DUAL MODIFIER ADAPTATION METHODOLOGY

DUAL MODIFIER ADAPTATION METHODOLOGY

For the On-line Optimization of Uncertain Processes

by

Eric Rodger, B.Sc (Eng)

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfillment of the Requirements

for the Degree

Master of Engineering

McMaster University

MASTER OF ENGINEERING (2010) McMaster University

(Chemical Engineering) Hamilton, Ontario, Canada

TITLE:	Dual Modifier Adaptation Methodology
	For the On-line Optimization of Uncertain Processes
AUTHOR:	Eric Rodger, B.Sc(Eng)
	(University of Waterloo, Canada)
SUPERVISOR:	Dr. Benoît Chachuat
NUMBER OF PAGES:	xvi, 194

ABSTRACT

The current industry standard in real-time optimization (RTO) is the two-step method. In this approach, mismatch between the plant and process model is compensated for by continuously updating a subset of the parameters in the process model. It is suitably resistant to measurement noise, however it is not guaranteed to move toward the plant optimum if structural plant-model mismatch exists. Due to this deficiency, a number of alternative methods have been developed over the years, including ISOPE and modifier adaptation. These methods, however, utilize plant derivative information, which must be estimated because a precise plant model is typically not known in practice. This makes these methods particularly susceptible to measurement noise. Therefore, in this thesis, the development of an RTO technology which is both optimum seeking and resistant to measurement noise is considered.

This research can be separated into two parts. In the first phase, the current state-of-the-art modifier adaptation algorithm is modified by employing Broyden's method to estimate the plant output derivatives. A pair of deficiencies of Broyden's method are then detailed, and a modification to the algorithm, designed to mitigate these deficiencies, is proposed. This consists of the inclusion of additional constraints in the model-based optimization problem, designed to limit both offset and variance in the Broyden derivative estimates. Since the new algorithm possesses two distinct goals, optimality and the accuracy of the Broyden estimates, it is referred to as dual modifier adaptation.

In the second phase of this research, the design of dual modifier adaptation systems is considered. The design methodology is built around the design cost criterion, a metric which had previously been developed for the two-step approach of RTO. The calculation procedure for the metric is adapted in this research in order to address dual modifier adaptation systems. In addition, an approach designed to compute the constraint back-off necessary to ensure a certain level of feasibility is developed.

The concepts discussed in both the first and second phases of the research are illustrated using the Williams-Otto Reactor case study. This is a benchmark problem that has been used in the RTO literature for many years. A more involved case study, a propane furnace, is introduced in the last main chapter of this thesis. Both the performance of the dual modifier adaptation algorithm itself and the design of dual modifier adaptation systems are discussed for this case study.

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisor, Dr. B. Chachuat, for all of his advice, guidance and encouragement throughout the development of this research.

I would also like to thank my examining committee members, Dr. C.L.E. Swartz and Dr. T.E. Marlin, for their insight into how to improve this thesis.

Special thanks to Gillian for her love and patience as I completed this work as well as to my family for all their support over the years.

Finally, I would like to thank the McMaster Advanced Control Consortium and NSERC for their generous financial support.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Real-Time Optimization (RTO), if properly utilized, can be a very effective tool in the chemical process industry. The ability to adjust controller set-points based on online process conditions can provide great financial benefit [Marlin and Hrymak [1997], Cutler and Perry [1983]]. It is only effective, however, if the RTO system is able to correctly identify the optimal, or at the very least a good operating point for the current state of the plant.

1.1 Motivation and Thesis Objectives

The RTO task would be very easy if an accurate and complete model of the plant were available. This model could simply be optimized, using any of a number of known techniques, to arrive at the best possible controller set-points for implementation. The optimization would then only have to be carried out periodically, to check if the optimal operating point had moved due to process disturbances. Unfortunately, a perfect model of the plant is never available in practice. This means that precise plant derivative information cannot be obtained, making the direct application of traditional derivative-based NLP solution techniques to the plant optimization problem impossible.

