UNIVERSITY OF STELLENBOSCH

A Unified Strategy for Windup Prevention in Control Systems with Multiple Saturating Actuators

Michael John Browne 9462589

Thesis submitted in partial fulfilment of the requirements for the degree of Masters in Engineering in the Department Electrical and Electronic Engineering at the University of Stellenbosch

STUDY LEADER: Professor J.J. du Plessis

December 2000

DECLARATION

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Michael John Browne

ABSTRACT

A unified method is proposed to treat saturation in both Multi-Input-Multi-Output MIMO and Single-Input-Single-Output controllers. This method offers superior performance over existing MIMO anti-saturation schemes.

The anti-saturation problem is posed as a linear programming problem. A practical and efficient implementation of the algorithm is developed by transforming the problem into its dual linear programming form. The problem, in dual form, is then solved using the dual simplex method rather than the primal simplex method. The nature of the problem when expressed in dual form and the properties of the dual simplex method are harmonised to guarantee an initial basic feasible solution and an optimal bounded final solution in a finite, predictable and minimal number of iterations.

The resultant controller never saturates, hence cannot windup. Furthermore the resultant controller always applies the optimal control effort to the plant to minimise the error signal input as follows:

- The controller is governed such that while the future free response combined with the present forced response of the controller results in no saturation limits being exceeded, now or at some time in the future, the normal linear response of the controller prevails.
- When the future free response combined with the present forced response of the controller will result in a saturation limit being reached, now or at some time in the future, the present time input signal into the controller is optimally governed to prevent the saturation limit from being exceeded at any future time.

ABSTRAK

'n Metode word voorgestel waarmee versadiging in enkel-inset enkel-uitset en meer-inset meer-uitset (MIMU) stelsels beheer kan word. Die metode presteer beter as ander huidige teen-versadiging-maatreels vir (MIMU) beheerders.

Die teen-versadigings-probleem word as 'n lineere programmeringsprobleem herformuleer. 'n Praktiese en effektiewe implementering van die algoritme word verkry deur die probleem na die duale vorm te transformeer. Die probleem, in duale vorm, word opgelos met die duale simplex metode, in plaas van die direkte metode. Die eienskappe van hierdie formulering is 'n gewaarborgde, aanvanklike, bereikbare oplossing en 'n optimale, begrensde, finale oplossing in 'n eindige, voorspelbare en minimum aantal stappe.

Die uiteindelike beheerder versadig nooit nie, en wen gevolglik nie op nie. Die beheerder wend altyd die optimale aanleg-inset aan om die foutsein te minimeer soos volg:

- Wanneer die nul-inset gedrag saam met die huidige inset-gedrag geen beperkings nou of in die toekoms sal oorskry nie, word geen beperkende aksie geneem nie, en tree die beheerder dus lineer op.
- Sodra die toekomstige nul-inset gedrag saam met die huidige inset-gedrag, nou of later versadiging sou veroorsaak, word die huidige inset tot die beheerder optimaal begrens om latere versadiging te voorkom.

ACKNOWLEDGEMENTS

I would like to thank:

- Professor J.J. du Plessis for his persistence and patience with me over the years as I progressed with this work.
- My parents for their encouragement and support throughout all my years of studies.
- Allen, my girl friend, for her love and understanding during the time I dedicated to this research.
- Professor Johan Gouws, a long time friend, for proof reading this thesis.
- My friends and colleagues Dr Claudio Antonini, Stephan van der Walt and Kor Snyman for the interesting debates on control systems and anti-saturation related issues during our careers at Kentron.
- Kentron for the use of their computer equipment and other facilities.

ABBREVIATIONS

$\mathbf{A}$	Continous state space state transition matrix
$\mathbf{A}$	Linear program constant coefficient matrix
$\mathbf{b}$	Linear program constraint constants
$\mathbf{B}$	Continous state space state input matrix
BIBO	Bounded input Bounded Output
$\mathbf{c}$	Linear program objective function coefficients
$\mathbf{C}$	State space output matrix
CAW	Conventional Anti Windup
D	State space direct coupling matrix
DC	Steady state
e	Error signal
$\mathbf{e}$	Vector error governor
$\mathbf{E}$	Matrix error governor
EG	Error Governor
$\mathbf{\Gamma}$	Discrete state space state input matrix
GCT	Generalised Conditioning Technique
$G(s)$	Dynamic plant
h	Optimisation horizon
IMC	Internal Model Control
k	Gain
K	Gain
$K(s)$	Dynamic compensator
$\lambda(t)$	Scalar error governor (Kapasouris)
LQG	Linear Quadratic Gaussian
LQR	Linear Quadratic Regulator
m	Number of controller inputs
MIMO	Multi Input Multi Output
n	Number of controller states
PI	Proportional + Integral controller
r	Reference signal
r^r	Realisable reference signal
$\mathbf{s}$	Linear program slack/surplus variables vector
SISO	Single Input Single Output
τ	Time constant
t	Time
u	Control signal
$\hat{u}$	Constrained control signal
u_{max}	Control maximum limit
u_{min}	Control minimum limit

x	State
$\mathbf{x}$	State vector
$\mathbf{x}$	Linear program optimisation variables vector
$\dot{x}$	State derivative
y	Plant output
p	Number of controller outputs
$\mathbf{\Phi}$	Discrete state space state transition matrix
$\mathbf{q}$	Vector error governor
$\mathbf{Q}$	Matrix error governor
z	Linear program objective function

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation for Research and Problem Statement

In practical control engineering applications, the actuators are subject to hard physical constraints such as limitations on the maximum force, torque output, stroke, power consumption, and linear/angular rates. For example, the elevator of an aircraft can only deflect over a certain limited angular range and a jet engine has a thrust limit than cannot be exceeded. Nevertheless, a fighter pilot might require full elevator deflection for maximum climb rate and full thrust from the engine for maximum speed in a combat situation. To operate a control system only in the actuator's linear region would require, in this example, an aircraft designed with an oversized elevator and engine that are purposefully under utilised in order to avoid reaching the saturation limits. This would amount to a very conservative control system design and an ineffective utilisation of the available aircraft resources. In order to maximise the system's performance it is preferable to optimally utilise all the control available and to take precautions in the controller to compensate for the effects of saturation in the actuator.

The typical closed loop feedback control system block diagram shown in Figure 1.1 can be analysed as follows:

- The controller is driven by the difference or error signal, e, between the reference command, r, and the system's output, y, as measured by the sensor. The outputs in an aircraft, for example, could be the altitude, air speed, heading and bank angle.

- The controller produces an output called the control signal, u, which commands the actuator. In an aircraft, for example, the actuator could be the elevator servo or engine throttle servo.

- The servo has maximum physical constraints that cannot be exceeded. This is modelled by the limiter preceding the servo that clips the controller output signal, u, to give the bounded servo input signal, $\hat{u}$.

- The linearised servo, system and sensor dynamics form the design plant for the linear controller design.

The control input saturations are often the dominant nonlinearity in the system that is to be controlled. It is well known that unless precautions are taken in the control law, undesirable

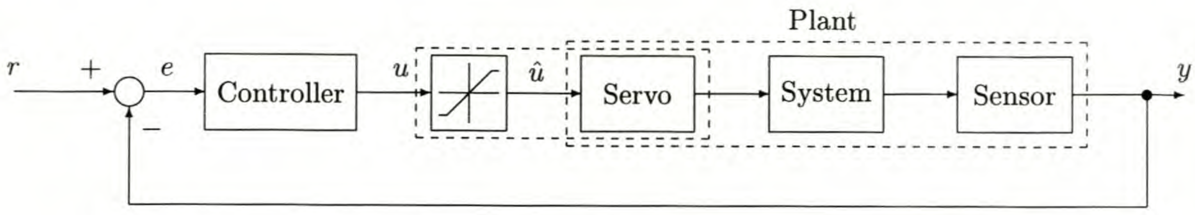

Figure 1.1: Typical control system with plant input constraint.

behaviour such as large overshoots, long settling times and even instability can result if the actual plant's input, $\hat{u}$, is different from the controller's output, u, [Yoshida *et al.*]. Thus, the control system design process must account for input constraints associated with the system's actuators. The presence of control saturations results in a nonlinear control design problem. Nevertheless, the relatively simple form of the input saturations, together with the extensive design theory available for linear systems, is such that the dominant means of dealing with the problem is to design a linear controller ignoring the input nonlinearities. Thereafter some form of post-compensation is added to deal with the effects of the nonlinearity [Pachter and Miller].

Saturation in a dynamic controller results in the phenomenon known as *windup*. Windup is said to occur when the states of the controller are driven by the error while the actuator is in saturation [Campo and Morari] and anti-windup compensation is taken to mean any attempt to reduce the undesirable effects of windup. The goal of anti-windup compensation is to limit the increase of the controller's state while the plant's input is in saturation, and thus to achieve a graceful performance degradation of the control system as compared to the fully linear system.

Several attempts to address the plant input saturation issue can be found in the literature. See Kothare *et al.* and numerous references therein. These methods can be divided into those specifically for single-input-single-output (SISO) controllers and those more generally intended for multi-input-multi-output (MIMO) controllers, where the SISO controller could be considered as a special case [1]. The methods intended for SISO applications are generally easier to apply and require less computational effort than the MIMO methods. Inside each of these classes of anti-saturation methods there are two further broad classifications:

- The *explicit* class of methods ignores the saturation limit during the design phase, and then places some form of governor around the linear controller to cope with the saturation limit during implementation.

- The *implicit* class of methods treat the saturation limit as part of the design plant and impact directly on the control law formulation. The implicit methods produce directly nonlinear controllers and are usually classed as nonlinear design methods. Constrained Model Predictive Control is such a method. These methods prescribe a particular control design philosophy on the designer.

Traditionally, the input saturation problem is treated by *ad hoc* desensitization schemes that are based on designer intuition with no *a priori* guarantees of system stability and performance [Davidson and Jiao]. These customised anti-saturation solutions are specific to the controller structure and may vary from controller to controller. Furthermore these schemes are best

[1]But the method would probably be an overkill for SISO applications.

applied to SISO controllers and they are not easily extended to MIMO controllers. Only a few methods have been proposed for MIMO controllers. Control saturation in MIMO controllers is a major contributor to the failure of many theoretically sound modern control design methods when applied to real plants in practice [Kapasouris].

This research was initiated to formulate a unified strategy for treating saturation in MIMO controllers, resulting in the following problem statement:

Saturation of a plant's input adversely affects the performance and robustness of any controller used with that plant. Several attempts to address this issue, have been described in the literature, but a successful unified strategy for all linear control systems has not yet been formulated. In particular, while the case for SISO systems has been well researched, there does not exist a unified practical and optimal engineering strategy to deal with saturation in MIMO control systems.

This research stemmed from the idea that it should be possible to prevent the windup and associated performance degradation resulting from controller saturation by preventing saturation from occurring in the first place.

It is hypothesised that it should be possible to condition the error signal input into the controller in such a way that the performance of the control system would be optimal up to the saturation limit of the actuator and that deeper saturation (or windup) beyond the limit would be averted.

1.2 Objectives

The purpose of this work was to investigate the issues associated with control saturations in both SISO and MIMO control systems and to look at typical solutions to these problems and the extension of these solutions to MIMO systems. The aim was to formalise a unified approach to treating saturation in MIMO controllers with the following objectives in mind:

- The methodology must not restrict or prescribe the linear control theory that may be used to design the controller. The method must form a jacket or shell around the linear controller during the implementation phase and must be transparent during the design phase of the controller.

- The method itself must not be dependent on any plant model that may contain inaccuracies or unmodelled dynamics, nor should it depend upon state feedback or state estimation of the plant's state. That is, state feedback may be used in the linear controller [2] but not by the anti-saturation algorithm since it would make the anti-saturation mechanism only as optimal as the *sub-optimal* plant's state measurement or estimation. Furthermore, this would restrict the application of the method to the few cases where all the states are measured or estimated.

- The closed loop system must remain stable at all times. That is, the anti-saturation mechanism must not be the cause of instability in the closed loop system.

[2]Such as a linear quadratic regulator which employs full state feedback or a linear quadratic gaussian model based controller employing state estimation.

- The method must allow for the maximum error signal to be applied to the controller before saturating the control signal, ensuring optimal performance up to the saturation limit.
- There must be no integrators or slow dynamics that windup in the controller during periods of control saturation.
- The method should be practical, easy to apply and efficient to compute in real time. The method must not require off line computation of high dimensional surfaces that have large storage requirements for gain scheduling of the controller when on line.

In summary, an algorithm is sought that will form a shell around the linear controller to guide the actions of the controller in such a way that performance is optimal under the constraints of actuator saturation. The algorithm must not impose any particular linear control design method on the designer. The control engineer must be able to design the controller based on the linear control theory of own choice and then simply pass the controller description, the saturation limits and error signal to this shell and be able to rest assured that the controller will have optimal and robust performance under all conditions and inputs.

1.3 Scope of Research and Assumptions

The scope of this research is constrained by the following assumptions:

- It is assumed that the controlled system is unconditionally stable. That is, for any reduction in loop gain the system remains stable. By implication this means that the plant is assumed to be strictly stable as well, meaning the plant has no positive real poles.
- It is assumed that the controller was designed for an assumed linear plant using linear control theory. In particular it is assumed that the actuator saturation limits were ignored during the linear controller design phase. Saturation in controllers designed using nonlinear control methods and controllers designed by methods that consider the actuator constraints implicitly during the design phase are therefore not considered. It does not necessarily follow that the proposed algorithm cannot be applied to these controllers, only that the application was not investigated. The proposed algorithm may be applicable in certain cases.
- It is assumed that the implementation of the controller will be by digital computer because the proposed algorithm requires the on line numerical solution of a linear programming problem.

1.4 Methodology

This research was tackled by examining the effects that actuator saturation introduced into a control system in general and thereafter for SISO and MIMO control systems in particular. Some existing schemes for the treatment of the saturation condition, as described in the literature, were then examined to highlight their relative strengths and weaknesses.

A new algorithm was developed from first principles by analysing the time propagation of the controller's state that eventually leads to saturation. The proposed algorithm then applies the adage, *prevention is better than cure*, to ensure that the controller never enters into saturation. Most of the existing anti-saturation (or anti-windup) schemes described in the literature wait for saturation to occur before any action is taken to prevent further and deeper saturation.

The advantages and improvements offered by this new proposed methodology over existing anti-saturation methodologies, particularly for MIMO control systems, are demonstrated by comparative simulations.

1.5 Contribution of Research

A unified method is proposed to treat saturation in both MIMO and SISO controllers, subject to the scope and assumptions stated above. This method offers superior performance over existing MIMO anti-saturation schemes.

The anti-saturation problem is posed as a linear programming problem. A practical and efficient implementation of the algorithm is developed by transforming the problem into its dual linear programming form. The problem, in dual form, is then solved using the dual simplex method rather than the primal simplex method. The nature of the problem when expressed in dual form and the properties of the dual simplex method are harmonised to guarantee an initial basic feasible solution and an optimal bounded final solution in a finite, predictable and minimal number of iterations.

The resultant controller never saturates, hence cannot windup. Furthermore the resultant controller always applies the optimal control effort to the plant to minimise the error signal input as follows:

- The controller is governed such that while the future free response combined with the present forced response of the controller results in no saturation limits being exceeded, now or at some time in the future, the normal linear response of the controller prevails.
- When the future free response combined with the present forced response of the controller will result in a saturation limit being reached, now or at some time in the future, the present time input signal into the controller is optimally governed to prevent the saturation limit from being exceeded at any future time.

In order to determine the future response of the controller due to the present state of the controller (hence free response) and present error input (hence forced response), the controller state must be propagated into the future. The number of time steps of propagation into the future is referred to as the *Anti-saturation Horizon* in this research.

1.6 Organisation of Thesis

Throughout this thesis the emphasis is placed on practical issues. Control saturation is largely ignored by most texts on linear control theory because it is considered, correctly so, to be a nonlinear issue. Nevertheless, it is a practical problem that must be solved for the correct and reliable implementation of any realistic control system.

The current chapter, Chapter 1, introduced the control saturation problem, it motivated why this research was necessary and it stated the objectives of the research. The effects of control saturation are examined in more depth in Chapter 2. In Chapter 3, the results of a literature survey are presented to expose the existing methods for handling control saturation in both SISO and MIMO controllers. One piece of work stands out for MIMO systems, that of Kapasouris. This method is evaluated and used as a benchmark to compare the performance of the method proposed in this thesis. The merits and pitfalls of each of these methods are presented. In Chapter 4 a method, resulting from this research, is developed to handle saturation in both MIMO and SISO controllers in a unified and general way. Application of this method is, however, limited to unconditionally stable systems. A practical method for solution of this algorithm, using linear programming, is discussed in Chapter 5. Chapter 6 documents the adaptation of the proposed algorithm to the linear programming dual standard form. The reasons for, and application of the dual simplex method to solve the optimisation problem are also presented. In Chapter 7, simulations are performed to demonstrate the advantages of this new method over existing methods. Conclusions and recommendations for future research are presented in Chapter 8. The Appendices contain support material for the linear programming used in this thesis.

Chapter 2

Effects of Saturation

2.1 Overview

The main possible effects of actuator saturation on a control system are poor performance and/or instability [Middleton]. The mechanisms by which actuator saturation affect a control system are presented in this chapter. These mechanisms are categorised below and are discussed further in the sections that follow:

- **Windup**

  Both SISO and MIMO controllers that contain free integrators and/or slow dynamics suffer from the phenomenon termed integral windup [Middleton].

- **Gain decrease**

  The apparent loop gain of the system decreases as a result of actuator saturation. This affects both SISO and MIMO control systems. This has important implications for unstable plants and conditionally stable systems [Middleton, Ogata].

- **Control vector direction**

  The direction of the control signal vector may be altered by the saturation nonlinearity in MIMO systems. For SISO systems, the control signal is a scalar and thus direction is not a consideration [Kapasouris].

2.2 Terminology

A controller with n-states is said to have a *state vector*, $\mathbf{x} = [x_1, \cdots, x_n]^T$. The individual elements, x_i, of the state vector being the actual states of the controller. In a minimal state space representation, these states are orthogonal (linearly independent) to one another and can be plotted in n-dimensional Cartesian space as a vector, $\mathbf{x}$, where each axis represents the individual components, x_i, of this vector. Both SISO and MIMO controllers can have more than one state.

By definition, only MIMO controllers have more than one input and/or output. Assume the controller has m-inputs and p-outputs. In the same sense then as the n-dimensional state vector was defined, the m-dimensional input *error vector*, $\mathbf{e} = [e_1, \cdots, e_m]^T$, and the p-dimensional output *control vector*, $\mathbf{u} = [u_1, \cdots, u_p]^T$, are defined.

2.3 The Saturation Element

Mathematically, the action of a saturating actuator can be described as:

$$\hat{u} = \text{sat}(u) = \left\{ \begin{array}{lll} u_{max} & : & u \geq u_{max} \\ u & : & u_{min} < u < u_{max} \\ u_{min} & : & u \leq u_{min} \end{array} \right\} \tag{2.1}$$

where u is the input, $\hat{u}$ the output, u_{max} and u_{min} the upper and lower bounds of the actuator respectively.

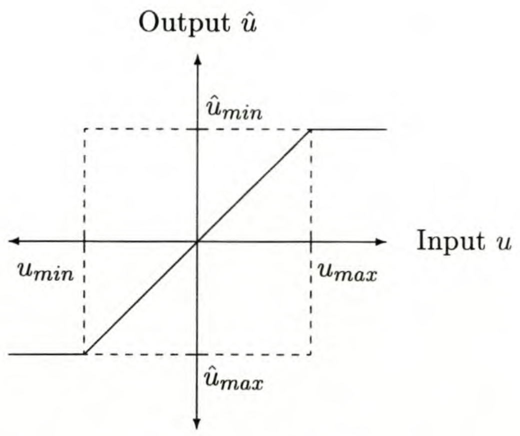

Figure 2.1: Saturation nonlinearity

2.4 Windup

The linear controller, in Figure 1.1, is designed ignoring actuator nonlinearities, but in practice the hard limits of the actuator will present themselves in the control loop for $u < u_{min}$ and for $u > u_{max}$. Controllers that contain pure integrators and/or slow dynamics will windup during the periods when the controller's output exceeds the plant's input range. The adverse effect of windup caused by the presence of such nonlinearities is in the form of performance degradation, as compared with the expected linear performance. This can result in large overshoots and/or poor response time in the output and sometimes even instability [Middleton].

Windup occurs when the error input, e, to the controller persists for sufficient time for the controller's output, u, to reach a saturation limit. If the error persists thereafter then the controller goes *deeper* into saturation in an attempt to drive (or control) the plant such as to reduce the error. However, the plant only experiences the saturated control signal, $\hat{u}$, which is less than the output the controller assumes is being applied to the plant. There is therefore an information mismatch, $|u| \neq |\hat{u}|$, between the controller's output and the plant's input. The linear design assumes that no mismatch can occur between the controller's output and the plant's input and thus the linear design becomes invalid during the periods of saturation.

The simulated response for the example system depicted in Figure 2.2 is shown in Figure 2.3.

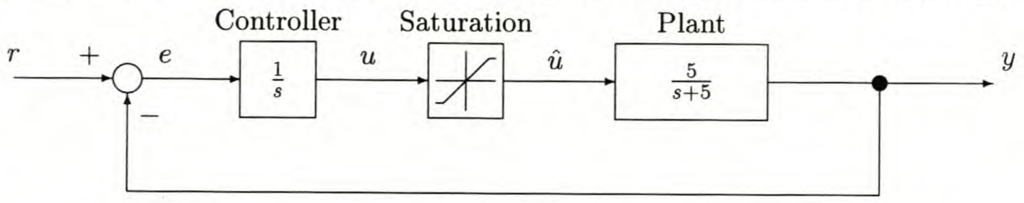

Figure 2.2: Block diagram of a control system with saturating actuator.

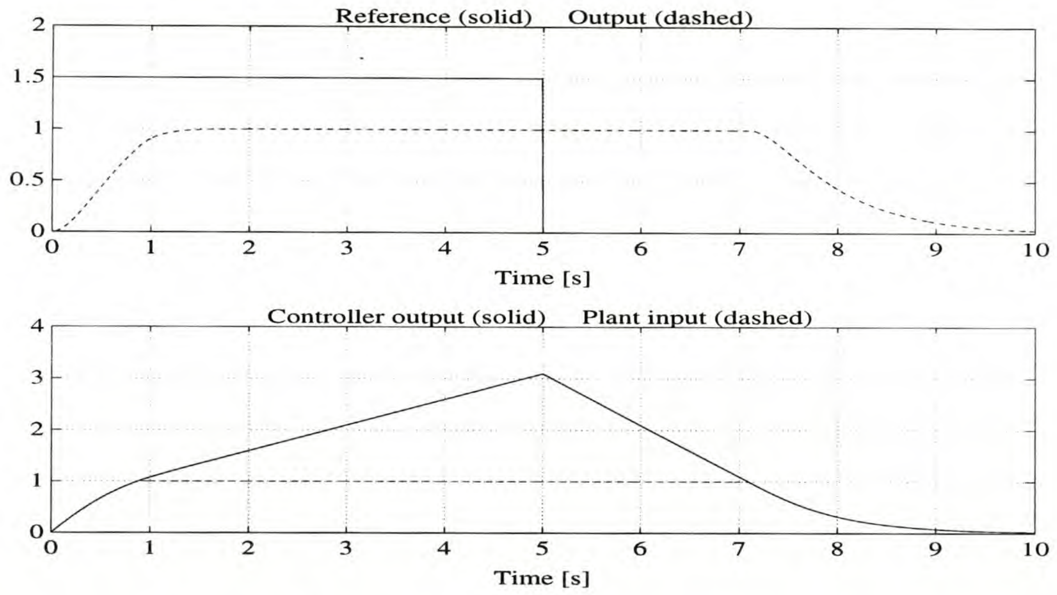

Figure 2.3: Response of a control with saturating actuator.

