Inclusion of Input Saturation in the Design of Dynamically Operable Plants

RHODA BAKER

Submitted in fulfillment of the requirements for the degree of
MASTER OF SCIENCE IN ENGINEERING

Department of Chemical Engineering
University of Cape Town
Cape Town, South Africa.

Supervisor: A/PROF. C.L.E. SWARTZ

September 2000

Synopsis

Dynamic operability reflects the quality with which a plant can be controlled using feedback, and is a function of both the design of the plant and its associated control system. A plant designed on the basis of steady-state considerations alone could exhibit poor dynamic characteristics, leading to a loss of economic performance and a reduced capacity to effectively handle safety and environmental constraints. This motivates the need for the development of quantitative techniques for dynamic operability assessment, as well as its incorporation into procedures for process plant design.

Optimization-based approaches to dynamic operability assessment permit simultaneous consideration of performance-limiting factors of nonminimum phase characteristics, input constraints and model uncertainty, and also provide considerable flexibility in the choice of performance criteria, decision variables and constraints. Recent work has incorporated operability requirements as constraints within a single optimal plant design problem formulation (Mohideen *et al.*, 1997; Bahri *et al.*, 1996).

Young and Swartz (1997) considered the rigorous inclusion of input saturation effects in optimizing control. Actuator saturation introduces discontinuities in the system model and, to avoid potential problems using a sequential optimization approach, two alternative formulations were proposed for solving the problem within a simultaneous solution framework. Input saturation discontinuities were handled by the introduction of slack variables and their inclusion in either bilinear or mixed-integer constraints resulting in a nonlinear or mixed-integer linear programming problem respectively. The formulations were applied to a linear system with dead time to find the economically optimal operating point for a controller with fixed structure and tunings when disturbance deviations are taken into account. It was shown that using a strictly linear controller in this case would lead to an overly conservative estimate of the feasible operating range and consequently, a suboptimal operating point.

The goals of this thesis are to investigate and propose an optimization framework for the synthesis and design of dynamically operable nonlinear plants that incorporate input saturation, thereby extending the work of Young and Swartz (1997),

Acknowledgements

Firstly, I'd like to thank my mom for all her love and support, and thanks to Mrs. Tyler for proofreading my thesis.

A big "*Doomo arigato gozaimashita*" goes out to my boyfriend and fellow sushi enthusiast, Gordon, who made sure I didn't go lonely or hungry on those long nights. *Kimo no koto igai wa kangaerarenai.*

Many thanks to my supervisor, Prof. C.L.E. Swartz for giving me the opportunity to work under him now and in the future.

And finally, thanks to the guys in the office (and some not in the office anymore) for an atmosphere that was as conducive to work as only 4 cups of coffee can be. I wish you the best of luck in your future ventures.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation and Goals

Dynamic operability reflects the quality with which a plant can be controlled using feedback, and is a function of both the design of the plant and its associated control system. A plant designed on the basis of steady-state considerations alone could exhibit poor dynamic characteristics, leading to a loss of economic performance and a reduced capacity to effectively handle safety and environmental constraints. This motivates the need for the development of quantitative techniques for dynamic operability assessment, as well as its incorporation into procedures for process plant design.

Optimization-based approaches to dynamic operability assessment permit simultaneous consideration of performance-limiting factors of nonminimum phase characteristics, input constraints and model uncertainty, and also provide considerable flexibility in the choice of performance criteria, decision variables and constraints. Recent work has incorporated operability requirements as constraints within a single optimal plant design problem formulation (Mohideen *et al.*, 1997; Bahri *et al.*, 1996).

Young and Swartz (1997) considered the rigorous inclusion of input saturation effects in optimizing control. Actuator saturation introduces discontinuities in the system model and, to avoid potential problems using a sequential optimization approach, two alternative formulations were proposed for solving the problem

and to propose a method whereby the resultant formulation can be efficiently solved.

It is shown through the use of case studies involving a CSTR and a binary distillation column how the optimization framework can be applied at the design stage. The case studies show that a less pessimistic economic optimum is obtained if rigorous input saturation handling is allowed, in comparison to the case where a strictly linear controller is used. However, it also takes into account the dynamic performance of the process and is thus guaranteed to be able to cope with specific variations in the operating conditions of the plant, unlike a design based on a purely steady state analysis.

An algorithm is presented that is able to reduce the computation time of the MILP problem that arises from the inclusion of the constraints developed by Young and Swartz (1997) for input saturation in a linear optimization framework. The extension of this algorithm to the nonlinear case is also provided.

Finally, it is recommended that improved methods be found for the solution of MINLP problems, and that the input saturation framework be extended to include other control schemes, as well as rigorous flexibility and robust stability analyses.

within a simultaneous solution framework. Input saturation discontinuities were handled by the introduction of slack variables and their inclusion in either bilinear or mixed-integer constraints resulting in a nonlinear or mixed-integer linear programming problem respectively. The formulations were applied to a linear system with dead time to find the economically optimal operating point for a controller with fixed structure and tunings when disturbance deviations are taken into account. It was shown that using a strictly linear controller in this case would lead to an overly conservative estimate of the feasible operating range and consequently, a suboptimal operating point.

The goals of this thesis are to investigate and propose an optimization framework for the synthesis and design of dynamically operable nonlinear plants that incorporate input saturation, and to propose a method whereby the resultant formulation can be efficiently solved.

The motivation follows:

To date, dynamic optimization-based design synthesis problems have only focused on linear controllers and have ignored the effects of actuator saturation. Young and Swartz (1997) arrived at a formulation that included saturation effects in optimizing control, but this was only applied to a linear model of a plant and the controller parameters were chosen beforehand as opposed to being variables in the optimization problem. Since a number of plant models derived using first principles are not linear it is necessary to extend this approach so that it is able to deal with processes with nonlinear dynamics. In addition, including saturation provides a less pessimistic estimate of the dynamically operable economic cost (Young and Swartz, 1997).

The resultant optimization superstructure is an example of a mixed-integer programming problem. Although methods do exist to solve problems of this class, arriving at an optimal solution can take considerable computation time. The amount of computation time required is proportional to the number of integer variables in the formulation. Since the inclusion of saturation constraints in the optimization problem requires the inclusion of a number of integer variables, the amount of time required to solve the problem can take considerably longer if input constraints are unnecessarily included.

1.2 Major Contributions

The issues addressed in this thesis and its achievements can be organized under two areas of interest.

1. **Formulation of an optimization superstructure for dynamically operable nonlinear plants that rigorously handles input saturation.**
   A superstructure that encompasses the traditional dynamically operable plant optimization structure, as well as allowing for input saturation constraints, is formulated. The structure utilizes orthogonal collocation on finite elements to discretize the nonlinear dynamic equations, and binary variables are used to represent the presence or absence of saturation at particular points of the controlled input profile. Using collocation allows the control trajectories to be optimized simultaneously in the problem thereby avoiding the potential instability of sequential dynamic optimization methods.

2. **Reduction in computation time for the resultant mixed-integer problem.**
   An algorithm that reduces the number of binary variables required to represent input saturation is presented. The algorithm is able to find a local optimum for the general case of a mixed-integer nonlinear problem. For the special case of a mixed-integer linear problem, the algorithm is able to guarantee a solution which is globally optimal.

1.3 Thesis Overview

Chapter 2

The basic principles and some of the recent work in the field of dynamic operability and the synthesis of dynamically operable plants are presented. A brief discussion on the current theory of saturation follows. Since the optimization of dynamic plants results in dynamic algebraic equations, the solution of dynamic algebraic equations is then covered with emphasis on orthogonal collocation on

finite elements. Lastly some methods on the solution of mixed-integer nonlinear problems are discussed, since this is the form of the resulting optimization superstructure.

Chapter 3

The optimization superstructure for a dynamically operable nonlinear process with allowance for input saturation is developed. A simple example which demonstrates the applications is then presented.

Chapter 4

An algorithm that reduces the number integer variables representing saturation in linear problems to only those which are required, is presented. The proof of the algorithm is discussed with illustrations and its applicability is extended to mixed-integer nonlinear problems. The progression of the algorithm towards the global optimum of a mixed-integer linear problem is detailed with illustrations from an example problem.

Chapter 5

Two case studies in the design of dynamically operable plants are investigated. The first is a simple continuous stirred tank reactor and the second is a binary distillation column. The results for the two examples are presented and compared to the corresponding case without rigorous saturation handling.

Chapter 2

Literature Review

2.1 Introduction

Dynamic operability reflects the quality with which a plant can be controlled using feedback and is a function of both the design of the plant and the associated control system. A plant should not be designed solely on the basis of steady state considerations because this would cause the process to exhibit poor dynamic characteristics leading to a decrease in economic performance and a reduced ability to effectively handle safety and environmental constraints. This motivates the need for the development of quantitative methods to assess dynamic operability and the inclusion of these assessment techniques into procedures for process design.

Historically, plants have been designed in a sequential manner, where the process is first designed and then a control system is synthesized to control the plant. The problem with this approach is that it is often found that the expected level of performance cannot be achieved because of operational difficulties. When this occurs, sophisticated control technology is then needed to correct operability problems that could have been moderated by a modification of the original design developed during the early design phase (Downs and Doss, 1991).

Although the research community has been aware of this phenomenon since the early 1940's when it was explicitly stated by Ziegler and Nichols (1943), it is only since the early 1980s that methods have been developed to measure dynamic operability. However, few of these methods are able to simultaneously assess the impact of performance-limiting factors such as nonminimum phase characteristics,

input constraints, and model uncertainty. One group of operability assessment techniques that is able to simultaneously assess these factors are optimization-based methods. Section 2.2 summarizes a selection of dynamic operability assessment methods and then details recent work in the field of optimization-based operability assessment and design.

Optimization-based methods require a mathematical model of the process under study, but it is only recently that a mathematical model was formulated that is able to describe the nonlinearity caused by the input saturation of proportional integral controllers. The formulation is detailed in Section 2.3 and the surrounding theory on input saturation is briefly discussed. The formulation handles input saturation discontinuities by the introduction of slack variables and their inclusion in either mixed-integer or bilinear constraints, resulting in either a mixed-integer or nonlinear programming problem respectively. Solving mixed-integer nonlinear optimization problems requires more sophisticated techniques than those used for linear programming problems, therefore Section 2.4 discusses current methods which are available to do so.

Any mathematical model of a dynamic process will consist of a set of dynamic algebraic equations that cannot be directly handled using conventional solvers, therefore special techniques are required to represent equations of this nature in a computationally tractable form. Methods of handling these equations are discussed briefly in Section 2.5 with emphasis on orthogonal collocation on finite elements.

2.2 Dynamic Operability

Morari (1983) defined dynamic operability as a measure of the quality of closed-loop control performance that can be achieved for a plant by means of feedback. This phenomenon has also been referred to as (dynamic) resiliency or controllability (Ross, 1997).

A typical feedback loop consists of a controller, G_c, providing input to a process, G, which then outputs a measured signal which is compared to a setpoint, y_{set}, and the error fed back as an input to the controller as shown in Figure 2.1.

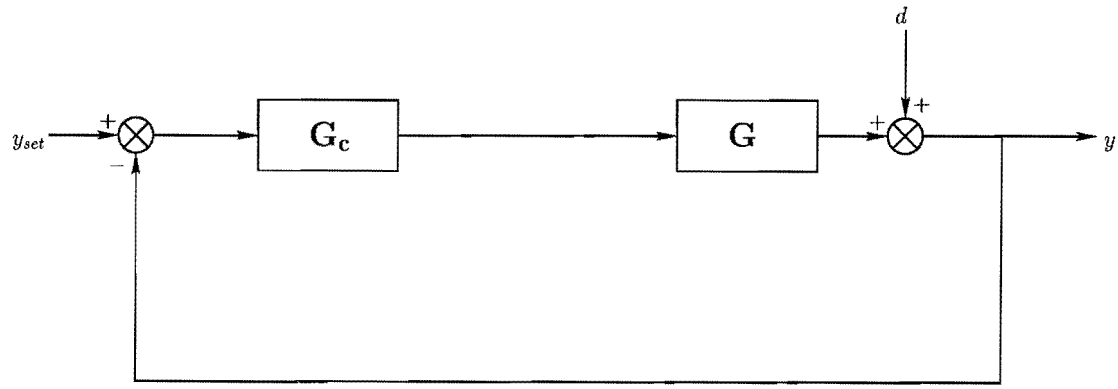


Figure 2.1: Block diagram of classical feedback framework

The quality of the closed-loop performance depends on the type of controller used, the controller tuning parameters, the variables chosen as the manipulated inputs and regulated outputs, and the plant structure and model itself. The latter becomes especially significant when model uncertainty is taken into account (Ross, 1997). Since the performance quality depends so greatly on the controller chosen (structure, tuning parameters and choice of variables) Morari (1983) proposed that dynamic operability should be calculated as a *limit* of the achievable closed-loop performance. This ensures that the measure of performance is only dependent on the inherent properties of the process and the controller structure.

It has long been known that some plants perform better than others when the same controller structure is used; these plants are referred to as being more controllable. Dynamic operability is an attempt to quantify and measure this phenomenon so that during the design phase a plant which is intrinsically easier to control can be chosen. The technique is also able to assess the controllability of a currently existing design thereby evaluating the degree of improvement that can be achieved if an alternative scheme is used. As an illustration of the potential uses of including dynamic operability assessment at the design stage, Ross and Swartz (1995) performed a study on the choice of layout for a three cell-bank flotation circuit.

Quantitative measures to assess dynamic operability have mainly been developed since the early 1980s (Ross, 1997), and the next section will attempt to summarize the important findings of some of these studies.

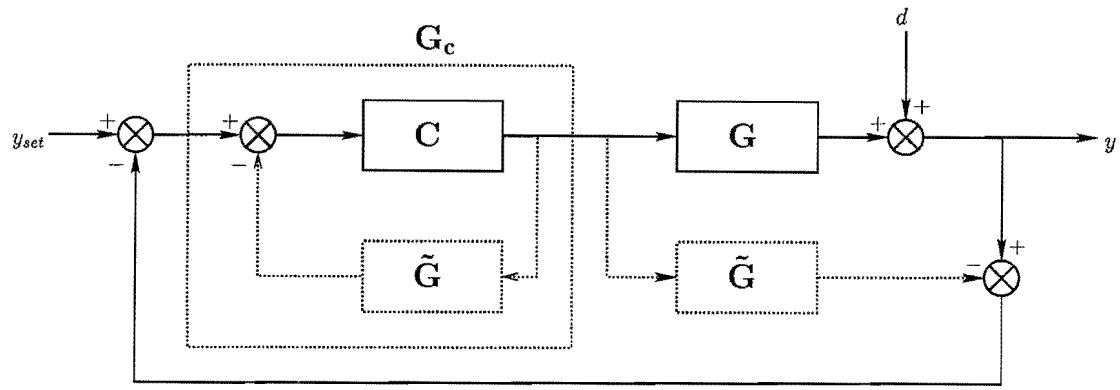


Figure 2.2: Block diagram of IMC framework

2.2.1 Internal Model Control Framework

Morari (1983) proposed the use of the internal model control (IMC) framework to assess the closed-loop properties of a system independently of its control structure. In doing so, he isolated factors inherent in processes which are the cause of operability problems.

As can be seen in Figure 2.2, the IMC framework is set up by the addition of two blocks containing the plant model $\widetilde{G}$ around the controller C and the actual plant G. The blocks are added such that the effect of each cancels out the other. An important result of this manipulation is that if the model is assumed to be perfect then the closed-loop system will be stable if and only if the plant and IMC controller, G_c, are stable. In addition, a controller that achieves perfect control of the plant can be found by calculating the right inverse of the plant model. Therefore, in order for G_c to be implementable it must be causal, realizable and stable. Morari used this argument to analytically prove the following results:

- Right half plane transmission (RHPT) zeros prevent the achievement of perfect control since they would result in unstable poles when the plant is inverted to obtain the controller, G_c.
- Process models with time delays cannot be perfectly controlled because the inversion of the process to obtain G_c results in a controller that requires prediction (non-causal behavior).

- Input constraints limit the extent to which the IMC controller G_c can be approximated by the inverse of the plant model while still remaining proper, thereby limiting the achievable closed-loop performance.

- One of the key derivations of the IMC framework is that if the model of the system is perfect, then the closed-loop system is stable if and only if the IMC controller and plant are stable. Therefore, model uncertainty limits the achievable performance because the above derivation relies on the assumption of a perfect model. It is also possible to show, using the IMC framework, that model uncertainty limits the frequency range over which perfect control can be achieved.

Morari and co-workers performed further studies on the effects that the above properties have on the dynamic operability of a plant (Skogestad and Morari, 1987; Holt and Morari, 1985a,b). However, a shortcoming of this approach is that the effects of the various factors can only be studied separately. They also analytically derived dynamic operability measures, however these methods can only be applied to processes of a particular structure, which limits the applicability of the method.

2.2.2 Controllability Analysis Framework

This is an alternative to the IMC approach and was developed by Perkins and co-workers (Cao *et al.*, 1994; Russell and Perkins, 1987; Perkins and Wong, 1985). The approach is based on the the concept of functional controllability which was first proposed by Rosenbrock (1970). Once again the inversion of the plant model was used to analyze which factors prohibit perfect control.

The concept of functional controllability states that given any output trajectory, y, that is zero for $t < 0$ and satisfies certain smoothness criteria, there will exist an input trajectory, u, that is able to generate y, provided that the state vector $x(0) = 0$.

A system with a polynomial transfer function matrix $G(s)$ is functionally controllable if and only if $G(s)$ is invertible, i.e. non-singular. This is readily apparent since given a particular output trajectory it is possible to calculate the controlled input response by multiplying it with the inverse of the plant.

This theory also established that perfect control is limited by the presence of RHPT zeros, dead-time, input constraints and model uncertainty. Further work by Perkins and coworkers studied the effect that these properties have on the performance, but the detailed results will not be reflected here (Cao *et al.*, 1994; Russell and Perkins, 1987; Perkins and Wong, 1985).

2.2.3 Psarris and Floudas

Psarris and Floudas (1991a,b) focused their studies on multivariable systems containing both RHPT zeros and dead-time. The problem is approached by the realization that time-delays play a non-intuitive role in multivariable systems. This is because of the following competing effects:

- Time delays limit the minimum time before the effect of input action is seen in the outputs.
- Because of the transcendental nature of the characteristic equation, they can result in an infinite number of RHPT zeros.
- It is possible to reduce the RHPT zeros to a finite number for systems with an infinite number of RHPT zeros by increasing the time delays of certain elements.

Psarris and Floudas (1991a,b) developed a strategy for identifying systems with infinite RHPT zeros and developed formulae for the computation thereof without the need to solve for the existence of these explicitly. Having found a means to locate the RHPT zeros, they attempted to study the performance-limiting effects of RHPT zeros and dead-time simultaneously.

The approach gives greater insight into the behavior of time-delayed systems but does not provide a clear method with which to analyze the benefits and trade-offs of reducing the number of RHPT zeros by increasing the time delays. Furthermore, there is no actual guarantee that a system with a reduced number of RHPT zeros will behave any better than a system with an infinite number of RHPT zeros. Also, they do not examine the effect that input constraints and model uncertainty would have on the controllability of the plant.

2.2.4 Open Loop Measures of Dynamic Operability

A number of open-loop measures have been developed for the approximate analysis of dynamic operability. For example:

- measures derived from the singular value,
- the disturbance condition number,
- frequency-dependent relative gain array,
- performance relative gain array,
- closed-loop disturbance gain and relative disturbance gain,
- the relative order matrix, and
- the disturbance cost.

Skogestad *et al.* (1991) define open loop measures of dynamic operability as theoretical and heuristic arguments to derive measures of dynamic operability with the aim of reaching results consistent with those achieved by more rigorous analyses of the problem.

What follows is a brief description of what some of the measures indicate.

Singular Value Analysis

Cao *et al.* (1996) used the inherent properties of processes to predict the closed-loop performance. They investigated the effects of the following process properties:

- non-minimum phase behavior.
- manipulated variable constraints, (MVCs).
- process/model mismatch.

They focused their investigation on the effect of MVCs on controllability using a singular value analysis technique. This method provides a means to quantify the limitations imposed on the achievable performance when the process is subjected to input constraints.

A shortcoming of this method is that it requires a linear model, which means that a nonlinear model will have to linearized around a particular operating point in order for this method to be applied. Cao *et al.* (1996) propose an alternative technique that utilizes an optimization strategy and does not require linearization to analyze the effect of MVCs on the controllability of the system.

Robust Performance Number

The use of the robust performance number (RPN) and the robust performance number of a plant (RPPN) set to measure the controllability of a system was first proposed by Trierweiler and Engell (1997). The RPN is a measure of how difficult it might be for a system to achieve the desired performance robustly, while the RPPN is sensitive to nonlinearity and plant uncertainty. Small differences of the RPPN and the RPN indicate that the system has small sensitivity to time invariant uncertainties (Trierweiler and Engell, 1997).

The RPN depends on the attainable closed-loop performance and because of this it automatically takes into account the effect of non-minimum phase behavior and frequency-dependent system directionality. They also describe how other performance-limiting factors can be taken into account in their 1997 paper.

The Frequency-Dependent RGA

The relative gain array was originally proposed by Bristol in 1966. It was initially applied at steady state only, but was later extended to higher frequencies. The relative gain is the ratio of the "open loop" and "closed-loop" gains.

Hovd and Skogestad (1992) used the RGA to analyze the effects of the following factors:

- *Right half plane zeros.* It was shown that if an RGA element changes sign as ω goes from 0 to ∞ then RHP zeros or RHPT zeros are present in the system. However, this is a only a sufficient condition and not a necessary condition, therefore RHP zeros could be present even if the RGA elements do not change sign.

- *Individual element uncertainty.* It was shown that perturbations in the individual elements of the RGA could lead to the transfer function matrix becoming singular, with stability repercussions for element uncertainty, process identification and state matrix uncertainty.

- *Diagonal input uncertainty.* If the system has large RGA elements and an inverse based-controller is used, then the overall system will be sensitive to input uncertainty.

The Performance Relative Gain Array (PRGA)

Since a triangular $G(s)$ results in the RGA being an identity matrix, a shortcoming of the RGA is that in these cases it may indicate that interaction can be ignored when one-way coupling might actually be significant. Therefore Hovd and Skogestad (1992) introduced the performance relative gain array (PRGA) to circumvent this drawback.

Although the diagonal elements of the RGA and PRGA are identical, the PRGA does not have all the algebraic properties of the RGA. The PRGA must be recomputed whenever G is rearranged, whereas the RGA only needs to be rearranged in the same way that G has been arranged. In addition, the PRGA is independent of input scaling, but depends on output scaling.

They then proceeded to define the closed-loop disturbance gain (CLDG) matrix. Plotting the magnitudes of the individual elements of the CLDG matrix provides information about which disturbances the plant will find difficult to reject. Similarly, plotting the magnitudes of the individual elements of the PRGA will indicate which setpoints the plant will adjust with difficulty.

Relative Order Matrices

Soroush (1994) showed that the relative orders of a process could be used to determine the best achievable closed-loop response for each controlled output. For square, multivariable, nonlinear discrete-time systems with the sampled-data representation of the form:

$$\begin{cases} x(k+1) &= \Phi(x(k), u(k), d(k)) \\ y(k) &= h(x(k)) \end{cases} \tag{2.1}$$

where $x = [x_1 \dots x_n]^T$, $u = [u_1 \dots u_m]^T$, $y = [y_1 \dots y_m]^T$, and $d = [d_1 \dots d_p]^T$ denote the vectors of state variables, manipulated inputs, controlled outputs, and measurable disturbance inputs respectively.

The relative order of the output y_i with respect to the manipulated input u_j is the smallest integer r_{ij} for which:

$$\left[\frac{\partial h_i(x)}{\partial x}\right]\left[\frac{\partial \Phi(x,u,d)}{\partial x}\right]^{r_{ij}-1}\left[\frac{\partial \Phi(x,u,d)}{\partial u_j}\right] \not\equiv 0$$

If such and integer does not exist, then $r_{ij} = \infty$.

The relative order matrix for a process with a model of the form described in Equation (2.1) is then the $m \times m$ matrix:

$$M_r = \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1m} \\ r_{21} & r_{22} & \dots & r_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ r_{m1} & r_{m2} & \dots & r_{mm} \end{bmatrix}$$

He noted that it may not always be feasible to induce the best closed-loop response in all the outputs and at the same time achieve complete input-output decoupling. He used theoretical arguments to show how the relative order matrix could be used to determine when a completely decoupled response is feasible.

The Disturbance Cost

The disturbance cost (DC) is a resiliency screening method originally proposed by Lewin and Bogle (1996) and Weitz and Lewin (1996). This measure accounts for the degree to which the controlled process is insensitive to disturbances. Essentially, the method assumes that the system has perfect control and then measures the integral square error of the control action that satisfies this assumption. The technique can also be extended to handle parametric uncertainty, but was mainly intended to be a tool for the rapid screening of designs at the design stage.

2.2.5 Optimization-Based Approaches

These approaches use mathematical optimization routines to assess dynamic operability and synthesize dynamically operable plants. The advantages of these methods over heuristic and open-loop methods are listed below.

- It is possible to examine the effects of differing performance-limiting factors simultaneously.
- It is a simple matter to express the problems in terms of the economics, thus demonstrating the benefits of good control in readily understandable terms.
- It is possible to include constraints on the behavior of the plant, and explicitly state what the measure of "good" performance should be.

In addition to being used for the assessment of dynamic operability, the optimization approach can also be extended to integrate the design of the controller and process structure with the simultaneous assessment of feasilibility or controllability constraints. In this case the mathematical formulation leads to a mixed-integer nonlinear programming (MINLP) problem of the general form:

$$\min_{u,x,y} \quad \Phi\left(z,u,x,y,d,p\right) \tag{2.2}$$

subject to

$$\begin{aligned}
\dot{z} &= f(z, u, x, y, d, p) \\
z(0) &= z_0 \\
h(z, u, x, y, d, p) &= 0 \\
g(z, u, x, y, d, p) &\leq 0 \\
z &\in Z \\
u &\in U \\
x &\in X \\
d &\in D \\
p &\in P \\
y &\in \{0, 1\}
\end{aligned}$$

where z is the set of state variables, u represents the degrees of freedom, x represents the set of continuous decision variables, d is the set of measurable disturbances, p is the set of model parameters, and y is a set of binary variables that is used to denote the existence of certain structures. The objective is to minimize the cost function, Φ, subject to the set of dynamic equations, f, equality constraints, h, and inequality constraints, g.

Integrated Design and Control

Mohideen *et al.* (1996) present a unified process synthesis framework for obtaining process designs and control systems. The optimization framework ensures that the process designs generated will be economically optimal and still be able to cope with variations in the process caused by disturbances and parametric uncertainty.

The method employs a dynamic mathematical model which consists of:

- path and end point constraints,
- a model for the description of the parametric uncertainty and disturbances, and
- a set of process design and control system alternatives.

In order to fully describe the time-varying behavior of a process which involves parametric uncertainty and disturbances, the following should be considered:

- feasible operation in the presence of any possible parametric uncertainty or disturbances, and
- control scheme definition.

The feasibility requirement for the general system described by the constraints in Equation (2.2) can be stated as follows: *Ensure that, for any possible profile of* p *and* $d(t)$*, there exist profiles for* x *and* $u(t)$ *such that the constraints in Equation (2.2) are satisfied at any time,* $t \in [0, t_f]$(Mohideen *et al.*, 1996).

This can also be expressed mathematically as:

$$\max_{\substack{p \in P \\ d(t) \in D(t)}} \min_{\substack{x \in X \\ u(t) \in U(t)}} \max_{\substack{k \in K \\ t \in [0, t_f]}} g_k \left[z(t), u(t), x, y, d(t), p \right] \leq 0 \tag{2.3}$$

$$\begin{aligned} \dot{z} &= f(z, u, x, y, d, p) \\ z(0) &= z_0 \\ h(z, u, x, y, d, p) &= 0 \end{aligned}$$

where

$$\begin{aligned} P &= \left\{ p, p^L \leq p \leq p^U \right\} \\ X &= \left\{ x, x^L \leq x \leq x^U \right\} \\ D(t) &= \left\{ d(t), d(t)^L \leq d(t) \leq d(t)^U \right\} \\ U(t) &= \left\{ u(t), u(t)^L \leq u(t) \leq u(t)^U \right\} \end{aligned}$$

Mohideen *et al.* (1996) introduce a set of constraints which employ binary variables in order to define the existence of a suitable control scheme. The constraints are mathematically able to describe the set of alternative choices for controlled variable / manipulated variable pairings. This effectively is able to find an optimal controller which can guarantee stability for any disturbance or parameter uncertainty within the given range.

They also include in the economic-based cost function, a term that considers the cost of the associated control structure. The objective function is the minimization of an expectation term that takes into account the contribution that the possible

realization of all the parametric uncertainties and disturbances would have on the cost.

The resultant optimization problem is an infinite dimensional stochastic MINLP. The direct solution of these problems is tedious and there are three main difficulties that need to be addressed in order to deal with the resulting problem.

1. The profiles of the states, manipulated variables, uncertain parameters and disturbances represent an infinite number of decision variables, resulting in a semi-infinite problem.

2. The objective function employs an expectation term over an optimization problem.

3. A max-min-max optimization subproblem representing the feasibility requirements appears as part of the constraint space.

They then propose that the problem can be solved by first converting the differential equations to algebraic residual equations by means of orthogonal collocation on finite elements, which effectively reduces the infinite space to a finite (stochastic) mixed integer nonlinear problem. Then, by approximating the parametric uncertainty and disturbances by a finite number of scenarios, the expectancy term can be replaced by a weighted summation term over the specified set of periods and the max-min-max operation can be removed, provided that the inequalities in g are enforced over all periods.

However, instead of arbitrarily selecting periods, Mohideen *et al.* (1996) introduce an explicit time-varying feasibility analysis step to identify additional critical parameters and disturbances that may still violate the inequality constraints. For a fixed design and control structure, such a feasibility analysis step corresponds to the solution of the problem by means of an active set strategy.

The work of Bahri *et al.* (1996) is similar to that of Mohideen *et al.* (1996). It considers the effect of parametric uncertainty and disturbances in control structure selection and shows a method whereby an optimization strategy can be used to assess the flexibility and controllability of an existing plant for the purposes of assessment or retrofit design. They then explain how the assessment method

can be incorporated into an optimization strategy for the purposes of integrated process and control system design

The resulting integrated design optimization framework is similar to that of Mohideen *et al.* (1996), but the latter's formulation is more comprehensive since it allows for disturbances and parametric uncertainty with a non-uniform distribution. In addition, parametric uncertainty is dealt with directly, as opposed to treating these as slowly varying disturbances, as was the case in Bahri *et al.* (1996).

Perkins and Walsh (1996) perform a study of controllability analysis by assessing the effect of time delays and bounded parametric uncertainty on the disturbance rejection capabilities of a plant. They accomplish this by analyzing the performance of an idealized controller under worst-case conditions.

The controller is idealized since it is assumed to act perfectly as soon as knowledge of any disturbance becomes available. In general, the response of any controller to a disturbance is never immediate because of time delays in the system. If the idealized controller performs inadequately, then, because the controller is perfect, by inference the inadequacy is caused by a limiting factor of the process itself. Optimization is required for this technique in order to determine the worst-case disturbance from the set of possible disturbances.

Recent work at Imperial College in this field includes application to industrial scale operations, new theory for mixed-integer dynamic optimization and the design of processes under uncertainty (Bansal *et al.*, 2000).

Limit of Achievable Performance

The work of Ross and Swartz (Ross, 1997; Ross and Swartz, 1997; Swartz, 1996; Ross and Swartz, 1995), Chenery and Walsh (1998), and Webers and Engell (1996) employ Q-parametrization, an approach proposed by Boyd *et al.* (1990, 1988), for the study of dynamic operability and synthesis of plant layouts.

The method parametrizes the closed-loop mapping of the generalized feedback structure with a free stable parameter, Q. Essentially, the parametrization covers the set of all possible linear controllers thereby allowing the formulation of an

optimization problem that is able to select the best possible linear controller for the process. The resulting controller can be implemented, unlike the open loop approaches of Cao *et al.* (1996), and can be used as a limit of the attainable performance.

Webers and Engell (1996) uses Q-parametrization to evaluate the differences in attainable performance through the use of centralized and decentralized control structures, whereas Swartz (1996) and Chenery and Walsh (1998) use it to analyze controllability. The work of Chenery and Walsh (1998) follows on that of Swartz (1996), but proposes an alternative formulation for the computation of Q.

An important distinction between the work of Chenery and Walsh (1998) and Swartz (1996) and that of Bahri *et al.* (1996) and Mohideen *et al.* (1996), is that a linear optimization problem is generated. This means that a globally optimal solution is guaranteed, which implies that if no solution is found then none exists.

2.3 Input Saturation

All physical control systems have to deal with limitations on the controlled input. For example, a valve controlling the flowrate of cooling water to a reactor can only operate between being fully open or completely closed. The actuator is said to *saturate* at an upper or lower bound when the controller attempts to push the actuator beyond these limits. The result is that there is a discrepancy between the input to the process, $\widetilde{u}$, and the controller output, u, shown in Figure 2.3. This appears on the control input trajectory as though the input is "hugging" the bound. Thus a linear, time invariant plant with an otherwise linear controller might experience control input nonlinearities if there are constraints on the actuator.

Any controller with relatively slow or unstable modes will experience windup problems if there are actuator constraints (Doyle *et al.*, 1987). Windup is then interpreted as an inconsistency between the plant input and the states of the controller when the control signal saturates (Zheng *et al.*, 1994). Kothare *et al.* (1994) presented a generalized framework for the study of anti-windup designs, which was later used by Mulder *et al.* (2000, 1999) to arrive at a formulation that was able

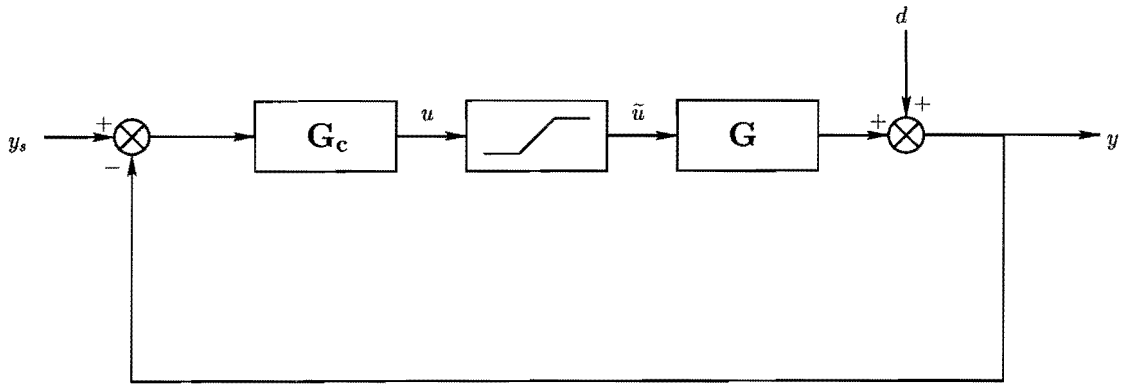


Figure 2.3: Block diagram showing the effects of input saturation

to design the optimal anti-windup bumpless transfer (AWBT) compensator for a given controller.

The input seen by the process for a proportional integral controller shown in Equation (2.4) can be represented by the three states as shown in Equation (2.5).

$$u = K_c \left(\varepsilon + \frac{1}{\tau_I} \int \varepsilon \, dt \right) \tag{2.4}$$

$$\begin{aligned} \tilde{u} &= \text{sat}\,(\text{u}) \\ &= \begin{cases} u_{\min} & \text{if } u < u_{\min} \\ u & \text{if } u_{\min} < u < u_{\max} \\ u_{\max} & \text{if } u > u_{\max} \end{cases} \end{aligned} \tag{2.5}$$

In the case of a proportional integral controller shown in Equation (2.4); if the error term in the standard control equations remains positive for a substantial time, then the control signal will become saturated at the upper limit. If after this saturation the error continues to remain positive, the integral term will accumulate the error and the signal will become "more" saturated and the actuator will appear to hug the upper limit. If the error then becomes negative, the controller will not respond by immediately changing the input proportionally, but will instead continue to act at the limit until the error remains negative for the amount of time necessary to reduce the integral term sufficiently to allow the proportional term to dominate. This nonlinearity is typically dealt with by the inclusion of anti-reset windup measures, or through the use of the velocity form of the PI

equation. However, although the work by Mulder *et al.* (2000, 1999) allows the optimal compensator to be designed for a particular controller, the formulation does not describe the nonlinear behavior mathematically so that it can be included as part of an optimization formulation.

Closed-loop optimization problems that attempt to take actuator limits into account by merely including constraints on the upper and lower bounds of the actuator are generally not taking input saturation into account. In the optimization strategy, the actuator signal would be calculated based on a linear relationship between the error and the manipulated variable change, thus the input action will merely touch the limit and then move away from the limit. This would occur even if hugging the constraint would result in an improvement in the objective function. Most classical feedback analyses that involve optimization are based on the assumption that the closed-loop transfer function is linear, because no mathematical formulation was available to describe this nonlinearity. However, Young and Swartz (1997) recently arrived at two mathematical formulations for input saturation that can be used in an optimization strategy when studying structured controllers. They are:

- a bilinear approach, and
- a mixed-integer programming formulation.

2.3.1 The Bilinear Approach

The formulation is derived in the discrete time domain for a constant discrete time interval, Δt. The closed-loop responses for the regulated outputs and the actuator inputs are put into vectors of length P, where a particular element in the vector represents the signal value at the corresponding time instant (Young and Swartz, 1997).

$$\begin{aligned} \mathbf{y} &= (y(1), y(2), \ldots, y(P))^T \\ \mathbf{u}_a &= (u_a(1), u_a(2), \ldots, u_a(P))^T \end{aligned}$$

The velocity form of the controller expresses the changes in the controller output with respect to the position of the actuator at the previous time step, and is used

to take into account the consequences of integral wind-up as follows:

$$\Delta u_c(k) = u_c(k) - u_a(k-1)$$

$$\Delta u_c(k) = K_c\left(\varepsilon(k) - \varepsilon(k-1) + \frac{\Delta t}{\tau_I}\varepsilon(k)\right)$$

Saturation causes a discrepancy between the controller signal, $u_c(k)$, and the actuator position, $u_a(k)$, which can be expressed as an equality through the addition of slack variables for the upper, $S_U(k)$, and lower, $S_L(k)$, saturation limits at the corresponding time interval, k.

$$u_c(k) = u_a(k) - S_L(k) + S_U(k) \tag{2.6}$$

Therefore the relationship between the slack variables, the control signal and the actuator input can be expressed as follows:

$$u_a(k) = \begin{cases} u_L & \text{for} \quad u_L > u_c(k) & S_L(k) > 0,\ S_U(k) = 0 \\ u_c(k) & \text{for} \quad u_L \leq u_c(k) \leq u_U & S_L(k) = 0,\ S_U(k) = 0 \\ u_U & \text{for} \quad u_U < u_c(k) & S_L(k) = 0,\ S_U(k) > 0 \end{cases} \tag{2.7}$$

Thus the set of constraints that model closed-loop dynamic behavior can be written as (Young and Swartz, 1997):

$$\begin{aligned} u_a(k) - S_L(k) + S_U(k) &= u_a(k-1) + K_c\left(\varepsilon(k) - \varepsilon(k-1) + \frac{\Delta t}{\tau_I}\varepsilon(k)\right) \\ S_L(k)\left(u_a(k) - u_L\right) &= 0 \\ S_U(k)\left(u_a(k) - u_U\right) &= 0 \\ u_L &\leq u_a(k) \leq u_U \\ 0 &\leq S_L(k) \\ 0 &\leq S_U(k) \end{aligned}$$

It has been shown that by including nonlinear inequalities as constraints automatically implies that the resulting optimization formulation will be nonlinear and thus there is the likelihood that the formulation will be nonconvex (Peressini *et al.*, 1988). This means that the solution to the problem might encounter convergence

difficulties and, more importantly, the solution need not necessarily be the global optimum. Thus this formulation has a number of disadvantages and this motivates the need for an alternative solution which does not pose these problems. Young and Swartz (1997) accomplish this by developing a mixed-integer linear programming formulation to rigorously handle saturation.

2.3.2 The Mixed Integer Linear Programming Formulation

The constraints of Equation (2.7) are logical constraints, therefore it is possible to express these constraints by defining binary variables $z_L(k)$ and $z_U(k)$ at each time interval, k, such that $(z_L(k), z_U(k) \in Z \in [0,1])$. Thus the constraints of Equation (2.8) force the slack variables to be zero unless either one of the saturation limits is active (Young and Swartz, 1997).

$$\begin{array}{rcccl} 0 & \leq & S_L(k) & \leq & \beta z_L(k) \\ 0 & \leq & S_U(k) & \leq & \beta z_U(k) \end{array} \tag{2.8}$$

Here β is a constant positive scalar that must be sufficiently large so that it does not constrain the value of the slack variables when one of the saturation limits is active.

