedback, optimization (8.95) and the open-loop optimization (8.93). As in the LMI approach of [KBM96] described in Section 8.4, the proposal here is to optimize over *feedback policies*¹⁰ rather than over input trajectories. The difference is perhaps best illustrated by an example, which is taken from [SM98].

Example 8.1

The plant equation is

$$x(k+1) = x(k) + u(k) + w(k) \quad (8.99)$$

where x, u and w are scalars. The disturbance w is bounded by $|w(k)| \leq 1$, and the state constraint $|x(k)| \leq 1.2$ is imposed. Suppose that the prediction horizon is $H_p = 3$. The prediction of the state $x(k+3)$ (assuming the state $x(k)$ is known) is given by

$$\begin{aligned} \hat{x}(k+3|k) = & x(k) + \hat{u}(k|k) + \hat{u}(k+1|k) + \hat{u}(k+2|k) \\ & + \hat{w}(k|k) + \hat{w}(k+1|k) + \hat{w}(k+2|k) \end{aligned} \quad (8.100)$$

Over the 3-step horizon there are 8 ($= 2^3$) possible realizations of the disturbance sequence in which the disturbances take their extreme values ± 1 at each step. Two of these possible realizations are

$$w_{k|k}^1 = w_{k+1|k}^1 = w_{k+2|k}^1 = -1 \quad (8.101)$$

and

$$w_{k|k}^8 = w_{k+1|k}^8 = w_{k+2|k}^8 = +1 \quad (8.102)$$

The corresponding state predictions are

$$\hat{x}_{k+3|k}^1 = x(k) + \hat{u}(k|k) + \hat{u}(k+1|k) + \hat{u}(k+2|k) - 3 \quad (8.103)$$

and

$$\hat{x}_{k+3|k}^8 = x(k) + \hat{u}(k|k) + \hat{u}(k+1|k) + \hat{u}(k+2|k) + 3 \quad (8.104)$$

Now to perform the open-loop optimization (8.93) one has to find a single sequence of input signals, $\hat{u}(k|k), \hat{u}(k+1|k), \hat{u}(k+2|k)$ which will satisfy the state constraints $|\hat{x}_{k+3|k}^1| \leq 1.2$ and $|\hat{x}_{k+3|k}^8| \leq 1.2$. Clearly this is impossible, since $\hat{x}_{k+3|k}^8 - \hat{x}_{k+3|k}^1 > 2 \times 1.2$. So in this case the optimization would fail, since the problem is infeasible.

¹⁰ A feedback policy is a *function* from the state-space (or alternatively from the space of input–output measurement histories) to input signals; that is, a *rule* which assigns input signals on the basis of state values (or input–output histories).

For the closed-loop optimization (8.95), however, one can look for a different sequence of input signals for each different realization of the disturbance sequence. The space W of possible disturbance sequences,

$$W = \{w_{k+i|k}^\ell : i = 0, 1, 2, |w_{k+i|k}^\ell| \leq 1\} \quad (8.105)$$

is a convex polyhedron with eight vertices:

$$(w_{k|k}^1, w_{k+1|k}^1, w_{k+2|k}^1) = (-1, -1, -1) \quad (8.106)$$

$$(w_{k|k}^2, w_{k+1|k}^2, w_{k+2|k}^2) = (+1, -1, -1) \quad (8.107)$$

$$(w_{k|k}^3, w_{k+1|k}^3, w_{k+2|k}^3) = (-1, +1, -1) \quad (8.108)$$

$$(w_{k|k}^4, w_{k+1|k}^4, w_{k+2|k}^4) = (+1, +1, -1) \quad (8.109)$$

$$(w_{k|k}^5, w_{k+1|k}^5, w_{k+2|k}^5) = (-1, -1, +1) \quad (8.110)$$

$$(w_{k|k}^6, w_{k+1|k}^6, w_{k+2|k}^6) = (+1, -1, +1) \quad (8.111)$$

$$(w_{k|k}^7, w_{k+1|k}^7, w_{k+2|k}^7) = (-1, +1, +1) \quad (8.112)$$

$$(w_{k|k}^8, w_{k+1|k}^8, w_{k+2|k}^8) = (+1, +1, +1) \quad (8.113)$$

Since the plant model is linear, these eight disturbance sequences are the worst cases, as regards feasibility. If the control problem is feasible for each of these sequences, it will be feasible for any other disturbance sequence in W . Now consider the following four input sequences:

$$(u_{k|k}^1, u_{k+1|k}^1, u_{k+2|k}^1) = (-x(k), 1, 1) \quad (8.114)$$

$$(u_{k|k}^2, u_{k+1|k}^2, u_{k+2|k}^2) = (-x(k), -1, +1) \quad (8.115)$$

$$(u_{k|k}^3, u_{k+1|k}^3, u_{k+2|k}^3) = (-x(k), 1, -1) \quad (8.116)$$

$$(u_{k|k}^4, u_{k+1|k}^4, u_{k+2|k}^4) = (-x(k), -1, -1) \quad (8.117)$$

Since $w_{k+i|k}^5 = w_{k+i|k}^1$ for $i = 0, 1$, and the causality constraint (8.98) holds, we get $x_{k+i|k}^5 = x_{k+i|k}^1$ for $i = 1, 2$. This leads to $u_{k+i|k}^5 = u_{k+i|k}^1$, and similarly we get $u_{k+i|k}^6 = u_{k+i|k}^2$, $u_{k+i|k}^7 = u_{k+i|k}^3$, and $u_{k+i|k}^8 = u_{k+i|k}^4$ for $i = 0, 1, 2$. This gives the state predictions:

$$x_{k+i+1|k}^\ell = w_{k+i|k}^\ell = \pm 1 \quad \ell = 1, 2, \dots, 8, \quad i = 0, 1, 2 \quad (8.118)$$

Thus all the predicted states satisfy the constraint $|x(k)| \leq 1.2$, so the problem is feasible even if worst-case disturbances occur, providing that the search is over feedback policies.

In [SM98] a dual-mode control law is assumed, of the type introduced in [MM93]; a linear state-feedback gain $-K$ is used if the state is in a ‘terminal constraint set’ X_0 , and the terminal constraint (8.97) insists that the state should be driven into this terminal set in N steps. (Driving the state to a single point such as the origin is not an achievable objective when persistent disturbances are present.) The set X_0 must be such that once

a state is in X_0 , the linear feedback law can keep it in X_0 , despite the disturbances. That is, the set inclusion

$$FX_0 + W = (A - BK)X_0 + W \subset W \quad (8.119)$$

must hold, and for any state in X_0 all the constraints must be satisfied. Such a set is called *robust control invariant*. If all the eigenvalues of $A - BK$ are at the origin, so that $F^s = (A - BK)^s = 0$ for some s , then a suitable candidate for X_0 is the set $W + FW + \dots + F^{s-1}W$. Although the optimization must be performed over infinitely many possible disturbance trajectories, it is pointed out in [SM98] that if W is convex then with a linear model the search is over a convex polytope, and it is sufficient to search over the (finitely many) vertices of this polytope. On each vertex a convex optimization is required, the exact nature of which depends on the cost function $L(\cdot)$ in (8.95) — usually one would envisage an LP or QP being needed here. Nevertheless, the number of vertices increases exponentially with the horizon length, so closed-loop min-max is computationally a relatively demanding procedure.

Example 8.2

In Example 8.1 we established feasibility, but the particular input signals considered there were not necessarily optimal. If the cost function $L(\cdot, \cdot)$ is convex then the optimal solution to the min-max problem (8.95) will occur for one of the eight worst-case disturbance sequences considered in that example, so eight convex optimizations need to be performed to find the optimal solution.

In [Bem98] an alternative, compromise, proposal is made, namely that the optimization (8.93) should be performed, but that a fixed stabilizing state-feedback gain K should be assumed. This allows relatively simple determination of whether $x(k)$ is in the maximal (A, B) -invariant set, but it retains some of the conservativeness of the open-loop approach, so there is a risk of falsely concluding that it is not. If it is found that $x(k)$ is in the maximal (A, B) -invariant set then determination of a future input trajectory can proceed by optimizing deviations from the control law $u(k+j) = -Kx(k+j)$. This is in effect identical to the use of ‘stabilized predictions’, which we discussed in Section 5.2. Roughly speaking, making $A - BK$ ‘more stable’, in particular placing its eigenvalues closer to 0, gives a better chance of concluding that a feasible solution exists, if $x(k)$ is in the maximal (A, B) -invariant set. (If all the eigenvalues are at zero, so that $(A - BK)^s = 0$ for some s , then the maximal (A, B) -invariant set depends on only a finite portion (of length s) of the disturbance trajectory.) An example in [Bem98] shows that, if one uses ‘open-loop’ predictions, namely $K = 0$, $F = A$, then one may falsely conclude that no feasible trajectory exists, because the uncertainty about the disturbance accumulates with time, and can result in an empty output admissible set being indicated. This is most likely to happen if A has eigenvalues close to the unit circle, namely slowly decaying dynamics. A further proposal is made in [KRSar]. This uses stabilized predictions again, but chooses K for (possibly robust) performance of the unconstrained system, then chooses deviations from this fixed control law in order to maximize the volume of a positively-invariant ellipsoid which is output admissible. The computations required for this are based on LMIs.

The third idea that is needed relates to horizon lengths. Infinite-horizon predictive control formulations require the constraints to be enforced over a finite horizon only. In general, the minimum length of that horizon, which ensures that the constraints will not be active beyond the horizon, depends on the current state — recall Section 6.2. Let t^* be the smallest t , such that $\mathcal{O}_t = \mathcal{O}_{t+1}$, and hence that $\mathcal{O}_\infty = \mathcal{O}_{t^*}$. t^* indicates the length of the horizon over which the constraints must be enforced, beyond the end of the control horizon. That is, one should choose $H_p - H_u > t^*$. (In this case, F , and hence \mathcal{O}_{t^*} , depends on the linear law implemented by the predictive controller when the constraints are inactive.)

The literature on the concepts discussed in this section is growing rapidly. Robust feasibility is an active current research area. It addresses a most important problem for predictive control, but it is probably fair to say that the solutions advanced to date are still some way from implementation in real applications. Some possibilities of what to do if one finds that infeasibility is unavoidable are discussed in Section 10.2. Most of the concepts introduced in this section can be applied to nonlinear systems and arbitrary (non-convex) sets [KM00a], but computation of the relevant sets then becomes very difficult, in general.

Exercises

- 8.1** Suppose an ‘output multiplicative’ uncertainty description of the plant is assumed:

$$P(z) = [I + \Delta]P_0(z) \quad (8.120)$$

where Δ is a stable operator, and the controller $K(z)$ stabilizes the nominal plant $P_0(z)$. Show that the feedback connection of the controller with the actual plant $P(z)$ can be represented as the feedback connection of the uncertainty Δ with the complementary sensitivity function $T(z) = P_0(z)K(z)[I + P_0(z)K(z)]^{-1}$ (see Section 7.1). Hence infer that a sufficient condition for robust stability is

$$\bar{\sigma}[T(e^{j\omega T_s})]\|\Delta\|_\infty < 1 \quad (8.121)$$

- 8.2** Show that the observer poles obtained from using the simplified disturbance/noise model of Section 8.2 are the plant poles, together with the eigenvalues of the matrices

$$\begin{bmatrix} \alpha_i & -\phi_i \\ \alpha_i & 1 - \psi_i \end{bmatrix} \quad (8.122)$$

for $i = 1, 2, \dots, p$.

- 8.3** Show that, if the plant model (8.41)–(8.42) is used, and $\text{cov}\{v(k)\} = V = 0$, then $L_\eta = I$, where the Kalman filter gain is partitioned as

$$L_\infty = \begin{bmatrix} L_{\Delta x} \\ L_{\Delta w} \\ L_\eta \end{bmatrix}$$

- 8.4** Show that any inequality of the form $F(v) \geq 0$, in which $F(v)$ is *symmetric* and *affine* in the vector v , is an *LMI*.
- 8.5** (a) Verify that the minimization problem (8.65) corresponds to the QP problem (8.63).
 (b) Verify that (8.65) is in the form of (8.59) — use Exercise 8.4.
- 8.6** Show that the standard LP problem: $\min_x c^T x$ subject to $Ax \geq b$ is equivalent to a problem in the form of (8.59).
- 8.7** Show that the inequality (8.84) is equivalent to the *LMI* (8.85).
- 8.8** Supply the missing step in the proof of robust stability in Section 8.4.3: note that the only *LMI* which depends explicitly on the state measurement $x(k|k)$ is (8.67). Use the fact that $x(k+1|k+1)$ is obtained by the state feedback law K_k , and that $\mathcal{E}_{V(x(k|k))}$ is an invariant set for $x(k+j|k+j)$ (in the absence of disturbances), to show that

$$\begin{bmatrix} Q_k & x(k+1|k+1) \\ x(k+1|k+1)^T & 1 \end{bmatrix} \geq 0 \quad (8.123)$$

and hence that the solution obtained at time k remains feasible at time $k+1$.

- 8.9** Prove that the maximal output admissible set is positively invariant (unless it is empty).

Two case studies

9.1	Shell oil fractionator	248
9.2	Newell and Lee evaporator	269
	Exercises	274

9.1 Shell oil fractionator

9.1.1 Process description

In this section we shall consider the well-known ‘Shell’ heavy oil fractionator (distillation column) problem, as defined in [PM87]. This is a linear model which does not represent a real, existing process, but was devised to exhibit the most significant control engineering features faced on real oil fractionators. It was intended to serve as a standard problem on which various control solutions could be demonstrated (in simulation). We shall simplify the problem definition of [PM87] slightly, and adopt the problem definition assumed in [PG88].

The fractionator is shown in Figure 9.1. A gaseous feed enters at the bottom of the fractionator. Three product streams are drawn off, at the top, side and bottom of the fractionator (shown on the right-hand side in the figure). There are also three circulating (*reflux*) loops at the top, middle (subsequently referred to as *intermediate*) and bottom of the fractionator, which are shown on the left-hand side of the figure. Heat is carried into the fractionator by the feed, and these circulating reflux loops are used to remove heat from the process, by means of heat exchangers. The heat removed in this way is used to supply heat to other processes, such as other fractionators. The amount of heat removed by each reflux loop is expressed as *heat duty*; the larger the duty, the more heat is recirculated back into the fractionator, and thus the smaller the amount of heat removed. The gains from heat duties to temperatures are therefore positive. The heat removed in the top two reflux loops is determined by the requirements of other processes, and these therefore act as disturbance inputs to the fractionator. An important difference between them is that the

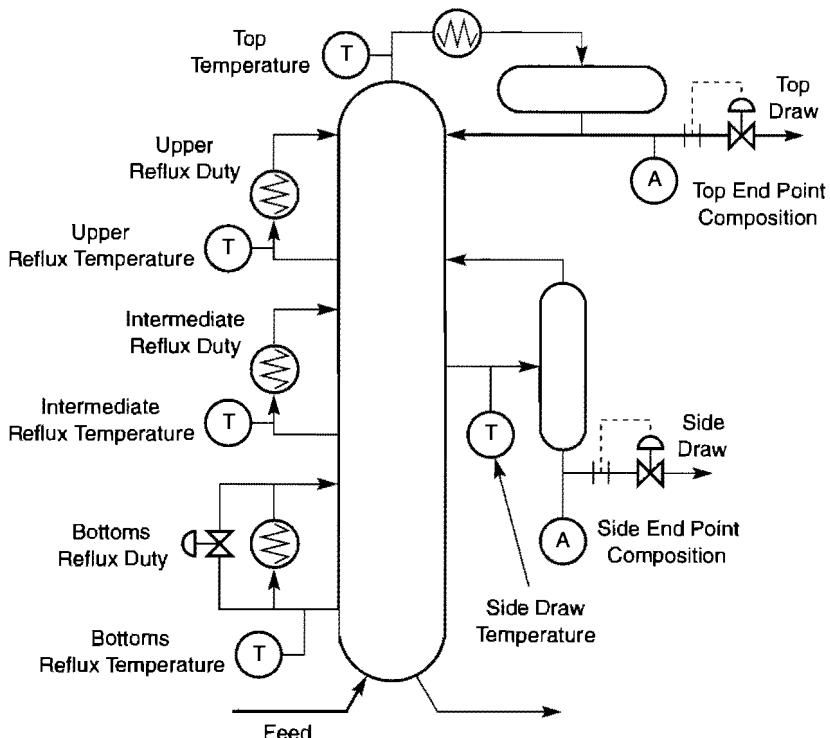


Figure 9.1 The 'Shell' heavy oil fractionator.

Intermediate Reflux Duty is considered to be measured, and is therefore available for feedforward control, whereas the Upper Reflux Duty is an unmeasured disturbance. The Bottoms Reflux Duty, by contrast, is a manipulated variable, which can be used to control the process. The Bottoms Reflux is used to generate steam for use on other units; so, although the Bottoms Reflux Duty can be used to control the oil fractionator, keeping it as low as possible corresponds to generating as much steam as possible, and is therefore economically advantageous. There are two additional manipulated variables (that is, control inputs): the Top Draw and the Side Draw, namely the flow rates of the products drawn off at the top and side of the fractionator. Again, these are defined in such a way that a positive change leads to an increase of all temperatures in the fractionator. Altogether, then, there are five inputs to the fractionator, of which three are manipulated variables (control inputs) and two are disturbances.

There are seven measured outputs from the fractionator. The first two outputs are the compositions of the Top End Point product and of the Side End Point product. These are measured by analysers, and the composition is represented by a scalar variable in each case. The remaining five outputs are all temperatures: the Top Temperature, the Upper Reflux Temperature, the Side Draw Temperature, the Intermediate Reflux Temperature, and the Bottoms Reflux Temperature. Only three of these outputs are to

Table 9.1 Names, roles and symbols of input and output variables.

Variable	Role	Symbol
Top Draw	Control input	u_1
Side Draw	Control input	u_2
Bottoms Reflux Duty	Control input	u_3, z_4
Intermediate Reflux Duty	Measured disturbance	d_m
Upper Reflux Duty	Unmeasured disturbance	d_u
Top End Point	Controlled and measured output	y_1, z_1
Side End Point	Controlled and measured output	y_2, z_2
Top Temperature	Measured output	y_3
Upper Reflux Temperature	Measured output	y_4
Side Draw Temperature	Measured output	y_5
Intermediate Reflux Temperature	Measured output	y_6
Bottoms Reflux Temperature	Controlled and measured output	y_7, z_3

Table 9.2 Transfer functions from control inputs to all outputs.

	u_1	u_2	u_3
y_1, z_1	$4.05e^{-27s} \frac{1}{50s+1}$	$1.77e^{-28s} \frac{1}{60s+1}$	$5.88e^{-27s} \frac{1}{50s+1}$
y_2, z_2	$5.39e^{-18s} \frac{1}{50s+1}$	$5.72e^{-14s} \frac{1}{60s+1}$	$6.90e^{-15s} \frac{1}{40s+1}$
y_3	$3.66e^{-2s} \frac{1}{9s+1}$	$1.65e^{-20s} \frac{1}{30s+1}$	$5.53e^{-2s} \frac{1}{40s+1}$
y_4	$5.92e^{-11s} \frac{1}{12s+1}$	$2.54e^{-12s} \frac{1}{27s+1}$	$8.10e^{-2s} \frac{1}{20s+1}$
y_5	$4.13e^{-5s} \frac{1}{8s+1}$	$2.38e^{-7s} \frac{1}{19s+1}$	$6.23e^{-2s} \frac{1}{10s+1}$
y_6	$4.06e^{-8s} \frac{1}{13s+1}$	$4.18e^{-4s} \frac{1}{33s+1}$	$6.53e^{-1s} \frac{1}{9s+1}$
y_7, z_3	$4.38e^{-20s} \frac{1}{33s+1}$	$4.42e^{-22s} \frac{1}{44s+1}$	$7.20 \frac{1}{19s+1}$

be controlled: the Top End Point and Side End Point compositions, and the Bottoms Reflux temperature. The remaining four output measurements are available for use by the controller. (In the original problem definition of [PM87] the possibility is raised that the two analysers may be unreliable, in which case these additional output measurements play an important role in allowing control to continue without having the analyser measurements available.)

The transfer function from each input to output is modelled as a first-order lag with time delay. We shall order the inputs and outputs as shown in Table 9.1. The transfer functions from the control inputs (manipulated variables) to all the outputs are shown in Table 9.2. Note that all the time constants and time delays are expressed in minutes. The transfer functions from the two disturbance inputs to the outputs are shown in Table 9.3. These tables show the *nominal* transfer functions. The gain in each transfer function is uncertain, the uncertainties associated with each input being correlated with each other. Table 9.4 specifies the uncertainty in each gain. The nominal model is obtained when $\delta_j = 0$ for each j ; but it is assumed that each δ_j can vary in the range $-1 \leq \delta_j \leq 1$.

Table 9.3 Transfer functions from disturbance inputs to all outputs.

	u_1	u_2	d_m	d_u
y_1, z_1	$1.20e^{-27s} \frac{1}{45s+1}$	$1.44e^{-27s} \frac{1}{40s+1}$		
y_2, z_2	$1.52e^{-15s} \frac{1}{25s+1}$	$1.83e^{-15s} \frac{1}{20s+1}$		
y_3	$1.16 \frac{1}{11s+1}$		$1.27 \frac{1}{6s+1}$	
y_4	$1.73 \frac{1}{5s+1}$		$1.79 \frac{1}{19s+1}$	
y_5	$1.31 \frac{1}{2s+1}$		$1.26 \frac{1}{22s+1}$	
y_6	$1.19 \frac{1}{19s+1}$		$1.17 \frac{1}{24s+1}$	
y_7, z_3	$1.14 \frac{1}{27s+1}$		$1.26 \frac{1}{32s+1}$	

Table 9.4 Extent of gain uncertainty in each transfer function. $|\delta_j| \leq 1$ for each j .

	u_1	u_2	u_3	d_m	d_u
y_1, z_1	$4.05 + 2.11\delta_1$	$1.77 + 0.39\delta_2$	$5.88 + 0.59\delta_3$	$1.20 + 0.12\delta_4$	$1.44 + 0.16\delta_5$
y_2, z_2	$5.39 + 3.29\delta_1$	$5.72 + 0.57\delta_2$	$6.90 + 0.89\delta_3$	$1.52 + 0.13\delta_4$	$1.83 + 0.13\delta_5$
y_3	$3.66 + 2.29\delta_1$	$1.65 + 0.35\delta_2$	$5.53 + 0.67\delta_3$	$1.16 + 0.08\delta_4$	$1.27 + 0.08\delta_5$
y_4	$5.92 + 2.34\delta_1$	$2.54 + 0.24\delta_2$	$8.10 + 0.32\delta_3$	$1.73 + 0.02\delta_4$	$1.79 + 0.04\delta_5$
y_5	$4.13 + 1.71\delta_1$	$2.38 + 0.93\delta_2$	$6.23 + 0.30\delta_3$	$1.31 + 0.03\delta_4$	$1.26 + 0.02\delta_5$
y_6	$4.06 + 2.39\delta_1$	$4.18 + 0.35\delta_2$	$6.53 + 0.72\delta_3$	$1.19 + 0.08\delta_4$	$1.17 + 0.01\delta_5$
y_7, z_3	$4.38 + 3.11\delta_1$	$4.42 + 0.73\delta_2$	$7.20 + 1.33\delta_3$	$1.14 + 0.18\delta_4$	$1.26 + 0.18\delta_5$

Figure 9.2 shows the nominal unit step response from each input — including the two disturbance inputs, the columns being arranged in the order u_1, u_2, u_3, d_m, d_u — to each output of the fractionator. This confirms that the fractionator is asymptotically stable, and that the settling times range from about 10 minutes (for element (5,4) — from the Intermediate Reflux Duty to the Side Draw Temperature) to about 250 minutes (for element (2,2) — from the Side Draw to the Side End Point). The various time delays can also be seen clearly in the figure.

9.1.2 Control specifications

The focus of the control performance specifications is on rejecting disturbances, while minimizing the Bottoms Reflux Duty (u_3) and satisfying constraints. There are set-point specifications on the Top End Point and Side End Point compositions. Since we have a linearized model, we may as well assume that the set-points are at 0. There is a requirement that at steady state these two outputs should be within ± 0.005 of the set-point. The dynamic response requirement is more ambiguous, being that ‘closed-loop speed of response should be between 0.8 and 1.25 of the open-loop speed’. We shall assume that this refers to set-point response, rather than disturbance response; that removes most of the ambiguity because the open-loop response speed of the Top End Point and the Side End Point compositions to each of the three control inputs is

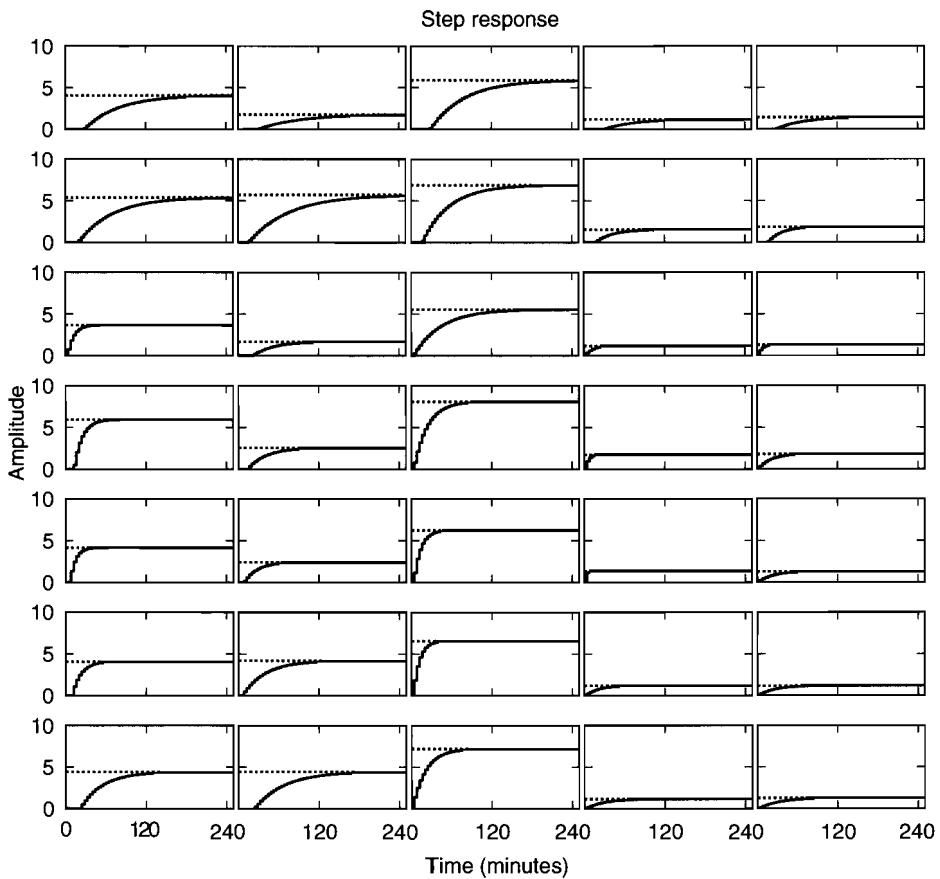


Figure 9.2 Open-loop step responses of the ‘Shell’ heavy oil fractionator.

nearly the same, with similar time delays and time constants. For disturbance rejection, the requirement is to return the Top and Side End Point compositions to their set-points as quickly as possible, subject to satisfying constraints. The trade-off between returning these compositions to their set-points and minimizing the Bottoms Reflux Duty, should these conflict, is not specified. (It is very common to have significant parts of a specification missing; not because it does not matter, but because it is too difficult to articulate in advance.) In addition there are constraints on inputs and outputs as shown in Table 9.5. The variables have been scaled so that they all have similar constraints, and the input move constraints, which are 0.05 per minute for each input, have been expressed as limits on $|\Delta u_j|$, consistently with our earlier notation, in terms of the sampling and control update interval T_s . Note that the Bottoms Reflux Temperature has constraints but no set-point, so it has a ‘zone’ objective, as discussed in Section 5.5.

The two disturbance inputs are known to take values in the range ± 0.5 , but they are otherwise unspecified.

Table 9.5 Input and output constraints for the oil fractionator. (T_s is the control update interval.)