Initially the system is at rest. The controller is a pure integrator and the plant is a unity gain first order system with input constraints of ± 1. At time $t = 0$ s, a reference step of 1.5 units is applied to the system. The control signal output of the controller starts to increase in magnitude. When it reaches +1 units at time $t = 0.9$ s, the input to the plant saturates. Because the plant has unity gain, the output of the plant is restricted to unity as well. The error signal into the controller is therefore $e = 0.5$ units (the difference between the reference input and the plant output). Because the plant's output cannot increase further, due to the plant's input saturation limit, this error will persist indefinitely. The controller's state (a pure integrator) will therefore ramp or windup indefinitely. At time $t = 5$ s the reference command was reduced to zero again. By this stage, the controller's state is 3 times the input limit of the plant and therefore in deep saturation. It takes about 2 s for the controller's state to unwind to the linear region again. The plant's output therefore only responds to the change in reference command at about time $t = 7$ s after the 2 s delay period, resulting in a response degradation due to saturation.

Quite clearly this response delay will depend upon how long the controller was driven into saturation for. Thus the performance degradation can be anything from a brief moment for short periods of saturation to any large (infinite) time period. In an analogue control circuit there will be some ceiling (due to the power supply voltage) where the analogue circuitry will limit. In a digital implementation, in the worst case, a numerical overflow would result if the integration persisted for long enough. This could have catastrophic effects on the rest of the computerised system.

2.5 Gain Decrease

Heuristically, it can be seen that once in saturation, the incremental or small signal gain of the actuator becomes zero [Middleton]. That is, once the saturation limits have been reached, no further increase in the input will result in any corresponding increase in the output. The output remains constant at the value of the saturation limit. Because there is no change in the output for an incremental increase in the input one says that the incremental gain is zero.

Alternatively, from a transfer function point of view, the saturation element is a nonlinearity that has a range of gains starting at unity and reducing asymptotically to zero as the input amplitude increases beyond the linear region. The saturation element in Figure 2.1 can be modelled as follows. Here, for notational convenience alone, the saturation limits have been assumed to be ± 1. The arguments that follow are not restricted by this simplification and apply to the more general case where the limits are not unity, as well.

$$N(u) = \left\{ \begin{array}{ccc} 1 & : & |u| < 1 \\ \frac{1}{u} & : & |u| \geq 1 \end{array} \right\} \tag{2.2}$$

The normalised nonlinear gain, $N(u)$, is plotted for increasing values of amplitude, u, in Figure 2.4. The output, $\hat{u}$, is linear for amplitudes less than the saturation limit (a gain of unity) and decreases inversely with amplitude for amplitudes greater than the saturation limit. This illustrates the effective loop gain decrease in a control system when saturation occurs.

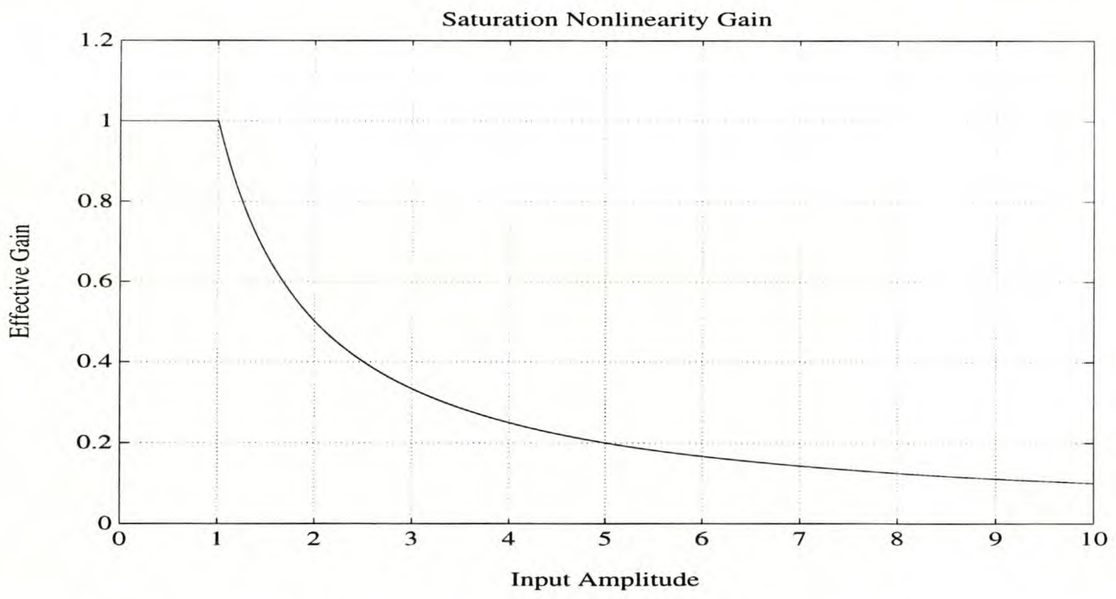

Figure 2.4: Transfer function Gain of the Normalised Saturation Nonlinearity

From a sinusoidal input describing function perspective the saturation element is an example of a sector nonlinearity with the following normalised transfer function:

$$N(u) = \left\{ \begin{array}{lcl} 1 & : & |u| < 1 \\ \frac{2}{\pi}\left[\operatorname{asin}\left(\frac{1}{u}\right) + \frac{1}{u}\sqrt{1 - \left(\frac{1}{u}\right)^2}\right] & : & |u| \geq 1 \end{array} \right\} \tag{2.3}$$

This describing function [Franklin *et al.*] confirms the nonlinear gain decrease of the saturation nonlinearity for input amplitudes greater than the saturation limits. Furthermore, describing function theory shows that the saturation nonlinearity does not introduce a phase shift between the input, u, and output, $\hat{u}$. That is, the saturation limit transfer function is independent of the frequency of the input signal. Thus it is sufficient to analyse and model the saturation element as a nonlinear gain.

2.5.1 Controller with Unstable Plant

An unstable plant may be stabilised by a closed loop control system employing the appropriate feedback network. Stability is achieved by the specific response of the applied control signal at the plant input. For this reason, once the required stabilising control signal can no longer be applied to the input of the plant, due to actuator saturation, the plant can no longer be stabilised. Another way of seeing this is that once the incremental gain of the control signal becomes zero (i.e. at the point of entry into saturation) the closed loop feedback system reverts to the open loop system of the original unstable plant and is thus unstable. It has long been known that strictly unstable plants with actuator constraints can never be globally stabilised [Middleton, Pajunen and Erdol].

For control systems with unstable plants, actuator saturation must be avoided at all times through judicious selection of the operation envelope of the controlled system such that there is always sufficient excess control signal available to stabilise the plant in the presence of disturbances. The operation envelope is defined by the range and rate of change of the reference commands to the controller. Saturation prevention, once saturation has occurred, cannot be an effective solution in a control system with an unstable plant. Instead, command conditioning that limits the action envelope of the system should be implemented to completely avoid saturation. This will result in a conservative system that remains well within the linear bounds of the controllable performance envelope.

2.5.2 Conditionally Stable Systems

A conditionally stable system is stable for values of the open loop gain lying between critical values, but is unstable if the open loop gain is either increased or decreased sufficiently. Such a system becomes unstable when large input signals are applied since a large signal can cause actuator saturation, which in turn reduces the open-loop gain of the system [Ogata].

The plant itself may be stable, but the combined effect of the plant and controller dynamics with the reduced loop gain due to saturation at the plant input may result in an unstable system. Simple describing function arguments show that the combination of a conditionally stable control system and a saturating actuator can give rise to limit cycle behaviour. It is advisable to avoid such control system designs [Middleton]. The addition of a well designed compensating network will eliminate conditional stability [Ogata].

Note that actuator saturation results only in a decrease in loop gain, not an increase. Therefore systems that are conditionally stable for increased loop gain but that are unconditionally stable for reduced loop gain will remain stable even during actuator saturation.

Consider the system [Ogata] shown in Figure 2.5. The root locus for this system is plotted in Figure 2.6. It can be seen that this system is stable for limited ranges of the value of K, namely for $0 < K \leq 14$ and $64 < K \leq 195$, while the system is unstable for $14 < K \leq 64$ and $K > 195$.

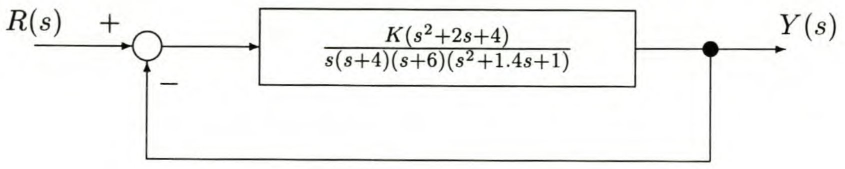

Figure 2.5: Block diagram of the conditionally stable control system

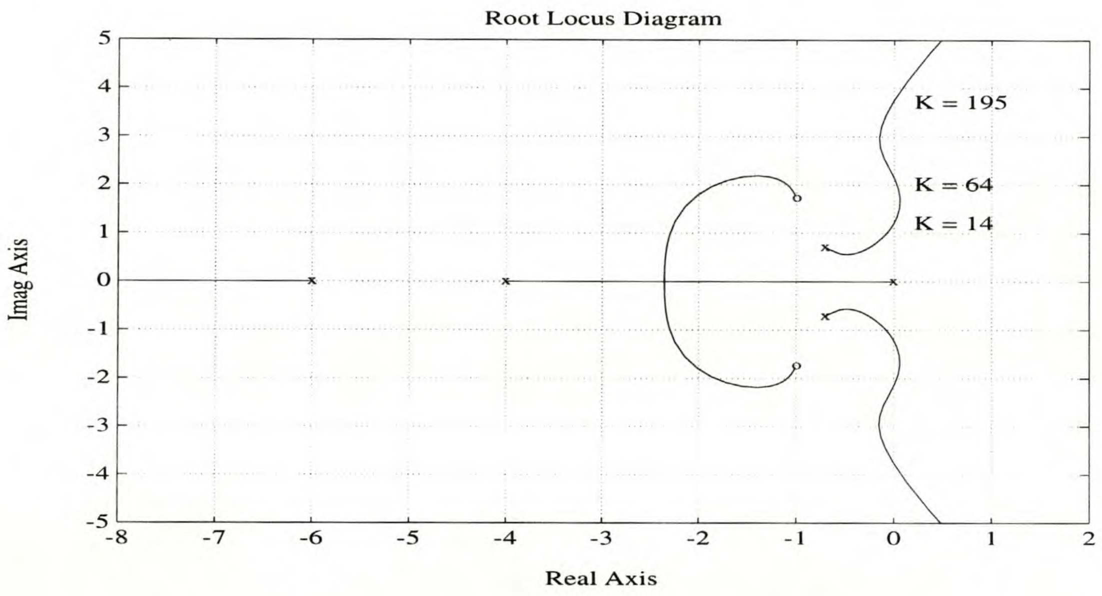

Figure 2.6: Root locus of the conditionally stable control system

It is noted that most of the current work into saturation avoidance is limited to stable open loop plants [Pachter and Miller].

2.6 Control Vector Direction

Kapasouris states:

"Almost every current design methodology for linear systems inverts the plant and replaces the open loop system with a desired design loop. This inversion is done through the controls with signals at specific frequencies and directions. The saturations alter the direction and frequency of the control signal and thus interfere with the inversion process. The main problem is that although both the compensator and the plant are multivariable highly coupled systems, the saturations operate as SISO systems. Each saturation operates on its input signal independently from the other saturation elements."

Figure 2.7 shows an illustration of four different control vectors $u_\alpha^a, u_\alpha^b, u_\beta^a, u_\beta^b$ of a two output controller are mapped by the saturation element to only two vectors $\hat{u}_\alpha$ and $\hat{u}_\beta$. Note how both the magnitude and direction of the resultant control vector is affected. This argument applies to controllers with more than two outputs as well, however, it is more difficult to represent this graphically.

- The saturation limits convert the desired control vectors u_α^a and u_α^b to $\hat{u}_\alpha$. Therefore both control vectors that violate different control bounds are converted by the saturation bounds to the same control vector.
- Likewise, the control vectors u_β^a and u_β^b are reduced to the same vector $\hat{u}_\beta$ which is applied to the input of the plant.

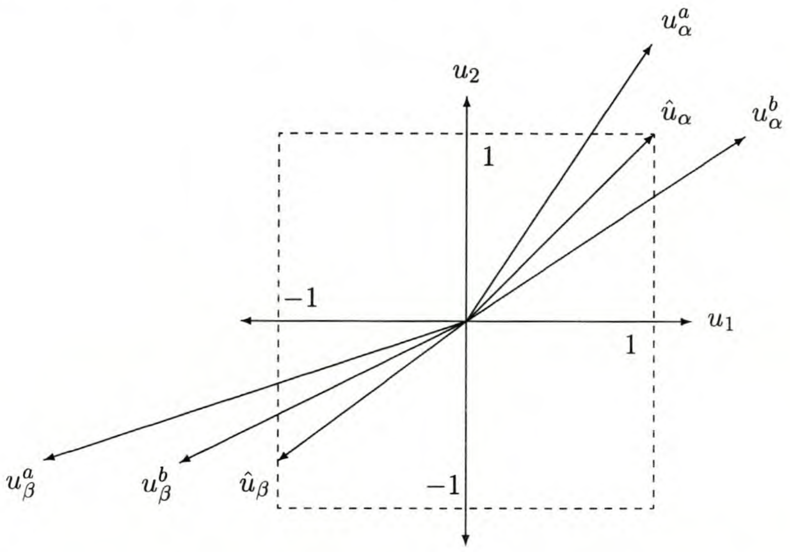

Figure 2.7: Examples of control directions at the input of the saturation $u_\alpha^a, u_\alpha^b, u_\beta^a, u_\beta^b$ and at the output of the saturation $\hat{u}_\alpha, \hat{u}_\beta$.

2.7 Summary

The following main conclusions are drawn at this stage:

- It is not possible to ensure global stability in the face of actuator saturation for a control system with an unstable plant. It is frivolous to attempt to develop an anti-saturation algorithm for the case of an unstable plant. Rather one must condition the reference signal to conservatively limit the operational envelope of the control system to the actuator's linear region only. The same applies to systems that are unstable for reduced loop gain.

- In MIMO control systems, the saturation elements on the controller outputs act in a scalar sense and thus can alter both the magnitude and direction of the effective control vector.
- The saturation element can be treated as a nonlinear gain. There are no phase dependancies.

Chapter 3

Prevention of Saturation

3.1 Overview

Actuator saturation cannot always be completely avoided. Consequently several methods have been devised to reduce the effects of actuator saturation. These methods are the subject of this chapter. Middleton sites three broad principles that can be applied to reduce the effects of actuator saturation, namely:

- **Avoid conditionally stable control systems where possible.**

  Note that a controller for a plant can be designed that gives unconditional stability if and only if the plant has:

  - No poles with positive real parts.
  - No purely imaginary poles of repetition greater than 2.

  Therefore, a plant that is open loop strictly unstable can only be conditionally stabilised.

- **Avoid applying unrealistic commands to a control system.**

  The implementation of this type of idea is often termed a reference governor or reference conditioner for the system. Refer to Bemporad *et al.* for an example of a nonlinear command governor for constrained control systems. Kapasouris also shows a development of a reference governor.

- **Utilise saturation feedback to implement the controller.**

  Saturation feedback is a signal derived by taking the difference between the controller output and the plant input and feeding this difference back into the controller to prevent windup and deeper saturation. When the controller output is within the bounds of the actuator, the difference signal will be zero and no corrective action will occur. During saturation the difference signal will be non-zero and proportional to the depth of saturation.

  Kothare *et al.* presents a summary of all known linear time-invariant anti-windup schemes.

The first two principles are not the topic of this research. The last principle, employing saturation feedback, is discussed in the sections that follow, and is largely sourced from Kothare *et al.* Comments are made to highlight the strengths and weaknesses of each method and its suitability for application to MIMO control systems.

3.2 SISO Windup Prevention Schemes

3.2.1 Anti-reset Windup

The windup phenomenon was first encountered when using PI and PID controllers with saturating actuators. One of the earliest works to overcome windup was by Fertik and Ross (1967). Their strategies have been referred to as *anti-reset windup, back calculation and tracking* and *integrator resetting* [Kothare *et al.*]. Integrator windup is only a special case of the more general case whereby any controller with relatively slow dynamics or unstable modes (pole at the origin) can windup.

The equations of the PI controller shown in Figure 3.1 are:

$$\begin{aligned} K(s) &= k(1+\tfrac{1}{\tau_i s}) \\ u &= k(e+\tfrac{1}{\tau_i}\int_0^t e\,dt) \\ \hat{u} &= \text{sat}(u) \\ e &= r-y \end{aligned}$$

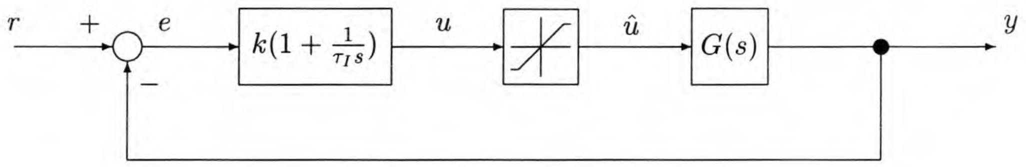

Figure 3.1: PI control of plant $G(s)$.

Windup occurs when the error, e, is a non-zero constant for a sufficient period that the control signal, u, exceeds the control limits. The integrator continues to integrate this error signal resulting in deeper saturation. Refer to Chapter 2 for an example.

An *ad hoc* approach to preventing windup is to simply stop the integral action in the controller when the output of the controller exceeds the saturation limit. This approach is often used in computer implemented controllers where it is relatively easy to stop the integral action in software. The exact implementation depends upon the insight and ingenuity of the programmer.

A formal approach is to use an extra feedback path in the controller as illustrated in Figure 3.2. This method is known as *classical anti-reset windup*. The difference between u and $\hat{u}$ is fed back via the gain $\frac{1}{\tau_r}$ to the input of the integrator. This feedback signal attempts to drive the difference between the controller output and the plant input to zero by recomputing the integral such that the output of the controller remains exactly at the saturation limit [Kothare *et al.*]. This prevents the integrator from winding up.

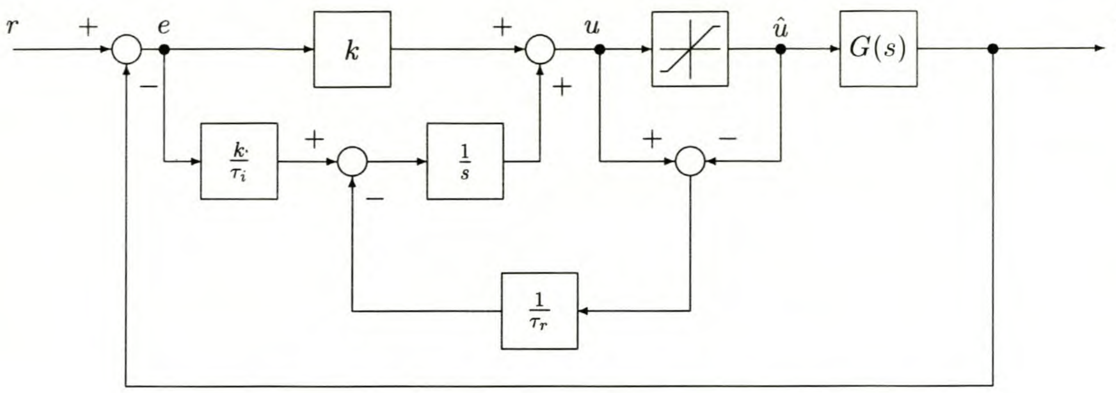

Figure 3.2: Classical anti-reset windup.

The modified controller output equation is:

$$u \quad = \quad k[e + \tfrac{1}{\tau_i}\int_0^t (e - \tfrac{\tau_i}{k\tau_r}(u - \hat{u}))\, dt]$$

Clearly when $u = \hat{u}$ the modified controller output equation is identical to the linear PI controller output equation, as would be expected.

An alternative implementation to the classical anti-reset windup is shown in Figure 3.3. See Astrom and Hagglund (1988) for further details. This seemingly different anti-reset windup scheme for PI controllers is identical for the well known heuristic choice of $\tau_r = \tau_i$ [Kothare *et al*].

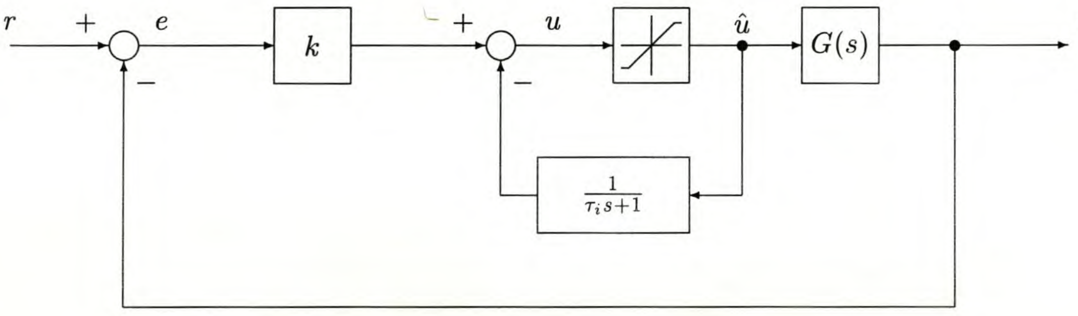

Figure 3.3: Alternate anti-reset windup implementation.

Comment:

- These techniques are only suited to SISO controllers and in particular to PI controllers.
- In a MIMO controller with p-output control signals one would need to determine the $p \times p$ mix of transfer function feedback paths to deal with the presently saturated controller output. The saturated controller output may well be the sum of more than one integrator, neither of which may be exceeding the saturation limit individually. No procedures or guidelines are given in the literature on how to extend this anti-windup mechanism to MIMO systems.
- By nature of the formulation, the anti-windup mechanism only engages when the difference between u and $\hat{u}$ is non-zero. Technically, by this stage the controller is already

in saturation. While these methods have been applied successfully they are nevertheless suboptimal in that they do allow a brief period of saturation and hence windup to occur.

3.2.2 Conventional Anti-windup (CAW)

It is reported by Kothare *et al.* that Doyle (1987) adopted a similar philosophy to that of anti-reset windup. Thus, in some sense, CAW can be considered a direct extension of the anti-reset windup to general controllers. CAW is not too dissimilar to the methods described above. An attempt is made by the method to cater for MIMO systems. Compensation is provided by feedback of the difference $u - \hat{u}$ through a high gain matrix, $\mathbf{X}$, to the controller input as illustrated in Figure 3.4. For a MIMO controller, typically $\mathbf{X} = \alpha\mathbf{I}$, where the scalar $\alpha >> 1$.