The set of constraints listed in Equation (2.9) force the actuator to be positioned at the appropriate limit if one of the saturation limits is active. If the bounds are not active then the difference between the actuator position and the saturation bound will not be constrained (Young and Swartz, 1997).

$$\begin{array}{rcccl} u_L - \beta(1 - z_L(k)) & \leq & u_a(k) & \leq & u_L + \beta(1 - z_L(k)) \\ u_U - \beta(1 - z_U(k)) & \leq & u_a(k) & \leq & u_U + \beta(1 - z_U(k)) \end{array} \tag{2.9}$$

Thus the constraints for the new formulation can be written as:

$$
\begin{aligned}
u_a(k) - S_L(k) + S_U(k) &= u_a(k-1) + K_c\left(\varepsilon(k) - \varepsilon(k) + \frac{\Delta t}{\tau_I}\varepsilon(k)\right) \\
u_L - \beta(1 - z_L(k)) &\leq u_a(k) \leq u_L + \beta(1 - z_L(k)) \\
u_U - \beta(1 - z_U(k)) &\leq u_a(k) \leq u_U + \beta(1 - z_U(k)) \\
0 &\leq S_L(k) \leq \beta z_L(k) \\
0 &\leq S_U(k) \leq \beta z_U(k) \\
u_L &\leq u_a(k) \leq u_U
\end{aligned}
$$

This formulation corresponds to a mixed-integer linear problem where the decision variables now include the integer variables. The mixed-integer formulation, while not as easy to solve as a quadratic programming problem, at least does not result in a nonconvex search space.

2.4 Solving MINLP Problems

Mixed-integer nonlinear programming problems occur when integer variables are used together with continuous variables in a nonlinear problem. For example, integer variables can be used to model sequences of events and the existence or nonexistence of units, while the continuous variables model the processes taking place inside the units. As a result, MINLPs arise frequently in optimization-based process synthesis or design and have a wide range of applications. The nonlinearity of a problem can arise from the product of two integer variables, the product of continuous variables only, or the product of discrete and continuous variables.

The solution of MINLP problems faces two major challenges. The first is that as the number of integer variables increases, the number of possible combinations increases exponentially. If the discrete variables are binary i.e. either equal to zero or one, then the number of possible combinations is equal to 2^n, where n is the number of binary variables. In addition to this, nonlinear problems are generally nonconvex leading to the existence of multiple local optima, which imply that a globally optimal solution cannot be guaranteed.

2.4.1 MINLP optimization algorithms

Floudas (1995) gives a list of MINLP algorithms that have been developed to solve MINLP optimization problems.

1. Generalized Benders Decomposition.
2. Branch and Bound.
3. Outer Approximation.
4. Feasibility Approach.
5. Outer Approximation with Equality Relaxation.
6. Outer Approximation with Equality Relaxation and Augmented penalty.
7. Generalized Outer Approximation.
8. Generalized Cross Decomposition.

A brief summary of the various methods and their applicability follows.

Generalized Benders Decomposition

At each iteration an upper and lower bound are generated on the solution to the MINLP model. The upper bound is the result of solving a primal problem where the integer variables have been fixed and provides information on the Lagrange multipliers for the equality and inequality constraints. The lower bound corresponds to the the solution of a master problem that provides the set of values at which the integer variables are to be fixed in the primal problem. As the iterations proceed the upper and lower bounds converge on the answer to within a finite error tolerance in a finite number of iterations.

This method can only be used if it can be shown that the variables can be split up into two groups such that one set of mixed continuous and integer variables is a non-empty, convex set, and the objective function and inequality constraints are convex for each fixed integer variable in the other set. Furthermore, the equality

equations must be linear with respect to each fixed integer variable in the second set.

The method has a number of other variations and corresponding restrictions that are detailed in Floudas (1995).

Outer Approximation

This method is similar to the GBD in that it generates an upper and lower bound that converge on the solution. In fact, it solves a sub-class of the problem that can be handled by the GBD method. It assumes that the continuous and integer variables are separable, that the problem is linear with respect to the integer variables, and that nonlinear equalities can be eliminated algebraically or numerically. Through these assumptions the master problem is based on an outer linearization of the objective function and constraints around the solution of the primal problem.

The principal advantage of using this method is that it requires fewer iterations than the GBD method, though this does not necessarily imply that it is faster, since it adds more constraints to the problem as the iterations proceed.

Outer Approximation with Equality Relaxation

This method is able to handle nonlinear equality constraints by relaxing them into inequalities and then applying the outer approximation algorithm to the relaxed problem.

Outer Approximation with Equality Relaxation and Augmented Penalty

This method attempts to avoid the limitations imposed by the convexity assumptions in the OA/ER algorithm by introducing a penalty function. By not imposing the convexity assumptions the relaxed inequalities might not equate to the original equalities, the linearizations may not be valid starting points, and the master problem may not provide a valid lower bound on the solution. This implies that

there is a possibility that part of the feasible region with candidate integer solutions may be cut off and only sub-optimal solutions will be identified.

There is no guarantee that the algorithm will not cut off part of the feasible region, but the augmented penalty function attempts to reduce the size of the feasible region that might be cut off.

It should be borne in mind that this study is concerned with the design of dynamically operable plants in general and thus no assumptions can be made concerning the convexity of the equalities and constraints since these will be unique to the particular problem. Accordingly, this algorithm was used in this study to demonstrate the principles and objectives behind the research even though there is no assurance that the solution found is a global optimum. However, if this problem formulation is known then a more appropriate algorithm that guarantees a global optimum should be used. The OA/ER/AP method will now be covered in more detail.

The algorithm is able to address MINLPs formulated as:

$$\min_{\mathbf{x},\mathbf{y}} \quad c^T\mathbf{y} + f(\mathbf{x}) \tag{2.10}$$

subject to the constraints

$$\begin{aligned} \mathbf{h}(\mathbf{x}) &= \mathbf{0} \\ \mathbf{g}(\mathbf{x}) &\leq \mathbf{0} \\ C\mathbf{x} + B\mathbf{y} &\leq \mathbf{d} \\ \mathbf{x} \in \mathbf{X} &= \{\mathbf{x} : \mathbf{x} \in \Re^n, A_1\mathbf{x} \leq a_1\} \subseteq \Re^n \\ \mathbf{y} \in \mathbf{Y} &= \{\mathbf{y} : \mathbf{y} \in \{0,1\}^q, A_2\mathbf{y} \leq a_2\} \end{aligned}$$

under the conditions that:

- f, $\mathbf{h}$, and $\mathbf{g}$ are continuously differentiable,
- a constraint qualification holds at the solution of every nonlinear programming problem resulting from Equation (2.10) when $\mathbf{y}$ is fixed.

The relaxed master problem used in the OA/ER/AP algorithm is formulated as:

$$Z_L^K = \min \quad c^T\mathbf{y} + \mu + \sum_k w_k^o s_k^o + \sum_{i,k} w_{i,k}^p p_{i,k} + \sum_{i,k} w_{i,k}^q q_{i,k} \tag{2.11}$$

subject to the constraints

$$\left.\begin{array}{rcl} \mu + s_k^o & \geq & f\left(\mathbf{x}^k\right) + \bigtriangledown f\left(\mathbf{x}^k\right)\left(\mathbf{x} - \mathbf{x}^k\right) \\ \mathbf{p}_k & \geq & \mathbf{g}\left(\mathbf{x}^k\right) + \bigtriangledown \mathbf{g}\left(\mathbf{x}^k\right)\left(\mathbf{x} - \mathbf{x}^k\right) \\ \mathbf{q}_k & \geq & T^k\left[\mathbf{h}\left(\mathbf{x}^k\right) + \bigtriangledown \mathbf{h}\left(\mathbf{x}^k\right)\left(\mathbf{x} - \mathbf{x}^k\right)\right] \end{array}\right\} \quad k = 1, 2, ..., K$$

$$\begin{array}{rcll} C\mathbf{x} + B\mathbf{y} & \leq & \mathbf{d} & \\ \mathbf{x} \in \mathbf{X} & = & \{\mathbf{x} : \mathbf{x} \in \Re^n, A_1\mathbf{x} \leq a_1\} \subseteq \Re^n & \\ \mathbf{y} \in \mathbf{Y} & = & \{\mathbf{y} : \mathbf{y} \in \{0-1\}^q, A_2\mathbf{y} \leq a_2\} & \\ \sum\limits_{i \in \mathbf{B}^k} \mathbf{y}_i - \sum\limits_{i \in \mathbf{NB}^k} \mathbf{y}_i & \leq & \left|B^k\right| - 1 & k = 1, 2, ..., K-1 \\ s_k^o, p_{i,k}, q_{i,k} & \geq & 0 & k = 1, 2, ..., K \end{array}$$

where

$\mathbf{p}_k = \{p_{i,k}\}$ is a vector of positive slack variables

$\mathbf{q}_k = \{q_{i,k}\}$ is a vector of positive slack variables

s_k^o are positive slack scalars

w_k^o is the weight on the slack variable s_k^o

$w_{i,k}^p$ is the weight on the slack variable vector $p_{i,k}$,

$w_{i,k}^q$ is the weight on the slack variable vector $q_{i,k}$, and

T^k is a diagonal matrix $m \times m$ with elements t_{ii}^k defined as:

$$t_{ii}^k = \left\{\begin{array}{rl} -1 & \text{if } \lambda_i^k < 0 \\ +1 & \text{if } \lambda_i^k > 0 \\ 0 & \text{if } \lambda_i^k = 0 \end{array}\right\} \quad i = 1, 2, ..., m$$

In addition, the weights must satisfy the following constraints:

$$\begin{array}{rcl} w_k^o & > & \left|\overline{\mu}_k\right| \\ w_{i,k}^p & > & \left|\lambda_{i,k}\right| \\ w_{i,k}^q & > & \left|\mu_{i,k}\right| \end{array}$$

where $\overline{\mu}_k$, $\lambda_{i,k}$, $\mu_{i,k}$ are the Lagrange multipliers of the primal problem for $y = y^k$:

$$\min_{\mathbf{x}} \quad c^T \mathbf{y}^k + \mu \tag{2.12}$$

subject to

$$\begin{aligned} f\left(\mathbf{x}^k\right) - \mu &\leq 0 \leftarrow \overline{\mu}_k \\ \mathbf{h}\left(\mathbf{x}\right) &= \mathbf{0} \leftarrow \lambda_k \\ \mathbf{g}\left(\mathbf{x}\right) &\leq \mathbf{0} \leftarrow \mu_k \\ C\mathbf{x} + B\mathbf{y}^k &\leq \mathbf{d} \end{aligned}$$

$$\mathbf{x} \in \mathbf{X} = \{\mathbf{x} : \mathbf{x} \in \Re^n, A_1\mathbf{x} \leq a_1\} \subseteq \Re^n$$

The OA/ER/AP algorithm will now be stated.

1. Solve the NLP relaxation of Equation (2.10). In other words, treat the integer variables, y, as being continuous with the condition that $0 \leq y \leq 1$ to obtain an initial solution (x^0, y^0). If y^0 is an 0-1 combination, then terminate the search, else proceed to step 2.

2. Solve the relaxed MIP master problem in Equation (2.11) to identify a new integer combination, y^1.

3. Fix the binary variables $y = y^1$ and solve the primal problem in Equation (2.12) to find the upper bound as well as the Lagrange multipliers. Let the solution correspond to (x^1, y^1).

4. Define the $(m \times m)$ matrix T^1.

5. Solve the relaxed MIP master problem to determine y^2 and the lower bound, Z_L^1, for Equation (2.10).

6. Repeat steps 3 and 4 until there is an increase in the optimal value of the feasible NLP problem and then exit the algorithm.

Note that there is no theoretical reason why the algorithm should terminate at step 6 and that this is merely a heuristic (GAMS Development Corporation, 1999; Floudas, 1995).

The OA/ER/AP algorithm has been implemented in an MINLP solver called DICOPT, which is able to make use of independent mixed-integer and nonlinear

solvers to solve MINLPs. DICOPT is also able to interface with GAMS, a general algebraic modeling language. The case studies investigated made use of the OA/ER/AP algorithm using DICOPT as a solver and GAMS as the modeler.

The OA/ER/AP algorithm used in DICOPT has two alternative stopping rules. The first continues until an MIP master problem becomes infeasible, unless an iteration or time limit is reached first. The second rule terminates the search when the bound defined by the objective of the last MIP master problem is worse than the best NLP solution found. The latter gives the global optimum if the problem is convex, provided the weights are sufficiently large (GAMS Development Corporation, 1999).

The next section discusses the subject of convexity since the choice of MINLP algorithm is heavily dependent on the convex nature of the particular problem under investigation.

2.4.2 Convexity

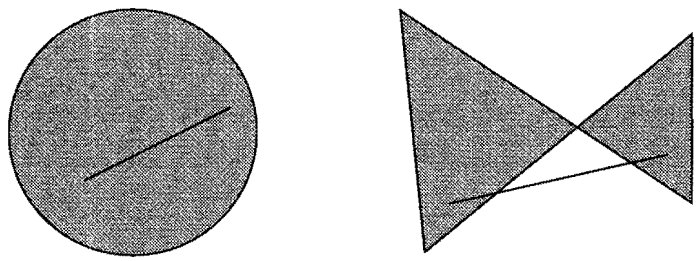
The advantage of modeling a convex problem formulation is that there are numerous optimization methods that are able to guarantee that the solution is the global optimum. In order to facilitate the identification of convex problem formulations, and analyze methods of computation time reduction, the basic definitions and properties of convex sets and functions will be summarized.

Basic Definitions

Convex Set. A set $S \in \Re^n$ is *convex* if for any two points $\mathbf{x}_1$ and $\mathbf{x}_2$ of set S, the closed line segment, $(1-\lambda)\,\mathbf{x}_1 + \lambda \mathbf{x}_2$, belongs to the set S for each λ where $0 \leq \lambda \leq 1$.

An illustration of convex and nonconvex sets is given in Figure 2.4. Some examples of convex sets are:

- open and closed lines,



Convex set Nonconvex set

Figure 2.4: Convex and nonconvex sets

- open and closed half-spaces, i.e. $\{\mathbf{x}|\mathbf{c}^t\mathbf{x} \leq z, \mathbf{x} \in \Re^n\}$ where $\mathbf{c} \in \Re^n$, $\mathbf{c} \neq 0$, and $z \in \Re$
- all points inside and on a circle.

If S_1 and S_2 are convex sets in $\Re^n$ then the following are true:

1. the intersection of $S_1 \cap S_2$ is a convex set.
2. the sum of two convex sets, $S_1 + S_2$, is a convex set.
3. the product of the real number θ and set S_1, θS_1, is a convex set.

In addition to convex sets, there exist convex and concave functions.

Convex Function. Let S be a convex subset of $\Re^n$, and $f(\mathbf{x})$ be a real valued function defined on S. The function is termed convex if for any $\mathbf{x}_1, \mathbf{x}_2 \in S$, and $0 \leq \lambda \leq 1$, the following condition applies:

$$f[(1-\lambda)\mathbf{x}_1 + \lambda\mathbf{x}_2] \leq (1-\lambda) f(\mathbf{x}_1) + \lambda f(\mathbf{x}_2)$$

A strictly convex function is defined similarly, except the condition is a strict inequality.

Concave Function. Let S be a convex subset of $\Re^n$, and $f(\mathbf{x})$ be a real valued function defined on S. The function is termed concave if for any $\mathbf{x}_1, \mathbf{x}_2 \in S$, and $0 \leq \lambda \leq 1$, the following condition applies:

$$f\left[(1-\lambda)\,\mathbf{x}_1+\lambda\mathbf{x}_2\right]\geq(1-\lambda)\,f\left(\mathbf{x}_1\right)+\lambda f\left(\mathbf{x}_2\right)$$

A strictly concave function is defined similarly, except the condition is a strict inequality.

Note that the function $f(\mathbf{x})$ is concave on S if and only if $-f(\mathbf{x})$ is convex on S. This implies that results for convex functions can be modified into results for concave functions by multiplication with -1. The reverse is also true.

A graphical example highlighting the differences between convex, concave and nonconvex functions has been depicted in Figure 2.5.

Convex functions can be combined to produce new convex functions, for example:

1. Let $f(\mathbf{x}),...,f_n(\mathbf{x})$ be convex functions on a convex subset $S\in\Re^n$. Then the sum of the functions:

$$f_1(\mathbf{x})+...+f_n(\mathbf{x})$$

   is convex. If at least one $f_i(\mathbf{x})$ is strictly convex on S, then their sum is strictly convex.

2. Let $f(\mathbf{x})$ be convex (strictly convex) on a convex subset of $S\in\Re^n$ and $\lambda>0$, then $\lambda f(\mathbf{x})$ is convex (strictly convex).

3. Let $f(\mathbf{x})$ be convex (strictly convex) on a convex subset of $S\in\Re^n$, and $g(\mathbf{y})$ be an increasing convex function defined on the range of $f(\mathbf{x})$ in $\Re$. Then the composite function $g[f(\mathbf{x})]$ is convex (strictly convex) on S.

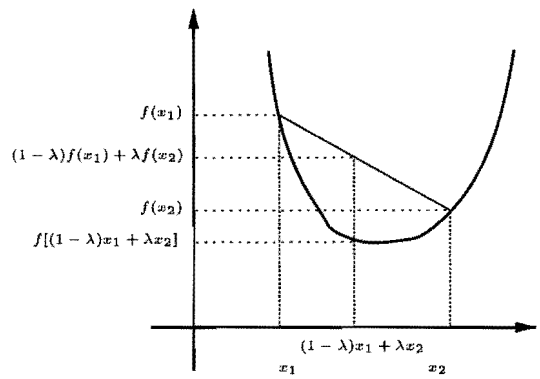
4. Let $f_1(\mathbf{x}),...,f_n(\mathbf{x})$ be convex functions and bounded from above on a convex subset $S\in\Re^n$. Then the point-wise supremum function:

$$f(\mathbf{x})=\max\left\{f_1(\mathbf{x}),...,f_n(\mathbf{x})\right\}$$

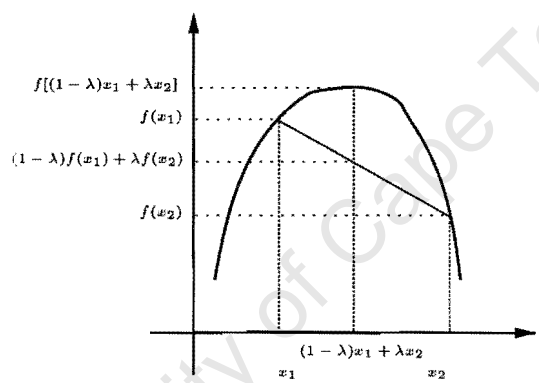
   is a convex function on S.

5. Let $f_1(\mathbf{x}),...,f_n(\mathbf{x})$ be concave functions and bounded from below on a concave subset $S\in\Re^n$. Then the point-wise infimum function:

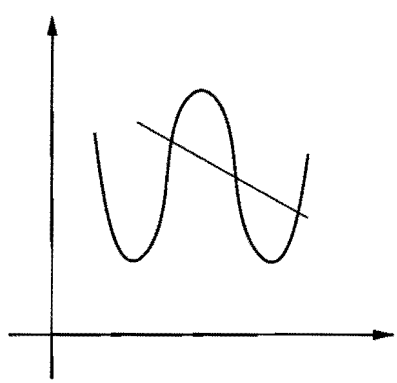
$$f(\mathbf{x})=\min\left\{f_1(\mathbf{x}),...,f_n(\mathbf{x})\right\}$$



(a) Convex function



(b) Concave function



(c) Nonconvex function

Figure 2.5: Convex, concave and nonconvex functions

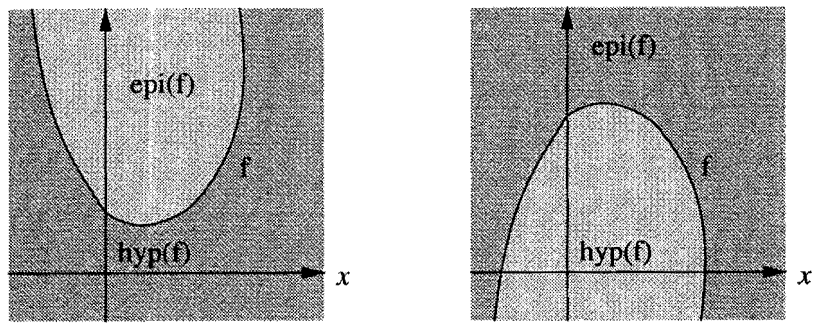


Figure 2.6: Epigraph and hypograph of a function

is a concave function on S.

Epigraph of a function. Let S be a non-empty set in $\Re^n$. The epigraph of a function $f(\mathbf{x})$, denoted by epi (f), is a subset of $\Re^{n+1}$ defined as the set of $(n+1)$ vectors $(\mathbf{x}, \mathbf{y})$:

$$\{(\mathbf{x}, \mathbf{y}) : f(\mathbf{x}) \leq \mathbf{y}, \mathbf{x} \in S, \mathbf{y} \in \Re\}$$

Hypograph of a function. The hypograph of $f(\mathbf{x})$, denoted by hyp (f), is a subset of $\Re^{n+1}$ defined as the set of $(n+1)$ vectors $(\mathbf{x}, \mathbf{y})$:

$$\{(\mathbf{x}, \mathbf{y}) : f(\mathbf{x}) \geq \mathbf{y}, \mathbf{x} \in S, \mathbf{y} \in \Re\}$$

Theorem *Let S be a non-empty set in $\Re^n$. The function $f(\mathbf{x})$ is convex if and only if epi(f) is a convex set.*

Note that the epigraph of a convex function and the hypograph of a concave function will therefore be convex sets (Floudas, 1995).

Figure 2.6 illustrates the epigraph and hypograph of convex and concave functions.

However, if it is not readily apparent from the algebraic form of a function that it is convex, then in practice it is not always easy to use the theorem to prove convexity, however the advantages of a convex formulation are considerable.

2.5 Dynamic Optimization

Two of the major aspects of plant or process design are the selection of an economically optimal steady state operating point and ensuring that the dynamic operation of the plant is able to deal with deviations from this point. The operating conditions of the plant will typically vary either because of disturbances in the process, such as changes in the feed composition, or to operate at a new steady state, perhaps because of changes in demand for the product. The variations from the optimal steady state used to be taken into account by means of over-design, but it has been shown that this can actually cause the process to become less controllable (Downs and Doss, 1991).

This necessitated the need for a method whereby the dynamics of the plant in response to changes in operation could be analyzed at the design stage. With the advent of computers, the simulation of process responses finally became feasible and the trend is to now research methods of including the dynamics as part of an optimization problem during the design.

This requires that a mathematical model of the process be available for inclusion in the optimization problem, where the model will consist of dynamic mass and energy balances. The result is a set of differential and algebraic equations or DAEs.

This results in a general dynamic optimization problem of the form

$$\min \quad F\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right),\mathbf{y},\mathbf{p},t_f\right) \tag{2.13}$$

subject to

$$\begin{aligned}
\dot{x} &= \mathbf{f}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right),\mathbf{y},\mathbf{p}\right)\\
\mathbf{x}_0 &= \mathbf{x}\left(0\right)\\
\mathbf{h}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right),\mathbf{y},\mathbf{p}\right) &= \mathbf{0}\\
\mathbf{g}\left(\mathbf{x}\left(t\right),\mathbf{u}\left(t\right),\mathbf{y},\mathbf{p}\right) &\leq \mathbf{0}\\
\mathbf{x}^{\mathbf{L}} &\leq \mathbf{x}\left(t\right) \leq \mathbf{x}^{\mathbf{U}}\\
\mathbf{u}^{\mathbf{L}} &\leq \mathbf{u}\left(t\right) \leq \mathbf{u}^{\mathbf{U}}\\
\mathbf{y}^{\mathbf{L}} &\leq \mathbf{y} \leq \mathbf{y}^{\mathbf{U}}
\end{aligned}$$

where

t is the independent variable, time

t_f is the final time

$\mathbf{x}(t)$ is a vector of state variables, $\mathbf{x}(t) \in \Re^{\mathcal{X}}$

$\mathbf{x}_0$ is a vector of initial conditions for the boundary value problem

$\mathbf{u}(t)$ is a vector of the control profiles to be specified, $\mathbf{u}(t) \in \Re^{\mathcal{U}}$

$\mathbf{y}$ is a decision variable vector, $y \in \Re^{\mathcal{Y}}$

$\mathbf{p}$ is a parameter vector that is not a function of time

F is a scalar valued objective function

$\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})$ is the vector of state equations, $\mathbf{f}(t) \in \Re^{\mathcal{X}}$

$\mathbf{h}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})$ is a vector of equality constraints

$\mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), \mathbf{p})$ is a vector of inequality constraints.

Unlike the unified framework proposed by Mohideen *et al.* (1996), Equation (2.13) assumes that p is fixed, thus assuming that the parametric uncertainty can be ignored.

However, while it is reasonably straightforward to simulate a process response, it is not so simple to optimize a design which includes the response as part of the optimization problem itself, especially if there are constraints on the response path, such as input or safety constraints. The resultant optimization problem cannot be solved directly by conventional nonlinear programming techniques.

It is not possible to use NLP solvers directly nor impose bounds or general restrictions on constraints involving the state or input variables because the profiles are continuous, which leads to a semi-infinite programming (SIP) problem. While algorithms for solving SIPs do exist, they are not as advanced as nonlinear programming methods. Moreover, they are more difficult to implement. Optimal control methods are able to handle continuous control profiles but normally cannot manage general algebraic equality and inequality constraints that are unrelated to the profile without tedious numerical solution of the necessary conditions (Cuthrell and Biegler, 1989).

Key solution methods for problems of this type are listed below.

1. Iterative dynamic programming.

2. Coupling a dedicated differential algebraic equation solver to an optimization routine. Here, the differential equation solver integrates the model in an inner loop while the the outer loops performs the optimization of the process. This approach is termed *sequential*. The disadvantage of this approach is that by not being able to apply path constraints to the response there is the possibility that an unfortunate initial guess for the controller parameters can lead to the response becoming unstable and the optimization procedure halts without a valid solution.

3. Discretizing the profile and treating the variables as decisions variables can require repeated, potentially expensive, solution of the model and would probably require the inclusion of a number of sensitivity test equations to ensure accuracy of the solution. However, unlike the sequential method, provided constraints like open-loop stability are met, it is possible to only include the manipulated variables (semi-inclusive) as decision variables and the solution will not "blow up". Taking a fully inclusive approach (manipulated and controlled variables included as decision variables) allows this method to be applied to open loop unstable processes as well (Bloss *et al.*, 1999; Barton *et al.*, 1998; Cuthrell and Biegler, 1989).

4. Orthogonal collocation on finite elements (OCFE), which is a sub-category of the simultaneous approaches mentioned previously, converts a set of differential equations to a set of algebraic equations by means of discretization where orthogonal collocation is applied to the choice of finite elements. There are a number of additional numerical benefits associated with applying OCFE and hence the implementation of orthogonal collocation on finite elements will be discussed next (Cuthrell and Biegler, 1989).

2.5.1 Orthogonal Collocation

Orthogonal collocation on finite elements is a stable and accurate method of numerically discretizing a set of ordinary differential equations by means of a polynomial approximation of the time varying profiles (Cuthrell and Biegler, 1989).

The state and control variable profiles are approximated by the piecewise polynomials over each finite element, α_i, as follows:

for $\alpha_i \leq t_i \leq \alpha_{i+1}$

$$\mathbf{x}(t) = \sum_{j=0}^{\text{nCOL}} \mathbf{x}_{ij}\phi_j(t)$$

$$\mathbf{u}(t) = \sum_{j=0}^{\text{nCOL}} \mathbf{u}_{ij}\psi_j(t)$$

where

$$\phi_j(t) = \prod_{\substack{n=0 \\ n\neq j}}^{\text{nCOL}} \frac{(t - t_{in})}{(t_{ij} - t_{in})}$$

$$\psi_j(t) = \prod_{\substack{n=1 \\ n\neq j}}^{\text{nCOL}} \frac{(t - t_{in})}{(t_{ij} - t_{in})}$$

t_{ij} is the time at the jth collocation point for finite element i

$\mathbf{x}_{ij}$ is the vector of piecewise polynomial coefficients at finite element i and collocation point j for the state variables

$\mathbf{u}_{ij}$ is the vector of piecewise polynomial coefficients at finite element i and collocation point j for the manipulated variables

$\phi_j(t)$ polynomial basis functions for the states

$\Psi_j(t)$ polynomial basis functions for the manipulated variables

nFE is the number of finite elements, α

nCOL is the number of collocation points

A diagram illustrating the difference between collocation points and finite elements is shown in Figure 2.7.

Substituting the above piecewise polynomials into the differential equations and solving exactly yields the state coefficients at each collocation point. This is sufficient to solve general dynamic algebraic optimization problems provided that all the profiles are smooth. If this is the case, then choosing a sufficiently large number of finite elements will yield an accurate solution.

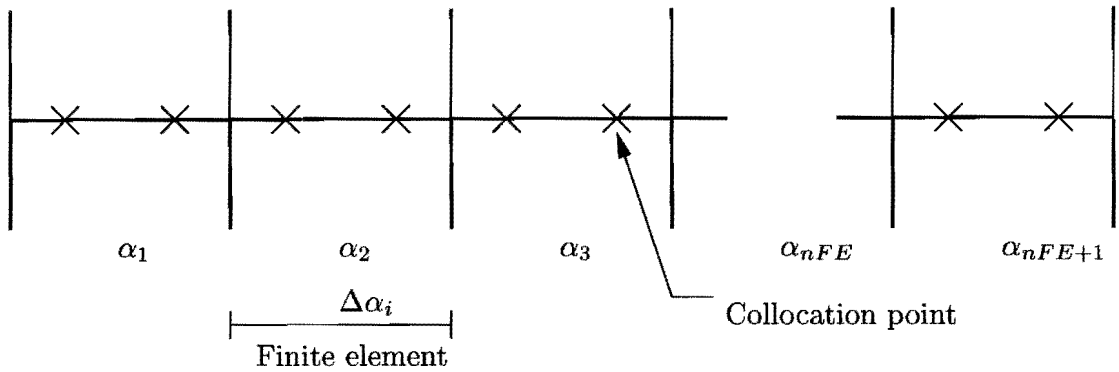


Figure 2.7: Collocation on finite elements

Cuthrell and Biegler (1989) propose an alternative formulation that includes the size of the finite element at time i as part of the decision variables, and places the elements at the best points to minimize the approximation error of the state variable profiles.

If the relative positions of the points in each element are constant, then the basis functions can be rewritten as:

for $0 \leq \tau \leq 1$

$$\begin{aligned} t &= \alpha_i + \tau\left(\alpha_{i+1} - \alpha_i\right) \\ t_{ij} &= \alpha_i + \tau_j\left(\alpha_{i+1} - \alpha_i\right) \end{aligned}$$

thus

$$\begin{aligned} \phi_j(t) = \phi_j(\tau) &= \prod_{\substack{n=0 \\ n\neq j}}^{\text{nCOL}} \frac{(\tau - \tau_n)}{(\tau_j - \tau_n)} \\ \psi_j(t) = \psi_j(\tau) &= \prod_{\substack{n=1 \\ n\neq j}}^{\text{nCOL}} \frac{(\tau - \tau_n)}{(\tau_j - \tau_n)} \end{aligned}$$

The polynomial representations are in their Lagrange form and have the following properties:

- $\phi_j(t_{ij}) = 1$
- $\phi_n(t_{ij}) = 0 \qquad n \neq j$
- $\mathbf{x}(t_{ij}) = \sum_{n=0}^{\text{nCOL}} \mathbf{x}_{in}\phi_n(\tau_j) = \mathbf{x}_{ij} \qquad i=1,\ldots,\text{nFE}; j=0,\ldots,\text{nCOL}$

The state coefficients are determined by substituting the piecewise polynomials into the differential equations and solving these exactly for a given number of collocation points. The residual equations are written as:

$$\begin{aligned}\dot{\mathbf{x}}(t_{ij}) - \mathbf{f}(\mathbf{x}(t_{ij}), \mathbf{u}(t_{ij}), \mathbf{p}) &= \sum_{n=0}^{\text{nCOL}} \mathbf{x}_{in}\dot{\phi}_n(t_{ij}) - \mathbf{f}(\mathbf{x}_{ij}, \mathbf{u}_{ij}, \mathbf{p}) \\ &= \sum_{n=0}^{\text{nCOL}} \mathbf{x}_{in}\frac{\dot{\phi}_n(\tau_j)}{\Delta\alpha_i} - \mathbf{f}(\mathbf{x}_{ij}, \mathbf{u}_{ij}, \mathbf{p}) \\ &= \mathbf{0}\end{aligned}$$

$$\Rightarrow \sum_{n=0}^{\text{nCOL}} \mathbf{x}_{in}\dot{\phi}_n(\tau_j) - \mathbf{f}(\mathbf{x}_{ij}, \mathbf{u}_{ij}, \mathbf{p})\Delta\alpha_i = \mathbf{0} \tag{2.14}$$

for $i = 1, ..., \text{nFE}$; $j = 1, ..., \text{nCOL}$.

The term $\frac{\dot{\phi}_n(\tau_j)}{\Delta\alpha_i}$ is the simplification of $\dot{\phi}_n(t_{ij})$ and is obtained by applying the chain rule to the derivatives. The expression $\dot{\phi}_n(\tau_j)$ can be easily calculated offline since it depends only on the Legendre root polynomials, see Villadsen and Michelsen (1978) (Cuthrell and Biegler, 1989).

To ensure that the state profiles are continuous across finite elements the following continuity constraints that equate the state variable at the beginning of element i to the state variable at the end of the previous element $i-1$ are included in the formulation as follows:

$$\mathbf{x}_{i0} = \sum_{n=0}^{\text{nCOL}} \mathbf{x}_{i-1,n}\phi_n(\tau = 1)$$

Thus the general collocation form of the new dynamic optimization problem is the following nonlinear programming problem:

$$\min \quad F(\mathbf{x}_{ij}, \mathbf{u}_{ij}, \mathbf{p}, t_f) \tag{2.15}$$

subject to

$$\sum_{n=0}^{\text{nCOL}} \mathbf{x}_{ij}\dot{\phi}_n(\tau_j) - \mathbf{f}(\mathbf{x}_{ij}, \mathbf{u}_{ij}, \mathbf{p})\Delta\alpha_i = \mathbf{0} \qquad i = 1, ..., \text{nFE}$$

$$j = 1, ..., \text{nCOL}$$

$$\sum_{n=0}^{\text{nCOL}} \mathbf{x}_{i-1,n}\phi_n(\tau = 1) = \mathbf{x}_{i0} \qquad i = 2, ..., \text{nFE}$$

$$\mathbf{x}_{10} = \mathbf{x}(0)$$

$$\mathbf{h}(\mathbf{x}_{ij}, \mathbf{u}_{ij}, \mathbf{p}) = \mathbf{0}$$

$$\mathbf{g}(\mathbf{x}_{ij}, \mathbf{u}_{ij}, \mathbf{p}) \leq \mathbf{0}$$

$$\Delta\alpha_i \geq 0$$

$$\sum_{i=1}^{\text{nFE}} \Delta\alpha_i = t_f$$

The resulting formulation is a nonlinear programming problem for which a number of commercial solvers exist.

Chapter 3

Input Saturation in Optimization-Based Methods

In the literature review in Section 2.2 it was discussed why it was preferable to design a dynamically operable plant using a technique that involves optimization. In this chapter a general mathematical formulation will be presented and discussed that can be used to design an economically optimal process subject to design and performance constraints. In addition, the formulation is able to take into account the phenomenon of input saturation and it is shown in a subsequent chapter that doing so results in a design that lies closer to the economic optimum for a steady state design than a dynamically operable design that does not allow for input saturation.

3.1 Problem Formulation

The optimization-based method for the design of a dynamically operable plant requires an objective function, a set of decision variables, and a mathematical formulation of the process, composed of constraints. The constraints consist of a set of equalities, which must be met exactly, and inequalities, which describe physical constraints on the process. This results in the following mathematical superstructure:

$$\min_{\mathbf{z},\mathbf{y},\mathbf{u},\mathbf{x},\mathbf{Z}} \quad \Phi\left(\mathbf{z},\mathbf{y},\mathbf{u},\mathbf{x},\mathbf{Z},\mathbf{p},t_f\right) \tag{3.1}$$

subject to

$$\begin{aligned}
\dot{\mathbf{z}} &= \mathbf{f}(\mathbf{z}, \mathbf{y}, \mathbf{u}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t) \\
\mathbf{z}_0 &= \mathbf{z}(\mathbf{0}) \\
\mathbf{h}(\mathbf{z}, \mathbf{y}, \mathbf{u}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t) &= \mathbf{0} \\
\mathbf{g}(\mathbf{z}, \mathbf{y}, \mathbf{u}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t) &\leq \mathbf{0} \\
\mathbf{z} &\in \Re^{\mathcal{K}} \\
\mathbf{y} &\in \Re^{\mathcal{Y}} \\
\mathbf{u} &\in \Re^{\mathcal{U}} \\
\mathbf{x} &\in \Re^{\mathcal{X}} \\
\mathbf{p} &\in \Re^{\mathcal{P}} \\
\mathbf{Z} &\in \{0,1\}^{\mathcal{Z}}
\end{aligned}$$

where

$\mathbf{z}(t)$	is a vector of $\mathcal{K}$ state variables
$\mathbf{y}(t)$	is a vector of $\mathcal{Y}$ measured (controlled) variables
$\mathbf{u}(t)$	is a vector of $\mathcal{U}$ input variables
$\mathbf{x}$	is a vector of $\mathcal{X}$ continuous variables
$\mathbf{Z}$	is a vector of $\mathcal{Z}$ binary variables
$\mathbf{p}$	is a vector of $\mathcal{P}$ uncertain parameters and disturbances
Φ	is a scalar-valued objective function
$\mathbf{f}$	is a set of $\mathcal{K}$ dynamic algebraic equations
$\mathbf{h}$	is a set of equality constraints, and
$\mathbf{g}$	is a set of inequality constraints.

The study assumes a fixed set of parameters that do not vary, and considers only a single disturbance. However, the formulation can be extended to include parametric uncertainty and multiple disturbances.

The set of continuous variables can be used to represent molar flowrates and compositions, steady state values and structural design parameters such as diameter or height, while binary variables are generally used to indicate the existence or nonexistence of units, thereby providing the means to create a plant superstructure that is able to represent all possible unit configurations.

3.2 The Optimization Superstructure

3.2.1 Objective Function

For the purposes of this study it is assumed that the objective is a scalar-valued function that is used to evaluate the effectiveness of a design in realizing a particular goal. In this light, the objective function can be represented by cost and profit functions, controllability and flexibility measures, the integral square error of a control trajectory, or even the approximation error of a problem. For the rest of the study it is assumed that the objective function will need to be minimized. Since the investigation is concerned with a process design at the design stage, an economic-based objective function will be used in the case studies since it makes it readily apparent how allowing for rigorous input saturation handling in the design affects the estimated cost of a project.

The objective function is in future denoted by:

$$\min \quad \Phi\left(\mathbf{z}, \mathbf{y}, \mathbf{u}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t_f\right)$$

3.2.2 Dynamic Algebraic Equations

Describing a dynamic chemical process mathematically entails the solution of a number of mass and energy balances. However, these derivatives describe continuous profiles resulting in a semi-infinite problem which cannot be easily solved using conventional optimization techniques. Fortunately, it is possible to accurately approximate the dynamic equations using the technique of orthogonal collocation on finite elements, as was discussed in Section 2.5. The result is a discretized, finite

dimensional problem that can be solved using an ordinary nonlinear programming solver.

First it is necessary to decide on a final time horizon for the state equations, t_f. Then the horizon is broken up into a series of control elements, k, which in turn are broken up into finite elements, δ_i, such that the total number of finite elements per control element is nFE. All finite elements are assumed to have the same number of collocation points, nCOL, within them.

The process state equations and dynamics can then be approximated using orthogonal collocation on finite elements by solving exactly for the residual Equation (3.2) using the formulation proposed by (Cuthrell and Biegler, 1989) in Equation (2.14):

$$\sum_{n=1}^{\text{nCOL}} \mathbf{z}_{k,in}\dot{\phi}_n(\gamma_j) = \mathbf{f}(\mathbf{z}_{k,ij}, \mathbf{y}_{k,ij}, \mathbf{u}_{k,ij}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, \mathbf{t}_{k,ij})\,\Delta\delta_{k,i} \quad \begin{matrix} k = 1, ..., \text{nCE} \\ i = 1, ..., \text{nFE} \\ j = 1, ..., \text{nCOL} \end{matrix} \tag{3.2}$$

$$\mathbf{z}(t=0) = \mathbf{z}_0 \tag{3.3}$$

where

$$\phi_j(\gamma) = \prod_{\substack{n=0 \\ n \neq j}}^{\text{nCOL}} \frac{(\gamma - \gamma_n)}{(\gamma_j - \gamma_n)}$$

for $0 \leq \gamma \leq 1$ such that

$$\begin{aligned} t &= \delta_{k,i} + \gamma(\delta_{k,i+1} - \delta_{k,i}) \\ t_{k,ij} &= \delta_{k,i} + \gamma_j(\delta_{k,i+1} - \delta_{k,i}) \end{aligned}$$

$\dot{\phi}_n(\gamma_j)$ can be calculated offline as described in Section 2.5.

The initial values of the states, $\mathbf{z}_0$, are equivalent to the optimal steady state operating conditions and should be included as decision variables in the optimization problem. Accordingly, the set of initial value Equations (3.3) should be included in the equality constraints.

In the formulation proposed by Cuthrell and Biegler (1989), the size of each finite element was included as part of the optimization problem. In practice, however, it is not simple to apply and, for the rest of the study, only finite elements of a fixed size are used. Thus there is no need to keep track of the size of each finite element and Equation (3.2) becomes:

$$\sum_{n=1}^{\mathrm{nCOL}} \mathbf{z}_{k,in}\dot{\phi}_n(\gamma_j) - \mathbf{f}(\mathbf{z}_{k,ij}, \mathbf{y}_{k,ij}, \mathbf{u}_{k,ij}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, \mathbf{t}_{k,ij})\,\Delta\delta = 0$$

The formulation proposed by Cuthrell and Biegler (1989) is based on the principle that the finite elements are chosen at the points were input action is most required. However, for the case of a process which is controlled by feedback, the controller can generally only act at specific intervals, which correspond to the controller time step. In selecting a step size, one aspect of the control system is specified and this becomes a problem parameter denoting a particular set of control systems with that time step. Thus the finite element size is not made variable in this study, and the controlled input is assumed to remain constant over each sampling period. However, this does not necessarily imply that the control elements are equivalent to the finite elements since more than one finite element might be needed to adequately describe the state profiles between control moves.