Input move constraints:	$ \Delta u_j \leq 0.05T_s$ for $j = 1, 2, 3$
Input range constraints:	$ u_j \leq 0.5$ for $j = 1, 2, 3$
Output constraints:	$ z_j \leq 0.5$ for $j = 1, 2, 3$

9.1.3 Initial controller design

One of the objectives is to minimize the Bottoms Reflux Duty, which is one of the control inputs. The most straightforward way to do this is to make this input also an output, and make its set-point equal to its lowest permitted value, which is -0.5 . Since we require the model to be strictly proper (no direct feed-through from input to output — see Section 2.1) we define a new controlled output z_4 as

$$z_4(k) = u_3(k - 1) \quad (9.1)$$

and its set-point will be $s_4(k) = -0.5$ for all k . This approach is not ideal, however, because this set-point cannot be held in the steady state without violating the constraints, even when the model is exact and there are no disturbances. To see this, consider the matrix of steady-state gains from the control inputs to the controlled outputs:

$$\begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} 4.05 & 1.77 & 5.88 \\ 5.39 & 5.72 & 6.90 \\ 4.38 & 4.42 & 7.20 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad (9.2)$$

In order to hold the vector $[z_1, z_2, z_4]$ at the set-point vector $[0, 0, -0.5]$, the input vector would have to be $[u_1, u_2, u_3] = [0.7860, -0.1375, -0.5]$, so the Top Draw (u_1) would have to violate its upper constraint. Furthermore, this input vector would give $z_3 = -0.7651$, so the Bottoms Reflux Temperature would have to violate its lower constraint. The unsatisfactory aspect of this formulation is that the distinction between genuine set-points for z_1 and z_2 , and the constraint on z_4 , has been lost. Making the outcome reflect the real objectives may require tuning the weights in the MPC formulation, and there is no guarantee that the same weights will be suitable for various combinations of disturbances and model errors.

The real solution is to adopt one of the strategies discussed in Section 10.1.2. But here we will continue to use the simple approach of defining a set-point for $z_4 (= u_3)$.

One way of alleviating the difficulty is to find a set-point which does not lead to constraint violations at steady state. Assuming no disturbances and a perfect model, obtaining $z_1 = z_2 = 0$ requires

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 4.05 & 1.77 & 5.88 \\ 5.39 & 5.72 & 6.90 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \quad (9.3)$$

This can be attained by any vector u in the null-space of the matrix in (9.3), which is one-dimensional:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ -0.175 \\ -0.6361 \end{bmatrix} \quad (9.4)$$

Choosing $\alpha = 0.5$ gives $u = [0.5, -0.0875, -0.3181]^T$, which satisfies the input constraints, and this in turn gives $z_3 = -0.4867$, which satisfies (just!) the lower constraint on the Bottoms Reflux Temperature. So -0.3181 may be a better set-point for u_3 than -0.5 . However, there may be some combinations of disturbances, and some departures of the real fractionator gains from the modelled ones, which will allow a lower value of u_3 to be held.

The next issue to consider is the sampling interval to be used, T_s . There are three aspects to consider:

1. The impact on the closed-loop behaviour.
2. The impact on the computation time.
3. The impact on memory storage requirements.

The last of these is the least important, given modern memory capacities. Table 9.6 shows the storage requirements of z -transform transfer function models, of exact state-space models, and of step response models for various sampling times.¹ Later we shall see that approximate state-space models of considerably smaller state dimension can in fact be used, and their storage requirements are considerably smaller than those shown in the table. The transfer function models are represented as *LTI objects* in MATLAB's *Control System Toolbox*, whereas the state-space models are in the *Model Predictive Control Toolbox*'s *MOD* format, and the step response models are in the *Model Predictive Control Toolbox*'s *STEP* format. The step response model is truncated after 250 minutes (and so gets smaller as the sampling time is increased), and all variables are stored in double precision. It should be emphasized that the table shows the amount of storage needed just to represent the model, not the amount required by the controller algorithm. In each case the state-space model has had redundant modes removed, using the *Control System Toolbox*'s minimal realization function *minreal*. The dependence of the state dimension on the sampling interval is due to the fact that each delay requires a state variable to represent it, and the number of delays is reduced by increasing the sampling interval. (The state dimension includes the one additional state required because of definition (9.1).)

Since we have no information about measurement noise, and very little about the disturbances, there is not much scope for being clever with observers and disturbance models in this case. In addition the model is asymptotically stable, so all three model representations — that is, step response, transfer function, or state-space — can be used. In these circumstances, each of the three model representations could be used equally well.

¹ All these models can be created using the function `mkoilfr` which is available from this book's web site.

Table 9.6 Memory storage, in kilobytes, needed for various model representations and various sampling times. (State-space models are exact.)

Sampling interval T_s (minutes)	State dimension n	Transfer function (LTI object)	State-space model (MOD format)	Step response (STEP format)
1	363	17.7	1098.1	80.4
2	204	15.1	357.8	40.4
4	122	13.9	134.1	20.6
8	86	13.3	69.9	10.3

The impact of the sampling time on computational requirements is more significant. Closed-loop settling times in response to set-point changes are required to be in the region of 120–250 minutes (the relevant open-loop time constants range from 40 to 60 minutes; closed-loop time constants are to be between 80% and 125% of these values; take the settling time to be $3 \times$ time constant and add the time delay of between 14 and 28 minutes). Recall from Section 7.2.2 that ‘mean-level’ design gives a closed-loop set-point response speed similar to the open-loop speed, and is achieved by making the prediction horizon very large and the control horizon very small. We will not go to the extreme of $H_p = \infty$ and $H_u = 1$, but, guided by that example, we will try a relatively long prediction horizon. Since constraints are to be enforced at all times, the option of choosing a small number of coincidence points is not available, at least for constraint enforcement — it remains a possibility for penalizing tracking errors in the cost function. Choosing a small sampling interval will therefore lead to a large prediction horizon, which will be computationally demanding. On the other hand, the effect of the Intermediate Reflux Duty, which is a measured disturbance, on the Bottoms Reflux Temperature, is relatively fast, with no time delay and a time constant of 27 minutes. Therefore choosing a sampling interval as large as 10 minutes may not give enough opportunity for effective feedforward action. We shall therefore try $T_s = 4$ minutes — the same value as chosen in [PG88].

If only the Top and Side End Point compositions and the Bottoms Reflux Temperature have been retained as outputs, and the Bottoms Reflux Duty is brought out as an additional output (with a one-step delay), then a minimal state-space realization of a discrete-time model, with $T_s = 4$ minutes, has 80 states. Reducing the number of states would speed up the controller computations and reduce the risks of hitting numerical problems. A lower-order approximation can be obtained in a number of ways. One way is to simulate the pulse response, and use the algorithm presented in Section 4.1.4. Since a state-space model is already available, a related but better way is to first obtain a *balanced realization* of the 80-state system, and (from the same computation) its *Hankel singular values*. The balanced realization has the property that its state variables are ordered according to the contribution they make to the input–output behaviour, loosely speaking, and approximations can therefore be obtained by truncating the state vector, and truncating the matrices of the state-space model correspondingly. The Hankel singular values can be used as a guide to how many state variables should be truncated. (State variables which make no contribution to the input–output behaviour correspond to zero Hankel singular values.) This approach to obtaining low-order approximate

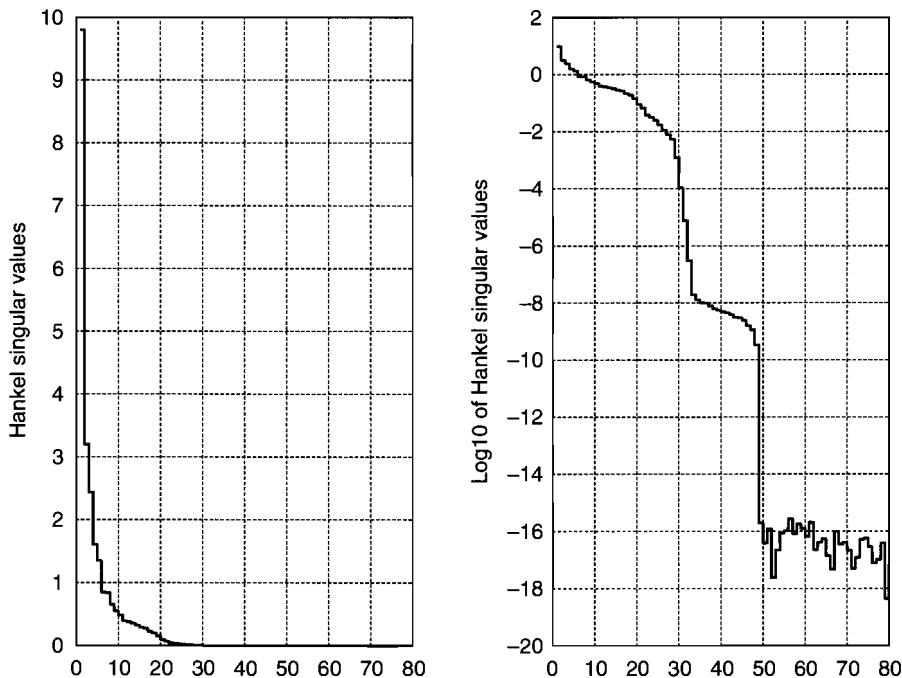


Figure 9.3 Hankel singular values of the ‘Shell’ heavy oil fractionator, with $T_s = 4$ minutes.

systems is known as *balanced truncation*, and is described in detail in [Mac89, ZDG96]. This method was applied to the 80-state oil fractionator model.²

Figure 9.3 shows the Hankel singular values of the fractionator model, when only the Top and Side End Point compositions and the Bottoms Reflux Temperature have been retained as outputs, and the Bottoms Reflux Duty has been brought out as an additional output (with a one-step delay). On the left these singular values are shown on a linear scale, while on the right their logarithm (to base 10) is shown. The logarithmic plot suggests that a ‘natural’ choice of order would be either 30 or 50, since that is where reductions of several orders of magnitude occur. But in fact reducing to only 20 states gives step responses which are almost indistinguishable from those of the full 80-state model. We shall use the 20-state approximate model for all our investigations.³

Table 9.7 shows the storage requirements for the full 80-state and the approximate 20-state models. These are seen to be much smaller than for the exact model of the dynamics of all the measured outputs. In fact the 20-state model requires about 50% less memory than the step response model.

In order to predict beyond the expected settling time — up to 300 minutes, say — the choice of sampling interval $T_s = 4$ minutes will require a prediction horizon $H_p = 75$.

² Obtaining a balanced realization proved problematic. The *Control System Toolbox* function `balreal` did not work, because of numerical finite-precision effects — the 80-state model is too close to being uncontrollable. But the *Robust Control Toolbox* function `dobal` worked without difficulty.

³ This model is available in the *MATLAB*-readable file `oilfr20.mat` on this book’s web site.

Table 9.7 Memory storage, in kilobytes, needed for the exact and approximate state-space models for outputs z_1 , z_2 and z_3 only, with $T_s = 4$ minutes.

State dimension n	Memory required (kbytes)
80	58.5
20	5.2

Because of the significant time delays in many of the transfer functions between control inputs and controlled outputs, it will be advisable not to enforce constraints over the initial part of the prediction horizon, or to soften them, or both. The largest time delay is 28 minutes, which corresponds to seven steps into the prediction horizon.

The choice of control horizon will have a large impact on the computational load on the controller algorithm. Since there is a large spread of slow and fast dynamics in the various transfer functions of the fractionator, the use of *blocking*, namely non-uniform intervals between control decisions — see Exercise 1.14 and Exercise 3.4 — seems appropriate here, as that allows control adjustments to be made throughout the predicted transient period without having too many decision variables in the optimization problem. Guided by the responses shown in Figure 9.2, we guess that it may be appropriate to keep $\hat{u}(k|k)$, $\hat{u}(k+1|k)$, $\hat{u}(k+3|k)$, $\hat{u}(k+7|k)$ and $\hat{u}(k+15|k)$ as the decision variables, so that the blocking pattern is as follows, each block being twice as long as the previous one:

$$\hat{u}(k|k) \quad (9.5)$$

$$\hat{u}(k+1|k) = \hat{u}(k+2|k) \quad (9.6)$$

$$\hat{u}(k+3|k) = \hat{u}(k+4|k) = \hat{u}(k+5|k) = \hat{u}(k+6|k) \quad (9.7)$$

$$\hat{u}(k+7|k) = \hat{u}(k+i|k) \quad \text{for } i = 8, 9, \dots, 14 \quad (9.8)$$

$$\hat{u}(k+15|k) = \hat{u}(k+i|k) \quad \text{for } i = 16, 17, \dots \quad (9.9)$$

In this way we have only 15 decision variables (3 control inputs \times 5 decision times), but control decisions are taken up to 60 minutes into the prediction horizon.

Next we must choose weights for the cost function. Since the Bottoms Reflux Temperature (z_3) has a zone objective only, the tracking error weight for it is 0. For the Top End Point and Side End Point compositions we have very little information on which to base our choice. We assume that the variables have already been scaled such that equal errors on each of these are equally important, and hence give them equal weight. We will also keep the weights constant over the prediction horizon, except that we will not weight the Top End Point errors during the first seven steps, or the Side End Point errors during the first four steps, because of the long time delays in the transfer functions involved. The Bottoms Reflux Duty (z_4) tracking error will also be given equal weight with the others, since there is no clear reason for any other choice. This weight in particular may be adjusted in the light of experience, in order to change the trade-off between tracking the Top End Point and Side End Point compositions, and

minimizing the Bottoms Reflux Duty. Thus we have

$$Q(i) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{for } i \leq 4 \quad (9.10)$$

$$Q(i) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{for } 5 \leq i \leq 7 \quad (9.11)$$

$$Q(i) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \text{for } i \geq 8 \quad (9.12)$$

No penalties will be imposed on control moves initially; the specification does not require input moves to be minimized. If the response is later found to be too erratic, or too sensitive to model uncertainty, then this decision will be reviewed. Thus we have $R(i) = 0_{3,3}$ for each i .

9.1.4 Controller performance and refinement

Figure 9.4 shows the response with these tuning parameters, when the set-points for z_1 and z_2 are both 0, the set-point for u_3 is initially -0.32 , and is changed to -0.5 after 300 minutes. This response was obtained using the *Model Predictive Control Toolbox* function `smpc`; the 80-state model was taken as the plant, and the 20-state approximation as the controller's internal model. Both the plant and the internal model had zero initial state vectors. In the initial transient z_3 violates its lower constraint for about 150 minutes by a small amount before settling at the constraint value. z_1 and z_2 take about 200 minutes to settle to their set-points and during this time u_1 gradually increases up to its constraint. After 300 minutes all the variables have settled to the values expected from our earlier steady-state analysis. We note that the settling time of about 200 minutes is in accordance with the specification. It is also noteworthy that the control activity does not appear to be excessive, despite the fact that control moves have not been penalized.

After 300 minutes the set-point for u_3 is changed to -0.5 , in order to encourage it to become as close as possible to its lower constraint. u_1 does not change at all, since it is already on its upper constraint. u_2 and u_3 change very slightly, with the result that z_1 and z_2 depart from their set-points, and settle at $+0.025$ and -0.05 , respectively, which is outside the steady-state specification. This indicates that the current weighting of tracking errors, as defined by (9.10)–(9.12), gives too much importance to the u_3 set-point, relative to the z_1 and z_2 set-points, since we want the latter two to be the primary objectives of control. We therefore increase the z_1 and z_2 weights (namely element (2, 2) in (9.11) and elements (1, 1) and (2, 2) in (9.12)) from 1 to 10, leaving the weight on u_3 unchanged at 1. This was found to reduce the steady-state errors on z_1 and z_2 to $+0.001$ and -0.001 , respectively, which is within specification. But the

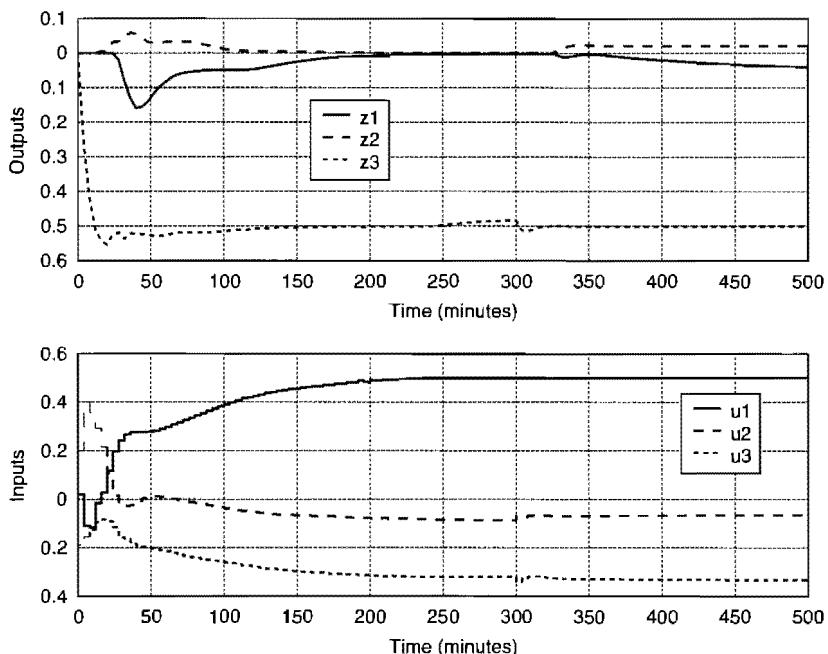


Figure 9.4 Response with initial tuning parameters.

inputs, especially u_2 and u_3 , were now seen to be oscillating considerably during the transient, and z_3 showed some oscillation about its constraint before settling down. The penalty weighting $R(i) = I_3$ was therefore imposed on control moves. This removed the oscillations, but now caused the steady-state error on z_1 to increase to just outside the specified value of ± 0.001 . The weight on z_1 tracking errors was therefore increased from 10 to 20, which had the desired effect. The corresponding response is shown in Figure 9.5.

Next we examine the effect of a disturbance on the Intermediate Reflux Duty (d_m), which is a measured disturbance, and can therefore be countered by feedforward action. Figure 9.6 shows the fractionator initially at the same steady state as reached at the end of the run shown in Figure 9.5. A disturbance occurs as follows — this is expressed in steps, each step lasting 4 minutes:

$$d_m(k) = \begin{cases} 0 & \text{for } k \leq 12 \\ +0.5 & \text{for } 12 < k \leq 37 \\ 0 & \text{for } 37 < k \leq 62 \\ -0.5 & \text{for } 62 < k \leq 87 \\ 0 & \text{for } k > 87 \end{cases} \quad (9.13)$$

Recall that an increase in the Intermediate Reflux Duty corresponds to less heat being taken away from the fractionator. Thus the initial disturbance of +0.5 allows the Bottoms Reflux Duty (u_3) to be reduced at first. This, however, leads to the Bottoms Reflux Temperature (z_3) falling below its constraint, so u_3 increases again, this being

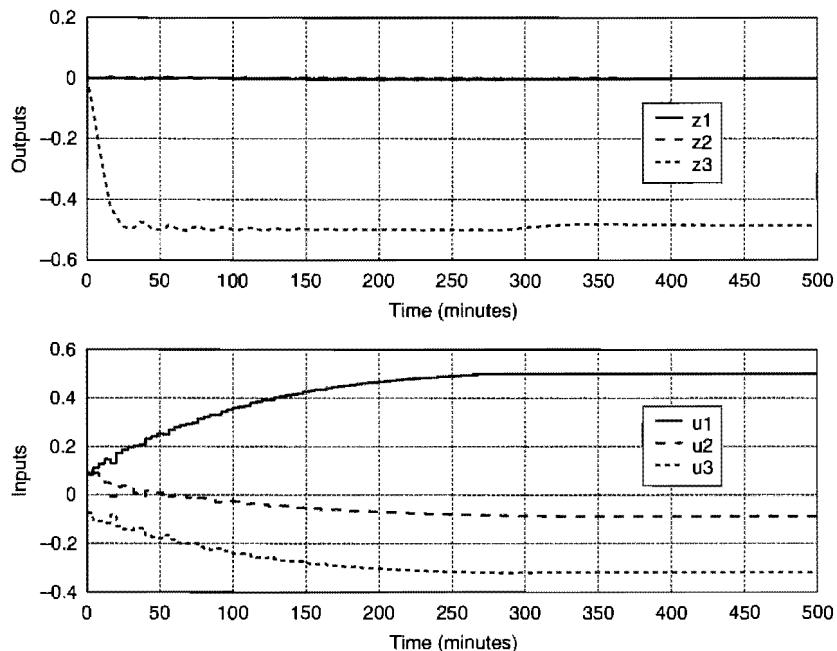


Figure 9.5 Response with increased weight on z_1 and z_2 tracking errors, and penalized control moves.

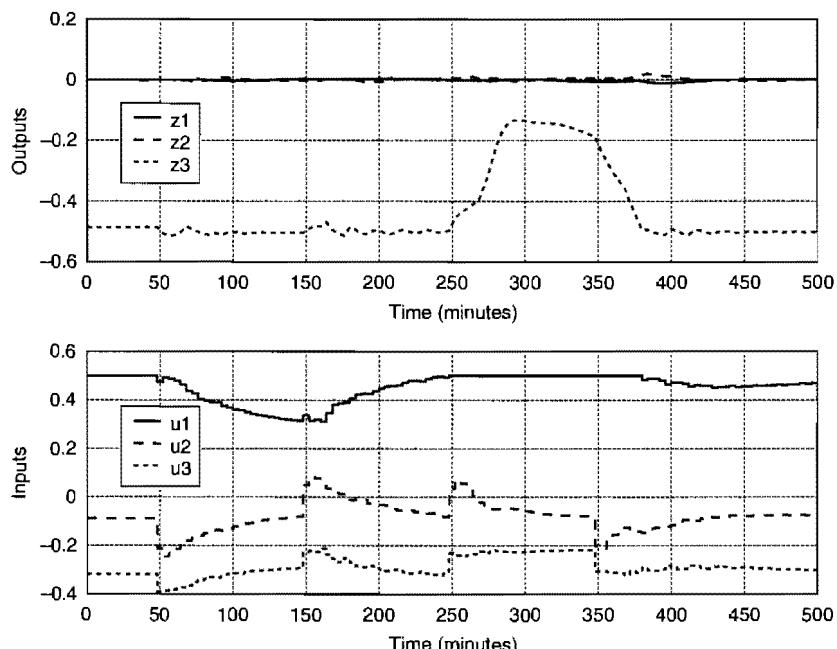


Figure 9.6 Response to measured disturbance.

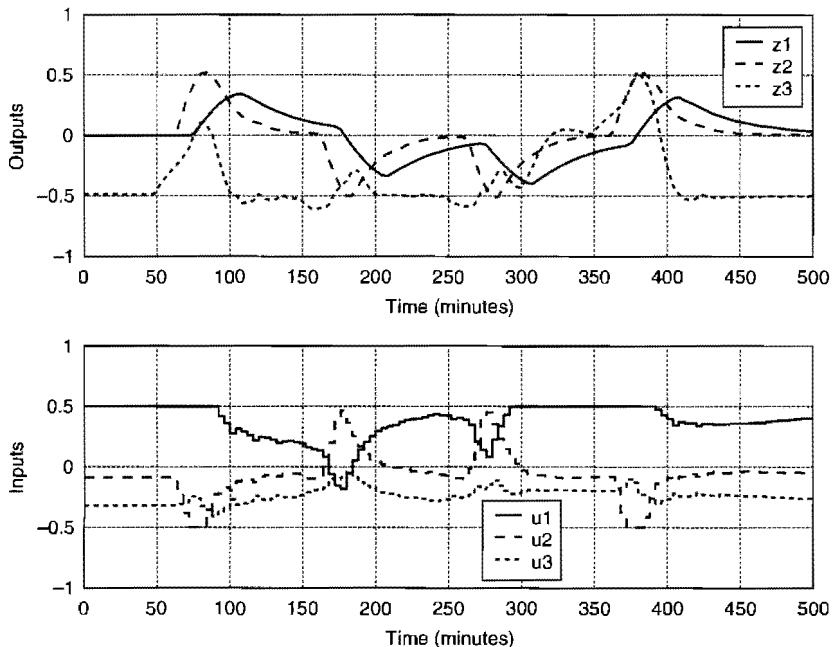


Figure 9.7 Response to unmeasured disturbance.

compensated by a reduction of u_1 away from its constraint. u_2 follows a similar pattern to u_3 . When the disturbance ceases, there is a transient lasting about 100 minutes, by the end of which all the variables have returned to their initial values. Then the disturbance $d_m = -0.5$ arrives; u_1 cannot increase now, since it is already at its upper constraint, so u_3 has to increase (from -0.32 to -0.22) for as long as the disturbance lasts. Note that the Bottoms Reflux Temperature (z_3) now moves away from its lower constraint, since it is a ‘zone’ rather than a ‘set-point’ variable. Finally (for $k > 87$, namely after about 350 minutes) the disturbance ceases again, and all the variables return to their initial values.

We now consider the response to a change of the Upper Reflux Duty, which is an unmeasured disturbance, so that feedforward cannot be used to correct its effects. We will apply the same pattern of disturbance as defined by (9.13). But we expect the control to be less effective in this case, since only feedback action is available. Figure 9.7 shows that this is indeed the case. The departures of z_1 and z_2 from their set-points are much bigger now, and take longer to correct, and the controller barely manages to keep the outputs within their constraints — in fact, several minor constraint violations occur.

9.1.5 Constraint softening

All the results shown so far were obtained with hard constraints enforced on outputs as well as inputs. For more severe disturbances it is necessary to soften the output constraints. For example, if measured and unmeasured disturbances occur together

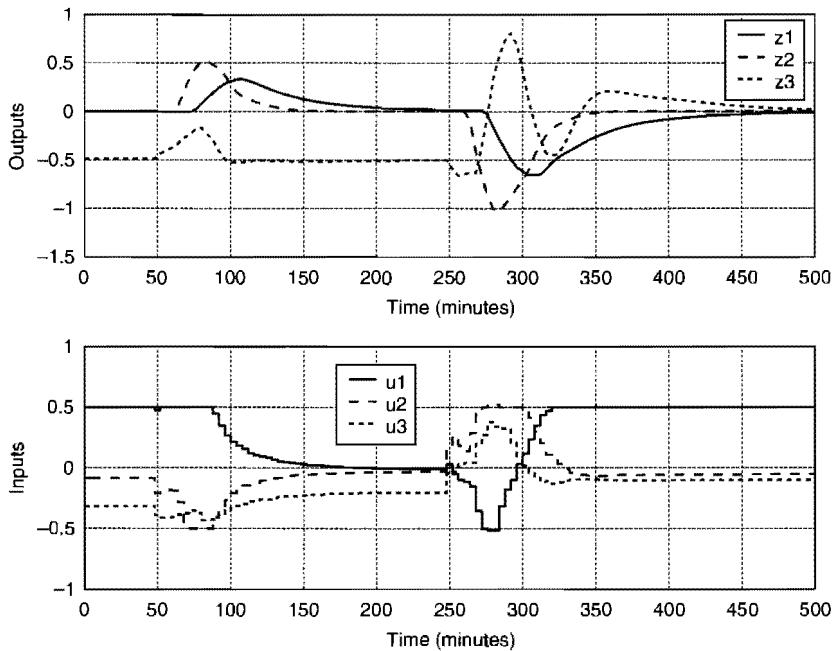


Figure 9.8 Response to measured and unmeasured disturbances, and soft constraints.

with the following form:

$$d_m(k) = d_u(k) = \begin{cases} 0 & \text{for } k \leq 12 \\ +0.5 & \text{for } 12 < k \leq 62 \\ -0.5 & \text{for } k > 62 \end{cases} \quad (9.14)$$

then infeasibility occurs when both disturbances take the value -0.5 . (Note that not only do both disturbances occur together, but they both suddenly jump from $+0.5$ to -0.5 , which is a much bigger change than in the simulations shown up to now.) If, however, the constraints are softened using an exact ∞ -norm penalty function, with penalty coefficient $\rho = 10^3$ (see Section 3.4), then the result is shown in Figure 9.8 — larger values of ρ give essentially the same results. When both disturbances are positive, there is no particular difficulty. All output variables are kept within constraints, and all inputs come off their constraints. When both disturbances become negative (after about 250 minutes), however, it is not possible to hold any of the three outputs within their constraints initially.

Very similar results are obtained if hard constraints are enforced, but not for the first three steps in the prediction horizon. That is, if there are no constraints enforced on $\hat{z}(k+i|k)$ for $i = 1, 2, 3$. The results are shown in Figure 9.9. Although this is an alternative way of recovering feasibility, it can be seen that there are more z_3 constraint violations than seen in Figure 9.8, and the peak violations are slightly larger. In particular, note the small but sustained z_3 constraint violation from about 100 to about 250 minutes into the simulation, which is not present in Figure 9.8.