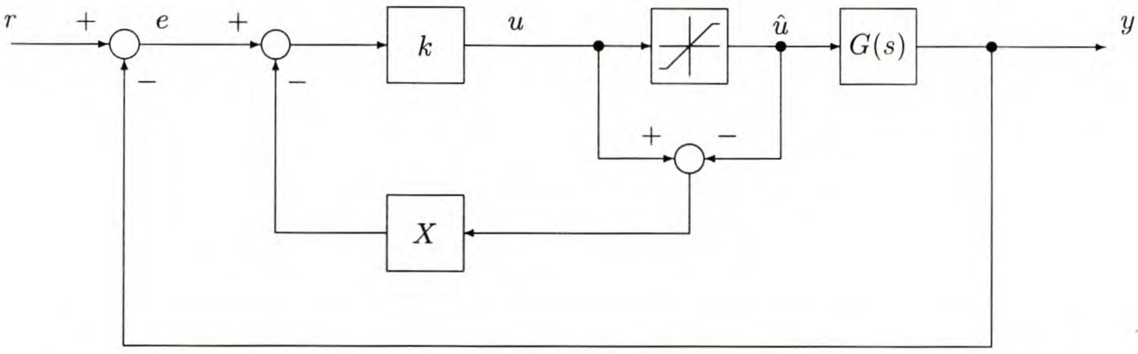

Figure 3.4: Conventional anti-reset windup.

Comment:

- The gain matrix $\mathbf{X}$ must be chosen to somehow relate the controls u_m to the error inputs e_l. The gain is static and could therefore only cater for the static (DC gain) relationship in the controller between controller input and controller output.
- The choice of $\mathbf{X} = \alpha\mathbf{I}$ would be a very poor choice for cross-coupled controllers because the saturations would be acting in a SISO sense and directly relating particular controller outputs to controller inputs and ignoring the cross-coupling dependancies and side effects.
- Since $\mathbf{X}$ has no dynamics, it can only cancel the error input when the residue between u and $\hat{u}$ is non-zero. There is the likelihood therefore of a limit cycle occurring at the saturation limit as the controller *bounces* against and away from the saturation limit in alternate controller update cycles, as the error cancellation is effectively switched on and off. This will be particularly evident in digital implementations.

3.2.3 Hanus Conditioned Controller

This conditioning technique is attributed to Hanus and Kinnaert, 1989. It is an extension of the back calculation method proposed by Fertik and Ross (1967) [Kothare *et al*]. In this technique the reference is modified to what is called a *realisable reference* r^r such that if the realisable reference had been applied to the controller instead of the actual reference then the controller would not have issued a saturating control. The realisable reference is chosen to maintain consistency with the controller state and the plant input, thus avoiding windup.

The equations for the anti-windup conditioned controller are:

$$\begin{aligned}\dot{x} &= Ax + B(r - y)\\ u &= Cx + D(r - y)\\ \hat{u} &= \mathrm{sat}(u)\end{aligned}$$

Following Hanus *et al., (1987)*, a realisable reference can be applied to the controller such that the output of the controller is $\hat{u}$. Thus

$$\begin{aligned}\dot{x} &= Ax + B(r^r - y)\\ \hat{u} &= Cx + D(r^r - y)\end{aligned}$$

Subtracting u from $\hat{u}$ gives

$$\hat{u} - u = D(r^r - r)$$

Hence

$$r^r = r + D^{-1}(\hat{u} - u)$$

Combining the equations leads to the conditioned controller:

$$\begin{aligned}\dot{x} &= (A - BD^{-1}C)x + BD^{-1}\hat{u}\\ u &= Cx + D(r - y)\\ \hat{u} &= \mathrm{sat}(u)\end{aligned}$$

Several drawbacks have been reported for this scheme. Kothare *et al.* sites the following reports:

- The strategy fails for controllers having rank deficient D matrices (Hanus and Peng, 1992).
- The strategy is an inflexible design tool since it modifies the controller without using any additional tuning parameters for optimising the nonlinear performance.
- The technique can only handle one saturation level, either the upper or the lower limit.

Comment:

- The error signal that drives the controller into saturation is made up of two components, namely the reference and the plant output.

  $$e = r - y$$

  While a realisable reference takes care of the case where the reference could have been the cause for saturation, it does not take care of the case where the plant output causes the saturation. The plant output can be modelled as the ideal plant output summed with an unknown plant output disturbance as shown below:

$$y = y_p + d_o$$

$$\begin{aligned} e &= r^r - y \\ &= r^r - (y_p + d_o) \end{aligned}$$

All the signals, with exception of the disturbance can be accounted for in constructing the realisable reference. Thus it is not possible to prevent the controller from saturating under all conditions.

3.2.4 Generalised Conditioning Technique (GCT)

The generalised conditioning technique attempts to address the problems with the Hanus conditioning controller. Two modifications are proposed:

- The input conditioning mechanism is improved by introducing a degree of *caution* so that the change in the modified set point at controller saturation is smoothed [Walgama et al., 1992: Section 4]. The realisable reference r^r is modified by introducing the user chosen parameter ρ as follows:

$$r^r = r + (D + \rho I)^{-1}(\hat{u} - u), \qquad \text{where} \quad 0 \leq \rho \leq \infty.$$

- A more general modification is presented in Walgama *et al., 1992: Section 5* where the prefiltered reference r_f command is used instead of the direct reference r. The derivation of the controller equations can be followed in Kothare *et al.* The method, although overcoming some of the problems in the Hanus conditioned controller introduces new problems in that the filter must be tuned to obtain the desired transient performance of the saturated system. No guidelines are given as to how to tune this filter while maintaining acceptable performance and stability.

Comment:

- The modification, however, introduces the new problem of finding a suitable and optimal choice for ρ.

3.2.5 Observer-based Anti-windup

The windup problem results in the states of the controller not corresponding to the control signal input of the plant due to the saturation nonlinearity. In order to correctly estimate the controller's states during saturation an observer is introduced into the controller. By obtaining the correct or consistent state estimates, windup is avoided. Kothare attributes this work to Astrom and Hagglund, 1988 and to Astrom and Rundqwist, 1989.

Assuming that (C,A) is detectable, the nonlinear observer equations for the controller are given by:

$$\begin{aligned} e &= r - y \\ \dot{\hat{x}} &= A\hat{x} + Be + L(\hat{u} - C\hat{x} - De) \\ u &= C\hat{x} + De \\ \hat{u} &= \text{sat}(u) \end{aligned}$$

where $\hat{x}$ is an estimate of the controller state and L is the observer gain.

These nonlinear equations describe the observer as well as the controller. In the linear region when $\hat{u} = u$ and hence $\hat{x} = x$ the innovations multiplied by the gain matrix L are zero and the linear controller equations for $K(s)$ remain:

$$\begin{aligned} e &= r - y \\ \dot{x} &= Ax + Be \\ u &= Cx + De \end{aligned}$$

When $\hat{u} \neq u$, during saturation, the observer introduces a corrective action due to the non-zero innovations $(\hat{u} - C\hat{x} - De)$ through the gain matrix L to the controller's states.

Comment:

- The observer gain must be determined such that the controller is stable, but also fast enough so that it can track the states of the real plant.
- The observer gain is chosen to minimise some cost function. The gain will only be optimal for this cost function and not for a different cost function. No guidelines are given for choosing the cost function. Therefore the same controller, implemented by different engineers may well have difference performance characteristics depending on their choice of cost function.
- Brief periods of saturation are still possible because the scheme only takes action once the plant output has saturated.

3.2.6 Internal Model Control (IMC)

This method, although never originally intended as an anti-windup scheme, has potential for application to the anti-windup problem, for cases where the plant is stable [Kothare].

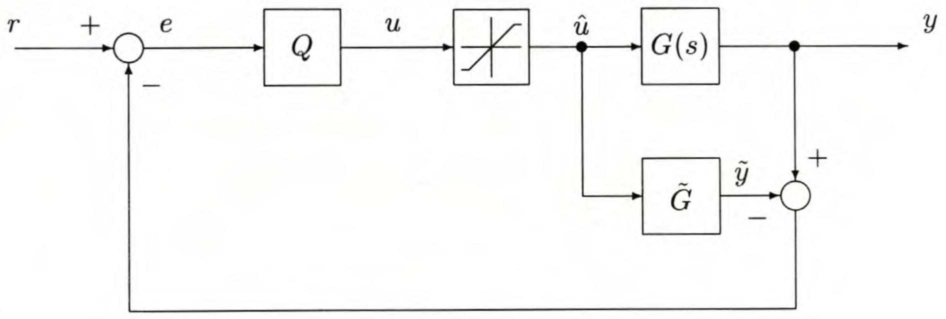

Figure 3.5: The IMC structure.

The plant model, $G(s)$, is driven by the saturated control $\hat{u}$ as shown in Figure 3.5. Kothare states that, under the assumption that the plant model is exact, it can be shown that the IMC structure remains effectively open loop, since y and $\hat{y}$ cancel exactly and that stability is guaranteed by the stability of the open loop plant $G(s)$ and the IMC controller Q. Furthermore the IMC structure offers the opportunity to implement complex (even nonlinear) control algorithms without generating complex stability issues providing the plant model is exact.

Kothare continues that although the IMC implementation guarantees nominal stability in the presence of actuator constraints for stable plants, the cost paid for this global stability is *sluggish* performance. This is because (for exact plant models) the output of the controller is independent of the plant output in both linear and nonlinear regimes. Although this does not matter in the linear regime, the implication in the nonlinear regime is that the controller is unaware of the effects of its control actions on the output of the plant. The sluggishness effect is reported to be most pronounced when the IMC controller has fast dynamics that are *chopped off* by the saturation. Furthermore the IMC controller does not give satisfactory performance for a saturating system unless it has been specifically designed to optimise the nonlinear performance.

Lastly, Kothare sites Zheng *et al.*(1994) who have proposed a modified IMC structure for optimising performance in the face of saturation. The controller $Q(s)$ is factorised as

$$Q(s) = [1 + Q_2(s)]^{-1}Q_1(s)$$

and implemented as shown in Figure 3.6. See Zheng et al. (1994) and Goodwin et al. for details.

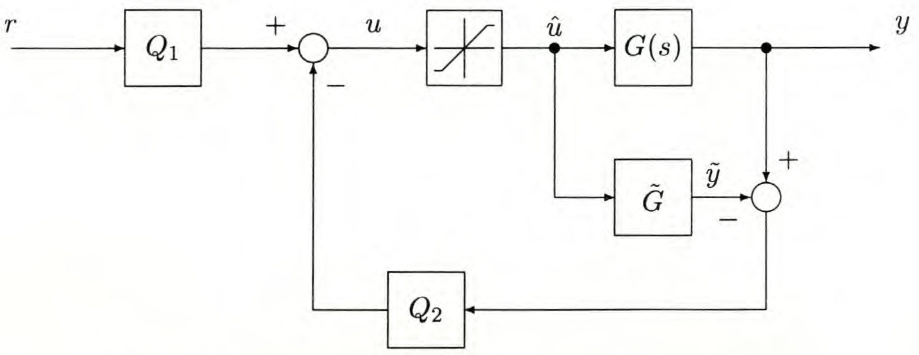

Figure 3.6: Anti-windup implementation of the IMC.

Comment:

- In both of these schemes, if $G(s)$ and $\hat{G}(s)$ are exact then the only way that their outputs could differ is if an unknown disturbance, d_o, was acting on the output. In essence then, this IMC scheme is an open loop reference command follower and a closed loop disturbance regulator.
- In practice is would never be possible to obtain an exact model of the plant. If it were, then one could simply premultiply the plant by the plant model inverse resulting in an input to output transfer function of unity. Thus the output would exactly follow the reference command and there would be no need for feedback control.

3.2.7 The Extended Kalman Filter

Anti-windup compensation for controllers that are observer based (for example, LQG controllers) revolves around maintaining consistency of the controller (and hence the observer's) states with the actual input of the plant. This requires a modification to the observer's equations such that the state estimate is based on the actual input $\hat{u}$ to the plant [Kothare].

According to Kothare, the classical separation principle of the observer/state-feedback controller is lost with this implementation during the period of saturation. Thus even though the controller and observer are stable, the overall closed loop nonlinear system need not be asymptotically stable.

3.3 MIMO Windup Prevention Schemes

3.3.1 Scalar Error Governor

The method proposed by Kapasouris is specifically designed to deal with multiple saturations in MIMO controllers. As discussed in Chapter 2, the two major problems that multiple saturations can introduce into the performance of the MIMO system are:

- The windup problem.
- Multiple saturations may alter the direction of the control vector.

To prevent the above two effects from occurring, Kapasouris introduces an operator called the Error Governor (EG) into the controller as shown in Figure 3.7. Here the signals are vectors because the system is MIMO.

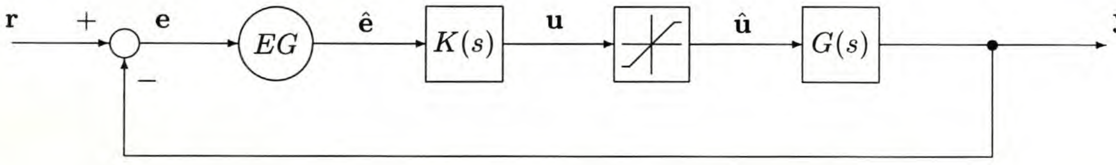

Figure 3.7: MIMO controller with Error Governor (EG)

The saturation is effectively transferred, by the action of the Error Governor, from the controller output to the controller input. Kapasouris makes the following claim:

"The selection of the EG operator will be such that the controls will never saturate; and if, for example, the compensator was designed to invert or partially invert part of the plant, then the inversion process will not be disturbed by the saturation and both G and K will remain linear and equal to GK. In the closed loop system with the EG operator the compensator will never cause windups. The integrators and slow dynamics of the compensator will never cause the controls to exceed the limits of the saturation and thus windups can never occur."

The EG is implemented as a time-varying gain. With the EG operator at the error signal, the system will remain unaltered (linear) when the references and disturbances are such that they do not cause saturation. For large references and disturbances the EG operator will scale the error signal to ensure that the controls will never saturate. The value of the EG operator gain $\lambda(t)$ is thus bounded to the range, $0 \leq \lambda(t) \leq 1$. The error governor can therefore reduce the loop gain and for this reason this control structure is only useful in unconditionally stable systems. In order to maintain direction of the control vector (i.e. linearity) Kapasouris scales all the error inputs equally by the same factor $\lambda(t)$ as shown in Figure 3.8

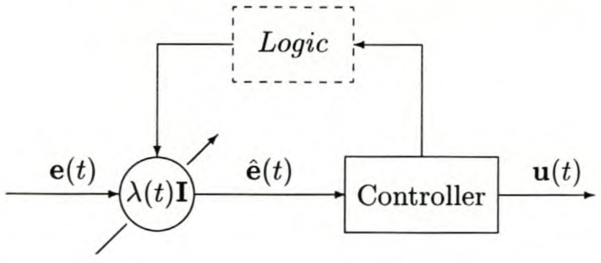

Figure 3.8: Time-varying Error Governor gain.

In dealing with the anti-saturation problem, that is the design of a time varying gain that bounds the outputs of a linear system, Kapasouris considers three regions, namely the:

- Internal region, a space where no saturation occurs.
- External region, a space where saturation has occurred.
- Boundary region, a surface between the internal and external regions.

The mathematical background supporting the development of Kapasouris's algorithm can be found in Kapasouris, Athans, and Stein (1988 IEEE). Central to the notation is the hyper space of admissible states, denoted by $\mathbf{B}_{A,C}$. This hyper space $\mathbf{B}$ is a set of states due to the controller dynamics represented by the A-matrix subscript and their mapping to the control vector represented by the C-matrix subscript. The construction of $\lambda(t)$ is performed as follows. Let the compensator $K(s)$ be defined by the state space system:

$$\begin{aligned} e(t) &= r(t) - y(t) \\ \dot{x}(t) &= Ax(t) + B\lambda(t)e(t) \\ u(t) &= Cx(t) \end{aligned}$$

Note: A slightly modified algorithm is given for controllers with a D matrix, i.e. $u(t) = Cx(t) + De(t)$.

It is assumed that the initial state of the controller is such that the output of the controller is unsaturated, i.e. $x(t) \in \mathbf{B}_{A,C}$ where $\mathbf{B}_{A,C}$ is the set of admissible states the controller can assume without leading to saturation.

Following Kapasouris, a function $g(x)$ and a set $\mathbf{B}_{A,C}$ are defined and then the construction of $\lambda(t)$ follows.

$$g(\mathbf{x}_0) : g(\mathbf{x}_0) = ||\mathbf{u}(t)||_\infty$$

where

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) \qquad\qquad \mathbf{x}(0) = \mathbf{x}_0 \\ \mathbf{u}(t) &= \mathbf{C}\mathbf{x}(t) \\ \mathbf{B}_{A,C} &= \{\mathbf{x} : g(\mathbf{x}) \leq 1\} \end{aligned}$$

For $g(\mathbf{x})$ to be finite, for all $\mathbf{x}$, the compensator has to be neutrally stable. With finite $g(\mathbf{x})$ and for no $\mathbf{D}$ matrix, the EG operator $\lambda(t)$ is given by:

For every time t choose $\lambda(t)$ as follows:

- If $\mathbf{x}(t) \in \text{Int } \mathbf{B}_{A,C}$

  If the controller's states $\mathbf{x}(t)$ are within the internal (Int) region of the admissible states hyper space $\mathbf{B}_{A,C}$ then no saturation will occur and so the error governor is unity. i.e. the EG plays no role.

  $\lambda(t) = 1$.

- If $\mathbf{x}(t) \in \text{Bd } \mathbf{B}_{A,C}$

  If the controller's states are on the boundary (Bd) of the admissible states hyper space, then chose the largest $\lambda(t)$ such that:

$$0 \leq \lambda(t) \leq 1$$

$$\limsup_{\epsilon \to 0} \frac{g(\mathbf{x}(t) + \epsilon[\mathbf{A}\mathbf{x}(t) + \mathbf{B}\lambda(t)\mathbf{e}(t)]) - g(\mathbf{x}(t))}{\epsilon} \leq 0$$

  or for plants where $g(\mathbf{x})$ is differentiable choose the largest $\lambda(t)$ such that:

$$0 \leq \lambda(t) \leq 1$$

$$\mathrm{D}\ g(\mathbf{x}(t))[\mathbf{A}\mathbf{x}(t) + \mathbf{B}\lambda(t)\mathbf{e}(t)] \leq 0 \qquad \forall \quad t > 0$$

  where D $g(\mathbf{x}(t))$ is the Jacobian matrix of $g(\mathbf{x}(t))$.

- if $\mathbf{x}(t) \ni \mathbf{B}_{A,C}$

  If the controller's states are outside of the admissible set then then choose $\lambda(t)$ such that:

$$0 \leq \lambda(t) \leq 1$$

$$\limsup_{\epsilon \to 0} \frac{g(\mathbf{x}(t) + \epsilon[\mathbf{A}\mathbf{x}(t) + \mathbf{B}\lambda(t)\mathbf{e}(t)]) - g(\mathbf{x}(t))}{\epsilon} \leq 0$$

  is a minimum.

An additional important property, besides avoiding saturation, is that the saturation operator no longer alters the direction of the control vector, nor the relative magnitude of the controls. Thus, if the compensator inverts part of the plant, the saturation does not alter the inversion process [Kapasouris].

3.3.2 Scalar Reference Governor

This method, originally proposed by Kapasouris and also worked on by Patcher and Miller, derives a conditioned reference command from the applied reference command so as to avoid the controller attaining an inadmissible state and hence avoid issuing a control that violates the control bounds. The method of Kapasouris is reported by Patcher and Miller to be computationally intensive. Another problem is that the reference governor only operates on the reference signal, not the error signal, hence the influence of output disturbances are not catered for in the anti-saturation scheme. The reference governor does have a particular usefulness in systems with unstable plants in that it can be used to constrain the operational envelope such as to avoid saturation induced by excessive reference commands. Bemporad *et al.* also derive a command (reference) governor mechanism, from a different perspective to that of Kapasouris, based on ideas originating from predicative control.

It should also be noted that although the linear system may be Bounded-Input-Bounded-Output (BIBO) stable, that saturation mitigation strategies which solely modify the incoming reference in order to avoid saturation do not necessarily yield an overall BIBO stable control system [Patcher and Miller].

3.3.3 Neural Net Based Suboptimal Controller

In a paper by Sznaier and Damborg 1992, a suboptimal minimum time controller, based upon on-line optimisation, for systems with linear state and control inequality constraints is proposed. The algorithm is suited to MIMO systems and an example is given. The algorithm yields asymptotically stable closed loop systems but prescribes the design methodology to be used. The algorithm can be coded as an analogue neural network yielding fast solution which could be advantageous in environments where digital computation was limited, although this is hardly a consideration only a decade later. They note that the controller is overly conservative.

3.3.4 Full State Feedback

Methods derived by Castelan *et al.*, Lin *et al.* and Patcher and Miller all rely upon full state feedback of the plants state to guarantee that any state trajectory emanating from a given set of admissible initial states remains in that set. Full state feedback is now days more likely possible given modern compact sensors and in the case of aircraft where space and cost is not such a premium as would be the case in a missile. Nevertheless, under the scope of this research, an anti-saturation scheme that did not rely upon plant state feedback was sought. Therefore these methods were not pursued further.

3.4 Summary

In this chapter, existing anti-windup and anti-saturation schemes were considered. The primary conclusion was that most schemes were suited only to SISO systems. They were sub-optimal in that they only took action once the controller had already entered into saturation. They should rather be classified as neither anti-windup nor saturation avoidance, but rather as performance enhancement schemes [Patcher and Miller]. The only MIMO scheme of note was that developed by Kapasouris. This was an attempt to completely avoid saturation from occurring, thus avoiding the negative consequences of saturation completely. The method has been reported to be computationally intensive and complex which has practical limitations in the application of the method [Patcher and Miller]. The method has only one degree of freedom, namely the scalar error governor $\lambda(t)$, whereas the controller may have multiple inputs and/or outputs and thus multiple degrees of freedom.

Sznaier and Damborg (1990), state that in spite of the significance of the saturation problem, it has not been solved satisfactorily and most of the design procedures available are severely restricted in their domain of application.

Chapter 4

Proposed Anti-saturation Algorithm

4.1 Overview

In the previous chapters the controller saturation issue was considered in detail. The most significant result was that saturation of the plant's input adversely affects the performance and robustness of all control systems. Several attempts have been made in the literature on the subject to address this issue but a successful unified strategy for all linear control systems has not yet been formulated.

In this chapter the foundation is laid for the development of an anti-saturation algorithm for MIMO control systems. The algorithm can be applied to SISO systems as the special case of the more general MIMO case.

The proposed algorithm introduces, as did the method of Kapasouris, an *error governor* into the input path of the controller. Kapasouris used a *scalar* error governor to scale the entire error vector. The proposed algorithm uses a *vector* error governor to scale the individual elements of the error vector.

4.2 Preliminaries

As was illustrated in Chapter 2, the effects of controller saturation in a control system are threefold, namely:

- The controller dynamics windup.
- The loop gain decreases.
- In a MIMO control system the direction of the control vector is altered.

An error governor (EG), depicted in Figure 4.1, was introduced into the forward path of the feedback control loop to provide a mechanism to deal with saturation in the controller.