In the formulation described in Equation (2.15) the controlled input trajectory is a dynamic variable which is dependent on the state and is calculated using OCFE. Here, however, the control trajectory is already a discretized formulation and there is no need to apply OCFE, since it can be calculated exactly using conventional discrete methods. Thus there is no need for the control profile to satisfy conditions of smoothness or continuity since these requirements are only necessary if the profile is to be approximated using collocation.

However, to ensure that the state profiles are accurately approximated it is necessary to include the following constraints on the continuity of the states between finite elements, Equation (3.4), and between control elements, Equation (3.5).

$$\mathbf{x}_{k,i0} = \sum_{n=0}^{\text{nCOL}} \mathbf{x}_{k,(i-1),n}\phi_n\left(\gamma = 1\right) \qquad k = 1, ..., \text{nCE} \qquad (3.4)$$

$$i = 2, ..., \text{nFE}$$

$$\mathbf{x}_{k,10} = \sum_{n=0}^{\text{nCOL}} \mathbf{x}_{(k-1),\text{nFE},n}\phi_n\left(\gamma = 1\right) \qquad k = 2, ..., \text{nCE} \qquad (3.5)$$

Another advantage of using collocation on finite elements is that it handles nonlinear equations directly and does not require linearization of the dynamics. Since most process models are to some extent derived from first principles, which generally yields nonlinear expressions, this is a desirable property for any method used to handle DAEs. The case studies investigated by Young and Swartz (1997) are derived from transfer function descriptions of the models and thus their formulation does not have to deal with nonlinearities in the dynamics.

3.2.3 Equality Constraints

General equalities are taken into account by the set of equality constraints:

$$\mathbf{h}\left(\mathbf{z}, \mathbf{y}, \mathbf{u}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t\right) = \mathbf{0}$$

This section considers general equality constraints that pertain to all PI controlled processes.

Steady State Constraints

When performing an optimization-based design that utilizes dynamics, it is important to include the steady state balances as part of the equality constraints. These provide an optimal operating point as well as assigning values to the initial states for the dynamic equations. Steady state balances are denoted by:

$$\mathbf{f}\left(\mathbf{z_0}, \mathbf{y_{ss}}, \mathbf{u_{ss}}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t_0\right) = \mathbf{0}$$

As mentioned previously, the steady state values are included as part of the design variables (decision variables).

The Control Equation

This study investigates the design of dynamically operable plants controlled using a standard proportional integral controller which has been optimally tuned for the process. This entails that the controller gains and time constants be included as decision variables in the optimization problem. These are denoted by $\mathbf{K_c^{yu}}$ and $\tau_\mathbf{I}^{yu}$ respectively, where yu refers to the pairing of the controlled variable y with manipulated input u. In addition, it is possible to include the control loop pairing yu as part of the constraint set to determine the best pairing through the inclusion of binary variables indicating whether or not a controlled variable is paired with a particular input as is the case in Schweiger and Floudas (1998).

It is preferable to separate the control element size from the finite element size in order for the formulation to more closely reflect reality. In this study the control time element is assumed to be the time super element and thus the time horizon is first divided up into nCE control elements. Thereafter, each control element is subdivided into constant, equally sized finite elements such that there are nFE finite elements per control element. Accordingly, nCOL collocation points per finite element are assumed to be sufficient to approximate the control profile between finite elements. The control element is distinguished from the finite element by representing it with the letter k. The finite elements are numbered from the start of the control element as shown in Figure 3.1. Alternatively, using GAMS as an algebraic modeler, this can be accomplished using a remainder function to implement the control action only when nFE finite elements are completed. Therefore control elements do not need to be defined and the finite element becomes the time super element. This removes the need for continuity equations that maintain continuity across the control elements for the state profiles.

The standard PI control equation is:

$$u = u_{ss} + K_c \left(\varepsilon(t) + \frac{1}{\tau_I} \int \varepsilon(t)\, dt \right) \tag{3.6}$$

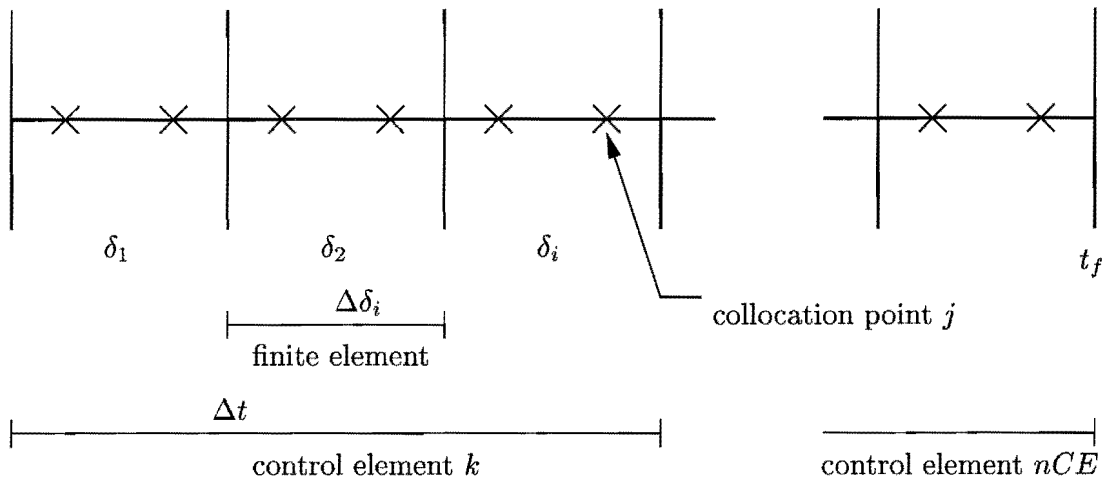


Figure 3.1: Collocation on finite elements across control elements

where

$$\varepsilon(t) = y_{ss} - y(t) \tag{3.7}$$

Once the problem has been discretized and extended to multiple inputs, Equation (3.6) becomes:

$$\mathbf{u}_{k,10} = \mathbf{u}_{ss} + \mathbb{K}_c \left(\boldsymbol{\varepsilon}_k + \mathbb{T}_I \Delta t \sum_{n=1}^{k} \boldsymbol{\varepsilon}_n \right) \tag{3.8}$$

where the index $(k, 10)$ refers to the start of control element k. $\mathbb{K}_c$ is a square diagonal matrix where the diagonal consists of controller gains. Similarly, $\mathbb{T}_I$ is a diagonal matrix of inverse controller time constants.

For the purposes of the study it is assumed that the controller is a digital controller that maintains a zero order hold between control time steps, thus the control profile would be similar to that depicted in Figure 3.2. This is accomplished mathematically by forcing the initial value of a finite element to be maintained throughout the element as follows:

$$\mathbf{u}_{k,10} = \mathbf{u}_{k,ij} \qquad \begin{aligned} k &= 1, ..., \text{nCE} \\ i &= 1, ..., \text{nFE} \\ j &= 1, ..., \text{nCOL} \end{aligned}$$

At the start of each control element the deviation between the set point, $\mathbf{y}_{ss}$, and the controlled variable, $\mathbf{y}_{k,10}$, is calculated. This assumes that the controlled

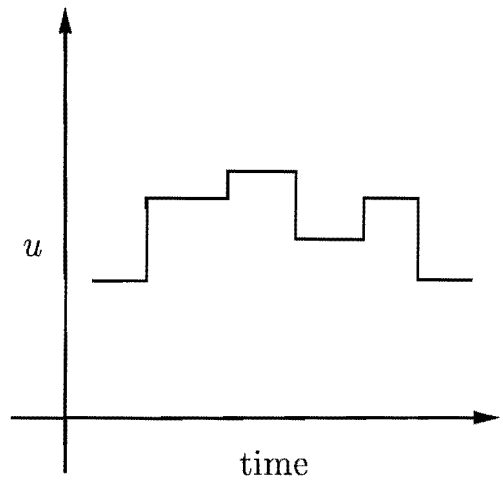


Figure 3.2: Zero order hold

state variable is not equal to the measured variable by default, thus allowing time delays and lag to implemented into the formulation. Thus the steady state values of the controlled variables are assigned by:

$$\mathbf{y}_{ss} = \mathbf{z}_0^y$$

where the set $\mathbf{z}^y$ refers to the set of states that are measured. Lag can be implemented for the case with a measurement lag on the controlled states, for example, by the inclusion of a differential algebraic equation such as:

$$\tau \frac{d\mathbf{y}}{dt} + \mathbf{y} = \mathbf{z}^y$$

Using the above strategy, deadtime can be approximated using first order lags in series.

The dynamic equation shown can, of course, be converted, as discussed earlier, to a normal algebraic equation using OCFE. Thus the measured variable $\mathbf{y}_{k,ij}$ is a function of the measured state variable and time and can be expressed as:

$$\mathbf{y}_{k,ij} = \varphi\left(\mathbf{z}_{l,ij}^y, t_{l,ij}\right)$$

The error Equation (3.7), once discretized, becomes:

$$\varepsilon_k = \mathbf{y}_{ss} - \mathbf{y}_{k,10}$$

and it is assumed that the error remains constant for the duration of the control action step, since it is only necessary to compare the measured variable to its steady state setpoint at the start of each control step. Thus the error at any time needs only to be indexed with a control element.

Since the study considers the effects of input saturation, it is necessary to include some form of anti-reset windup in the formulation. This is accomplished through the use of the velocity form of the PI control equation, which is derived from the difference of Equation (3.8) between two subsequent control elements. The discrete PI control equation thus becomes:

$$\mathbf{u}_{k,10} = \mathbf{u}_{(k-1),10} + \mathbb{K}_c \left(\boldsymbol{\varepsilon}_k - \boldsymbol{\varepsilon}_{(k-1)} + \mathbb{T}_I \boldsymbol{\varepsilon}_k \Delta t \right) \tag{3.9}$$

Another advantage of this formulation is that it is not necessary to calculate the integral error using a dynamic equation.

Input saturation is included in the optimization strategy according to the MILP formulation developed by Young and Swartz (1997) which was detailed in Section 2.3, thus combining the discrete form of the velocity PI controller in Equation (3.9) with the slack variables for input saturation yields:

$$\mathbf{u}_{k,10} - \mathbf{S}_k^L + \mathbf{S}_k^U = \mathbf{u}_{(k-1),10} + \mathbb{K}_c \left(\boldsymbol{\varepsilon}_k - \boldsymbol{\varepsilon}_{(k-1)} + \mathbb{T}_I \boldsymbol{\varepsilon}_k \Delta t \right) \tag{3.10}$$

with the addition of the following inequality constraints:

$$\begin{aligned}
\mathbf{u}^{\mathbf{L}} - \beta \left(\mathbf{1} - \mathbf{Z}_k^L \right) &\leq \mathbf{u}_{k,10} \leq \mathbf{u}^{\mathbf{L}} + \beta \left(\mathbf{1} - \mathbf{Z}_k^L \right) \\
\mathbf{u}^{\mathbf{U}} - \beta \left(\mathbf{1} - \mathbf{Z}_k^U \right) &\leq \mathbf{u}_{k,10} \leq \mathbf{u}^{\mathbf{U}} + \beta \left(\mathbf{1} - \mathbf{Z}_k^U \right) \\
\mathbf{u}^{\mathbf{L}} &\leq \mathbf{u}_{k,10} \leq \mathbf{u}^{\mathbf{U}} \\
\mathbf{0} &\leq \mathbf{S}_k^L \leq \beta \mathbf{Z}_k^L \\
\mathbf{0} &\leq \mathbf{S}_k^U \leq \beta \mathbf{Z}_k^U \\
\mathbf{Z}_k^L, \mathbf{Z}_k^U &\in \{0,1\}^{\mathcal{U}}
\end{aligned}$$

where

$\mathbf{1}$ is a vector of ones, $[1, \ldots, 1]^T$

$\mathbf{u}^{\mathbf{L}}$ is the lower bound on the input trajectory

$\mathbf{u^U}$ is the upper bound on the input trajectory

$\mathbf{S}_k^L$ is the lower slack variable for control element k

$\mathbf{S}_k^U$ is the upper slack variable for control element k

$\mathbf{Z}_k^L$ is the binary variable for control element k that determines whether there is saturation at the lower input bound at this time, and

$\mathbf{Z}_k^U$ is the binary variable for control element k that determines whether there is saturation at the upper input bound at this time.

Young and Swartz (1997) applied the input saturation technique only to a linear process using finite pulse response data calculated from the plant transfer functions using MATLAB, thus their methodology was based on the solution of a linear problem only. Using OCFE allows the applicability to be extended to nonlinear systems.

Combining this formulation with collocation results in a finite mixed-integer nonlinear problem. Problems of this nature are not as easy to solve as ordinary linear, nonlinear, or mixed-integer linear programming problems. However, Floudas (1995) describes a number of methods to solve problems of this kind, which are summarized in Section 2.4.

3.2.4 Inequality Constraints

Aside from the inequalities that arise from the inclusion of input saturation handling, there are general inequalities that apply to the process, denoted by:

$$\mathbf{g}\left(\mathbf{z}, \mathbf{y}, \mathbf{u}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t\right) \leq \mathbf{0}$$

This section highlights certain types of inequalities which are advantageous to include in the problem formulation.

Ensuring Adequate Control

Since part of the problem formulation is concerned with a control profile that optimally handles a particular disturbance, it is necessary to include a measure of the effectiveness of the control trajectory so that the response path does eventually approach the set point. This has to be included in the formulation in some form otherwise the optimization routine will simply choose the trajectory and control parameters to minimize the objective function, which might not be in any way related to control effectiveness.

This can be handled by including the integral square error of the output response as the objective function, but this is not always possible because the formulation is only able to optimize the problem with respect to a single objective function and this might be needed to minimize some other factor such as cost. The integral square error could be forced to be less than a given positive number, as was the case with Schweiger and Floudas (1998), and the set of noninferior points plotted on a curve. This problem is handled in this study by forcing the error to return to within certain tolerance limits, tol, at the end of the trajectory for a certain number of final points, nTOL, for the special case of disturbance rejection operability investigations.

$$-\mathbf{tol} \leq \varepsilon_k \leq \mathbf{tol} \qquad k = (\mathrm{nCE} - \mathrm{nTOL}), ..., \mathrm{nCE}$$

To ensure that the system is closed-loop stable it is also sometimes necessary to include a restriction in the rate of change of the controlled input at the final point, since the input is expected to level out to its new steady state value by the end of the time horizon.

It is not always necessary to explicitly include this constraint, especially if the end point error is indirectly constrained or minimized. For example, the objective function could be a profit expression that is dependent on the amount of "good" product that can be sold. Thus, if the controlled variable was the product quality, then the aim of the optimization routine would be to design the controller to be able to return the product composition as soon as possible to the specified steady state concentration, thereby indirectly forcing the error to return to zero at the end of the time horizon, which is essentially what the above equation does.

Operating Ranges

It is also useful to include operating ranges for the states and controlled variables. These should be included for safety and practical reasons as part of the design.

$$\begin{array}{rcccl} \mathbf{u}^{\mathrm{L}} & \leq & \mathbf{u}_a & \leq & \mathbf{u}^{\mathrm{U}} \\ \mathbf{y}^{\mathrm{L}} & \leq & \mathbf{y} & \leq & \mathbf{y}^{\mathrm{U}} \end{array} \tag{3.11}$$

Controller Parameters

While in theory it is desirable for the controller to have a large gain, this is not always feasible because deadtime, right half plane zeros and input constraints limit the gain (Morari, 1983). In addition, even if the former are not considerations in the formulation, in practice the controller gain and time constant should be limited in order to safeguard against unmodelled deadtime and dynamics that could lead to instability. Doing so also reduces the search space, which often assists the optimization routine in finding a feasible solution.

3.2.5 Final Problem Superstructure

Collating the equations in the previous subsections yields the following final optimization superstructure.

$$\min_{\substack{\mathbf{x}\ \mathbf{Z}\ \mathbb{K}_c\ \mathbb{T}_I\ \boldsymbol{\varepsilon}_k \\ \mathbf{u}_{k,ij}\ \mathbf{y}_{k,ij}\ \mathbf{z}_{k,ij}\ \mathbf{S}_k^L\ \mathbf{S}_k^U\ \mathbf{Z}_k^L\ \mathbf{Z}_k^U}} \quad \Phi\left(\mathbf{z}_{k,ij}, \mathbf{y}_{k,ij}, \mathbf{u}_{k,ij}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t_f\right) \tag{3.12}$$

$$\begin{aligned} \sum_{n=1}^{\mathrm{nCOL}} \mathbf{z}_{k,in}\dot{\phi}_n\left(\gamma_j\right) - \mathbf{f}\left(\mathbf{z}_{k,ij}, \mathbf{y}_{k,ij}, \mathbf{u}_{k,ij}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t_{k,ij}\right)\Delta\delta &= \mathbf{0} \\ \mathbf{u}_{k,10} - \mathbf{S}_k^L + \mathbf{S}_k^U - \left[\mathbf{u}_{k-1,10} + \mathbb{K}_c\left(\boldsymbol{\varepsilon}_k - \boldsymbol{\varepsilon}_{k-1} + \mathbb{T}_I\boldsymbol{\varepsilon}_k\Delta t\right)\right] &= \mathbf{0} \\ \mathbf{h}\left(\mathbf{z}_{k,ij}, \mathbf{y}_{k,ij}, \mathbf{u}_{k,ij}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t_{k,ij}\right) &= \mathbf{0} \\ \mathbf{g}\left(\mathbf{z}_{k,ij}, \mathbf{y}_{k,ij}, \mathbf{u}_{k,ij}, \mathbf{x}, \mathbf{Z}, \mathbf{p}, t_{k,ij}\right) &\leq \mathbf{0} \\ \mathbf{y}_{k,ij} - \boldsymbol{\varphi}\left(\mathbf{z}_{l,ij}, t_{l,ij}\right) &= \mathbf{0} \\ \boldsymbol{\varepsilon}_k - \left(\mathbf{y}_{ss} - \mathbf{y}_{k,10}\right) &= \mathbf{0} \\ \mathbf{z}\left(0\right) - \mathbf{z}_0 &= \mathbf{0} \\ \mathbf{y}_{ss} - \mathbf{z}_0^y &= \mathbf{0} \end{aligned}$$

$$
\begin{aligned}
\mathbf{u^L} - \beta\left(\mathbf{1} - \mathbf{Z}_k^L\right) &\leq \mathbf{u}_{k,10} \leq \mathbf{u^L} + \beta\left(\mathbf{1} - \mathbf{Z}_k^L\right) \\
\mathbf{u^U} - \beta\left(\mathbf{1} - \mathbf{Z}_k^U\right) &\leq \mathbf{u}_{k,10} \leq \mathbf{u^U} + \beta\left(\mathbf{1} - \mathbf{Z}_k^U\right) \\
\mathbf{u^L} &\leq \mathbf{u}_{k,10} \leq \mathbf{u^U} \\
\mathbf{0} &\leq \mathbf{S}_k^L \leq \beta\mathbf{Z}_k^L \\
\mathbf{0} &\leq \mathbf{S}_k^U \leq \beta\mathbf{Z}_k^U
\end{aligned}
$$

for $k = 1, ..., \text{nCE}$, $i = 1, ..., \text{nFE}$, and $j = 1, ..., \text{nCOL}$,

$$
\mathbf{z}_{k,i0} - \sum_{n=1}^{\text{nCOL}} \mathbf{z}_{k,(i-1),n}\phi_n(1) = \mathbf{0}
$$

for $k = 1, ..., \text{nCE}$ and $i = 2, ..., \text{nFE}$, and

$$
\mathbf{z}_{k,10} - \sum_{n=1}^{\text{nCOL}} \mathbf{z}_{(k-1),\text{nFE},n}\phi_n(1) = \mathbf{0}
$$

for $k = 2, ..., \text{nCE}$.

where

$$
\begin{aligned}
\mathbf{z}_{k,ij} &\in \Re^{\mathcal{K}} \\
\mathbf{y}_{k,ij} &\in \Re^{\mathcal{Y}} \\
\mathbf{u}_{k,ij} &\in \Re^{\mathcal{U}} \\
\mathbf{x} &\in \Re^{\mathcal{X}} \\
\mathbf{p} &\in \Re^{\mathcal{P}} \\
\mathbf{S}_k^L, \mathbf{S}_k^U &\in \Re^{\mathcal{U}} \\
\mathbf{Z} &\in \{0,1\}^{\mathcal{Z}} \\
\mathbf{Z}_k^L, \mathbf{Z}_k^U &\in \{0,1\}^{\mathcal{U}}
\end{aligned}
$$

This superstructure can be used to convert general nonlinear dynamic algebraic design problems that do not take input saturation into account, into a MINLP problem that handles saturation rigorously.

3.3 Dynamic Operability

The design is now able to take into account the dynamic operation of the plant, since disturbances of a fixed step size can be included as parameters to the prob-

lem, **p**. This is best illustrated with an example. For instance, if it is known that a plant is likely to be at its least controllable when a disturbance of a specific size is applied, then by including the disturbance to the process and gauging its effect on the controlled and manipulated variable trajectories, it is possible to exclude plants that will not be able to cope with the disturbance. The optimization routine will be able to find the best process structure and controller parameters to deal with this disturbance.

Since low cost and good controllability are two of the major objectives of every design it is not surprising to find that they often run counter to each other, i.e. a low-cost plant might be less controllable than one which costs more. Therefore it would be ideal to find a happy medium and design a plant that costs as little as possible while still being able to control the process adequately.

Optimization-based designs that do not include rigorous saturation handling, although they might offer an adequate degree of control, do not necessarily achieve the lowest possible cost for the same amount of controllability, since they ignore the fact that the inputs can be operated at their limits until the error is sufficiently reduced. The strictly linear formulation forces the input to merely touch the bound and then move away from it, while in reality input saturation still provides adequate control despite the fact that it causes the input trajectory to become temporarily nonlinear. This is nevertheless valid operation of a plant. Permitting the input to saturate allows the steady state actuator to operate closer to the input trajectory bounds, for the controller to have a higher gain, etc. The result is a knock-on effect that synthesizes a plant which is a more accurate representation of the potential cost of the plant.

The formulation can also be extended to include uncertainties in the disturbances and plant parameters using the work of Bahri *et al.* (1996) and Mohideen *et al.* (1997).

The benefits of this formulation will now be illustrated with a simple example.

3.4 Example

The following is a similar example to that used in Young and Swartz (1997) to illustrate the advantages of including input saturation in the formulation. Al-

though the example is based on the transfer functions, the dead times have been excluded to simplify the problem and reduce the number of states. Later case studies do give examples of dead time and lag handling, though.

Operation of a fluid catalytic cracking unit (FCCU) is vital for the economic and operational performance of refineries. These units are characterized by highly interactive dynamics and complex constraints and are not simple to operate. The function of the FCCU is to take heavy refinery streams and crack these into lighter, more valuable products which can then be processed further downstream. These reactions require a catalyst in order to break the molecules into their more valuable products. However, the catalyst deactivates over time as a consequence of coking and thus the catalyst needs to be regenerated. This is accomplished by the FCCU which consists of a reactor and a catalyst regenerator as depicted in Figure 3.3.

The feed streams to the FCCU consist of a hot gas oil stream, recycle oil, and a cold gas oil stream. These are mixed in the reactor riser and it is here that the cracking reactions occur. These reactions are endothermic and the heat required is obtained from the hot catalyst from the generator. After exiting the riser the catalyst falls down into the regenerator and the gaseous reaction products exit through the top of the reactor. In the regenerator, the spent catalyst is fluidized by an air stream provided by air blowers. This serves to fluidize the catalyst bed as well as supporting combustion, which burns off the coke on the surface of the catalyst and regenerates it. Cyclones at the top of the regnerator separate the gaseous combustion products from any entrained catalyst. In general, the most important constraints are associated with the regenerator and it is these constraints which usually constrain the optimal operation of the unit. For a more detailed discussion of FCCUs, refer to Mcfarlane *et al.* (1993) and Grosdidier *et al.* (1993).

The model used by Young and Swartz (1997) was originally based on that described by Grosdidier *et al.* (1993). Young and Swartz (1997) only concern themselves with the regenerator, and dynamic interactions with the reactor are neglected. A PI control scheme regulates the flue gas O_2 concentration, y_1, by manipulating the air flowrate, $u_{c,1}$, while the regenerator bed temperature, y_2, is controlled by manipulating the hot gas oil flowrate, $u_{c,2}$. The example considers a single temporary disturbance caused by a variation in the feed composition, which

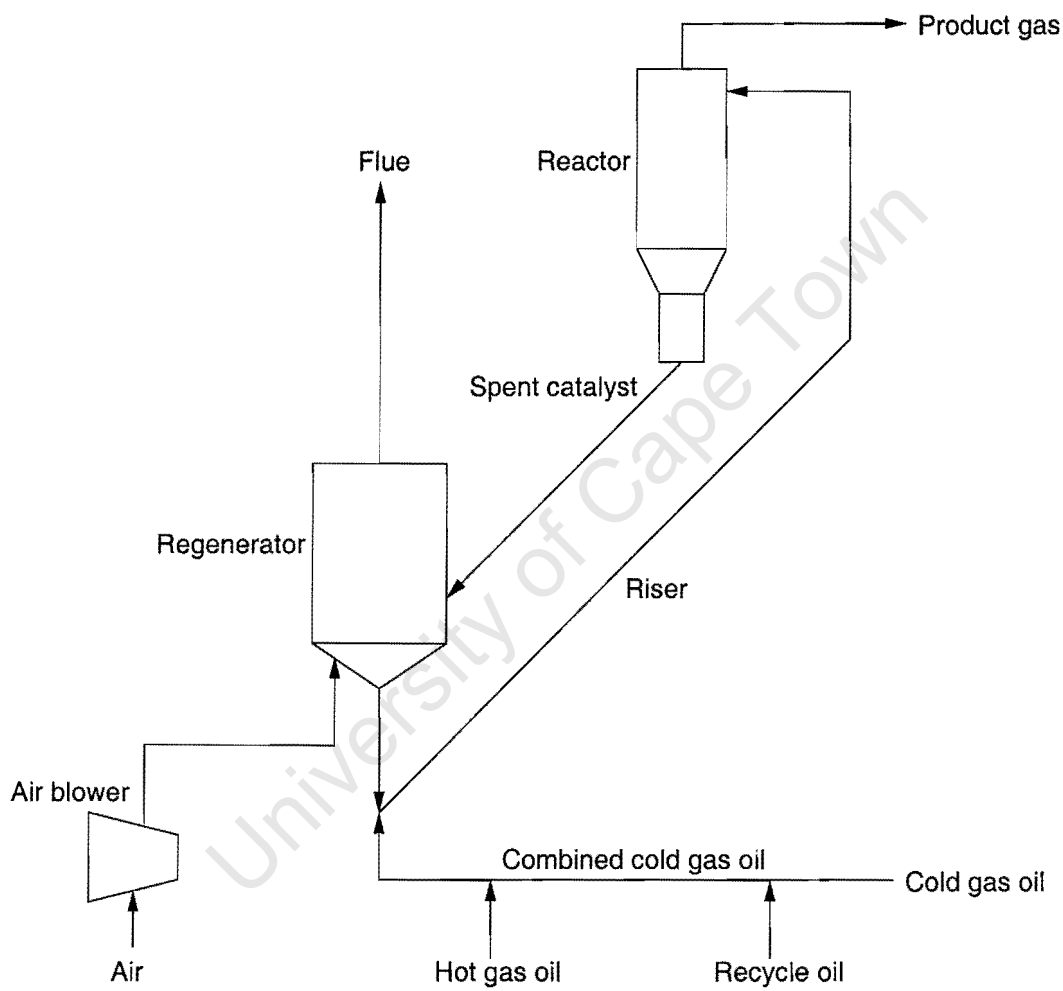


Figure 3.3: Typical FCCU flow diagram

is modeled by changing the amounts of recycle oil, d'_{tmp} in the combined gas oil feed. The economic objective serves to maximise the combined gas oil flow, $u_{c,3}$. This is shown by Young and Swartz (1997) to be equivalent to the maximization of the profit. The design variables focus on the selection of an optimum steady state.

3.4.1 Mathematical Model

The transfer functions without dead times were converted into a state space dynamic form using MATLAB and the resulting dynamic equations were then converted via collocation into a set of linear equations. In the study by Young and Swartz (1997), the dynamics were handled by discretizing the profile, but unlike this example, OCFE was not used.

The process constraints are as follows:

$$\begin{array}{rcll} 1\,\% & \leq & \text{flue } O_2 & \\ 705\,^oC & \leq & \text{regenerator bed temperature} & \leq\ 735\,^oC \\ 140\,\text{tph} & \leq & \text{air flowrate} & \leq\ 155\,\text{tph} \\ 90\,\text{m}^3/\text{hr} & \leq & \text{hot gas oil} & \leq\ 110\,\text{m}^3/\text{hr} \\ 90\,\text{m}^3/\text{hr} & \leq & \text{combined cold gas oil} & \leq\ 110\,\text{m}^3/\text{hr} \end{array}$$

The linearized process dynamics are given by the following transfer functions:

$$\begin{bmatrix} y'_1 \\ y'_2 \end{bmatrix} = \begin{bmatrix} \frac{0.097(1.7s+1)}{19s^2+6.5s+1} & \frac{-0.87}{13s^2+4.9s+1} \\ 0 & \frac{0.55}{27s^2+8.7s+1} \end{bmatrix} \begin{bmatrix} u'_{c,1} \\ u'_{c,2} \end{bmatrix} + \begin{bmatrix} \frac{-0.48}{48s^2+14s+1} \\ \frac{0.36}{33s^2+6.5s+1} \end{bmatrix} d'_{\text{tmp}}$$

The steady-state linear model around the nominal linearization point is given by the process gains:

$$\begin{bmatrix} y_1^{set} \\ y_2^{set} \end{bmatrix} = \begin{bmatrix} 0.097 & -0.87 & -0.092 \\ 0 & 0.55 & 0.55 \end{bmatrix} \begin{bmatrix} u^i_{c,1} \\ u^i_{c,2} \\ u_{c,3} \end{bmatrix} + \begin{bmatrix} -0.48 \\ 0.36 \end{bmatrix} d^i_{\text{tmp}}$$

Here the superscript i denotes the initial steady state value.

The following operating conditions are assumed by Young and Swartz (1997):

$$\begin{aligned} \overline{y}_1 &= 1\,\% \\ \overline{y}_2 &= 715\,^{o}\mathrm{C} \\ \overline{u}_{c,1} &= 147\,\mathrm{tph} \\ \overline{u}_{c,2} &= 94\,\mathrm{m}^3/\mathrm{hr} \\ \overline{u}_{c,3} &= 90\,\mathrm{m}^3/\mathrm{hr} \end{aligned}$$

For this operating point the constraint set becomes:

$$\begin{aligned} 0\,\% &\leq \text{flue } O_2 \\ -10\,^{o}\mathrm{C} &\leq \text{regenerator bed temperature} \leq 20\,^{o}\mathrm{C} \\ -7\,\mathrm{tph} &\leq \text{air flowrate} \leq 8\,\mathrm{tph} \\ -4\,\mathrm{m}^3/\mathrm{hr} &\leq \text{hot gas oil} \leq 16\,\mathrm{m}^3/\mathrm{hr} \\ 0\,\mathrm{m}^3/\mathrm{hr} &\leq \text{combined cold gas oil} \leq 20\,\mathrm{m}^3/\mathrm{hr} \end{aligned}$$

To enable the problem to exhibit input saturation, Young and Swartz (1997) modified the constraint on the cold gas oil as follows:

$$0\,\mathrm{m}^3/\mathrm{hr} \leq u_{c,1} \leq 40\,\mathrm{m}^3/\mathrm{hr}$$

The following PI controller tuning parameters give stable closed-loop behavior.

Controller	**Manipulated Input**	**Controlled Output**	K_c	τ_I
1	Air flow ($u_{c,1}$)	Flue O_2 conc. (y_1)	4.5	10
2	Hot gas oil flow ($u_{c,2}$)	Regenerator bed temp. (y_2)	1.0	15

The disturbance uncertainty range was assumed to be:

$$\Gamma = \{d_{\mathrm{tmp}} | -1 \leq d_{\mathrm{tmp}} \leq 1\}$$

resulting in the critical points shown in the table below.

Critical Point	**1**	**2**
d^i_{tmp}	1	-1
d'_{tmp}	-2	2

Table 3.1: Results for example with saturation

Variable	Value	Description
Objective	36.52	
y_1^{set}	0.25	Flue gas O_2 concentration
y_2^{set}	18.24	Regenerator bed temperature
$u_{a,1}$	8.00	Combustion air flow
$u_{a,2}$	-2.71	Hot gas oil flow
$u_{a,3}$	36.52	Combined cold gas oil flow

The critical points of the disturbance range refer to those values of d_{tmp} which limit the feasibility of the inequality constraints to the greatest degree (Young and Swartz, 1997).

The control time step used is 2 minutes, with the closed-loop settling time for both outputs nTOL = 5 time intervals, and the time horizon, nCE 100 intervals. The number of finite elements per control element is 1, and there are 2 collocation points per finite element. The value of the arbitrary large constant, β, is chosen as being 20.

The result is a dynamic MILP problem which is put into the optimization form required by Equation(3.12). The solver used is Cplex, which is implemented via GAMS. It is assumed that the polynomial in each finite element could be approximated with a small number of collocation points.

3.4.2 Results

The response was investigated for critical point 2 and the results have been reported in Table 3.1 and Figures 3.4 and 3.5. It should be noted that when the problem is dealt with in a strictly linear fashion the objective value is 35.41. The problem had 400 binary variables and the full problem took 2041.330 seconds to solve when Cplex was run with the options: subalg 1, varsel 2, and nodesel 2.

The option "subalg 1" forces the mixed integer optimization routine to use a primal simplex method to solve the linear sub-problems at each node. The primal simplex algorithm was used instead of the default dual simplex method because the latter method ran into computational difficulties and terminated without a solution. Using the option "varsel 2" selects the next variable to branch based on pseudo

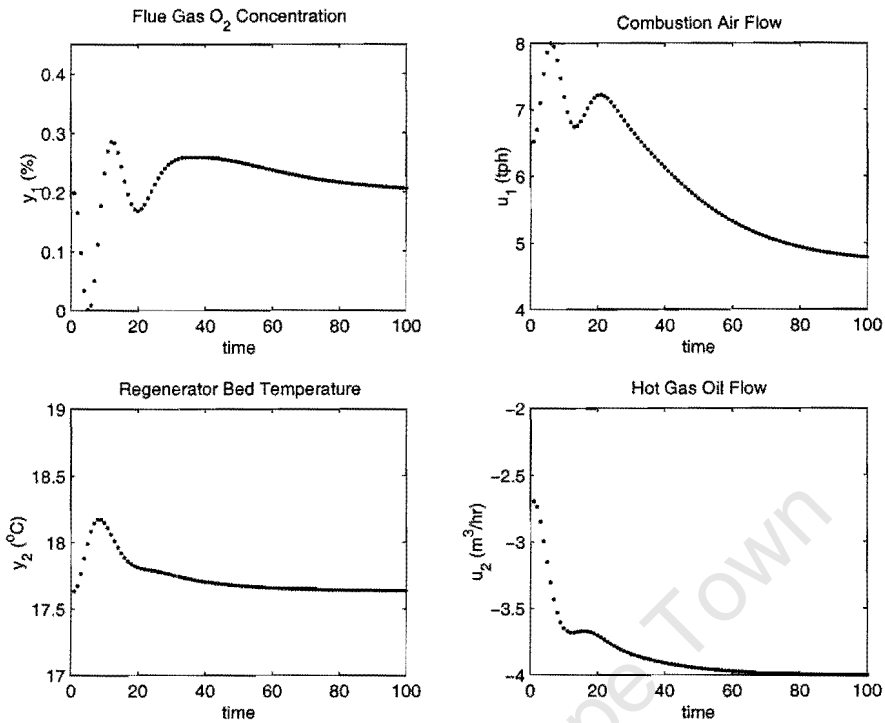


Figure 3.4: Strictly linear control of an FCCU

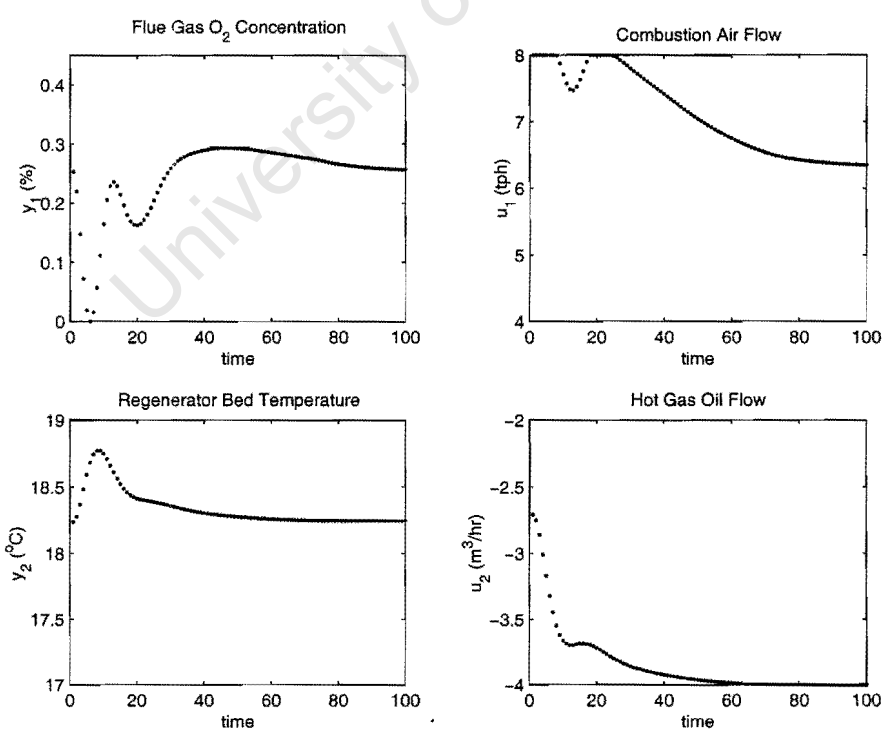


Figure 3.5: Rigorous input saturation handling of an FCCU

costs. "Nodesel 2" uses a best-estimate search to select the next node to process when backtracking. The options "varsel 2" and "nodesel 2" are recommended by Cplex to allow the solver to quickly find an initial integer feasible solution. If the default options are used instead, the optimization routine for this example can proceed for a number of *days* without finding any integer feasible solution.

It can be seen in Figure 3.5 that the combustion air flowrate saturates in comparison to the linear case depicted in Figure 3.4.

3.5 Summary

An optimization superstructure that is able to design a dynamically optimal plant that rigorously handles input saturation was presented. The formulation extends the work of Young and Swartz (1997) in that it is now able to handle nonlinear dynamic equations through the use of orthogonal collocation on finite elements. A simple example was presented to illustrate the application of the formulation.

Chapter 4

Computation Time Reduction

4.1 Introduction

Solving the optimization problem formulated in the previous chapter results in a mixed-integer problem. The integers used in this formulation have the special property that they are binary variables, i.e. they can only take on values of zero or one. Since the variables are not continuous, mixed-integer problems cannot be solved using conventional linear programming methods such as Newton's method. Special algorithms were thus developed to handle these kinds of problems. The main categories of MINLP algorithms are listed below.

- Branch and bound methods
- Cutting plane methods
- Decomposition methods
- Logic-based methods

The problem is essentially a combinatorial issue, since each binary variable can have only two values, zero or one, with no other value between them. A brute force method could be used to solve the problem by trying out each possible combination of zero or one for each variable. However, this is not a viable method

because for n binary variables there are 2^n possible combinations. Thus a problem with 15 binary variables would have 32768 possible combinations of binary variables, any number of which might not be feasible. If the problem is linear and each combination could be solved in 1/2 second, it would still take over 4 hours to go through all the possible solutions. Solving a problem with the time domain subdivided into 100 discrete elements and two binary variables to represent saturation at the upper and lower input bounds for a 2×2 control system, results in 400 binary variables. Even with current advances in computer technology this would take a considerable amount of time.

The "branch and bound class" of methods are the algorithms most commonly used to solve large scale problems. First, the algorithm treats the binary variables as being continuous and solves the relaxed linear programming problem. If the solution is an integer solution then the algorithm terminates, otherwise it splits the node into two candidate subproblems. If one of the candidate problems results in an integer solution then one of the other candidate subproblems is solved. Otherwise, the candidate subproblem is split into two more subproblems and its children are added to the list of candidate subproblems. To avoid having to solve all of the subproblems, fathoming tests are employed to eliminate nodes and their children nodes from the tree (Floudas, 1995).

Since the example given in the previous chapter is a mixed-integer linear problem, the problem is solved using the "branch and bound" algorithm implemented by Cplex. Even this problem took a significant amount of time to arrive at a solution and the example is not even nonlinear.

In the following section an algorithm that is able to reduce the computation time for problems formulated with input saturation formulation of Young and Swartz (1997) is proposed and the proof thereof discussed.

4.2 Factors Affecting the Computation Time

One of the dominant factors influencing the computation time of solving mixed-integer linear problems is the number of binary variables. If a method can be developed that reduces the number of integer variables in a formulation then the

computation time will be correspondingly reduced. However, it is not merely sufficient to just eliminate variables since it must be shown that the two formulations are equivalent and that the solution obtained by the reduced method must equal the solution that would be obtained by the full problem.