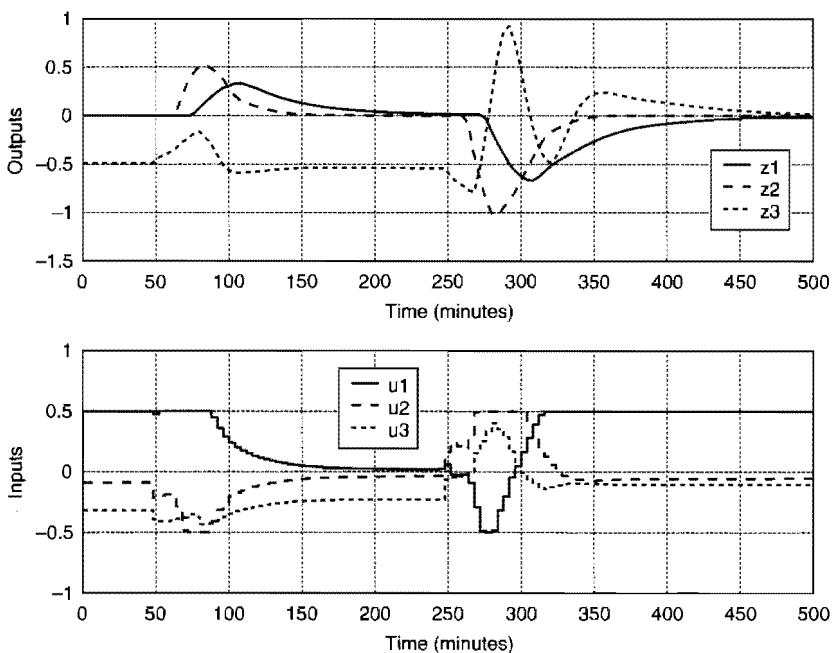


Figure 9.9 Response to measured and unmeasured disturbances, and hard constraints on $\hat{z}(k+i|k)$ for $i > 3$.

In both the cases shown in Figures 9.8 and 9.9 the steady-state values eventually assumed by z_1 and z_2 are within the ± 0.001 range permitted by the specification. (This is not visible on the figures, as the variables do not reach their steady-state values within the 500 minutes shown.) This is to be expected, because the steady-state values are determined by the same trade-off as in Figure 9.5. The disturbances cause the inputs to have different steady-state values now ($u_1 = 0.5$, $u_2 = -0.0525$, $u_3 = -0.1042$). The first two of these play no role in the cost function; the third does play a role, but the weight on its tracking error (away from its set-point of -0.5) is sufficiently small, compared with the weights on z_1 and z_2 , that the primary objectives are met within specification. Note that the set-point for u_3 prevents offset-free tracking from being obtained, despite the low error weight associated with it.

9.1.6 Robustness to model errors

It is not enough that our controller works when the plant is exactly as expected. The gains of the real fractionator may vary as described in Table 9.4, and the controller must continue to work satisfactorily as they do so.

Recall from Sections 7.3 and 8.1.1 that singular values of multivariable frequency responses give some indication of the robustness of a feedback design to modelling errors. However, these can only be computed for the linear controller which is obtained when all the constraints are inactive. Figure 9.10 shows the largest singular values of the

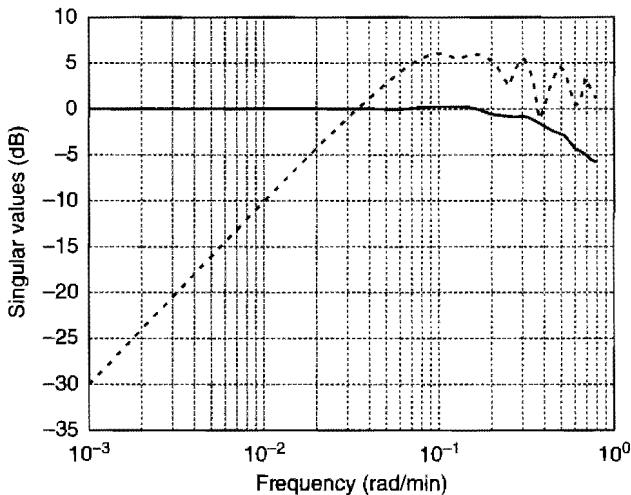


Figure 9.10 Largest singular values of sensitivity $S(z)$ (broken line) and complementary sensitivity $T(z)$ (solid line).

sensitivity and complementary sensitivity functions, $S(s)$ and $T(s)$ respectively, for the linear controller which results from the design presented in the previous section, when all the constraints are inactive. Since only outputs z_1 and z_2 are controlled to set-points, these singular values have been computed on the assumption that these two are the only outputs. (The singular values are shown for frequencies ω up to the Nyquist frequency π/T_s .) It will be seen that $\bar{\sigma}[T(e^{j\omega T_s})]$, the largest singular value of $T(z)$, has a value 1 (0 dB) at low frequencies. This indicates offset-free tracking, which is expected so long as constraints are not active. It has a peak value barely larger than 1; this can be shown to imply reasonably good robustness of the design to multiplicative perturbations of the plant's transfer function [Mac89, ZDG96]. It can also be seen that $\bar{\sigma}[T(e^{j\omega T_s})]$ falls to about 0.5 (-6 dB) at 'high' frequencies, namely near $\omega = \pi/T_s$. This indicates that not much attenuation of any measurement noise takes place, so that z_1 and z_2 will tend to follow any measurement noise that is present on the analyser outputs. This is not a serious problem, since the Nyquist frequency is rather low ($0.78 \text{ rad min}^{-1}$, or 0.002 Hz), so that any electrical noise can be assumed to have been removed from the measurements; however, no specification of measurement noise is given in the problem definition [PM87].

The plot of $\bar{\sigma}[S(e^{j\omega T_s})]$ shown in Figure 9.10 shows it falling to zero at low frequencies, which is again what is expected of the unconstrained behaviour, since it indicates offset-free tracking and is consistent with the behaviour of $\bar{\sigma}[T(e^{j\omega T_s})]$. Its peak value is 2 (6 dB), however, which is usually a little too large for comfort. It crosses the 0 dB line just above $0.03 \text{ rad min}^{-1}$, which shows that a sinusoidal (unmeasured) output disturbance with period shorter than about 200 ($= 2\pi/0.03$) minutes will be amplified rather than reduced by the feedback action.

We can employ the robust stability tests introduced in Section 8.1.1 to get a more accurate indication of the designed system's vulnerability to the possible gain changes

defined in Table 9.4. The nominal (continuous-time) plant model $P_0(s)$ is defined in Table 9.2. The actual plant can be represented as

$$P(s) = P_0(s) + W(s)\Delta \quad (9.15)$$

where $\Delta = \text{diag}\{\delta_1, \delta_2, \delta_3\}$, and $W(s)$ is a transfer function matrix with the same lag and delay structure as $P_0(z)$, but gains taken from Table 9.4. So, for example,

$$W(s)_{11} = \frac{2.11e^{-27s}}{50s + 1}. \quad (9.16)$$

We shall abuse notation slightly and refer to the transfer functions corresponding to $P_0(s)$, $P(s)$ and $W(s)$, after conversion to discrete-time representations, as $P_0(z)$, $P(z)$ and $W(z)$. Note that the diagonal block Δ remains unchanged in the discrete-time representation.⁴

This is an example of ‘structured uncertainty’, as discussed in Section 8.1.1. Since we have the bounds $|\delta| \leq 1$, it is certainly true that $\|\Delta\|_\infty \leq 1$. But we have much more information than that about the structure of the uncertainty: only diagonal elements in Δ can be non-zero, and furthermore these elements must be real. Since we have an additive representation of plant uncertainty, the test (8.7) for robust stability would be applicable if the uncertainty were unstructured. Adapted to this case, in which the uncertainty is $W(z)\Delta$, and $\|\Delta\|_\infty \leq 1$, the sufficient condition for robust stability becomes

$$\tilde{\sigma}[K(e^{j\omega T_s})S(e^{j\omega T_s})W(e^{j\omega T_s})] < 1 \quad (9.17)$$

As was said in Section 8.1.1, this test is likely to be too conservative — if it is not satisfied, this does not mean that the closed loop is not robustly stable. A less conservative condition is

$$\mu[K(e^{j\omega T_s})S(e^{j\omega T_s})W(e^{j\omega T_s})] < 1 \quad (9.18)$$

where $\mu[.]$ denotes the structured singular value for the diagonal uncertainty structure. But this is still conservative, because it allows for possibly complex gain perturbations. The conservativeness can be reduced further by checking the ‘real- μ ’ condition

$$\mu_R[K(e^{j\omega T_s})S(e^{j\omega T_s})W(e^{j\omega T_s})] < 1 \quad (9.19)$$

where $\mu_R[.]$ denotes the real structured singular value.

Figure 9.11 shows the largest singular value $\tilde{\sigma}[KSW]$, the complex structured singular value $\mu[KSW]$, and an upper bound for the real structured singular value $\mu_R[KSW]$, computed for an uncertainty structure of 1×1 blocks — these correspond to the scalar uncertainties δ_i in the process gains. Although this figure suggests that the designed controller should give instability with one of the possible plants, it is difficult to find such a plant — recall that the small-gain and complex- μ robustness tests give only sufficient, not necessary, conditions for robust stability, and that only an upper bound

⁴ The model $W(z)$ can be created by the MATLAB function `mkdelta.m`, which is available on this book’s web site.

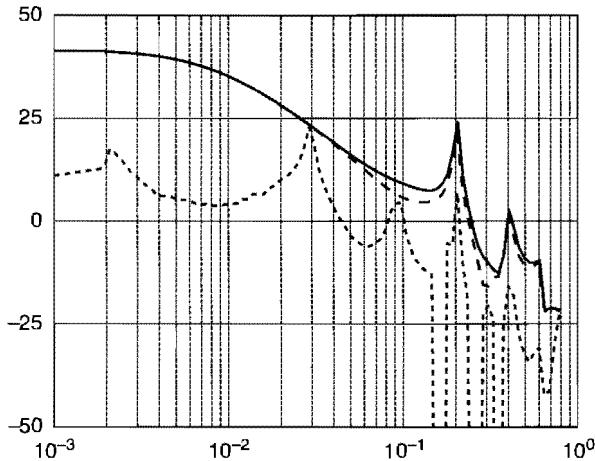


Figure 9.11 Largest singular value (solid line), and estimates of complex μ (broken line) and real μ (dotted line) of $K(z)S(z)W(z)$.

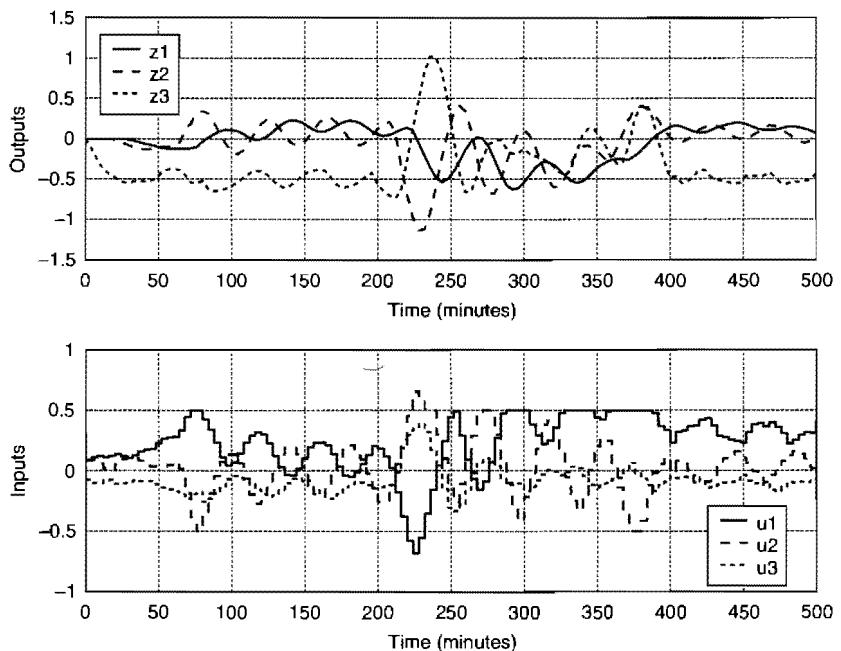


Figure 9.12 Response with plant–model mismatch, unmeasured disturbance, and soft constraints.

is available for μ_R . With $\delta_1 = -1$, $\delta_2 = \delta_3 = 1$, very underdamped closed-loop behaviour is obtained, which suggests that the system is close to instability for this perturbation. Figure 9.12 shows the response with this plant perturbation, when the

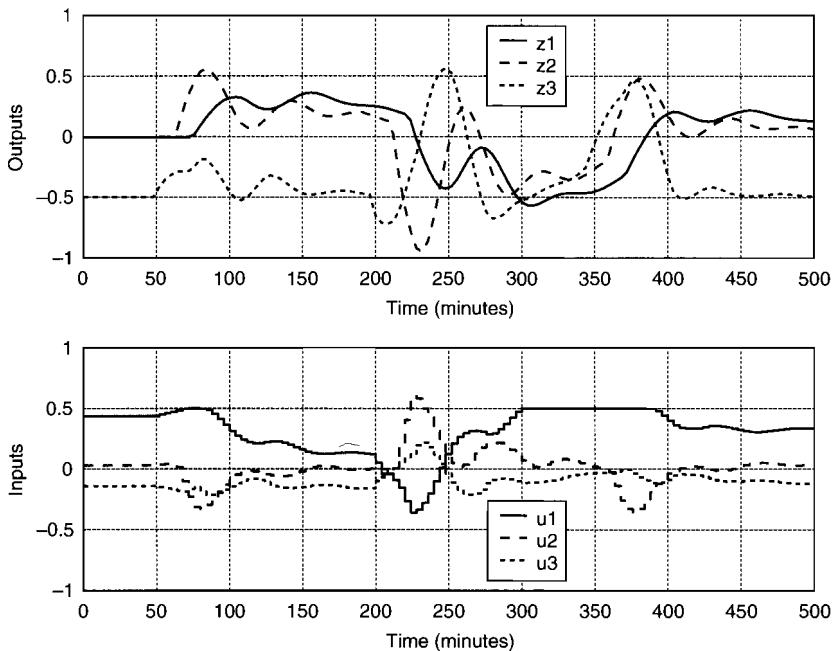


Figure 9.13 Response with plant–model mismatch, unmeasured disturbance, soft constraints, and increased control move penalty.

same unmeasured disturbance is present as defined in (9.13) and $\delta_5 = 1$. This response was obtained with soft constraints ($\rho = 10^3$), since the optimization became infeasible with hard constraints, as a result of the plant–model mismatch. Although not technically unstable, the response is now so erratic that the controller is in effect useless.

The simplest way of trying to increase robustness to plant–model mismatch is to reduce the controller gain. This is not always a successful strategy (because there are feedback systems in which a reduction of loop gain reduces the stability margins), but we saw from Figures 9.10 and 9.11 that $\bar{\sigma}(S)$ is quite small at all frequencies, whereas $\bar{\sigma}(KSW)$ is very large. Also it can be checked that $\bar{\sigma}(W)$ is relatively small, so that decreasing the controller gain is indeed indicated as a good strategy for increasing robustness. This is achieved most easily by increasing the control move penalty weighting matrices $R(i)$ in the MPC cost function. In the design discussed so far, we used $R(i) = I$. The easiest alternative is $R(i) = rI$, for some $r > 1$; the drawback to be expected, however, is that the control action will be slower and less effective. After some experimentation, $R(i) = 10I$ was found to be a reasonable compromise between good robustness and good control. Figure 9.13 shows the response to the same unmeasured disturbances as in Figure 9.12. The control of the outputs is a little better, but the movements of the inputs are much less erratic.

Figure 9.14 shows the response with $R(i) = 10I$ when the model is exact ($\delta_i = 0$ for all i). Comparing this figure with Figure 9.7, it is seen that not very much has been lost in nominal performance — the control of the outputs is not much worse than with the

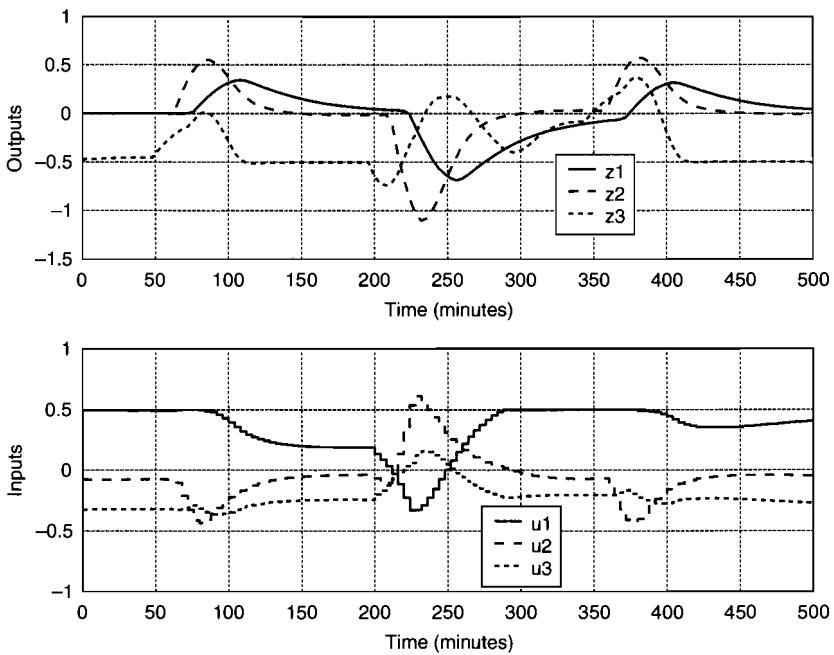


Figure 9.14 Response with correct model, unmeasured disturbance, and increased control move penalty.

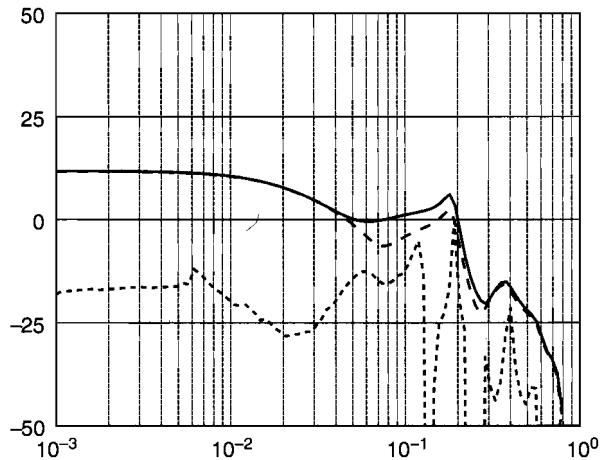


Figure 9.15 Largest singular value (solid line), complex μ (broken line) and real μ (dotted line) of $K(z)S(z)W(z)$ with the reduced-gain controller.

original design. Figure 9.15 shows the largest singular value, complex μ and real μ of $K S W$ for the reduced-gain controller. Comparing this figure with Figure 9.11, it can be seen that a significantly more robustly stable system is indicated for the reduced-gain controller.

In cases where the control specification is more demanding, the simple solution of reducing the controller gain by increasing $R(i)$ may not be sufficient. It may be necessary to introduce disturbance models in order to control the controller gain more selectively in some frequency ranges. (The Lee-Yu tuning procedure presented in Section 8.2 offers one way of doing this.)

9.2 Newell and Lee evaporator

The main objective of this case study is to illustrate the implementation of predictive control with a nonlinear plant. A nonlinear *Simulink* model of an evaporator serves as the plant to be controlled, and the *Model Predictive Control Toolbox* is used to control it using MPC.

This case study is based on the forced-circulation evaporator described by Newell and Lee [NL89], and shown in Figure 9.16. A feed stream enters the process at concentration X_1 and temperature T_1 , with flow rate F_1 . It is mixed with a recirculating liquor, which is pumped through the evaporator at a flow rate F_3 . The evaporator itself is a heat exchanger, which is heated by steam flowing at a rate F_{100} , with entry temperature T_{100} and pressure P_{100} . The mixture of feed and recirculating liquor boils inside the heat exchanger, and the resulting mixture of vapour and liquid enters a separator, in which the liquid level is L_2 . The operating pressure inside the evaporator is P_2 . Most of the

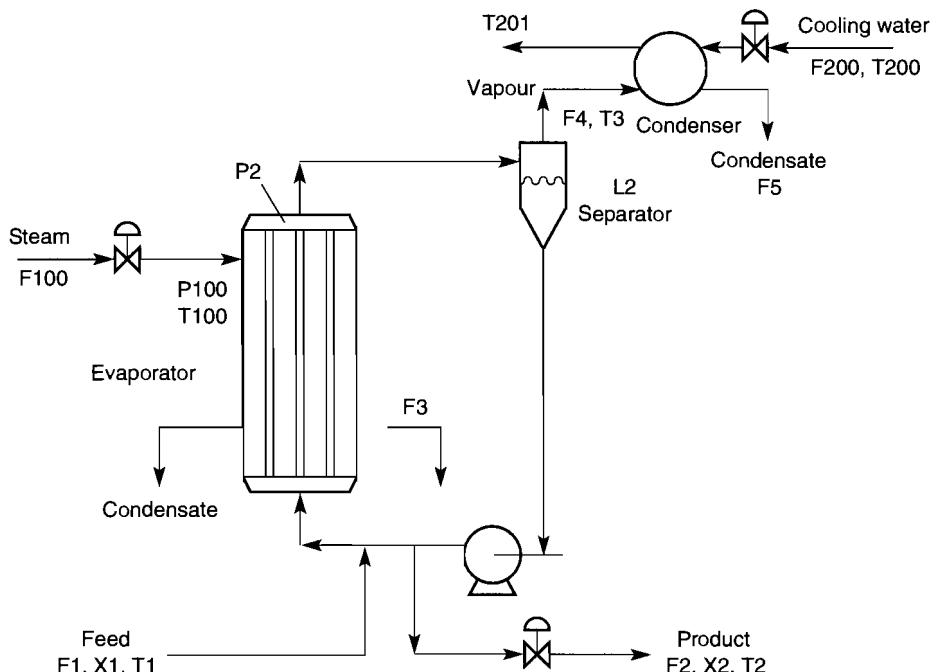


Figure 9.16 Forced-circulation evaporator.

Table 9.8 Evaporator output variables.

Output variable		Equilibrium value	Lower limit	Upper limit
Separator level	L_2	1 m	0	2
Product composition	X_2	25 %	0	50
Operating pressure	P_2	50.5 kPa	0	100

Table 9.9 Evaporator input variables.

Input variable		Equilibrium value	Lower limit	Upper limit
Product flow rate	F_2	2.0 kg/min	0	4
Steam pressure	P_{100}	194.7 kPa	0	400
Cooling water flowrate	F_{200}	208.0 kg/min	0	400

Table 9.10 Evaporator disturbance variables.

Disturbance		Equilibrium
Circulating flow rate	F_3	50.0 kg/min
Feed flow rate	F_1	10.0 kg/min
Feed concentration	X_1	5.0 %
Feed temperature	T_1	40.0 °C
Cooling water entry temperature	T_{200}	25.0 °C

liquid from the separator becomes the recirculating liquor; a small proportion of it is drawn off as product, with concentration X_2 , at a flow rate F_2 and temperature T_2 . The vapour from the separator flows to a condenser at flow rate F_4 and temperature T_3 , where it is condensed by being cooled with water flowing at a rate F_{200} , with entry temperature T_{200} and exit temperature T_{201} .

The three controlled outputs, together with their initial equilibrium (steady-state) values and constraints, are given in Table 9.8. The three manipulated variables (control inputs) to be considered in this study, together with the corresponding equilibrium values and their constraints, are given in Table 9.9. There are no rate constraints on the manipulated variables; the first-order lags, combined with the amplitude constraints, result in rate constraints being satisfied without imposing them explicitly.

There are five disturbance variables, although in this study we will leave them fixed. These, together with their corresponding equilibrium values, are given in Table 9.10.⁵ The nonlinear equations describing the operation of the evaporator are given in [NL89], and will not be repeated here.

A Simulink simulation model which implements these equations has been implemented, and is shown in Figure 9.17. The three manipulated variables, and the circulation flow rate F_3 , would in practice be controlled by local servos operating valves. These are not modelled in detail, but are represented by first-order lags with time-constants

⁵ In Chapter 12 of [NL89] the circulating flow rate F_3 is considered to be another manipulated variable.

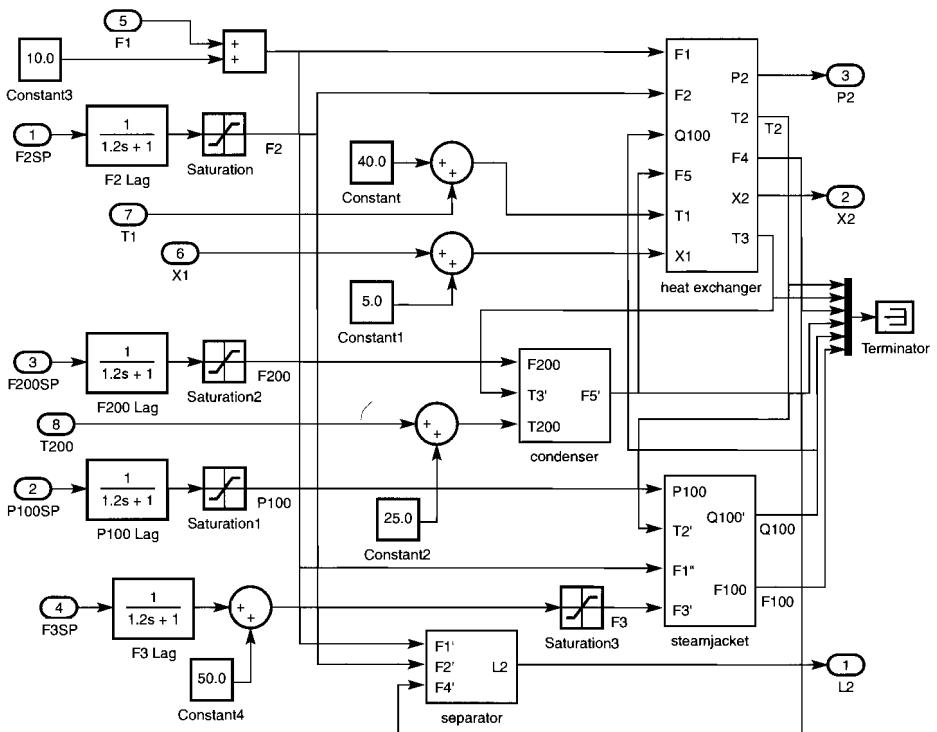


Figure 9.17 Simulink model of the evaporator.

of 1.2 minutes. For this reason inputs to the model are labelled as F_{2SP} rather than F_2 (etc.), to denote that they are set-points for the manipulated variables, rather than the variables themselves. This model is available on this book's web site, including the lower-level sub-models, namely the heat exchanger, the condenser, the separator, and the steam-jacket (in file *evaporator.mdl*). The model is in a form suitable for use with the *Model Predictive Control Toolbox* function *scmpcn1*, and its modified version *scmpcn12*, which is also available on the web site.

Since the model is nonlinear, it is linearized in order to get a linear internal model to be used by the predictive controller. An initial linearization is performed at the equilibrium condition defined in Tables 9.8–9.10. Figure 9.18 shows the result of using predictive control when the set-point for the product concentration X_2 is ramped down linearly from 25% to 15% over a period of 20 minutes (between 20 and 40 minutes into the simulation run), and the operating pressure P_2 is simultaneously ramped up from 50.5 kPa to 70 kPa. It should be noted that the controller had no advance information about the planned set-point trajectories, so it was reacting to tracking errors, rather than anticipating set-point changes.