The information that is generally available to the RTO system from the plant is a set of measurements, taken from sensors located throughout the process. RTO systems make use of these measurements to update the set-points, as they can contain important information about fundamental changes in the state of the process. Note that the measurements also usually contain some level of measurement noise which should be rejected.

Model-based RTO systems are a common type of RTO technology. They utilize a process model which is generally a simplification of the plant. The model is then altered in a clever way in order to mimic the input-output behaviour of the plant as closely as possible. One method of accomplishing this is the two-step approach, [Chen and Joseph, 1987], which alters one or more of the parameters in the process model based on information garnered from the plant. This approach, while fairly adept at rejecting measurement noise, will not always be able to find the optimal set-points if the process model is fundamentally different from the actual plant (structural plant-model mismatch) [Forbes *et al.*, 1994]. This is almost always the case in practice.

In response to this drawback, a number of other RTO approaches have been developed, including modifier adaptation [Marchetti *et al.*, 2009], which is the focus of this report. Unlike the two-step approach, this technology utilizes estimated plant gradient information as well as the plant measurements to update the process model. Using the gradient information in an intelligent way can guarantee that if the scheme converges, it will converge to a KKT point of the plant (in the absence of measurement noise). Therefore, in certain situations, it is much more likely to identify the plant optimum than the two-step approach is. The difficulty here is that the plant output gradient must be estimated. Since no closed-form representation of the plant is available for differentiation, an estimation procedure must be selected. It is essential that this procedure is both accurate and adequately resistant to measurement noise.

The objective of this thesis is, using modifier adaptation as a conceptual basis, to develop an RTO technology that is both optimum seeking and resistant to measurement noise. Both on-line execution aspects and the off-line design of the new RTO method are explored.

1.2 Real-Time Optimization

The Real-Time Optimization (RTO) system does not function in isolation. It is a part of a set of inter-connected subsystems that make up a multi-level control structure. This hierarchy consists of all the core elements of the Process Systems Engineering (PSE) field. It is shown as Figure 1.1.

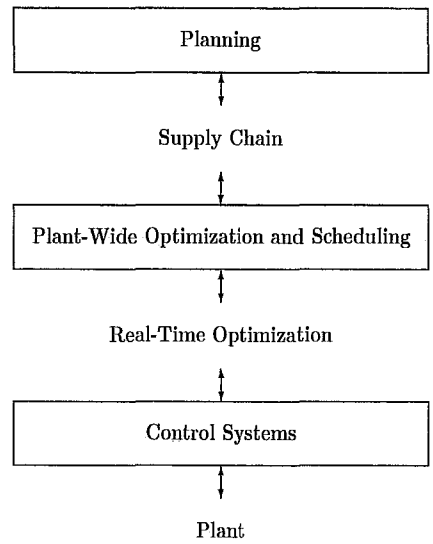

Figure 1.1: PSE hierarchy (adapted from: Seborg *et al.* [2004])

Upon examination of the multilevel control structure illustrated in Figure 1.1, the RTO system lies between the plant-wide optimization and scheduling layer and the process control layer. Therefore, in short, it takes information from the scheduling layer, makes a set of computations and sends the resulting information to the process control layer. This is consistent with the goal of the RTO system described in Section 1.1.

The basis for comparing potential set-points is generally some sort of economic criterion. This criterion often has to do with decreasing production costs or improving product quality. The computed set-points can vary between RTO iterations based on new information obtained from the plant. The iteration frequency depends on the process in question. Generally, the main requirement is that this frequency must be much longer than the closed-loop

process dynamics. If this is not possible then some sort of dynamic RTO approach must be considered [Kadam *et al.* [2007], Srinivasan and Bonvin [2007]].

The popularity of RTO technologies has greatly increased over the years [Forbes and Marlin, 1996]. This is due to a couple of key factors. The first is a general increase in the computing power available. This change has allowed the implementation of RTO on more complex systems than before, and has also allowed the RTO system to be executed more frequently in some cases, increasing its effectiveness. The second factor is the increasing competitiveness of the global marketplace. More than ever before, many companies are competing to sell very similar products. This makes saving every possible dollar during the production process more vital than ever. RTO is uniquely positioned to do this, given its goal of optimizing an economic objective. This has made RTO particularly popular in the petrochemical industry and the commodities sector, where it has been widely used [Bailey *et al.* [1993], Krishnan *et al.* [1992a]].