In a similar vein, if the number of candidate solutions that need to be evaluated can be reduced then the solver will arrive at a solution sooner.

The algorithm that is described is able to select the points at which input saturation will occur and then limits the binary variables to those points only. The algorithm is presented first for a *linear* mixed-integer problem. The extension to nonlinear problems is discussed in Section 4.3.4.

4.3 MILP Computation Time Reduction Algorithm

4.3.1 The Algorithm

Step 1 Solve the linear programming problem without rigorous input saturation handling. If the solution is such that the input trajectory has a maximum or minimum at an upper or lower input bound, then proceed to step 2. For the purpose of illustration, assume that the result produced an input trajectory that touched the upper bound at control time step k and the lower input bound at time l.

Step 2 Apply the input saturation handling constraints to those points where the linear programming problem determined that input variables are at an upper or lower bound. For the example in the previous step, this entails creating binary variables Z_k^U and Z_l^L. The remaining points are solved for using the standard PI control equation with the slack variables forced to zero, since the binary variables are effectively zero at these points.

Step 3 Solve the problem again using an MILP solver to determine the optimal values for the binary variables.

Step 4 If the new solution has an input trajectory that touches the input bound at a point other than those already being handled by rigorous input saturation handling, then repeat steps 2 and 3. Thus the new saturation points are added to the current set of saturation points. For example, this means that if the input trajectory now touches a bound at a point other than at time k or l then add the new point at time m to the set of input saturation points and solve the MINLP over points k, l, m.

Step 5 If the input trajectory does not touch the bounds at any new points, terminate the algorithm.

The theory behind the steps in the algorithm will now be discussed and used to prove that the resulting solution is the global optimum.

4.3.2 Discussion and Proof of Algorithm

The algorithm described above is able to reduce the number of points at which binary variables are required to represent input saturation by only choosing the points at which it is necessary to apply input saturation. This only applies for the case of *linear* design optimization problems. The restriction to the linear case is necessary because the proof of the algorithm requires that the linear problem is solved to the global optimum at each step.

The first step terminates the problem if a solution is found for which the input trajectory does not touch a bound. This step is a special case of steps 4 and 5. The reasoning behind this is that the problem constraints have obviously been met without having to approach a bound, therefore there is no point in creating saturation binary variables, because the input bounds are not limiting constraints and no further improvement in the objective can be achieved even if the constraints are removed in their entirety. Thus the algorithm is allowed to terminate.

If the first step arrives at a solution that does have a control trajectory that touches the bound at a point, then this implies that the input bound at that particular point is one of the constraints which limits further improvement of the

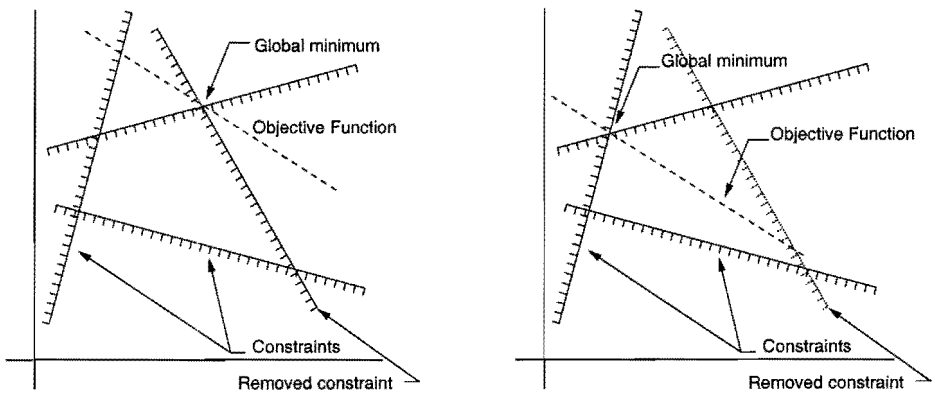


Figure 4.1: Change in the optimum when a constraint is removed

objective function. This is because the input path constraints are applied at each control time step, since:

$$u^L \leq u_{k,10} \leq u^U$$

The constraint is thus *active*, i.e. the actuator position is exactly at one of its limits at this time.

If an active constraint were able to be "removed", then solving the problem again would lead to an improvement in the optimum. If, even after removing the constraint, the optimum solution does not change, then this implies that the problem is already at the achievable optimum. This can be illustrated with the following example shown in Figures 4.1, 4.2 and 4.3. In these diagrams the shaded side of the lines indicates the direction of constraint violation, i.e. the infeasible region.

Assume that the problem has been solved and a feasible minimum has been found. Then one of the problem constraints, either an inequality or an equality, is removed and the modified problem is then solved again. There are two possible outcomes.

In the first case, the optimum comes to settle at a new improved point, as shown in Figure 4.1. The change in the optimum implies that the removed constraint was active.

In the second case, the optimum remains where it was originally. Whatever the case, the optimum cannot worsen because the original solution was the global

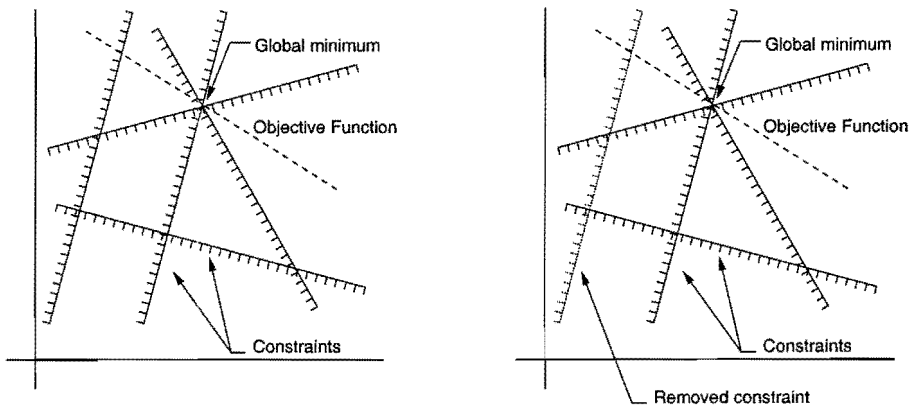


Figure 4.2: An inactive constraint is removed

optimum, since this is a linear problem and is thus convex, and the new problem is therefore solved to the global optimum as well.

The second case will now be examined. If the optimum does not improve and the solution remains the same then there are two possible causes:

1. The constraint that was "removed" was not an active constraint.
2. Another constraint is active and is keeping the optimum at that point.

The first and second possibilities have been illustrated in Figures 4.2 and 4.3 respectively. This holds even for a mixed-integer linear programming problem, since the integer constraints do not affect the convexity of the problem. No other possibilities exist because it is assumed that the problem is solved to optimality.

Returning to the problem of rigorous input saturation handling, it can be similarly demonstrated that if the active bound constraint on the input trajectory point could be removed in some manner and the result changes, then the new solution will always be an improvement on the previous optimum as was shown. If the optimum does not change then this implies that some other constraint is binding.

The algorithm takes advantage of the fact that it is straightforward to determine if one of the constraints in the set of bound constraints is active, thereby making

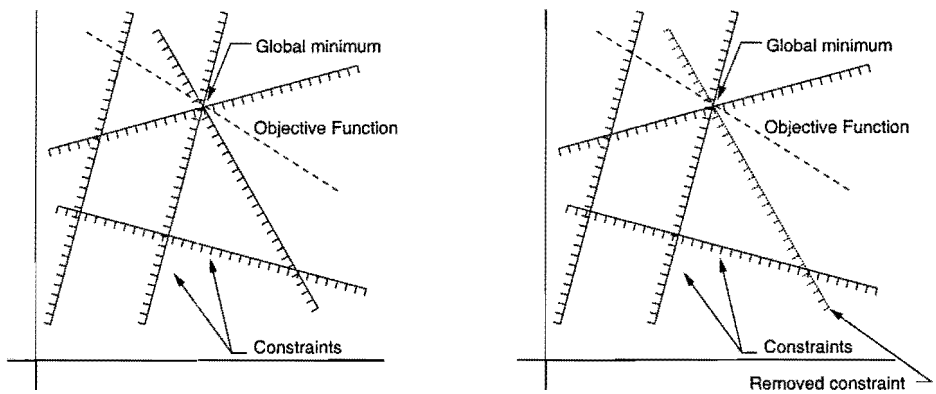


Figure 4.3: Another constraint is active

it a simple matter to decide which constraint must be removed in order to allow the optimum to improve.

A constraint can be "loosened" by the addition of slack variables. Essentially what occurs is that the slack variable will take on any value such that the constraint is met. If the slack variables are not constrained then the feasible region of the constraint is extended and the constraint to which the slack has been added is essentially removed. Not all optimization routines are able to handle variables without bounds, though, but if it can be shown that the bound on the slack variables is not active then the above still applies. The "loosening" of the constraint by slack variables has been illustrated in Figure 4.4.

The rigorous input saturation handling formulation developed by Young and Swartz (1997) is characterized by the addition of slack variables to the control equation. Without saturation handling the actuator position is limited by the input bounds which thereby limits the PI control equation of Equation (3.9). However, with saturation handling, even though the actuator position is still limited by the bounds, the addition of the slack variables to the control equation allows the right hand side of the PI control equation to take on any value and consequently the equality constraint is "removed".

Since the actuator position is constrained to lie between the upper and lower actuator bounds whether or not rigorous saturation handling is included, it can be deduced that it is not the inequality bounds of Equation (3.11) on the actuator

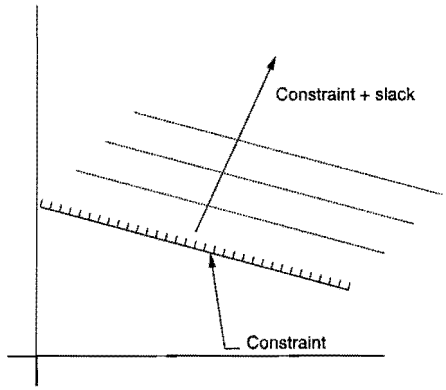


Figure 4.4: Addition of slack variables to a constraint

position that limit performance. However, the assumption that the control action is strictly linear, forces the actuator value to equal the right hand side of Equation (3.9). This implies that if the actuator value is constrained by an operating range bound, then the right hand side of the control equation will be constrained. However, this is not the case when rigorous saturation handling is implemented. Thus it can be deduced that the limiting constraint that must be "removed" is the linear control equality and it was previously shown that the addition of slack variables is able to accomplish this.

The algorithm only includes slack and binary variables for points at which the input trajectory has hit an upper or lower bound. This implies that a linear controller equation had been the performance-limiting constraint prior to the inclusion of slack variables at these points. It was previously shown that the only two possible reasons why the optimum would not change if a constraint is removed is if the eliminated constraint is not binding, or if another constraint is limiting further improvement in the objective function. Thus, if there is no change in the optimum solution after the removal of the binding linear controller equation, then this leaves only the second possibility, namely that another non-removable constraint is preventing the improvement of the solution. Thus, if we repeat steps 2-3 until there is no change in the the optimal solution, then slack variables will be added and the optimum will improve until the constraint that is limiting improvement in the solution is *not* an input saturation constraint. The result is a guaranteed global optimum for the case of rigorous input saturation handling.

Table 4.1: Summary of iteration progress

Iteration	Binary variables	Objective value
0	0	35.4125
1	2	35.5592
5	10	36.2737
10	20	36.3657
31	44	36.5188

There are, of course, two very important restrictions:

1. that the problem is linear, and
2. the the problem is solved to optimality at each step.

The algorithm's progression towards the global optimum for MILP systems will now be demonstrated with an example.

4.3.3 Example

The example under study is the same FCCU example mentioned in the previous chapter. The problem is a 2×2 system with upper and lower bounds on both inputs to the system for a 100 time steps and a controller time step of 2 minutes.

First, the problem is solved for the case with strictly linear control. The resulting input trajectories have been depicted in Figure 4.5.

It can be seen that the combustion air flow has reached its upper bound at time equal to 11 and that the hot gas oil flow has reached its lower bound at -4 at the final time step. Thus the binary variable that allows slack is included for points $k = 6$ and $k = 100$. The problem is solved again with the binary variables and the result is shown in Figure 4.6. It is apparent from the graph that the combustion air flow now saturates at the points around $k = 6$.

To show how the algorithm approaches the global optimum, the results for iterations 5, 10 and the final iteration, have been shown in Figures 4.7, 4.8 and 4.9 respectively.

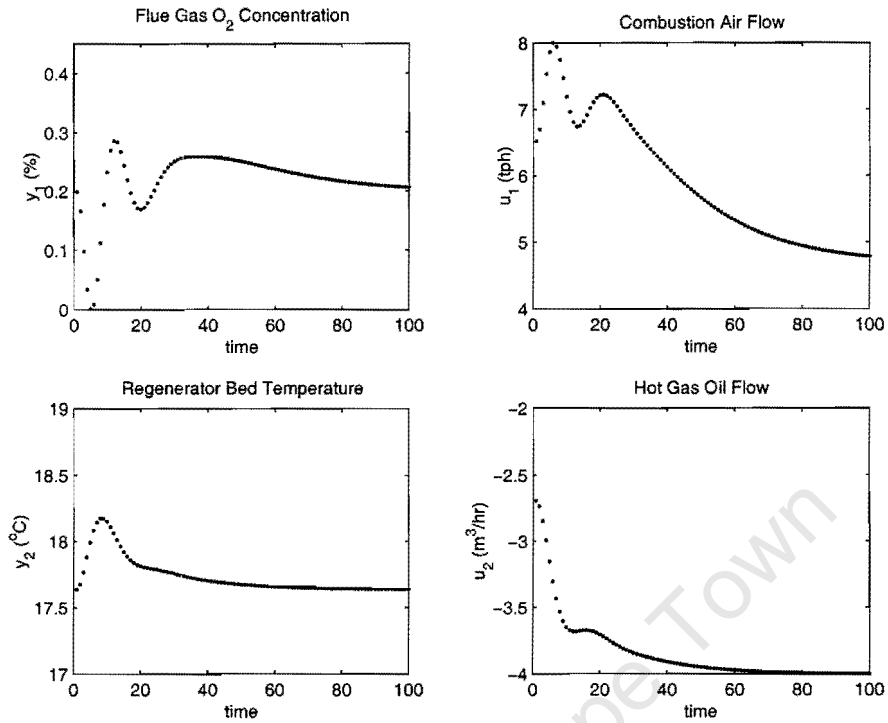


Figure 4.5: Solution for a strictly linear controller (iteration 0)

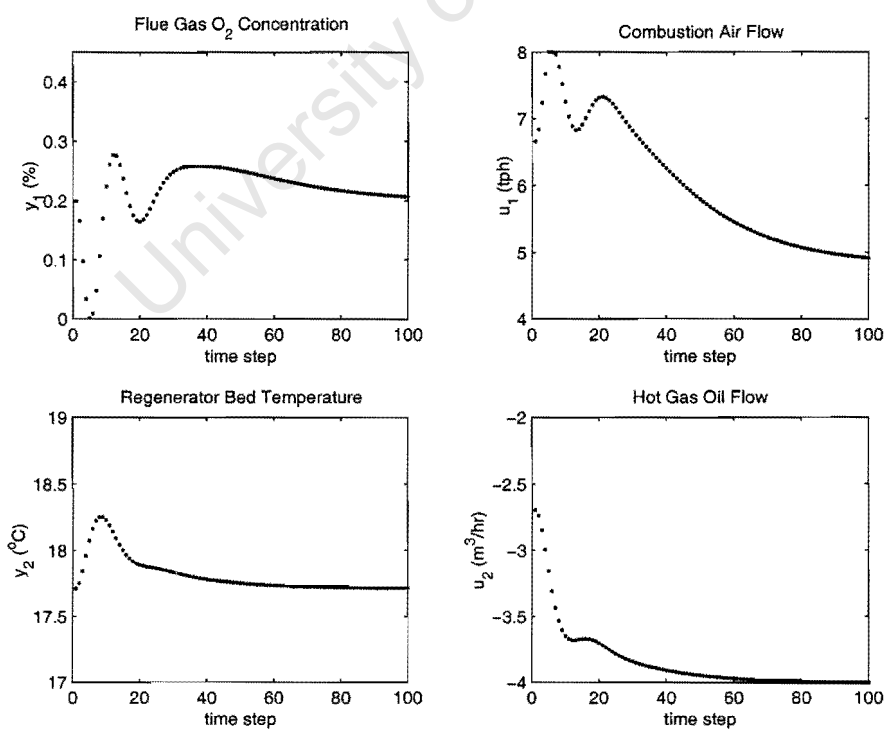


Figure 4.6: Solution after iteration 1

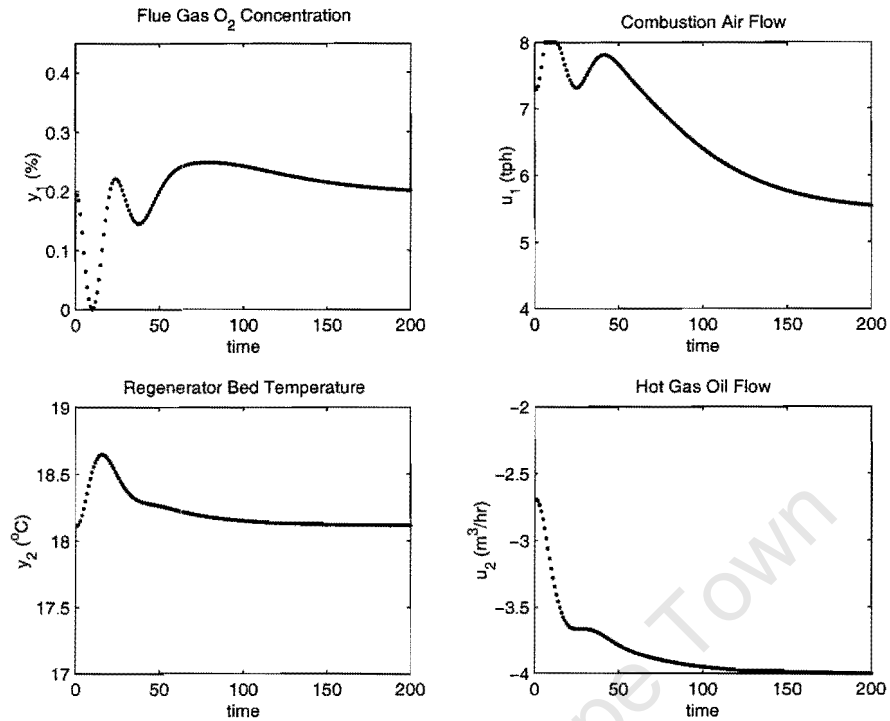


Figure 4.7: Solution after iteration 5

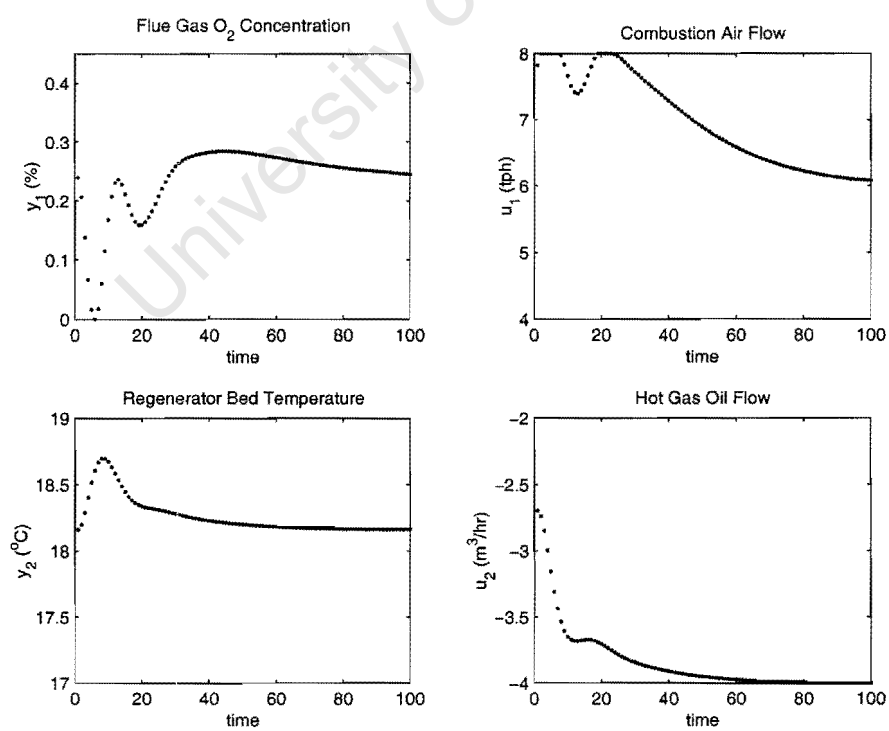


Figure 4.8: Solution after iteration 10

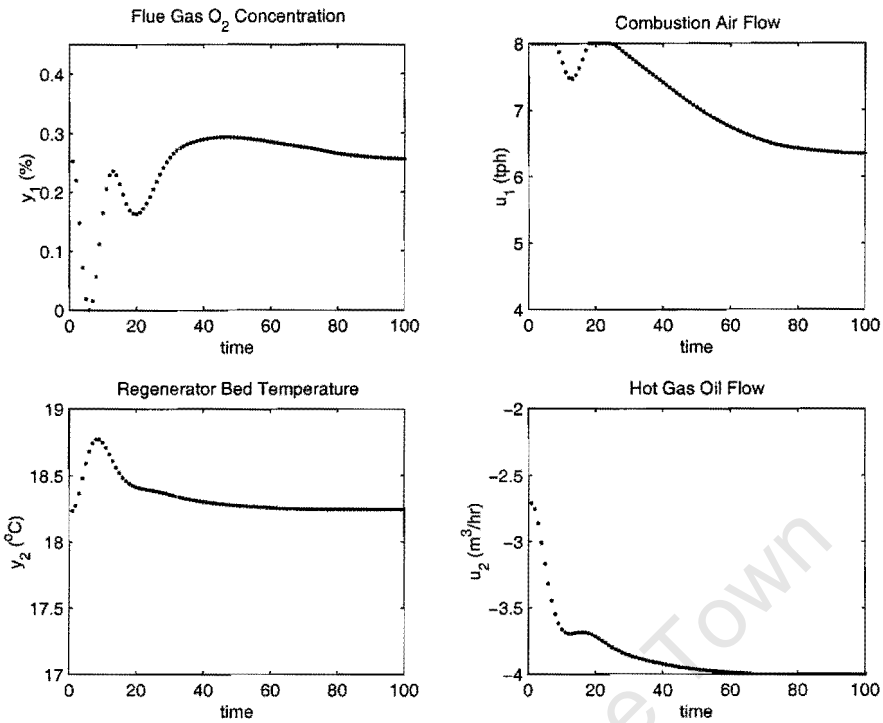


Figure 4.9: Solution after final iteration

Table 4.2: Comparison of algorithm to full MILP

	Full Problem	With Algorithm
Integer Variables	400	44
Iterations	256190	5878
Time (min)	37.16	2.29

The algorithm, using Cplex as a MILP solver, is applied on a dual PIII XEON 450 MHz with 500 MB RAM, running Linux. A comparison of the algorithm-based solution and the MILP problem is given in Table 4.2. The two results are compared to one another in Figures 4.11 and 4.10, and appear to coincide.

4.3.4 Extension to Nonlinear systems

Unfortunately, as mentioned previously, most design optimization formulations will not be linear, therefore an extension of this algorithm to the nonlinear case would be advantageous.

The extension is applicable to nonlinear systems with a guarantee of finding the global optimum, provided that the nonlinear system is also convex, or that the

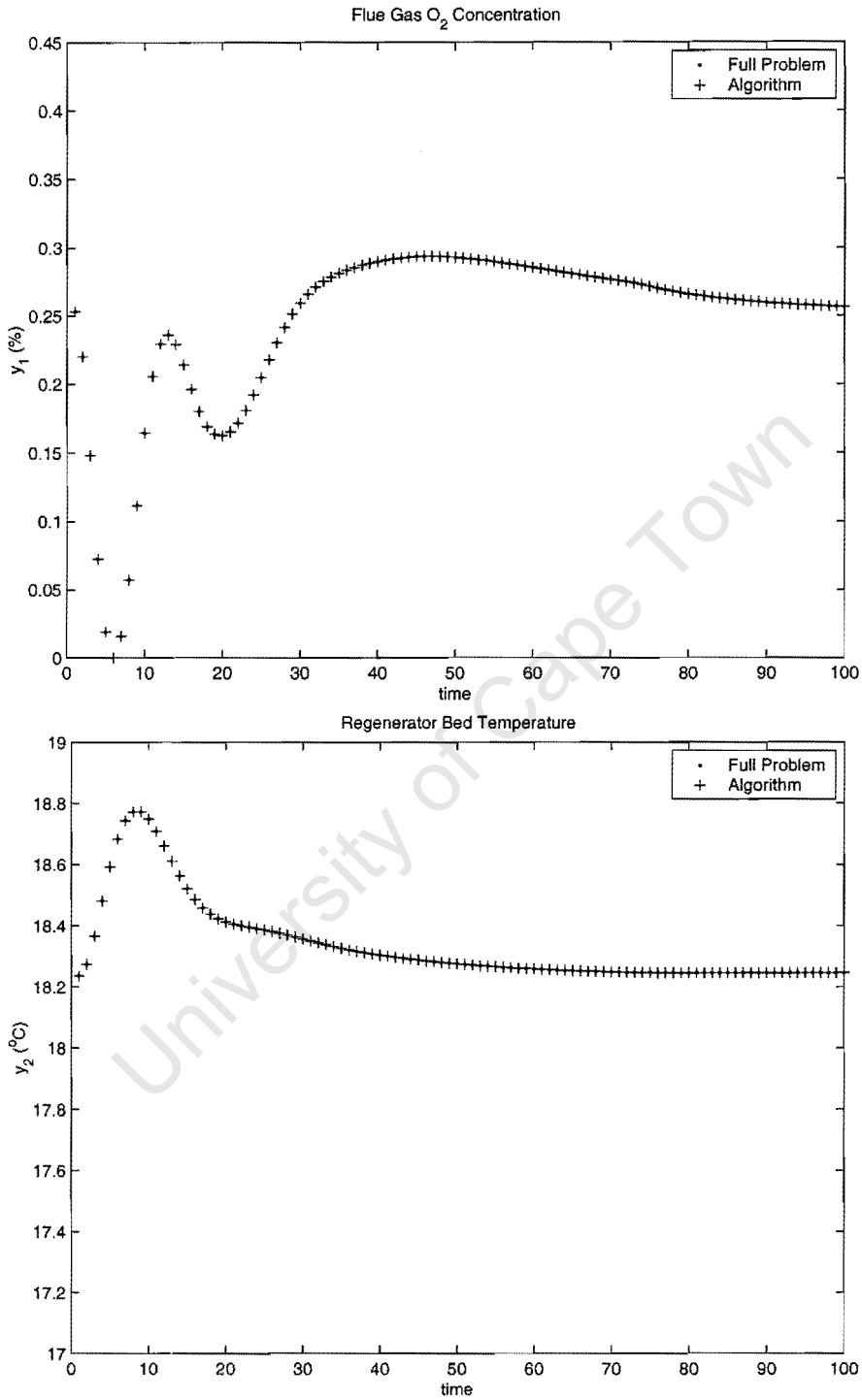


Figure 4.10: Comparison of outputs of algorithm-based solution to full MILP problem

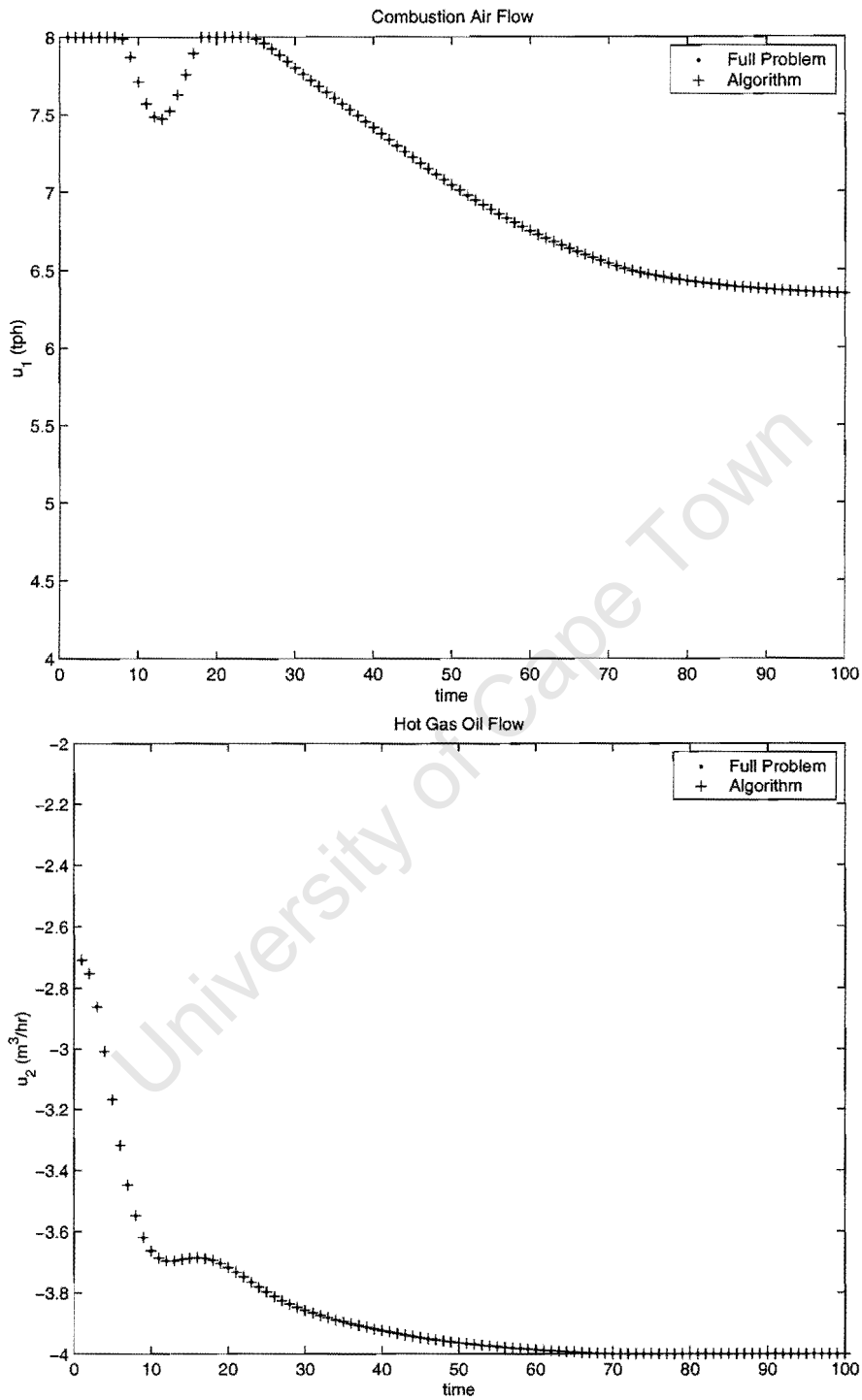


Figure 4.11: Comparison of inputs of algorithm-based solution to full MILP problem

Table 4.3: Comparison of cutoff method to full MILP

	Full Problem	With Cutoff
Iterations	256190	254286
Time (min)	37.16	36.93

MINLP solver is able to guarantee that the solution found is the global optimum.

If the algorithm is applied to a nonlinear problem that is nonconvex, then, although there is no guarantee of it finding the global optimum, the local optimum can at least be used as a lower or upper bound on the global solution depending on whether the objective is being maximized or minimized, respectively.

4.4 An Alternative Approach

An alternative method to reduce computation time is now proposed, that is based on the proof above, that shows that a model which allows saturation must have a solution that is equivalent or better than the case where a strictly linear controller is used.

Certain MILP solvers, such as Cplex, have a feature that makes it possible to immediately discard all nodes that are worse than a given value. Cplex has a "cutoff" option that is able to do this. The formulation for a strictly linear controller contains only continuous variables and is thus fairly simple and quick to solve. As was proven in Section 4.3.2, allowing the input to saturate can only result in a better or identical solution to the linear problem. Therefore, the optimum found by solving the case without saturation handling can provide a lower or upper bound on the possible solution, depending on whether the problem is maximized or minimized respectively. Thus, since "branch and bound" techniques employ fathoming, all nodes that have a value worse than this can immediately be eliminated.

The method is applied to the same problem above and the results are reported in Table 4.3. The results show only a small decrease in the amount of time needed to solve the problem, however, this could potentially aid the solver in finding an initial feasible integer solution.

If there are MINLP techniques which are able to fathom the solution to a node, then potentially a worst-case NLP node strategy could be used provided that it can be shown that the NLP solution is the global optimum.

4.5 Summary

In this chapter, an algorithm was presented that was able to reduce the number of integer variables required to represent saturation at the input bounds for a mixed-integer linear programming problem. The proof of the algorithm is based on convexity of linear systems and can therefore be extended to nonlinear systems that display this property. However, it is only able to obtain a local optimum for nonconvex mixed-integer nonlinear problems.

A second method that utilizes the solution to the strictly linear controller as an upper or lower bound, was also presented. The proof of this method also depends on the convexity of the system.

Chapter 5

Case Studies

5.1 Case Study 1 - A Stirred Tank Reactor

The first case study is concerned with the design of a single continuous stirred tank reactor (CSTR). CSTRs are frequently found in industry and it is fairly straightforward to model the process of reactions and heat exchange taking place using first principles, thus this serves as an example of a simple nonlinear dynamic process design.

The reaction taking place in this example is a first order, exothermic irreversible reaction ($A \rightarrow B$). The rate of reaction in the CSTR is dependent on the reactor volume, and hence the height and diameter of the tank, as well as the temperature and reactant concentration inside the reactor. The height and diameter of the tank are two examples of design variables that can be included in the optimization problem as decision variables. It is assumed that the reactor is perfectly mixed, thus the temperature of the mixture and the concentration of reactants are uniform throughout the tank. A perfectly mixed, cylindrical cooling jacket surrounds the walls of the reactor to maintain the temperature inside the tank. It is also assumed that the volume and density of the reactor and jacket contents remain constant.

The variable to be controlled is the temperature of the CSTR, and the manipulated variable which achieves this is the flowrate of cooling water through the cooling jacket. The disturbance to the process is an increase in the temperature of the feed into the reactor.

The setpoint of the reactor temperature is taken as its steady state value, which is included in the problem as one of the decision variables. The steady state temperature, in turn, is determined by the steady state flowrate of cooling water through the reactor, which is also included as a decision variable. Other design variables included are the controller gain and controller time constant. The steady state flowrate of cooling water is used to determine the utility cost of the CSTR over 4 years of operation. Together with the capital cost, which is a function of the diameter and height of the tank, this is used to calculate the total cost, which is the economic criterion that is to be minimized.

5.1.1 Mathematical Model

The mathematical model of the design problem follows. The model parameters are taken from the example given by Schweiger and Floudas (1998), and the model itself is based on the "multiple CSTRs in series" example by Luyben and Luyben (1997). The model parameters and bounds have been modified to demonstrate the advantages of including input saturation. Throughout the formulation the subscripts R and J are used to denote a variable that pertains to the reactor or jacket respectively. The indices k, i, j indicate the particular control element, finite element and collocation point respectively, while a subscript of ss is used to denote the steady state value.

Total Cost

It is assumed that the period of operation of the process is 4 years, to be able to calculate the total cost.

$$\text{cost}_{\text{tot}} = \text{cost}_{\text{cap}} + \beta_{\text{pay}} \times \text{cost}_{\text{util}}$$

Utility Cost ($/yr)

$$\text{cost}_{\text{util}} = 8760 \times c_{\text{CW}} \times F_{J,\text{ss}}$$

Capital Cost

(Schweiger and Floudas, 1998)

$$\text{cost}_{\text{cap}} = 1916.9 D_R^{1.066} H_R^{0.802}$$

Mass Balance in CSTR

$$V_R \frac{dC_R}{dt} = C_{R,\text{in}} F_R - C_R F_R - k_1 V_R C_R$$

Energy Balance in CSTR

$$V_R \frac{dT_R}{dt} = T_{R,\text{in}} F_R - T_R F_R - \frac{\Delta H}{\rho C_p} k_1 V_R C_R - \frac{U}{\rho C_p} A_R \left(T_R - T_J\right)$$

Energy Balance in Cooling Jacket

$$V_J \frac{dT_J}{dt} = T_{J,\text{in}} F_J - T_J F_J + \frac{U}{\rho_J C_J} A_R \left(T_R - T_J\right)$$

Reaction Rate

$$k_1 = k_0 \times \exp\left(\frac{-E}{RT_R}\right)$$

Reactor Volume

$$V_R = \frac{\pi}{4} D_R^2 H_R$$

Reactor Surface Area

$$A_R = \pi D_R H_R$$

Cooling Jacket Volume

A 4-inch clearance for the jacket is assumed in order to calculate the volume of the cooling jacket (Schweiger and Floudas, 1998).

$$V_J = \frac{1}{3} A_R$$

Control Equation

$$\varepsilon(t) = T_{R,ss} - T_R(t)$$

$$F_J(t) = F_{J,ss} + K_c \left(\varepsilon(t) + \frac{1}{\tau_I} \int \varepsilon(t) \right)$$

In addition, there is an 18 minute deadtime on the input into the process. Since the control time step fits exactly into the time delay, the time delay can be represented directly with a lag θ on the control element. Alternatively the delay can be approximated using a dynamic equation; however, this would be handled using OCFE and can result in at least twice as many equations. The case study in Section 5.2 has a dynamic lag that is handled using OCFE.

5.1.2 Optimization Formulation

The dynamic model is then transformed into a nonlinear algebraic problem including rigorous input saturation handling using the optimization superstructure of Equation (3.12). The resulting optimization problem formulation is given by:

$$\min_{\substack{D_R \; H_R \; K_c \; \tau_I F_{J,\mathrm{ss}} \; T_{J,\mathrm{ss}} \; T_{R,\mathrm{ss}} \\ C_{R,\mathrm{ss}} \; F_{J,kij} \; T_{J,kij} \; T_{R,kij} \; C_{R,kij}}} \quad \mathrm{cost}_{\mathrm{tot}} = \mathrm{cost}_{\mathrm{cap}} + \beta_{\mathrm{pay}} \times \mathrm{cost}_{\mathrm{util}}$$

subject to

$$\begin{aligned}
\text{cost}_{\text{cap}} &= 1916.9 D_R^{1.066} H_R^{0.802} \\
\text{cost}_{\text{util}} &= 8760 \times c_{CW} \times F_{J,\text{ss}} \\
k_{1,kij} &= k_0 \times \exp\left(\frac{E_a}{R T_{R,kij}}\right) \\
A_R &= \pi D_R H_R \\
V_J &= \frac{1}{3} A_R \\
\varepsilon_k &= T_{R,\text{ss}} - T_{R,k10}
\end{aligned}$$

Through the introduction of slack variables to allow input saturation, the control equation becomes:

$$\begin{aligned}
F_{Jc,k10} - S_k^L + S_k^U &= F_{Jc,\text{ss}} + K_c\left(\varepsilon_k - \varepsilon_{(k-1)} + \frac{\Delta t}{\tau_I}\varepsilon_k\right) \\
F_{Jc,k10} &= F_{Jc,kij} \\
F_{J,kij} &= F_{Jc,(k-\theta),ij} \\
F_J^L - \beta\left(1 - Z_k^L\right) \le F_{J,k10} &\le F_J^L + \beta\left(1 - Z_k^L\right) \\
F_J^U - \beta\left(1 - Z_k^U\right) \le F_{J,k10} &\le F_J^U + \beta\left(1 - Z_k^U\right) \\
0 \le S_k^L &\le \beta Z_k^L \\
0 \le S_k^U &\le \beta Z_k^U
\end{aligned}$$

Discretizing the dynamic equations using OCFE allows the state equations for C_R, T_R and T_J to be represented by:

$$\begin{aligned}
\sum_{n=1}^{\text{nCOL}} C_{R,kin}\dot{\phi}_n(\gamma_j) &= \frac{\Delta\delta}{V_R}\left(C_{R,\text{in}}F_R - C_{R,kij}F_R - k_{1,kij}V_R C_{R,kij}\right) \\
\sum_{n=1}^{\text{nCOL}} T_{R,kin}\dot{\phi}_n(\gamma_j) &= \frac{\Delta\delta}{V_R}\left(T_{R,\text{in}}F_R - T_{R,kij}F_R - \frac{\Delta H}{\rho C_p}k_{1,kij}V_R C_{R,kij}\right. \\
&\qquad \left. - \frac{U}{\rho C_p}A_R\left(T_{R,kij} - T_{J,kij}\right)\right) \\
\sum_{n=1}^{\text{nCOL}} T_{J,kin}\dot{\phi}_n(\gamma_j) &= \frac{\Delta\delta}{V_J}\left(T_{J,\text{in}}F_{J,kij} - T_{J,kij}F_{J,kij} + \frac{U}{\rho_J C_J}A_R\left(T_{R,kij} - T_{J,kij}\right)\right)
\end{aligned}$$

where

$$k_{1,kij} = k_0 \times \exp^{\left(\frac{E_a}{RT_{R,kij}}\right)}$$

To ensure state continuity between finite elements and control elements the following constraints are added:

$$\begin{aligned}
C_{R,ki0} &= \sum_{n=1}^{\text{nCOL}} C_{R,k,(i-1),n}\phi_n\,(\gamma = 1) \\
C_{R,k10} &= \sum_{n=1}^{\text{nCOL}} C_{R,(k-1),nFE,n}\phi_n\,(\gamma = 1) \\
T_{R,ki0} &= \sum_{n=1}^{\text{nCOL}} T_{R,k,(i-1),n}\phi_n\,(\gamma = 1) \\
T_{R,k10} &= \sum_{n=1}^{\text{nCOL}} T_{R,(k-1),nFE,n}\phi_n\,(\gamma = 1) \\
T_{J,ki0} &= \sum_{n=1}^{\text{nCOL}} T_{J,k,(i-1),n}\phi_n\,(\gamma = 1) \\
T_{J,k10} &= \sum_{n=1}^{\text{nCOL}} T_{J,(k-1),nFE,n}\phi_n\,(\gamma = 1)
\end{aligned}$$

These are applied over $k = 2, ..., \text{nCE}$ and $i = 2, ..., \text{nFE}$.