The sampling interval used was $T_s = 1$ minute, and the MPC tuning parameters used to obtain these results were:

$$H_p = 30, \quad H_u = 3, \quad Q(i) = \text{diag}(1000, 100, 100), \quad R(i) = I_3 \quad (9.20)$$

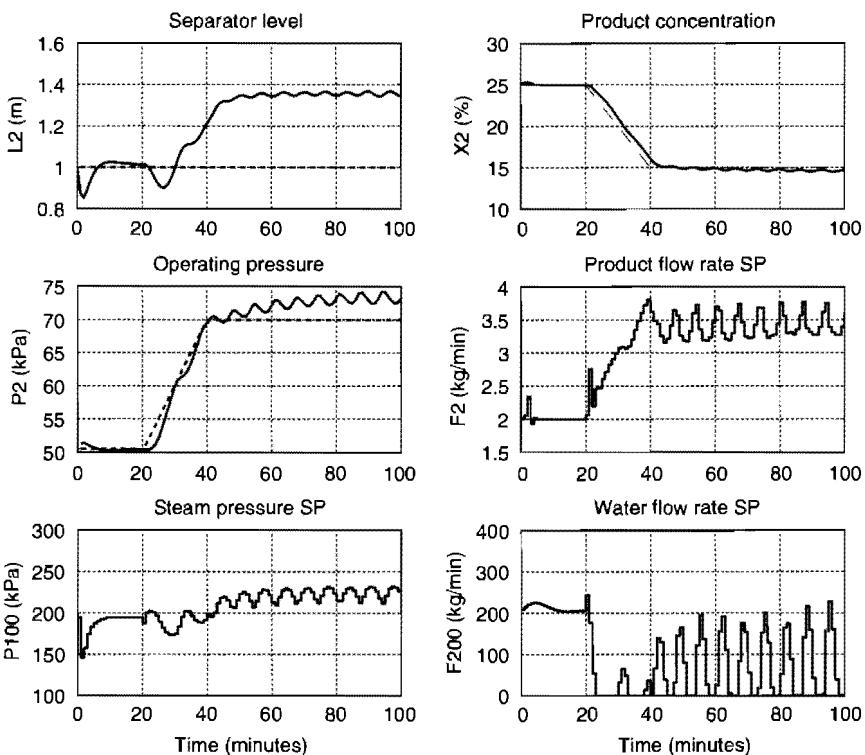


Figure 9.18 Simulation with single linearized model, obtained at the initial equilibrium condition. (Broken lines show the set-points.)

These parameters were not tuned carefully, but were almost the first set of parameters tried, the objective here being to show the effects of nonlinearities. The penalty on L_2 tracking errors is 10 times larger than those on X_2 and P_2 because L_2 is confined between 0 and 2 m, whereas the numerical values of X_2 and P_2 are considerably larger; these penalty weights indicate that a 1 m error in the separator level is considered equally undesirable as a 10% error in the product concentration or a 10 kPa error in the operating pressure.

It is seen from Figure 9.18 that control is initially satisfactory. The initial set-points are held quite closely (after a brief initial transient during which all the variables in the simulation settle to their equilibrium values), even when X_2 and P_2 start changing as they follow their ramp set-points. But about half-way through the ramp the separator level L_2 drifts considerably off its set-point and does not return to it. Although the product concentration X_2 tracks the set-point change quite closely, and settles correctly at the required steady-state level, all the other variables indicate underdamped closed loop behaviour, with the operating pressure P_2 oscillating and drifting away from its set-point. The manipulated variables show undesirable oscillation, with the cooling water flow rate F_{200} being particularly erratic, switching between shutting off the water flow completely and restoring it to its initial value.

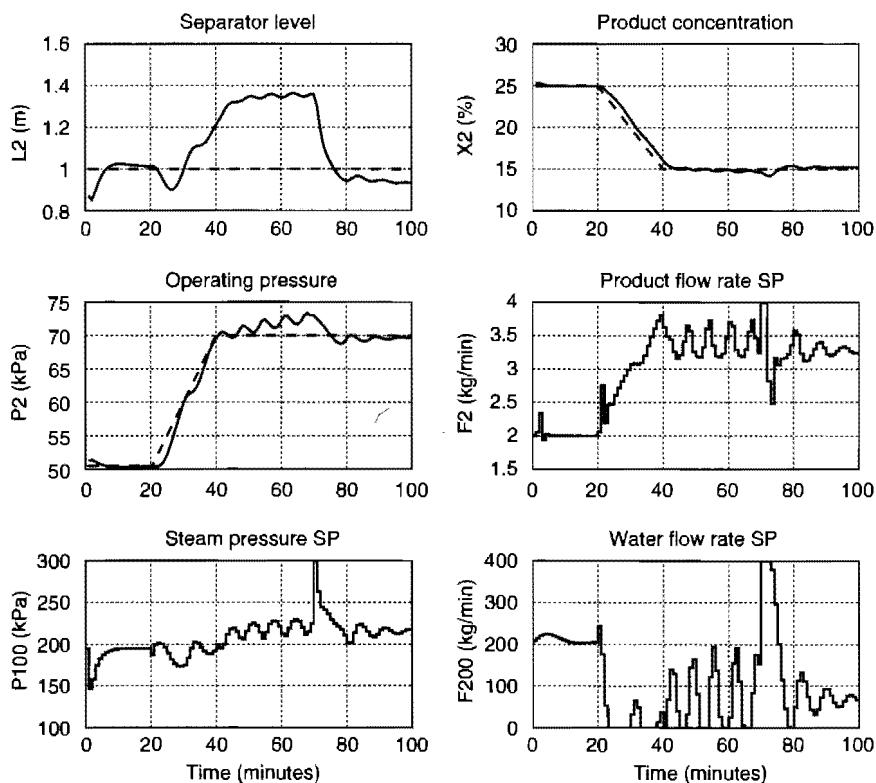


Figure 9.19 Simulation with a re-linearized internal model, obtained 70 minutes into the simulation run. (Broken lines show the set-points.)

This deterioration of closed-loop behaviour is due to the fact that the linearized model used by the predictive controller has become a very inaccurate representation of the small-deviation behaviour of the evaporator at the new operating conditions. Figure 9.19 shows the same scenario, but with the internal model used by the predictive controller re-linearized 70 minutes into the simulation run, namely 30 minutes after the new operating conditions have been (approximately) reached. Large transients are seen on all three manipulated variables as the new model is switched in — no attempt has been made to achieve ‘bumpless transfer’ in the simulation. Both L_2 and P_2 are brought back close to their set-points within 10 minutes of updating the internal model, and the manipulated variables have become less oscillatory within this time.

Figure 9.20 shows the results when the internal model is re-linearized every 10 minutes. Now it remains a reasonably good representation of the plant behaviour throughout the change of operating conditions, and it is clearly seen that the quality of control is much better. The product concentration and operating pressure are both held very close to their set-points throughout, and the separator level, although still undergoing considerable deviations from its set-point, is now controlled much more satisfactorily than before.

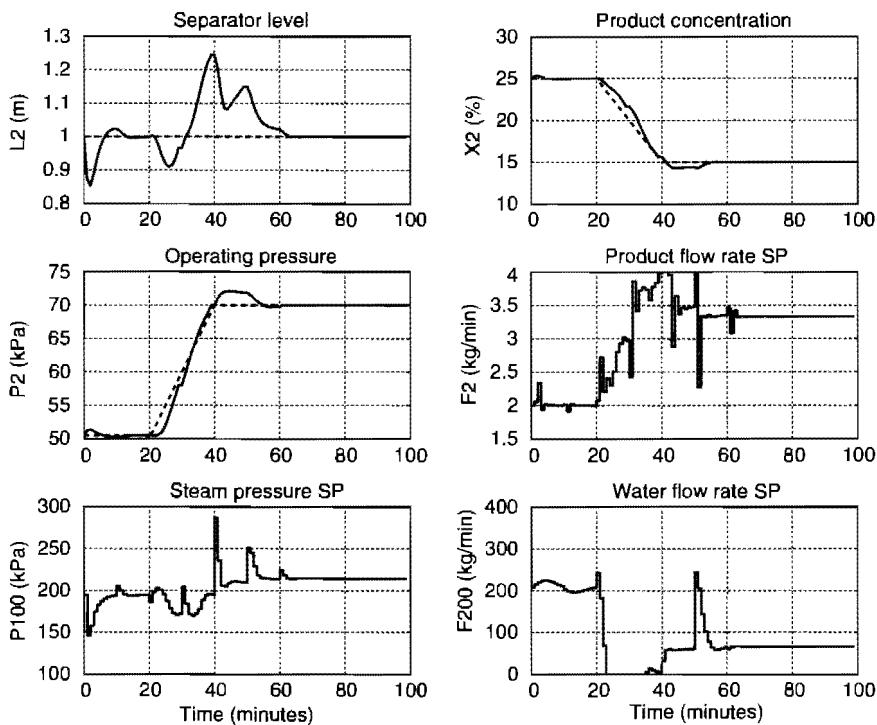


Figure 9.20 Simulation with a re-linearized internal model obtained every 10 minutes of simulated time. (Broken lines show the set-points.)

Re-linearization, or adaptation, of the linear internal model, is the most common way of dealing with plant nonlinearities in practice. It is not always adequate, particularly if the nonlinearities are more severe than those encountered in the evaporator. Section 10.3 discusses some other possibilities for dealing with nonlinear plants.

The reader is encouraged to experiment with re-linearizing the evaporator model at different intervals, using the *MATLAB* program contained in file *simevap.m*, and with different scenarios. It should be noted, however, that this program does not implement any of the checks which would be required in practice to check that a reasonable model has been obtained. Also, the linearized model obtained is not quite correct if the plant is not at equilibrium — see Section 2.1.2.

Exercises

These exercises are based on the two case studies which appear in this chapter. They require use of the *Model Predictive Control Toolbox* and of files provided on this book's web site. They are open-ended in the sense that there is no unique correct answer, and they may require considerable time to solve. The problems related to the Evaporator require some familiarity with *Simulink*.

- 9.1** In the design developed in Section 9.1.4, ‘blocking’ was used, corresponding to the parameter $M=[1, 2, 4, 8]$ in *Model Predictive Control Toolbox* function `smpc` (or `smpc2`). Investigate use of the simpler alternative of not using blocking, but still retaining five decision times (namely $H_u = M=5$). Simulate the same scenarios as shown in Figures 9.5–9.9. Show that the responses are not very different from the ones shown in these figures. (Use the *Model Predictive Control Toolbox* and the function `smpc2` available from this book’s web site.)
- 9.2** Continue exercise 9.1: Show that the singular values of the linear controller obtained in unconstrained operation are not affected much by abandoning ‘blocking’. (Use the *Model Predictive Control Toolbox*’s functions `smpccon`, `smpccl`, `svdfrsp`.)
- 9.3** Investigate by simulation the use of longer prediction horizons than used in Section 9.1.4, as regards both resulting performance of the closed-loop system and computation time requirements.
- 9.4** Investigate alternative choices of the prediction horizon H_p , the control horizon H_u , and the weights Q and R , for the evaporator. Consider how well the controller deals with disturbances on the feed flow rate F_1 , the feed concentration X_1 , and the cooling water temperature T_{200} . (To simulate disturbances, attach suitable *Source* blocks in place of the input ports on the disturbance variables in the *Simulink* model `evaporator.mdl`.)

Suppose the evaporator is to operate with product concentrations (X_2) of 25%, 15%, and 40%, all at an operating pressure of 50 kPa. Investigate whether you can find a single set of tuning parameters which gives good performance at all these concentrations, including transitions between them.

Is there a significant advantage in re-linearizing the internal model as the product concentration changes?

Note: Use, and modify as necessary, the file `simevap.m`, which is available on this book’s web site.

- 9.5** Simulate the same scenario as shown in Figures 9.18–9.20, but with the separator level treated as a *zone* variable — namely with no set-point, but only the constraints $0.5 < L_2 < 1.5$. (See Section 5.5.)
- 9.6** It makes more sense to re-linearize the internal model when the operating point has changed significantly, rather than at regular intervals of time. Examine the *MATLAB* script file `simevap.m`. It allows the evaporator model to be re-linearized at regular intervals of time. Modify it so that the model is only re-linearized if the state has changed by more than some threshold amount. (You will still need to interrupt the simulation at regular intervals, then test whether the state has changed significantly.)

Using the modified file, simulate the same scenario as shown in Figures 9.18–9.20, re-linearizing the model whenever the product concentration X_2 has changed by more than 5%.

Perspectives

10.1 Spare degrees of freedom	276
10.2 Constraint management	281
10.3 Nonlinear internal models	284
10.4 Moving-horizon estimation	289
10.5 Concluding remarks	290

This chapter introduces a few topics which are potentially very important, but which are mostly still at the research stage.

10.1 Spare degrees of freedom

10.1.1 Ideal resting values

If the plant has more inputs than outputs, then a particular constant set-point y_s can be held by a whole family of constant input vectors u_s . There may be good reasons for preferring one of these, usually because there are different economic costs associated with the different control inputs. (Reducing air flow may be cheaper than burning more fuel, for instance.) The traditional story in optimal control has been to say that such preferences should be reflected in the choice of the $R(i)$ matrices. But in practice that is not so easy as it sounds, since $Q(i)$ and $R(i)$ have complicated effects on the dynamic performance. The desirable constant values of the inputs are usually determined at the same time as the set-points are determined, using static optimization — usually by solving an LP problem.

If for some reason set-points are known, but the control inputs are not determined uniquely, then the solution sometimes adopted is to first do a steady-state least-squares optimization, to find the best combination of steady-state inputs which will give a desired set-point vector. If the numbers of inputs and outputs are equal and the

steady-state gain matrix is nonsingular, then this is unnecessary because the solution is unique. If there are more inputs than outputs, then the equation

$$y_s - S(\infty)W_u u_s = 0 \quad (10.1)$$

is solved in a least-squares sense, where $S(\infty)$ is the steady-state gain matrix and W_u is a diagonal weighting matrix. This gives the solution with the smallest value of $\|W_u u_s\|$. (If there are more outputs than inputs then it is usually impossible to reach the correct set-point, and in this case the equation

$$W_y(y_s - S(\infty)u_s) = 0 \quad (10.2)$$

is solved in a least-squares sense.)

When there are more inputs than outputs the elements of u_s are called the *ideal resting values* of the plant inputs for the given set-points. Once they have been found the cost function used in the predictive control problem formulation is augmented with the term $\sum_{i=0}^{H_u-1} \|\hat{u}(k+i|k) - u_s\|_{\Sigma(i)}^2$, to encourage the inputs to converge to these 'ideal' values. A danger of this is that, if the steady-state gains are not known accurately, or if there are constant disturbances, then the computed values of u_s will not be exactly correct for attaining y_s at the output, and offset-free tracking may be lost as a result.

10.1.2 Multiobjective formulations

It is difficult to express all the objectives of control into a single cost function. For example, assigning a high weight to the control of a particular output is not really the same as assigning it a high priority. It does not express the objective 'Keep output j near its set-point only if you can also keep output ℓ at its set-point'. A 'multiobjective' formulation makes it easier to do this. This can be applied also in situations in which it may be impossible to meet all the objectives simultaneously. There are several approaches to multiobjective optimization, but one well-suited to the situation in which there are various priorities for various objectives is to have a hierarchy of optimization problems. The most important one is solved first, then this solution is used to impose equality constraints when performing the second optimization, and so on. For example, the 'ideal resting value' problem treated above may be better solved by making the achievement of set-points the primary objective, and the achievement of ideal resting values a secondary objective to be pursued only when the controlled outputs are close to their set-points, and imposing the equality constraints that the outputs should equal their set-points.

The complexity of the optimization problem to be solved may not increase, and may even decrease, as a result of adopting such a strategy. Predictive control problems solved in this way can keep a QP formulation at each step, and it is generally quicker to solve several small problems than one large equivalent problem. In particular, the first, most important problems to be solved have the smallest number of conflicts and the largest number of degrees of freedom, so have the best prospects of a good solution being found, and of this being done quickly. If one runs out of computing time before the next control signal must be produced, then the still-unsolved problems are the least important ones.

10.1.3 Fault tolerance

Predictive controllers seem to offer very good possibilities for fault-tolerant control. If a sensor of a controlled output fails, for example, it is possible to abandon control of that output by removing the corresponding output from the cost function (by setting the corresponding elements of $Q(i)$ to zero). An actuator jam is easily represented by changing the constraints ($|\Delta u_j| = 0$). If a failure affects the capabilities of the plant, then it is possible to change the objectives, or the constraints, or both, accordingly. This is also possible with other control strategies, but it is particularly clear, in the predictive control formulation, how the required changes should be introduced into the controller. This is mainly due to the fact that the control signal is re-computed at each step by solving an optimization problem, so that *one can make changes in the problem formulation*. This is a very different proposition to changing the gains or time constants in a pre-computed controller.¹

Note that it may not be an easy problem to know how to change the problem formulation if a failure occurs. All that can be said is that, if an appropriate change is known, then it is easy to introduce it and implement it in the predictive control framework.

In [Mac97] it is argued that predictive control offers the possibility of a generic approach to achieving fault tolerance by reconfiguring the controller in response to a serious failure. This proposal relies on combining predictive control with the increasing use of detailed ‘high-fidelity’ first-principles models to represent complex processes, and with progress in the area of Fault Detection and Identification (FDI).

But it has also been noticed that predictive control has certain fault-tolerant properties even in the absence of any knowledge of the failure. In [Mac98] it is shown that predictive control, with the standard ‘DMC’ model of disturbances and with hard constraints on the input levels, can automatically redistribute the control effort among actuators, if there are more actuators than controlled outputs, and one (or more) actuator fails.

Some of these capabilities are illustrated in Figure 10.1, which is taken from [Mac97]. It shows a predictive controller responding to a step demand in the yaw angle of a large civilian aircraft when the rudder is jammed.

Case 1 is the normal response when the rudder is working correctly. Case 2 shows the response when the rudder is jammed, and the predictive control problem formulation has no constraints; even in this situation the set-point is eventually achieved, which indicates that other control strategies could achieve this too. Case 3 shows the response when the controller is aware of the usual $\pm 20^\circ$ rudder angle limit. The response is now faster, because after a few seconds the controller ‘thinks’ that it has driven the rudder to the limit, and starts to move the other control surfaces; but the controller has no knowledge of the failure in this case. Case 4 shows the response when the rudder failure is communicated to the controller by reducing the ‘rudder move’ constraint to zero. This makes surprisingly little difference in this case (the curves for Cases 3 and 4 are indistinguishable); the only difference is that the controller knows immediately that it cannot move the rudder, whereas in Case 3 it took about 3 seconds before it realized this. This does not make much difference to the complete manoeuvre. Case 5 shows

¹ However, it might be possible to re-compute on-line controllers for other strategies, which have traditionally been pre-computed off-line, rather like ‘classical’ adaptive control proposals.

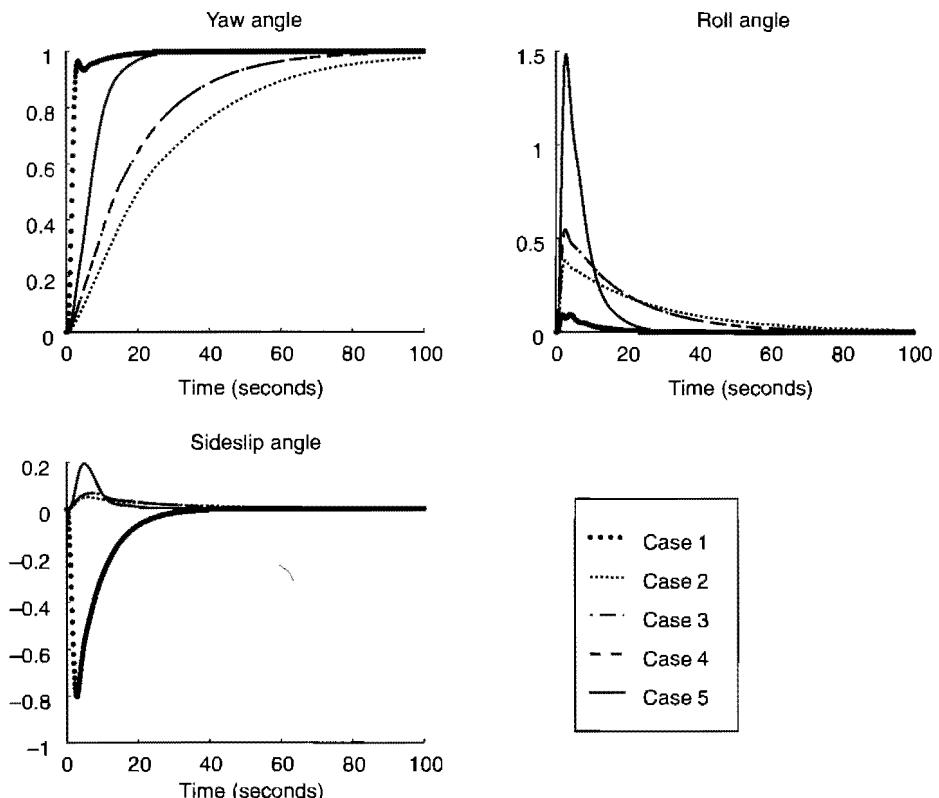


Figure 10.1 Execution of a step change in yaw angle with a jammed rudder.

that better response can be achieved by changing the weights — in this case the weight on keeping the roll angle set-point was reduced, so that the controller made more use of roll angle to reach its objective; this also shows a potential difficulty of this approach to fault-tolerant control, namely that in general some re-tuning may be necessary, and this may not be an easy thing to do on-line.

Another example of the ‘daisy-chaining’ property arises in connection with a plant for producing liquefied natural gas (LNG), shown in Figure 10.2, which is taken from [Row97]. It has six inputs: the compressor speed, and the positions of the five valves; and five outputs: the levels of liquid in the two separators, and the exit temperatures of the three heat exchangers (of the natural gas). The compressor speed is limited to the range 25 to 50 revs/sec; the valve fully closed to fully open range is represented by -500 to +500. In the example considered the sampling interval is 10 sec, and at this long interval the input rates are effectively unlimited. Each separator level is limited to the range -2 to +2 m (relative to the operating point), and the evaporator exit temperatures are limited to the ranges -6 to +30 K, -13 to +7 K, and -11 to +4 K, relative to their nominal values.

Figure 10.3 shows the simulated response to a step demand of +10 K on the exit

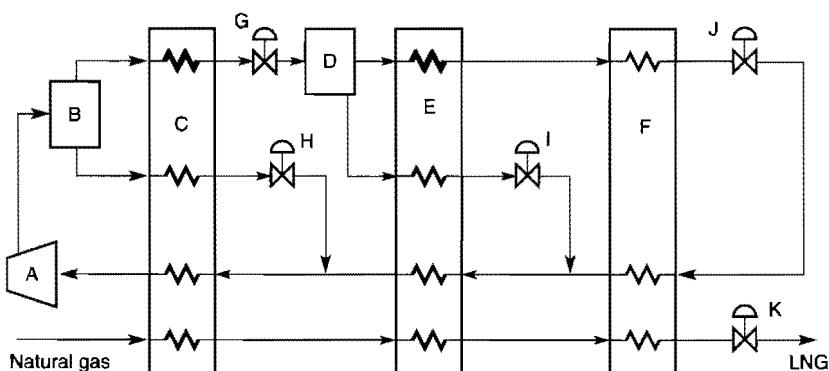


Figure 10.2 LNG liquefaction plant.

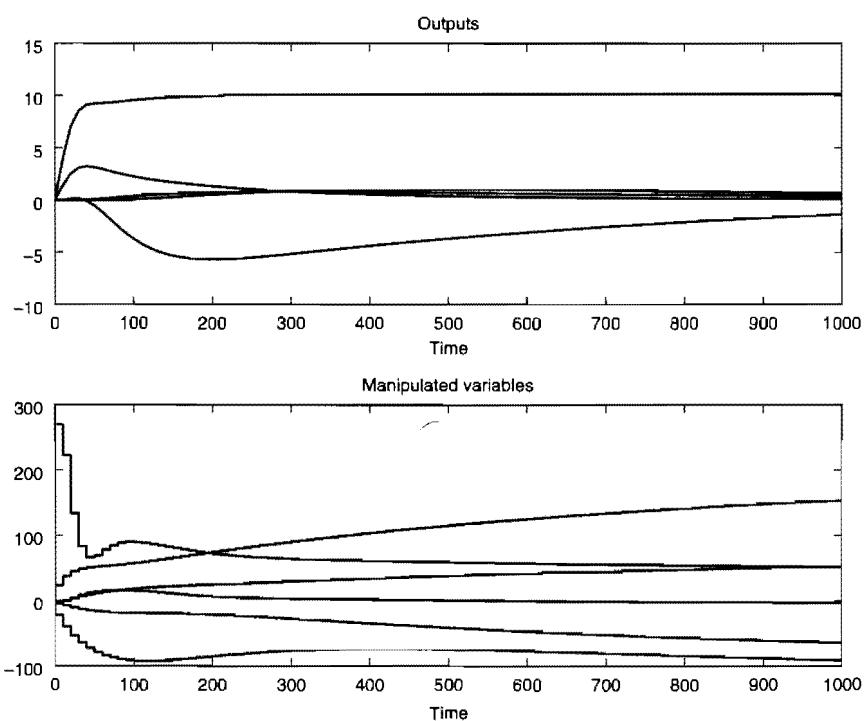


Figure 10.3 LNG plant: response in normal operation.

temperature of heat exchanger C when there is no failure, the set-points for both separators and heat exchanger F are zero, and the exit temperature of heat exchanger E is not controlled, providing it remains within its constraints. The horizons are $H_p = 80$ and $H_u = 5$.

Figure 10.4 shows the response in the same situation, but with valve H stuck at

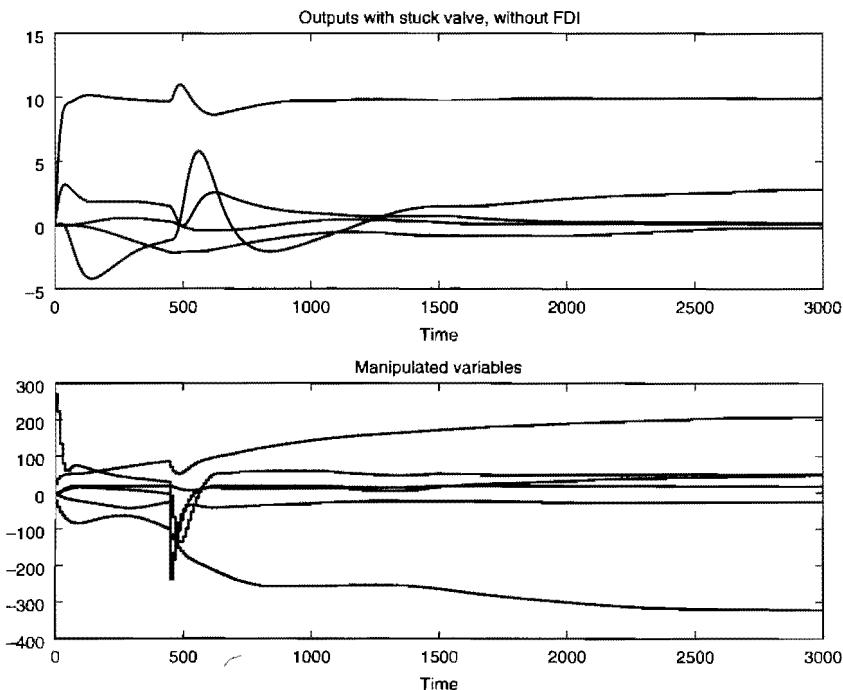


Figure 10.4 LNG plant: valve H stuck, controller unaware.

+60, and with the controller unaware that this failure has occurred. It will be seen that the inputs have been reconfigured to adjust for the failure, with all outputs eventually returning to their set-points, except for the uncontrolled one, which remains within its allowed constraints. Note that the time-scale is different in this figure; the transient takes much longer than in normal operation, but all constraints are respected, and effective control over the process is maintained. Typically the penalty of such reconfiguration will be reduced production rate (valve K not at its nominal position), but at least continued production is possible.

When the controller *is* aware that valve H is stuck, and of its position then, not surprisingly, the transient is finished much more quickly, and in fact resembles very closely the one obtained in normal operation. (All the simulations shown here were performed on a linearized model.)

The potential of predictive control for fault-tolerant control has also been considered in [GBMR98, GMR99].

10.2 Constraint management

A major problem with using constrained predictive control is reacting to infeasibilities. In general this requires some change to be made to the constraints. The strategy recommended by Rawlings and Muske [RM93], and (previously) implemented in DMC,

is to have a *constraint window* within which constraints are enforced (defined in our notation by C_w and C_p in Chapter 6). Infeasibilities can always be removed from the problem formulation by opening the constraint window late enough. Of course this may not remove ‘real-world’ infeasibilities from the plant, and is only a useful strategy if some constraints can be relaxed without causing serious damage, or posing an unacceptable risk.

One school of thought argues that infeasibility is such an unacceptable situation to face, that ‘hard’ output constraints should never be used, but should be replaced by ‘soft’ constraints, as described in Section 3.4. Again, this is only a useful prescription if some violation of the constraints can be tolerated; in practice this means that one has to impose relatively conservative constraints. ‘Hard’ input constraints, on the other hand, are acceptable, both because constraints on inputs are really ‘hard’ in nature, and because input constraints do not cause infeasibilities (with stable plants).