Note that the error governor can only pass or attenuate the error signal, e. It cannot amplify or change the polarity of this signal. The mechanism by which the error governor addresses the effects of saturation on a control system are discussed below:

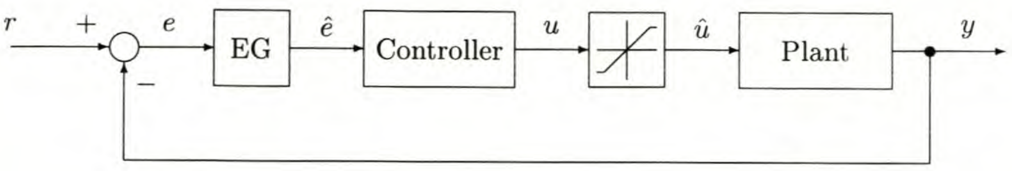

Figure 4.1: Control system with error governor EG.

- **Windup**

  The idea is to regulate or govern the error input (energy into the controller) such that the controller output never exceeds the saturation limits of the actuator. Thus the controller dynamics can never windup and the first effect of saturation has been addressed.

- **Loop gain**

  In the scope of this thesis the plant was assumed to be stable. Also the control system (combined controller and plant) was assumed to be unconditionally stable for loop gain decreases. These assumptions were made on the grounds that it is well known that is not possible to globally stabilise an unstable plant or a conditionally stable control system in the face of actuator saturation.

  The effective loop gain decrease that would have been introduced by the saturation element has now been shifted to (or replaced by) the controlled loop gain decrease introduced by the error governor. Thus the unavoidable loop gain decrease due to saturation has been moved from the controller output (plant input) to the controller input where it can be effectively managed to control the otherwise negative effects of saturation.

- **Control vector direction**

  Kapasouris maintained a linear relationship between the components of the control vector and hence the direction of the control vector by linearly scaling all the error inputs by the same scale factor, effectively maintaining the linear state of the controller. Kapasouris argued that maintenance of the direction of the control vector was critical in ensuring correct and stable operation in controllers that had partially inverted some of the plant's dynamics.

  Besides the fact that the newly proposed algorithm is developed along different lines to the algorithm of Kapasouris's and the fact that the implementations are completely different, the *fundamental* difference between these two algorithms is the *divergence* in philosophy taken regarding maintenance of the direction of the control vector.

4.3 Divergence from Kapasouris' Methodology

In Chapter 2, Figure 2.7 and the related argument illustrated what happened when individual saturation elements acted on the individual components of a MIMO control vector in a SISO sense. Although this illustration provided a good insight into what physically happens during saturation, it is the opinion of this author that the illustration can be particularly misleading for the following reasons:

- **Weakly coupled dynamics**

Consider, for example, the extreme case where two totally unrelated SISO control systems are implemented as a MIMO controller as depicted in Figure 4.2.

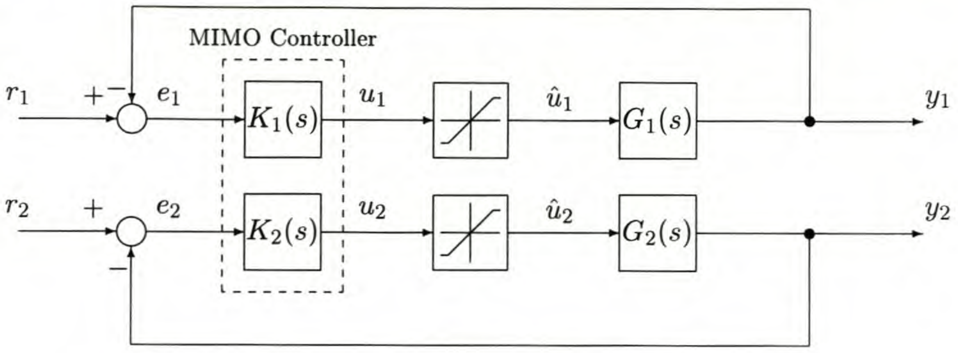

Figure 4.2: MIMO implementation of two SISO controllers.

Assume that the controller $K_1(s)$ has issued a saturated control signal (i.e. $|u_1| > |\hat{u}_1|$) and that $K_2(s)$ is unsaturated (i.e. $u_2 = \hat{u}_2$). Following Kapasouris' approach both error signals e_1 and e_2 would be equally scaled by the error governor such that controller $K_1(s)$ does not saturate (i.e. $u_1 = \hat{u}_1$). The net result is that the control vector $\mathbf{u} = u_1 + u_2$ does not saturate and its direction remains unaltered. Nevertheless, the performance of controller $K_2(s)$ has been compromised by a saturation in controller $K_2(s)$.

The purpose of this example is evident. Control $loop_1$ and $loop_2$ are totally unrelated to each other. The effect of a saturation in one control loop should have no performance bearing on the other control loop whatsoever. Control signal u_2 is not a function of control signal u_1, plant $G_1(s)$, controller $K_1(s)$ or the error signal e_1 in any way and vice-versa. Therefore control vector direction is irrelevant in this example.

It may be argued that it is unlikely to ever implement two SISO controllers as a MIMO structure in practice and that this example is extreme. Consider then a controller in which the cross coupling between channels is very weak or only unidirectional as in a controller with triangular structure. Clearly it would be extremely conservative to scale both controls that are weakly related or have a unidirectional relationship simply because one of them has saturated.

- **Direct feed through**

  As another counter illustration, consider a MIMO control system with cross coupled dynamics. In the state space form the control signal is given by:

$$\begin{aligned}\mathbf{u}(k) &= \mathbf{Cx}(k) + \mathbf{De}(k) \\ &= \mathbf{Cx}(k) + \mathbf{D}(\mathbf{r}(k) - \mathbf{y}(k))\end{aligned}$$

  For a non-zero D matrix (i.e. a controller with proportional control) the magnitude and polarity of each component of the control vector can be directly manipulated from the reference input $\mathbf{r}(k)$. Therefore the direction of the control vector is arbitrary since the reference input, $\mathbf{r}(k)$ which is set by the operator, can be arbitrary and different at each instant k.

What then is the real issue? — The real issue is that the direction of the control vector must accurately reflect or map to the internal state of the controller. That is, the control vector must be consistent (linearly related) to the states of the controller and the states of the controller must be such that saturation does not occur. It is, however, improper to force the control vector direction to be maintained since this may imply forcing an *unnatural* relationship between *unrelated* states and error inputs in the controller.

4.4 Controller State Propagation

It is proposed that one cannot begin to treat saturation in a MIMO control system only once it has occurred. Rather one must prevent saturation from ever occurring in the first place. In this section the time propagation of the controller states are considered, in the light of the above proposition.

Given the discrete state space description of the linear time-invariant controller:

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{\Phi x}(k) + \mathbf{\Gamma e}(k)\\ \mathbf{u}(k) &= \mathbf{Cx}(k) + \mathbf{De}(k)\end{aligned}$$

The future time response of the propagated controller states for h time steps are:

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{\Phi x}(k) + \mathbf{\Gamma e}(k)\\ \mathbf{x}(k+2) &= \mathbf{\Phi x}(k+1) + \mathbf{\Gamma e}(k+1)\\ &\vdots\\ \mathbf{x}(k+h+1) &= \mathbf{\Phi x}(k+h) + \mathbf{\Gamma e}(k+h)\end{aligned}$$

The associated controller outputs are:

$$\begin{aligned}\mathbf{u}(k) &= \mathbf{Cx}(k) + \mathbf{De}(k)\\ \mathbf{u}(k+1) &= \mathbf{Cx}(k+1) + \mathbf{De}(k+1)\\ &\vdots\\ \mathbf{u}(k+h) &= \mathbf{Cx}(k+h) + \mathbf{De}(k+h)\end{aligned}$$

The response of the controller at the present time $\mathbf{u}(k)$ and future time $\mathbf{u}(k+h)$ due only to the present state vector $\mathbf{x}(k)$ (free response) and present input error vector $\mathbf{e}(k)$ (forced response) can be determined by setting:

$$\mathbf{e}(k) = \mathbf{r}(k) - \mathbf{y}(k)$$

and making the error vector zero for all time steps thereafter, i.e.

$$\mathbf{e}(k+1) = \mathbf{e}(k+2) = \cdots = \mathbf{e}(k+h) = \mathbf{0}$$

The error vector may be set to exactly zero since this is a signal internal to the controller's workings. Although the measurement cannot be changed, the applied reference could be made equal and opposite to the measurement thus forcing the error to zero. The reason for making the error zero for all future time steps is to investigate the time response that the free response of the controller due to its current state combined with the present forced response due to the present error would be. It should be borne in mind that the controller is an exact entity. It is not some approximation or model of reality as is the case with the plant model. Thus it is possible to make exact predictions or extrapolations of future free response behaviour given the present state and present forcing function.

Thus the propagated controller's state can be written as:

$$\begin{aligned}\mathbf{x}(k+1) &= \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}\mathbf{e}(k)\\ \mathbf{x}(k+2) &= \mathbf{\Phi}\mathbf{x}(k+1)\\ &\vdots\\ \mathbf{x}(k+h+1) &= \mathbf{\Phi}\mathbf{x}(k+h)\end{aligned}$$

and the propagated controller's output as:

$$\begin{aligned}\mathbf{u}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{e}(k)\\ \mathbf{u}(k+1) &= \mathbf{C}\mathbf{x}(k+1)\\ &\vdots\\ \mathbf{u}(k+h) &= \mathbf{C}\mathbf{x}(k+h)\end{aligned}$$

With suitable substitutions, the recursive state equation and the output equation after h time steps can be written in terms of the present state and present error (assuming all future errors were zero) as follows:

$$\mathbf{x}(k+h+1) = \mathbf{\Phi}^{h+1}\mathbf{x}(k) + \mathbf{\Gamma}^{h+1}\mathbf{e}(k) \quad : h \geq 0$$

and

$$\mathbf{u}(k+h) = \left\{ \begin{array}{ll} \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{e}(k) & : h = 0 \\ \mathbf{C}[\mathbf{\Phi}^{h}\mathbf{x}(k) + \mathbf{\Phi}^{h-1}\mathbf{\Gamma}\mathbf{e}(k)] & : h > 0 \end{array} \right\}$$

Therefore, given the state $\mathbf{x}(k)$ of the controller and the error $\mathbf{e}(k)$ at the time step k, one can predict the present forced and future free response of the controller for any future time h steps ahead.

Assume now that for the present state $\mathbf{x}(k)$ and error $\mathbf{e}(k)$ that at some time h steps in the future the controller issues a saturating control (i.e. $|\mathbf{u}(k+h)| \geq |\hat{\mathbf{u}}|$). To try to establish the path or sequence of events (or error signals) that caused the saturation once the saturation has occurred is practically impossible. Even if it were practically possible to back propagate the saturating control signal to the instant when the responsible error signal (forcing function) was applied to the controller, this would be of no value since it is not possible to go back into the time history of the controller and of the plant and *undo* the past. The *damage*, as it were, has already been done. That is, the controller has attained a state that will lead to saturation even if the error input was made zero. The free response of the controller, due to its state alone, will in time saturate and windup. The sequence of control signals that eventually led up to the moment of saturation have already been applied to the input of the plant and the plant has reacted to this input and this history cannot be retrieved from the plant and re-applied in a modified and more desirable manner with the recently acquired hindsight.

In an interacting MIMO controller the state transition matrix, $\mathbf{\Phi}$, will have off-diagonal entries and therefore the higher powers of $\mathbf{\Phi}$ will be full matrices. Therefore the mix of the original state $\mathbf{x}(k)$ (free response) as well as the distribution of the original error $\mathbf{e}(k)$ (forced response) in the future response $\mathbf{u}(k+h)$ of the controller becomes very complex to unravel and even more so if the error vectors $\mathbf{e}(k+1), \mathbf{e}(k+2), \cdots, \mathbf{e}(k+h)$ had been non-zero. Likewise, if the controller had been time varying (for example, due to gain scheduling) the problem becomes even more complex as one would need to keep a history of the gains of the controller at each time step so that the state propagation of the controller could be consistently reversed or back propagated. The possibility exists that for the same state the controller could issue either saturated or unsaturated control signals depending upon the particular value of the controller's gain at that instant.

It has thus been established that one cannot start to address the saturation of the controller once the saturation has occurred because the free response, due to the dynamic behaviour of the controller, may lead to deeper saturation even after the error signal driving (forcing) the controller has been removed. The only way to avoid saturation is to never allow the controller to attain a state that will eventually lead to saturation. The bottom line, to use an old cliche, is that *prevention is better than cure.* In the MIMO case, prevention may well be the only viable option.

Therefore, in summary, one must never apply an error signal to the controller that will result in saturation now nor at any time step in the future response of the controller, thus preventing saturation altogether rather than trying to remedy it only once it has occurred.

4.5 Methodology behind the Proposed Algorithm

In the proposed anti-saturation algorithm, the underlying methodology is to condition the error input of the controller such that saturation can never occur, not now nor at any time in the future free or forced response of the controller. The following two definitions regarding the controller's state are made:

- A controller state that leads to the controller issuing a saturated control for the free and unforced response of the controller is defined to be an *inadmissible state.*
- A controller state that does not lead to the controller issuing a saturated control for the free and unforced response of the controller is defined to be an *admissible state.*

Let the applied error input, $\hat{\mathbf{e}}(k)$, into an m-input controller be defined by:

$$\hat{\mathbf{e}}(k) = \mathbf{Q}(k)\mathbf{e}(k)$$

or

$$\begin{bmatrix} \hat{e}_1(k) \\ \hat{e}_2(k) \\ \vdots \\ \hat{e}_m(k) \end{bmatrix} = \begin{bmatrix} q_1(k) & 0 & \cdots & 0 \\ 0 & q_2(k) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_m(k) \end{bmatrix} \begin{bmatrix} e_1(k) \\ e_2(k) \\ \vdots \\ e_m(k) \end{bmatrix}$$

where the diagonal error scaling matrix $\mathbf{Q}(k)$ is constrained by $0 \leq ||\mathbf{Q}(k)||_2 \leq 1$ and $\mathbf{e}(k)$ is the actual error between the reference set point and the plant output as shown in Figure 4.3.

In other words the problem then reduces to determining the maximum error vector $\hat{\mathbf{e}}(k)$ that can be applied to the controller, taking into account the controllers current state $\mathbf{x}(k)$ and hence free response, such that the controller never saturates now nor at any future propagation of the controller output.

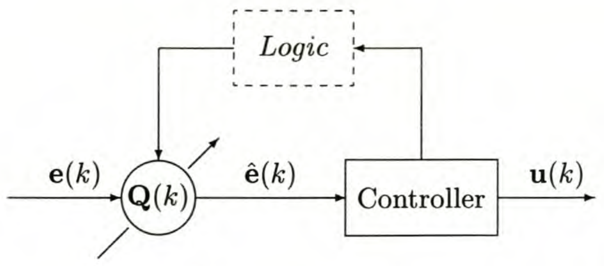

Figure 4.3: MIMO controller with diagonalised error governor, $\mathbf{Q}$, on error input.

Assuming that the current controller state $\mathbf{x}(k)$ is an admissible state and that one can optimally determine the matrix $\mathbf{Q}(k)$ for all time steps in the future, one can ensure that no external error input $\mathbf{e}(k)$ can ever make the controller acquire a state that is inadmissible. Therefore saturation can always be avoided by regulating the external energy into the controller.

Note that by having individual operators, $q_i(k)$, each acting on an individual component of the error vector, $\mathbf{e}(k)$, it means that each component, $e_i(k)$, of the error vector can be individually scaled. This deviates *fundamentally* from the ideas of Kapasouris who scales the entire error vector by the same scalar, $\lambda(k)$, (see Figure 3.8) in order to hold to the premise that the direction control vector must be maintained during saturation. Therefore the newly proposed algorithm forsakes the linearity dependance requirement of the error vector components that was enforced by Kapasouris. Nevertheless the proposed algorithm does not allow the controller to saturate. Furthermore, the proposed algorithm does not suffer from the conservatism of Kapasouris's algorithm in that weakly coupled or uncoupled controllers will now be able to perform optimally in all channels up to the saturation limit of each channel.

4.6 Anti-saturation Algorithm

The anti-saturation algorithm for an m-input, n-state, p-output MIMO controller can be stated as:

Find the maximum individual error, e_i, scale factors, q_i, such that $0 \leq q_i \leq 1$ and such that all the future propagated outputs of the controller due to the present controller state and this scaled error lie between the actuator saturation bounds. This can be stated mathematically as follows:

$$\begin{array}{l}
\text{maximise } z = q_1 + q_2 + \cdots + q_m \\
\text{subject to} \\
\qquad \begin{array}{ccccc}
\mathbf{u}_{min} & \leq & \mathbf{u}(k) & \leq & \mathbf{u}_{max} \\
\mathbf{u}_{min} & \leq & \mathbf{u}(k+1) & \leq & \mathbf{u}_{max} \\
 & & \vdots & & \\
\mathbf{u}_{min} & \leq & \mathbf{u}(k+h) & \leq & \mathbf{u}_{max} \\
 & & & & \\
0 & \leq & q_1 & \leq & 1 \\
0 & \leq & q_2 & \leq & 1 \\
 & & \vdots & & 1 \\
0 & \leq & q_m & \leq & 1
\end{array}
\end{array}$$

where z is the maximised scalar sum of the individual error, e_i, scale factors, q_i, and the saturation limit vectors and controller's output vector are given by:

$$\mathbf{u}_{min} = \begin{bmatrix} u_{1_{min}} \\ u_{2_{min}} \\ \vdots \\ u_{p_{min}} \end{bmatrix} \qquad \mathbf{u}(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_p(k) \end{bmatrix} \qquad \mathbf{u}_{max} = \begin{bmatrix} u_{1_{max}} \\ u_{2_{max}} \\ \vdots \\ u_{p_{max}} \end{bmatrix}$$

with the propagated controller's output being given by:

$$\begin{aligned} \mathbf{u}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{Q}(k)\mathbf{e}(k) \\ \mathbf{u}(k+1) &= \mathbf{C}[\mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}\mathbf{Q}(k)\mathbf{e}(k)] \\ &\vdots \\ \mathbf{u}(k+h) &= \mathbf{C}[\mathbf{\Phi}^h\mathbf{x}(k) + \mathbf{\Phi}^{h-1}\mathbf{\Gamma}\mathbf{Q}(k)\mathbf{e}(k)] \end{aligned}$$

where h is the number is steps (optimisation horizon) it takes for the controller to settle to steady state, with $h \geq 1$ for a controller with dynamics. For a controller with no dynamics, $(h = 0)$, steady state is reached immediately and windup prevention and therefore saturation prevention is not a concern.

Once the optimal error governor matrix, $\mathbf{Q}(k)$, has been found the controller is implemented by:

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}\hat{\mathbf{e}}(k) \\ \mathbf{u}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\hat{\mathbf{e}}(k) \end{aligned}$$

or

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}\mathbf{Q}(k)\mathbf{e}(k) \\ \mathbf{u}(k) &= \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{Q}(k)\mathbf{e}(k) \end{aligned}$$

Note that the optimisation problem must be solved afresh for every discrete time step k.

In essence the proposed anti-saturation algorithm for MIMO controllers, within the assumptions and constraints posed, has been formulated. In practice, however, certain difficulties still need to be overcome, notably:

- A suitable numerical optimisation algorithm must be identified to determine the error governor $\mathbf{Q}(k)$.
- The minimum (optimal) horizon or number of look-ahead steps h needs to be determined.
- Calculation of Φ^h may be computationally intensive for controllers with many states.

4.7 Summary

- The requirement for maintaining control vector direction was dispensed with.
- An Error Governor was introduced into the controller loop to prescale the individual error vector components such that the controller never attains an inadmissible state and therefore never saturates.
- The anti-saturation problem was cast as a numerical optimisation problem with a linear cost function subject to inequality constraints.

Chapter 5

Linear Programming

5.1 Introduction

The anti-saturation algorithm for MIMO controllers developed in Chapter 4 was cast as a bounded or constrained optimisation problem. Although by design intentionally so, it can be verified by inspection that the optimisation problem is linear in both the objective function as well as the constraints, both being linear combinations of the optimisation variables.

While sophisticated complex nonlinear algorithms or quadratic optimisation algorithms (of which many exist as routines in numerical software toolboxes) may be employed to solve this optimisation problem, the definite linearity in the problem should rightly be exploited to the fullest advantage in simplifying the solution. Ultimately this algorithm is intended for practical application. Thus, despite the ever increasing computational capacities of embedded micro controllers and computers used in control applications, the optimisation process must itself be as optimal as possible so that the process can be completed and repeated every update cycle of the discrete controller. Typical update cycles used for aircraft autopilots are 20 ms (50 Hz) and for missiles this may be an order of magnitude faster, say 2 ms (500 Hz).

Linear optimisation techniques can be broadly classified as those:

- Methods that systematically cross the interior region of the solution space following steepest gradients in search of the optimum.
- Methods that examine the boundary of the region of the solution space for the optimum.

In a linear optimisation problem, the linear cost function to be minimised or maximised is called the objective function. There are usually infinitely many solutions to the system of linear constraints, which may be equalities or inequalities. These solutions are called feasible solutions or feasible points. The aim is to find one such solution that is an optimum solution, that is, one that gives the minimum or maximum value of the objective function.

Following the arguments presented in most texts on *linear programming* [1] one can conclude from the underlying theoretical aspects of the geometry of the problem that in order to determine the optimal solution of a constrained linear optimisation problem one need only consider

[1]Linear programming is a well established and popular technique for solving linear optimisation problems. The linear optimisation problem is generally referred to as a *linear program*.

a finite number of candidate solution points. Furthermore, it turns out that these points all exist on the boundary of the region of all (infinite) feasible points [Ignizio and Cavalier]. Nevertheless, to actually identify these points in a linear program of nontrivial size is not so straightforward. It is necessary to examine the linear program from an algebraic viewpoint. This algebraic viewpoint relies heavily on linear algebra, and, in particular, on the concept of a basic solution to a linear system. Linear programming, developed by George B. Danzig in the late 1940's, is the topic of a great many authors and these references should be consulted for a detailed exposition on the subject [2]. In this thesis only the relevant subset of the available linear programming theory has been extracted and applied to specify a mechanical procedure that can be used to solve the MIMO anti-saturation algorithm optimisation problem in a practical implementable way.

It is necessary to examine linear programming, relative definitions and terminology in order to set the stage for customising the application of linear programming to the MIMO anti-saturation algorithm. Most texts on linear programming treat the subject systematically, beginning with the concepts, moving on to the theory behind the algorithm, various methods of solution, examples and so on. In the process, one has to study all the material sequentially to grasp the full picture. In this section a summary is presented to put the various aspects into context and to provide an immediate reference framework for the reader. It is not the intention of this thesis to present a full treatment of linear programming.

5.2 General Form

In a linear program an objective function must be optimised (minimised or maximised) subject to a list of constraints that are placed on the values that the variables in the objective function may assume.

The general form of the (single-objective) linear programming model can be stated mathematically as:

Find $\mathbf{q} = [q_1, q_2, \dots, q_m]^T$ so as to optimise, the objective function, m-variables, subject to the specified l-constraints.

$$\begin{array}{l}
\text{maximise } z = c_1q_1 + c_2q_2 + \cdots + c_mq_m \\
\text{subject to} \\
\quad \begin{array}{ccccccccc}
a_{1,1}q_1 & + & a_{1,2}q_2 & + & \cdots & + & a_{1,m}q_m & \{\leq,=,\geq\} & b_1 \\
a_{2,1}q_1 & + & a_{2,2}q_2 & + & \cdots & + & a_{2,m}q_m & \{\leq,=,\geq\} & b_2 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
a_{l,1}q_1 & + & a_{l,2}q_2 & + & \cdots & + & a_{l,m}q_m & \{\leq,=,\geq\} & b_l
\end{array} \\
\text{and } q_1, q_2, \dots, q_m \geq 0
\end{array}$$

where c_i are constant coefficients assigning relative weighting to the decision variables, q_i, in the cost function, z, and the $a_{i,j}$ are the constant coefficients of the decision variables in the constraints. The b_i are the constants, independent of the decision variables, in the constraints.