The steady state equations are then solved to attain a feasible steady state operating point:

$$\begin{aligned}
0 &= C_{R,\text{in}}F_R - C_{R,\text{ss}}F_R - k_{1,\text{ss}}V_R C_{R,\text{ss}} \\
0 &= T_{R,\text{in}}F_R - T_{R,\text{ss}}F_R - \frac{\Delta H}{\rho C_p}k_{1,\text{ss}}V_R C_{R,\text{ss}} - \frac{U}{\rho C_p}A_R\,(T_{R,\text{ss}} - T_{J,\text{ss}}) \\
0 &= T_{J,\text{in}}F_{J,\text{ss}} - T_{J,\text{ss}}F_{J,\text{ss}} + \frac{U}{\rho_J C_J}A_R\,(T_{R,\text{ss}} - T_{J,\text{ss}})
\end{aligned}$$

where

$$k_{1,\text{ss}} = k_0 \times \exp\left(\frac{E_a}{RT_{R,\text{ss}}}\right)$$

These steady state values are then used as the initial values for the dynamic equations by the inclusion of the following equality constraints:

$$\begin{aligned} C_{R,\text{ss}} &= C_{R,1,10} \\ T_{R,\text{ss}} &= T_{R,1,10} \\ T_{J,\text{ss}} &= T_{J,1,10} \end{aligned}$$

The tolerance constraints are then applied over $k = (\text{nCE} - \text{nTOL}), ..., \text{nCE}$ to force the trajectory to approach the set point at the end of the time horizon.

$$-\text{tol} \leq \varepsilon_k \leq \text{tol} \tag{5.1}$$

In addition, constraints on the upper and lower values of the reactor and jacket temperatures as well as coolant flowrate are added to demonstrate the advantages of including input saturation:

$$\begin{aligned} 600\ ^\circ\text{R} &\leq T_R \leq 650\ ^\circ\text{R} \\ 600\ ^\circ\text{R} &\leq T_J \leq 650\ ^\circ\text{R} \\ 450\ \text{ft}^3/\text{hr} &\leq F_J \leq 500\ \text{ft}^3/\text{hr} \end{aligned}$$

A breakdown and description of each of the variables is given in Table 5.1, the values of the parameters used in the design are listed in Table 5.2, and a summary of additional run parameters are listed in Table 5.3. Note that this is a nonlinear nonconvex problem.

5.1.3 Results and Discussion

The resulting nonlinear programming problem is first solved without input saturation and is then compared to the case where input saturation has been rigorously handled. The results show that not only is it possible to design CSTR that is able to handle a step increase of 7.5 % increase in the temperature of the feed, but that a plant design that allows saturation reduces the total cost by 18 % operating over the same period than one which doesn't allow saturation. This shows that the purely linear controller is overly pessimistic and that a cheaper design alternative

Table 5.1: Summary of variables for CSTR design

Time varying	Description
C_R	reactor outlet concentration
T_R	reactor outlet temperature
F_J	jacket coolant flowrate
T_J	jacket outlet temperature
k_1	reaction rate
ε	reactor temperature error
Time invariant	**Description**
$T_{R,ss}$	steady state reactor temperature
V_R	volume of reactor
A_R	reactor heat exchange area
D_R	diameter of reactor
H_R	height of reactor
V_J	volume of cooling jacket
$F_{J,\text{ss}}$	steady state cooling water flowrate
K_c	controller gain
τ_I	controller time constant
Saturation	**Description**
S^L	lower slack variable
S^U	upper slack variable
Z^L	lower binary variable
Z^U	upper binary variable

Table 5.2: Parameters for CSTR design

Parameter	Value		Description
ΔH	-30 000	Btu/lbmol	heat of reaction
U	300	Btu/hr ft^2 °R	heat transfer coefficient
E_a/R	15075	°R	activation energy
k_0	4.08×10^{10}	hr^{-1}	kinetic rate constant
ρ	50	lb/ft^3	liquid density
ρ_J	62.3	lb/ft^3	coolant density
C_p	0.75	Btu/lb °R	liquid heat capacity
C_J	1.0	Btu/lb °R	coolant heat capacity
F_R	100	ft^3/hr	feed flowrate
$C_{R,\text{in}}$	1	lbmol A/ft^3	feed composition
$T_{R,\text{in}}$	600	°R	feed temperature
$T_{J,\text{in}}$	530	°R	coolant inlet temperature
c_{CW}	3.74×10^{-3}	\$/ft^3	coolant cost

Table 5.3: Run-specific information for CSTR design

Parameter	Value		Description
t_0	0	hrs	start time horizon
t_f	10	hrs	end of time horizon
Δt	0.1	hrs	controller time step
t_d	0.5	hrs	time of disturbance step
$\Delta T_{R,\text{in}}$	45	°R	feed temperature step increase
tol	$\pm 1 \times 10^{-6}$		reactor temperature error at end points
nCE	100		number of control elements
nFE	1		number of finite elements per CE
nCOL	2		number of collocation points per FE
nTOL	10		number of end tolerance points

which also takes dynamic performance into account can be found if saturation is allowed.

Table 5.4 shows the difference in results between the purely linear controller and the case with input saturation. In the latter, the optimization routine reduces the size of the CSTR but increases the steady-state cooling water flowrate, since there is an economic trade-off between the two factors.

The larger tank diameter required by the linear controller can be explained as follows. The increase in reactant feed temperature causes the reactor temperature to increase. The controller therefore increases the flow of cooling water through the jacket, which reduces the temperature in the reactor and returns it to its setpoint. However, the control valve for the cooling water has an operating range of 450 - 500 ft^3/hr, which limits the available cooling water. The linear controller is forced to handle this by only allowing the valve to remain at its maximum for an instant. However, allowing the valve to remain in that position longer would also allow the temperature to return to its setpoint (assuming identical plants), and thus provide adequate control. The disadvantage is that now the steady-state cooling water flowrate is forced lower in order to accommodate the limits on the flow. Since the optimization routine forces the steady state cooling water flowrate to the lower bound for the case were saturation is ignored, this means that an increased heat transfer area is required in order to achieve the same amount of cooling, therefore the diameter and/or height of the tank is forced to increase.

The cooling water flowrate saturates at its upper bound in Figure 5.1, and its effect on the controlled variable, T_R, can be seen in Figure 5.2. The dynamics of

Table 5.4: Solution results for CSTR design

Variable	No Saturation	Saturation
Total Cost	366778	299664
Capital Cost	307792	239755
Utility Cost	58986	59909
D_R	36.71	25.33
H_R	4.68	5.61
V_R	4954	2829
$T_{R,\mathrm{ss}}$	647.3	648.62
$F_{J,\mathrm{ss}}$	450	457
K_c	-51.91	-18.76
τ_I	1.538	1.451

the other state variables are shown in the graphs of Figures 5.3 and 5.4.

Different initial values were chosen for the nonlinear optimization problem dealing with a purely linear controller, and the resultant solution was the same for all, thus the value given for the objective is probably the global minimum.

The problem with input saturation has 200 integer variables and was solved on a PIII Xeon 450 MHz computer with 500 MB RAM running Linux using DICOPT as a MINLP via GAMS. The heuristic "stop when the NLP worsens" was used. However, even this yielded no feasible integer solution after two days and it was decided to evaluate the MINLP using the algorithm described in Section 4.3.1. The solution is a local minimum and it is not guaranteed to be the global minimum, but it is still a reduction in the total cost compared to the purely linear controller, illustrating that including input saturation provides a less economically pessimistic design for cases where the dynamics of the design are important.

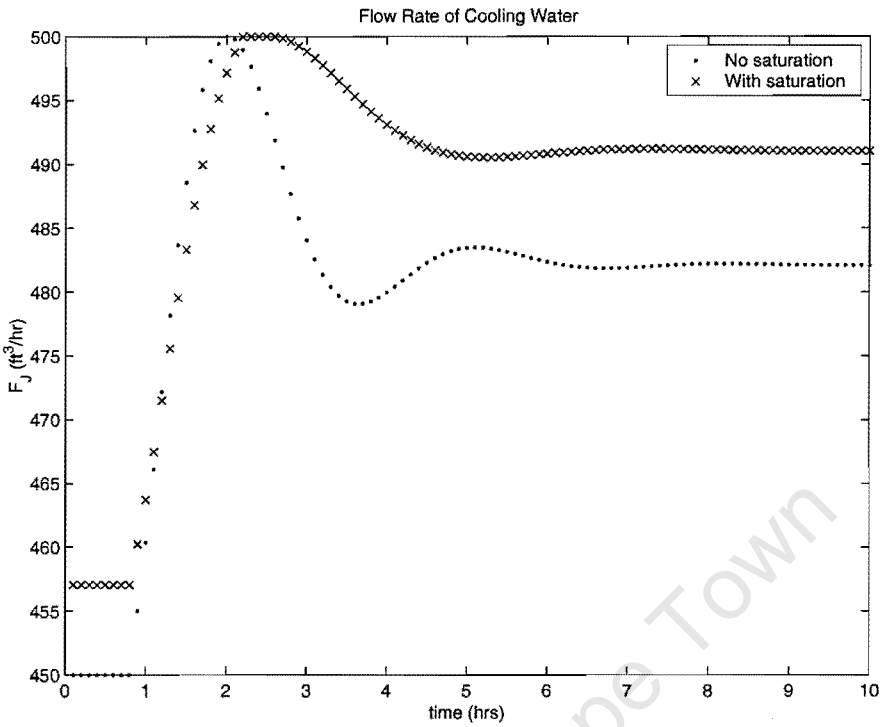


Figure 5.1: Comparison of F_J profile with and without saturation

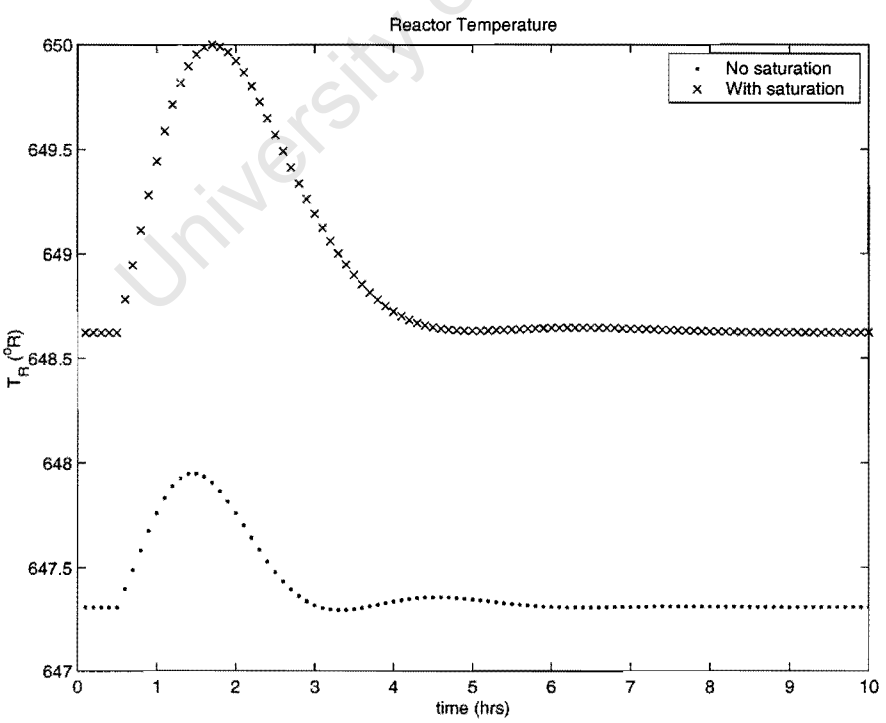


Figure 5.2: Comparison of T_R profile with and without saturation

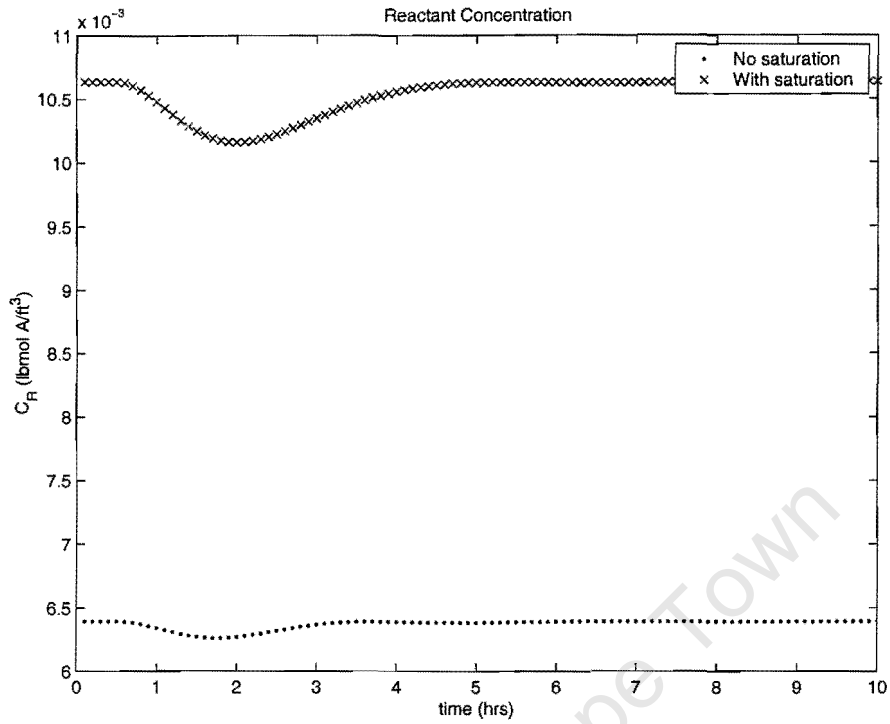


Figure 5.3: Comparison of C_R profile with and without saturation

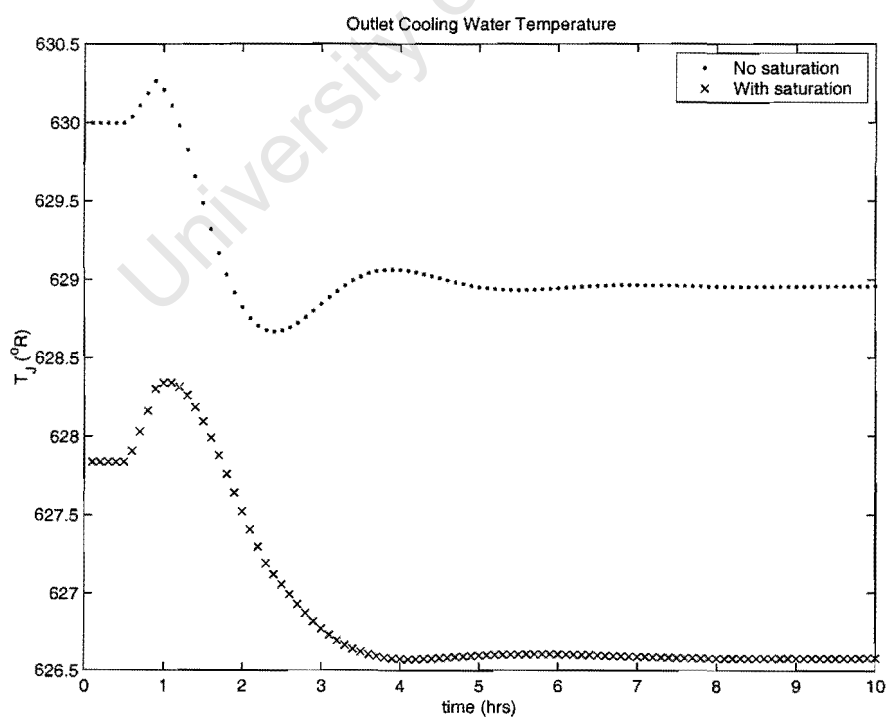


Figure 5.4: Comparison of T_J profile with and without saturation

5.2 Case Study 2 - A Binary Distillation Column

The second case study is the design of an ideal binary distillation column. The purpose of the column is to separate a mixture at its bubble point into distillate and bottom products of a specified purity. The case study demonstrates how a distillation column that achieves specified product quality can be designed while taking into account dynamic operability considerations, and the application of the formulation to a multi-output system. It also demonstrates that allowing for the phenomenon of input saturation, results in a less pessimistic economic objective.

The diameter of the column is an example of a characteristic that will be designed by minimizing the annualized cost of the column over a specific payback period. The reflux and vapor boilup capacities are designed on the basis of the steady state values, which are also included as decision variables. The liquid and vapor compositions and flowrates of each tray in the column are variable and time-dependent. The process is controlled using a proportional integral (PI) control scheme where the distillate composition is controlled by the reflux rate and the bottoms composition is controlled by the vapor boilup. The disturbance being investigated is a step change in the feed composition to the column.

The following assumptions were made in order to simplify the development of the mathematical model that describes the design of the column.

- The mixture has a constant relative volatility throughout the column and the trays are 100% efficient. This implies that the vapor leaving the tray is in equilibrium with the liquid on the tray, which allows the following simple vapor - liquid equilibrium relationship to be used:

$$y^n = \frac{\alpha x^n}{1 + (\alpha - 1)\, x^n}$$

  where x^n is the liquid composition on the nth tray (mole fraction more volatile component), y^n is the vapor composition on the nth tray (mole fraction more volatile component), and α is the relative volatility of the mixture.

- Equimolal overflow (the result is that an energy balance for each tray is not required).

- Total condenser.

There is a first-order lag between the calculated control move and the actual control level, and a 5-minute deadtime on the composition measurement of the bottoms and distillate is included. A fifth order system is used to approximate this deadtime.

The list of variables has been summarized in Table 5.5.

5.2.1 Mathematical Model

The mathematical model for the binary distillation column which follows comes from Schweiger and Floudas (1998), which is based on the general model given by Luyben (1990) assuming that the tray holdup is constant and time invariant for all trays.

Condenser and Reflux Drum

- Total continuity

$$\beta_D \frac{dD}{dt} = V - R - D$$

- Component continuity (more volatile component)

$$M_D \frac{dx^D}{dt} = V\left(y^{nT} - x^D\right)$$

Top Tray ($n = nT$)

- Total continuity

$$\beta \frac{dL^{nT}}{dt} = R - L^{nT}$$

- Component continuity

$$M \frac{dx^{nT}}{dt} = R\left(x^D - x^{nT}\right) + V\left(y^{nT-1} - y^{nT}\right)$$

- Equilibrium

$$y^{nT} = \frac{\alpha x^{nT}}{1 + (\alpha - 1)\, x^{nT}}$$

Table 5.5: Summary of variables for binary distillation design

Time invariant	Description
M	tray liquid holdup
M_B	reboiler holdup
M_D	condenser holdup
β	hydraulic tray time constant
β_B	reboiler time constant
β_D	condenser time constant
x_{ss}^n	steady state liquid composition on tray n
y_{ss}^n	steady state vapor composition on tray n
L_{ss}^n	steady state liquid flowrate on tray n
B_{ss}	steady state bottoms flowrate
D_{ss}	steady state distillate flowrate
R_{ss}	steady state reflux ratio
V_{ss}	steady state vapor boilup
D_c	column diameter
K_V	vapor boilup controller gain
K_R	reflux rate controller gain
τ_V	vapor boilup controller time constant
τ_R	vapor boilup controller time constant
Time variant	**Description**
x^n	liquid composition on tray n
y^n	vapor composition on tray n
L^n	liquid flowrate on tray n
B	bottoms flowrate
D	distillate flowrate
$x^{B,\mathrm{m}}$	measured bottom composition
$x^{D,\mathrm{m}}$	measured distillate composition
ε^B	error in bottoms composition
ε^D	error in distillate composition
V	vapor boilup flowrate
R	reflux flowrate

Tray (n)

- Total continuity

$$\beta \frac{dL^n}{dt} = L^{n+1} - L^n$$

- Component continuity

$$M \frac{dx^n}{dt} = L^{n+1}\left(x^{n+1} - x^n\right) + V\left(y^{n-1} - y^n\right)$$

- Equilibrium

$$y^n = \frac{\alpha x^n}{1 + (\alpha - 1)\, x^n}$$

Feed Tray $(n = nF)$

- Total continuity

$$\beta \frac{dL^{nF}}{dt} = L^{nF+1} - L^{nF} + F$$

- Component continuity

$$M \frac{dx^{nF}}{dt} = L^{nF+1}\left(x^{nF+1} - x^{nF}\right) + V\left(y^{nF-1} - y^{nF}\right) + F\left(z - x^{nF}\right)$$

- Equilibrium

$$y^{nF} = \frac{\alpha x^{nF}}{1 + (\alpha - 1)\, x^{nF}}$$

Tray 1

- Total continuity

$$\beta \frac{dL^1}{dt} = L^2 - L^1$$

- Component continuity

$$M \frac{dx^1}{dt} = L^2\left(x^2 - x^1\right) + V\left(y^B - y^1\right)$$

- Equilibrium

$$y^1 = \frac{\alpha x^1}{1 + (\alpha - 1)\, x^1}$$

Reboiler

- Total continuity

$$\beta_B \frac{dB}{dt} = L^1 - V - B$$

- Component continuity

$$M_B \frac{dx^B}{dt} = L^1 \left(x^1 - x^B\right) + V \left(x^B - y^B\right)$$

- Equilibrium

$$y^B = \frac{\alpha x^B}{1 + (\alpha - 1)\, x^B}$$

Measurement Lags

These represent lags for a 5-minute delay by means of a fifth-order system (Schweiger and Floudas, 1998).

- Bottoms composition

$$\frac{d^5 x^{B,\mathrm{m}}}{dt} + 5\frac{d^4 x^{B,\mathrm{m}}}{dt} + 10\frac{d^3 x^{B,\mathrm{m}}}{dt} + 10\frac{d^2 x^{B,\mathrm{m}}}{dt} + 5\frac{dx^{B,\mathrm{m}}}{dt} + x^{B,\mathrm{m}} = x^B$$

- Distillate composition

$$\frac{d^5 x^{D,\mathrm{m}}}{dt} + 5\frac{d^4 x^{D,\mathrm{m}}}{dt} + 10\frac{d^3 x^{D,\mathrm{m}}}{dt} + 10\frac{d^2 x^{D,\mathrm{m}}}{dt} + 5\frac{dx^{D,\mathrm{m}}}{dt} + x^{D,\mathrm{m}} = x^D$$

Standard PI Controller

It is assumed that the system is controlled with a PI controller that can represented by:

$$\begin{aligned} V_c &= V_{\mathrm{ss}} + K_V \left(\varepsilon^B + \frac{1}{\tau_V} \int \varepsilon^B \, dt\right) \\ R_c &= R_{\mathrm{ss}} + K_R \left(\varepsilon^D + \frac{1}{\tau_R} \int \varepsilon^D \, dt\right) \end{aligned}$$

where

$$\begin{aligned} \varepsilon^B &= x^{B,\mathrm{ss}} - x^{B,\mathrm{m}} \\ \varepsilon^D &= x^{D,\mathrm{ss}} - x^{D,\mathrm{m}} \end{aligned}$$

Final control lags on the vapor boilup and reflux rate are given by:

$$\begin{aligned} 0.9\frac{dV}{dt} &= V_c - V \\ 0.5\frac{dR}{dt} &= R_c - R \end{aligned}$$

Tray Hydraulics

The linearized Francis weir formula was used to calculate the molar hold-ups, M, and tray time constants, β, as a function of the column diameter (Luyben and Floudas, 1994). The tray time constants and molar hold-ups were assumed to be the same for all the trays and time invariant. The reboiler and condenser hold-ups were taken as being 10 times the hold-up of the trays in the column, while the time constants used for the reboiler and condenser were 100 times the size of the normal tray time constant (Schweiger and Floudas, 1998).

- Molar hold-ups

$$M = 7.538115\left(\left(\left(\frac{0.0014134}{D_c}\right)^{\frac{2}{3}}\right) + h_w\right) D_c^2$$

- Time constants

$$\beta = 0.05271 D_c^{1.3333}$$

The above relationships were obtained from Schweiger and Floudas (1998).

Performance Constraints

Flooding in the column is determined by the vapor flowrate inside the column, therefore the diameter of the column has to be sufficiently large. Flooding will

not occur if the constraints in Equations (5.2) and (5.3) are met exactly, thus to ensure flooding does not occur in the column, Equation (5.4) must be satisfied.

$$v - k_v f_f \sqrt{\frac{(\rho_L - \rho_V)}{\rho_V}} = 0 \tag{5.2}$$

$$D_c^2 - \frac{4V_{ss}MW}{\pi v \rho_v} = 0 \tag{5.3}$$

$$D_c > 0.6719\sqrt{V_{ss}} \tag{5.4}$$

Objective Function

The economic objective of the design is the minimization of the annualized cost.

$$\text{cost} = \beta_{\text{tax}}\text{cost}_{\text{util}} + \text{cost}_{\text{cap}}/\beta_{\text{pay}}$$

- Utility cost ($/yr)

$$\text{cost}_{\text{util}} = \left(c_{\text{LPS}}\Delta H_{\text{vap}} + c_{\text{CW}}\Delta H_{\text{cond}}\right) V_{ss}$$

- Capital cost ($) (Luyben and Floudas, 1994)

$$\text{cost}_{\text{cap}} = 12.3\left[615 + 324D_c^2 + 486\left(6 + 0.76N_T\right)D_c\right] + 245N_T\left(0.7 + 1.5D_c^2\right)$$

The values of the parameters β_{tax}, the tax factor, β_{pay}, the payback period, the latent heats of vaporization and condensation, $\Delta H_{\text{vap}}$and ΔH_{cond}, and the utility cost coefficients c_{LPS} and c_{CW} as well as other physical data used in the example are recorded in Table 5.6.

5.2.2 Optimization Formulation

The problem can be reformulated into the optimization framework of Equation (3.12). Using collocation on finite elements allows the state balances for tray n to

Table 5.6: Parameters for binary distillation design

Parameter	Value	
F	1	kmol/min
z_F	0.45	
α	0.98	
k_v	0.106	m/s
f_f	0.8	
MW	92	kg/kmol
ρ_L	883	kg/m^3
ρ_V	2.9	kg/m^3
ΔH_{vap}	0.031	10^6kJ/kmol
ΔH_{cond}	0.032	10^6kJ/kmol
c_{LPS}	6.1×10^5	(\$ min) / (10^6 kJ yr)
c_{CW}	1.5×10^4	(\$ min) / (10^6 kJ yr)
β_{tax}	0.4	
β_{pay}	4	yr

be represented by the following set of equations. The extension to other trays in the column is direct.

$$\sum_{l=0}^{\text{nCOL}} L_{k,il}^{n} \dot{\phi}_l(\gamma_j) = \frac{\Delta\delta}{\beta}\left(L_{k,ij}^{n+1} - L_{k,ij}^{n}\right)$$

$$\sum_{l=0}^{\text{nCOL}} x_{k,il}^{n} \dot{\phi}_l(\gamma_j) = \frac{\Delta\delta}{M}\left(L_{k,ij}^{n+1}\left(x_{k,ij}^{n+1} - x_{k,ij}^{n}\right) + V\left(y_{k,ij}^{n-1} - y_{k,ij}^{n}\right)\right)$$

$$y^n = \frac{\alpha x^n}{1 + (\alpha - 1)\, x^n}$$

where $n = 1, ..., nT$, $k = 1, ...,$ nCFE, $i = 1, ...,$ nFE, $j = 1, ...,$ nCOL and

$$\phi_j(\gamma) = \prod_{\substack{n=0 \\ n \neq j}}^{\text{nCOL}} \frac{(\gamma - \gamma_n)}{(\gamma_j - \gamma_n)}$$

To enforce continuity between control and finite elements the following constraints

are added for $k = 2, ..., nCE$ and $i = 2, .., nFE$:

$$\begin{aligned} L^n_{k,i0} &= \sum_{l=1}^{\text{nCOL}} L^n_{k,(i-1),l}\phi_l\,(\gamma = 1) \\ L^n_{k,10} &= \sum_{l=1}^{\text{nCOL}} L^n_{(k-1),\text{nFE},l}\phi_l\,(\gamma = 1) \end{aligned}$$

The PI control equations are modified to allow for input saturation by the introduction of slack and binary variables.

$$\begin{aligned} \varepsilon^B_k &= x^{B,\text{ss}}_{k,ij} - x^{B,\text{m}}_{k,ij} \\ \varepsilon^D_k &= x^{D,\text{ss}}_{k,ij} - x^{D,\text{m}}_{k,ij} \end{aligned}$$

$$\begin{aligned} V_{c,k,10} - S^L_{V,k} + S^U_{V,k} &= V_{c,(k-1),10} + K_V\left(\varepsilon^B_k - \varepsilon^B_{(k-1)} + \frac{\Delta t}{\tau_V}\varepsilon^B_k\right) \\ R_{c,k,10} - S^L_{R,k} + S^U_{R,k} &= R_{c,(k-1),10} + K_R\left(\varepsilon^D_k - \varepsilon^D_{(k-1)} + \frac{\Delta t}{\tau_R}\varepsilon^D_k\right) \end{aligned}$$

$$\begin{aligned} V_{c,k,10} &= V_{c,k,ij} \\ R_{c,k,10} &= R_{c,k,ij} \end{aligned}$$

$$\begin{aligned} 0 &\le S^L_{V,k} \le \psi z^L_{V,k} \\ 0 &\le S^U_{V} \le \psi z^U_{V,k} \\ 0 &\le S^L_{R,k} \le \psi z^L_{R,k} \\ 0 &\le S^U_{R,k} \le \psi z^U_{R,k} \end{aligned}$$

$$\begin{aligned} V^L_c - \psi\left(1 - z^L_{V,k}\right) &\le V_{c,k,10} \le V^L_c + \psi\left(1 - z^L_{V,k}\right) \\ V^U_c - \psi\left(1 - z^U_{V,k}\right) &\le V_{c,k,10} \le V^U_c + \psi\left(1 - z^U_{V,k}\right) \\ R^L_c - \psi\left(1 - z^L_{R,k}\right) &\le R_{c,k,10} \le R^L_c + \psi\left(1 - z^L_{R,k}\right) \\ R^U_c - \psi\left(1 - z^U_{R,k}\right) &\le R_{c,k,10} \le R^U_c + \psi\left(1 - z^U_{R,k}\right) \end{aligned}$$

where

$S^L_{V,k}$, $S^U_{V,k}$	lower and upper slack for vapor boilup rate at control element k
$S^L_{R,k}$, $S^U_{R,k}$	lower and upper slack for reflux rate at control element k
$z^L_{V,k}$, $z^U_{V,k}$	lower and upper binary variables for vapor boilup rate at control element k
$z^L_{R,k}$, $z^U_{R,k}$	lower and upper binary variables for reflux rate at control element k
ψ	arbitrary large constant

The lags are converted into discrete nonlinear equations by defining variables equivalent to the higher order derivatives until all are ordinary differential equations, and then applying collocation to the resulting first order dynamic equations. Solution of the steady-state balances provide initial values for the dynamic equations.

The disturbance step size is an increase of 0.045 in the feed composition, z, and the constraints on the steady-state product compositions are:

$$\begin{aligned} x_B &\leq 0.03 \\ x_D &\geq 0.94 \end{aligned}$$

To demonstrate the effects of allowing input saturation on the economic objective, the following constraints on the reflux rate and vapor boilup are added.

$$\begin{aligned} 1.30 &\leq V_c \leq 1.54 \\ 0.95 &\leq R_c \leq 0.99 \end{aligned}$$

Additional run-specific data is listed in Table 5.7. Note that this is a nonlinear nonconvex problem.

Table 5.7: Run-specific data for binary distillation design

Parameter	Value		Description
t_0	0	min	start time horizon
t_f	100	min	end of time horizon
Δt	2	min	controller time step
t_d	5	min	time of disturbance step
Δz	0.045		feed composition step increase
tol	$\pm 1 \times 10^{-3}$		composition error at end points
nCE	50		number of control elements
nFE	1		number of finite elements per control element
nCOL	2		number of collocation points per FE
nTOL	5		number of end tolerance points

Table 5.8: Results for binary distillation design

Variable	No Saturation	Saturation
Total Cost	35510.31	35403.37
Utility Cost	10831.6871	10780.5814
Trays, nT	15	15
Feed Tray	5	5
D_c (m)	0.7940	0.7921
R_{ss} (kmol/min)	0.9577	0.9506
V_{ss} (kmol/min)	1.3966	1.3900
K_V	-3.0163	-2.8251
τ_V	7.4297	6.7211
K_R	0.0100	0.0100
τ_R	0.0804	0.1068

5.2.3 Results and Discussion

Results for the design of a distillation column where input saturation has been rigorously handled and the case where saturation has been neglected are reported in Table 5.8. Graphs of the dynamic responses for the two investigations are shown in Figures 5.5, 5.6, 5.7, and 5.8.

From the graphs of the dynamic responses the improvement gained from allowing input saturation is not immediately obvious. However, in Table 5.8 it can be seen that the total cost has been reduced. The improvement was gained by reducing the steady state utility cost, which is achieved by reducing the steady state vapor boilup rate. In order to achieve the same purity specifications for the same disturbance, the reflux rate saturates between 18-30 minutes.

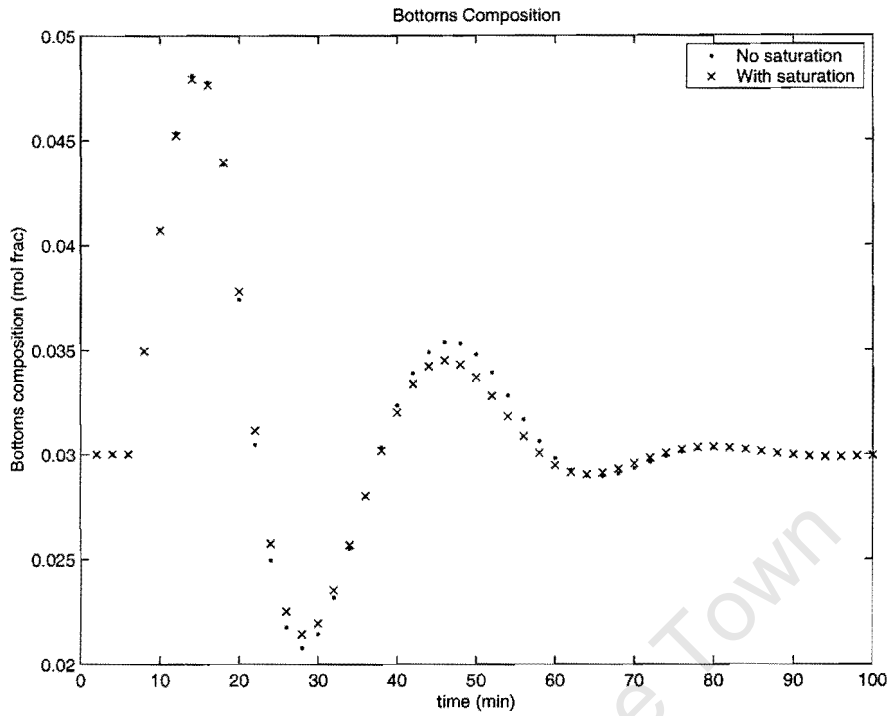


Figure 5.5: Dynamic bottoms composition response with and without input saturation handling

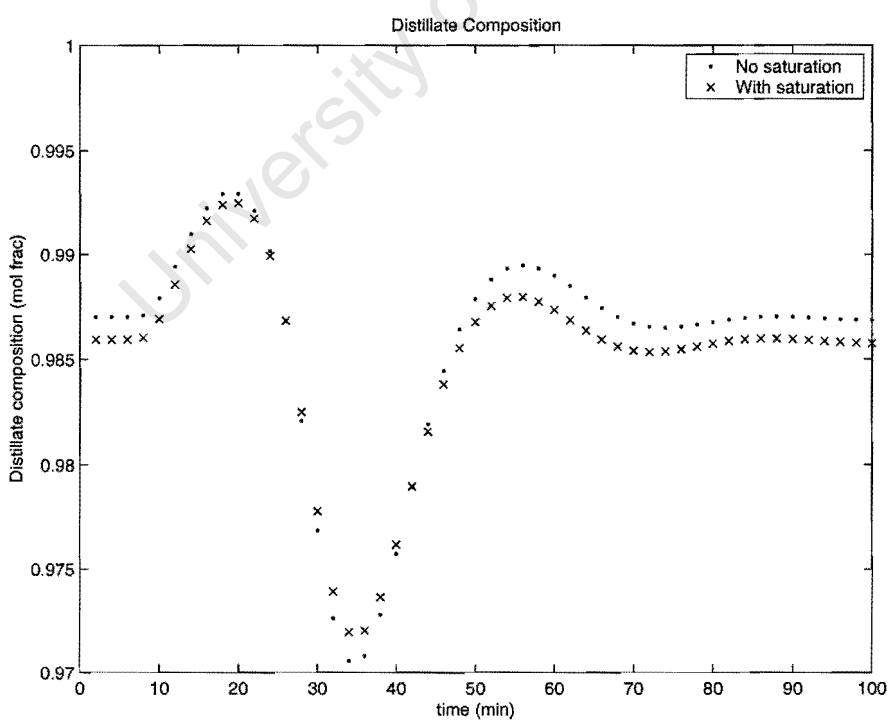


Figure 5.6: Dynamic distillate composition response with and without input saturation handling

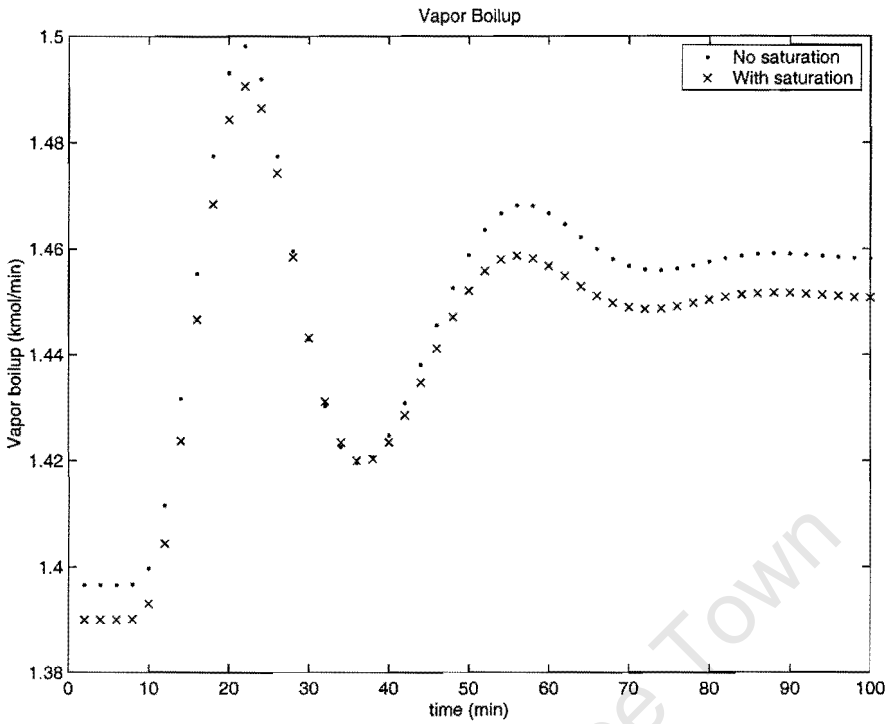


Figure 5.7: Dynamic vapor boilup response with and without input saturation handling

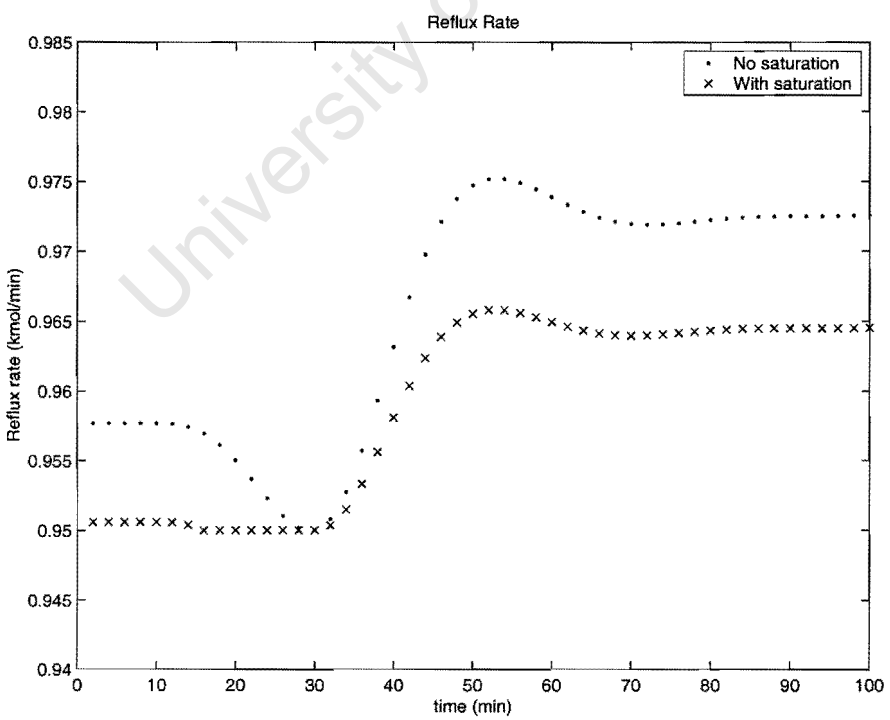


Figure 5.8: Dynamic reflux rate response with and without input saturation handling

The problem was solved using DICOPT calling CONOPT2 as its NLP solver, and Cplex was used as the mixed integer solver on a dual PIII Xeon 450 with 500 MB of RAM. However, numerical difficulties were experienced when the full 300 binary variable problem was attempted, therefore the algorithm described in Section 4.3.1 was used. This method was only able to proceed until four saturation points were defined. On the addition of the fifth saturation point the NLP solver experienced numerical difficulties, and DICOPT terminated without finding an integer feasible solution. This means that there is the potential for the solution to improve with the addition of more saturation points. The solution reported was obtained by manually fixing the integer variables at the points suggested by the aforementioned algorithm, and then solving the nonlinear programming problem with CONOPT2 directly. This means the solution presented is not necessarily integer optimal or a local minimum, but the solution still demonstrates that including input saturation allows the solution to lie closer to the steady-state economic optimum, in comparison to the case where input saturation is neglected. If this is not the local minimum, then further addition of saturation variables would merely have emphasized the improvement that can be gained by the rigorous inclusion of saturation handling.