It is widely accepted that the ‘correct’ thing to do in case of infeasibility is to decide on the relative importance of the constraints, and to release the relatively unimportant ones until feasibility is recovered. It has recently been proposed [TM99] that a systematic way of integrating this process into the predictive control framework can be developed, by using propositional logic. The idea is that statements about priorities of constraints and objectives can be translated into statements expressed in propositional logic. These can be put into a canonical form, which can be translated into an integer-valued optimization problem. Finally this problem can be combined with the standard QP problem of predictive control, to give a *mixed integer quadratic program (MIQP)* — or a mixed integer linear program if the underlying predictive control problem is formulated as an LP problem (see Section 5.4). Global optima of such problems can be obtained reliably, but the computational complexity involved is much greater than for ordinary QP or LP problems.

Example 10.1

This example is taken from [TM99, Example 1]. For the steady-state model

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} + \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \quad (10.3)$$

the most important constraint is $\max_j |y_j| \leq 2$, while a secondary objective is $|y_2| \leq \epsilon$, where $\epsilon > 0$ is very small. The top-priority constraint can be represented by the conjunction (simultaneous satisfaction) of three pairs of inequalities such as (for $|y_1| \leq 2$):

$$u_2 + d_1 \leq 2 + 10(1 - \ell_1) \quad (10.4)$$

$$-u_2 - d_1 \leq 2 + 10(1 - \ell_1) \quad (10.5)$$

where $\ell_1 = 0$ or $\ell_1 = 1$, and 10 is chosen as being sufficiently large that it is an upper bound on $|y_j|$ under any circumstances. The secondary priority can be represented by

the pair of inequalities:

$$u_1 + d_2 \leq \epsilon + 10(1 - \ell_2) \quad (10.6)$$

$$-u_1 - d_2 \leq \epsilon + 10(1 - \ell_2) \quad (10.7)$$

where again $\ell_2 \in \{0, 1\}$. The idea is that $\ell_1 = 1$ corresponds to the first constraint being enforced, and $\ell_2 = 1$ corresponds to the second one being enforced. Now the clever trick is that the priorities are represented by the linear inequality

$$\ell_1 - \ell_2 \geq 0 \quad (10.8)$$

because this cannot be satisfied if $\ell_2 = 1$ unless $\ell_1 = 1$ also.

The optimal control can now be found by solving the (mixed-integer) optimization problem:

$$\min_{u_1, u_2} -\ell_1 - \ell_2 \quad (10.9)$$

(In this case the only objective is the satisfaction of the constraints. There is no linear or quadratic penalty of deviations from the set-point.)

A refinement of this ensures that if one or both constraints cannot be met, then the amount by which the highest-priority constraint is violated is minimized.

This approach has recently been extended to a very promising framework for handling a variety of *hybrid system* problems that involve mixtures of numerical and logical variables [BM99a]. A *Mixed Logical Dynamical* (MLD) system is one of the form

$$x(k+1) = Ax(k) + B_1u(k) + B_2\delta(k) + B_3z(k) \quad (10.10)$$

$$y(k) = Cx(k) + D_1u(k) + D_2\delta(k) + D_3z(k) \quad (10.11)$$

$$E_1\delta(k) + E_2z(k) \leq E_3u(k) + E_4\delta(k) + E_5 \quad (10.12)$$

in which the state vector x , the input vector u and the output vector y all contain both real-valued and $\{0, 1\}$ -valued elements. δ is a vector of binary $\{0, 1\}$ -valued auxiliary variables, and z a vector of real-valued auxiliary variables. These auxiliary variables are needed for transforming propositional logic statements into linear inequalities. Inequality (10.12) specifies all the constraints on the system, as well as specifying priorities for objectives and constraints (as was done in (10.8)).

In a parallel development [VSJ99] it has been shown how the magnitudes of low-priority constraint violations can be minimized, without the use of mixed-integer optimization. In fact, only a single linear programme is required, provided that the objective function is properly chosen, which gives a great simplification compared with the use of mixed-integer programming. The idea is to formulate an optimization problem such that a lower-priority constraint violation cannot be decreased without increasing a higher-priority constraint violation. The solution to such a problem is called a *lexicographic minimum* of the violated constraints (by analogy with the way in which entries are ordered in a telephone directory, for example). The solution proposed in [VSJ99] is to minimize a linear combination of the slack variables which appear in a penalty function, such as was introduced in Section 3.4 for softening constraints. If the slack variables

are ordered such that the first one corresponds to the highest-priority constraint, and the last one to the lowest-priority constraint, then a lexicographic minimum can be ensured by choosing a sufficiently large ratio between succeeding weights in the linear combination. Determining how large this ratio needs to be involves solving another LP problem (but off-line).

In [KBM⁺00] a similar idea to that of [VSJ99] is used in combination with MLD systems. Mixed-integer optimization is relied on, but greater flexibility of specification is gained. For example, the satisfaction of the greatest number of constraints, either taking priorities into account or not, can be enforced, as can prioritized minimization of the duration of constraint violations. It is also shown how the same approach can be used to formulate multiobjective problems, and prioritizing multiple objectives as well as constraints.

10.3 Nonlinear internal models

10.3.1 Motivation and approaches

If the plant exhibits severe nonlinearities, the usefulness of predictive control based on a linearized model is limited, particularly if it is used to transfer the plant from one operating point to another one. The obvious solution is to use a nonlinear model, and quite a lot of research is going on into this possibility. The main disadvantage is that convexity of the optimization problem is then lost, which is a serious drawback for on-line application. The disadvantage is not so much that the global optimum may not be found, since the use of a linearized model only gives the illusion of finding a global optimum — one finds the global optimum of the wrong problem. The real disadvantage for on-line use is that one has very little idea of how long each optimization step will take to complete, whether it will ever terminate, what to do if a feasible solution cannot be found, and so on.

The most straightforward approach to the nonlinear MPC problem is to retain the formulation of MPC as we have developed it in this book, simply replacing the linear model by a nonlinear one, and rely on suitable optimization methods to try to overcome the loss of convexity. This approach may lead to perfectly acceptable performance in practice, at least in suitable applications, but is usually very difficult or impossible to analyse. Lack of analysis, however, has not been an obstacle to the application of predictive control in the past.

A compromise solution, and the one which is most commonly used in practice, is to re-linearize the nonlinear model as the plant moves from one operating point to another, and to use the latest linear model as the internal model at each step. The linearized model is not necessarily updated at each step, but may be left unaltered as long as the plant is in the vicinity of a particular operating condition. This results in a QP problem to be solved at each step, although the model changes from time to time. If the nonlinearity is expected to be significant because set-points are due to be changed — for example from one production rate to another — then one can use a time-varying linear model which varies during the prediction horizon [HM98]. At each point in the horizon one uses the linearization corresponding to the state at which one expects to be at that time.

Note that this linearization cannot depend on the optimized input trajectory if convexity of the optimization is to be maintained.

Another problem which becomes very prominent with nonlinear MPC is state estimation. In most applications the true state $x(k)$ is not measured, and an estimate $\hat{x}(k|k)$ must be obtained. This becomes much more problematic when the model is nonlinear [GZ92, Beq91, QB99].

10.3.2 Sequential quadratic programming

If a nonlinear internal model is used, then several optimization algorithms are available. The one which is closest to the algorithms introduced in Chapter 3 is *sequential quadratic programming* (or ‘SQP’). Suppose that a general constrained optimization problem is to be solved, of the form:

$$\min_{\theta} \{V(\theta) : H_i(\theta) = 0, \Omega_j(\theta) \leq 0\} \quad (10.13)$$

where $\{H_i(\cdot)\}$ and $\{\Omega_j(\cdot)\}$ are sets of nonlinear functions, and that an iterate θ_k has been obtained. The SQP algorithm makes a quadratic approximation to $V(\theta_k)$:

$$q_k(d) = \frac{1}{2} d^T \nabla_{\theta\theta}^2 L(\theta_k, \lambda_k) d + \nabla V(\theta_k)^T d \quad (10.14)$$

where $L(\theta, \lambda) = V(\theta) + \sum_i \lambda_i H_i(\theta) + \sum_j \lambda_j \Omega_j(\theta)$ is the Lagrangian. The next iterate, θ_{k+1} , is given by

$$\theta_{k+1} = \theta_k + d_k \quad (10.15)$$

where d_k is found by solving the quadratic programming problem which results from minimizing $q_k(d)$, subject to local linear approximations of the constraints:

$$\min_d \{q_k(d) : H_i(\theta_k) + \nabla H_i(\theta_k)^T d = 0, \Omega_j(\theta_k) + \nabla \Omega_j(\theta_k)^T d \leq 0\} \quad (10.16)$$

This is the basic idea of SQP, but there are several variations, for example taking different approaches to estimating the vector of Lagrange multipliers, λ_k , or the Hessian matrix $\nabla_{\theta\theta}^2 L(\theta_k, \lambda_k)$. The iterate θ_k may well violate the original constraints, since it only satisfies the local linear approximations to the constraints. A variation which may be of particular significance for predictive control is *feasible* SQP, which ensures that each iterate satisfies the original nonlinear constraints. This may be an important feature for real-time optimization, if there is insufficient time to allow the iterates to converge, and one has to take the best iterate found so far.

The models used in nonlinear MPC can be either detailed ‘first-principles’ models, or ‘black box’ models such as neural nets or Volterra models. Both kinds lead to the basic problem of a non-convex optimization problem. Bock *et al* [BDLS99, BDLS00] propose a *multiple shooting* approach for solving nonlinear MPC problems, which is particularly intended for use with complex ‘first-principles’ nonlinear models, consisting of large sets of differential or differential-algebraic equations. When such models are used, solution of the model by numerical integration is not trivial, and should be considered

together with the problem of solving the optimization problem to find the control input trajectory. The proposed ‘multiple shooting’ approach exploits the structure of the nonlinear MPC problem, taking the numerical integration into account, in particular by suitable adaptations of the SQP algorithm. The term ‘multiple shooting’ refers to the solution strategy, which divides the prediction horizon up into smaller intervals, and initially finds almost-independent iterates on each interval, gradually reconciling the iterates with each other so that a continuous input trajectory is obtained as the solution.² This strategy lends itself naturally to parallelization, if multiple processors are available for the optimization.

In [DOP95] a ‘black box’ model in the form of a second-order Volterra series model is used. It is shown that this results in an MPC controller which has the structure of a ‘standard’ MPC controller for the linear part of the model, together with a ‘correction’ for the nonlinear behaviour. This paper also contains a brief but interesting discussion of how such Volterra models may be obtained.

10.3.3 Neural net models

For our purposes, neural net-based models are just one class of black-box nonlinear models. They are of interest because they are increasingly being used with success in a wide range of applications, because they have theoretically attractive approximating power, and because there are established techniques for estimating them. Here we will outline two specific ways of using neural nets for nonlinear MPC which have been proposed, without going into any details of the neural nets themselves. In order to find such details the reader is referred to [NRPH00], which includes a section on the use of neural nets for predictive control. Also see [Lee00], and [LD99] for an interesting application.

One interesting attempt to use nonlinear neural net models, while retaining the advantages of convex optimization, is presented in [LKB98], in which it is suggested that the nonlinear system should be approximated by a *set* of neural-net predictors which are affine in the inputs. A different predictor in the set is used for each prediction length:

$$\hat{x}(k + P_1 | k) = f^1(x(k)) + \sum_{i=0}^{P_1-1} g_i^1(x(k)) \hat{u}(k+i|k) \quad (10.17)$$

⋮

$$\hat{x}(k + P_c | k) = f^c(x(k)) + \sum_{i=0}^{P_c-1} g_i^c(x(k)) \hat{u}(k+i|k) \quad (10.18)$$

where the functions $f^j(\cdot)$ and $g_i^j(\cdot)$ (for $j = 1, \dots, c$) are nonlinear functions implemented by the set of c neural nets. This results in a convex optimization problem, so long as the cost function is linear or quadratic (or some other convex function). This approach may be of particular interest if the number of coincidence points c is

² On each interval, the input trajectory is assumed to be structured as a low-order polynomial, which makes an interesting connection with predictive functional control — see Section 5.6.

small. The identification of suitable neural nets is discussed in [LKB98]. (The idea of using a separate predictor for each prediction interval can also be found in the research literature emphasizing the use of predictive control schemes for adaptive control, such as *MUSMAR*. There linear models are used, and the benefit of separate predictors is that implicit system identification algorithms can be kept relatively simple, because only linear regression is required [Mos95].)

Neural net models are used differently in the commercial MPC product *Process Perfecter* developed by Pavilion Technologies [PSRJG00]. Here a neural net is used to approximate the steady-state input–output map. It is assumed that the plant is to be transferred from one initial steady state, say y_i , to another final steady state, say y_f . Steady-state optimization is performed to find suitable steady-state input values, say u_i and u_f , respectively. Consider a single-input, single-output plant, for simplicity — the ideas carry over to multivariable plant, but the notation becomes messy.³ The neural net model is then used to obtain the ‘local’ input–output gains, at the initial and final steady states:

$$g_i = \left[\frac{\partial y}{\partial u} \right]_{u_i, y_i}, \quad g_f = \left[\frac{\partial y}{\partial u} \right]_{u_f, y_f} \quad (10.19)$$

(These are the steady-state gains of the locally linearized models at the two steady states.) Finally a second-order dynamic model is assumed, in difference-equation form, which is *quadratic* in the inputs:

$$\begin{aligned} \delta y(t) = & -a_1 \delta y(t-1) - a_2 \delta y(t-2) + v_1 \delta u(t-d-1) + v_2 \delta u(t-d-2) \\ & + w_1 [\delta u(t-d-1)]^2 + w_2 [\delta u(t-d-2)]^2 \end{aligned} \quad (10.20)$$

where δ now signifies deviations from initial steady-state values: $\delta u(t) = u(t) - u_i$, $\delta y(t) = y(t) - y_i$, and d is an input–output delay. The coefficients a_1 and a_2 are fixed, as in a linear difference equation model, but v_1 and v_2 depend on g_i , while w_1 and w_2 depend on both g_i and g_f . In an exact Taylor series expansion one would have $v_j = \delta y(t)/\partial u(t-d-j)$, and $w_j = \partial^2 y(t)/\partial u(t-d-j)^2$. It is not surprising, then, that these coefficients are given a gain-dependence of the following form:

$$v_j = b_j g_i, \quad w_j = b_j \frac{g_f - g_i}{u_f - u_i} \quad (j = 1, 2) \quad (10.21)$$

where b_1 and b_2 are some fixed coefficients. If the v_j and w_j coefficients were fixed, a nonlinear model would still be obtained; but allowing them to vary in this way results in a much better approximation being obtained while retaining the simple quadratic model, because the coefficients are ‘tuned’ for the particular transient that needs to be performed. Note that since the resulting dynamic model is quadratic in the inputs, the optimization which needs to be performed for MPC is not convex, in general. However, the model has been kept very simple, so that the optimization needs to be performed over relatively few decision variables. The hope is that this results in a tractable optimization problem, and apparently this turns out to be the case in many applications [PSRJG00].

³ In essence we need a multivariate Taylor series expansion, up to quadratic terms.

The assumption behind the form of (10.21) is that the change in steady-state gains is the principal nonlinearity in the plant. The identification of suitable models of this form is aided by the fact that there is often an extensive database of historical data available, from which one can estimate steady-state gains for a wide range of operating conditions.

10.3.4 Sub-optimal nonlinear MPC

A more radical way of avoiding the problems associated with non-convex optimization, or at least of ameliorating them, is to shift the emphasis away from optimization, and towards the satisfaction of constraints as the primary objective. An interesting proposal in this direction has been made by Scokaert, Mayne and Rawlings [SMR99]. This involves a dual-mode strategy, in which the state is steered towards a terminal constraint set X_0 , as in [MM93] and [SM98]. It is assumed that, once the state has entered X_0 , a robust asymptotically stabilizing controller $u(k) = h_L(x(k))$ is available which drives the state to the origin, and such that X_0 is a positively invariant set for $x(k+1) = f(x(k), h_L(x(k)))$. (The subscript L denotes a ‘local’ controller.) The original feature of the proposal, however, is the MPC strategy which is used while the state is outside X_0 . The system is assumed to be nonlinear:

$$x(k+1) = f(x(k), u(k)) \quad (10.22)$$

with input and state constraints of the general form:

$$u(k) \in U, \quad x(k) \in X \quad (10.23)$$

The cost function is also assumed to be of a general form:

$$V(k) = \sum_{j=1}^N l(\hat{x}(k+j|k), \hat{u}(k+j-1|k)) \quad (10.24)$$

and it is assumed that the cost falls to zero once the state has entered X_0 , namely $l(x, h_L(x)) = 0$ if $x \in X_0$. The following terminal constraint is imposed:

$$\hat{x}(k+N|k) \in X_0 \quad (10.25)$$

Now the control is chosen not by minimizing the cost (10.24), but by finding a predicted control trajectory which gives a *sufficient reduction* of the cost. Any control trajectory which gives

$$V(k) \leq V(k-1) - \mu l(\hat{x}(k|k-1), \hat{u}(k-1|k-1)) \quad (10.26)$$

where $0 < \mu \leq 1$, is acceptable, providing that (10.23) and (10.25) are satisfied. As usual in MPC, the applied control is $u(k) = \hat{u}^0(k|k)$, where $\hat{u}^0(k+j-1|k)$, $j = 1, \dots, N$ is the selected input trajectory. It is shown in [SMR99] that, under mild technical conditions, which we omit, this sub-optimal MPC strategy gives an asymptotically stable closed loop, and the region of attraction — namely the set of initial states which are driven to the origin by this strategy — is precisely the set of

states which can be driven to X_0 in N steps without violating the constraints. It is also shown that, in the absence of disturbances, the state is driven into X_0 in a finite number of steps. In order to prove these results, a generalized Lyapunov stability theorem is proved in [SMR99], which allows the control law to be nonunique and discontinuous — both features being needed because of the way the future control trajectory is chosen.

The main significance of this result is that it removes the need for finding the global minimum of a non-convex function. In fact, it is not even necessary to find a local minimum. All that is needed is to find a solution which gives a sufficient reduction in the cost function at each step, as defined in (10.26). If sufficient time is available for the computation, one can continue searching for solutions which give a bigger reduction, but that is not necessary. Also, the difficulty of finding such a solution can be controlled by adjusting the value of μ . Values close to zero make it relatively easy to find a solution, though of course at the cost of poor performance, in the sense of relatively small reduction of the cost function at each step. Such small values may be appropriate if there are large modelling errors, or large disturbances, so that it is difficult to find a feasible trajectory. Another interesting feature of this MPC strategy is that, if there were no modelling errors or disturbances, then one would only need to calculate the future trajectory initially (at $k = 0$), and not recalculate it again. This is because the initially calculated trajectory is feasible and provides the required cost reduction at each step — optimality is no longer an issue. In practice, of course, recalculation is necessary, but previously calculated trajectories provide good initial guesses ('hot starts').

10.4

Moving-horizon estimation

The standard way of estimating the state of a dynamic system from input–output measurements is to use a state observer. For a linear model there is a complete theory — see Mini-Tutorial 2 — and if statistical information is available about disturbances and measurement noise then optimal estimates can be obtained by using the Kalman filter. Although there is some observer theory for use with nonlinear models, it is much less complete. Optimal state estimation problems can still be defined, but usually they cannot be solved exactly, and *ad hoc* approximations must be used, such as the Extended Kalman Filter. One such *ad hoc* approach is *moving-horizon estimation*, which is a kind of 'dual' of predictive control.

Instead of predicting ahead, one looks back at input–output data over a finite horizon, and tries to find an estimated state trajectory which is consistent with an assumed model, and which fits the measured data as closely as possible. For example, with a linear model and a moving horizon of length N , one can pose a problem such as

$$\min_{\hat{x}(k-i|k)} \sum_{i=0}^{N-1} \{ \|\hat{y}(k-i|k) - y(k-i)\|_Q^2 + \|\hat{x}(k-N|k) - x_0\|^2 \} \quad (10.27)$$

subject to

$$\hat{x}(k-i+1|k) = A\hat{x}(k-i|k) + Bu(k-i) \quad (10.28)$$

$$\hat{y}(k-i|k) = C\hat{x}(k-i|k) \quad (10.29)$$

where the data $u(k-i), y(k-i) : i = 0, 1, \dots, N$ are given, and the last term in (10.27) is an *initial cost* which penalizes deviations of the initial estimate $\hat{x}(k-N|k)$ from some known value x_0 , which may perhaps be the corresponding estimate obtained at the previous step, $\hat{x}(k-N|k-1)$. This is a linear least-squares problem; but if the real state variables are known to be bounded, from physical considerations, or more general linear constraints are known to hold, then linear inequality constraints of the form

$$M_i^k \hat{x}(k-i+1|k) \leq m_i^k \quad (10.30)$$

can be added, and the problem becomes a quadratic programming problem. This problem is outside the scope of conventional state estimation theory.

We have given only the simplest formulation here; the problem can be generalized, especially to the use of nonlinear models. Similar questions arise as in predictive control, in particular:

- ➊ How to deal with lack of convexity when nonlinear models are used?
- ➋ How to ensure stability of the estimator — that is, how to ensure that the estimate $\hat{x}(k|k)$ converges to $x(k)$, assuming the model is correct?
- ➌ How to ensure that the stability of the estimator is robust with respect to errors in the model?

The investigation of such questions has begun [SR95, RR99a]. The use of moving horizon estimation has also been suggested with systems described in the mixed logical-dynamic (MLD) framework — see (10.10)–(10.12) — for estimating states in hybrid systems, for detecting faults, etc. [BMM99].

Constrained moving horizon estimation has also been proposed for the related problem of *data reconciliation* [LEL92]. This arises when one has not only a partly unknown state, but also limited confidence in the correctness of the measurements, and possibly uncertainty about the model. The objective is to modify the measurements so that they become consistent with the model. The model is often a nonlinear first-principles model, and one frequently knows constraints on the real values of measured variables, and on states. This approach has also been suggested for detecting failed sensors and other faults.

10.5 Concluding remarks

A recent survey, quoted in [QB96], states that a conservative estimate of the number of predictive control installations in industry worldwide in 1996 was 2200. Most of these were in refining and petrochemical applications, but some penetration into other industries such as food processing and pulp-and-paper was evident. (The company *Adersa* reported 20 applications in the aerospace, defence and automotive industries. No other vendor reported any applications in these industries.) The largest of these was an olefines plant, with 283 inputs and 603 outputs.

Clearly predictive control technology is a successful technology. Where does this success come from? In particular, why has it been so much more successful than

'classical' LQ control, which uses a similar quadratic cost function, which offers stability and (since about 1980) robustness guarantees, but which has had very little application in the process industries.⁴ I believe that its success is due to a combination of the following factors, listed here in decreasing order of importance:

1. Systematic handling of constraints. No other technique provides this. This reduces the need for *ad hoc* 'safety-jacket' software to handle things like integrator wind-up, unexpected conditions etc. It probably also provides some robustness, in the sense that if the unconstrained control does not do a good job and the plant moves towards dangerous regions, then the constraints 'catch' it and return it to safer regions. This property is probably quite robust, because it depends largely on 'coarse' information, such as signs of gains rather than the precise values.
2. Commercial predictive control software does a lot more than solve predictive control problems as formulated in this book. A hierarchy of optimization problems is solved typically, and there are features which aid constraint and goal management in case of failures or unusually large disturbances. Combining such functionality with other control strategies might be equally effective — and there is no particular reason why this could not be done.
3. Commercial vendors usually supply a 'turnkey' system. In addition to selling the software, they model the plant and tune the controller before handing it over to the customer. The great majority of the time goes into modelling and identification. It is not customary for vendors of PI controllers, say, to provide a similar service, or for the customer to devote a comparable effort to modelling and tuning.⁵
4. The basic ideas behind predictive control are easy to explain and understand. Although the technicalities can become very intricate when plant-wide constrained optimization is implemented, it is still possible to present information to plant operators in such a way that they retain confidence in the actions taken by a predictive controller. This has been very important for the acceptance of predictive control.

I believe that these features of predictive control will lead to its very wide use in control engineering, beyond the process industries. One can already find references to its use in flight control [LP95, SSD95, PHGM97], submarines [SB98] and medical applications [ML98], for example. For a practitioner's view of future applications of predictive control see [San98].

⁴ But LQ control has been successful in aerospace applications, and in high-performance servos, such as those used in disk drives.

⁵ In fact, predictive control vendors usually insist on 'tuning up' the low-level PI controllers which receive set-points from the predictive controllers. One wonders how much of the success is already due to this preliminary activity.

References

- [AM79] B.D.O. Anderson and J.B. Moore. *Optimal Filtering*. Prentice Hall, 1979.
- [AM90] B.D.O. Anderson and J.B. Moore. *Optimal Control, Linear Quadratic methods*. Prentice Hall, 1990.
- [Åst80] K.J. Åström. A robust sampled regulator for stable systems with monotone step responses. *Automatica*, 16:313–315, 1980.
- [ÅW84] K.J. Åström and B. Wittenmark. *Computer Controlled Systems: Theory and Design*. Prentice Hall, 1984.
- [ÅW89] K.J. Åström and B. Wittenmark. *Adaptive Control*. Addison-Wesley, 1989.
- [AZ00] F. Allgöwer and A. Zheng, editors. *Nonlinear Model Predictive Control*. Birkhäuser, 2000.
- [BB91] S. Boyd and C. Barratt. *Linear Controller Design, Limits of Performance*. Prentice Hall, 1991.
- [BCM97] A. Bemporad, A. Casavola, and E. Mosca. Nonlinear control of constrained linear systems via predictive reference management. *IEEE Transactions on Automatic Control*, 42:340–349, 1997.
- [BDLS99] H.G. Bock, M. Diehl, D. Leineweber, and J. Schlöder. Efficient direct multiple shooting in nonlinear model predictive control. In Keil, Mackens, Voss, and Werther, editors, *Scientific Computing in Chemical Engineering II: Simulation, Image Processing, Optimization, and Control*. Springer, 1999.

- [BDLS00] H.G. Bock, M. Diehl, D. Leineweber, and J. Schlöder. A direct multiple shooting method for real-time optimization of nonlinear DAE processes. In Allgöwer and Zheng [AZ00], pages 245–267.
- [Bem98] A. Bemporad. Reducing conservativeness in predictive control of constrained systems with disturbances. In *Proceedings, 37th IEEE Conference on Decision and Control*, pages 1384–1391, Tampa, 1998.
- [Beq91] B.W. Bequette. Nonlinear predictive control using multi-rate sampling. *Canadian Journal of Chemical Engineering*, 69:136–143, 1991.
- [BGFB94] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. SIAM, Philadelphia, 1994.
- [BGW90] R.R. Bitmead, M. Gevers, and V. Wertz. *Adaptive Optimal Control: The Thinking Man's GPC*. Prentice Hall, Englewood Cliffs, NJ, 1990.
- [BH75] A.E. Bryson and Y-C. Ho. *Applied Optimal Control; Optimization, Estimation, and Control*. Hemisphere, Washington, 1975.
- [Bla99] F. Blanchini. Set invariance in control. *Automatica*, 35:1747–1767, 1999.
- [BM99a] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 1999.
- [BM99b] A. Bemporad and M. Morari. Robust model predictive control: a survey. In A. Garulli, A. Tesi, and A. Vicino, editors, *Robustness in Identification and Control*, volume 245 of *Lecture Notes in Control and Information Sciences*. Springer, 1999.
- [BMDP] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit solution of model predictive control via multiparametric quadratic programming. In *Proceedings, American Control Conference, Chicago, June 2000*. IEEE. (CD-ROM).
- [BMDP99] A. Bemporad, M. Morari, V. Dua, and E.N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. Technical Report AUT 99-16, Automatic Control Laboratory, ETH Zurich, 1999.
- [BMM99] A. Bemporad, D. Mignone, and M. Morari. Moving horizon estimation for hybrid systems and fault detection. In *Proceedings, American Control Conference*, San Diego, 1999.
- [CA98] H. Chen and F. Allgöwer. A quasi-infinite horizon nonlinear model predictive control scheme with guaranteed stability. *Automatica*, 34:1205–1217, 1998.
- [CBB94] E.F. Camacho, M. Berenguel, and C. Bordons. Adaptive generalized predictive control of a distributed collector field. *IEEE Trans. Control Systems Technology*, 2:462–468, 1994.