Each constraint may be either a $(\leq)$ inequality, a $(\geq)$ inequality or an equality $(=)$. Also in some instances the nonnegativity restrictions on the decision variables may not be appropriate.

[2]The book **Linear Programming** by James P. Ignizio and Tom M. Cavalier is worth particular consideration and was used extensively during the development of this thesis.

It is always possible, through the use of certain linear transformations, to convert any linear programming model into the foregoing form [Ignizio and Cavalier:18,19].

5.3 Assumptions of the linear programming model

There are four basic assumptions that ensure that a real situation can be represented as a linear programming problem [Ignizio and Cavalier:19].

- **Certainty**

  The problem data (c_j,b_i and $a_{i,j}; i = 1, \ldots, l; j = 1, \ldots, m$) are assumed to be known with certainty. That is, there is no stochastic element to the data.

- **Proportionality**

  The contribution of a decision variable, q_j, to the objective function is $c_j q_j$ and its contribution to the ith constraint is $a_{i,j} q_j$. That is, the contribution of q_j is always directly proportional to the level of the variable q_j. This simply means, for example, that if the value of q_j doubles, then its contribution to the objective function also doubles. There are no setup costs, discounts, or economies of scale.

- **Additivity**

  This assumption ensures, for example, that the total cost is the sum of the cost contributions of each individual variable. That is, the contributions from individual variables combine linearly in both the objective function and the constraints. There are no interactions that could reduce or increase the level of the combined contributions.

- **Divisibility**

  Divisibility implies that the decision variables are continuous variables, that is, they can be divided into fractional parts.

Despite what appears to be somewhat restrictive assumptions, linear programming remains one of the most widely used modelling techniques [Ignizio and Cavalier:19].

5.4 Canonical Form

In the canonical form, in general, the constraints are written as inequalities. Typically either all as ($\leq$) or ($\geq$) inequalities depending upon the problem. The entire constraint may be multiplied by -1 and the direction of the inequality reversed to get it into this form if necessary. If an equality constraint was specified by the optimisation problem, then equalities may exist in the constraints of the canonical form as well. In the case of the anti-saturation algorithm developed in this thesis all the constraints are inequalities, there are no equality constraints.

5.5 Standard Form

Since it is easier to work with equations rather than inequalities, the constraints are converted into equations by the addition of surplus and/or slack variables as necessary. Therefore in the standard form there are only equalities, no inequalities. In the case of the anti-saturation algorithm developed in this thesis there are no surplus variables, only slack variables. The presence of a surplus variable would indicated that one of the saturation bounds have been violated.

5.6 Constraint Conversion

Any inequality can be converted into an equation by the introduction of slack or surplus variables. For simplicity it is assumed that regardless of the form of the inequality, the right-hand side is nonnegative (that is, $b_i \geq 0$). Thus if the constraint initially has a negative b_i, then multiply the entire constraint by -1 and reverse the direction of the inequality.

5.6.1 Conversion of a ($\leq$) inequality

Consider an inequality, say constraint r, of the following form:

$$\sum_{j=1}^{m} a_{r,j} q_j \leq b_r$$

Introduce a new variable, $s_r \geq 0$, called the *slack* variable, so that:

$$\sum_{j=1}^{m} a_{r,j} q_j + s_r = b_r$$

That is:

$$s_r = b_r - \sum_{j=1}^{m} a_{r,j} q_j$$

In words, s_r is the (nonnegative) difference between the right-hand side constant and the original left-hand side and, thus, it *takes up the slack*. Physically, it often represents the amount of resource r (that is b_r) that is unused or idle. From a mathematical view, it allows one to express an inequality in a more convenient format.

5.6.2 Conversion of a ($\geq$) inequality

Consider an inequality, say constraint t, of the form:

$$\sum_{j=1}^{m} a_{t,j} q_j \geq b_t$$

In this case, introduce a new variable, $s_t \geq 0$, so that:

$$\sum_{j=1}^{m} a_{r,j} q_j = b_t + s_t$$

Thus:

$$s_t = \sum_{j=1}^{m} a_{r,j} q_j - b_t$$

That is, s_t represents the (nonnegative) difference between the left-hand side and right-hand side of the inequality. Such a variable is termed a *surplus* variable. Physically, the surplus variable represents the amount by which the right-hand side is exceeded.

5.7 The Objective Function

The choice of decision variables (that is, the $q_{j's}$) directly affects the value of the objective function. This holds true as well for the slack and surplus variables. As a result, each variable introduced in the constraint conversion process should also be introduced, with a proper coefficient, into the objective function. Consider the standard form of the objective function to be *maximization*. This in no way eliminates the consideration of *minimization* type objectives, because if a function z is to be minimized one can use the simple equivalence:

$$\text{minimize } z \quad \equiv \quad -\,\text{maximize } (-z)$$

Thus, given a maximization objective z, and p surplus and slack variables, the modified objective is

$$\text{maximise } z = \sum_{j=1}^{l} c_j q_j + \sum_{k=1}^{p} c_k s_k$$

The first term, is simply the original objective function, and the second term corresponds to the impact of the slack and surplus variables.

5.8 Companion Form

Associated with every linear program there is a companion form called the *dual*. The original form of the linear program then being referred to as the *primal*. Defining the problem at hand as the primal and the companion form as the dual is arbitrary and interchangeable. Both forms are linear programs, defining either one to be primal implies the other form is its dual. In this thesis and in keeping with the conventions used in other texts, the original form of the maximisation problem will be defined to be the primal form. This primal-dual relationship is far more than a curious relationship [Ignizio and Cavalier:167]. In linear programming, duality is used in a wide variety of both theoretical and practical ways. Included among these are the following:

1. In some cases, it may be easier (less iterations, etc.) to solve the dual than the primal.
2. The dual variables provide important economic interpretations of the results obtained when solving a linear programming problem.
3. Duality is used as an aid when investigating changes in the coefficients or formulation of a given linear programming problem (i.e. in *sensitivity analysis*).
4. Duality will be utilised to allow one to employ the simplex method to solve problems in which the initial basis is *infeasible* (the technique itself is known as the *dual simplex*).
5. Duality is used to develop a number of important theoretical results in linear programming.

Points (1) and (4) are exploited in this thesis.

Formulation of the dual problem is actually a mechanical, straightforward process. However the mechanics differ somewhat according to the form of the primal. Refer to Ignizio and Cavalier:168-188 for a comprehensive treatment. In the context of this thesis, the dual can be viewed as the *transpose* of the primal.

During the execution of the simplex algorithm one not only derives the optimal solution for a given linear system, but one also solves a companion problem called the *dual* problem. In this context, the original problem is generally referred to as the *primal* problem.

The following rules [Ignizio and Cavalier:169] are followed when specifying the dual from the primal:

- The objective of the primal is to be maximised; the objective of the dual is to be minimised.
- The maximisation problem must have all ($\leq$) constraints and the minimisation problem has all ($\geq$) constraints.
- All primal and dual variables must be nonnegative.
- Each *constraint* in one problem corresponds to a *variable* in the other problem. For example, given m primal constraints, there are m dual variables, and, given n primal variables there are n dual constraints. Consequently, if one problem is of order $m \times n$, the other is of order $n \times m$.
- The elements of the right-hand side of the constraints in the one problem are the respective coefficients of the objective function in the other problem.
- The matrix of constant coefficients for the one problem is the transpose of the matrix of coefficients for the other problem.

5.9 Methods of Solution

Various methods have been devised to solve linear programming problems when presented in the standard form. The best known of these methods is the *simplex method*, which is a mechanical process that systematically steps from one feasible solution of the linear program to the next until the optimum solution (if it exists [3]) has been reached. The *revised simplex method* is a computationally more efficient version of the simplex method. Another method, of which little mention is made in most texts, is the *dual simplex method* (often just called the *dual method*).

Note that the *dual method* should not be confused with the *dual form*. That is:

- The one is a *method* or algorithm used for solving linear programming problems and the other is a *form* or way of stating a linear programming problem. The use of the word *dual* in both cases is perhaps a little unfortunate.
- The simplex method can be used to solve linear programming problems written in both the primal and the dual form.
- Likewise the dual method can be used to solve linear programming problems written in both the primal and the dual form.

5.10 Problem Transformation Options

Figure 5.1 shows graphically the options available in linear programming. In words, the linear problem in canonical form can be transformed between the primal canonical form and dual canonical form. Likewise, the linear problem in standard form can be transformed between the primal standard form and the dual standard form. All the solution methods (either simplex or dual method) operate on the linear program when in the standard form. Thus the canonical form (either primal or dual) must first be converted to the standard form (either primal or dual).

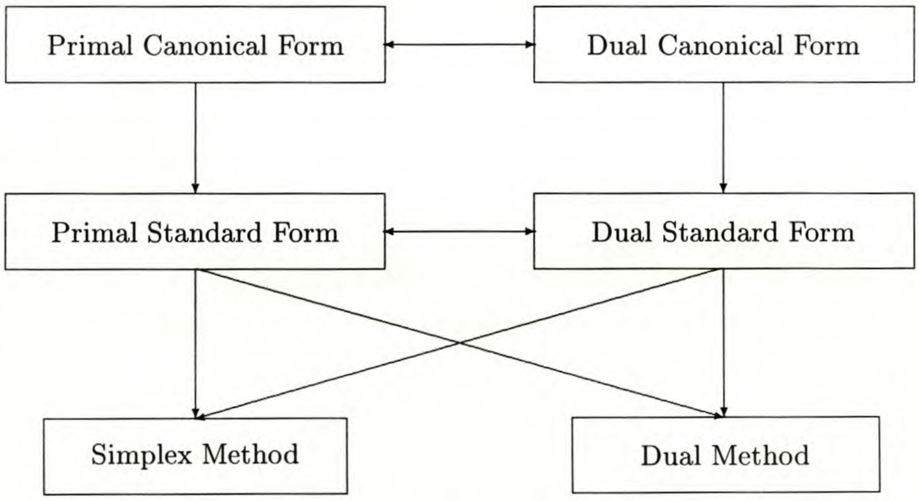

Figure 5.1: Linear Programming Overview

Assuming one always begins with the primal canonical form, there are four routes one could take to the standard form, each eventually being solved by applying either the simplex method or the dual method.

[3]For an unbounded solution space, an optimum cannot be found.

1. Primal Canonical Form
   $\rightarrow$ Primal Standard Form
   $\rightarrow$ Solve

2. Primal Canonical Form
   $\rightarrow$ Primal Standard Form
   $\rightarrow$ Dual Standard Form
   $\rightarrow$ Solve

3. Primal Canonical Form
   $\rightarrow$ Dual Canonical Form
   $\rightarrow$ Dual Standard Form
   $\rightarrow$ Solve

4. Primal Canonical Form
   $\rightarrow$ Dual Canonical Form
   $\rightarrow$ Dual Standard Form
   $\rightarrow$ Primal Standard Form
   $\rightarrow$ Solve

It is important to note that although (1) and (4) are both in primal standard form, because of the different routes taken to reach this form, the two linear programs are not equal in dimension. Likewise, although (2) and (3) are both in dual standard form, their dimensions are very different. Therefore, while the forms are equivalent and the solutions the same, the actual linear programs are very different in dimension. The dimensions of these seemingly equivalent problems are different because the slack and/or surplus variables are introduced at different stages of the transformation. The various routes to standard form from canonical form and different methods of solution have advantages and disadvantages relative to one another.

Assume that all l constraints in canonical form were inequalities. Then one would require l slack or surplus variables to convert all the inequalities into equalities to get the standard form. The derivation of the resulting dimensions of the various routes can be found in Appendix A. The table below summarises these results:

Route number	Number of equalities	Number of variables	Standard form
1	l	$m+l$	Primal
2	$m+l$	l	Dual
3	m	$l+m$	Dual
4	$l+m$	m	Primal

In the proposed MIMO anti-saturation algorithm optimisation problem, there will typically be few variables m (only as many as the number of inputs to the controller), while the number of constraints l depends upon the update period of the controller and the time it takes for the dynamics of the controller to reach steady state.

It is thus typical to have hundreds or even thousands of constraints. The number of constraints l is therefore orders of magnitude larger than the number of variables m to be solved for. Since the number of pivot operations (i.e. iterations) that the solution method, either simplex or dual, takes to find the optimum is proportional to the number of constraints, one would prefer to solve the problem that has fewest constraints.

Thus in the context of the MIMO anti-saturation algorithm where $m << l$ it makes computational economic sense to adopt Route (3) since this results in the fewest iterations in determining the optimum. One proviso is that an initial basic feasible solution *must* exist as a starting point for the solution method (simplex or dual) to begin the search. If no initial basic feasible solution exists, then procedures exist for creating this basis, but the overheads of implementing these procedures might not justify the use of this particular form.

5.11 The Simplex Method

If a finite optimal solution to a linear programming problem exists, then an extreme-point optimal solution exists [Ignizio and Cavalier:80]. Thus, in the search for an optimal solution, one need only consider the extreme points of the feasible region. This is a very important fact because the solution space is reduced to a finite number of candidate points. Algebraically, these extreme points correspond to basic feasible solutions of the linear constraint set:

$$\mathbf{Ax} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0}$$

The simplex method is a systematic procedure for iteratively moving from one extreme point to an adjacent extreme point in the search for an optimal solution. Tabular formats have been established to simplify the *bookkeeping* aspects of the simplex method.

To start the simplex algorithm one must identify an initial basic feasible solution. In some cases this is straight forward, but in some cases it is not always available. When the initial basis is not available, other methods can be employed to artificially generate such a basis in order to begin the simplex algorithm. Such methods are the *Two Phase Method* and the *Big-M Method*, the details of which can be found in Ignizio and Cavalier:103-112. The general approach of the two phase method is to create an identity submatrix by adding the necessary artificial variables to the original constraints and proceeding from there.

For the purposes of the anti-saturation algorithm it would be preferable if an initial basic solution always existed so that the simplex algorithm could be started without any need for artificial variables. The reason for this is that the anti-saturation algorithm needs to be deterministic and guaranteed to work for it to be practically applicable in real-time in real-world control problems. So a formulation of the anti-saturation linear programming problem that always has an initial basic solution is sought.

5.12 The Dual Method

The Dual Simplex Algorithm was initially developed by Lemke (1954) [Ignizio and Cavalier:197]. This text should be consulted for a full treatment, but in summary, the dual simplex method operates on the linear program in much the same way as the primal simplex method, except that it operates on the dual space rather than on the primal space. Such an approach is only possible if the current solution is *dual feasible*. That is, primal optimality conditions are satisfied. If the initial solution is not dual feasible, then artificial variable techniques exist to create such a solution.

- Thus, the dual simplex method maintains dual feasibility and complementary slackness throughout its operation, while trying to achieve primal feasibility.
- The primal simplex method maintains primal optimality and complementary slackness throughout, and tries to restore dual feasibility.

A sufficient and necessary condition for the dual feasible solution to exist is that the coefficients of the objective function (in minimisation form) must be all nonnegative.

Chapter 6

Implementation

6.1 Introduction

It is necessary to examine the characteristics of the MIMO anti-saturation algorithm to gain the necessary insight into what considerations must be addressed when applying linear programming to this optimisation problem.

6.2 Problem Transformation

The anti-saturation algorithm optimisation problem developed in Chapter 4 is repeated here for convenience. That is:

$$\begin{array}{l} \text{maximise } z = q_1 + q_2 + \cdots + q_m \\ \text{subject to} \\ \qquad \begin{array}{ccccc} \mathbf{u}_{min} & \leq & \mathbf{u}(k) & \leq & \mathbf{u}_{max} \\ \mathbf{u}_{min} & \leq & \mathbf{u}(k+1) & \leq & \mathbf{u}_{max} \\ & & \vdots & & \\ \mathbf{u}_{min} & \leq & \mathbf{u}(k+h) & \leq & \mathbf{u}_{max} \end{array} \end{array} \tag{6.1}$$

where the control vector values and limits are made up of the individual control values and their limits as follows:

$$\mathbf{u}_{min} = \begin{bmatrix} u_{1_{min}} \\ u_{2_{min}} \\ \vdots \\ u_{p_{min}} \end{bmatrix} \qquad \mathbf{u}(k) = \begin{bmatrix} u_1(k) \\ u_2(k) \\ \vdots \\ u_p(k) \end{bmatrix} \qquad \mathbf{u}_{max} = \begin{bmatrix} u_{1_{max}} \\ u_{2_{max}} \\ \vdots \\ u_{p_{max}} \end{bmatrix}$$

with:

$$\begin{array}{rcl} \mathbf{u}(k) & = & \mathbf{C}\mathbf{x}(k) + \mathbf{D}\mathbf{Q}(k)\mathbf{e}(k) \\ \mathbf{u}(k+1) & = & \mathbf{C}[\mathbf{\Phi}\mathbf{x}(k) + \mathbf{\Gamma}\mathbf{Q}(k)\mathbf{e}(k)] \\ & \vdots & \\ \mathbf{u}(k+h) & = & \mathbf{C}[\mathbf{\Phi}^h\mathbf{x}(k) + \mathbf{\Phi}^{h-1}\mathbf{\Gamma}\mathbf{Q}(k)\mathbf{e}(k)] \end{array}$$

and:

$$\mathbf{Q} = \begin{bmatrix} q_1 & 0 & \cdots & 0 \\ 0 & q_2 & \cdots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \cdots & q_m \end{bmatrix} \qquad \text{with } 0 \leq ||\mathbf{Q}||_2 \leq 1$$

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix}$$

where h is the number is steps or look ahead horizon it takes for the controller to settle to steady state.

Objective function

By inspection the objective function is already in linear programming primal canonical form.

$$\text{maximise } z = q_1 + q_2 + \cdots + q_m = \mathbf{cq}$$

where $\mathbf{c}$ is a unity $1 \times m$ vector of the coefficients of the optimisation variables $\mathbf{q}$ of the objective function. These coefficients are all unity since the relative weighting between the optimisation variables is equal. That is, no one error input into the controller is more important than any other.

The primal canonical form requires that $\mathbf{c} \geq \mathbf{0}$, which is the case in the proposed anti-saturation algorithm.

Constraints

The constraints require transformation to get them into the desired linear programming primal canonical form. One can expand each of the control constraints with suitable substitutions to give:

$$\begin{bmatrix} \mathbf{u}_{min} & \leq & \mathbf{Cx}(k) + \mathbf{DQ}(k)\mathbf{e}(k) & \leq & \mathbf{u}_{max} \\ \mathbf{u}_{min} & \leq & \mathbf{C}[\mathbf{\Phi x}(k) + \mathbf{\Gamma Q}(k)\mathbf{e}(k)] & \leq & \mathbf{u}_{max} \\ & & \vdots & & \\ \mathbf{u}_{min} & \leq & \mathbf{C}[\mathbf{\Phi}^h\mathbf{x}(k) + \mathbf{\Phi}^{h-1}\mathbf{\Gamma Q}(k)\mathbf{e}(k)] & \leq & \mathbf{u}_{max} \end{bmatrix} \tag{6.2}$$

Since it is necessary to resolve this optimisation problem afresh every time step of the discrete controllers update period, one can consider the k^{th} time step to always be the present time step. Thus the suffix (k) will be dropped at this stage of the development in favour of conciseness. Each constraint of (6.2) is a double inequality. These double inequalities can be rewritten as single inequalities as follows:

$$\left[\begin{array}{rcl}
\mathbf{Cx} + \mathbf{DQe} & \leq & \mathbf{u}_{max} \\
\mathbf{C\Phi x} + \mathbf{C\Gamma Qe} & \leq & \mathbf{u}_{max} \\
 & \vdots & \\
\mathbf{C\Phi}^h\mathbf{x} + \mathbf{C\Phi}^{h-1}\mathbf{\Gamma Qe} & \leq & \mathbf{u}_{max} \\
\mathbf{Cx} + \mathbf{DQe} & \geq & \mathbf{u}_{min} \\
\mathbf{C\Phi x} + \mathbf{C\Gamma Qe} & \geq & \mathbf{u}_{min} \\
 & \vdots & \\
\mathbf{C\Phi}^h\mathbf{x} + \mathbf{C\Phi}^{h-1}\mathbf{\Gamma Qe} & \geq & \mathbf{u}_{min}
\end{array}\right] \tag{6.3}$$

The primal canonical form requires that:

- All the constraints be of the ($\leq$) inequality form.
- Terms on the left-hand side be linearly related to the optimisation variables $\mathbf{Q}$.
- The terms on the right-hand side be all non-negative constants.

Thus (6.3) can be reworked to give (6.4)

$$\left[\begin{array}{rcl}
\mathbf{DQe} & \leq & \mathbf{u}_{max} - \mathbf{Cx} \\
\mathbf{C\Gamma Qe} & \leq & \mathbf{u}_{max} - \mathbf{C\Phi x} \\
 & \vdots & \\
\mathbf{C\Phi}^{h-1}\mathbf{\Gamma Qe} & \leq & \mathbf{u}_{max} - \mathbf{C\Phi}^h\mathbf{x} \\
-\mathbf{DQe} & \leq & -\mathbf{u}_{min} + \mathbf{Cx} \\
-\mathbf{C\Gamma Qe} & \leq & -\mathbf{u}_{min} + \mathbf{C\Phi x} \\
 & \vdots & \\
-\mathbf{C\Phi}^{h-1}\mathbf{\Gamma Qe} & \leq & -\mathbf{u}_{min} + \mathbf{C\Phi}^h\mathbf{x}
\end{array}\right] \tag{6.4}$$

The constraints in (6.4) are now almost in the primal canonical form, apart from the fact that the optimisation variables, $\mathbf{Q}$, are *caught-up* in the matrix algebra of the terms on the left-hand side and as such the coefficients of the optimisation variables have not been isolated.

Recall that the elements q_i of the error governor, $\mathbf{Q}$, were stacked to form a diagonal matrix of dimension $m \times m$ and then multiplied by the error vector, $\mathbf{e}$, to produce the applied input to the controller, that is

$$\hat{\mathbf{e}} = \mathbf{Qe} = \mathbf{Q}(\mathbf{r} - \mathbf{y})$$

Each element q_i of $\mathbf{Q}$ scales only the corresponding element e_i of $\mathbf{e}$ since $\mathbf{Q}$ is diagonal by definition.