5.3 Summary

Two case studies in the design of dynamically operable plants with rigorous input saturation handling were presented. The first case study was the design of a CSTR for a fixed step change in the feed temperature, and the second was the design of a binary distillation column with a fixed step increase in the feed composition. The results for the two case studies were reported and compared with the case where a strictly linear controller was used. This showed that the strictly linear controller was overly conservative in its estimate for the capital cost, while the case with input saturation was able to operate closer to the steady-state economic optimum.

Chapter 6

Conclusions and Recommendations

The optimization superstructure of Equation (3.12) is able to synthesize dynamically-operable process designs that rigorously handle input saturation effects. In addition, the formulation can be applied to nonlinear and linear systems through the use of orthogonal collocation on finite elements. It was shown through the use of case studies, that allowing for rigorous input saturation handling in nonlinear systems yields a design which is economically superior in comparison to a design that is based on the use of a strictly linear controller. This implies that by allowing input saturation, the proposed design cost lies closer to economic steady state optimum than a design that ignores saturation, but at the same time the design takes into account deviations from steady state optimum, thereby allowing for increased controllability.

The work in this study is an extension of the work of Young and Swartz (1997) to nonlinear systems. In Young's thesis he recommends that an improved method be found to solve the mixed-integer linear problem which develops as a result of the inclusion of saturation handling in a linear problem. In this thesis, an algorithm has been proposed in Section 4.3.1 that is able to reduce the number of mixed-integer variables required to represent saturation at the input bounds, thereby significantly reducing the computation time required while still providing the same solution which is also the global optimum. It is also described in Section 4.3.4 how this algorithm can be extended for use in nonlinear optimization problems.

Further work in this area lies in the inclusion of robust stability and flexibility analyses in the optimization formulation for the design of robustly stable and

flexible plants that are able to operate over a range of disturbances and are able to cope with model uncertainty. The latter is especially important since, in general, plant models are derived from a combination of first principles and empirical work.

There is also a need to extend the description of the nonlinearity caused by actuator saturation to other linear control systems, such as the Internal Model Control (IMC) framework, so that it can be used in the formulation for the design of economically optimal dynamic plants. Thus the input saturation efects could be compared for designs using a number of different control schemes.

Finally, it is recommended that improved algorithms be found that are able to reliably solve MINLPs, reduce the computation time of mixed-integer nonlinear programming problems, and simultaneously guarantee a global optimum.

References

Bahri, P., Bandoni, A., and Romagnoli, J. (1996). Operability Assessment in Chemical Plants. *Computers and Chemical Engineering*, 20(Suppl.):S787–S792.

Bansal, V., Ross, R., Perkins, J., and Pistikopoulos, E. (2000). The Interactions of Design and Control: Double-Effect Distillation Systems. *Journal of Process Control*, 10:219–227.

Barton, P. I., Allgor, R. J., Feehery, W. F., and Galan, S. (1998). Dynamic Optimization in a Discontinuous World. *Industrial and Engineering Chemistry Research*, 37:966–981.

Bloss, K. F., Biegler, L. T., and Schiesser, W. E. (1999). Dynamic Process Optimization through Adjoint Formulations and Constraint Aggregation. *Industrial and Engineering Chemistry Research*, 38:421–432.

Boyd, S., Barratt, C., and Norman, S. (1990). Linear Controller Design: Limits of Performance Via Convex Optimization. *Proceedings of the IEEE*, 78(3):529–574.

Boyd, S., Venkataramanan, B., Barratt, C., Khraishi, N., Xiaoming, L., Meyer, D., and Norman, S. (1988). A New CAD Method and Associated Architectures for Linear Controllers. *IEEE Transactions on Automatic Control*, 33(3):268–282.

Cao, Y., Biss, D., and Perkins, J. (1994). Assessment of Input-Output Controllability in the Presence of Control Constraints. In *Proceedings of the IFAC Workshop on Integration of Process Design and Control*, pages 35–40. Baltimore.

Cao, Y., Biss, D., and Perkins, J. (1996). Assessment of Input-Output Controllability in the Presence of Control Constraints. *Computers and Chemical Engineering*, 20(4):337–346.

Chenery, S. and Walsh, S. (1998). Process Controllability Analysis using Linear Programming. *Journal of Process Control*, 8(3):165–174.

Cuthrell, J. E. and Biegler, L. T. (1989). Simultaneous optimization and solution methods for batch reactor control profiles. *Computers and Chemical Engineering*, 13(1/2):49–62.

Downs, J. and Doss, J. (1991). Present Status and Future Needs - A View from North American Industry. In Arkun, Y. and Ray, W., editors, *Proceedings of Fourth International Conference on Chemical Process Control CPC IV*, pages 297–318. Elsevier, Amsterdam, The Netherlands.

Doyle, J. C., Smith, R. S., and Enns, D. F. (1987). Control of Plants with Input Saturation Nonlinearities. In *Proceedings of the 1987 American Control Conference*, pages 1034–1039, Minneapolis, Minnesota.

Floudas, C. A. (1995). *Nonlinear and Mixed-Integer Optimization.* Oxford university Press.

GAMS Development Corporation (1999). *GAMS - The Solver Manuals.* GAMS Development Corporation.

Grosdidier, P., Mason, A., Aitolahti, A., Heinonen, P., and Vanhamaki, V. (1993). FCC Unit Reactor-Regenerator Control. *Computers and Chemical Engineering*, 17:165–179.

Holt, B. and Morari, M. (1985a). Design of Resilient Processing Plants - V. The Effect of Deadtime on Dynamic Resilience. *Chemical Engineering Science*, 40(7):1229–1237.

Holt, B. and Morari, M. (1985b). Design of Resilient Processing Plants - VI. The Effect of Righthalf-plane Transmission Zeros on Dynamic Resilience. *Chemical Engineering Science*, 40(1):59–74.

Hovd, M. and Skogestad, S. (1992). Simple Frequency-Dependent Tools for Control System Analysis, Structure Selection and Design. *Automatica*, 28:989–996.

Kothare, M. V., Campo, P. J., Morari, M., and Nett, C. N. (1994). A Unified Framework for the Study of Anti-Windup Designs. *Automatica*, 30(12):1869–1883.

Lewin, D. and Bogle, D. (1996). Controllability Analysis of an Industrial Polymerization Reactor. *Computers and Chemical Engineering*, 20(Suppl.):S871–S876.

Luyben, M. L. and Floudas, C. A. (1994). Analyzing the Interaction of Design and Control - 1. A Multiobjective Framework and Application to Binary Distillation Synthesis. *Computers and Chemical Engineering*, 18(10):933–969.

Luyben, W. L. (1990). *Process Modeling, Simulation and Control for Chemical Engineers.* McGraw-Hill, 2 edition.

Luyben, W. L. and Luyben, M. L. (1997). *Essentials of Process Control.* McGraw-Hill.

Mcfarlane, R., Reineman, R., Bartee, J., and Georgakis, C. (1993). Dynamic Simulator for a Model IV Fluid Catalytic Cracking Unit. *Computers and Chemical Engineering*, 17:275.

Mohideen, M., Perkins, J., and Pistikopoulos, E. (1996). Optimal Design of Dynamic Systems under Uncertainty. *AIChe Journal*, 42(8):2251–2272.

Mohideen, M., Perkins, J., and Pistikopoulos, E. (1997). Robust Stability Considerations in Optimal Design of Dynamic Systems under Uncertainty. *Journal of Process Control*, 7(5):371–385.

Morari, M. (1983). Design of Resilient Processing Plants - III. A General Framework for the Assessment of Dynamic Resilience. *Chemical Engineering Science*, 38:1881–1891.

Mulder, E. F., Kothare, M. V., and Morari, M. (1999). Multivariable Anti-Windup Controller Synthesis using Linear Matrix Inequalities. *Submitted to Automatica.*

Mulder, E. F., Kothare, M. V., and Morari, M. (2000). Multivariable Anti-Windup Controller Synthesis using Bilinear Matrix Inequalities. *Submitted to European Journal of Control.*

Peressini, A., Sullivan, F., and Uhl Jr., J. (1988). *The Mathematics of Nonlinear Programming.* Springer-Verlag, New York.

Perkins, J. and Walsh, P. (1996). Optimization as a Tool for Design/Control integration. *Computers and Chemical Engineering*, 20(4):315–323.

Perkins, J. and Wong, M. (1985). Assessing Controllability of Chemical Plants. *Chemical Engineering Research and Design*, 63:358–362.

Psarris, P. and Floudas, C. (1991a). Dynamic Operability of MIMO Systems with Time Delays and Transmission Zeros - I. Assessment. *Chemical Engineering Science*, 46(10):2691–2707.

Psarris, P. and Floudas, C. (1991b). Dynamic operability of MIMO systems with time delays and transmission zeros - II. enhancement. *Chemical Engineering Science*, 46(10):2709–2728.

Rosenbrock, H. (1970). *State-Space and Multivariable Theory.* Nelson, London, United Kingdom.

Ross, R. (1997). *Dynamic Operability Assessment: A Mathematical Programming Approach Based on Q-Parameterization.* PhD thesis, University of Cape Town.

Ross, R. and Swartz, C. (1995). The Application of Dynamic Operability Assessment to Flotation Circuit Design. In Barker, I., editor, *8th IFAC Annual International Symposium on Automation in Mining, Mineral and Metal Processing.*

Ross, R. and Swartz, C. (1997). Inclusion of Model Uncertainty in a Computational Framework for Dynamic Operability Assessment. *Computers and Chemical Engineering*, 21(Supplement):S415–S420.

Russell, L. and Perkins, J. (1987). Towards a Method for Diagnosis of Controllability Problems in Chemical Plants. *Chemical Engineering Research and Design*, 65:453–461.

Schweiger, C. A. and Floudas, C. A. (1998). *Optimal Control: Theory, Algorithms, and Applications*, chapter Interaction of Design and Control: Optimization with Dynamic Models, pages 388–435. Kluwer Academic Publishers.

Skogestad, S., Hovd, M., and Lundstrom, P. (1991). Towards Integrating Design and Control: Use of Frequency-Dependent Tools for Controllability Analysis. In *PSE '91 4th International Symposium on Process Systems Engineering*, pages III.3.1–III.3.15. Montebello, Quebec, Canada.

Skogestad, S. and Morari, M. (1987). Design of Resilient Processing Plants - IX. Effect of Model Uncertainty on Dynamic Resilience. *Chemical Engineering Science*, 42(7):1765–1780.

Soroush, M. (1994). Evaluation of Achievable Control Quality in Nonlinear Processes. In *IFAC Workshop on Integration of Process Design and Control*, pages 41–46. Baltimore, Maryland, USA.

Swartz, C. (1996). A Computational Framework for Dynamic Operability assessment. *Computers and Chemical Engineering*, 20(4):365–371.

Trierweiler, J. and Engell, S. (1997). The Robust Performance Number: A New Tool for Control Structure Design. *Computers and Chemical Engineering*, 21(Suppl.):S409–S414.

Villadsen, J. V. and Michelsen, M. L. (1978). *Solution of Differential Equation Models by Polynomial Approximation.* Prentice-Hall.

Webers, K. and Engell, S. (1996). Controller Structure Evaluation by Convex Optimization. *Computers and Chemical Engineering*, 20(Suppl.):S949–954.

Weitz, O. and Lewin, D. (1996). Dynamic Controllability and Resiliency Diagnosis using Steady State Process Flowsheet Data. *Computers and Chemical Engineering*, 20(4):325–335.

Young, J. C. C. and Swartz, C. L. E. (1997). Input Saturation Effects in Optimizing Control. In *AIChe Annual Meeting.*

Zheng, A., Kothare, M. V., and Morari, M. (1994). Anti-Windup Design for Internal Model Control. *International Journal of Control*, 60(5):1015–1024.

Ziegler, J. and Nichols, N. (1943). Process Lags in Automatic-Control Circuits. *Transactions of the American Society of Mechanical Engineers*, 65:433–444.

Appendix A

Program Code

```
$TITLE      Grosdidier's FCCU example
$OFFUPPER
$OFFSYMXREF OFFSYMLIST
$OFFDIGIT

*=============================================================================

SETS    K       equation # (max 10)             / K1*K10 /
        I       finite elements # (max 20)      / I1*I200 /
        J       collocation coeff. #            / J1*J5 /
        COL     # possible coll pt (max 3)      / C1*C3 /

ALIAS   (K,KK,KS,KY,KU,KD)
        (J,JP,JJ,JS) ;

*=============================================================================

SCALARS     NK      actual # of state equations     /10/
            NY      actual # of ouptut var.         /2/
            NU      actual # of manip. var.         /3/
            ND      actual # of dist. var.          /1/
            NFE     actual # of FE used             /100/
            NLIM    actual # points inside envelope /95/
            NCOL    actual # coll. pts used         /2/
            ;

SCALAR      NCOF    equal to ncol+1 ;
            NCOF = ncol + 1 ;

SCALAR      NCOT    equal to ncol+2 ;
            NCOT = ncol + 2;

*=============================================================================

ABORT $ (nk GT 10)      "Error in defining NK   (max 10)", nk   ;
ABORT $ (ny GT 10)      "Error in defining NY   (max 10)", ny   ;
ABORT $ (nu GT 10)      "Error in defining NU   (max 10)", nu   ;
ABORT $ (nd GT 10)      "Error in defining ND   (max 10)", nd   ;
ABORT $ (nfe GT 200)    "Error in defining NFE  (max 200)", nfe ;
ABORT $ (ncol GT 3)     "Error in defining NCOL (max  3)", ncol ;

*=============================================================================

* File Declaration

FILE    fcculpout       /colfcculp.xls/
        fccumip1out     /colfccumip1.xls/
        fccumip2out     /colfccumip2.xls/
        fccumip3out     /colfccumip3.xls/
        fccublpout      /colfccublp.xls/
        fccuwhileout    /colfccuwhile.xls/
        fccucutout      /colfccucut.xls/
        opt             /cplex.opt/
        ;


PARAMETER
        YOUT(k,i)   absolute y measurement  for output
        UOUT(k,i)   absolute u measurements for output
        ;

*=============================================================================

SET SXCOL(k,i,jp)       actual dim of coll. coeff. (XCOL)   ;
    SXCOL(k,i,jp)   =   YES $ ( (ORD(k) LE nk) $ (ORD(i) LE nfe)
                            $ (ORD(jp) LE ncof) ) ;
```



```
SET SY(ky,i,j)          actual dim of control var. ;
    SY(ky,i,j)      =   YES $ ( (ORD(ky) LE ny) $ (ORD(i) LE nfe)
                            $ (ORD(j) LE ncof) ) ;

SET SU(ku,i,j)          actual dim of manip. var. ;
    SU(ku,i,j)      =   YES $ ( (ORD(ku) LE nu) $ (ORD(i) LE nfe)
                            $ (ORD(j) GT 1) $ (ORD(j) LE ncof) ) ;
***

SET SNFE(i)             number of finite elements ;
    SNFE(i)         =   YES $ (ORD(i) LE nfe) ;

SET SYSET(k)            actual dim of setpoints ;
    SYSET(k)        =   YES $ ( ORD(k) LE ny ) ;

SET SKP(ky,ku)          actual dim of process gains ;
    SKP(ky,ku)      =   YES $ ( (ORD(ky) LE ny) $ (ORD(ku) LE nu) ) ;

SET SKD(ky,kd)          actual dim of disturbance gains ;
    SKD(ky,kd)      =   YES $ ( (ORD(ky) LE ny) $ (ORD(kd) LE nd) ) ;

SET SD(kd)              actual dim of disturbance ;
    SD(kd)          =   YES $ ( (ORD(kd) LE nd) ) ;

SET SDTMP(kd,i,j)       actual dim of disturbance vector ;
    SDTMP(kd,i,j)   =   YES $ ( (ORD(kd) LE nd) $(ORD(i) LE nfe)
                            $ (ORD(j) LE ncof) ) ;

SET SPHIPR(j,jp)        actual dim of PHIPR ;
    SPHIPR(j,jp)    =   YES $ ( (ORD(j) GT 1) $ (ORD(j) LE ncot)
                            $ (ORD(jp) LE ncof) ) ;

SET SDPHI(jp)           actual dim of denominator of PHI ;
    SDPHI(jp)       =   YES $ ( (ORD(jp) LE ncof) ) ;

SET SRES(k,i,j)         actual dim of residual eq ;
    SRES(k,i,j)     =   YES $ ( (ORD(k) LE nk) $ (ORD(i) LE nfe)
                            $ (ORD(j) GT 1) $ (ORD(j) LE ncof) )  ;

SET SALF(i)             actual dim of alpha ;
    SALF(i)         =   YES $ (ORD(i) LE nfe) ;

SET SCONT(k,i)          actual dim of continuity eq ;
    SCONT(k,i)      =   YES $ ( (ORD(k) LE nk) $ (ORD(i) GT 1)
                            $ (ORD(i) LE nfe) );

SET SYOUT(ky,i,j)       actual dim of output equation ;
    SYOUT(ky,i,j)   =   YES $ ( (ORD(ky) LE ny) $ (ORD(i) LE nfe)
                            $ (ORD(j) LE ncof) )  ;

SET SYINI(k)            actual dim of initial variabl;
    SYINI(k)        =   YES $ ( (ORD(k) LE ny) )  ;

SET SUINI(k)            actual dim of initial variabl;
    SUINI(k)        =   YES $ ( (ORD(k) LE nu) )  ;

SET SYB(ky)             actual dim of bounds on cont. var ;
    SYB(ky)         =   YES $ ( (ORD(ky) LE nk) )  ;

SET SUB(ku)             actual dim of bounds on manip. var ;
    SUB(ku)         =   YES $ ( (ORD(ku) LE nu) )  ;

SET SPI(ky,ku)          actual dim of controller ;
    SPI(ky,ku)      =   YES $ ( (ORD(ky) LE ny) $ (ORD(ku) LE nu) ) ;

SET SYBNDSL(ky,i,j)     actual dim of bounds on y for points le nlim ;
    SYBNDSL(ky,i,j) =   YES $ ( (ORD(ky) LE ny) $ (ORD(i) LE nfe)
                            $ (ORD(j) LE ncof) $ (ORD(i) LE nlim) ) ;
```

```
SET SYBNDSG(ky,i,j)     actual dim of bounds on y for points gt nlim ;
    SYBNDSG(ky,i,j) =   YES $ ( (ORD(ky) LE ny) $ (ORD(i) LE nfe)
                            $ (ORD(j) LE ncof) $ (ORD(i) GT nlim) ) ;

SET SUBNDS(ku,i,j)      actual dim of bounds on u ;
    SUBNDS(ku,i,j)  =   YES $ ( (ORD(ku) LE nu) $ (ORD(i) LE nfe)
                            $ (ORD(j) LE ncof) ) ;

SET SCTRLL(ky,ku,i)     actual dim of linear control eq ;
    SCTRLL(ky,ku,i) =   YES $ ( (ORD(ky) LE ny) $ (ORD(ku) LE nu)
                            $ (ORD(ku) EQ ORD(ky)) $ (ORD(i) LE nfe) ) ;

SET SCTRLIS(ky,ku,i)    actual dim of saturated control ;
    SCTRLIS(ky,ku,i)=   YES $ ( (ORD(ky) LE ny) $ (ORD(ku) LE nu)
                            $ (ORD(ku) EQ ORD(ky)) $ (ORD(i) LE nfe) ) ;

SET SMANIP(ku)          manipulated variables only ;
    SMANIP(ku)    =     YES $ ( (ORD(ku) LE ny) ) ;

SET SSLKL(ku,i)         apply lower slack variable ;

SET SSLKU(ku,i)         apply upper slack variable ;

*=============================================================================

PARAMETERS  TAU(jp)         tau at specified level
            PHIPR(j,jp)     1-st deriv of phi
            DPHI(jp)        denominator of phi
            ALPHA(i)        size of finite element
            DTMPI(kd)       initial disturbance
            DTMPSTEP(kd)    disturbance step change
            DTMP(kd,i,j)    disturbance trajectory
            KP(ky,ku)       steady state process gain
            KDST(ky,ku)     steady state disturbance gain
            KC(ky,ku)       controller gain
            TAUI(ky,ku)     controller time constant
            YL(ky)          lower output bounds
            YU(ky)          upper output bounds
            UL(ku)          lower controller bounds
            UU(ku)          upper controller bounds
            YTOL            end tolerance envelope
            BETA            arbitrary constant ;

*-----------------------------------------------------------------------------

TABLE   GENTAU(col,jp)      the roots of Langrange polynomial

    J2                      J3
C1  .5                      1.
C2  .211324865405187        .788675134594813
C3  .1127016653792585       .5

+   J4                      J5
C1  0                       0
C2  1.                      0
C3   .8872983346207415      1.

;
*-----------------------------------------------------------------------------

* Assign tau according to the specified NCOL

TAU(jp) $ ( ORD(jp) LE ncot ) =
    gentau('C1',jp) $ (ncol EQ 1) +
    gentau('C2',jp) $ (ncol EQ 2) +
    gentau('C3',jp) $ (ncol EQ 3) ;
```



```
*-----------------------------------------------------------------------------

* Calculate DPHI (needed for calculating PHIPR)

DPHI(JP) $ SDPHI(JP) = PROD(J $ ( (ORD(J) LE ncof) $ (ORD(J) NE ORD(JP)) ),
                            ( TAU(JP) - TAU(J)) ) ;

*-----------------------------------------------------------------------------

* Calculate PHIPR (1-st dervative of PHI)

PHIPR(J,JP) $ SPHIPR(J,JP) =
    SUM (JS $ ( (ORD(JS) LE ncof) $ (ORD(JS) NE ORD(JP)) ),
    PROD(JJ $ ( (ORD(JJ) LE ncof) $ (ORD(JJ) NE ORD(JP))
            $ (ORD(JJ) NE ORD(JS)) ),
              (TAU(J) - TAU(JJ)) )) / DPHI(JP) ;

*-----------------------------------------------------------------------------

* Set alpha

ALPHA(i) $ SALF(i) = 2 ;

*-----------------------------------------------------------------------------

TABLE   MA(k,kk)    State space matrix A

        k1          k2          k3          k4          k5          k6
k1      -0.34211    -0.21053
k2      0.25
k3                              -0.37692    -0.15385
k4                              0.5
k5                                                      -0.32222    -0.14815
k6                                                      0.25

+       k7          k8          k9          k10
k7      -0.29167    -0.083333
k8      0.25
k9                              -0.19697    -0.12121
k10                             0.25
;

*-----------------------------------------------------------------------------

TABLE   MB(k,ku)    State space matrix B

        k1          k2          k3
k1      0.125
k3                  0.5
k5                  0.25
;

*-----------------------------------------------------------------------------

TABLE   MC(ky,kk)       State space matrix C

        k1          K2          K3          K4          K5
K1      0.069432    0.16337                 -0.26769
K2

+       k6          K7          K8          K9          K10
K1                              -0.16
K2      0.32593                                         0.17455
;

*-----------------------------------------------------------------------------

TABLE   MD(k,kd)        State Space matrix lambda

        k1
k7      0.25
```

```
k9      0.25
;


*-------------------------------------------------------------------------------

* Set steady state process gains
* For y1

KP('K1','K1') $ SKP('K1','K1') =   0.097 ;
KP('K1','K2') $ SKP('K1','K2') = -  0.87 ;
KP('K1','K3') $ SKP('K1','K3') = - 0.092 ;

*-------------------------------------------------------------------------------

* Set steady state process gains
* For y2

KP('K2','K1') $ SKP('K2','K1') =       0 ;
KP('K2','K2') $ SKP('K2','K2') =    0.55 ;
KP('K2','K3') $ SKP('K2','K3') =    0.55 ;

*-------------------------------------------------------------------------------

* Set steady state disturbance gains

KDST('K1','K1') $ SKD('K1','K1') = - 0.48 ;
KDST('K2','K1') $ SKD('K2','K1') =   0.36 ;

*-------------------------------------------------------------------------------

* Set controller gain

KC(ky,ku) $ SPI(ky,ku) = 0 ;

KC('K1','K1') $ SPI('K1','K1') = 4.5 ;
KC('K2','K2') $ SPI('K2','K2') = 1.0 ;

*-------------------------------------------------------------------------------

* Set controller time constant
* To avoid problems with division by zero

TAUI(ky,ku) $ SPI(ky,ku) = 1 ;

* Actual controller parameters

TAUI('K1','K1') $ SPI('K1','K1') = 10 ;
TAUI('K2','K2') $ SPI('K2','K2') = 15 ;

*-------------------------------------------------------------------------------

* Set steady state disturbance

DTMPI('K1') $ SD('K1') = -1 ;

* Set disturbance step size

DTMPSTEP('K1') $ SD('K1') = 2 ;

*-------------------------------------------------------------------------------

* Calculate disturbance profile

DTMP(k,i,j) $ SDTMP(k,i,j) = DTMPSTEP(k) ;

*-------------------------------------------------------------------------------

* Set bounds on Y
* lower bounds
```



```
YL('K1') $ SYB('K1') =    0 ;
YL('K2') $ SYB('K2') = - 10 ;

* upper bounds

YU('K1') $ SYB('K1') = 1000 ;
YU('K2') $ SYB('K2') =   20 ;

*-------------------------------------------------------------------------------

* Set bounds on  U
* lower bounds
UL('K1') $ SUB('K1') = -  7 ;
UL('K2') $ SUB('K2') = -  4 ;
UL('K3') $ SUB('K3') =    0 ;

* upper bounds
UU('K1') $ SUB('K1') =    8 ;
UU('K2') $ SUB('K2') =   16 ;
UU('K3') $ SUB('K3') =   40 ;

*-------------------------------------------------------------------------------

* Set tolerance envelope

YTOL = 0.01 ;

*-------------------------------------------------------------------------------

* Set arbitrary large constant

BETA = 20 ;

*===============================================================================

VARIABLES
    XCOL(k,i,jp)    collocation coefficients
    YSET(ky)        steady state setpoint
    UAI(ku)         steady state actuator output
    Y(ky,i,j)       measured output profiles
    U(ku,i,j)       manipulated variable profile
    SLL(ku,i)       lower slack variables
    SLU(ku,i)       upper slack variables
    ZL(ku,i)        lower binary variables
    ZU(ku,i)        upper binary variables
    OBJ             objective function value
    ;

POSITIVE VARIABLES
    SLL
    SLU
    ;

BINARY VARIABLES
    ZL
    ZU
    ;


*===============================================================================

* EQUATIONS

*-------------------------------------------------------------------------------

* residual equations

EQUATION ERES(k,i,j) ;
ERES(k,i,j)  $ SRES(k,i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), XCOL(k,i,jp) * PHIPR(j,jp) ) -
```

```
    ALPHA(i) * (
    SUM(kk $ (ORD(kk) LE nk), MA(k,kk) * XCOL(kk,i,j) )     +
    SUM(ku $ (ORD(ku) LE nu), MB(k,ku) *    U(ku,i,j) )     +
    SUM(kd $ (ORD(kd) LE nd), MD(k,kd) * DTMP(kd,i,j) )
     )
    =E= 0 ;

*-------------------------------------------------------------------------------

* continuity equations

EQUATION ECONT(k,i) ;
ECONT(k,i) $ SCONT(k,i) ..
    XCOL(k,i,'J1') =E= SUM(j $ (ORD(j) LE ncof), XCOL(k,i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

*-------------------------------------------------------------------------------

* steady state

EQUATION EYSET(ky) ;
EYSET(ky) $ SYSET(ky) ..
    YSET(ky) =E= SUM(ku , KP(ky,ku) * UAI(ku) ) +
                 SUM(kd , KDST(ky,kd) * DTMPI(kd) ) ;

*-------------------------------------------------------------------------------

* measured output

EQUATION EYOUT(ky,i,j) ;
EYOUT(ky,i,j) $ SYOUT(ky,i,j) ..
    Y(ky,i,j) =E= SUM(kk, MC(ky,kk) * XCOL(kk,i,j) );
*-------------------------------------------------------------------------------

* calculate step wise input U

EQUATION EU(ku,i,j) ;
EU(ku,i,j) $ SU(ku,i,j) ..
    U(ku,i,j) =E= U(ku,i,'J1') ;

*-------------------------------------------------------------------------------

* bounds on controlled variable for points le nlim
* lower bounds

EQUATION EYLL(ky,i,j) ;
EYLL(ky,i,j) $ SYBNDSL(ky,i,j) ..
    Y(ky,i,j) + YSET(ky) =G= YL(ky) ;

* upper bounds

EQUATION EYUL(ky,i,j) ;
EYUL(ky,i,j) $ SYBNDSL(ky,i,j) ..
    Y(ky,i,j) + YSET(ky) =L= YU(ky) ;

*-------------------------------------------------------------------------------

* bounds on controlled variable for points gt nlim
* lower bounds

EQUATION EYLG(ky,i,j) ;
EYLG(ky,i,j) $ SYBNDSG(ky,i,j) ..
    Y(ky,i,j) =G= - YTOL ;

* upper bounds

EQUATION EYUG(ky,i,j) ;
```



```
EYUG(ky,i,j) $ SYBNDSG(ky,i,j) ..
    Y(ky,i,j) =L=   YTOL ;

*-------------------------------------------------------------------------------

* bounds on manipulated variables
* lower bounds

EQUATION EUL(ku,i,j) ;
EUL(ku,i,j) $ SUBNDS(ku,i,j) ..
    U(ku,i,j) + UAI(ku) =G= UL(ku) ;

* upper bounds

EQUATION EUU(ku,i,j) ;
EUU(ku,i,j) $ SUBNDS(ku,i,j) ..
    U(ku,i,j) + UAI(ku) =L= UU(ku) ;

*-------------------------------------------------------------------------------

* linear controller equation

EQUATION ECTRLL(ky,ku,i) ;
ECTRLL(ky,ku,i) $ SCTRLL(ky,ku,i) ..
    U(ku,i,'J1') =E= U(ku,i-1,'J1') + kc(ky,ku) *
        ( - Y(ky,i,'J1') + Y(ky,i-1,'J1') -
        (alpha(i)/taui(ky,ku)) * Y(ky,i,'J1') ) ;

*-------------------------------------------------------------------------------

* controller equation with input saturation

EQUATION ECTRLIS(ky,ku,i) ;
ECTRLIS(ky,ku,i) $ SCTRLIS(ky,ku,i) ..
    U(ku,i,'J1') - SLL(ku,i)$SSLKL(ku,i) + SLU(ku,i)$SSLKU(ku,i) =E=
        U(ku,i-1,'J1') + kc(ky,ku) *
        ( - Y(ky,i,'J1') + Y(ky,i-1,'J1') -
        (alpha(i)/taui(ky,ku)) * Y(ky,i,'J1') ) ;

*-------------------------------------------------------------------------------

* slack variable bounds
* lower slack
* lower bounds

EQUATION ESLLL(ku,i) ;
ESLLL(ku,i) $ SSLKL(ku,i) ..
    SLL(ku,i) =G= 0 ;

* upper bounds

EQUATION ESLLU(ku,i) ;
ESLLU(ku,i) $ SSLKL(ku,i) ..
    SLL(ku,i) =L= beta * ZL(ku,i) ;

*-------------------------------------------------------------------------------

* slack variable bounds
* upper slack
* lower bounds

EQUATION ESLUL(ku,i) ;
ESLUL(ku,i) $ SSLKU(ku,i) ..
    SLU(ku,i) =G= 0 ;

* upper bounds

EQUATION ESLUU(ku,i) ;
ESLUU(ku,i) $ SSLKU(ku,i) ..
```

```
    SLU(ku,i) =L= beta * ZU(ku,i) ;

*---------------------------------------------------------------------------

** actuator bounds
** lower slack
** lower bound

EQUATION EZLL(ku,i) ;
EZLL(ku,i) $ SSLKL(ku,i) ..
    -beta * (1 - ZL(ku,i)) =L= UL(ku) - U(ku,i,'J1') - UAI(ku) ;

* upper bound

EQUATION EZLU(ku,i) ;
EZLU(ku,i) $ SSLKL(ku,i) ..
    UL(ku) - U(ku,i,'J1') - UAI(ku) =L=  beta * (1 - ZL(ku,i)) ;

*---------------------------------------------------------------------------

* actuator bounds
* upper slack
* lower bound

EQUATION EZUL(ku,i) ;
EZUL(ku,i) $ SSLKU(ku,i) ..
    -beta * (1 - ZU(ku,i)) =L= UU(ku) - U(ku,i,'J1') - UAI(ku) ;

* upper bound

EQUATION EZUU(ku,i) ;
EZUU(ku,i) $ SSLKU(ku,i) ..
    UU(ku) - U(ku,i,'J1') - UAI(ku) =L=  beta * (1 - ZU(ku,i)) ;

*---------------------------------------------------------------------------

* bilinear constraints
* lower slack

EQUATION EBSLL(ku,i) ;
EBSLL(ku,i) $ SSLKL(ku,i) ..
    SLL(ku,i) * ( U(ku,i,'J1') + UAI(ku) - UL(ku) ) =E= 0 ;

* upper slack

EQUATION EBSLU(ku,i) ;
EBSLU(ku,i) $ SSLKU(ku,i) ..
    SLU(ku,i) * ( U(ku,i,'J1') + UAI(ku) - UU(ku) ) =E= 0 ;

*---------------------------------------------------------------------------

* lower

EQUATION EZZL(ku,i) ;
EZZL(ku,i) $ (SMANIP(ku) AND SNFE(i)) ..
    ZL(ku,i)$SSLKL(ku,i) + ZU(ku,i)$SSLKU(ku,i) =G= 0 ;

*---------------------------------------------------------------------------

* objective function

EQUATION EZZU(ku,i) ;
EZZU(ku,i) $ (SMANIP(ku) AND SNFE(i)) ..
    ZL(ku,i)$SSLKL(ku,i) + ZU(ku,i)$SSLKU(ku,i) =L= 1 ;

*---------------------------------------------------------------------------

* objective function

EQUATION EOBJ ;
EOBJ ..
```



```
    OBJ =E= UAI('K3') ;

*---------------------------------------------------------------------------

* Initial conditions

    Y.FX('K1','I1','J1') = 0;
    Y.FX('K2','I1','J1') = 0;
    U.FX('K1','I1','J1') = 0;
    U.FX('K2','I1','J1') = 0;

    XCOL.FX(k,'I1','J1') $ SXCOL(k,'I1','J1') = 0;

*===========================================================================

MODEL FCCULP / ERES, ECONT, EYSET, EYOUT, EU, EYLL, EYUL, EYLG, EYUG,
               EUL, EUU, ECTRLL, EOBJ / ;

MODEL FCCUMIP1 / ERES, ECONT, EYSET, EYOUT, EU, EYLL, EYUL, EYLG, EYUG,
                 EUL, EUU,
                 ECTRLIS,
                 ESLLL, ESLLU,
                 ESLUL, ESLUU,
                 EZLL, EZLU,
                 EZUL, EZUU,
                 EOBJ / ;

MODEL FCCUMIP2 / ERES, ECONT, EYSET, EYOUT, EU, EYLL, EYUL, EYLG, EYUG,
                 EUL, EUU,
                 ECTRLIS,
                 ESLLL, ESLLU,
                 ESLUL, ESLUU,
                 EZLL, EZLU,
                 EZUL, EZUU,
                 EZZL, EZZU,
                 EOBJ / ;

MODEL FCCUBLP / ERES, ECONT, EYSET, EYOUT, EU, EYLL, EYUL, EYLG, EYUG,
                EUL, EUU,
                ECTRLIS,
                ESLLL,
                ESLUL,
                EBSLL, EBSLU
                EOBJ / ;

*===========================================================================
```

GAMS - FCCU EXAMPLE LP CODE

```
$TITLE      Grosdidier's FCCU example (no input saturation)

*=============================================================================

    OPTION LIMCOL   =   0 ;
    OPTION LIMROW   =   0 ;
    OPTION SOLPRINT =  ON ;
*    OPTION SYSOUT   =     ON;
*    OPTION RESLIM   =  43200 ;
*    OPTION OPTCR    = 0.0001;
*-----------------------------------------------------------------------------

    FCCULP.scaleopt = 0;
    SOLVE FCCULP USING LP MAXIMISING OBJ ;
*-----------------------------------------------------------------------------

* Output values for LP

    DISPLAY YSET.L, UAI.L ;
    YOUT(k,i) = Y.L(k,i,'J1') + YSET.L(k) ;
    UOUT(k,i) = U.L(k,i,'J1') + UAI.L(k) ;
*-----------------------------------------------------------------------------

* Output to file "colfcculp.xls"

fcculpout.pc    =     6 ;
fcculpout.nd    =    10 ;
fcculpout.nr    =     0 ;
fcculpout.nz    = 1e-10 ;

put fcculpout;
loop(i $ (ORD(i) LE nfe),
        put ORD(i);
        loop(ky $ (ORD (ky) LE ny), put YOUT(ky,i));
        loop(ku $ (ORD (ku) LE nu), put UOUT(ku,i));
    put /;
        );
putclose;

*=============================================================================
```


GAMS - FCCU EXAMPLE FULL MIP CODE

```
$TITLE      Grosdidier's FCCU example (with MIP input saturation)

*=============================================================================

    SSLKL(ku,i)$ ( SMANIP(ku) AND SNFE(i) ) = YES ;
    SSLKU(ku,i)$ ( SMANIP(ku) AND SNFE(i) ) = YES ;

    SSLKL('k1',i)$ ( SMANIP('k1') AND SNFE(i) ) = NO ;
    SSLKU('k2',i)$ ( SMANIP('k2') AND SNFE(i) ) = NO ;

*=============================================================================

put opt ;
put 'subalg 1' /
    'mipinterval 10' /
    'trelim 32' /
    'nodefileind 2' /
    'nodefilelim 500' /
    'nodesel 2' /
    'varsel 2' ;
putclose ;

*=============================================================================

    OPTION LIMCOL   =   0 ;
    OPTION LIMROW   =   0 ;
    OPTION SOLPRINT =  ON ;
    OPTION SYSOUT   =  ON ;
    OPTION ITERLIM  =  10000000;
    OPTION RESLIM   =  86400000 ;
    OPTION OPTCR    = 0;
    OPTION BRATIO   = 0;
*-----------------------------------------------------------------------------

    FCCUMIP1.scaleopt = 1 ;
    FCCUMIP1.workspace = 50 ;
    FCCUMIP1.optfile = 1 ;
    SOLVE FCCUMIP1 USING MIP MAXIMISING OBJ ;
*-----------------------------------------------------------------------------

* Output values for MIP

    DISPLAY YSET.L, UAI.L ;
    YOUT(ky,i) = Y.L(ky,i,'J1') + YSET.L(ky) ;
    UOUT(ku,i) = U.L(ku,i,'J1') + UAI.L(ku) ;
*-----------------------------------------------------------------------------

* Output to file "colfccumip.xls"

fccumip1out.pc   =     6 ;
fccumip1out.nd   =    10 ;
fccumip1out.nr   =     0 ;
fccumip1out.nz   = 1e-10 ;

put fccumip1out;
loop(i $ (ORD(i) LE nfe),
        put ORD(i) ;
        loop(ky $ (ORD (ky) LE ny),  put YOUT(ky,i));
        loop(ku $ (ORD (ku) LE nu),  put UOUT(ku,i));
    put /;
        );
putclose;

*=============================================================================
```

```
GAMS - FCCU EXAMPLE MIP WITH ALGORITHM CODE

$TITLE      Grosdidier's FCCU example (with selective MIP input saturation)

*=============================================================================

SCALAR newslk /1/ ;
SSLKL(ku,i) $ ( SMANIP(ku) AND SNFE(i) ) = NO ;
SSLKU(ku,i) $ ( SMANIP(ku) AND SNFE(i) ) = NO ;

*-----------------------------------------------------------------------------

* Global output file options

    fccuwhileout.pc    =     6 ;
    fccuwhileout.nd    =    10 ;
    fccuwhileout.nr    =     0 ;
    fccuwhileout.nz    = 1e-10 ;
    fccuwhileout.ap    =     0 ;
*-----------------------------------------------------------------------------

* Global solver options

    OPTION LIMCOL   =   0 ;
    OPTION LIMROW   =   0 ;
    OPTION SOLPRINT = OFF ;
*    OPTION SYSOUT   =     ON;
*    OPTION RESLIM   =  43200 ;
    OPTION OPTCR    = 0;
*-----------------------------------------------------------------------------

* Global problem specific options

    FCCUMIP1.scaleopt= 0 ;
*-----------------------------------------------------------------------------

* Loop while improvement in MIP

while ( (newslk ne 0),

    newslk = 0 ;

    loop(ku $ SMANIP(ku),
      loop(i $ (ORD(i) LE nFE),
          if (   (UOUT(ku,i) EQ UL(ku)) AND (NOT(SSLKL(ku,i))),
              SSLKL(ku,i) = YES ;
              newslk = newslk + 1 ;
          elseif (UOUT(ku,i) EQ UU(ku)) AND (NOT(SSLKU(ku,i))),
              SSLKU(ku,i) = YES ;
              newslk = newslk + 1 ;
          ) ;
      ) ;
    ) ;
*-----------------------------------------------------------------------------

    SOLVE FCCUMIP1 USING MIP MAXIMISING OBJ ;
*-----------------------------------------------------------------------------

* Output values for MIP

    DISPLAY YSET.L, UAI.L ;
    DISPLAY zL.L, zU.L ;
    YOUT(ky,i) = Y.L(ky,i,'J1') + YSET.L(ky) ;
    UOUT(ku,i) = U.L(ku,i,'J1') + UAI.L(ku) ;
*-----------------------------------------------------------------------------

* Output to file "colwhile.xls"

put fccuwhileout;
    loop(i $ (ORD(i) LE nfe),
```



```
        put ORD(i)
        loop(ky $ (ORD (ky) LE ny),  put YOUT(ky,i));
        loop(ku $ (ORD (ku) LE nu),  put UOUT(ku,i));
        put /;
    );
    put // ;
putclose;
*-----------------------------------------------------------------------------

* End of while loop
    fccuwhileout.ap    =     1 ;
) ;

*-----------------------------------------------------------------------------

*  Final iteration

    OPTION SOLPRINT = ON ;
    SOLVE FCCUMIP1 USING MIP MAXIMISING OBJ ;

    display YOUT, UOUT ;
    display SSLKL, SSLKU ;

*=============================================================================
```