The scope of RTO systems has been the topic of debate over the years. One option is using the RTO system to address the operation of the entire plant at once [Bailey *et al.*, 1993]. The alternative is to adopt a distributed approach which tries to solve individual RTO problems on each unit in a plant [Darby and White, 1988]. For example, if a plant consists of a reactor followed by a heat exchanger and a distillation column, three separate RTO problems would be solved, one for each unit. A higher-level system may then be tasked with coordinating the actions of the three separate RTO implementations.

In Darby and White [1988], the distributed approach is supported. The greatest advantage of the distributed approach is the ease of running the RTO procedure on each individual unit. For instance, if one unit went offline as a result of a fault or for maintenance, the active units of the distributed system could still function as normal. The different RTO systems in the distributed framework could also be executed at different frequencies, depending on the specific units that make up the process in question [Darby and White, 1988]. Additionally, smaller models are also easier to maintain and in general computationally easier to solve than one large model.

The plant-wide RTO strategy is supported by Bailey *et al.* [1993] and Marlin and Hrymak [1997], among others. While it clearly does result in a larger, more complex model, with the power of computers constantly increasing, larger models are becoming less difficult to deal with. If the entire plant is not considered at once, the optimal set-points may not be identified due to the individual models not completely representing the interactions between the different process units [Zhang and Forbes, 2000]. Also, no coordinator needs to be designed to exchange information between the subsystems. In this work, although none of the examples are the size of a complete plant, this is assumed to be the strategy of choice. Therefore, the design of a coordinator is not considered.

The potential economic benefit of an RTO system does not come without a set of drawbacks and limitations. For instance, the RTO system is limited by the quality of the information it receives from the plant. If the measurements are corrupted significantly, the RTO system will not identify a good operating point. In fact, the process performance at this sub-optimal operating point may be worse than the performance had there been no RTO implementation at all.

The RTO system is also limited by the control system tasked with implementing the set-points it provides. The slower the control system is in driving the process to the new set-points, the smaller the benefit from the RTO system. Detecting that the controllers have pushed the process to a new steady-state can also be a problem in practice. Another limitation is the reactionary nature of the RTO system. It cannot predict process disturbances ahead of time and therefore there will always be a time lag between the disturbance and the appropriate response of the RTO system.

1.3 Approaches to Real-Time Optimization

Model-based RTO approaches were very briefly mentioned in Section 1.1. The two-step approach is probably the most well-known and widely used of these methodologies. It consists of updating a subset of the model parameters so that the model represents actual plant

input-output behaviour as closely as possible [Chen and Joseph, 1987]. There is another subclass of model-based methods however, which instead of updating a set of parameters directly in the model, adds additional parameters to the model to aid in accurately matching input-output data. These methods were developed with the aim of alleviating a major deficiency of the two-step approach. This deficiency is the inability of the algorithm to guarantee convergence (under noiseless conditions) to the optimal operating point of the plant if the process model is fundamentally different from the plant (structural plant-model mismatch).

The ISOPE (Integrated System Optimization and Parameter Estimation) method carries out the same parameter estimation step as the two-step approach, but also introduces an additional parameter into the cost function [Roberts, 1979]. This parameter matches the gradient of the plant cost function with that of the model. Constraint bias updating adds an extra parameter to each of the constraint functions, attempting to match the constraints of the model to those of the plant [Forbes and Marlin, 1994]. In Gao and Engell [2005], the constraints are updated not only with a zeroth order term (as in constraint bias updating), but also a first order term, which takes into account gradient information. Finally, modifier adaptation combines the ideas of both of the preceding methods, by introducing a set of so-called modifiers which alter both the values and gradients of the cost and constraint functions [Marchetti *et al.*, 2009]. More details and discussion about all of these methods can be found in Chapter 2.