The matrix and vector notation for $\mathbf{Q}$ and $\mathbf{e}$ respectfully was useful for the controller representation. The problem is that this now spoils the formulation of the linear programming problem. In the context of linear programming it would have been more convenient if the error vector had been a diagonal matrix and the error governor the column vector. Fortunately this can be arranged by exploiting the diagonality of the error governor $\mathbf{Q}$ as follows:

Define a diagonal matrix **E** and a column vector **q** such that the product **E q** is given by (6.5)

$$\mathbf{E}\,\mathbf{q} = \begin{bmatrix} e_1 & 0 & \cdots & 0 \\ 0 & e_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e_m \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_m \end{bmatrix} = \begin{bmatrix} e_1q_1 & 0 & \cdots & 0 \\ 0 & e_2q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e_mq_m \end{bmatrix} \tag{6.5}$$

and by earlier definition recall that

$$\mathbf{Q}\,\mathbf{e} = \begin{bmatrix} q_1 & 0 & \cdots & 0 \\ 0 & q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_m \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_m \end{bmatrix} = \begin{bmatrix} q_1e_1 & 0 & \cdots & 0 \\ 0 & q_2e_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_me_m \end{bmatrix} \tag{6.6}$$

Now because $\mathbf{e}_i$ and $\mathbf{q}_i$ are scalars the following equality holds:

$$\begin{bmatrix} e_1q_1 & 0 & \cdots & 0 \\ 0 & e_2q_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e_mq_m \end{bmatrix} = \begin{bmatrix} q_1e_1 & 0 & \cdots & 0 \\ 0 & q_2e_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & q_me_m \end{bmatrix} \tag{6.7}$$

Hence from (6.7) the equality $\mathbf{E}\,\mathbf{q} = \mathbf{Q}\,\mathbf{e}$ has been established. Therefore the constraints can be rewritten, making the substitution $\mathbf{E}\,\mathbf{q} = \mathbf{Q}\,\mathbf{e}$, to give:

$$\begin{bmatrix} \mathbf{DEq} & \leq & \mathbf{u}_{max} - \mathbf{Cx} \\ \mathbf{C\Gamma Eq} & \leq & \mathbf{u}_{max} - \mathbf{C\Phi x} \\ & \vdots & \\ \mathbf{C\Phi}^{h-1}\mathbf{\Gamma Eq} & \leq & \mathbf{u}_{max} - \mathbf{C\Phi}^{h}\mathbf{x} \\ & & \\ -\mathbf{DEq} & \leq & -\mathbf{u}_{min} + \mathbf{Cx} \\ -\mathbf{C\Gamma Eq} & \leq & -\mathbf{u}_{min} + \mathbf{C\Phi x} \\ & \vdots & \\ -\mathbf{C\Phi}^{h-1}\mathbf{\Gamma Eq} & \leq & -\mathbf{u}_{min} + \mathbf{C\Phi}^{h}\mathbf{x} \end{bmatrix} \tag{6.8}$$

Besides the constraints on the control variable, there was a constraint on the decision variables as well, namely:

$$0 \leq ||\mathbf{Q}||_2 \leq 1$$

This implies that the elements of q_i of **Q** must all satisfy:

$$0 \leq q_i \leq 1$$

which intern implies:

$$0 \leq ||\mathbf{q}||_2 \leq 1$$

By definition of the linear programming form, the decision variables are always nonnegative, thus, the lower bound inequality is automatically catered for and may be dispensed with leaving the additional constraint:

$$||\mathbf{q}||_2 \leq \mathbf{I}$$

and hence the optimisation variable, $\mathbf{q}$, can now be isolated from its coefficients to give the full set of constraints:

$$\begin{bmatrix} \mathbf{DE} \\ \mathbf{C\Gamma E} \\ \vdots \\ \mathbf{C\Phi}^{h-1}\mathbf{\Gamma E} \\ -\mathbf{DE} \\ -\mathbf{C\Gamma E} \\ \vdots \\ -\mathbf{C\Phi}^{h-1}\mathbf{\Gamma E} \\ \mathbf{I} \end{bmatrix} \mathbf{q} \leq \begin{bmatrix} \mathbf{u}_{max} - \mathbf{Cx} \\ \mathbf{u}_{max} - \mathbf{C\Phi x} \\ \vdots \\ \mathbf{u}_{max} - \mathbf{C\Phi}^h\mathbf{x} \\ -\mathbf{u}_{min} + \mathbf{Cx} \\ -\mathbf{u}_{min} + \mathbf{C\Phi x} \\ \vdots \\ -\mathbf{u}_{min} + \mathbf{C\Phi}^h\mathbf{x} \\ \mathbf{I} \end{bmatrix} \tag{6.9}$$

where $\mathbf{I}$ is the $m \times 1$ unity vector.

and by setting

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{DE} \\ \mathbf{C\Gamma E} \\ \vdots \\ \mathbf{C\Phi}^{h-1}\mathbf{\Gamma E} \end{bmatrix} = [(h+1)p \times m]$$

$$\hat{\mathbf{u}}_{max} = \begin{bmatrix} \mathbf{u}_{max} \\ \mathbf{u}_{max} \\ \vdots \\ \mathbf{u}_{max} \end{bmatrix} \qquad [(h+1)p \times 1]$$

$$\hat{\mathbf{u}}_{min} = \begin{bmatrix} \mathbf{u}_{min} \\ \mathbf{u}_{min} \\ \vdots \\ \mathbf{u}_{min} \end{bmatrix} \qquad [(h+1)p \times 1]$$

$$\hat{\mathbf{b}} = \begin{bmatrix} \mathbf{Cx} \\ \mathbf{C\Phi x} \\ \vdots \\ \mathbf{C\Phi}^h\mathbf{x} \end{bmatrix} \qquad [(h+1)p \times 1]$$

Then:

$$\mathbf{A} = \begin{bmatrix} \hat{\mathbf{A}} \\ -\hat{\mathbf{A}} \\ \mathbf{I} \end{bmatrix} \qquad [2(h+1)p \times m]$$

$$\mathbf{b} = \begin{bmatrix} \hat{\mathbf{u}}_{max} - \hat{\mathbf{b}} \\ -\hat{\mathbf{u}}_{min} + \hat{\mathbf{b}} \\ \mathbf{I} \end{bmatrix} \qquad [2(h+1)p \times 1]$$

So finally the constraints can be written as:

$$\mathbf{A}\,\mathbf{q} \leq \mathbf{b}$$

While the constraints have the appearance of the desired linear programming primal canonical form, the following question and answer check list proves that this is indeed the case:

- **Optimisation variable coefficients isolated?**
  The $\mathbf{A}$ matrix contains only constants and these constants are the coefficients of the optimisation variables, $\mathbf{q}$, only.

- **Optimisation variables $\mathbf{q} \geq \mathbf{0}$?**
  The definition of the optimisation variables was that $0 \leq ||\mathbf{q}||_2 \leq 1$. Thus the optimisation variables satisfy the linear programming restriction that $\mathbf{q} \geq \mathbf{0}$.

- **Inequalities ?**
  All the inequalities are written as $(\leq)$ inequalities.

- **Right hand side terms all positive?**
  An important observation to note is that $\mathbf{b} \geq \mathbf{0}$ for all admissible states, $\mathbf{x}$.

  Proof:
  An admissible state was defined, in Chapter 4, to be a controller state that does not lead to the controller issuing a saturated control for the free and unforced response of the controller, not now nor at any time in the future. Thus $\hat{\mathbf{b}}$ will always lie between the control bounds, $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$, that is $\mathbf{u}_{min} \leq \hat{\mathbf{b}} \leq \mathbf{u}_{max}$.

  Therefore
  $$\begin{array}{lcl} \hat{\mathbf{b}} \leq \mathbf{u}_{max} & \Rightarrow & \hat{\mathbf{u}}_{max} - \hat{\mathbf{b}} \geq 0 \\ \hat{\mathbf{b}} \geq \mathbf{u}_{min} & \Rightarrow & -\hat{\mathbf{u}}_{min} + \hat{\mathbf{b}} \geq 0 \end{array}$$

  Hence
  $$\mathbf{b} \geq \mathbf{0}$$

6.3 Summary

The anti-saturation problem has been properly transformed into the linear programming primal canonical form given by:

$$\begin{array}{l} \text{maximise } z = \mathbf{cq} \\ \text{subject to} \\ \begin{array}{rcl} \mathbf{Aq} & \leq & \mathbf{b} \\ \mathbf{q} & \geq & \mathbf{0} \end{array} \end{array} \tag{6.10}$$

In Chapter 5 it was determined that the most economical route to solving this linear programming problem was to transform the primal canonical problem to the dual canonical form as shown below:

$$\begin{array}{l} \text{minimise } z = \mathbf{b}^t\mathbf{w} \\ \text{subject to} \\ \begin{array}{rcl} \mathbf{A}^t\mathbf{w} & \geq & \mathbf{c}^t \\ \mathbf{w} & \geq & \mathbf{0} \end{array} \end{array} \tag{6.11}$$

Note now that because **b** is always positive, that the coefficients of the new optimisation variables **w** are thus all positive and thus of the correct form. The right-hand-side terms of the ($\geq$) inequality are all positive since **c** was unity in the original problem. This is also of the correct form. The canonical dual is therefore of the correct form as a whole and any controller can be processed by the proposed anti-saturation algorithm providing that the initial state of the controller is an admissible state.

The dual canonical form can then be converted to dual standard form by the addition of surplus variables, **s**, to give:

$$\begin{aligned} &\text{minimise } z = \bar{\mathbf{b}}\bar{\mathbf{w}} \\ &\text{subject to} \\ &\bar{\mathbf{A}}\bar{\mathbf{w}} \;=\; \bar{\mathbf{c}} \\ &\quad\bar{\mathbf{w}} \;\geq\; \mathbf{0} \end{aligned} \tag{6.12}$$

where

$$\begin{aligned} \bar{\mathbf{A}} &= \begin{bmatrix} \mathbf{A}^T & \mathbf{I} \end{bmatrix}_{[m\times(2(h+1)p+m)]} \\ \bar{\mathbf{b}} &= \begin{bmatrix} \mathbf{b}^T & \mathbf{I} \end{bmatrix}_{[1\times(2(h+1)p+m)]} \\ \bar{\mathbf{c}} &= \begin{bmatrix} \mathbf{c}^T \end{bmatrix}_{[m\times 1]} \\ \bar{\mathbf{w}} &= \begin{bmatrix} \mathbf{w} \\ \mathbf{s} \end{bmatrix}_{[(2(h+1)p+m\times 1)]} \end{aligned}$$

A necessary and sufficient condition for finding a dual feasible basis was that $\bar{b} \geq \mathbf{0}$. This is the case. Thus, the anti-saturation algorithm transformed from primal canonical to dual canonical to dual standard form is directly amenable to solution by the dual simplex method. This form also requires the minimum iterations for solution, namely, m.

Chapter 7

Simulation and Evaluation

In this chapter, simulation examples of the application of the proposed anti-saturation algorithm are performed. The purpose of these simulations are to demonstrate that the algorithm:

- Does prevent windup in both SISO and MIMO controllers.
- Offers enhanced performance over the existing MIMO anti-saturation algorithm proposed by Kapasouris.

Simulation is the only general method of analysis applicable to finding solutions of arbitrary linear and nonlinear differential and difference equations. Simulation, however, only finds particular solutions, that is, solutions to the equations with specific inputs, initial conditions and parametric conditions. For this reason, simulation does not supplant other forms of analysis. Important properties such as stability and conditional stability are not proven with simulations. Nevertheless, in the case of a nonlinear process, even if there is no proof or guarantee of stability, simulation can build confidence that under certain operating conditions expected, the system is stable. Although proof of stability is always desirable, sometimes it is necessary to rely on simulation alone [Franklin, Powell and Workman:517].

7.1 Example 1 - SISO system

This is a simulation of a second order SISO controller as shown in Figure 7.1. The linear controller is marked by the dashed section. The input to the controller, $\hat{e}$, is derived from the real error, e, scaled by the error governor EG. The output of the controller, u, is bounded by the actuator saturation limits of the plant to ± 1. The input to the plant is given by $\hat{u} = \mathrm{sat}(u)$.

The controller was discretised with a sample period of $\tau = 0.1$ seconds in the simulation. The initial conditions on the integrator states were zero. The controller was specified as a stable second order system with a damping of $\zeta = 0.2$, a natural frequency $\omega_n = 1.0$ rad/s and a steady state gain of unity. Note that the controller was not designed for any particular plant. The controller is synthetic and for illustrative purposes only.

During the simulations, the error input was held constant at $e = 0.9$. This can easily be arranged by letting the reference $r = y + 0.9$. That is, the reference input is always 0.9 units larger than the current plant output. The steady state gain of the controller is unity and the

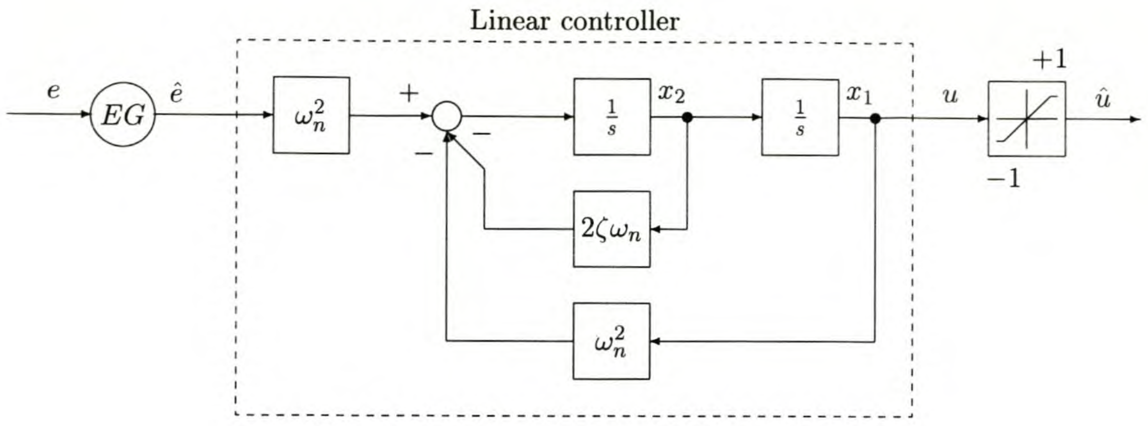

Figure 7.1: Block diagram of second order SISO controller.

controller output should therefore settle at $u = 0.9$ in the steady state. The controller state x_1 maps directly to the controller output u. The controller state x_2 remains an internal state of the controller.

Three simulations were performed, namely:

- A linear simulation.
- A nonlinear simulation with error governor horizon of 8 steps.
- A nonlinear simulation with error governor horizon of 3 steps.

The linear simulation is used as a benchmark for performance evaluation of the two nonlinear simulations. The two nonlinear simulations contrast the effect of the error governor horizon. These simulations are discussed in the subsections that follow.

7.1.1 Linear simulation

In the linear simulation the error governor is unity and thus $\hat{e} = e$. The output constraints play no role.

- Figure 7.2 shows the linear unconstrained response of the states of the controller. As expected the system is underdamped and oscillatory but stable.
- The magnitude of the controller state x_1 exceeds unity between approximately 2 and 4.5 seconds and between 9 and 10.25 seconds. It would be during these periods that controller windup would occur if the output, u, was constrained by the saturation element to unity.
- The time for x_1 to reach unity (the upper saturation limit) is about 2 seconds.
- The controller reaches steady state after about 20 seconds.

7.1.2 Nonlinear simulation with error governor horizon of 8 steps

This is the first of the two nonlinear simulations, demonstrating that saturation can be avoided by the action of the error governor. Refer to Figure 7.3 which shows the controller's states and

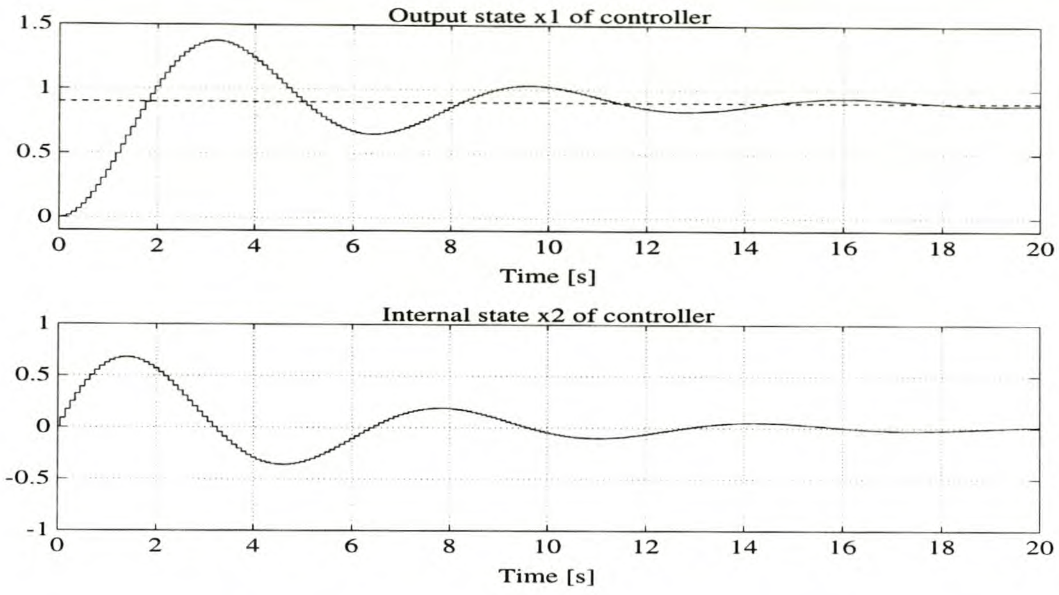

Figure 7.2: Linear simulation. The state signals are shown by the solid trace and the error signal by the dashed trace.

the error signal for comparison with the linear simulation in Figure 7.2. Refer to Figure 7.4 which shows the controller's output and error governor's signals.

- The output state, x_1, has been correctly constrained between the saturation limits of ± 1.
- The rise time of state x_1 to the saturation limit remains unchanged compared to the linear case.
- Note that the rate state, x_2, of the controller is modified by the error governor to ensure that the position state, x_1, does not overshoot the saturation bound.
- The settling time is also the same as for the linear case.
- The action of the error governor gain is to effectively remove or scale the input, $\hat{e}$, (forcing function) to the controller at the correct time step such that the controller's future unforced free response does not violate the saturation bound. Once the controller's output state starts to move away from the bound, the error governor returns to a gain of unity restoring linearity and loop gain to the controller loop.
- The error governor horizon of $h = 8$ steps was determined by trial and error.

7.1.3 Nonlinear simulation with error governor horizon of 3 steps

This is the second of the two nonlinear simulations, demonstrating that saturation cannot be avoided if the error governor horizon is not long enough. Refer to Figure 7.5 which shows the controller's states and the error signal for comparison with the linear simulation in Figure 7.2. Refer to Figure 7.6 which shows the controller's output and error governor's signals.

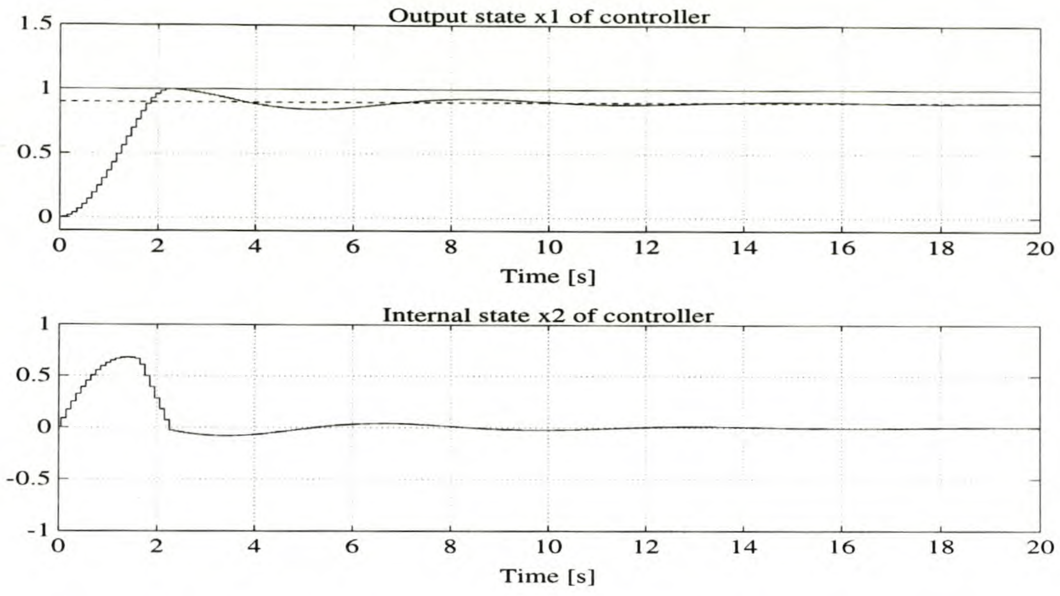

Figure 7.3: Controller states for the nonlinear simulation with an error governor horizon of 8 steps. The state signals are shown by the solid trace and the error signal by the dashed trace.

- The output state, x_1, has not been correctly constrained between the saturation limits ± 1. The controller thus winds up between 2 and 2.75 seconds. The error governor does not have sufficient horizon to predict the impending saturation bound violation and so by the time that the error governor removes the input, $\hat{e}$, (forcing function) to the controller, the controller has already attained an inadmissible state. The unforced free dynamic response of the controller therefore drives the output beyond the bounds. The error governor only has the freedom to limit the energy into the controller. No mechanism exists to remove excess energy from the controller, since the error governor may not become negative.
- The rise time of x_1 to the saturation limit remains unchanged compared with the linear case.
- The settling time was also the same as the linear case, in this example.
- The error governor horizon of $h = 3$ steps was chosen to demonstrate the significance of this parameter to the saturation prevention algorithm.

7.1.4 Summary

The following observations are made:

- The linear simulation output state, x_1, is driven to exceed the saturation bounds.
- The error governor, with a horizon of 8 steps, prevents the controller output from saturating by controlling the amount of energy into the controller. Thus the controller's state has been successfully constrained to the set of admissible states.

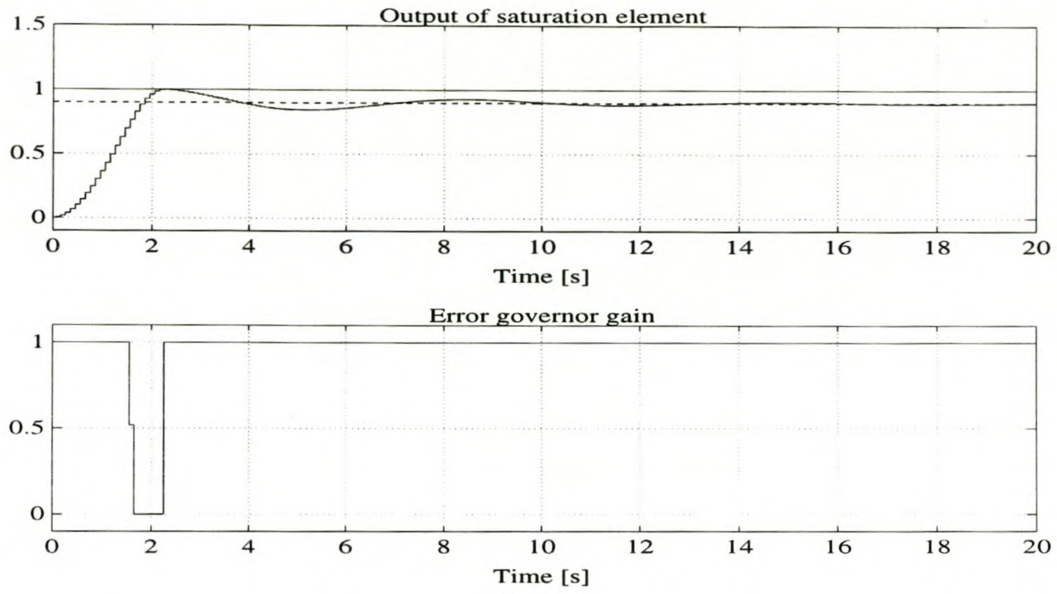

Figure 7.4: Controller output and error governor gain for the nonlinear simulation with an error governor horizon of 8 steps.