GAMS - CASE 1 MODEL CODE

```
$TITLE      Schweiger and Floudas CSTR example
$OFFUPPER
$OFFSYMXREF OFFSYMLIST
$OFFDIGIT

*=============================================================================

SCALARS
    NK      actual # of state equations         /   3 /
    NY      actual # of ouptut var.             /   2 /
    NU      actual # of manip. var.             /   1 /
    ND      actual # of dist. var.              /   1 /
    NCOL    actual # coll. pts used             /   2 / ;

SCALARS
    NTOL    # points within tolerance           /  10 /
    TTOT    actual final time                   /  10 / ;

SCALARS
    TDSTP1  time of first disturbance step      / 0.5 /
    TDSTP2  time of second disturbance step     / 100 /
    TLAG    input deadtime                      /   3 /;

SCALAR
    TSTEP   desired controller timestep         / 0.1 /
    NFE     # of finite elements per timestep   /   1 / ;

*-----------------------------------------------------------------------------

* Physical constants

SCALARS
    dH      heat of rxn [btu.lbmol^-1]              / -30000  /
    Ua      heat trans coeff [btu.[hr.ft^2.R]^-1]   / 300     /
    EaR     activation energy [R]                   / 15075   /
    k0      rxn rate const [hr^-1]                  / 4.08e10 /
    rho     liquid density [lb.ft^-3]               / 50      /
    Cp      liquid heat capacity [btu.[lb.R]^-1]    / 0.75    /
    rhoj    coolant density [lb.ft^-3]              / 62.3    /
    Cpj     coolant heat capacity [btu.[lb.R]]      / 1.0     /
    Fr1     feed flow rate [ft^3.hr^-1]             / 100     /
    Cr1i    feed composition [lbmol A.ft^-3]        / 1       /
    Tr1in   feed temperature [R]                    / 600     /
    Tj1i    coolant inlet temp [R]                  / 530     /
    Ccost   coolant cost [$.ft^-3]                  / 3.74e-3 / ;

*-----------------------------------------------------------------------------

* Upper and lower variable bounds

SCALARS
    Tr1LO   /  600 /
    Tr1UP   /  650 /
    Tj1LO   /  600 /
    Tj1UP   /  650 /
    Fj1LO   /  450 /
    Fj1UP   /  500 /

    Cr1LO   / 1e-8 /
    Cr1UP   /    1 /
    Dr1LO   /    1 /
    Dr1UP   /  100 /
    Hr1LO   /    1 /
    Hr1UP   /  100 /
    tauLO   /  0.1 /
    tauUP   / 100 / ;
```



```
*-----------------------------------------------------------------------------

* Tolerance bounds

SCALARS
    eta     ise tolerance           / 100  /
    ytol    output ss trajectory    / 1e-6 /
    utol    input ss trajectory     / 1e-4 /
    etol    ss trajectory           / 1e-6 / ;
*-----------------------------------------------------------------------------

* Point constraints

SCALARS
    Cr1TF   / 0.04 / ;

*-----------------------------------------------------------------------------

* Problem parameters

SCALARS
    dstep   disturbance step size   / 45   /
    tauv    process input lag       / 0.1  / ;
*-----------------------------------------------------------------------------

SCALARS
    beta    / 1000 /
    pi      / 3.1415927 / ;

*-----------------------------------------------------------------------------

    OPTION LIMCOL   =        0 ;
    OPTION LIMROW   =        0 ;
    OPTION SOLPRINT =       ON ;
    OPTION SYSOUT   =      OFF ;
    OPTION ITERLIM  = 50000000 ;
    OPTION RESLIM   =    28800 ;
*-----------------------------------------------------------------------------

    OPTION NLP      =   CONOPT2 ;
    OPTION RMINLP   =   CONOPT2 ;
    OPTION MINLP    =   DICOPT  ;

*=============================================================================

SETS
    K       equation # (max 10)             / K1*K10  /
    I       finite elements # (max 300)     / I1*I300 /
    J       collocation coeff. #            / J1*J5   /
    COL     # possible coll pt (max 3)      / C1*C3   /

ALIAS
    (K,KK,KS,KY,KU,KD)
    (J,JP,JJ,JS)
    (I, II);

*=============================================================================

SCALARS NFETOT      actual # of FE used ;
    NFETOT = (ttot / tstep) * nfe ;

*=============================================================================

ABORT $ (nk GT 10)      "Error in defining NK   (max 10)", nk   ;
ABORT $ (ny GT 10)      "Error in defining NY   (max 10)", ny   ;
ABORT $ (nu GT 10)      "Error in defining NU   (max 10)", nu   ;
ABORT $ (nd GT 10)      "Error in defining ND   (max 10)", nd   ;
ABORT $ (nfetot GT 300) "Error in defining FE   (max 300)", nfetot ;
ABORT $ (ncol GT 3)     "Error in defining NCOL (max  3)", ncol ;
```

```
*=======================================================================

SCALAR NCOF    equal to ncol+1 ;
    NCOF = ncol + 1 ;

SCALAR NCOT    equal to ncol+2 ;
    NCOT = ncol + 2;

*=======================================================================

SET SXCOL(i,jp)          actual dim of coll. coeff. (XCOL)   ;
    SXCOL(i,jp)      =   YES $ ( (ORD(i) LE nfetot)
                             $ (ORD(jp) LE ncof) ) ;

SET SPHIPR(j,jp)         actual dim of PHIPR ;
    SPHIPR(j,jp)     =   YES $ ( (ORD(j) GT 1) $ (ORD(j) LE ncot)
                             $ (ORD(jp) LE ncof) ) ;

SET SDPHI(jp)            actual dim of denominator of PHI ;
    SDPHI(jp)        =   YES $ ( (ORD(jp) LE ncof) ) ;

SET SNFE(i)              actual dim of nfe ;
    SNFE(i)          =   YES $ (ORD(i) LE nfetot) ;

SET SRES(i,j)            actual dim of residual eq ;
    SRES(i,j)        =   YES $ ( (ORD(i) LE nfetot) $ (ORD(j) GT 1)
                             $ (ORD(j) LE ncof) )  ;

SET SCONT(i)             actual dim of continuity eq ;
    SCONT(i)         =   YES $ ( (ORD(i) GT 1) $ (ORD(i) LE nfetot) );

SET SCTRL(i)             actual dim of control eq ;
    SCTRL(i)         =   YES $ (ORD(i) LE nfetot) ;

SET STCTRL(i)            actual dim of control eq ;
    STCTRL(i)        =   YES $ ((ORD(i) LE nfetot) $ (mod(ORD(i),nfe) EQ 0)) ;

SET SINI(i)              starting point ;
    SINI(i)          =   YES $ ((ORD(i) EQ 1)) ;

SET SEND(i,j)            endpoint ;
    SEND(i,j)        =   YES $ ((ORD(i) EQ nfetot) $ (ORD(j) LE ncof)) ;

SET SSLKL(i)             lower slack variable points ;
    SSLKL(i)         =   NO ;

SET SSLKU(i)             upper slack variable points ;
    SSLKU(i)         =   NO ;

*=======================================================================

PARAMETERS
    TAU(jp)          tau at specified level
    PHIPR(j,jp)      1-st deriv of phi
    DPHI(jp)         denominator of phi
    NEWSLK           current # slack added
    ALPHA            size of finite element
    TIME(i)          time at i ;

*-----------------------------------------------------------------------

TABLE   GENTAU(col,jp)       the roots of Langrange polynomial

    J2                       J3
C1  .5                       1.
C2  .211324865405187         .788675134594813
C3  .1127016653792585        .5

+   J4                       J5
```



```
C1  0                        0
C2  1.                       0
C3   .8872983346207415       1. ;

*-----------------------------------------------------------------------

* Assign tau according to the specified NCOL

TAU(jp) $ ( ORD(jp) LE ncot ) =
    gentau('C1',jp) $ (ncol EQ 1) +
    gentau('C2',jp) $ (ncol EQ 2) +
    gentau('C3',jp) $ (ncol EQ 3) ;

*-----------------------------------------------------------------------

* Calculate DPHI (needed for calculating PHIPR)

DPHI(JP) $ SDPHI(JP) =
    PROD(J $ ( (ORD(J) LE ncof) $ (ORD(J) NE ORD(JP)) ),
            ( TAU(JP) - TAU(J)) ) ;

*-----------------------------------------------------------------------

* Calculate PHIPR (1-st dervative of PHI)

PHIPR(J,JP) $ SPHIPR(J,JP) =
    SUM (JS $ ( (ORD(JS) LE ncof) $ (ORD(JS) NE ORD(JP)) ),
    PROD(JJ $ ( (ORD(JJ) LE ncof) $ (ORD(JJ) NE ORD(JP))
            $ (ORD(JJ) NE ORD(JS)) ),
              (TAU(J) - TAU(JJ)) )) / DPHI(JP) ;

*-----------------------------------------------------------------------

* Set alpha

ALPHA = TTOT/NFETOT ;

TIME(i) $ SNFE(i) = ORD(i) * alpha ;

*-----------------------------------------------------------------------

SCALAR TTOL    time to apply tolerance bounds ;
    TTOL = ntol * alpha;

SCALAR NISSTRT  time to start applying input saturation ;
    NISSTRT = ceil(tdstp1/alpha);

SCALAR NISSTP   time to stop applying input saturation ;
    NISSTP = NFETOT ;
*-----------------------------------------------------------------------

* Set tolerance sets

SET STOL(i)              tolerance envelope ;
    STOL(i)      =   YES $((
                     ( (time(i) GT (tdstp2 - ttol) ) AND (time(i) LE tdstp2) )
                        OR
                     ( (ORD(i) GT (nfetot - ntol) ) AND (ORD(i) LE nfetot) ) )
                        $ (ORD(i) LE nfetot) );

SET SXEND(i,j)           endpoint ;
    SXEND(i,j)   =   YES $( (ORD(i) EQ nfetot)
                        $  (ORD(j) EQ 1) );

*-----------------------------------------------------------------------

* Feed temperature disturbance

PARAMETER Tr1i(i) ;
Tr1i(i) $ SNFE(i) =
```

```
    Tr1in + dstep$(time(i) GE tdstp1) - dstep$(time(i) GE tdstp2) ;

*---------------------------------------------------------------------------

* Output variables

PARAMETER
    Tr1OUT(i)
    Fj1OUT(i)
    Fj1cOUT(i)
    Cr1OUT(i)
    Tj1OUT(i)
    Tr1iOUT(i)
    ERROUT(i) ;

FILE
    r1nlpout       /r1nlp.xls/
    r1rminlpout    /r1rminlp.xls/
    r1minlpout     /r1minlp.xls/ ;

*===========================================================================

VARIABLES

* Time varying
    Cr1(i,j)          output concentration
    Tr1(i,j)          output temperature from r1 (controlled)
    Tj1(i,j)          jacket coolant temperature
    Fj1(i,j)          input jacket coolant flowrate (lagged)
    Fj1c(i,j)         controller output (manipulated)
    Cprd(i,j)         product concentration
    Tprd(i,j)         product temperature

    Err(i)            setpoint tracking error
    dFj1(i)           change in the process input
    dFj1c(i)          change in controller output

    sL(i)             lower slack
    sU(i)             upper slack
    zL(i)             lower binary
    zU(i)             upper binary

* Cost variables
    OBJ               objective function value
    ISE               integral square error
    CCap              capital cost
    CUtil             utility cost
    CTot              total cost

* Reactor 1
    Dr1               diameter of r1
    Hr1               height of r1
    Vr1               volume of r1
    Ar1               surface area of r1
    Vj1               volume of jacket on

    Cr1n              nominal concentration from
    Tr1n              nominal output temperature from

* Reactor 1 jacket
    Tj1n              nominal temperature of coolant out of r1
    Fj1n              nominal flowrate of coolant around jacket
    Fj1cn             nominal flowrate of coolant around jacket


* Controller parameters
    Kc                controller gain
    taui              controller time constant
```



```
    CNTRLs1(i)        intermediate variable for control equation
    CNTRLs2(i)        intermediate variable for control equation
    CNTRLs3(i)        intermediate variable for control equation

    EoTn              activation energy over temperature
    expEoTn           exponent of activation energy over temperature
    CR1ssS1           intermediate variable for ss concentration in reactor
    CR1ssS2           intermediate variable for ss concentration in reactor
    TR1ssS1           intermediate variable for ss temperature in reactor
    TR1ssS2           intermediate variable for ss temperature in reactor
    TR1ssS3           intermediate variable for ss temperature in reactor
    TJ1ssS1           intermediate variable for ss temperature in jacket
    TJ1ssS2           intermediate variable for ss temperature in jacket
    TJ1ssS3           intermediate variable for ss temperature in jacket
    ;

*---------------------------------------------------------------------------

POSITIVE VARIABLES
* Time varying
    Cr1(i,j)
    Tr1(i,j)
    Fj1(i,j)
    Fj1c(i,j)
    Tj1(i,j)
    Cprd(i,j)
    Tprd(i,j)
    sL(i)
    sU(i)

* Reactor 1
    Dr1
    Hr1
    Vr1
    Ar1

    Cr1n
    Tr1n
    Cr1end
    Tr1end

* Reactor 1 jacket
    Vj1

    Tj1n
    Fj1n
    Fj1cn

    Tj1end
    Fj1end
    Fj1cend

* Controller parameters
    taui
    ;

*---------------------------------------------------------------------------

NEGATIVE VARIABLES
    Kc
    ;

*---------------------------------------------------------------------------

BINARY VARIABLES
    zL(i)
    zU(i)
    ;

*===========================================================================
```

```
* EQUATIONS

*-------------------------------------------------------------------------------

* residual equations

EQUATION EEoT(i,j) ;
EEoT(i,j) $ SXCOL(i,j) ..
    EoT(i,j) =E= -EaR/Tr1(i,j) ;

EQUATION EexpEoT(i,j) ;
EexpEoT(i,j) $ SXCOL(i,j) ..
    expEoT(i,j) =E= exp(EoT(i,j)) ;

*-------------------------------------------------------------------------------

* CSTR mole balance

EQUATION ERES1(i,j) ;
ERES1(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), Cr1(i,jp) * PHIPR(j,jp) ) -
    ALPHA * (
    1/Vr1 * ( Cr1i*Fr1 - Cr1(i,j)*Fr1 -
    ( k0*exp(-EaR/Tr1(i,j)) ) * Vr1 * Cr1(i,j) )
*    RES1s3(i,j)
    )
    =E= 0 ;

*-------------------------------------------------------------------------------

* CSTR energy balance

EQUATION ERES2(i,j) ;
ERES2(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), Tr1(i,jp) * PHIPR(j,jp) ) -
    ALPHA * (
    1/Vr1 * ( Tr1i(i)*Fr1 - Tr1(i,j)*Fr1 - dH/(rho*Cp) *
    ( k0 * exp(-EaR/Tr1(i,j)) ) * Vr1 * Cr1(i,j) -
    Ua/(rho*Cp) * Ar1 * ( Tr1(i,j) - Tj1(i,j) ) )
*    RES2s4(i,j)
    )
    =E= 0 ;

*-------------------------------------------------------------------------------

* Jacket energy balance

EQUATION ERES3(i,j) ;
ERES3(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), Tj1(i,jp) * PHIPR(j,jp) ) -
    ALPHA * (
    1/Vj1 * ( Tj1i*Fj1(i,j) - Tj1(i,j)*Fj1(i,j) +
    Ua/(rhoj*Cpj) * Ar1 * ( Tr1(i,j) - Tj1(i,j) ) )
*    RES3s4(i,j)
    )
    =E= 0 ;

*-------------------------------------------------------------------------------

** Lag on process input
*EQUATION ERES4(i,j) ;
*ERES4(i,j)  $ SRES(i,j) ..
*    SUM(jp $ (ORD(jp) LE ncof), Fj1(i,jp) * PHIPR(j,jp) ) -
*    ALPHA * (
*    1/tauv * ( Fj1c(i,j) - Fj1(i,j) )
*    )
```



```
*    =E= 0 ;

*-------------------------------------------------------------------------------

* Delay on process input
EQUATION ELAG(i,j) ;
ELAG(i,j)  $ SXCOL(i,j) ..
    Fj1(i,j) =E= Fj1cn$(ORD(i) LE TLAG) + Fj1c(i-TLAG,j)$(ORD(i) GT TLAG) ;

*-------------------------------------------------------------------------------

* continuity equations

EQUATION ECONT1(i) ;
ECONT1(i) $ SCONT(i) ..
    Cr1(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), Cr1(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT2(i) ;
ECONT2(i) $ SCONT(i) ..
    Tr1(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), Tr1(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT3(i) ;
ECONT3(i) $ SCONT(i) ..
    Tj1(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), Tj1(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

*EQUATION ECONT4(i) ;
*ECONT4(i) $ SCONT(i) ..
*    Fj1(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), Fj1(i-1,j) *
*            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
*                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

*-------------------------------------------------------------------------------

* Initial Conditions

EQUATION ECR1ini ;
ECR1ini ..
    Cr1('I1','J1') =E= Cr1n ;

EQUATION ETR1ini ;
ETR1ini ..
    Tr1('I1','J1') =E= Tr1n ;

EQUATION ETJ1ini ;
ETJ1ini ..
    Tj1('I1','J1') =E= Tj1n ;

EQUATION EFJ1ini ;
EFJ1ini ..
    Fj1('I1','J1') =E= Fj1n ;

EQUATION EFJ1Cini ;
EFJ1Cini ..
    Fj1c('I1','J1') =E= Fj1cn ;

*-------------------------------------------------------------------------------

* starting steady state balances

EQUATION EEoTn ;
EEoTn ..
    EoTn =E= -EaR/Tr1n ;

EQUATION EexpEoTn ;
```

```
EexpEoTn ..
    expEoTn =E= exp(EoTn) ;

*-------------------------------------------------------------------------------

* CSTR mole ss balance

EQUATION ECR1ssS1 ;
ECR1ssS1 ..
    CR1ssS1 =E= k0 * expEoTn * Vr1 * Cr1n ;

EQUATION ECR1ssS2 ;
ECR1ssS2 ..
    CR1ssS2 =E= Cr1i*Fr1 - Cr1n*Fr1 - CR1ssS1 ;

EQUATION ECR1ss ;
ECR1ss ..
*    Cr1i*Fr1 - Cr1n*Fr1 -
*    ( k0 * exp(-EaR/Tr1n) ) * Vr1*Cr1n
*    CR1ssS2
    Cr1i*Fr1 - Cr1n*Fr1 - CR1ssS1
    =E= 0 ;

*-------------------------------------------------------------------------------

* CSTR ss energy balance

EQUATION ETR1ssS1 ;
ETR1ssS1 ..
    TR1ssS1 =E= dH/(rho*Cp) * k0 * expEoTn * Vr1 * Cr1n ;

EQUATION ETR1ssS2 ;
ETR1ssS2 ..
    TR1ssS2 =E= Ua/(rho*Cp) * Ar1 * ( Tr1n - Tj1n ) ;

EQUATION ETR1ssS3 ;
ETR1ssS3 ..
    TR1ssS3 =E= Tr1in*Fr1 - Tr1n*Fr1 - TR1ssS1 - TR1ssS2 ;

EQUATION ETR1ss ;
ETR1ss ..
*    Tr1in*Fr1 - Tr1n*Fr1 - dH/(rho*Cp) *
*    ( k0 * exp(-EaR/Tr1n) ) * Vr1*Cr1n -
*    Ua/(rho*Cp) * Ar1 * ( Tr1n - Tj1n )
*    TR1ssS3
    Tr1in*Fr1 - Tr1n*Fr1 - TR1ssS1 - TR1ssS2
    =E= 0 ;

*-------------------------------------------------------------------------------

* Jacket ss energy balance

EQUATION ETJ1ssS1 ;
ETJ1ssS1 ..
    TJ1ssS1 =E= Ua/(rhoj*Cpj) * Ar1 * (Tr1n -Tj1n) ;

EQUATION ETJ1ssS2 ;
ETJ1ssS2 ..
    TJ1ssS2 =E= Tj1n*Fj1n ;

EQUATION ETJ1ssS3 ;
ETJ1ssS3 ..
    TJ1ssS3 =E= Tj1i*Fj1n - TJ1ssS2 + TJ1ssS1 ;

EQUATION ETJ1ss ;
ETJ1ss ..
*    Tj1i*Fj1n - Tj1n*Fj1n +
*    Ua/(rhoj*Cpj) * Ar1 * ( Tr1n - Tj1n )
*    TJ1ssS3
    Tj1i*Fj1n - TJ1ssS2 + TJ1ssS1
```



```
    =E= 0 ;

*-------------------------------------------------------------------------------

* ss Input lag

EQUATION EFJ1ss ;
EFJ1ss ..
    Fj1cn - Fj1n
    =E= 0 ;

*-------------------------------------------------------------------------------

* Design volume and surface area of reactors

EQUATION ER1VOL ;
ER1VOL ..
    Vr1 =E= pi/4 * SQR(Dr1) * Hr1;

EQUATION ER1AREA ;
ER1AREA ..
    Ar1 =E= pi * (Dr1) * Hr1;

*-------------------------------------------------------------------------------

* Design volume of jacket

EQUATION EJ1VOL ;
EJ1VOL ..
    Vj1 =E= 1/3 * Ar1 ;

*-------------------------------------------------------------------------------

* Error equations

EQUATION ER1ERR(i) ;
ER1ERR(i) $ SNFE(i) ..
    ERR(i) =E= Tr1n - Tr1(i,'J1');

*EQUATION ER1DFJ1(i);
*ER1DFJ1(i) $ STOL(i) ..
*    dFJ1(i) =E= Fj1(i + (NFETOT - ORD(i)),'J1') - Fj1(i,'J1');

EQUATION ER1DFJ1(i);
ER1DFJ1(i) $ STOL(i) ..
    dFJ1c(i) =E= Fj1c(i + (NFETOT - ORD(i)),'J1') - Fj1c(i,'J1');

*-------------------------------------------------------------------------------

* PI control equations

EQUATION ECNTRLs1(i) ;
ECNTRLs1(i) $ STCTRL(i) ..
    CNTRLs1(i) =E= tstep/taui * ERR(i) ;

EQUATION ECNTRLs2(i) ;
ECNTRLs2(i) $ STCTRL(i) ..
    CNTRLs2(i) =E= ERR(i) - ERR(i-1) + CNTRLs1(i);

EQUATION ECNTRLs3(i) ;
ECNTRLs3(i) $ STCTRL(i) ..
    CNTRLs3(i) =E= Kc*( CNTRLs2(i) ) ;

EQUATION ER1CNTRL(i) ;
ER1CNTRL(i) $ SCTRL(i) ..
    Fj1c(i,'J1') =E= Fj1cn$(SINI(i)) + Fj1c(i-1,'J1') +
          CNTRLs3(i)$(STCTRL(i)) ;

EQUATION ER1CNTRLU(i,j) ;
ER1CNTRLU(i,j) $ SXCOL(i,j) ..
    Fj1c(i,'J1') =E= Fj1c(i,j) ;
```

```
*-------------------------------------------------------------------------------

* Input Saturation

EQUATION ECTRLIS(i) ;
ECTRLIS(i) $ SCTRL(i) ..
    Fj1c(i,'J1') - sL(i)$(SSLKL(i)) + sU(i)$(SSLKU(i)) =E=
            Fj1cn$(SINI(i)) + Fj1c(i-1,'J1') +
            CNTRLs3(i)$(STCTRL(i)) ;

*-------------------------------------------------------------------------------

* slack variable bounds
* lower slack

* lower bounds
EQUATION ESLLL(i) ;
ESLLL(i) $ SSLKL(i) ..
    sL(i) =G= 0 ;

* upper bounds
EQUATION ESLLU(i) ;
ESLLU(i) $ SSLKL(i) ..
    sL(i) =L= beta * zL(i) ;

*-------------------------------------------------------------------------------

* slack variable bounds
* upper slack

* lower bounds
EQUATION ESLUL(i) ;
ESLUL(i) $ SSLKU(i) ..
    sU(i) =G= 0 ;

* upper bounds
EQUATION ESLUU(i) ;
ESLUU(i) $ SSLKU(i) ..
    sU(i) =L= beta * zU(i) ;

*-------------------------------------------------------------------------------

* actuator bounds
* lower slack

EQUATION EZLL(i) ;
EZLL(i) $ SSLKL(i) ..
    Fj1LO - Fj1c(i,'J1') =L=  beta * (1 - zL(i)) ;

EQUATION EZLU(i) ;
EZLU(i) $ SSLKL(i) ..
    Fj1LO - Fj1c(i,'J1') =G= -beta * (1 - zL(i)) ;

*-------------------------------------------------------------------------------

* actuator bounds
* upper slack

EQUATION EZUL(i) ;
EZUL(i) $ SSLKU(i) ..
    Fj1UP - Fj1c(i,'J1') =L=  beta * (1 - zU(i)) ;

EQUATION EZUU(i) ;
EZUU(i) $ SSLKU(i) ..
    Fj1UP - Fj1c(i,'J1') =G= -beta * (1 - zU(i)) ;

*-------------------------------------------------------------------------------

* Open Loop Conditions
```



```
EQUATION EFJ1OL ;
EFJ1OL(i,j) $ SXCOL(i,j)..
    Fj1c(i,j) =E= Fj1cn ;

*-------------------------------------------------------------------------------

* Cost functions

* capital cost
EQUATION ECCAP      capital cost ;
ECCAP ..
    CCap =E= 1916.9*exp(1.066*log(Dr1))*exp(0.802*log(Hr1));

* utility cost
EQUATION ECUTIL     utility cost ;
ECUTIL ..
    CUtil =E= 32.77 * Fj1n ;

* total cost
EQUATION ECTOTAL    total cost;
ECTOTAL ..
    CTOT =E= CCAP + 4 * CUTIL ;

*-------------------------------------------------------------------------------

* ISE
EQUATION EISE ;
EISE ..
    ISE =E= SUM(i, SQR(Tr1(i,'J1') - Tr1n) );

* ISE
EQUATION EISEOBJ ;
EISEOBJ ..
    OBJ =E= ISE ;

* Multiobjective constraint
EQUATION EISEMOBJ ;
EISEMOBJ ..
    ISE =L= ETA ;

*-------------------------------------------------------------------------------

* objective function
EQUATION ECOBJ ;
ECOBJ ..
    OBJ =E= CTOT ;

*===============================================================================

MODEL CASE1NLP /
    ERES1, ERES2, ERES3,
    ECONT1, ECONT2, ECONT3,
    ELAG,
    ECr1ini, ETr1ini, ETj1ini, EFJ1ini, EFJ1Cini,
    ECR1ss, ETR1ss, ETJ1ss, EFJ1ss,
    ER1VOL, ER1AREA, EJ1VOL,
    ER1ERR, ER1DFJ1,
    ECNTRLs1, ECNTRLs2, ECNTRLs3,
    ER1CNTRL, ER1CNTRLU
*    EFJ1OL
    ECCAP, ECUTIL, ECTOTAL,
    EEoTn, EexpEoTn,
    ECR1ssS1, ETR1ssS1, ETR1ssS2, ETJ1ssS1, ETJ1ssS2
    / ;

*-------------------------------------------------------------------------------

MODEL CASE1MINLP /
    ERES1, ERES2, ERES3,
    ECONT1, ECONT2, ECONT3,
```

```
    ELAG,
    ECR1ini, ETR1ini, ETJ1ini, EFJ1ini, EFJ1Cini,
    ECR1ss, ETR1ss, ETJ1ss, EFJ1ss,
    ER1VOL, ER1AREA, EJ1VOL,
    ER1ERR, ER1DFJ1,
    ECNTRLs1, ECNTRLs2, ECNTRLs3,
    ECTRLIS, ER1CNTRLU,
*    EFJ1OL
    ECCAP, ECUTIL, ECTOTAL,
    ESLLL, ESLLU, ESLUL, ESLUU,
    EZLL, EZLU, EZUL, EZUU,
    EEoTn, EexpEoTn,
    ECR1ssS1, ETR1ssS1, ETR1ssS2, ETJ1ssS1, ETJ1ssS2,
    / ;

*=============================================================================

* Applying tolerance bounds

    ERR.LO(i) $ STOL(i) = - YTOL ;
    ERR.UP(i) $ STOL(i) =   YTOL ;

    DFJ1.LO(i) $ STOL(i)= - UTOL ;
    DFJ1.UP(i) $ STOL(i)=   UTOL ;

    DFJ1.LO(i) $ STOL(i)= - UTOL ;
    DFJ1.UP(i) $ STOL(i)=   UTOL ;

    CR1fss.LO = - ETOL ;
    CR1fss.UP =   ETOL ;
    TR1fss.LO = - ETOL ;
    TR1fss.UP =   ETOL ;
    TJ1fss.LO = - ETOL ;
    TJ1fss.UP =   ETOL ;
    FJ1fss.LO = - ETOL ;
    FJ1fss.UP =   ETOL ;

*-----------------------------------------------------------------------------

* Upper and Lower Bounds on Profile

    Tr1.LO(i,j) $ SXCOL(i,j) = Tr1LO ;
    Tr1.UP(i,j) $ SXCOL(i,j) = Tr1UP ;

    Fj1.LO(i,j) $ SXCOL(i,j) = Fj1LO ;
    Fj1.UP(i,j) $ SXCOL(i,j) = Fj1UP ;

    Fj1c.LO(i,j) $ SXCOL(i,j) = Fj1LO ;
    Fj1c.UP(i,j) $ SXCOL(i,j) = Fj1UP ;

    Cr1n.LO = Cr1LO ;
    Cr1n.UP = Cr1UP ;

    Tr1n.LO = Tr1LO ;
    Tr1n.UP = Tr1UP ;

    Tj1n.LO = Tj1LO ;
    Tj1n.UP = Tj1UP ;

    Fj1n.LO = Fj1LO ;
    Fj1n.UP = Fj1UP ;

    Fj1cn.LO = Fj1LO ;
    Fj1cn.UP = Fj1UP ;

    Dr1.LO = Dr1LO ;
    Dr1.UP = Dr1UP ;

    Hr1.LO = Hr1LO ;
    Hr1.UP = Hr1UP ;
```



```
    Vr1.LO = pi/4 * SQR(Dr1.LO)*HR1.LO ;
    Vr1.UP = pi/4 * SQR(Dr1.UP)*HR1.UP ;

    Ar1.LO = pi * Dr1.LO*Hr1.LO;
    Ar1.UP = pi * Dr1.UP*Hr1.UP;

    Vj1.LO = 1/3 * Ar1.LO;
    Vj1.UP = 1/3 * Ar1.UP;

    taui.LO = tauiLO ;
    taui.UP = tauiUP ;

*=============================================================================
$INCLUDE 'ini_guess.gms'
*=============================================================================
```

GAMS - CASE 1 INITIAL GUESS CODE

```
* INITIAL VALUES

* Time varying
        Tr1.L(i,j) = 619.3;
        Fj1.L(i,j) $ SXCOL(i,j) = 450 ;
        Fjic.L(i,j) $ SXCOL(i,j) = Fj1.L(i,j) ;

        Dr1.L = 24.80;
        Hr1.L = 8.307;

        Vr1.L = pi/4 * SQR(Dr1.L)*HR1.L;
        Ar1.L = pi*DR1.L*Hr1.L;

        Cr1.L(i,j) = Cr1i*Fr1 / (Fr1 + k0*exp(-Ear/Tr1.L(i,j))*Vr1.L);
        Tj1.L(i,j) = (Tj1i*Fj1.L(i,j) + Ua/(rhoj*Cpj) * Ar1.L *
                      Tr1.L(i,j))/(Fj1.L(i,j) + Ua/(rhoj*Cpj)*Ar1.L);

* Reactor 1
        Tr1n.L = Tr1.L('I1','J1');
        Cr1n.L = Cr1i*Fr1 / (Fr1 + k0*exp(-Ear/Tr1n.L)*Vr1.L);

        Cr1end.L = Cr1n.L;
        Tr1end.L = Tr1n.L;

* Reactor 1 jacket
        Vj1.L = 1/3 * Ar1.L;
        Fj1n.L = 450;
        Fj1cn.L = Fj1n.L ;
        Tj1n.L = (Tj1i*Fj1n.L + Ua/(rhoj*Cpj)*Ar1.L*Tr1n.L)/
                 (Fj1n.L + Ua/(rhoj*Cpj)*Ar1.L);

        Tj1end.L = Tj1n.L;
        Fj1end.L = Fj1n.L;
        Fj1cend.L = Fj1cn.L;

* Controller parameters
        Kc.L = -300;
        taui.L = 1;

* Binary variables
        zL.L(i) = 0 ;
        zU.L(i) = 0 ;

* Cost variables
        CCap.L = 1916.9*exp(1.066*log(Dr1.L))*exp(0.802*log(Hr1.L));
        CUtil.L = 32.77*(Fj1n.L);  ;
        CTOT.L = CCAP.L + 4 * CUTIL.L ;

        EoTn.L = -EaR/Tr1n.L ;
        expEoTn.L = exp(EoTn.L) ;
        CR1ssS1.L = k0 * expEoTn.L * Vr1.L * Cr1n.L ;
        CR1ssS2.L = Cr1i*Fr1 - Cr1n.L*Fr1 - CR1ssS1.L ;
        TR1ssS1.L = dH/(rho*Cp) * k0 * expEoTn.L * Vr1.L * Cr1n.L ;
        TR1ssS2.L = Ua/(rho*Cp) * Ar1.L * ( Tr1n.L - Tj1n.L ) ;
        TR1ssS3.L = Tr1in*Fr1 - Tr1n.L*Fr1 - TR1ssS1.L - TR1ssS2.L ;
```