Besides model-based RTO methods, there is another class of methods, sometimes referred to as *model-free* methods, which do not employ a process model at all [Chachuat *et al.*, 2009]. These methods use measurements from the plant, along with a set of heuristics, to choose the best operating point for the plant. One example of these methods is self-optimizing control [Skogestad, 2000]. In this method, constant values are chosen for a set of controlled variables and the system is manipulated by changing the set-points in order to try to keep these variables at their reference values. The choice of the controlled variables and corresponding references is important here. Choosing variables which do not vary significantly in the presence of uncertainty is very important. Other important criteria are

given in Skogestad [2000].

NCO tracking is another effective model-free method [Francois *et al.*, 2005]. In this procedure, the inputs are adjusted so as to try to satisfy the necessary conditions of optimality for the plant. Therefore, this method can be seen as a self-optimizing method where the controlled variables are the active constraints and reduced gradient (KKT quantities), with both of these having a reference of zero. Plant measurements, and sometimes estimated plant output derivatives, are used in this procedure. Since derivative estimates are sometimes used, this approach can be sensitive to measurement noise.

The inputs for NCO tracking are generally divided into two subsets, inputs that help satisfy the active constraints, and sensitivity seeking inputs, which are used to optimize the economic function. A bi-level formulation has been proposed to separate the relatively easy task of controlling the constraints and the more difficult task of using the sensitivity-seeking inputs to achieve optimality [Francois *et al.*, 2005]. A good comparison of model-based and model-free RTO approaches is given in Chachuat *et al.* [2009].

1.4 Thesis Overview

This work focuses on model-based RTO technologies. Within this sub-group, it focuses primarily on the modifier adaptation approach [Marchetti *et al.*, 2009]. Chapter 2 begins with a review of the basic components of a model-based RTO system, followed by a description of several previously developed technologies. These include the classic two-step approach, ISOPE, constraint bias updating and ideal modifier adaptation. The background section ends with a comparison of some of these methods using the Williams-Otto Reactor benchmark problem. This benchmark RTO problem will be used in simulations throughout this thesis.

Chapter 3 begins with an explanation of the choice of Broyden's method for implementation with modifier adaptation. Next, a convergence analysis is carried out for the modifier adaptation algorithm employing Broyden's method. The performance of this algorithm is

then evaluated using the Williams-Otto Reactor test case, and modifications are suggested to improve this performance. Specifically, the inclusion of additional constraints in the model-based optimization problem is suggested, creating an algorithm with two distinct goals: finding the optimal operating point and generating accurate Broyden estimates. This new algorithm is referred to as dual modifier adaptation.

Chapter 4 addresses the issue of how to design the dual modifier adaptation system to achieve the best possible online performance. This design is done offline due to the computational burden of redesigning the system online after every iteration. To form the basis of this procedure, the design cost criterion [Forbes and Marlin, 1996] is adapted so it can address both unconstrained and constrained dual modifier adaptation systems. The design of an appropriate back-off for constrained problems is also investigated. These concepts are illustrated using the Williams-Otto Reactor test case at the end of the chapter.

In Chapter 5, the performance of dual modifier adaptation and the corresponding design procedure are evaluated using a more complex case study. This case study involves the real-time optimization of the operation of a propane pyrolysis reactor. First, the objective function, constraints and process model are discussed in detail. Next, several aspects of the dual modifier adaptation algorithm itself are discussed. These include the effect of various tuning parameters on the performance of the algorithm. The design of modifier adaptation for this case study is also explored. The computation of the design cost metric is discussed in detail here.

Conclusions are then drawn and future recommendations are made. These recommendations include the potential combination of the two-step approach and modifier adaptation with the goal of leveraging the unique advantages of each technology. Recommendations on how to improve the design procedure are also made.

Chapter 2

Background on RTO Technologies

The goal of this chapter is to give the reader an overview of the RTO research that has been completed in the last 30 years or so. An overview of the main components of a generic RTO system will be presented first, followed by a discussion of some of the specific approaches that have been developed. These include the two-step method, ISOPE, constraint bias updating and ideal modifier adaptation. Some of these methods necessitate the approximation of the plant output gradient and therefore potential plant output gradient estimation methods are discussed as well. Next, the Williams-Otto Reactor process, a standard RTO test case, is introduced. It will be used to illustrate concepts discussed throughout this report. The section finishes with a comparison of the two-step approach and ideal modifier adaptation.