- [CC95] E.F. Camacho and C. Bordons. *Model Predictive Control in the Process Industry*. Advances in Industrial Control. Springer, Berlin, 1995.
- [CC99] E.F. Camacho and C. Bordons. *Model Predictive Control*. Advanced Textbooks in Control and Signal Processing. Springer, London, 1999.
- [Cla88] D.W. Clarke. Application of generalized predictive control to industrial processes. *IEEE Control Systems Magazine*, 122:49–55, 1988.
- [Cla94] D.W. Clarke, editor. *Advances in Model-Based Predictive Control*. Oxford University Press, Oxford, 1994.
- [CM86] P.J. Campo and M. Morari. ∞ -norm formulation of model predictive control problems. In *Proceedings, American Control Conference*, pages 339–343, 1986.
- [CM87] P.J. Campo and M. Morari. Robust model predictive control. In *Proceedings, American Control Conference*, pages 1021–1026, 1987.
- [CM89] D.W. Clarke and C. Mohtadi. Properties of generalised predictive control. *Automatica*, 25:859–875, 1989.
- [CM97] C.T. Chou and J.M. Maciejowski. System identification using balanced parameterizations. *IEEE Transactions on Automatic Control*, 42:965–974, July 1997.
- [CMT87] D.W. Clarke, C. Mohtadi, and P.S. Tuffs. Generalised predictive control — Parts I and II. *Automatica*, 23:137–160, 1987.
- [CN56] J.F. Coales and A.R.M. Noton. An on-off servo mechanism with predicted change-over. *Proc. Institution of Electrical Engineers, Part B*, 103:449–462, 1956.
- [CR80] C.R. Cutler and B.L. Ramaker. Dynamic matrix control — a computer control algorithm. In *Proceedings, Joint American Control Conference*, San Francisco, 1980.
- [CRSV90] J.E. Cuthrell, D.E. Rivera, W.J. Schmidt, and J.A. Vergeais. Solution to the Shell standard control problem. In Prett *et al* [PGR90].
- [CS83] T.S. Chang and D.E. Seborg. A linear programming approach to multivariable feedback control with inequality constraints. *International Journal of Control*, 37:583–597, 1983.
- [DG91] H. Demircioglu and P.J. Gawthrop. Continuous-time generalised predictive control. *Automatica*, 27:55–74, 1991.
- [DG92] H. Demircioglu and P.J. Gawthrop. Multivariable continuous-time generalised predictive control. *Automatica*, 28:697–713, 1992.
- [DH99] C.E.T. Dorea and J.C. Hennet. (A,B)-invariant polyhedral sets of linear discrete-time systems. *Journal of Optimization Theory and Applications*, 103:521–542, 1999.

- [DOP95] F.J. Doyle, B.A. Ogunnaike, and R.K. Pearson. Nonlinear model-based control using second-order volterra models. *Automatica*, 31:697–714, 1995.
- [Doy78] J.C. Doyle. Guaranteed margins for LQG regulators. *IEEE Transactions on Automatic Control*, AC-23:756–757, 1978.
- [DS81] J.C. Doyle and G. Stein. Multivariable feedback design: concepts for a classical/modern synthesis. *IEEE Transactions on Automatic Control*, AC-26:4–16, 1981.
- [Fle87] R.R. Fletcher. *Practical Methods of Optimization*. Wiley, 2nd edition, 1987.
- [FPEN94] G.F. Franklin, J.D. Powell, and A. Emami-Naeini. *Feedback Control of Dynamic Systems*. Addison-Wesley, 3rd edition, 1994.
- [Fra74] P.M. Frank. *Entwurf von Regelkreisen mit vorgeschriebenem Verhalten*. G. Braun, Karlsruhe, 1974.
- [GBMR98] M. Gopinathan, J.D. Boskovic, R.K Mehra, and C. Rago. A multiple model predictive scheme for fault-tolerant flight control design. In *Proc. 37th IEEE Conference on Decision and Control*, Tampa, USA, 1998.
- [GDSA98] P.J. Gawthrop, H. Demircioglu, and I. Siller-Alcala. Multivariable continuous-time generalised predictive control: a state-space approach to linear and nonlinear systems. *IEE Proceedings, Part D*, 145:241–250, 1998.
- [GK95] E.G. Gilbert and I. Kolmanovsky. Maximal output admissible sets for discrete-time systems with disturbance inputs. In *Proceedings, American Control Conference*, pages 2000–2005, 1995.
- [GKT95] E.G. Gilbert, I. Kolmanovsky, and K.T. Tan. Discrete-time reference governors and the nonlinear control of systems with state and control constraints. *International Journal of Robust and Nonlinear Control*, 5:487–504, 1995.
- [GL89] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 2nd edition, 1989.
- [GM82] C.E. Garcia and M. Morari. Internal model control 1. a unifying review and some new results. *Ind.Eng.Chem. Process Design and Development*, 21:308–323, 1982.
- [GM86] C.E. Garcia and A.M. Morshedi. Quadratic programming solution of dynamic matrix control (QDMC). *Chemical Engineering Communications*, 46:73–87, 1986.
- [GMR99] M. Gopinathan, R.K Mehra, and J.C. Runkle. A model predictive fault-tolerant control scheme for hot isostatic pressing furnaces. In *Proc. American Control Conference*, San Diego, 1999.

- [GMW81] P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, London, 1981.
- [GN93] H. Genceli and M. Nikolau. Robust stability analysis of constrained ℓ_1 -norm model predictive control. *American Institute of Chemical Engineers' Journal*, 39:1954–1965, 1993.
- [GT91] E.G. Gilbert and K.T. Tan. Linear systems with state and control constraints: the theory and application of maximal output admissible sets. *IEEE Transactions on Automatic Control*, 36:1008–1020, 1991.
- [GZ92] G. Gattu and E. Zafiriou. Nonlinear quadratic dynamic matrix control with state estimation. *Ind. Eng. Chem. Research*, 31:1096–1104, 1992.
- [HM97] M. Huzmezan and J.M. Maciejowski. Notes on filtering, robust tracking and disturbance rejection used in model based predictive control schemes. In *Preprints, 4th IFAC Conference on System Structure and Control, Bucharest, October*, pages 238–243, 1997.
- [HM98] M. Huzmezan and J.M. Maciejowski. Reconfiguration and scheduling in flight using quasi-LPV high-fidelity models and MBPC control. In *Proc. American Control Conference*, Philadelphia, 1998.
- [Kai80] T. Kailath. *Linear Systems*. Prentice Hall, 1980.
- [Kal64] R.E. Kalman. When is a linear control system optimal? *Journal of Basic Engineering (Trans. ASME D)*, 86:51–60, 1964.
- [KBM96] M.V. Kothare, V. Balakrishnan, and M. Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32, 1996.
- [KBM⁺00] E.C. Kerrigan, A. Bemporad, D. Mignone, M. Morari, and J.M. Maciejowski. Multi-objective prioritisation and reconfiguration for the control of constrained hybrid systems. In *Proceedings, American Control Conference*, Chicago, June 2000.
- [KG88] S.S. Keerthi and E.G. Gilbert. Optimal infinite-horizon feedback laws for a general class of constrained discrete-time systems: stability and moving-horizon approximations. *Journal of Optimization Theory and Applications*, 57:265–293, 1988.
- [KGC96] J.C. Kantor, C.E. Garcia, and B. Carnahan, editors. *Conference on Chemical Process Control, CPC V*, Tahoe City, 1996.
- [Kle70] D. Kleinman. An easy way to stabilise a linear constant system. *IEEE Transactions on Automatic Control*, AC-15, 1970.
- [KM00a] E.C. Kerrigan and J.M. Maciejowski. Invariant sets for constrained nonlinear discrete-time systems with application to feasibility in model predictive control. In *Proceedings, IEEE Control and Decision Conference*, Sydney, December 2000.

- [KM00b] E.C. Kerrigan and J.M. Maciejowski. Soft constraints and exact penalty functions in model predictive control. In *Control 2000 Conference*, Cambridge, September 2000.
- [KP75] W.H. Kwon and A.E. Pearson. On the stabilization of a discrete constant linear system. *IEEE Transactions on Automatic Control*, AC-20, 1975.
- [KP77] W.H. Kwon and A.E. Pearson. A modified quadratic cost problem and feedback stabilization of a linear system. *IEEE Transactions on Automatic Control*, AC-22:838–842, 1977.
- [KR72] H. Kwakernaak and R. Sivan. *Linear Optimal Control Systems*. Wiley, New York, 1972.
- [KR92] B. Kouvaritakis and J.A. Rossiter. Stable generalized predictive control. *IEE Proceedings, Part D*, 139:349–362, 1992.
- [KRSar] B. Kouvaritakis, J.A. Rossiter, and J. Schuurmans. Efficient robust predictive control. *IEEE Transactions on Automatic Control*, vol. 45, no. 8, pages 1545–1549, August 2000.
- [Kun78] S.Y. Kung. A new low-order approximation algorithm via singular value decomposition. In *Proc. 12th Asilomar Conf. on Circuits, Systems and Computers*, 1978.
- [LC96] J.H. Lee and B. Cooley. Recent advances in model predictive control and other related areas. In Kantor *et al* [KGC96].
- [LD99] G.P. Liu and S. Daley. Output-model-based predictive control of unstable combustion systems using neural networks. *Control Engineering Practice*, 7:591–600, 1999.
- [Lee00] J.H. Lee. Modeling and identification for nonlinear model predictive control: requirements, current status and future research needs. In Allgöwer and Zheng [AZ00], pages 269–293.
- [LEL92] M.J. Lieberman, T.F. Edgar, and L.S. Lasdon. Efficient data reconciliation and estimation for dynamic processes using nonlinear programming techniques. *Computers in Chemical Engineering*, 16:963–986, 1992.
- [LGM92] J.H. Lee, M.S. Gelormino, and M. Morari. Model predictive control of multi-rate sampled-data systems: a state-space approach. *International Journal of Control*, 55:153–191, 1992.
- [Lju89] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1989.
- [LKB98] G.P. Liu, V. Kadirkamanathan, and S.A. Billings. Predictive control for non-linear systems using neural networks. *International Journal of Control*, 71:1119–1132, 1998.

- [LLF89] S. Li, K.Y. Lim, and D.G. Fisher. A state space formulation for model predictive control. *American Institute of Chemical Engineers' Journal*, 35:241–249, 1989.
- [LLMS95] P. Lundström, J.H. Lee, M. Morari, and S. Skogestad. Limitations of dynamic matrix control. *Computers in Chemical Engineering*, 19:409–421, 1995.
- [LMG94] J.H. Lee, M. Morari, and C.E. Garcia. State-space interpretation of model predictive control. *Automatica*, 30:707–717, 1994.
- [LP95] P. Lu and B.L. Pierson. Aircraft terrain following based on a nonlinear predictive control approach. *Journal of Guidance, Control and Dynamics*, 18:817–823, 1995.
- [LY94] J.H. Lee and Z.H. Yu. Tuning of model predictive controllers for robust performance. *Computers in Chemical Engineering*, 18:15–37, 1994.
- [LY97] J.H. Lee and Z.H. Yu. Worst-case formulations of model predictive control for systems with bounded parameters. *Automatica*, 33:763–781, 1997.
- [Mac85] J.M. Maciejowski. Asymptotic recovery for discrete-time systems. *IEEE Transactions on Automatic Control*, AC-30:602–605, 1985.
- [Mac89] J.M. Maciejowski. *Multivariable Feedback Design*. Addison-Wesley, Wokingham, UK, 1989.
- [Mac97] J.M. Maciejowski. Reconfiguring control systems by optimization. In *Proc. 4th European Control Conference, Brussels*, 1997.
- [Mac98] J.M. Maciejowski. The implicit daisy-chaining property of constrained predictive control. *Applied Mathematics and Computer Science*, 8:695–711, 1998.
- [Mat] The Mathworks. *The LMI Toolbox for Matlab*.
- [May96] D.Q. Mayne. Nonlinear model predictive control: an assessment. In Kantor *et al* [KGC96].
- [MHER95] E.S. Meadows, M.A. Henson, J.W. Eaton, and J.B. Rawlings. Receding horizon control and discontinuous state feedback stabilization. *International Journal of Control*, 62:1217–1229, 1995.
- [MKGW00] R.H. Miller, I. Kolmanovsky, E.G. Gilbert, and P.D. Washabaugh. Control of constrained nonlinear systems: a case study. *IEEE Control Systems Magazine*, 20(1):23–32, February 2000.
- [ML98] M. Mahfouf and D.A. Linkens. *Generalised Predictive Control and Bioengineering*. Taylor & Francis, 1998.

- [ML99] M. Morari and J.H. Lee. Model predictive control: past, present and future. *Computers and Chemical Engineering*, 23:667–682, 1999.
- [MM90] D.Q. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Transactions on Automatic Control*, AC-35:814–824, 1990.
- [MM93] H. Michalska and D.Q. Mayne. Robust receding horizon control of constrained nonlinear systems. *IEEE Transactions on Automatic Control*, 38:1623–1633, 1993.
- [Mor94] M. Morari. Model predictive control: multivariable control technique of choice in the 1990s? In Clarke [Cla94].
- [Mos95] E. Mosca. *Optimal, Predictive, and Adaptive Control*. Prentice Hall, 1995.
- [MR93a] K.R. Muske and J.B. Rawlings. Linear model predictive control of unstable processes. *Journal of Process Control*, 3:85, 1993.
- [MR93b] K.R. Muske and J.B. Rawlings. Model predictive control with linear models. *American Institute of Chemical Engineers' Journal*, 39:262–287, 1993.
- [MR95] M. Morari and N.L. Ricker. *Model Predictive Control Toolbox: User's Guide*. The Mathworks, 1995.
- [MRRS00] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: stability and optimality. *Automatica*, 36:789–814, 2000.
- [MS97a] L. Magni and R. Sepulchre. Stability margins of nonlinear receding-horizon control via inverse optimality. *Systems and Control Letters*, 32:241–245, 1997.
- [MS97b] D.Q. Mayne and W.R. Schroeder. Robust time-optimal control of constrained linear systems. *Automatica*, 33:2103–2118, 1997.
- [MZ88] M. Morari and E. Zafiriou. *Robust Process Control*. Prentice Hall, 1988.
- [MZ89] E. Mosca and G. Zappa. ARX modeling of controlled ARMAX plants and LQ adaptive controllers. *IEEE Transactions on Automatic Control*, 34:371–375, 1989.
- [NL89] R.B. Newell and P.L. Lee. *Applied Process Control — A Case Study*. Prentice Hall, New York, 1989.
- [NN94] J.E. Nesterov and A.S Nemirovsky. *Interior Point Polynomial Methods in Convex Programming: Theory and Applications*. SIAM, Philadelphia, 1994.

- [Nor86] J.P. Norton. *An Introduction to Identification*. Academic Press, New York, 1986.
- [NP97] V. Nevistic and J.A. Primbs. Finite receding horizon control: A general framework for stability and performance analysis. Technical Report AUT 97-06, Automatic Control Laboratory, ETH Zurich, 1997.
- [NRPH00] M. Nørgaard, O. Ravn, N.K. Poulsen, and L.K. Hansen. *Neural Networks for Modelling and Control of Dynamic Systems*. Springer, London, 2000.
- [OB94] N.M.C. de Oliveira and L.T. Biegler. Constraint handling and stability properties of model predictive control. *American Institute of Chemical Engineers' Journal*, 40:1138–1155, 1994.
- [OC93] A.W. Ordys and D.W. Clarke. A state-space description for GPC controllers. *International Journal of Systems Science*, 24:1727–1744, 1993.
- [PG88] D.M. Prett and C.E. Garcia. *Fundamental Process Control*. Butterworths, Boston, 1988.
- [PGR90] D.M. Prett, C.E. Garcia, and B.L. Ramaker, editors. *The Second Shell Process Control Workshop*. Butterworths, Boston, 1990.
- [PHGM97] G. Papageorgiou, M. Huzmezan, K. Glover, and J.M. Maciejowski. Combined MBPC/ H_∞ autopilot for a civil aircraft. In *American Control Conference*, 1997.
- [PLR⁺98] K. Preuß, M-V. LeLann, J. Richalet, M. Cabassud, and G. Casamatta. Thermal control of chemical batch reactors with predictive functional control. *Journal A*, 39:13–20, 1998.
- [PM87] D.M. Prett and M. Morari, editors. *The Shell Process Control Workshop. Process Control Research: Industrial and Academic Perspectives*. Butterworths, Boston, 1987.
- [PRC82] D.M. Prett, B.L. Ramaker, and C.R. Cutler. Dynamic matrix control method. *United States Patent 4349869*, 1982.
- [Pro63] A.I. Propoi. Use of linear programming methods for synthesising sampled-data automatic systems. *Automation and Remote Control*, 24:837–844, 1963.
- [PSRJG00] S. Piché, B. Sayyar-Rodsari, D. Johnson, and M. Gerules. Nonlinear model predictive control using neural networks. *IEEE Control Systems Magazine*, 20(3):53–62, June 2000.
- [QB96] S.J. Qin and T.A. Badgwell. An overview of industrial model predictive control technology. In Kantor *et al* [KGC96], pages 232–256.
- [QB99] S.J. Qin and T.A. Badgwell. An overview of nonlinear model predictive control applications. In Allgöwer and Zheng [AZ00], pages 369–392.

- [Ray89] W. H. Ray. *Advanced Process Control*. Butterworths, Boston, 1989.
- [ReADAK87] J. Richalet, S. Abu el Ata-Doss, C. Arber, and H.B. Kuntze. Predictive functional control: application to fast and accurate robots. In *Proc. IFAC World Congress, Munich*, 1987.
- [Ric85] N.L. Ricker. Use of quadratic programming for constrained internal model control. *Ind. Eng. Chem. Process Design and Development*, 24:925–936, 1985.
- [Ric90] N.L. Ricker. Model predictive control with state estimation. *Ind. Eng. Chem. Research*, 29:374–382, 1990.
- [Ric91] J. Richalet. *Pratique de l'Identification*. Editions Hermès, Paris, 1991.
- [Ric93a] J. Richalet. Industrial applications of model based predictive control. *Automatica*, 29, 1993.
- [Ric93b] J. Richalet. *Pratique de la Commande Prédictive*. Editions Hermès, Paris, 1993.
- [RK93] J.A. Rossiter and B. Kouvaritakis. Constrained stable generalized predictive control. *Proc. Institution of Electrical Engineers, Part D*, 140:243–254, 1993.
- [RM82] R. Rouhani and R.K. Mehra. Model algorithmic control (MAC): basic theoretical properties. *Automatica*, 18:401–414, 1982.
- [RM93] J.B. Rawlings and K.R. Muske. The stability of constrained receding horizon control. *IEEE Transactions on Automatic Control*, 38:1512–1516, 1993.
- [RM00] C.A. Rowe and J.M. Maciejowski. Tuning MPC using H_∞ loop shaping. In *Proceedings, American Control Conference*, Chicago, 2000.
- [Ros71] H.H. Rosenbrock. Good, bad or optimal? *IEEE Transactions on Automatic Control*, 1971.
- [Row97] C.A. Rowe. The fault-tolerant capabilities of constrained model predictive control. M.Phil. thesis, Cambridge Univeristy, 1997.
- [RPG92] D.E. Rivera, J.F. Pollard, and C.E. Garcia. Control-relevant prefiltering: a systematic design approach and case study. *IEEE Transactions on Automatic Control*, 37:964–974, 1992.
- [RR99a] C.V. Rao and J.B. Rawlings. Nonlinear moving horizon estimation. In Allgöwer and Zheng [AZ00].
- [RR99b] C.V. Rao and J.B. Rawlings. Steady states and constraints in model predictive control. *American Institute of Chemical Engineers' Journal*, 45:1266–1278, 1999.

- [RRK98] J.A. Rossiter, M.J. Rice, and B. Kouvaritakis. A numerically robust state-space approach to stable predictive control strategies. *Automatica*, 34:65–73, 1998.
- [RRTP78] J. Richalet, A. Rault, J.L. Testud, and J. Papon. Model predictive heuristic control: applications to industrial processes. *Automatica*, 14:413–428, 1978.
- [RTV97] C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and Algorithms for Linear Optimization — An Interior Point Approach*. Wiley, Chichester, 1997.
- [RWR98] C.V. Rao, S.J. Wright, and J.B. Rawlings. Application of interior-point methods to model predictive control. *Journal of Optimization Theory and Applications*, 99:723–757, 1998.
- [Rya82] E.P. Ryan. *Optimal Relay and Saturating Control System Synthesis*. Peter Peregrinus, Stevenage, UK, 1982.
- [SA77] M.G. Safonov and M. Athans. Gain and phase margin for multiloop LQG regulators. *IEEE Transactions on Automatic Control*, AC-22:173–179, 1977.
- [SA87] G. Stein and M. Athans. The LQG/LTR procedure for multivariable feedback control design. *IEEE Transactions on Automatic Control*, 32:105–113, 1987.
- [San76] J.M. Martin Sanchez. Adaptive predictive control system. *United States Patent 4197576*, 1976.
- [San98] D.J. Sandoz. Perspectives on the industrial exploitation of model predictive control. *Measurement + Control*, 31:113–117, 1998.
- [SB98] G.J. Sutton and R.R. Bitmead. Experiences with model predictive control applied to a nonlinear constrained submarine. In *Proceedings, 37th IEEE Conference on Decisions and Control*, Tampa, Florida, December 1998.
- [Sco97] P.O.M. Scokaert. Infinite horizon generalized predictive control. *International Journal of Control*, 66:161–175, 1997.
- [SCS93] A. Saberi, B.M. Chen, and P. Sannuti. *Loop Transfer Recovery: Analysis and Design*. Springer, 1993.
- [SD94] P.O.M. Scokaert and D.W. Clarke. Stability and feasibility in constrained predictive control. In Clarke [Cla94].
- [SM98] P.O.M. Scokaert and D.Q. Mayne. Min-max feedback model predictive control for constrained linear systems. *IEEE Transactions on Automatic Control*, 43:1136–1142, 1998.
- [Smi57] O.J.M. Smith. Close control of loops with deadtime. *Chemical Engineering Progress*, 53:217, 1957.

- [SMR99] P.O.M. Scokaert, D.Q. Mayne, and J.B. Rawlings. Suboptimal model predictive control (feasibility implies stability). *IEEE Transactions on Automatic Control*, 44:648–654, 1999.
- [SMS92] D.S. Shook, C. Mohtadi, and S.L. Shah. A control-relevant identification strategy for GPC. *IEEE Transactions on Automatic Control*, 37:975–980, 1992.
- [Soe92] R. Soeterboek. *Predictive Control: a Unified Approach*. Prentice Hall, New York, 1992.
- [SR95] P.O. Scokaert and J.B. Rawlings. Receding horizon recursive state estimation. In *Proc. IFAC Symposium on Nonlinear Control Systems*, Tahoe City, USA, 1995.
- [SR96a] J.M. Martin Sanchez and J. Rodellar. *Adaptive Predictive Control*. Prentice Hall, London, 1996.
- [SR96b] P.O. Scokaert and J.B. Rawlings. Infinite horizon linear quadratic control with constraints. In *Proceedings, IFAC World Congress*, pages 109–114, San Francisco, July 1996.
- [SR99] P.O.M. Scokaert and J.B. Rawlings. Feasibility issues in linear model predictive control. *American Institute of Chemical Engineers' Journal*, 45:1649–1659, 1999.
- [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*, 18:1023–1028, 1995.
- [SW97] H.J. Sussmann and J.C. Willems. 300 years of optimal control: from the brachystochrone to the maximum principle. *IEEE Control Systems Magazine*, 17:32–44, 1997.
- [TC88] T.T.C. Tsang and D.W. Clarke. Generalised predictive control with input constraints. *IEE Proceedings, Part D*, 135:451–460, 1988.
- [Ter96] T. Terlaky, editor. *Interior Point Methods of Mathematical Programming*. Kluwer Academic, Dordrecht, 1996.
- [TM99] M.L. Tyler and M. Morari. Propositional logic in control and monitoring problems. *Automatica*, 35:565–582, 1999.
- [vdBdV95] T.J.J. van den Boom and R.A.J. de Vries. Constrained predictive control using a time-varying Youla parameter: a state space solution. In *Proceedings, European Control Conference*, Rome, 1995.
- [vOdM96] P. van Overschee and B. de Moor. *Subspace Identification for Linear Systems: Theory, Implementation, Applications*. Kluwer Academic, 1996.

- [VSJ99] J. Vada, O. Slupphaug, and T.A. Johansen. Efficient infeasibility handling in linear MPC subject to prioritised constraints. In *Proceedings, European Control Conference*, Karlsruhe, Germany, 1999.
- [Won74] M. Wonham. *Linear Multivariable Systems*. Springer, Berlin, 1974.
- [Wri96] S.J. Wright. Applying new optimization algorithms to model predictive control. In Kantor *et al* [KGC96].
- [Wri97] S.J. Wright. *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia, 1997.
- [YP93] T.H. Yang and E. Polak. Moving horizon control of nonlinear systems with input saturations, disturbances and plant uncertainties. *International Journal of Control*, pages 875–903, 1993.
- [Zaf90] E. Zafiriou. Robust model predictive control of processes with hard constraints. *Computers in Chemical Engineering*, 14:359–371, 1990.
- [ZAN73] V. Zakian and U. Al-Naib. Design of dynamical and control systems by the method of inequalities. *Proceedings, Institution of Electrical Engineers*, 120:1421–1427, 1973.
- [ZB93] Y.C. Zhu and T. Backx. *Identification of Multivariable Industrial Processes for Simulation, Diagnosis and Control*. Springer, London, 1993.
- [ZC96] E. Zafiriou and H.-W. Chiou. On the dynamic resiliency of constrained processes. *Computers in Chemical Engineering*, 20:347–355, 1996.
- [ZDG96] K. Zhou, J.C. Doyle, and K. Glover. *Robust and Optimal Control*. Prentice Hall, New York, 1996.
- [Zhu98] Y. Zhu. Multivariable process identification for MPC: the asymptotic method and its applications. *Journal of Process Control*, 8:101–115, 1998.
- [ZM74] H.P. Zeiger and A.J. McEwen. Approximate linear realization of given dimension via Ho's algorithm. *IEEE Transactions on Automatic Control*, AC-19:153, 1974.
- [ZM93] A. Zheng and M. Morari. Robust stability of constrained model predictive control. In *Proceedings, American Control Conference, San Francisco*, pages 379–383, 1993.
- [ZM95a] A. Zheng and M. Morari. On control of linear unstable systems with constraints. In *Proceedings, American Control Conference, Seattle*, 1995.
- [ZM95b] A. Zheng and M. Morari. Stability of model predictive control with mixed constraints. *IEEE Transactions on Automatic Control*, 40:1818–1823, 1995.

Appendix A

Some commercial MPC products

A.1	Aspentech: <i>DMCPlus</i>	306
A.2	Honeywell: <i>RMPCT</i>	308
A.3	Simulation Sciences: <i>Connoisseur</i>	309
A.4	Adersa: <i>PFC</i> and <i>HIECON</i>	311
A.5	ABB: <i>3dMPC</i>	311
A.6	Pavilion Technologies Inc.: <i>Process Perfecter</i>	311

In this appendix we describe briefly some aspects of commercial predictive control products available from several vendors: Aspentech's *DMCPlus*, Honeywell's *RMPCT*, Simulation Sciences Inc's *Connoisseur*, Adersa's *PFC* and *HIECON*, ABB's *3dMPC*. Most of these products are designed for the needs of the process industries, and they have been used predominantly in the petrochemical sector, whereas Adersa's products — particularly *PFC* — have been developed for a wider range of applications, including servomechanisms with fast dynamics. We concentrate here on relating these products to the theoretical material presented earlier in this book — what cost functions are used, what constraints can be defined and how are they treated, what are the tuning parameters. But it should be appreciated that much of their value lies in aspects which are not emphasized here: how they interface with other control software, particularly the real-time distributed control systems used in the process industries, what kinds of modeling facilities they provide, what support for commissioning and maintenance is provided by the vendors, what user interfaces they provide, etc. In particular, most of these products are very strongly influenced by the 'culture' of control practice in the process industries — this is predominantly a culture in which human operators are entrusted with running plants, in which they are accustomed to having considerable authority over how their plants are controlled, and in which they expect to remain 'in the loop' at some level. All the

products go to impressive lengths to support plant operators in their traditional role, rather than trying to replace them.