- The error governor, with a horizon of 3 steps, is unable to prevent the controller output from saturating because the horizon is too short and once too much energy has been allowed into the controller the free and unforced response of the controller causes saturation to occur. Thus the controller has been allowed to attain an inadmissible state.
- The error governor horizon is therefore an important parameter in the application of the proposed anti-saturation algorithms performance. The shorter the horizon, the less computational effort required to compute the anti-saturation error governor gains. The minimum number of steps (optimisation horizon) needs to be determined for optimal application.

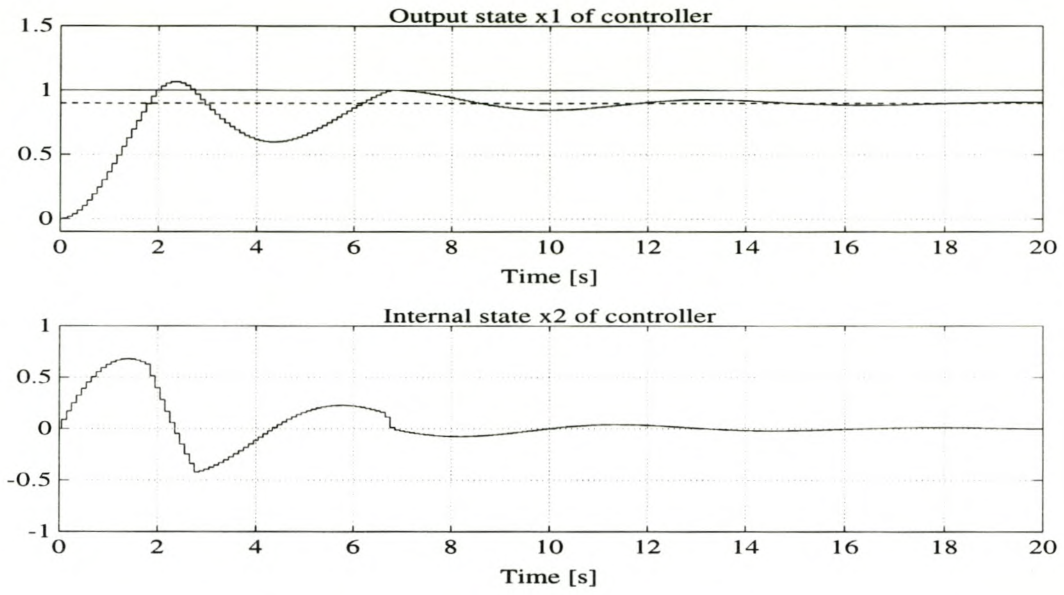

Figure 7.5: Controller states for the nonlinear simulation with an error governor horizon of 3 steps. The state signals are shown by the solid trace and the error signal by the dashed trace.

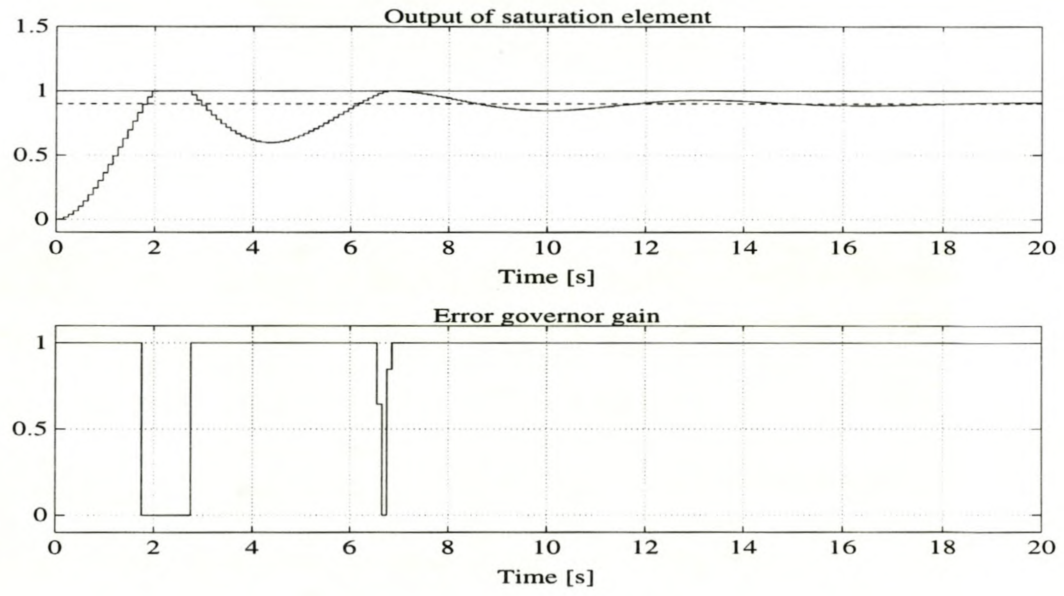

Figure 7.6: Controller output and error governor gain for the nonlinear simulation with an error governor horizon of 3 steps.

7.2 Example 2 - MIMO Simulation

Kapasouris introduced this 2-input 2-output example with the purpose of illustrating how the saturation can disturb the direction of the controls and hence alter the controller inversion of the plant. The *theoretical* plant, $G(s)$, has two zeros with low damping which the controller, $K(s)$, was designed to cancel. In addition, the controller does not have any free integrators and so the windup phenomena is not expected to occur [Kapasouris].

The plant, $G(s)$, has the following state space representation:

$$\begin{aligned}
\dot{\mathbf{x}}_p &= \begin{bmatrix} -1.5 & 1.0 & 0.0 & 1.0 \\ 2.0 & -3.0 & 2.0 & 0.0 \\ 0.0 & 0.5 & -2.0 & 1.0 \\ 1.0 & -1.5 & 0.0 & -5.0 \end{bmatrix} \mathbf{x}_p(t) + \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 0.0 \\ 1.0 & 1.0 \\ 0.0 & 1.8 \end{bmatrix} \hat{\mathbf{u}}(t) \\
\mathbf{y}(t) &= \begin{bmatrix} 0.0 & 2.4 & -3.1 & 1.0 \\ 1.0 & 6.0 & -0.5 & -2.8 \end{bmatrix} \mathbf{x}_p(t) \\
\hat{\mathbf{u}}(t) &= \mathrm{sat}(\mathbf{u}(t))
\end{aligned}$$

The plant has four real poles at $-0.577, -2.246, -2.722$ and -5.955 rad/s and a complex pair of zeros at $-0.544 \pm j\,2.423$ with a damping of 0.219 and two more zeros at infinity. Figure 7.7 shows the singular values of the plant. Note the lightly damped resonant zeros at 2.5 rad/s.

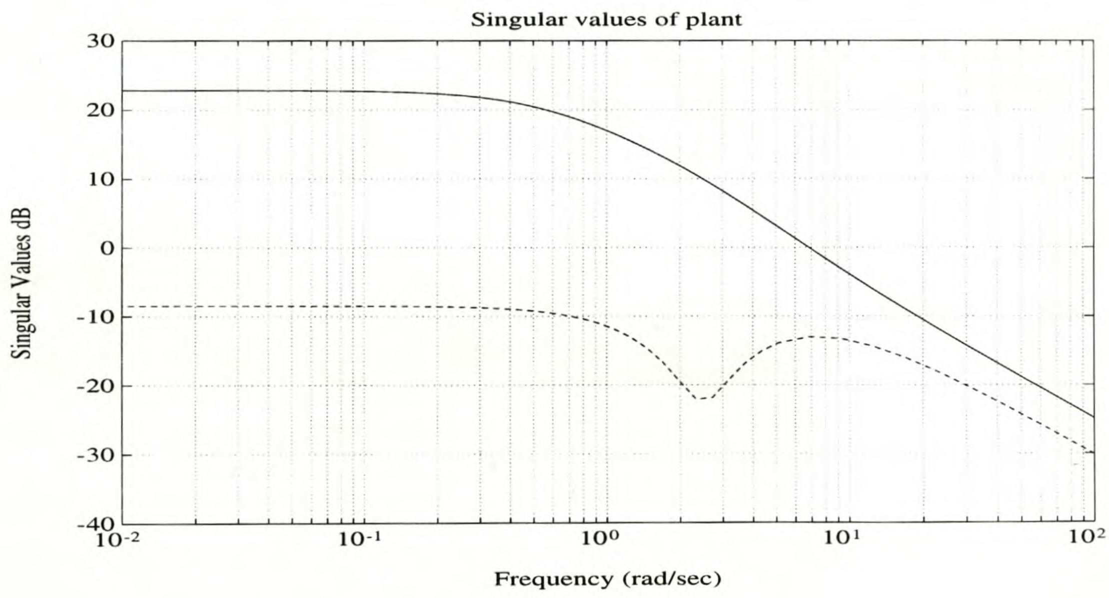

Figure 7.7: Singular values of the plant.

The controller, $K(s)$, as designed by Kapasouris, has the following state space representation:

$$\dot{\mathbf{x}}_c = \begin{bmatrix} -2.6093 & 1.4180 \\ -7.1476 & 1.5213 \end{bmatrix} \mathbf{x}_c(t) + \begin{bmatrix} -29.8308 & 2.9890 \\ -68.7543 & 10.8387 \end{bmatrix} \hat{\mathbf{e}}(t)$$

$$\mathbf{u}(t) = \begin{bmatrix} -1 & 1 \\ 2 & -1 \end{bmatrix} \mathbf{x}_c(t)$$

The compensator has a complex pole pair that cancels the complex zeros of the plant at $-0.544 \pm j\,2.423$ rad/s and has no finite zeros. The singular values of the open loop controller and plant are shown in Figure 7.8.

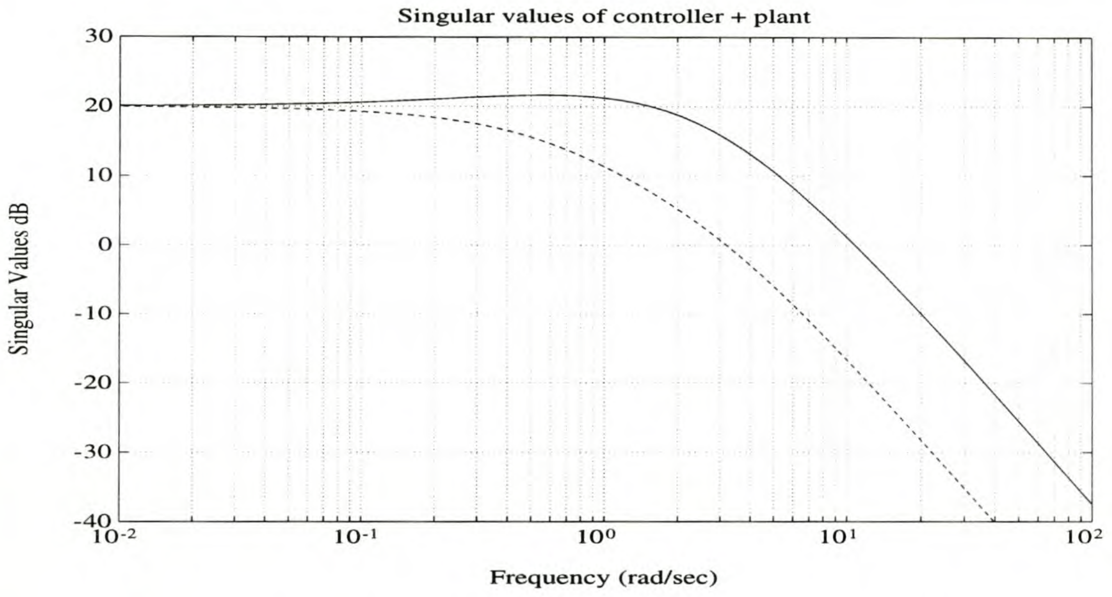

Figure 7.8: Singular values of the open loop controller and plant.

A linear simulation of the unconstrained system was performed. The results of this simulation are plotted as the dashed traces in the simulation plots in this section. The solid traces are the results of the constrained simulation.

Besides the reference linear simulation, the following three constrained simulations were performed:

- No anti-saturation mechanism.
- The proposed anti-saturation algorithm.
- Kapasouris's anti-saturation algorithm.

7.2.1 Simulation with no Anti-saturation Mechanism

In Figure 7.9 and Figure 7.10 the dotted trace in each graph represents the response of the linear unconstrained system. The solid trace in each graph is the corresponding response of the constrained system when no anti-saturation algorithm is employed.

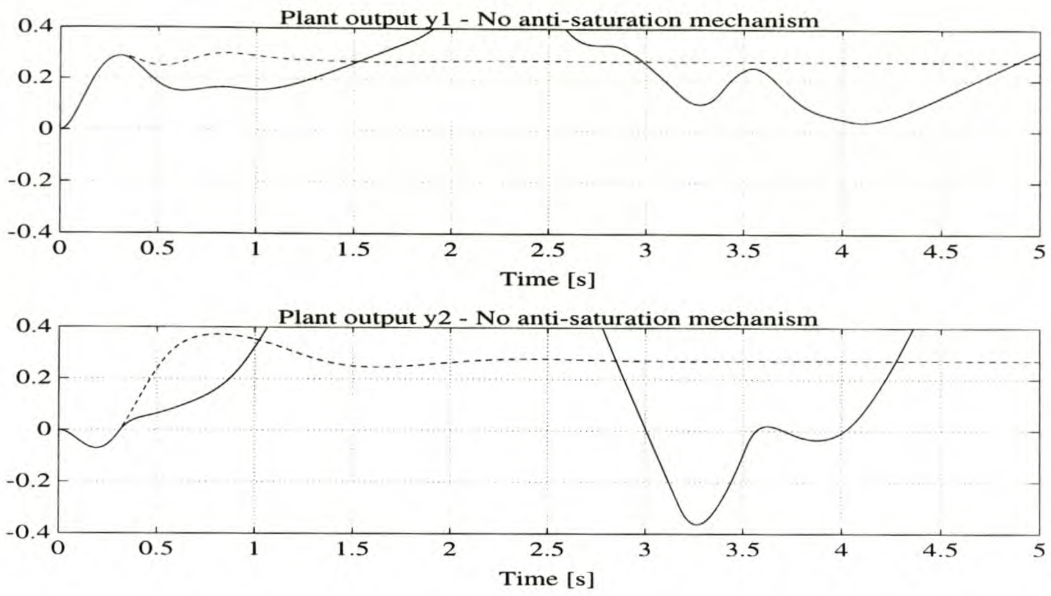

Figure 7.9: Outputs with no anti-saturation mechanism.

Figure 7.9 and Figure 7.10 show the response of the plant due to the saturation elements in both controls. The plant does not see the same control that the controller issues. Kapasouris argues that this changed the direction of the applied control vector resulting in the unsatisfactory performance shown. It is further argued by Kapasouris that because the controller contains no free integrators that this response is therefore not due to the controller winding up but rather due to the change in the control vector direction which spoils the inversion of the plant by the controller. In the comparative simulations that follow it is evident that the direction of the control vector is not the concern, but that consistency between the controller states and the applied control vector is relevant. The algorithm proposed in this thesis prevents saturation and maintains better performance than the algorithm of Kapasouris without taking the direction of the applied control vector in to consideration. This is discussed in the comparative simulation results in the sequel.

7.2.2 Simulation with Anti-saturation Mechanism

In Figure 7.11, Figure 7.12, Figure 7.13 and Figure 7.14 the dotted trace in each graph represents the response of the linear unconstrained system with the error governor $EG = 1$. The solid trace in each graph is the corresponding response of the parameter when the anti-saturation algorithm is employed. The upper graph of each Figure is the result of the proposed algorithm while the lower graph of each Figure is the result of applying Kapasouris's algorithm. The controller and plant are initially at rest and a step command of $r = [\,0.3 \quad 0.3\,]^T$ was issued to the system. These simulations replicate the results found in Kapasouris's work.

- Figure 7.11 and Figure 7.12 show the time histories of the two plant outputs. The proposed algorithm demonstrates a closer resemblance to the linear controller and reaches steady state faster than the response of Kapasouris. This is particularly evident in the second output, y_2.

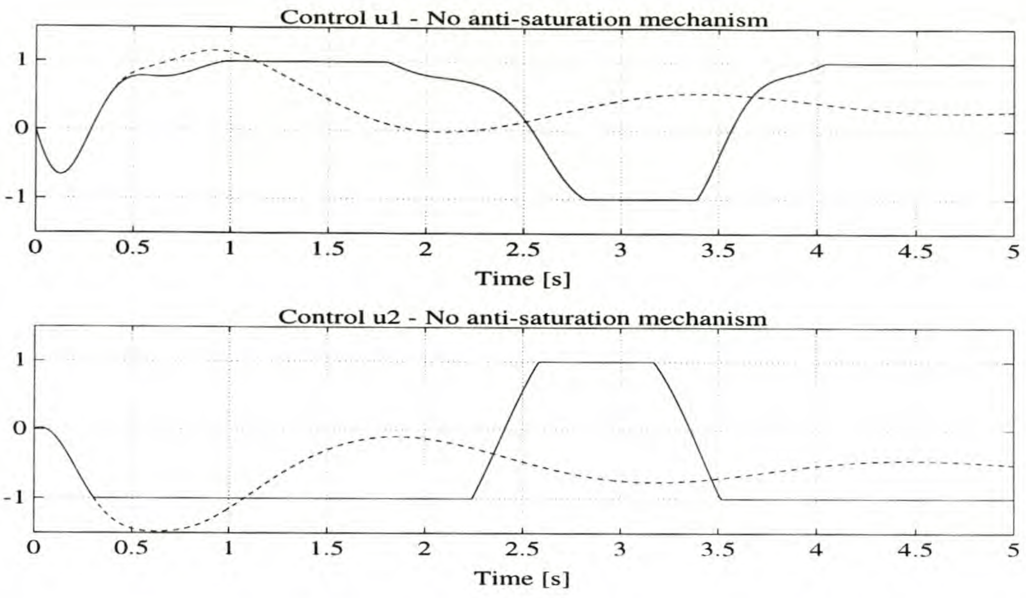

Figure 7.10: Controls with no anti-saturation mechanism.

- Figure 7.13 and Figure 7.14 show the time history of the two control signals. The proposed algorithm follows the linear response for longer before levelling off at the saturation bound. Kapasouris's algorithm shows earlier deviation from the ideal linear signal, particularly for the second control, u_2. Both algorithms constrain the system correctly. The proposed algorithm leaves the saturation bound earlier than Kapasouris's algorithm thus returning to the linear system behaviour quicker. The overall difference between the linear response and the constrained response is less for the proposed algorithm.

The improvement between both algorithms is evident, albeit it marginal. The margin of improvement was limited by the choice of the singular values of the controller and plant loop. This example, taken from Kapasouris for comparative purposes, had its singular values designed to be coincident at low frequencies. Thus the low frequency performance of the controller is identical in both channels. The cross-over frequency difference between maximum and minimum singular values is not too large either, giving more or less equal performance and bandwidth over all frequencies of interest. This is a somewhat contrived situation. For example, typical loop dynamics between the longitudinal channels of an aircraft MIMO autopilot, controlling airspeed and altitude with the elevator and engine will most likely have maximum and minimum singular values far from coincident. In order to force these singular values to be coincident, it would be necessary to either boost the performance of the engine or to drop the performance of the elevator. In practice neither are practical options. The elevator has large bandwidth over influencing both airspeed and altitude compared with the bandwidth of influence that the engine has on these same outputs. This is a typical case where the controller would have a triangular dominant structure. The algorithm of Kapasouris cannot discriminate between cause and effect, that is the error signal responsible for a control saturating and so deals with the problem by scaling all error signals equally. The proposed algorithm has the required degrees of freedom necessary to make this discrimination and will thus not penalise all error signals equally. Indeed, in the case of a totally unrelated error signal, as in the non-interacting channel of a triangular sturctured controller, this error signal will not be scaled at all when the interacting channel has saturated.

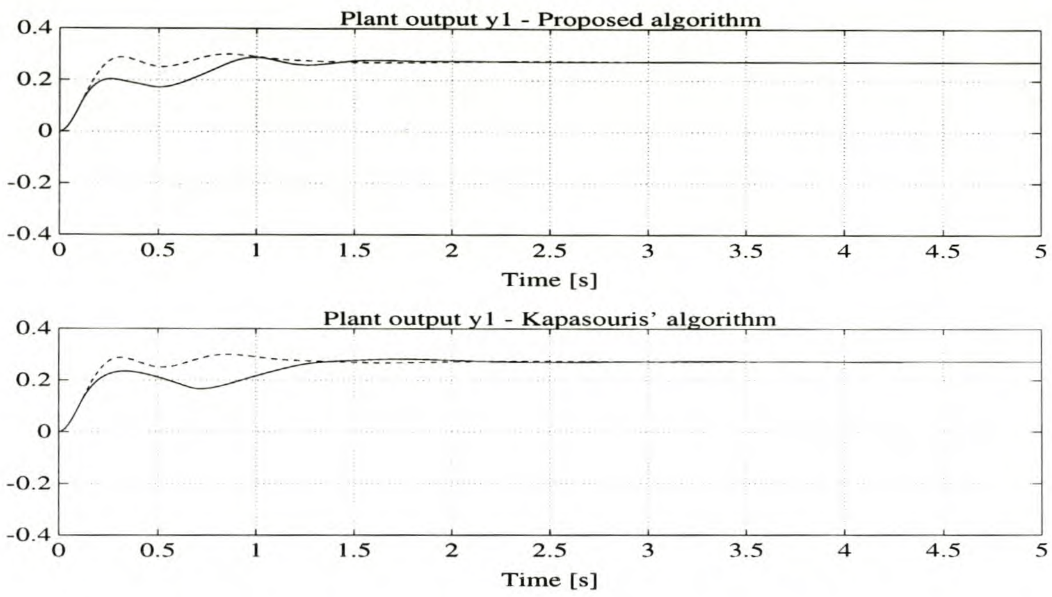

Figure 7.11: Output 1 with error governor anti-saturation mechanism.

Figure 7.15 shows the time history of the error governor gains for both algorithms. In the case of Kapasouris, the gain is a scalar. In the proposed algorithm there are individual gains acting on each error input into the controller. This extra freedom is used in the proposed algorithm to optimise the energy allowed into the controller. With Kapasouris's method, both channels are switched off simultaneously at time t=0.15 s. In the case of the proposed method, the second channel ϵ_2 is kept at full strength (as in the linear case) until $t = 0.3$ s. Likewise, the error governor in the proposed algorithm returns both ϵ_1 and ϵ_2 to unity before the error governor, λ, of Kapasouris returns to unity. Thus, the net energy into the controller is maximised by the proposed algorithm, while avoiding saturation. Furthermore, if the two channels of the particular plant and controller had not been so equally matched in dynamics and cross-coupling, then the performance of the one channel that might be saturating would have had far less impact on the unrelated channel. Clearly this would not have been true for the algorithm of Kapasouris whereby all channels (related or not) are equally penalised for a saturation anywhere in the control system.