GAMS - CASE 2 MODEL EQUATION CODE

```
$TITLE      Schweiger and Floudas Binary Distillation example
$OFFUPPER
$OFFSYMXREF OFFSYMLIST
$OFFDIGIT

*=============================================================================

SCALARS
    NK      # number of trays to use            / 15 /
    NF      # of feed tray                      /  5 /
    NCOL    actual # coll. pts to use          /  2 / ;

SCALARS
    NTOL    # points within tolerance           /   5 /
    TTOT    actual final time                  / 100 / ;

SCALARS
    TDSTP1  time of first disturbance step     / 5 / ;

SCALAR
    TSTEP   desired controller timestep        / 2 /
    NFE     # of finite elements per timestep  / 1 / ;

*-----------------------------------------------------------------------------

* Problem Parameters

SCALARS
    alpha   relative volatility                  / 2.5    /
    hw      height over weir [m]                 / 0.0254 /
    Cpay    payback period [yr]                  / 4      /
    Ctax    tax factor                           / 0.4    / ;

*-----------------------------------------------------------------------------

* Physical parameters

SCALARS
    zss     ss feed composition[mol frac]        / 0.45  /
    Fss     ss feed molar flowrate [kmol.min^-1] / 1.0   /
    dstep   disturbance step size                / 0.045 /
    ;

*-----------------------------------------------------------------------------

* Upper and lower variable bounds

SCALARS
    VcLO    / 1.30 /
    VcUP    / 1.54 /
    RcLO    / 0.95 /
    RcUP    / 0.99 /

    KvLO    / - 100 /
    KvUP    / -0.01 /
    KrLO    /  0.01 /
    KrUP    /   100 /

    tauvLO  /  0.01 /
    tauvUP  /   100 /
    taurLO  /  0.01 /
    taurUP  /   100 /

    xBssUP  /  0.02 /
    xDssLO  /  0.95 /
    ;

*-----------------------------------------------------------------------------
```

```
* Tolerance bounds

SCALARS
    xBtol   ss trajectory           / 1e-3 /
    xDtol   ss trajectory           / 1e-3 /;

*-------------------------------------------------------------------------------

SCALARS
    psi     / 10 / ;

*-------------------------------------------------------------------------------

    OPTION LIMCOL   =        0 ;
    OPTION LIMROW   =        0 ;
    OPTION SOLPRINT =       ON ;
    OPTION SYSOUT   =      OFF ;
    OPTION ITERLIM  =  1000000 ;
    OPTION RESLIM   =    28800 ;
*-------------------------------------------------------------------------------

    OPTION NLP      =   CONOPT2 ;
    OPTION RMINLP   =   CONOPT2 ;
    OPTION MINLP    =   DICOPT  ;

*===============================================================================

SETS
    K       # of trays (max 30)             / K1*K30  /
    I       finite elements # (max 300)     / I1*I300 /
    J       collocation coeff. #            / J1*J5   /
    COL     # possible coll pt (max 3)      / C1*C3   /

ALIAS
    (K,KK,KS,KY,KU,KD)
    (J,JP,JJ,JS)
    (I, II);

*===============================================================================

SCALARS NFETOT      actual # of FE used ;
    NFETOT = (ttot / tstep) * nfe ;

*===============================================================================

ABORT $ (nk GT 30)      "Error in defining NK   (max 30)", nk   ;
ABORT $ (nfetot GT 300) "Error in defining FE   (max 300)", nfetot ;
ABORT $ (ncol GT 3)     "Error in defining NCOL (max  3)", ncol ;

*===============================================================================

SCALAR NCOF    equal to ncol+1 ;
    NCOF = ncol + 1 ;

SCALAR NCOT    equal to ncol+2 ;
    NCOT = ncol + 2;

*===============================================================================

SET SXCOL(i,jp)         actual dim of coll. coeff. (XCOL)   ;
    SXCOL(i,jp)     =   YES $ ( (ORD(i) LE nfetot)
                            $ (ORD(jp) LE ncof) ) ;

SET SKXCOL(k,i,jp)      actual dim of coll. coeff. (XCOL)   ;
    SKXCOL(k,i,jp)  =   YES $ ( (ORD(k) LE nk) $ (ORD(i) LE nfetot)
                            $ (ORD(jp) LE ncof) ) ;

SET SPHIPR(j,jp)        actual dim of PHIPR ;
    SPHIPR(j,jp)    =   YES $ ( (ORD(j) GT 1) $ (ORD(j) LE ncot)
```



```
                            $ (ORD(jp) LE ncof) ) ;

SET SDPHI(jp)           actual dim of denominator of PHI ;
    SDPHI(jp)       =   YES $ ( (ORD(jp) LE ncof) ) ;

SET SNFE(i)             actual # of applicable finite elements ;
    SNFE(i)         =   YES $ (ORD(i) LE nfetot) ;

SET SRES(i,j)           actual dim of residual eq ;
    SRES(i,j)       =   YES $ ( (ORD(i) LE nfetot) $ (ORD(j) GT 1)
                            $   (ORD(j) LE ncof) )  ;

SET SKRES(k,i,j)        actual dim of residual eq ;
    SKRES(k,i,j)    =   YES $ ( (ORD(k) LE nk) $ (ORD(i) LE nfetot)
                            $ (ORD(j) GT 1) $   (ORD(j) LE ncof) )  ;

SET SCONT(i)            actual dim of continuity eq ;
    SCONT(i)        =   YES $ ( (ORD(i) GT 1) $ (ORD(i) LE nfetot) );

SET SKCONT(k,i)         actual dim of continuity eq ;
    SKCONT(k,i)     =   YES $ ( (ORD(k) LE nk) $ (ORD(i) GT 1)
                            $ (ORD(i) LE nfetot) );

SET SCTRL(i)            actual dim of control eq ;
    SCTRL(i)        =   YES $ (ORD(i) LE nfetot) ;

SET STCTRL(i)           actual dim of control eq ;
    STCTRL(i)       =   YES $ ((ORD(i) LE nfetot) $ (mod(ORD(i),nfe) EQ 0)) ;

SET SCTRLINI(i)         control starting point ;
    SCTRLINI(i)     =   YES $ (ORD(i) EQ 1) ;

SET SXINI(i,j)          starting point ;
    SXINI(i,j)      =   YES $ ( (ORD(i) EQ 1) $ (ORD(j) EQ 1) ) ;

SET SKINI(k,i,j)        starting point ;
    SKINI(k,i,j)    =   YES $ ( (ORD(k) LE nK) $ (ORD(i) EQ 1)
                            $   (ORD(j) EQ 1) ) ;

SET STRAY_1(k)          first tray ;
    STRAY_1(k)      =   YES $ ( (ORD(k) EQ 1) );

SET STRAY_n(k)          final tray ;
    STRAY_n(k)      =   YES $ ( (ORD(k) EQ nK) );

SET STRAY(k)            tray ;
    STRAY(k)        =   YES $ ( (ORD(k) LE nK) ) ;

SET SSLKLv(i)            lower slack variable points for V;
    SSLKLv(i)        =   NO ;

SET SSLKUv(i)            upper slack variable points for V;
    SSLKUv(i)        =   NO ;

SET SSLKLr(i)            lower slack variable points for R;
    SSLKLr(i)        =   NO ;

SET SSLKUr(i)            upper slack variable points for R;
    SSLKUr(i)        =   NO ;

*===============================================================================

PARAMETERS
    TAU(jp)         tau at specified level
    PHIPR(j,jp)     1-st deriv of phi
    DPHI(jp)        denominator of phi
*    NEWSLK          current # slack added
    CSTEP           size of finite element
```

```
    TIME(i)         time at i ;

*-------------------------------------------------------------------------------

TABLE   GENTAU(col,jp)      the roots of Langrange polynomial

    J2                      J3
C1  .5                      1.
C2  .211324865405187        .788675134594813
C3  .1127016653792585       .5

+   J4                      J5
C1  0                       0
C2  1.                      0
C3   .8872983346207415      1. ;

*-------------------------------------------------------------------------------

* Assign tau according to the specified NCOL

TAU(jp) $ ( ORD(jp) LE ncot ) =
    gentau('C1',jp) $ (ncol EQ 1) +
    gentau('C2',jp) $ (ncol EQ 2) +
    gentau('C3',jp) $ (ncol EQ 3) ;

*-------------------------------------------------------------------------------

* Calculate DPHI (needed for calculating PHIPR)

DPHI(JP) $ SDPHI(JP) =
    PROD(J $ ( (ORD(J) LE ncof) $ (ORD(J) NE ORD(JP)) ),
             ( TAU(JP) - TAU(J)) ) ;

*-------------------------------------------------------------------------------

* Calculate PHIPR (1-st dervative of PHI)

PHIPR(J,JP) $ SPHIPR(J,JP) =
    SUM (JS $ ( (ORD(JS) LE ncof) $ (ORD(JS) NE ORD(JP)) ),
    PROD(JJ $ ( (ORD(JJ) LE ncof) $ (ORD(JJ) NE ORD(JP))
             $ (ORD(JJ) NE ORD(JS)) ),
               (TAU(J) - TAU(JJ)) )) / DPHI(JP) ;

*-------------------------------------------------------------------------------

* Set cstep

CSTEP = TTDT/NFETOT ;

TIME(i) $ SNFE(i) = ORD(i) * CSTEP ;

*-------------------------------------------------------------------------------

* Set tolerance sets

SET STOL(i)         tolerance envelope ;
    STOL(i)     =   YES $((
                    ( (ORD(i) GT (nfetot - ntol) ) AND (ORD(i) LE nfetot) ) )
                        $ (ORD(i) LE nfetot) );
*-------------------------------------------------------------------------------

* Feed flowrate
PARAMETER F(i)          Feed flowrate ;
F(i) $ SNFE(i) = Fss ;

* Feed composition disturbance
PARAMETER z(i)          Feed composition disturbance ;
z(i) $ SNFE(i) = zss + dstep$(time(i) GE tdstp1) ;
*z(i) $ SNFE(i) = zss ;
```



```
* Column structure
PARAMETER p(k)          Feed tray position ;
PARAMETER q(k)          Reflux tray position ;

p(k)$STRAY(k) = 0 ;
q(k)$STRAY(k) = 0 ;
p('K5')$STRAY('K5') = 1 ;
q('K15')$STRAY('K15') = 1 ;

*-------------------------------------------------------------------------------

* Output variables

PARAMETER
    xBOUT(i)
    xDOUT(i)
    VOUT(i)
    ROUT(i)
    zOUT(i)
    xBerrOUT(i)
    xDerrOUT(i)
    VcOUT(i)
    RcOUT(i)
    xBmOUT(i)
    xDmOUT(i)
    ;

FILE
    case2nlpout       /case2nlp.xls/
    case2rminlpout    /case2rminlp.xls/
    case2minlpout     /case2minlp.xls/
    cplexopt            /cplex.opt/
    conoptopt           /conopt.opt/
    conopt2opt          /conopt2.opt/
    dicoptopt           /dicopt.opt/
     ;

*===============================================================================

VARIABLES

* Time varying
    x(k,i,j)        liquid composition on plate k
    y(k,i,j)        vapour composition on plate k
    L(k,i,j)        liquid flowrate from plate k
    B(i,j)          bottoms flowrate
    D(i,j)          distillate flowrate
    V(i,j)          vapour boilup
    R(i,j)          reflux flowrate

    L1(i,j)         tray 1 liquid flowrate

    x1(i,j)         tray 1 liquid composition
    xB(i,j)         bottoms lquid composition
    xD(i,j)         distillate liquid composition

    yB(i,j)         bottoms vapour composition
    yN(i,j)         bottoms vapour composition

    Vc(i,j)         controller vapour boilup
    Rc(i,j)         controller reflux rate

    xBerr(i)        setpoint tracking error for bottoms
    xDerr(i)        setpoint tracking error for distillate

    sLv(i)          lower slack for vapor boilup
    sUv(i)          upper slack for vapor biolup
    sLr(i)          lower slack for reflux rate
```

```
    sUr(i)          upper slack for reflux rate

    zLv(i)          lower binary for vapor boilup
    zUv(i)          upper binary for vapor biolup
    zLr(i)          lower binary for reflux rate
    zUr(i)          upper binary for reflux rate

* Time invariant
    MM              tray liquid holdup
    MB              reboiler tray liquid holdup
    MD              condensor tray liquid holdup

    beta            tray hydraulic time constant
    betaB           reboiler tray hydraulic time constant
    betaD           condensor tray hydraulic time constant

* Intermediate variables

    xBm(i,j)
    d1xBm(i,j)
    d2xBm(i,j)
    d3xBm(i,j)
    d4xBm(i,j)
    d5xBm(i,j)

    xDm(i,j)
    d1xDm(i,j)
    d2xDm(i,j)
    d3xDm(i,j)
    d4xDm(i,j)
    d5xDm(i,j)

    xBmss
    d1xBmss
    d2xBmss
    d3xBmss
    d4xBmss
    d5xBmss

    xDmss
    d1xDmss
    d2xDmss
    d3xDmss
    d4xDmss
    d5xDmss

* Physical constant variables

    MMTndx1         intermediate variable
    BTndx1          intermediate variable

* Steady state variables

    xss(k)          steady state liquid composition on tray k
    yss(k)          steady state vapour composition on tray k
    Lss(k)          steady state liquid flowrate on tray k

    Vcss            steady state controller vapour boilup
    Rcss            steady state controller reflux rate

    L1ss            steady state liquid flowrate on tray 1
    x1ss            steady state liquid composition on tray 1
    xBss            steady state bottoms liquid composition
    xDss            steady state distillate liquid composition
    yBss            steady state bottoms vapour composition
    yNss            steady state distillate vapour composition
    Bss             steady state bottoms flowrate
    Dss             steady state distillate flowrate
```



```
* Cost variables
    OBJ             objective function value
    CTot            total cost
    Cutil           utility cost
    Ccap            capital cost
    Nt              # of trays
    Dc              column diameter
    Rss             steady state reflux ratio
    Vss             steady state vapour boilup
    kV              vapour boilup controller gain
    tauV            vapour boilup controller time constant
    kR              reflux controller gain
    tauR            reflux controller time constant
    ;

*-------------------------------------------------------------------------------

BINARY VARIABLES

* Slack variables
    zLv(i)          lower binary for vapor boilup
    zUv(i)          upper binary for vapor biolup
    zLr(i)          lower binary for reflux rate
    zUr(i)          upper binary for reflux rate
    ;

*===============================================================================

* EQUATIONS

*-------------------------------------------------------------------------------

* Variable definitions

EQUATION Ex1def(i,j)            define variable x1 ;
Ex1def(i,j) $ SXCOL(i,j) ..
    x1(i,j) =E= x('K1',i,j) ;

EQUATION EyNdef(i,j)            define variable yN ;
EyNdef(i,j) $ SXCOL(i,j) ..
    yN(i,j) =E= SUM(k $ (ORD(k) LE nK), y(k,i,j)*Q(k)) ;

EQUATION EL1def(i,j)            define variable L1 ;
EL1def(i,j) $ SXCOL(i,j) ..
    L1(i,j) =E= L('K1',i,j) ;

*-------------------------------------------------------------------------------

* residual equations

* Reboiler component balances
EQUATION ERES1(i,j)             Reboiler component balances ;
ERES1(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), xB(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    1/MB * ( L1(i,j)*(x1(i,j) - xB(i,j))) +
    1/MB * ( V(i,j)*(xB(i,j) - yB(i,j)) )
    )
    =E= 0 ;

* Tray component balances
EQUATION ERES2(k,i,j)           Tray component balances ;
ERES2(k,i,j)  $ SKRES(k,i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), x(k,i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    1/MM * ( L(k+1,i,j)$((ORD(k)+1) LE nK) * (x(k+1,i,j) - x(k,i,j))) +
    1/MM * ( V(i,j)*(yB(i,j)$STRAY_1(k) + y(k-1,i,j) - y(k,i,j))) +
    1/MM * (P(k)*F(i)*(z(i) - x(k,i,j)) + Q(k)*R(i,j)*(xD(i,j) - x(k,i,j)))
```

```
    )
    =E= 0 ;

* Condenser component balances
EQUATION ERES3(i,j)             Condenser component balances ;
ERES3(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), xD(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    1/MD * ( V(i,j)*yN(i,j)) -
    1/MD * ( V(i,j)*xD(i,j))
    )
    =E= 0 ;

* Reboiler total balance
EQUATION ERES4(i,j)             Reboiler total balance ;
ERES4(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), B(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    1/betaB * (L1(i,j) ) -
    1/betaB * ( V(i,j) ) -
    1/betaB * ( B(i,j) )
    )
    =E= 0 ;

* Tray total balances
EQUATION ERES5(k,i,j)           Tray total balances ;
ERES5(k,i,j)  $ SKRES(k,i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), L(k,i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    1/beta * ( L(k+1,i,j)$( (ORD(k)+1) LE nK)) +
    1/beta * ( - L(k,i,j)) +
    1/beta * ( P(k)*F(i) + Q(k)*R(i,j) )
    )
    =E= 0 ;

* Condenser total balances
EQUATION ERES6(i,j)             Condenser total balances ;
ERES6(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), D(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    1/betaD * ( V(i,j) ) -
    1/betaD * ( R(i,j) ) -
    1/betaD * ( D(i,j) )
    )
    =E= 0 ;

* Measurement lag derivative B 1
EQUATION ERES7(i,j)             Measurement lag derivative B 1 ;
ERES7(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), xBm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    d1xBm(i,j)
    )
    =E= 0 ;

* Measurement lag derivative B 2
EQUATION ERES8(i,j)             Measurement lag derivative B 2 ;
ERES8(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), d1xBm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    d2xBm(i,j)
    )
    =E= 0 ;

* Measurement lag derivative B 3
EQUATION ERES9(i,j)             Measurement lag derivative B 3 ;
```



```
ERES9(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), d2xBm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    d3xBm(i,j)
    )
    =E= 0 ;

* Measurement lag derivative B 4
EQUATION ERES10(i,j)            Measurement lag derivative B 4 ;
ERES10(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), d3xBm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    d4xBm(i,j)
    )
    =E= 0 ;

* Measurement lag derivative B 5
EQUATION ERES11(i,j)            Measurement lag derivative B 5 ;
ERES11(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), d4xBm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    d5xBm(i,j)
    )
    =E= 0 ;

* Measurement lag derivative D 1
EQUATION ERES12(i,j)            Measurement lag derivative D 1 ;
ERES12(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), xDm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    d1xDm(i,j)
    )
    =E= 0 ;

* Measurement lag derivative D 2
EQUATION ERES13(i,j)            Measurement lag derivative D 2 ;
ERES13(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), d1xDm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    d2xDm(i,j)
    )
    =E= 0 ;

* Measurement lag derivative D 3
EQUATION ERES14(i,j)            Measurement lag derivative D 3 ;
ERES14(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), d2xDm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    d3xDm(i,j)
    )
    =E= 0 ;

* Measurement lag derivative D 4
EQUATION ERES15(i,j)            Measurement lag derivative D 4 ;
ERES15(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), d3xDm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    d4xDm(i,j)
    )
    =E= 0 ;

* Measurement lag derivative D 5
EQUATION ERES16(i,j)            Measurement lag derivative D 5 ;
ERES16(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), d4xDm(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
```

```
    d5xDm(i,j)
    )
    =E= 0 ;

* Final control lag on V
EQUATION ERES17(i,j)            Final control lag on V ;
ERES17(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), V(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    1/0.9 * (Vc(i,j) - V(i,j))
    )
    =E= 0 ;

* Final control lag on R
EQUATION ERES18(i,j)            Final control lag on R ;
ERES18(i,j)  $ SRES(i,j) ..
    SUM(jp $ (ORD(jp) LE ncof), R(i,jp) * PHIPR(j,jp) ) -
    CSTEP * (
    1/0.5 * ( Rc(i,j) - R(i,j) )
    )
    =E= 0 ;

*-------------------------------------------------------------------------------

* continuity equations

EQUATION ECONT1(i) ;
ECONT1(i) $ SCONT(i) ..
    xB(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), xB(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT2(k,i) ;
ECONT2(k,i) $ SKCONT(k,i) ..
    x(k,i,'J1') =E= SUM(j $ (ORD(j) LE ncof), x(k,i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT3(i) ;
ECONT3(i) $ SCONT(i) ..
    xD(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), xD(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT4(i) ;
ECONT4(i) $ SCONT(i) ..
    B(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), B(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT5(k,i) ;
ECONT5(k,i) $ SKCONT(k,i) ..
    L(k,i,'J1') =E= SUM(j $ (ORD(j) LE ncof), L(k,i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT6(i) ;
ECONT6(i) $ SCONT(i) ..
    D(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), D(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT7(i) ;
ECONT7(i) $ SCONT(i) ..
    xBm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), xBm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;
```



```
EQUATION ECONT8(i) ;
ECONT8(i) $ SCONT(i) ..
    d1xBm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), d1xBm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT9(i) ;
ECONT9(i) $ SCONT(i) ..
    d2xBm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), d2xBm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT10(i) ;
ECONT10(i) $ SCONT(i) ..
    d3xBm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), d3xBm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT11(i) ;
ECONT11(i) $ SCONT(i) ..
    d4xBm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), d4xBm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT12(i) ;
ECONT12(i) $ SCONT(i) ..
    xDm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), xDm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT13(i) ;
ECONT13(i) $ SCONT(i) ..
    d1xDm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), d1xDm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT14(i) ;
ECONT14(i) $ SCONT(i) ..
    d2xDm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), d2xDm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT15(i) ;
ECONT15(i) $ SCONT(i) ..
    d3xDm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), d3xDm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT16(i) ;
ECONT16(i) $ SCONT(i) ..
    d4xDm(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), d4xDm(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT17(i) ;
ECONT17(i) $ SCONT(i) ..
    V(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), V(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

EQUATION ECONT18(i) ;
ECONT18(i) $ SCONT(i) ..
    R(i,'J1') =E= SUM(j $ (ORD(j) LE ncof), R(i-1,j) *
            PROD(jp $ (ORD(jp) NE ORD(j) AND ORD(jp) LE ncof),
                ( (1.0 - tau(jp)) / (tau(j) - tau(jp)) ) ) ) ;

*-------------------------------------------------------------------------------
```

```
* Reboiler equilibrium

EQUATION EREQBM(i,j)                Reboiler equilibrium ;
EREQBM(i,j) $ SXCOL(i,j) ..
    yB(i,j) =E= ( alpha * xB(i,j) ) / (1 + xB(i,j)*(alpha - 1)) ;

* Tray equilibrium

EQUATION ETEQBM(k,i,j)              Tray equilibrium ;
ETEQBM(k,i,j) $ SKXCOL(k,i,j) ..
    y(k,i,j) =E= (alpha * x(k,i,j) ) / ( 1 + x(k,i,j)*(alpha - 1)) ;

*-------------------------------------------------------------------------------

* Measurement control lag

EQUATION EBMLAG(i,j)            bottoms measurement lag ;
EBMLAG(i,j) $ SXCOL(i,j) ..
    d5xBm(i,j) + 5*d4xBm(i,j) + 10*d3xBm(i,j) + 10*d2xBm(i,j) + 5*d1xBm(i,j) +
    xBm(i,j) =E= xB(i,j) ;

EQUATION EDMLAG(i,j)            distillate measurement lag ;
EDMLAG(i,j) $ SXCOL(i,j) ..
    d5xDm(i,j) + 5*d4xDm(i,j) + 10*d3xDm(i,j) + 10*d2xDm(i,j) + 5*d1xDm(i,j) +
    xDm(i,j) =E= xD(i,j) ;

*-------------------------------------------------------------------------------

* steady state balances
* Variable definitions

EQUATION Ex1ssdef           define variable x1 ;
Ex1ssdef ..
    x1ss =E= xss('K1') ;

EQUATION EyNssdef           define variable yN ;
EyNssdef(k) $ STRAY_n(k) ..
    yNss =E= yss(k) ;

EQUATION EL1ssdef           define variable L1 ;
EL1ssdef ..
    L1ss =E= Lss('K1') ;

*-------------------------------------------------------------------------------

* Steady state residual equations

EQUATION ExBss      ss reboiler component balance ;
ExBss ..
    L1ss * (x1ss - xBss) + Vss * (xBss - yBss)
    =E= 0 ;

EQUATION ExKss(k)       ss tray component balance ;
ExKss(k) $ STRAY(k) ..
    Lss(k+1) * ( xss(k+1) - xss(k) )$((ORD(k)+1) LE nK) +
    Vss * (yss(k-1) - yss(k) + yBss $ STRAY_1(k)) +
    P(k) * Fss * (zss - xss(k)) + Q(k) * Rss * (xDss - xss(k))
    =E= 0 ;

EQUATION ExDss          ss condensor component balance ;
ExDss ..
    Vss * ( yNss - xDss )
    =E= 0 ;

EQUATION EBss           ss reboiler total balance ;
EBss ..
    Lss('K1') - Vss - Bss
    =E= 0 ;
```



```
EQUATION EKss(k)            ss tray total balances ;
EKss(k) $ STRAY(k) ..
    Lss(k+1)$((ORD(k)+1) LE nK)  - Lss(k) + P(k) * Fss + Q(k) * Rss
    =E= 0 ;

EQUATION EDss           ss condensor total balance ;
EDss ..
    Vss - Rss - Dss
    =E= 0 ;

*-------------------------------------------------------------------------------

* Reboiler equilibrium

EQUATION ERssEQBM ;
ERssEQBM ..
    yBss =E= ( alpha * xBss ) / (1 + xBss*(alpha - 1)) ;

* Tray equilibrium

EQUATION ETssEQBM(k) ;
ETssEQBM(k) $ STRAY(k) ..
    yss(k) =E= (alpha * xss(k) ) / ( 1 + xss(k)*(alpha - 1)) ;

*-------------------------------------------------------------------------------

* Steady state measurement lag approximation derivatives

EQUATION Ed1xBmss       ss bottoms measurement lag derivative 1;
Ed1xBmss ..
    d1xBmss
    =E= 0 ;

EQUATION Ed2xBmss       ss bottoms measurement lag derivative 2;
Ed2xBmss ..
    d2xBmss
    =E= 0 ;

EQUATION Ed3xBmss       ss bottoms measurement lag derivative 3;
Ed3xBmss ..
    d3xBmss
    =E= 0 ;

EQUATION Ed4xBmss       ss bottoms measurement lag derivative 4;
Ed4xBmss ..
    d4xBmss
    =E= 0 ;

EQUATION Ed5xBmss       ss bottoms measurement lag derivative 5;
Ed5xBmss ..
    d5xBmss
    =E= 0 ;

EQUATION Ed1xDmss       ss distillate measurement lag derivative 1;
Ed1xDmss ..
    d1xDmss
    =E= 0 ;

EQUATION Ed2xDmss       ss distillate measurement lag derivative 2;
Ed2xDmss ..
    d2xDmss
    =E= 0 ;

EQUATION Ed3xDmss       ss distillate measurement lag derivative 3;
Ed3xDmss ..
    d3xDmss
    =E= 0 ;

EQUATION Ed4xDmss       ss distillate measurement lag derivative 4;
Ed4xDmss ..
    d4xDmss
```

```
    =E= 0 ;

EQUATION Ed5xDmss        ss distillate measurement lag derivative 5;
Ed5xDmss ..
    d5xDmss
    =E= 0 ;

EQUATION EVss            ss vapour flowrate final control lag ;
EVss ..
    Vcss - Vss
    =E= 0 ;

EQUATION ERss            ss reboiler flowrate final control lag ;
ERss ..
    Rcss - Rss
    =E= 0 ;

*-------------------------------------------------------------------------------

* Measurement control lag

EQUATION EBMLAGss        ss bottoms measurement lag ;
EBMLAGss ..
    d5xBmss + 5*d4xBmss + 10*d3xBmss + 10*d2xBmss + 5*d1xBmss +
    xBmss =E= xBss ;

EQUATION EDMLAGss        ss distillate measurement lag ;
EDMLAGss ..
    d5xDmss + 5*d4xDmss + 10*d3xDmss + 10*d2xDmss + 5*d1xDmss +
    xDmss =E= xDss ;

*-------------------------------------------------------------------------------

* Initial variable values for residual equations

EQUATION Exini(k,i,j)        initial liquid composition on plate k ;
Exini(k,i,j) $ SKINI(k,i,j) ..
    x(k,i,j) =E= xss(k) ;

EQUATION Eyini(k,i,j)        initial vapour composition on plate k ;
Eyini(k,i,j) $ SKINI(k,i,j) ..
    y(k,i,j) =E= yss(k) ;

EQUATION ELini(k,i,j)        initial liquid flowrate from plate k ;
ELini(k,i,j) $ SKINI(k,i,j) ..
    L(k,i,j) =E= Lss(k) ;

EQUATION EBini(i,j)          initial bottoms flowrate ;
EBini(i,j) $ SXINI(i,j) ..
    B(i,j) =E= Bss ;

EQUATION EDini(i,j)          initial distillate flowrate ;
EDini(i,j) $ SXINI(i,j) ..
    D(i,j) =E= Dss ;

EQUATION EVini(i,j)          initial vapour boilup ;
EVini(i,j) $ SXINI(i,j) ..
    V(i,j) =E= Vss ;

EQUATION ERini(i,j)          initial reflux flowrate ;
ERini(i,j) $ SXINI(i,j) ..
    R(i,j) =E= Rss ;

EQUATION EL1ini(i,j)         initial liquid flowrate on tray 1 ;
EL1ini(i,j) $ SXINI(i,j) ..
    L1(i,j) =E= L1ss ;

EQUATION Ex1ini(i,j)         initial liquid composition on  tray 1 ;
Ex1ini(i,j) $ SXINI(i,j) ..
```



```
    x1(i,j) =E= x1ss ;

EQUATION ExBini(i,j)         initial bottoms liquid composition ;
ExBini(i,j) $ SXINI(i,j) ..
    xB(i,j) =E= xBss ;

EQUATION ExDini(i,j)         initial distillate liquid composition ;
ExDini(i,j) $ SXINI(i,j) ..
    xD(i,j) =E= xDss ;

EQUATION EyBini(i,j)         initial bottoms vapour composition ;
EyBini(i,j) $ SXini(i,j) ..
    yB(i,j) =E= yBss ;

EQUATION EyNini(i,j)         initial vapour composition on tray N ;
EyNini(i,j) $ SXINI(i,j) ..
    yN(i,j) =E= yNss ;

EQUATION EVcini(i,j)         initial delayed steam flowrate ;
EVcini(i,j) $ SXINI(i,j) ..
    Vc(i,j) =E= Vcss ;

EQUATION ERcini(i,j)         initial delayed reflux flowrate ;
ERcini(i,j)  $ SXINI(i,j) ..
    Rc(i,j) =E= Rcss ;

EQUATION ExBmini(i,j) ;
ExBmini(i,j) $ SXINI(i,j) ..
    xBm(i,j) =E= xBmss ;

EQUATION Ed1xBmini(i,j) ;
Ed1xBmini(i,j) $ SXINI(i,j) ..
    d1xBm(i,j) =E= d1xBmss ;

EQUATION Ed2xBmini(i,j) ;
Ed2xBmini(i,j) $ SXINI(i,j) ..
    d2xBm(i,j) =E= d2xBmss ;

EQUATION Ed3xBmini(i,j) ;
Ed3xBmini(i,j) $ SXINI(i,j) ..
    d3xBm(i,j) =E= d3xBmss ;

EQUATION Ed4xBmini(i,j) ;
Ed4xBmini(i,j) $ SXINI(i,j) ..
    d4xBm(i,j) =E= d4xBmss ;

EQUATION Ed5xBmini(i,j) ;
Ed5xBmini(i,j) $ SXINI(i,j) ..
    d5xBm(i,j) =E= d5xBmss ;

EQUATION ExDmini(i,j) ;
ExDmini(i,j) $ SXINI(i,j) ..
    xDm(i,j) =E= xDmss ;

EQUATION Ed1xDmini(i,j) ;
Ed1xDmini(i,j) $ SXINI(i,j) ..
    d1xDm(i,j) =E= d1xDmss ;

EQUATION Ed2xDmini(i,j) ;
Ed2xDmini(i,j) $ SXINI(i,j) ..
    d2xDm(i,j) =E= d2xDmss ;

EQUATION Ed3xDmini(i,j) ;
Ed3xDmini(i,j) $ SXINI(i,j) ..
    d3xDm(i,j) =E= d3xDmss ;

EQUATION Ed4xDmini(i,j) ;
```

```
Ed4xDmini(i,j) $ SXINI(i,j) ..
    d4xDm(i,j) =E= d4xDmss ;

EQUATION Ed5xDmini(i,j) ;
Ed5xDmini(i,j) $ SXINI(i,j) ..
    d5xDm(i,j) =E= d5xDmss ;

*-------------------------------------------------------------------------------

* Tray liquid holdup

EQUATION EMB            calculate bottoms holdup ;
EMB ..
    MB =E= 10 * MM ;

EQUATION EMMT1          intermediate equation ;
EMMT1 ..
    MMTNDX1 =E= exp( (2/3)*( log((0.0014134/Dc)) ) ) ;

EQUATION EMMT           calculate tray holdup ;
EMMT ..
    MM =E= 7.538115*( MMTndx1 + hw ) * SQR(Dc) ;

EQUATION EMD            calculate distillate holdup ;
EMD ..
    MD =E= 10 * MM ;
*-------------------------------------------------------------------------------

* Tray time constants

EQUATION EbetaB         calculate bottoms tray time constant ;
EbetaB ..
    betaB =E= 100*beta ;

EQUATION EbetaT1        intermediate equation ;
EbetaT1 ..
    BTNDX1 =E= exp( 1.3333 * log (Dc) );

EQUATION EbetaT         calculate tray time constant ;
EbetaT ..
    beta =E= 0.05271*(BTndx1) ;

EQUATION EbetaD         calculate distillate tray time constant ;
EbetaD ..
    betaD =E= 100*beta ;

*-------------------------------------------------------------------------------

* Flooding constraint

EQUATION EFLOOD         flooding constraint ;
EFLOOD ..
    Dc =G= 0.6719 * SQRT(Vss) ;

*-------------------------------------------------------------------------------

* Number of trays in column

EQUATION ENTRAYS        calc # of trays in column ;
ENTRAYS ..
    Nt =E= SUM(k $ STRAY(k), ORD(k)*Q(k)) ;

*-------------------------------------------------------------------------------

* Logical tray existence constraint

EQUATION EL_1FD         existence of single feed ;
EL_1FD ..
    SUM(k $ STRAY(k), P(k)) =E= 1 ;

EQUATION EL_1RFLX       existence of single reflux ;
```



```
EL_1RFLX ..
    SUM(k $STRAY(k), Q(k)) =E= 1 ;

EQUATION EL_FDTIN       feed above tray 4 ;
EL_FDTIN ..
    SUM(k $STRAY(k), ORD(k)*P(k)) =G= 4 ;

EQUATION EL_RFLXT10     reflux above tray 10 ;
EL_RFLXT10 ..
    SUM(k $STRAY(k), ORD(k)*Q(k)) =G= 10 ;

EQUATION EL_RFLXTIN     reflux 4 above feed ;
EL_RFLXTIN ..
    SUM(k $STRAY(k), (ORD(k)*Q(k)) - (ORD(k)*P(k)) ) =G= 4 ;

*-------------------------------------------------------------------------------

* Error equations

EQUATION ExBerr(i)      calc bottoms composition error ;
ExBerr(i) $ SNFE(i) ..
    xBerr(i) =E= xBss - xBm(i,'J1') ;

EQUATION ExDerr(i)      calc distillate composition error ;
ExDerr(i) $ SNFE(i) ..
    xDerr(i) =E= xDss - xDm(i,'J1') ;

*-------------------------------------------------------------------------------

* PI control equations

EQUATION EBCNTRL(i)     control bottoms composition ;
EBCNTRL(i) $ SCTRL(i) ..
    Vc(i,'J1') =E= Vss$(SCTRLINI(i)) + Vc(i-1,'J1') +
        Kv * ( xBerr(i) )$(STCTRL(i)) -
        Kv * ( xBerr(i-1) )$(STCTRL(i)) +
        Kv * ( tstep/tauv * xBerr(i) )$(STCTRL(i)) ;

EQUATION EDCNTRL(i)     control distillate composition ;
EDCNTRL(i) $ SCTRL(i) ..
    Rc(i,'J1') =E= Rss$(SCTRLINI(i)) + Rc(i-1,'J1') +
        Kr * ( xDerr(i) )$(STCTRL(i)) -
        Kr * ( xDerr(i-1) )$(STCTRL(i)) +
        Kr * ( tstep/taur * xDerr(i) )$(STCTRL(i)) ;


EQUATION EBCNTRLIS(i)     input saturation control bottoms composition ;
EBCNTRLIS(i) $ SCTRL(i) ..
    Vc(i,'J1') - sLv(i)$SSLKLv(i) + sUv(i)$SSLKUv(i) =E=
        Vss$(SCTRLINI(i)) + Vc(i-1,'J1') +
        Kv * ( xBerr(i) )$(STCTRL(i)) -
        Kv * ( xBerr(i-1) )$(STCTRL(i)) +
        Kv * ( tstep/tauv * xBerr(i) )$(STCTRL(i)) ;

EQUATION EDCNTRLIS(i)     input saturation control distillate composition ;
EDCNTRLIS(i) $ SCTRL(i) ..
    Rc(i,'J1') - sLr(i)$SSLKLr(i) + sUr(i)$SSLKUr(i) =E=
        Rss$(SCTRLINI(i)) + Rc(i-1,'J1') +
        Kr * ( xDerr(i) )$(STCTRL(i)) -
        Kr * ( xDerr(i-1) )$(STCTRL(i)) +
        Kr * ( tstep/taur * xDerr(i) )$(STCTRL(i)) ;

*-------------------------------------------------------------------------------

* Open Loop Equations

EQUATION EBOPEN(i)      open loop vapour flowrate step test ;
EBOPEN(i) $ SCTRL(i) ..
    Vc(i,'J1') =E= Vss$(SCTRLINI(i)) + Vc(i-1,'J1') ;
```

```
EQUATION EDOPEN(i)      open loop reflux rate step test ;
EDOPEN(i) $ SCTRL(i) ..
    Rc(i,'J1') =E= Rss$(SCTRLINI(i)) + Rc(i-1,'J1') ;

*-------------------------------------------------------------------------------

EQUATION EBCNTRLU(i,j)      zero order hold controller ;
EBCNTRLU(i,j) $ SXCOL(i,j) ..
    Vc(i,j) =E= Vc(i,'J1') ;

EQUATION EDCNTRLU(i,j)      zero order hold controller ;
EDCNTRLU(i,j) $ SXCOL(i,j) ..
    Rc(i,j) =E= Rc(i,'J1') ;

*-------------------------------------------------------------------------------

* slack variable bounds

EQUATION ESLLLv(i)      lower bounds on lower slack Vc ;
ESLLLv(i) $ (SSLKLv(i) AND SCTRL(i)) ..
    sLv(i) =G= 0 ;

EQUATION ESLLUv(i)      upper bounds on lower slack Vc ;
ESLLUv(i) $ (SSLKLv(i) AND SCTRL(i)) ..
    sLv(i) =L= psi * zLv(i) ;


EQUATION ESLLLr(i)      lower bounds on lower slack Rc ;
ESLLLr(i) $ (SSLKLr(i) AND SCTRL(i)) ..
    sLr(i) =G= 0 ;

EQUATION ESLLUr(i)      upper bounds on lower slack Rc ;
ESLLUr(i) $ (SSLKLr(i) AND SCTRL(i)) ..
    sLr(i) =L= psi * zLr(i) ;


EQUATION ESLULv(i)      lower bounds on upper slack Vc ;
ESLULv(i) $ (SSLKUv(i) AND SCTRL(i)) ..
    sUv(i) =G= 0 ;

EQUATION ESLUUv(i)      upper bounds on upper slack Vc ;
ESLUUv(i) $ (SSLKUv(i) AND SCTRL(i)) ..
    sUv(i) =L= psi * zUv(i) ;


EQUATION ESLULr(i)      lower bounds on upper slack Rc ;
ESLULr(i) $ (SSLKUr(i) AND SCTRL(i)) ..
    sUr(i) =G= 0 ;

EQUATION ESLUUr(i)      upper bounds on upper slack Vc ;
ESLUUr(i) $ (SSLKUr(i) AND SCTRL(i)) ..
    sUr(i) =L= psi * zUr(i) ;

*-------------------------------------------------------------------------------

* Actuator Bounds

EQUATION EZLLv(i)       lower bounds on lower actuator bound Vc ;
EZLLv(i) $ (SSLKLv(i) AND SCTRL(i)) ..
    VcLO - Vc(i,'J1') =G= -psi*(1 - zLv(i)) ;

EQUATION EZLUv(i)       upper bounds on lower actuator bound Vc ;
EZLUv(i) $ (SSLKLv(i) AND SCTRL(i)) ..
    VcLO - Vc(i,'J1') =L=  psi*(1 - zLv(i)) ;


EQUATION EZLLr(i)       lower bounds on lower actuator bound Rc ;
EZLLr(i) $ (SSLKLr(i) AND SCTRL(i)) ..
    RcLO - Rc(i,'J1') =G= -psi*(1 - zLr(i)) ;
```



```
EQUATION EZLUr(i)       upper bounds on lower actuator bound Rc ;
EZLUr(i) $ (SSLKLr(i) AND SCTRL(i)) ..
    RcLO - Rc(i,'J1') =L=  psi*(1 - zLr(i)) ;


EQUATION EZULv(i)       lower bounds on upper actuator bound Vc ;
EZULv(i) $ (SSLKUv(i) AND SCTRL(i)) ..
    VcUP - Vc(i,'J1') =G= -psi*(1 - zUv(i)) ;

EQUATION EZUUv(i)       upper bounds on upper actuator bound Vc ;
EZUUv(i) $ (SSLKUv(i) AND SCTRL(i)) ..
    VcUP - Vc(i,'J1') =L=  psi*(1 - zUv(i)) ;


EQUATION EZULr(i)       lower bounds on upper actuator bound Rc ;
EZULr(i) $ (SSLKUr(i) AND SCTRL(i)) ..
    RcUP - Rc(i,'J1') =G= -psi*(1 - zUr(i)) ;

EQUATION EZUUr(i)       upper bounds on upper actuator bound Rc ;
EZUUr(i) $ (SSLKUr(i) AND SCTRL(i)) ..
    RcUP - Rc(i,'J1') =L=  psi*(1 - zUr(i)) ;

*-------------------------------------------------------------------------------

* Cost Functions

EQUATION ECUTIL         utility cost ;
ECUTIL ..
    CUTIL =E= 7756*Vss ;

EQUATION ECCAP          capital cost ;
ECCAP ..
    CCAP =E= 3.075*( 615 + 324*SQR(Dc) + 486 * (6 + 0.76*Nt)*Dc ) +
    61.25*Nt*( 0.7 + 1.5*SQR(Dc) )

EQUATION ECTOTAL        total cost ;
ECTOTAL ..
    CTOT =E= 7756*Vss + 3.075*( 615 + 324*SQR(Dc) + 486 * (6 + 0.76*Nt)*Dc ) +
    61.25*Nt*( 0.7 + 1.5*SQR(Dc) );

*-------------------------------------------------------------------------------

* Objective Function

EQUATION EOBJ           objective function ;
EOBJ ..
    OBJ =E= Ctot ;

*===============================================================================

* MODELS

$include 'case2mdls.gms'

*===============================================================================
```

```
$TITLE      Schweiger and Floudas Binary Distillation example

*=============================================================================

MODEL CASE2NLP_3
    /
    Ex1def, EyNdef, EL1def,
    ERES1, ERES2, ERES3, ERES4, ERES5, ERES6, ERES7, ERES8, ERES9, ERES10,
    ERES11, ERES12, ERES13, ERES14, ERES15, ERES16, ERES17, ERES18,
    ECONT1, ECONT2, ECONT3, ECONT4, ECONT5, ECONT6, ECONT7, ECONT8, ECONT9,
    ECONT10, ECONT11, ECONT12, ECONT13, ECONT14, ECONT15, ECONT16, ECONT17,
    ECONT18,
    Ex1ssdef, EyNssdef, EL1ssdef, ExBss, ExKss, ExDss, EBss, EKss, EDss,
    ERssEQBM, ETssEQBM,
    Ed1xBmss, Ed2xBmss, Ed3xBmss, Ed4xBmss, Ed5xBmss,
    Ed1xDmss, Ed2xDmss, Ed3xDmss, Ed4xDmss, Ed5xDmss,
    EVss, ERss,
    EBMLAGss, EDMLAGss,
    Exini, Eyini, ELini, EBini, EDini, EVini, ERini, EL1ini, Ex1ini, ExBini,
    ExDini, EyBini, EyNini, EVcini, ERcini,
    ExBmini, Ed1xBmini, Ed2xBmini, Ed3xBmini, Ed4xBmini, Ed5xBmini,
    ExDmini, Ed1xDmini, Ed2xDmini, Ed3xDmini, Ed4xDmini, Ed5xDmini,
    EREQBM, ETEQBM,
    EBMLAG, EDMLAG,
    EMB, EMMT1, EMMT, EMD,
    EbetaB, EbetaT1, EbetaT, EbetaD,
    EFLOOD, ENTRAYS,
    EL_1FD, EL_1RFLX, EL_FDTIN, EL_RFLXT10, EL_RFLXTIN,
    ExBerr, ExDerr,
    EBCNTRL, EDCNTRL, EBCNTRLU, EDCNTRLU,
    ECUTIL, ECTOTAL,
    EOBJ
    / ;

*=============================================================================

MODEL CASE2MINLP_1
    /
    Ex1def, EyNdef, EL1def,
    ERES1, ERES2, ERES3, ERES4, ERES5, ERES6, ERES7, ERES8, ERES9, ERES10,
    ERES11, ERES12, ERES13, ERES14, ERES15, ERES16, ERES17, ERES18,
    ECONT1, ECONT2, ECONT3, ECONT4, ECONT5, ECONT6, ECONT7, ECONT8, ECONT9,
    ECONT10, ECONT11, ECONT12, ECONT13, ECONT14, ECONT15, ECONT16, ECONT17,
    ECONT18,
    Ex1ssdef, EyNssdef, EL1ssdef, ExBss, ExKss, ExDss, EBss, EKss, EDss,
    ERssEQBM, ETssEQBM,
    Ed1xBmss, Ed2xBmss, Ed3xBmss, Ed4xBmss, Ed5xBmss,
    Ed1xDmss, Ed2xDmss, Ed3xDmss, Ed4xDmss, Ed5xDmss,
    EVss, ERss,
    EBMLAGss, EDMLAGss,
    Exini, Eyini, ELini, EBini, EDini, EVini, ERini, EL1ini, Ex1ini, ExBini,
    ExDini, EyBini, EyNini, EVcini, ERcini,
    ExBmini, Ed1xBmini, Ed2xBmini, Ed3xBmini, Ed4xBmini, Ed5xBmini,
    ExDmini, Ed1xDmini, Ed2xDmini, Ed3xDmini, Ed4xDmini, Ed5xDmini,
    EREQBM, ETEQBM,
    EBMLAG, EDMLAG,
    EMB, EMMT1, EMMT, EMD,
    EbetaB, EbetaT1, EbetaT, EbetaD,
    EFLOOD, ENTRAYS,
    EL_1FD, EL_1RFLX, EL_FDTIN, EL_RFLXT10, EL_RFLXTIN,
    ExBerr, ExDerr,
    EBCNTRLIS, EDCNTRLIS, EBCNTRLU, EDCNTRLU,
    ESLLLv, ESLLUv, ESLLLr, ESLLUr, ESLULv, ESLUUv, ESLULr, ESLUUr,
    EZLLv, EZLUv, EZLLr, EZLUr, EZULv, EZUUv, EZULr, EZUUr,
    ECUTIL, ECTOTAL,
    EOBJ
    / ;

*=============================================================================
```