Usually these predictive control products are implemented on top of a conventional control layer. That is, the outputs of the predictive control algorithms are usually set-points of conventional low-level controllers, such as flow controllers, temperature controllers, and pressure controllers. The objective of all the predictive control products is to increase profitability of operation beyond that obtained by the existing conventional controllers. So if unusual conditions are encountered, in which a predictive controller has trouble controlling the plant, then the operators have the option of switching the controller off — which does not mean that the plant is left uncontrolled, but that it is returned to the relative safety of conventional low-level control. It will be appreciated that if operated in this way, some of the issues treated earlier in the book, such as guarantees of closed-loop stability, are not of paramount importance.

These products, like most commercial products, undergo continuous development. The descriptions given here are believed to be accurate at the time of writing, and reflect the author's understanding of the products. *Their inclusion here does not imply any recommendation by the author or the publisher, and the descriptions are not claimed to be complete, as regards either the technical or the commercial features of the products.* It is hoped to update the information from time to time, and perhaps expand the range of products covered, on this book's web site, which contains links to the vendors of these and other predictive control products.

A.1 Aspentech: *DMCPlus*

DMCPlus is a derivative of the product *DMC*, which is an acronym for *Dynamic Matrix Control*. *DMC* was originally developed and patented by *Shell* [CR80], then further developed and marketed by the *DMC Corporation*, a company which was later acquired by *Aspentech*. *DMC* and *DMCPlus* are the most widely known and used of the commercially available predictive control products, with particularly deep penetration of the petrochemicals sector. A variant of *DMC*, known as *Quadratic DMC* or *QDMC*, was also developed by *Shell*, and there are some references to it in the literature [GM86], but the commercial rights to it have been retained by *Shell*.

DMCPlus uses multivariable finite step response models to represent plant models, as described in Chapter 4. An extension to these models allows integrators to be modelled as well as asymptotically stable plants. When making predictions, the disturbance model referred to earlier as the *DMC model* is used, namely any error between the prediction of a current output and its latest measurement is assumed to be due to an output disturbance, and that disturbance is assumed to remain unchanged in the future. For variables which are outputs of integrators, the operator is also allowed to define a 'rotation factor'. This is essentially the slope of a line used to correct the predictions of integrated outputs, so it is a means of correcting mismodelled integrator gains.

The predictive control law is computed in *DMCPlus* in two steps. First a set of steady-state 'target' levels or set-points is computed for the controlled variables (outputs). These are chosen on economic grounds, the operators having the ability to supply information on the current economic value of achieving product of a given quality, and on the current

economic costs of the input variables (typically flows of expensive goods). The cost function is linear in this step (which is possible because of the use of a linear model). Input and output constraints are imposed and respected in this step. These are in the form of linear inequality constraints, so that the target levels are chosen by solving an LP problem. It should be emphasized that, although a ‘steady-state’ problem is being solved in this first step, and so only the steady-state gain information from the model is needed, this problem is re-solved at the same rate as the dynamic control, and it is re-solved on the basis of the latest measurements. In order to solve a meaningful optimization problem, the LP solver must know the ‘current’ steady state. But at any particular time, the process is unlikely to be in steady state, so the dynamic model is used to compute what the steady state would be if the inputs all remained at their current levels and this steady state is taken as the ‘current’ one.

In the second step, a future control trajectory is computed, over a control horizon, which is predicted to bring the plant outputs to the target levels computed by the LP optimization. This is computed as the solution of a least-squares problem, which includes quadratic penalization of control moves (referred to in *DMCPlus* as ‘move suppression’). Constraints are not imposed explicitly, but the penalty weights are increased as the predicted inputs and/or outputs approach constraints. So in effect ‘soft constraints’ are imposed, by means of a kind of ‘penalty function’ approach — see Chapter 5. The time horizon for driving the plant to the new targets is taken to be the same as the prediction horizon. However, the dimension of the optimization problem is kept relatively small by using ‘blocking’, namely non-uniform spacing of the ‘decision times’ during this horizon.

The operators are asked to define two kinds of *equal concern errors*, ‘steady-state’ and ‘dynamic’. Essentially, they are asked to specify how big a deviation of each controlled output beyond its constraint boundary causes them significant concern. The ‘steady-state’ equal concern errors are used in the target-setting step, if the LP problem is found to be infeasible. In this case one of two strategies is adopted: either a ‘minimum distance’ solution is found, which minimizes a weighted sum of distances of targets from their constraint boundaries, with the weights reflecting the equal concern errors; or the equal concern errors are used to define priorities for the constraints, and the constraints with lowest priority are allowed to be violated, until feasibility of the LP is re-established.

The ‘dynamic’ equal concern errors are used to implement soft constraints in the second step of computing the solution. If an output approaches its constraint boundary, then the weight associated with its error from its target in the least-squares problem is increased. At what point the increase begins, and how much the weight is increased, depends on the equal concern errors.

Some support for the control of nonlinear processes is provided, primarily by the availability of a number of invertible transformations, such as square-root or logarithmic transformations, and facilities to define others. These are useful when the process is essentially (close to) linear, except for nonlinear transformations of input and output variables, due to factors such as valve characteristics. Also a kind of gain-scheduling is available for nonlinear processes, providing that a nonlinear model is available. Such a model is used to estimate the steady-state gains at the current operating point, and these are used in the LP step of the control calculation.

A.2 Honeywell: RMPCT

The name *RMPCT* is an acronym for *Robust Multivariable Predictive Control Technology*, a product which has been developed by Honeywell.

A distinctive feature of *RMPCT* is that it implements the ‘zone’ or ‘funnel’ concept, in which the objective of control is to keep plant outputs within defined regions, rather than at specific set-points, as discussed in Section 5.5. Each controlled output has an allowed range or zone, defined by a lower limit and a higher limit. So long as its value is within this range, it is not penalized in any way. If an output is outside the range, a straight-line trajectory is constructed from its current value to the nearest edge of the allowed range; the output variable is supposed to follow this trajectory to the target zone. The time taken by this trajectory to hit the allowed range is pre-determined for each controlled output, and it varies between 0 and twice the open-loop settling time of that output. Whenever the output is predicted to lie outside the ‘funnel’ formed by the target zone and this trajectory, its distance from the edge of the funnel is defined to be an error. A quadratic cost function is formed as a weighted sum-of-squares of these errors. Future inputs are chosen so as to minimize this cost function, subject to constraints on input levels and moves. The optimization problem to be solved is a QP problem, and an active set solution strategy is used.

It can be seen that the ‘aggressiveness’ with which an output is returned to its target zone is determined by the slope of the edge of the ‘funnel’ which is constructed to bring it back into the allowed range. Thus this edge serves as a straight-line reference trajectory, combining this concept with the funnel concept. Conventional set-point control of a variable is obtained by making the low and high limits the same for that variable.

Feedforward to combat the effects of measured disturbances is handled in a similar way. The effects of such disturbances on the controlled outputs are predicted, and any output driven outside its allowed range is again driven back by reference to a ‘funnel’. But typically the funnels for feedforward slope more steeply than those for recovery from unmeasured disturbances — namely those for correction by feedback action — because feedforward action, being outside any feedback loop, can be more aggressive without risk of becoming unstable.

Like *DMCPlus*, the *RMPCT* controller is also organized in two levels, one for optimization and one for control. A steady-state model is used to find the optimal steady-state for the process, on the assumption that the ‘current steady-state’ is the currently predicted steady state. The cost function used for this purpose is a combination of the traditional linear cost function which captures the economic benefits of producing particular quantities of various products and the economic costs of using control inputs, with a quadratic function which represents the costs of moving away from the currently predicted steady-state inputs and outputs. Process variables can appear in this cost function which do not appear in the cost function used for dynamic control, and vice versa. The steady-state optimization can be performed for several process units simultaneously, and the results passed down to the individual dynamic controllers. The interaction between the steady-state and dynamic optimization is not the conventional one, of simply passing steady-state objectives to the dynamic controller. Rather, the new steady-state values are included in the dynamic optimizations, but with a variable

weighting which allows the ‘strength’ with which the process is pushed to the new steady-state to be varied. If it is recalled that *RMPCT* encourages most controlled outputs to have target zones rather than set-points, it will be appreciated that putting a low weighting on the steady-state solution will result in the controller concentrating on correcting any predicted excursions outside the target zones, but pushing the process to the desirable steady-state when no such excursions are predicted.

The net result of this combination of steady-state and dynamic optimization is soft constraints on controlled variables, with an inexact penalty function — that is, in some circumstances zone boundary violations may occur even if there is a feasible trajectory which avoids them.

RMPCT allows ‘blocking’ of the predicted control moves, so that the moves computed by the optimization are not assumed to be spaced at equal intervals in the control horizon.

The dynamic model used by *RMPCT* is a multivariable ARX model.

RMPCT attempts to increase the robustness of its controller to modelling errors by using a number of strategies. The first of these is to introduce scaling of the input and output variables, in order to optimize the conditioning of the optimization problem. The idea is to reduce the sensitivity of the output trajectories to model errors. A second strategy is to approximate the prediction model by one with better conditioning, if the original model is ill-conditioned with respect to inversion. This is done by identifying the active constraint set involved in the optimization, performing a singular value decomposition (SVD) of the corresponding matrix, and approximating it by discarding any very small singular values.¹ The third strategy is adjusting the internal model used by the controller. The controller corresponding to a particular internal model is assessed by using simulation to evaluate the ‘ISE’, or ‘Integrated Squared Error’, criterion for a range of plant model errors. The internal model is then adjusted in a kind of extremum-seeking process until one is found for which the worst-case ISE criterion is the smallest.

A.3 Simulation Sciences: Connoisseur

Connoisseur has three control modes:

1. ‘LR’ or ‘Long-Range’ mode,
2. ‘QP-LR’ mode, a mixture of *LR* mode with *QP* mode,
3. *QP* mode, in which a *QP* problem is solved, although one with soft constraints.

In *LR* mode, a quadratic cost function is optimized, ignoring constraints at first. Since a linear model is used, as usual, this is a ‘linear-quadratic’ (‘LQ’) problem, for which a closed-form solution exists. This can be obtained by solving a Riccati equation, from which a fixed-gain state feedback matrix can be obtained — see Mini-Tutorial 7. The finite-horizon Riccati equation is iterated to convergence, so in effect an infinite-horizon feedback law is applied. This approach does not require open-loop predictions to be computed, but closed-loop predictions, with the fixed-gain feedback law in place, are

¹ Actually *RMPCT* computes a ‘URV’ decomposition rather than an SVD, in which *U* and *V* are orthogonal, as in SVD, but *R* is triangular rather than diagonal.

computed, in order to see whether any constraints are likely to be violated. If such violations are predicted, then the weights in the cost function are modified until no constraint violations are predicted.

In *QP-LR* mode, a quadratic cost function is optimized, taking into account (hard) input constraints, while output constraints are ignored. As in *LR* mode, weights are adjusted to remove any predicted output constraint violations. In *QP* mode, both input and output constraints are included in the optimization problem. All constraints are ‘soft’, an exact penalty function being used, with weights used to define constraint violation priorities.

The motivation for having these different modes is reducing computational effort. For example, in *LR* mode, the feedback gains need to be found only once, although the iterative adjustment of set-points (if necessary) of course requires additional computation. In *QP-LR* mode, the number of constraints is smaller than in *QP* mode, so the optimal solution is easier to compute.

The quadratic cost function can be defined to penalize both control moves, and deviations of the control input values from their target values, if these are defined.

As is the case with the other products summarized here, the optimization algorithm assumes uneven intervals between predicted control moves, or ‘blocking factors’, over the control horizon.

Connoisseur uses internal models in the *ARX* form, similar to those used by *RMPCT*. Recursive least-squares is used to estimate the order and parameters of the model. Adaptive control is possible, the recursive estimator being used to update the model continuously.

The model is of the form

$$A(z)y(k) = B(z)\Delta u(k) + C(z)\Delta v(k) \quad (\text{A.1})$$

where $A(z)$, $B(z)$, and $C(z)$ are matrices of z transform polynomials (or, equivalently, polynomials in the difference operator), and $v(k)$ is a vector of measured disturbance variables. Disturbances are not modelled explicitly, but when predictions are made, the model is used to predict output changes relative to the latest measured outputs:

$$A(z)[\hat{y}(k+i) - y(k)] = B(z)\Delta \hat{u}(k+i) + C(z)\Delta \hat{v}(k+i) \quad (\text{A.2})$$

This is the same procedure as used in *DMCPlus*, and is equivalent to adopting the ‘DMC disturbance model’, namely assuming that all disturbances are output disturbances which will remain unchanged in the future.

The actual model used is slightly modified from the form given above, because different sampling intervals are permitted for the input and output variables (and measured disturbances).

Optimization of set-points is done by solving an LP problem, at a greater interval than the control update interval. Typically, if the control update interval is a few seconds, the interval between re-computing optimal set-points would be a few minutes. As usual, the LP problem does not consider information about the dynamics of the process, so it needs to be given a ‘current’ steady-state. This is assumed to be equal to the current set-points, since it is assumed that the plant would eventually converge to

these if they remained unchanged. This set-point optimization can be done for the set-points of several dynamic controllers simultaneously, which achieves a degree of coordination between these controllers, and should give better economic performance than optimizing the set-points separately would.

A.4 Adersa: PFC and HIECON

PFC is primarily a methodology, a particular way of doing predictive control, as described in Section 5.6, but it is also a product of the French company Adersa, which implements this methodology. It differs from the other products described here in not being originally developed for, or particularly targeted at, the petrochemical industry, or even at process industries at all. As a result, it has been applied in a wider range of applications than the other products.

A basic form of PFC is implemented in the *Quantum* programmable logic controller (PLC) available from *Modicon*. The implementation complies with the IEC 1131-3 standard for controller software.

HIECON is an alternative product, based on the same methodology, but aimed at multivariable applications, and offering a wider range of solution algorithms, including full constrained optimization.

Successful applications of the Adersa products range from process applications such as batch reactors, rolling mills, and milk dryers, to ‘servo’ applications such as fuel injection in car engines, gun turret control, and attitude control of launchers and missiles.

A.5 ABB: 3dMPC

ABB’s product uses state-space models and quadratic cost functions, so of all the products described here it is the one whose technical framework is the closest to the standard one described in this book. Hard constraints can be defined for inputs (manipulated variables) and their rates of change, and these can be supplemented by soft constraints which indicate desirable rather than mandatory ranges for the inputs. Soft constraints can be defined on outputs (controlled variables), and these can be prioritized.

Considerable support is available for system identification from operating data, including direct identification of the state-space model using subspace methods, as an option.

Nonlinear transformations of input and output signals are available, as in most of the other products.

A.6 Pavilion Technologies Inc.: Process Perfecter

Pavilion Technologies Inc. offer a predictive control product which is aimed primarily at applications in which the use of nonlinear models brings large advantages, or is for some reason unavoidable. It is based on ‘black box’ neural net models, and relies on

the assumption that all the significant nonlinearities show up as changes in steady-state gains. A more detailed account of the technical basis of this product is given in Section 10.3.3.

MATLAB program

basicmpc

This is the complete listing of *MATLAB* program `basicmpc`. It is also available from the web site: <http://www.booksites.net/maciejowski/>

```
%BASICMPC Basic Predictive Control without constraints. (Script file)
%
% Implements unconstrained MPC for stable SISO discrete-time system
% defined as LTI object.
%
% The following are editable parameters (most have defaults):
% Tref: Time constant of exponential reference trajectory
% Ts: Sampling interval
% plant: Plant definition (discrete-time SISO LTI object)
% model: Model definition (discrete-time SISO LTI object)
% P: Vector of coincidence points
% M: Control horizon
% tend: Duration of simulation
% setpoint: Setpoint trajectory (column vector) - length must exceed
%           no of steps in simulation by at least max(P).
% umpast, uppast, ympast, yppast: Initial conditions of plant & model.
%
% Assumes Matlab 5.3. Uses Control Toolbox LTI object class.
% No other toolboxes required.

%% J.M.Maciejowski, 8 March 1999. Revised 22.3.99, 15.12.99.
%% Copyright(C) 1999. All Rights Reserved.
%% Cambridge University Engineering Department.

%%%%%%%%%%%%%%% PROBLEM DEFINITION:
```

```
% Define time-constant of reference trajectory Tref:
Tref = 6;

% Define sampling interval Ts (default Tref/10):
if Tref == 0,
    Ts = 1;
else
    Ts = Tref/10;
end

% Define plant as SISO discrete-time 'lti' object 'plant'
%%%%% CHANGE FROM HERE TO DEFINE NEW PLANT %%%%
nump=1;
denp=[1,-1.4,0.45];
plant = tf(nump,denp,Ts);
%%%%% CHANGE UP TO HERE TO DEFINE NEW PLANT %%%%
plant = tf(plant); % Coerce to transfer function form
nump = get(plant,'num'); nump = nump{:}; % Get numerator polynomial
denp = get(plant,'den'); denp = denp{:}; % Get denominator polynomial
nnump = length(nump)-1; % Degree of plant numerator
ndenp = length(denp)-1; % Degree of plant denominator
if nump(1)^~0, error('Plant must be strictly proper'), end;
if any(abs(roots(denp))>1), disp('Warning: Unstable plant'), end

% Define model as SISO discrete-time 'lti' object 'model'
% (default model=plant):
%%%%% CHANGE FROM HERE TO DEFINE NEW MODEL %%%%
model = plant;
%%%%% CHANGE UP TO HERE TO DEFINE NEW MODEL %%%%
model = tf(model); % Coerce to transfer function form
numm = get(model,'num'); numm = numm{:}; % Get numerator polynomial
denm = get(model,'den'); denm = denm{:}; % Get denominator polynomial
nnumm = length(numm)-1; % Degree of model numerator
ndenm = length(denm)-1; % Degree of model denominator
if numm(1)^~0, error('Model must be strictly proper'), end;
if any(abs(roots(denm))>1), disp('Warning: Unstable model'), end

nump=[zeros(1,ndenp-nnump-1),nump]; % Pad numerator with leading zeros
numm=[zeros(1,ndenm-nnumm-1),numm]; % Pad numerator with leading zeros

% Define prediction horizon P (steps)(default corresponds to 0.8*Tref):
if Tref == 0,
    P = 5;
else
    P = round(0.8*Tref/Ts);
end
```

```
end

% Define control horizon (default 1):
M = 1;

% Compute model step response values over coincidence horizon:
stepresp = step(model,[0:Ts:max(P)*Ts]);
theta = zeros(length(P),M);
for j=1:length(P),
    theta(j,:) = [stepresp(P(j):-1:max(P(j)-M+1,1))',zeros(1,M-P(j))];
end
S = stepresp(P);

% Compute reference error factor at coincidence points:
% (Exponential approach of reference trajectory to set-point)
if Tref == 0, % Immediate jump back to set-point trajectory
    errfac = zeros(length(P),1);
else
    errfac = exp(-P*Ts/Tref);
end

%%%%% SIMULATION PARAMETERS:

if Tref == 0,
    tend = 100*Ts;
else
    tend = 10*Tref; % Duration of simulation (default 10*Tref)
end
nsteps = floor(tend/Ts); % (Number of steps in simulation).
tvec = (0:nsteps-1)'*Ts; % Column vector of time points (first one 0)

% Define set-point (column) vector (default constant 1):
setpoint = ones(nsteps+max(P),1);

% Define vectors to hold input and output signals, initialized to 0:
uu = zeros(nsteps,1); % Input
yp = zeros(nsteps,1); % Plant Output
ym = zeros(nsteps,1); % Model Output

% Initial conditions:
umpast = zeros(ndenm,1);
uppast = zeros(ndenp,1);
ympast = zeros(ndenm,1); % For model response
yppast = zeros(ndenp,1); % For plant response

%%%%% SIMULATION:
```

```

for k=1:nsteps,

    % Define reference trajectory at coincidence points:
    errornow = setpoint(k)-yp(k);
    reftraj = setpoint(k+P) - errornow*errfac;

    % Free response of model over prediction horizon:
    yfpast = ympast;
    ufpast = umpast;
    for kk=1:max(P), % Prediction horizon
        ymfree(kk) = numm(2:nnumm+1)*ufpast-denm(2:ndenm+1)*yfpast;
        yfpast=[ymfree(kk);yfpast(1:length(yfpast)-1)];
        ufpast=[ufpast(1);ufpast(1:length(ufpast)-1)];
    end

    % Compute input signal uu(k):
    if k>1,
        dutraj = theta\ (reftraj-ymfree(P)');
        uu(k) = dutraj(1) + uu(k-1);
    else
        dutraj = theta\ (reftraj-ymfree(P)');
        uu(k) = dutraj(1) + umpast(1);
    end

    % Simulate plant:
    % Update past plant inputs
    uppast = [uu(k);uppast(1:length(uppast)-1)];
    yp(k+1) = -denp(2:ndenp+1)*yppast+nump(2:nnump+1)*uppast; % Simulation
    % Update past plant outputs
    yppast = [yp(k+1);yppast(1:length(yppast)-1)];

    % Simulate model:
    % Update past model inputs
    umpast = [uu(k);umpast(1:length(umpast)-1)];
    ym(k+1) = -denm(2:ndenm+1)*ympast+numm(2:nnumm+1)*umpast; % Simulation
    % Update past model outputs
    ympast = [ym(k+1);ympast(1:length(ympast)-1)];

end % of simulation

%%%%%% PRESENTATION OF RESULTS:

disp('***** Results from script file BASICMPC :')
disp(['Tref = ',num2str(Tref),', Ts = ',num2str(Ts),...

```

```
' P = ',int2str(P'),' (steps), M = ',int2str(M])  
diffpm = get(plant-model,'num');  
if diffpm{::}==0,  
    disp('Model = Plant')  
else  
    disp('Plant-Model mismatch')  
end  
  
figure  
subplot(211)  
% Plot output, solid line and set-point, dashed line:  
plot(tvec,yp(1:nsteps),'-',tvec,setpoint(1:nsteps),'--');  
grid; title('Plant output (solid) and set-point (dashed)')  
xlabel('Time')  
  
subplot(212)  
% plot input signal as staircase graph:  
stairs(tvec,uu,'-');  
grid; title('Input')  
xlabel('Time')
```



The MPC toolbox

C.1	General remarks	318
C.2	Functions <code>scmpc2</code> and <code>scmpcn12</code>	321
C.3	Functions <code>scmpc3</code> and <code>scmpc4</code>	322

c.1 General remarks

Most of the examples which appear in this book, after Chapter 1, were worked out with the help of the *Model Predictive Control Toolbox* for *MATLAB*. This same software (or other software with similar functionality) is needed for many of the Exercises.

A comprehensive *User's Guide* is available for the *Model Predictive Control Toolbox*. It includes a *Tutorial* section, and it is not proposed to repeat that material here. But it may be useful to relate some of the material in that documentation to the material in this book. Historically, predictive control was first developed for step response models of plants, and the layout of the *User's Guide* reflects this:

- ✿ Chapter 2 of the Tutorial deals with step response models,
- ✿ Chapter 3 deals with state-space models. (And with transfer function models, since the *Toolbox* includes functions for converting transfer functions to state-space models.)

This is in contrast with this book, which places much more emphasis on state-space models than on other forms. Fortunately, Chapter 3 of the *User's Guide* is written in such a way that it can be read without reading Chapter 2 first.

The *Model Predictive Control Toolbox* uses its own format for representing state-space models, which is called the *MOD* format. There is also a special format for representing step-response models. The *Toolbox* includes functions for converting models to and from these special formats. Before starting to use the *Model Predictive Control Toolbox* it is recommended that the following sections of the *User's Guide* (Chapter 3) should be read:

- ✿ State-space models

Table C.1 Approximate correspondence of variables and parameters in this book and in the *MPC Toolbox User's Guide*.

Entity	Name in book	Name in Toolbox
State transition matrix	A	Φ
Input distribution matrix	B	Γ_u
Disturbance distribution matrix	B_d	Γ_d
Output matrix	C_y and/or C_z	C
Inputs	u	u
Outputs	$\begin{bmatrix} y \\ z \end{bmatrix}$	y
Measured disturbance	d_m	d
Unmeasured disturbance	d_u	w and/or z
Prediction horizon	H_p	P
Control horizon	H_u	M
Tracking error weight	$Q(i)$	ywt
Control penalty weight	$R(i)$	uwt

MOD format

Converting state-space to MOD format

Converting MOD format to other model formats

It will be seen from Section 3.1 of the *Guide* that the model assumed in the *MPC Toolbox* is more general than the one which appears in the basic formulation introduced in Chapter 2 of this book, having additional elements such as disturbances and noise which are introduced in later chapters.

Unfortunately the variable names are also different. The approximate correspondences are given in Table C.1. The correspondences are only approximate because, for example, we distinguish between measured outputs y and controlled outputs z , whereas the *Toolbox* stacks both controlled and uncontrolled outputs in the same vector y , with the convention that the controlled outputs appear at the top of the vector.

The core functions which will be needed by readers are:

- **smpccon:** Solves unconstrained predictive control problem. In this case the solution is a constant matrix.
- **smpcsim:** Simulates solution of unconstrained problem, using result from smpccon.
- **scmpc:** Solves constrained predictive control problem with linear plant, and simulates solution.
- **scmpcnl:** Solves constrained predictive control problem with nonlinear plant in the form of a *Simulink* model, and simulates solution. (The internal model used by the MPC controller is linear.)
(Not described in *User's Guide*, but interface similar to scmpc).

There are many other functions, described in the *User's Guide*. The Tutorial section of the *Guide* shows how to use the functions, and the examples which appear in the Tutorial are also available as on-line demos.

The functions `smpccon`, `scmpc` and `scmpcnl` take the input argument M , which is used to specify which input moves the optimizer can make. If M is a scalar (integer) then it is just the control horizon, namely the same as our H_u . But if it is a row vector then it specifies that *blocking* should be used: each element specifies the number of steps over which $\Delta u = 0$, or over which the input u remains unchanged. So $M = [m_1, m_2, m_3]$ means that $\hat{u}(k|k) = \hat{u}(k+1|k) = \dots = \hat{u}(k+m_1-1|k)$, $\hat{u}(k+m_1|k) = \hat{u}(k+m_1+1|k) = \dots = \hat{u}(k+m_1+m_2-1|k)$, etc.

There is no parameter corresponding to our H_w . The parameter P specifies the prediction horizon (our H_p) and implicitly the *Model Predictive Control Toolbox* always assumes $H_w = 1$. However, the parameter which defines weights on tracking errors, `ywt`, can represent weights which vary over the prediction horizon, with each row representing one step. So we can get the effect of $H_w = 3$, for example, by making the first 2 rows of `ywt` zero. Note that you can have fewer than H_p rows in `ywt`; the last row specifies the weights to be used from that value of i onwards.

One significant restriction is that only diagonal tracking error weight matrices $Q(i)$ and control move penalty weight matrices $R(i)$ can be specified. This is done through the arguments `ywt` and `uwt`, respectively. The diagonal elements of $Q(i)$ ($R(i)$ respectively) are represented by the i th row of `ywt` (`uwt` respectively). There is no way of representing off-diagonal elements, so these are always assumed to be zero. This can be a restriction, for example if one wants to represent a $Q(i)$ obtained as the solution of a Lyapunov equation, as in Section 6.2.1. Exercise 3.7 asks you to modify `smpccon` so as to remove this restriction. The other functions could be modified similarly.

The way in which the *Model Predictive Control Toolbox* handles set-points is quite restricted. There is no distinction between set-point and reference signals, of the kind we made in Section 1.2, and have implemented in the *MATLAB* functions `basicmpc`, `trackmpc` and `unstampc`. Tracking errors are always defined as being differences between the set-point and output vectors. Furthermore, although it is possible to define a set-point trajectory over the duration of a simulation, the functions `scmpc` and `scmpcnl` do not make any use of future values of the set-point when computing the control signals. That is, the optimization algorithm is not aware, at each step, of what the future of the set-point trajectory will be, and just assumes that it will be constant at its current value. The function `scmpcr` allows the optimizer to use such information about the future; this can be exploited to perform limited simulations of the effects of defining specific reference trajectories, for example by defining exponential set-point trajectories. But it is not possible to simulate, say, the use of a reference trajectory in the presence of disturbances.

Most of the arguments can be omitted, and then take sensible default values. However, if P is omitted it defaults to 1, which is not usually a sensible prediction horizon. With all possible arguments set to default values one obtains what is sometimes called a ‘perfect controller’ — see Section 7.2.4. But in fact there are a lot of problems with such a controller.

Constraints are represented in the functions `scmpc` and `scmpcnl` by the arguments `ulim` (constraints on u and Δu) and `y1im` (constraints on y). Like the weight parameters, each row of these corresponds to one step in the prediction or control horizon, so that the constraints can vary over the horizons. The bounds on Δu are always assumed to be symmetric ($-U_j \leq \Delta u_j \leq U_j$), and they must be specified to be finite, although

they may be very large.