2.1 Components of an RTO System

Despite the fact that there are many different algorithms used for real-time optimization, the basic structure of any one model-based RTO system is fairly standard. All the separate components, and the basic flow of information, are shown in Figure 2.1. Note that only some of the components in Figure 2.1 are necessary for a system to properly operate, therefore some may not appear in any one particular system.

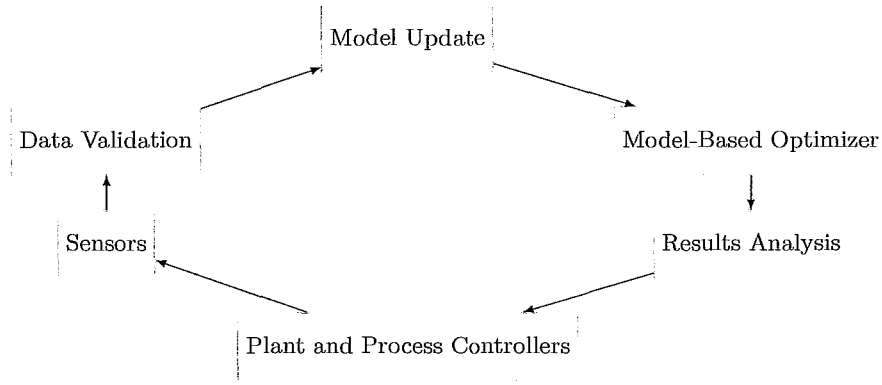

Figure 2.1: General RTO diagram (adapted from Yip and Marlin [2002])

An RTO iteration begins when the process sensors take measurements which reflect the state of the plant at the current time. At this time the plant must be at steady state. A check is often performed to ensure this, especially in the case of relatively slow processes. Generally there is also some form of validation procedure incorporated at this level to check if the measurements are plausible. This is sometimes called data reconciliation or gross error detection, and may involve comparing the measurements to the existing process model. A good discussion of this step, with references for steady-state validation and gross error detection, can be found in Marlin and Hrymak [1997].

From a design perspective, the decision of which measurements to take and where physically in the process to take them is not a trivial one. Considerable work was done in this area by Krishnan *et al.* [1992b], who applied a series of statistical tests to determine the set of measurements which would work best with the two-step approach, given the chosen set of adjustable parameters. Fraleigh *et al.* [2003] later explored the economics of this decision, taking into account the fact that the sensor and model updating systems cannot be viewed in isolation, and rather need to be analyzed as a part of the whole closed-loop RTO system.

The measurements are then sent to a model updater. The purpose of this step is to use the measurement information to improve the model of the process. This can be done in a variety of ways. Either a set of parameters directly present in the process model, or one or more "artificial" parameters, introduced specifically for the use of the RTO algorithm, are

updated. The specifics of this procedure will be left to the individual sections on each RTO technology.

Sometimes a test is performed at the end of the model update step to check the conditioning of the covariance matrix of the updated parameters. This test was originally proposed in Miletic and Marlin [1998b] for the two-step approach. If the test is failed, some parameters may need to be fixed for the current iteration in order to improve the conditioning of the covariance matrix. Alternatively, using multiple data sets for model updating can also reduce the condition number of the parameter covariance matrix [Yip and Marlin, 2002]. The required data can be collected by performing experiments on the plant, which involve slightly changing the operating point and taking a new set of measurements at the new point. Methods for designing such experiments can be found in Yip and Marlin [2003]. Although all of these results are specific to the two-step approach, the ideas could potentially be extended to other RTO approaches as well.