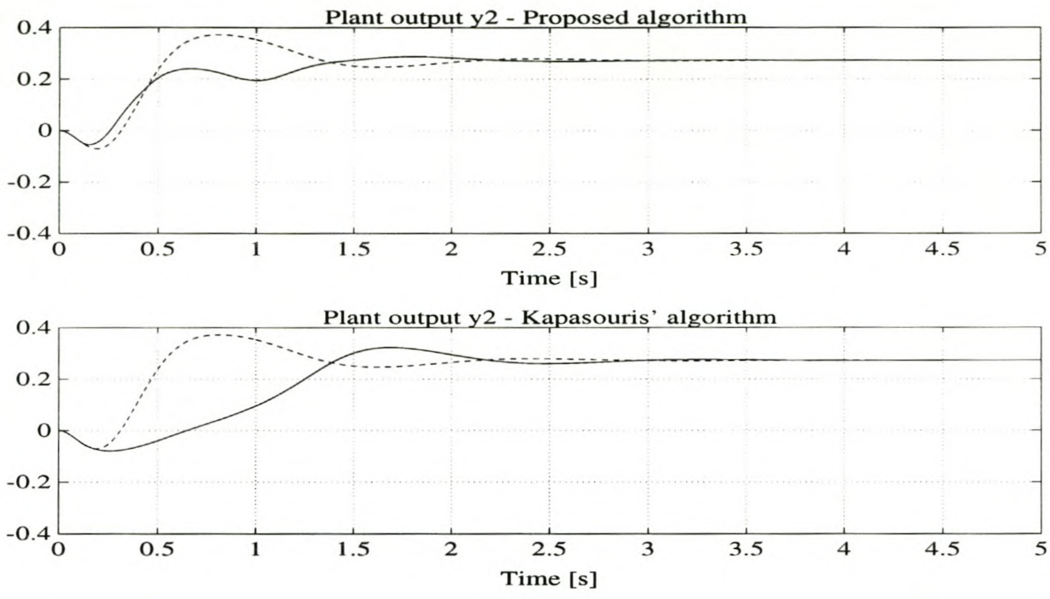

Figure 7.12: Output 2 with error governor anti-saturation mechanism.

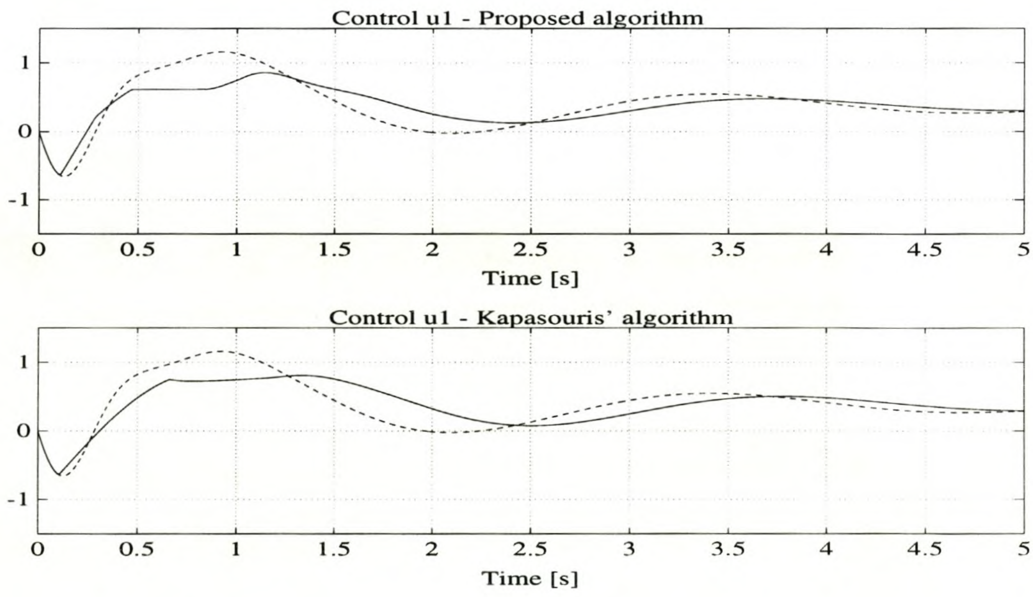

Figure 7.13: Control 1 with error governor anti-saturation mechanism.

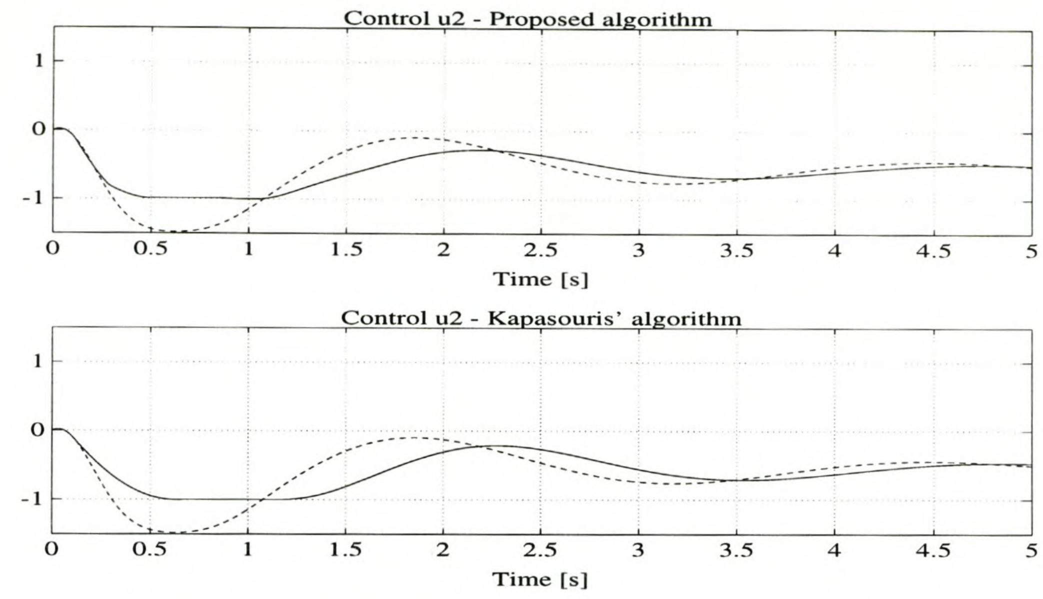

Figure 7.14: Control 2 with error governor anti-saturation mechanism.

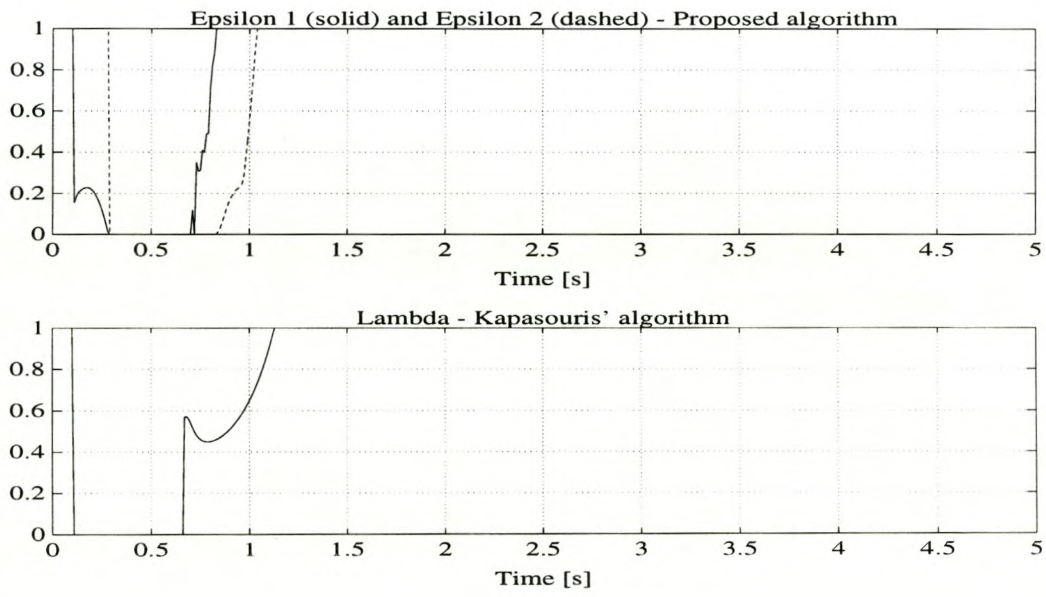

Figure 7.15: Error governor

Chapter 8

Conclusions and Recommendations

The following may be concluded from this research:

- This research stemmed from the idea that it should be possible to prevent the windup and associated performance degradation resulting from controller saturation by preventing saturation from occurring in the first place. The hypothesis was that it should be possible to condition the error signal input into the controller in such a way that the performance of the control system would be optimal up to the saturation limit of the actuator and that deeper saturation (or windup) beyond the limit would be averted.

  This has been demonstrated through simulation.

- An anti-windup algorithm was developed for unconditionally stable MIMO (or SISO) controllers.
  - This algorithm has superior performance compared with existing techniques and was demonstrated in simulation.
  - The algorithm is posed as a linear optimisation problem subject to linear inequality constraints, leading to solution by linear programming techniques.
  - A practical and efficient implementation of the algorithm is presented by transforming the problem into its dual linear programming form. This reduced the number of iterations required by the linear programming algorithm by orders of magnitude. The dual form problem is then solved using the dual simplex method, rather than the direct simplex method. This guaranteed an initial basic feasible solution and an optimal bounded final solution in a finite and predictable number of iterations.

The following topics are recommended for future work in this field of research:

- Extend the proposed algorithm to handle time-varying controllers, such as gain scheduled controllers. This will impact on the assumption made thus far that the initial state of the controller is feasible. As the controller gains are changed, the potential exists for the previously feasible controller state to become infeasible, resulting in windup of the controller dynamics.

- Determine a procedure for determining the minimal or optimal look ahead horizon.

- Investigate efficient matrix multiplication algorithms or other methods for determining the higher order state propagation matrices $\mathbf{\Phi}^h$ required by the proposed anti-saturation algorithm. This is of particular interest for controllers of high order, such as model based controllers that include a model of the plants dynamics in the form of a Kalman Filter or full state estimator, for example Linear Quadratic Gaussian (LQG) Controllers.
- Investigate the proposed anti-saturation algorithm in the context of a *Variable Structure Controller* framework. The anti-saturation algorithm dynamically modifies the structure of the linear controller by scaling input channel gains as well as effectively removing input channels completely by reducing the scaling gain to zero as and when necessary to avoid saturation in the controls.

Appendix A

Linear Programming Options

A.1 Purpose

The purpose of this section is to analyse the options available in linear programming and to determine the dimension of the problem when converted to standard form via the various possible transformation routes. The dimension of the problem when in standard form impacts directly on the number of iterations required to solve the linear program.

A.2 Problem Transformation Options

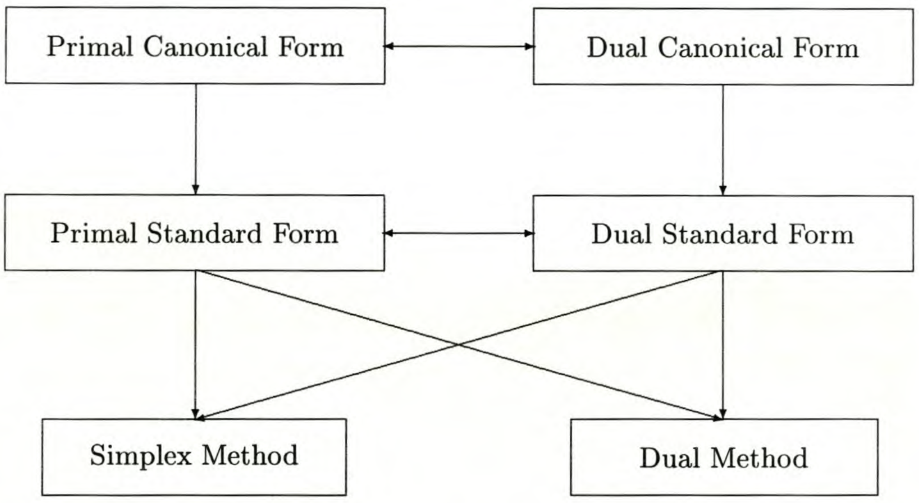

Figure A.1: Linear Programming Overview

Figure A.1 shows diagramitcally the options available in linear programming. The linear problem in canonical form can be transformed between the primal canonical form and dual canonical form. Likewise, the linear problem in standard form can be transformed between the primal standard form and the dual standard form. All the solution methods (either simplex or dual method) operate on the linear program when in the standard form. Thus the canonical form (either primal or dual) must first be converted to the standard form (either primal or dual).

Assuming one always begins with the primal canonical form, there are four routes one could take to the standard form, each eventually being solved by applying either the simplex method or the dual method.

1. Primal Canonical Form
   → Primal Standard Form
   → Solve
2. Primal Canonical Form
   → Primal Standard Form
   → Dual Standard Form
   → Solve
3. Primal Canonical Form
   → Dual Canonical Form
   → Dual Standard Form
   → Solve
4. Primal Canonical Form
   → Dual Canonical Form
   → Dual Standard Form
   → Primal Standard Form
   → Solve

It is important to note that although (1) and (4) are both in primal standard form, because of the different routes taken to reach this form, the two linear programs are not equal in dimension. Likewise, although (2) and (3) are both in dual standard form, their dimensions are very different. Therefore, while the forms are equivalent and the solutions the same, the actual linear programs are very different in dimension. The dimensions of these seemingly equivalent problems are different because the slack and/or surplus variables are introduced at different stages of the transformation.

Consider the linear programming problem in Primal Canonical Form, having l constraints and m variables.

$$\begin{array}{l} \text{maximise } z = \mathbf{cx} \\ \text{subject to} \\ \begin{array}{rcl} \mathbf{Ax} & \leq & \mathbf{b} \\ \mathbf{x} & \geq & \mathbf{0} \end{array} \end{array} \tag{A.1}$$

with

$$\begin{array}{l} \mathbf{A}_{l\times m} \\ \mathbf{b}_{l\times 1} \\ \mathbf{c}_{1\times m} \\ \mathbf{x}_{m\times 1} \end{array} \tag{A.2}$$

A.3 Route 1

- Primal Canonical Form
  → Primal Standard Form
  → Solve

To convert from the primal canonical form to the primal standard form, one must add l slack/surplus variables, $\mathbf{s}$, to convert the l inequality constraints into equalities, giving:

$$\text{maximise } z = \begin{bmatrix} \mathbf{c}_{1\times m} & \mathbf{I}_{1\times l} \end{bmatrix}_{1\times(m+l)} \begin{bmatrix} \mathbf{x}_{m\times 1} \\ \mathbf{s}_{l\times 1} \end{bmatrix}_{(m+l)\times 1}$$

$$\begin{aligned} &\text{subject to} \\ &\begin{bmatrix} \mathbf{A}_{l\times m} & \mathbf{I}_{l\times l} \end{bmatrix}_{l\times(m+l)} \begin{bmatrix} \mathbf{x}_{m\times 1} \\ \mathbf{s}_{l\times 1} \end{bmatrix}_{(m+l)\times 1} = \mathbf{b}_{l\times 1} \\ &\begin{bmatrix} \mathbf{x}_{m\times 1} \\ \mathbf{s}_{l\times 1} \end{bmatrix}_{(m+l)\times 1} \geq \mathbf{0} \end{aligned} \tag{A.3}$$

Hence, this Route to primal standard form results in:

- l equality constraints
- $m + l$ variables

A.4 Route 2

- Primal Canonical Form
  → Primal Standard Form
  → Dual Standard Form
  → Solve

To convert the primal standard form, of Route 1, to dual standard form requires transposing the problem. Hence this Route to the dual standard form results in:

- $m + l$ equality constraints
- l variables

A.5 Route 3

- Primal Canonical Form
  → Dual Canonical Form
  → Dual Standard Form
  → Solve

To convert the primal canonical form to the dual canonical form requires transposing the problem, giving:

$$\begin{array}{l} \text{minimise } z = \mathbf{b}^T\mathbf{w} \\ \text{subject to} \\ \begin{array}{rcl} \mathbf{A}^T\mathbf{w} & = & \mathbf{c}^T \\ \mathbf{w} & \geq & \mathbf{0} \end{array} \end{array} \tag{A.4}$$

where the new variables, $\mathbf{w}$, have the dimension $l \times 1$.

To convert this problem, in dual canonical form to dual standard form requires the addition of m slack/surplus variables, $\mathbf{s}$, to convert the inequality constraints into equalities, giving:

$$\text{minimise } z = \begin{bmatrix} (\mathbf{b}_{1\times l})^T & \mathbf{I}_{1\times m} \end{bmatrix}_{1\times(l+m)} \begin{bmatrix} \mathbf{w}_{l\times 1} \\ \mathbf{s}_{m\times 1} \end{bmatrix}_{(l+m)\times 1}$$

subject to

$$\begin{aligned} \begin{bmatrix} (\mathbf{A}_{l\times m})^T & \mathbf{I}_{m\times m} \end{bmatrix}_{l\times(m+l)} \begin{bmatrix} \mathbf{w}_{l\times 1} \\ \mathbf{s}_{m\times 1} \end{bmatrix}_{(l+m)\times 1} &= \begin{bmatrix} (\mathbf{c}_{l\times 1})^T \end{bmatrix}_{1\times l} \\ \begin{bmatrix} \mathbf{w}_{l\times 1} \\ \mathbf{s}_{m\times 1} \end{bmatrix}_{(l+m)\times 1} &\geq \mathbf{0} \end{aligned} \tag{A.5}$$

Hence, this Route to primal standard form results in:

- m equality constraints
- $l + m$ variables

A.6 Route 4

- Primal Canonical Form
  $\rightarrow$ Dual Canonical Form
  $\rightarrow$ Dual Standard Form
  $\rightarrow$ Primal Standard Form
  $\rightarrow$ Solve

To convert the dual standard form, of Route 3, to primal standard form requires transposing the problem. Hence this Route to the primal standard form results in:

- $l + m$ equality constraints
- m variables

Appendix B

Bibliography

B.1 Applicable References

1. A. Bemporad, A. Casavola and E. Mosca, *"A nonlinear Command Governor for Constrained Control Systems"*, 13th Triennial World Congress, IFAC 1996.

2. P.J. Campo and M. Morari, *"Robust Control of Processes Subject to Saturation Nonlinearities"*, Computers Chem Engng, vol 14, pp. 343-358, 1990.

3. E.B. Castelan, J.M. Gomes da Silva, Jr. and J.E.R. Cury, *"A Reduced-Order Framework Applied to Linear Systems with Constrained Controls"*, IEEE Transactions on Automatic Control, Vol. 41, No. 2 February 1996.

4. E.J. Davidson and J. Jiao, *"The Robust Servomechanism Problem with Input Saturation Constraint"*, pp. 2227-2231.

5. G.F. Franklin, J.D. Powel, M.L. Workman, *"Digital Control of Dynamic Systems"*, Addison-Wesley Publishing Company.

6. J.P. Ignizio, T.M. Cavalier *"Linear Programming"*, Prentice Hall International Series in Industrial and Systems Engineering.

7. P. Kapasouris *"Design for Performance Enhancement in Feedback Control Systems with Multiple Saturating Nonlinearities Ph.D. Thesis"*, Massachusetts Institute of Technology, 1988.

8. M.V. Kothare, P.J. Campo, M. Morari, and C.N. Nett, *"A Unified Framework for the Study of Anti-windup Designs"*, Automatica, Vol. 30 No. 12 pp 1869-1883, 1994.

9. Z. Lin, A.A. Stoorvogel and A. Saberi, *"Output Regulation for Linear Systems Subject to Input Saturation"*, Automatica, Vol. 32, No. 1, pp. 29-47, 1996.

10. R.H. Middleton, *"Dealing with Actuator Saturation"*, The Control Handbook. Editor: W.S. Levine. A CRC Handbook Published in Cooperation with IEEE Press, 1996.

11. K. Ogata, *"Modern Control Engineering"*, Prentice-Hall International Editions.

12. G.A. Pajunen and N. Erdol, *"Time-varying, Saturating Feedback Control of Linear Systems under State and Control Constraints"*, Proceedings of the 29th Conference on Decision and Control, December 1990.

13. M. Pachter and R.B. Miller, *"Manual Flight Control with Saturating Actuators IEEE Control Systems"*, pp. 10-19, February 1998.

14. M. Sznaier and M. J. Damborg, *"Heuristically enhanced feedback control of constrained discrete time systems"*, Automatica, 26, pp. 521-532, 1990.

15. M. Sznaier and M. J. Damborg, *"An Analog 'Neural Net' Based Suboptimal Controller for Constrained Discrete-time Linear Systems"*, Automatica, Vol. 28, No. 1, pp. 139-144, 1992.

16. K. Yoshida, H. Kawabe and M. Oya, *"An Optimal Design of Constrained Linear Control Systems using the Set of Admissible Gains"*, IFAC 13th Triennial World Congress, pp. 421-426, 1996.

B.2 Related References

1. M. Bikdash, E.M. Cliff and A.H. Nayfeh, *"Saturating and Time-Optimal Feedback Controls"*, Journal of Guidance, Control and Dynamics, Vol. 16, No. 3, May-June 1993.

2. G. Bitsoris and E. Gravalou, *"Comparison Principle, Positive Invariance and Constrained Regulation of Nonlinear Systems"*, Automatica, Vol. 31, No. 2, pp. 217-222, 1995.

3. K. A. Bordignon and W. C. Durham, *"Closed-Form Solutions to Constrained Control Allocation Problem"*, Journal of Guidance, Control and Dynamics, Vol. 18, No. 5, September-October 1995.

4. C.W. Chan and K. Hui, *"Design of Actuator Saturation Compensators Based on Phase Angle Specification"*, IFAC 13th Triennial World Congress, pp. 155-160, 1996.

5. G.P. Chen, O.P. Malik, G.S. Hope, *"Generalised Discrete Control System Design Method with Control Limit Considerations"*, IEE Proceedings Control Theory Applications, Vol. 141, No. 1, January 1994.

6. R.A. Hess, *"Feeback Design for Stable Plants with Input Saturation"*, Journal of Guidance, Control and Dynamics, Vol. 18, No. 5, pp. 1029-1035, September-October 1995.

7. P. Kapasouris, M. Athans, *"Multivariable Control Systems with Saturating Actuators Antireset Windup Strategies"*, Proceedings of the 1985 American Control Conference, pp.1579-1584, 1985.

8. P. Kapasouris, M. Athans and G. Stein, *"Design of Feedback Control Systems for Stable Plants with Saturating Actuators"*, IEEE Proceedings of the 27th Conference on Decision and Control, pp. 469-479, 1988.

9. P. Kapasouris, M. Athans, *"Control Systems with Rate and Magnitude Saturation for Neutrally Stable Open-Loop Systems"*, Proceedings of the 29th Conference on Decision and Control, Vol. 6, pp. 3404-3409, 1990.

10. J. Kim and Z. Bien, *"Robust Stability of Uncertain Linear Systems with Saturating Actuators"*, IEEE Transactions of Automatic Control, Vol. 39, No. 1, January 1994.

11. B.E.A. Milani, E.B. Castelan and S. Tarbouriech, *"Linear Regulator Design for Bounded Uncertain Discrete-Time Systems with Additive Disturbances"*, IFAC 13th Triennial World Congress, pp. 321-326, 1996.

12. M. Patcher, Chandler, Mears, *"Reconfigurable Tracking Control with Saturation"*, Journal of Guidance, Control and Dynamics, Vol. 18, No. 5, pp. 1016-1022, September-October 1995.

13. W.H. Press, B.P. Flannery, S.A. Teukolsky and W.T. Vetterling, *"Numerical Recipes in Pascal - The Art of Scientific Computing"*, Cambridge University Press.

14. G.V. Reklaitis, A. Ravindran and K.M. Radsdell, *"Engineering Optimisation Methods and Applications"*, John Wiley and Sons, Inc.

15. REA's Problem Solver, *"Operations Research"*, Research and Educational Association.

16. G.F. Wredenhagen and P.R. Belanger, *"A Piecewise-linear LQ Control for Systems with Input Constraints"*, Automatica, vol 30 No. 3, pp. 403-416, 1994.