The *MPC Toolbox* allows arbitrary disturbance and noise models to be defined, and disturbances can be specified to be measured or unmeasured, with feedforward action being applied if measured disturbances are present. If no disturbance or noise model is specified then by default the DMC disturbance model is assumed, namely a step disturbance on each output, with no measurement noise. Observer gains can also be specified, the default observer gain being taken to be the optimal observer for the DMC disturbance model, namely $[0, I]^T$ — see Section 2.6.3. If a more elaborate model is needed, there is a simplified way of specifying the disturbance/noise model of Lee and Yu — see Section 8.2. For other disturbance and noise models the user must specify the dynamics as state-space models, but there are facilities which help with building up the augmented model which includes the plant, disturbance and noise dynamics.

The *Model Predictive Control Toolbox* also contains functions for frequency-response analysis of predictive controllers, under the assumption that constraints are not active. For example, singular value plots of various closed-loop transfer functions, such as the sensitivity and complementary sensitivity — see Section 7.3 — can be obtained by using the functions `clmod`, `mod2frsp` and `svdfrsp`.

The method used to solve the QP problem which arises in functions `scmpc` and `scmpcnl` is described in [Ric85].

C.2 Functions `scmpc2` and `scmpcnl2`

Modified versions of the functions `scmpc` and `scmpcnl` are available on this book's web site. These modified functions have the names `scmpc2` and `scmpcnl2`, respectively, and the modifications extend the functionality of the original functions in the following ways.

1. Non-default observer gains can be used. Although this is also possible with the standard functions `scmpc` and `scmpcnl`, those provide no facility for initializing the observer state. When simulating a plant with non-zero initial state (a common requirement particularly when using `scmpcnl`) and using a non-default observer gain, it is important to be able to specify the initial observer state. This possibility has been added.
2. It is possible to continue a previously interrupted simulation. This is enabled by providing additional input arguments for specifying the initial states of the plant and observer, and the initial input values. Also the specification of the initial time has been enabled — it does not always have to be 0. This facility gives important additional functionality to the *Model Predictive Control Toolbox*. It allows a simulation to be interrupted, some changes to be made, and the simulation to be resumed. This allows, for example:
 - Time-dependent weights and constraints. The standard functions allow dependence on i , but not k . The modified functions allow dependence on both.
 - State-dependent weights and constraints.

Table C.2 Additional or modified input arguments available in functions `scmpc2` and `scmpcn12`.

Argument	Purpose
<code>x0</code>	Initial observer state
<code>tvec</code>	Specification of start and end times of simulation (replaces <code>tend</code>)
<code>swt, syt</code>	Specified penalty coefficient on constraint violations
<code>normtype</code>	Specifies 1-norm 2-norm, ∞ -norm, or mixed-norm penalty for constraint violations

- Modification of the linear internal model during a simulation run. With a nonlinear plant this allows the linearized model used for prediction to be updated from time to time. It could also be used to simulate adaptive MPC.

It is anticipated that this feature will be mostly used to interrupt and resume a simulation occasionally, but it is also possible to interrupt it at every step — that is, to perform one step of the simulation, make suitable updates, then perform one more step, etc.

- Soft constraints on outputs have been implemented. It is possible to specify a 1-norm, 2-norm, or ∞ -norm penalty, or a mixture of 1-norm and 2-norm penalties. It is also possible to specify different values of penalty coefficient (ρ in the notation of Section 3.4) for different outputs. A new function `qpsoft` has been written to help implement this new functionality, and is used by the modified functions.

Table C.2 lists the additional input arguments which have been made available in functions `scmpc2` and `scmpcn12`.

C.3 Functions `scmpc3` and `scmpc4`

Function `scmpc2` has been further modified to allow simulation with exponential reference trajectories, as described in Section 7.5. Function `scmpc3` implements reference trajectories without anticipation of future set-point changes (see Figure 7.17), while `scmpc4` assumes that future set-point changes are known, and anticipates them (see Figure 7.18).

Both functions have an additional argument, `Tref`, which is a vector of time constants, one for the reference trajectory of each controlled output. Zero values are allowed, which indicates that the reference trajectory coincides with the set-point trajectory.

Author index

- Abu el Ata-Doss, S., 28, 160
Al-Naib, U., 5
Anderson, B.D.O., 80, 226
Arber, C., 28, 160
Åström, K.J., 40, 126, 131, 218
Athans, M., 227–9
- Backx, T., 40
Badgwell, T.A., 79, 157, 285, 290
Balakrishnan, V., 95, 221, 233, 235, 237, 238, 243
Barratt, C., 84, 184
Bemporad, A., 87, 150, 214, 245, 283, 284, 290
Bequette, B.W., 285
Berenguel, M., 28
Biegler, L.T., 98, 99
Billings, S.A., 286, 287
Bitmead, R.R., 80, 132, 172, 177, 180–3, 226, 227, 229, 291
Bock, H.G., 285
Bordons, C., 28, 119, 132, 143, 171
Boskovic, J.D., 281
Boyd, S., 84, 95, 184, 233, 235
Bryson, A.E., 29, 30, 80, 181, 226
- Camacho, E.F., 28, 119, 132, 143, 171
Campo, P.J., 155, 220, 242
Casavola, A., 214
Chang, T.S., 152
- Chen, B.M., 229
Chou, C.T., 142
Clarke, D.W., 23, 28, 108, 140, 192, 193, 206
Coales, J.F., 25
Cutler, C.R., 25, 57, 108, 306
- Daley, S., 286
de Moor, B., 40, 121
de Oliveira, N.M.C., 98, 99
de Vries, R.A.J., 184
Demircioglu, H., 163–5
Diehl, M., 285
DiGirolamo, R.D., 291
Dorea, C.E.T., 241, 242
Doyle, F.J., 286
Doyle, J.C., 117, 119, 151, 183, 184, 200, 220, 228, 229, 256, 264
Dua, V., 87
- Edgar, T.F., 290
El Ghaoui, L., 95, 233, 235
Emami-Naeini, A., 6, 39, 52
- Feron, E., 95, 233, 235
Fletcher, R.R., 84, 88, 89, 95, 99
Franklin, G.F., 6, 39, 52
- Garcia, C.E., 40, 50, 106, 108, 119, 145, 202, 220–3, 227, 248, 255, 306
Gattu, G., 285

- Gawthrop, P.J., 163–5
 Genceli, H., 242
 Gerules, M., 287
 Gevers, M., 80, 132, 172, 177, 180–3,
 226, 227, 229
 Gilbert, E.G., 169, 172, 214, 240, 241
 Gill, P.E., 84
 Glover, K., 117, 119, 151, 183, 184,
 200, 220, 256, 264, 291
 Golub, G.H., 78, 91, 117
 Gopinathan, M., 281

 Hansen, L.K., 286
 Hennet, J.C., 241, 242
 Ho, Y-C., 29, 30, 80, 181, 226
 Huzmezan, M., 213, 284, 291

 Johansen, T.A., 283, 284
 Johnson, D., 287

 Kadirkamanathan, V., 286, 287
 Kailath, T., 52, 62, 116, 123
 Kalman, R.E., 227
 Keerthi, S.S., 169, 172
 Kerrigan, E.C., 99, 246, 284
 Kleinman, D., 25
 Kolmanovsky, I., 214, 241
 Kothare, M.V., 221, 233, 235, 237, 238,
 243
 Kouvaritakis, B., 103, 149, 172, 174,
 184, 245
 Kung, S.Y., 118
 Kuntze, H.B., 28, 160
 Kwakernaak, H., 30, 181, 226
 Kwon, W.H., 25

 Lasdon, L.S., 290
 Lee, J.H., 119, 201, 221–5, 227, 242,
 286
 Lee, P.L., 269, 270
 Leineweber, D., 285
 Liebman, M.J., 290
 Linkens, D.A., 291
 Liu, G.P., 286, 287
 Ljung, L., 40, 120
 Lu, P., 291

 Maciejowski, J.M., 24, 78, 99, 117, 119,
 142, 151, 178, 183, 184, 200,
 213, 220, 228, 229, 232, 246,
 256, 264, 278, 284, 291
 Magni, L., 183
 Mahfouf, M., 291
 Martin Sanchez, J.M., 25, 40
 Mayne, D.Q., 171, 172, 183, 239, 240,
 242–5, 288, 289
 McEwen, A.J., 118
 Mehra, R.K., 25, 281
 Michalska, H., 171, 239, 244, 288
 Mignone, D., 284, 290
 Miller, R.H., 214
 Mohtadi, C., 23, 40, 108, 192, 193
 Moore, J.B., 80, 226
 Morari, M., 87, 119, 150, 155, 172, 202,
 220–3, 227, 233, 235, 237, 238,
 242, 243, 248, 250, 264, 282–4,
 290
 Morshedi, A.M., 306
 Mosca, E., 40, 131, 132, 136, 143, 171,
 180, 181, 214, 287
 Murray, W., 84
 Muske, K.R., 149, 172, 174, 176, 281

 Nemirovsky, A.S., 88, 94
 Nesterov, J.E., 88, 94
 Newell, R.B., 269, 270
 Nikolau, M., 242
 Norton, J.P., 40, 120
 Noton, A.R.M., 25
 Nørgaard, M., 286

 Ogunaike, B.A., 286
 Ordys, A.W., 140

 Papageorgiou, G., 291
 Papon, J., 25, 159, 160
 Pearson, A.E., 25
 Pearson, R.K., 286
 Piché, S., 287
 Pierson, B.L., 291
 Pistikopoulos, E.N., 87
 Pollard, J.F., 40
 Poulsen, N.K., 286

- Powell, J.D., 6, 39, 52
 Prett, D.M., 25, 50, 57, 106, 108, 145,
 220, 248, 250, 255, 264
 Propoi, A.I., 25
 Qin, S.J., 79, 157, 285, 290
 Rago, C., 281
 Ramaker, B.L., 25, 57, 108, 306
 Rao, C.V., 2, 56, 95, 97, 172, 183, 290
 Rault, A., 25, 159, 160
 Ravn, O., 286
 Rawlings, J.B., 2, 56, 95, 97, 99, 104,
 149, 172, 174, 176, 183, 227,
 233, 240, 281, 288–90
 Redhead, S.N., 211
 Rice, M.J., 149
 Richalet, J., 9, 16, 23, 25, 28, 40, 150,
 159, 160, 162, 211
 Ricker, N.L., 50, 321
 Rivera, D.E., 40
 Rodellar, J., 40
 Roos, C., 2, 88, 94
 Rosenbrock, H.H., 228
 Rossiter, J.A., 103, 149, 172, 174, 184,
 245
 Rouhani, R., 25
 Rowe, C.A., 232, 279
 Runkle, J.C., 281
 Ryan, E.P., 25
 Saberi, A., 229
 Safonov, M.G., 227
 Sandoz, D.J., 291
 Sannuti, P., 229
 Sayyar-Rodsari, B., 287
 Schröder, J., 285
 Schuurmans, J., 245
 Scokaert, P.O.M., 99, 104, 149, 172,
 179, 183, 227, 233, 240, 242–
 5, 288–90
 Seborg, D.E., 152
 Sepulchre, R., 183
 Shah, S.L., 40
 Shook, D.S., 40
 Siller-Alcalá, I., 163–5
 Singh, S.N., 291
 Sivan, R., 30, 181, 226
 Slupphaug, O., 283, 284
 Smith, O.J.M., 26
 Soeterboek, R., 108, 126, 132, 188, 211
 Stein, G., 228, 229
 Steinberg, M., 291
 Sussmann, H.J., 30
 Sutton, G.J., 291
 Tan, K.T., 214, 240, 241
 Terlaky, T., 2, 88, 94
 Testud, J.L., 25, 159, 160
 Tuffs, P.S., 23, 108
 Tyler, M.L., 282
 Vada, J., 283, 284
 van den Boom, T.J.J., 184
 van Loan, C.F., 78, 91, 117
 van Overschee, P., 40, 121
 Vial, J.-Ph., 2, 88, 94
 Washabaugh, P.D., 214
 Wertz, V., 80, 132, 172, 177, 180–3,
 226, 227, 229
 Willems, J.C., 30
 Wittenmark, B., 40, 126, 131
 Wonham, M., 202
 Wright, M.H., 84
 Wright, S.J., 2, 56, 88, 89, 91, 93–5, 97
 Yu, Z.H., 201, 221, 222, 224, 225, 242
 Zafiriou, E., 202, 285
 Zakian, V., 5
 Zeiger, H.P., 118
 Zheng, A., 220, 242
 Zhou, K., 117, 119, 151, 183, 184, 200,
 220, 256, 264
 Zhu, Y.C., 40

Subject index

- 1-norm
 - cost, 155
 - penalty, 98, 100, 107
- absolute value cost, 151, 154
- active constraints, 84, 87, 89, 93, 94, 191, 199, 204, 214, 309
- Active Set method, 88, 89, 308
- actuator failure, 278
- actuator saturation, 6
- adaptive control, 28, 40, 125, 167, 227, 278, 287, 310
- additive uncertainty, 218, 265
- alternative state variables, 119
- analytic centre, 95
- applications of MPC, 290
- ARX model, 123, 142, 309, 310
- Åström's robustness result, 218
- augmented state vector, 49, 53, 59, 60, 71, 135, 140, 180, 207
- auto-compensation, 160, 161
- backslash operator, 11, 13, 79
- balanced realization, 255
- balanced truncation, 119, 256
- banded matrix, 91–3, 97
- bandwidth, 225, 231, 232
- barrier function, 95, 96
- basis function, 159
- Bezout identity, 126
- black-box model, 6, 40, 285, 286
- block-Hankel matrix, 114, 116, 117, 119
- blocking, 34, 105, 257, 275, 307, 309, 310, 320
- Bode plot, 200
- case study
 - evaporator, 269–74
 - Shell oil fractionator, 248–69
- Cauchy–Schwarz inequality, 237
- causality constraint, 242
- central path, 95–7
- certainty equivalence principle, 80, 132
- clipping, 163, 166
- closed-loop transfer function, 184
- coincidence horizon, 160
- coincidence point, 9, 43, 156, 158, 159, 161, 255, 286
- complementarity condition, 89, 97
- complementary sensitivity, 190, 200, 201, 206, 219, 225, 232, 246, 264, 321
- compressor control, 3
- computational complexity, 94, 101, 255, 277, 282
- computational delay, 50, 51, 53, 72
- Connoisseur*, 309
- constrained optimization, 28, 31, 83
- constraints, 1, 2, 5, 12, 31, 41, 43, 66, 81, 84, 98, 162, 176, 178, 191,

- 252
- equality, 85, 88, 90, 91, 277
- management, 97, 281, 291, 307
- priorities, 283, 307, 310, 311
- robust satisfaction, 233
- violation, 104, 284
- window, 282
- continuity of MPC controller, 87
- continuous time, 14, 33
- continuous-time predictive control, 163–5
- control hierarchy, 26
- control horizon, 42, 159, 162, 180, 224, 255, 257
- control invariant set, 245
- control weight, 258, 267
- controllability matrix, 116, 118
- controlled output, 36, 41, 109, 250, 270, 319
- controller form, 116
- convex hull, 221
- convex optimization, 2, 83, 84, 90, 184, 233, 235, 245, 286
- convolution, 109, 111
- coprime factor, 151, 166
 - uncertainty, 219
- coprime polynomial, 126
- correlated disturbances, 227
- cost function, 41, 74, 105
- CRHPC, 171
- current practice, 26
- danger of feedback, 189
- data reconciliation, 290
- deadbeat, 23, 59, 63, 118, 140, 149, 173
 - control, 192, 197
 - observer, 192, 195, 204, 214
- decomposition of unstable plant, 150
- decreasing function, 169, 170
- deterministic disturbance, 127, 131
- difference equation, 121, 122, 137
- differential-algebraic equation, 285
- Diophantine equation, 126, 129, 132
- discrete time, 39
- disturbance
 - large, 90, 97, 102
 - on state, 229
 - sinusoidal, 105
 - unmeasured, 261
- disturbance model, 201, 203, 206, 221, 227, 254, 321
- disturbance rejection, 252
- DMC, 25, 26, 57, 60, 108, 110, 128, 160, 195, 202, 221, 223, 278, 281, 306, 310, 321
- dual-mode predictive control, 172, 288
- dynamic matrix, 110
- ellipsoid, 237, 245
- EPSAC, 26
- equal concern error, 307
- equilibrium, 38, 169, 170, 270
- Euclidean norm, 42
- evaporator, 269–74
- exact penalty function, 99, 102, 103, 262
- examples
 - buffer tank, 5
 - Citation aircraft, 64–9, 72, 99, 102, 106, 107
 - helicopter, 24, 140, 165
 - internal combustion engine, 4
 - LNG plant, 279
 - paper machine headbox, 45–7, 194, 201, 202, 215
 - semi-batch reactor, 3
 - surge level control, 5
 - swimming pool, 104, 165, 166, 206–10, 215
 - underwater vehicles, 4
- fake
 - algebraic Riccati equation, 180–3
 - Hamilton–Jacobi–Bellman equation, 183
- fault-tolerant control, 278, 281
- feasibility, 150, 175, 233, 239
- feasible problem, 169, 174
- feasible solution, 90, 93–5
- feasible SQP, 285
- feed-through, 37
- feedback, 20, 23

- feedback policy, 243
- feedforward, 32, 145–8, 165, 206, 249, 255, 259, 308, 321
- finite impulse response, 115, 118, 124, 149, 185, 220
- finite precision, 56, 149
- finitely determined, 241
- first-principles model, 7, 40, 41, 119, 278, 285
- fixed point, 170
- flight control, 6, 28, 64
- free response, 10, 75, 131
- frequency response, 190, 199, 200, 222, 227, 321
- Frobenius norm, 117
- funnel, 156, 162, 308
- gain scheduling, 307
- Gaussian elimination, 91
- global optimum, 84, 90, 171, 174, 284, 289
- GPC, 23, 26, 31, 108, 133, 135, 137, 140, 144, 150, 171, 183, 202, 206, 214, 225
- H-infinity
 - control, 150, 232
 - norm, 218
- Hankel matrix, 114, 116, 117, 119
- Hankel singular value, 255, 256
- Hessian, 76, 84, 285
- HIECON, 311
- high-bandwidth applications, 2, 88
- historical data, 288
- hot starting, 94, 289
- hybrid system, 283
- ideal resting value, 43, 277
- IEC 1131-3 standard, 311
- IEEE Standard arithmetic, 56
- ill-conditioned problem, 78, 96
- impulse response, *see* pulse response
- independent model, 16, 22, 62, 140
- induced norm, 117, 200
- infeasibility, 83, 97, 100, 239, 262, 281
- infinite horizon, 149, 172, 175, 176, 178, 181, 227, 232, 233
- ∞ -norm
 - cost, 155
 - penalty, 98
- information filter, 182
- innovations representation, 136
- input constraints, 98
- input disturbance, 72, 248
- integral action, 20, 115, 134, 202, 231
- integrated white noise, 230
- integrator wind-up, 6, 163, 291
- interior point method, 88, 94
- internal model, 6, 8, 31, 174, 183, 258, 271, 273, 274, 284
 - control, 202
 - principle, 202, 203
- internal stability, 183, 194
- invariant set, 237, 240
- inverse response, 193
- Jordan decomposition, 178
- Kalman filter, 58, 80, 136, 182, 206, 207, 222–7, 230, 289
 - extended, 289
- Karush–Kuhn–Tucker conditions, 89, 90, 97
- Lagrange multiplier, 85, 89, 90, 94, 99, 103, 285
- Lagrangian, 285
- large disturbance, 103
- least-squares, 9, 11, 13, 77, 103, 118, 277, 290
 - recursive, 310
- Lee and Yu tuning, 221, 321
- lexicographic minimum, 283
- limits on feedback performance, 191
- linear controller, 80, 86, 154
- linear inequality, 240
- linear matrix inequality, 95, 233, 235, 238, 245
- linear programming, 84, 94, 152, 154, 156, 157, 165, 241, 276, 283, 307
- linear quadratic control, 150, 180, 181, 227, 291

- linearized model, 38, 39, 41, 53
 local optimum, 84
 loop transfer recovery, 225–32
 dual procedure, 229
 LQG design, 225
 LTR, *see* loop transfer recovery
 LU factorization, 91
 Lyapunov
 equation, 177, 179, 180, 320
 function, 169, 170, 173, 175, 239
 stability theorem, 170, 289
- manipulated variable, 249, 270
 Markov parameter, 111
MATLAB file
 basicmpc.m, 14–17
 disturb2.m, 107
 evaporator.mdl, 271, 275
 makecita, 72
 mismatch2.m, 107
 mkdelta.m, 265
 noisympc.m, 21
 oilfr20.mat, 256
 scmpc.m, 208
 scmpc2.m, 275, 321, 322
 scmpc3.m, 213, 322
 scmpc4.m, 213, 322
 scmpcnl.m, 271
 scmpcnl2.m, 271, 321, 322
 simevap.m, 274, 275
 smpcest.m, 208, 223
 swimpool.m, 210, 216
 trackmpc.m, 19
- matrix inversion lemma, 137
 matrix square-root, 77
 maximal output admissible set, 240, 241
 mean-level control, 192, 195, 255
 measured disturbances, 145–8, 249
 measured output, 36, 249, 319
 measured outputs, 109, 134
 measurement noise, 189, 205, 221, 232,
 254, 264
 min-max problem, 155–6, 234, 242, 245
 minimal state-space model, 116, 255
 minimum variance, 131, 132, 134
 mixed logical dynamic system, 283, 284
 mixed penalty function, 99
 mixed weighted least-squares, 103
 mixed-integer optimization, 282, 283
 MLD, *see* mixed logical dynamic system
 MOD format, 318
Model Predictive Control Toolbox, 140,
 144, 195, 207, 223, 318
 model reduction, 119
 monic polynomial, 127
 monotonicity property, 182, 183
 move suppression factor, 43, 191, 267
 moving-horizon estimation, 289
 multiobjective optimization, 277, 284
 multiparametric programming, 87
 multiple shooting, 285
 multiplicative uncertainty, 219, 246
 multivariable control, 6, 31
 multivariable system, 162, 188, 200, 225
 MUSMAR, 287
 myths, 109
- neural net model, 285–7
 nominal stability, 175
 non-diagonal weights, 106
 non-Gaussian distribution, 5
 non-minimum-phase, 193, 229
 non-quadratic cost, 151, 288
 nonlinear
 controller, 5, 47, 84, 153
 model, 7, 39, 183, 284, 307
 plant, 37, 269, 274
 predictive control, 284, 307
 nonlinear predictive control, 284
 norm-bounded uncertainty, 218, 237
 null space, 254
 Nyquist
 frequency, 201, 227, 232, 264
 stability theorem, 219
- observability matrix, 116, 118
 observer, 23, 31, 57–62, 80, 82, 127,
 134, 136, 139, 140, 175, 192,
 195, 226, 233, 254, 289, 321
 dynamics, 201, 203–10
 polynomial, 135, 206, 225

- off-line computation of controller, 88
- offset-free tracking, 17, 23, 33, 140, 160, 161, 202, 221, 230, 263, 277
- operator interface, 56
- output admissible set, 240, 245
- output constraints, 98
- output disturbance, 56, 59, 60, 114, 127, 128, 137, 160, 185, 189, 202, 221, 223
- output measurement, 8
- penalty function, 307
- 'perfect' control, 193, 199, 214, 320
- performance specification, 251
- persistent disturbance, 202, 244
- PFC, *see* predictive functional control
- PI control, 291
- piecewise-linear controller, 87
- pioneers of predictive control, 25
- plant operator, 153, 222, 291, 306
- plant testing, 40
- plant uncertainty, 218
- plant–model mismatch, 17, 41, 69, 99, 100, 190, 239, 267
- pole placement, 149, 207
- polynomial set-point, 160
- polytope, 221, 234, 236, 240, 241
 - uncertainty, 220, 233
- positive definite, 42, 170
- positively invariant set, 240, 288
- prediction, 41, 53, 60, 80, 112, 113, 123, 124, 126, 131, 134, 147, 226
- prediction horizon, 8, 41, 173, 180, 218, 224, 255
- predictive functional control, 26, 150, 159–63, 286, 311
- predictor corrector, 127
- primal-dual method, 97
- principle of optimality, 173
- priority of objective, 277, 283
- Process Perfecter*, 287, 311
- profit maximization, 6
- programmable logic controller, 311
- propositional logic, 282, 283
- pseudo-inverse, 117
- pulse response, 12, 108, 123, 126
 - uncertainty, 220
- purpose of feedback, 189
- QDMC, 26, 306
- QP, *see* quadratic programming
- QR algorithm, 78, 80
- quadratic
 - cost, 41
 - form, 42
 - inequality, 235
 - model, 287
 - penalty function, 98
- quadratic programming, 12, 83, 84, 88, 152, 164, 172, 177, 284, 285, 290, 308, 321
 - as LMI problem, 235
 - special structure, 88, 95
- ramp disturbance, 143
- random walk, 133
- re-linearized model, 274, 284
- realigned model, 22, 62, 63, 125, 130, 137
- receding horizon, 7, 9, 31
- reference governor, 214
- reference trajectory, 7, 31, 33, 41, 75, 159, 162, 211–14, 216, 308, 320, 322
- regulator problem, 174
- return ratio, 228, 230–2
- Riccati equation, 30, 180–2, 223, 224, 226, 309
- RMPCT*, 157, 162, 308
- robust
 - constraint satisfaction, 233
 - control, 200, 217, 263, 309
 - disturbance compensation, 209
 - feasibility, 239, 246
 - performance, 220
 - stability, 200, 219, 233, 239, 264
- rotation factor, 306
- sampling interval, 51, 254, 271
- sampling time, 255

- scaling of variables, 257, 309
 semi-positive definite, 42
 sensitivity, 190, 200, 201, 206, 216, 219, 225, 232, 264, 321
 sensor failure, 278
 separation principle, 80, 132
 sequential quadratic programming, 285
 servomechanism, 28, 160, 305, 311
 set-point, 7
 filtering, 213–14
 settling time, 218
 Shell oil fractionator, 248–69
 simplex, 84
Simulink, 269, 270, 319
 singular value, 117–19, 200, 219, 263, 268, 321
 decomposition, 117, 309
 structured, 220, 265, 268
 SIORHC, 171
 slack variable, 98, 99, 101, 283
 small-gain theorem, 219
 Smith predictor, 26
 soft constraints, 97–104, 106, 157, 239, 261, 267, 282, 283, 307, 309, 310, 322
 spectral density, 128, 230
 SQP, *see* sequential quadratic programming
 stability, 32, 167–87, 288, 306
 asymptotic, 170
 margin, 190, 200, 227, 228
 of equilibrium, 170
 stabilized predictions, 56, 148, 245
 state estimate, 41, 61, 80, 82, 134, 136, 147, 175, 204, 225, 226, 285
 state feedback, 148, 181, 226, 233, 236, 237, 245
 state-space model, 31, 36, 111, 254, 318
 steady state, 38, 287
 gain, 109, 253, 277
 optimization, 26, 308
 step response, 12, 31, 108, 189, 218, 251, 254, 306, 318
 stochastic disturbance, 127, 128, 131, 136, 221, 223
 strictly proper model, 8, 12, 37, 122, 253
 structured uncertainty, 220, 233, 250, 265
 subspace methods, 121, 311
 surge line, 3
 SVD, *see* singular value decomposition
 system identification, 40, 41, 120, 287, 288
 Taylor series, 287
 terminal constraint, 169, 172, 174, 177, 178, 182, 183, 244, 288
 set, 171, 244, 288
 terminal cost, 172, 177, 182, 183
 3dMPC, 311
 time delay, 122, 254, 257
 time-invariant controller, 48, 49
 tracking error, 75, 79, 80, 148, 213
 transfer function, 31, 108, 121, 254, 318
 matrix, 122
 tuning, 32, 188–216, 232
 tuning parameters, 43, 54, 61, 135, 188, 201, 222, 271
 two degree of freedom system, 189
 unconstrained predictive control, 74–81, 189
 unstable closed loop, 168
 unstable mode, 175, 178, 179
 unstable plant, 22, 34, 60, 63, 133, 140, 148, 150, 167, 175, 176, 178, 179, 184, 189, 191, 223, 227, 231
 unstructured uncertainty, 219
 variable horizon, 239
 Volterra model, 285, 286
 web site, xii, 14, 213, 256, 265, 271, 274, 275, 306, 321
 weighted norm, 42
 white noise, 123, 128
 Youla parameter, 183, 184
 zone, 5, 45, 156, 252, 257, 261, 275, 308
- 