The updated model is then used by the model-based optimizer. The optimizer will determine the set-points which minimize a particular cost function, satisfying both the process model as well as any process constraints (including bounds on the set-point). Since the model itself is usually non-linear, the entire problem is generally an NLP. There are many NLP solution methods that can be used for this purpose. In this work, the sequential quadratic programming (SQP) solver included in the MATLAB Optimization Toolbox, *fmincon*, was used to perform the optimization. When passing the problem to the optimizer, variable scaling can be very important, especially for large or difficult to solve problems. Providing analytical derivatives and a reasonable starting point to the optimizer can also increase the chances of finding a good solution. A discussion of this step can be found in Marlin and Hrymak [1997].

After the new set-point has been computed by the optimizer, the results analysis subsystem decides if it should be implemented on the plant. This check attempts to distinguish between common cause variation (i.e. measurement noise), which in general should not motivate an operating point change, and process disturbances (special cause variation),

which are a valid reason for changing the operating point. A statistical test was proposed in Miletic and Marlin [1998a] for this purpose. At this stage, operators or other plant personnel may also be required to approve the recommendations of the optimizer.

The last step involves the process control system implementing the prescribed operating point change. This should be achievable as long as the RTO system has provided a suitable operating point. For RTO systems with frequent execution times, the process dynamics can sometimes act as a bottleneck, preventing the next RTO execution from occurring until the current set-point is reached and the required measurements are taken [Zhang and Forbes, 2000]. Specific analysis of the control system (i.e. individual PIDs or MPC) is beyond the scope of this report. An investigation of the integration of the RTO and control systems from an economic viewpoint can be found in Contreras-Dordelly and Marlin [2000].

2.2 The Classic Two-Step Approach

The classic two-step approach to RTO has been well established for more than 20 years and is widely used in industry today. A description of the methodology can be found in Chen and Joseph [1987]. This algorithm has been successfully applied to many industrial processes. Examples include Bailey *et al.* [1993] and Krishnan *et al.* [1992a]. Additional implementations are listed in Marlin and Hrymak [1997]. The optimization problem that the RTO system is trying to solve for the plant is the following:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \phi(\mathbf{u}, \mathbf{y}^p) \\ \text{s.t.} \quad & \mathbf{y}^p = \mathbf{F}(\mathbf{u}) \\ & \mathbf{g}\left(\mathbf{u}, \mathbf{y}^p\right) \leq 0 \\ & \mathbf{u}^{min} \leq \mathbf{u} \leq \mathbf{u}^{max} \end{aligned} \tag{2.1}$$

where ϕ represents the objective function, $\mathbf{u}$ denotes the inputs, $\mathbf{y}^p$ are the plant outputs (measurements), $\mathbf{F}$ represents a set of explicit plant input-output relations (generally unknown), $\mathbf{g}$ denotes the output-dependent constraints and $\mathbf{u}^{min}$ and $\mathbf{u}^{max}$ are the bounds on the inputs.

In the design phase of the two-step approach, the model parameters must be separated into two subsets. The first subset should hold all the parameters that will remain fixed throughout the RTO process. The second subset should contain the parameters that are to be updated by the parameter estimation problem, in response to plant process changes or disturbances. In general, these parameters should be observable from the plant measurements taken [Stanley and Mah [1981], Krishnan *et al.* [1992b]], represent actual process variations, and aid in moving towards the plant optimum [Marlin and Hrymak, 1997]. There is a considerable amount of literature surrounding how to make this division of the set of model parameters [Forbes and Marlin [1996], Krishnan *et al.* [1992b]].

After an appropriate subset of the parameters is chosen for updating, and the appropriate measurements are taken, an optimization problem is solved that attempts to match the plant and model outputs as closely as possible. This corresponds to the model updating step described in the previous section. Through this optimization problem, values are assigned to the set of adjustable parameters $\boldsymbol{\beta}$:

$$\begin{aligned} \boldsymbol{\beta}_{k+1} \in \arg\min_{\boldsymbol{\beta}} \quad & \left(\mathbf{y}^p_{k+1} - \mathbf{y}^m_{k+1}\right)^2 \\ \text{s.t.} \quad & \mathbf{y}^m_{k+1} = \mathbf{f}(\mathbf{u}_{k+1}, \boldsymbol{\beta}) \end{aligned} \tag{2.2}$$

where $\mathbf{y}^m$ represents the model outputs and $\mathbf{f}$ is a set of explicit functions defining the model outputs. Note that in practice an implicit form of $\mathbf{f}$ is generally known, however it is assumed here that the implicit equations can be solved for the outputs using a solver for systems of non-linear equations (e.g. *fsolve* in MATLAB). This is only one possible update strategy, another common one involves weighting the output differences by the inverse of the covariance matrix of the plant measurements [Yip and Marlin, 2002].

After these new parameter values are obtained, the general model-based RTO problem is solved, striving to minimize the objective function (ϕ) by changing the process inputs,

$$\begin{aligned} \mathbf{u}_{k+1} \in \arg\min_{\mathbf{u}} \quad & \phi(\mathbf{u}, \mathbf{y}^m) \\ \text{s.t.} \quad & \mathbf{y}^m = \mathbf{f}(\mathbf{u}, \boldsymbol{\beta}_k) \\ & \mathbf{g}\left(\mathbf{u}, \mathbf{y}^m\right) \leq 0 \\ & \mathbf{u}^{min} \leq \mathbf{u} \leq \mathbf{u}^{max} \end{aligned} \tag{2.3}$$

Note that all the developments in this work will be done assuming the objective function for the RTO system represents a quantity to be minimized. One common example is the cost of production. For clarity, a diagram of this procedure is shown as Figure 2.2. In this figure, SS stands for steady-state.

One important drawback of the two-step approach is the fact that it may converge to a sub-optimal point if structural plant-model mismatch exists. Structural plant-model mismatch occurs when the model being used does not resemble the actual plant process in some fundamental way. For instance a side reaction that has not been well documented may be left out of the process model. This phenomenon occurs frequently in large-scale, complex industrial systems.

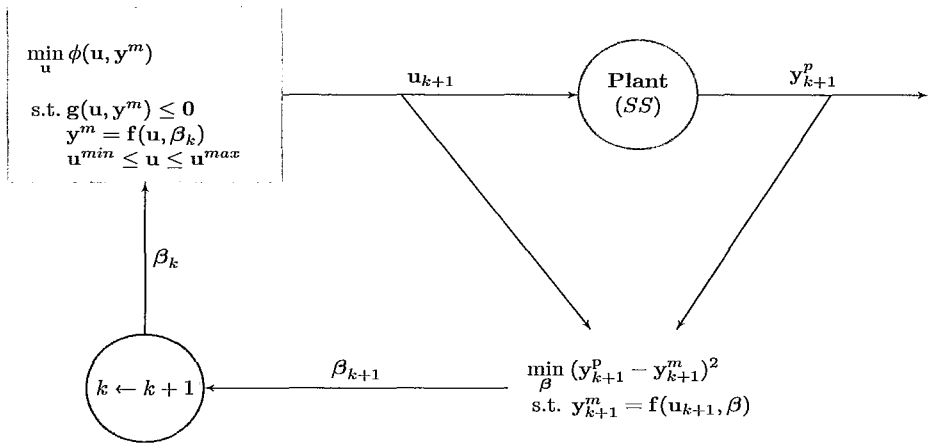

Figure 2.2: Two-step method diagram

Structural plant-model mismatch has been discussed extensively in the RTO literature. The difficulty in using a simple model to approximate a much more complex model in an optimization procedure (called the inside-out method or surrogate model optimization in the literature), is discussed in Biegler *et al.* [1985]. In this procedure, parameters in the simplified model are periodically updated using values obtained from the rigorous model. Note that this is essentially the same situation existing in RTO, with the more complex model being the plant and the process model being the simpler model. Biegler *et al.* [1985] show that unless the optimum of the rigorous model is also a KKT point of the simple model, it will not be found through the parameter update procedure. Since the KKT conditions

involve gradient information, unless the cost and constraint gradients match at the rigorous model optimum, this condition will not be satisfied.

The concept of model adequacy, [Forbes *et al.*, 1994], was developed for the two-step approach to deal with this problem. It is a tool that allows the RTO designer to check if his/her process model will be capable of recognizing the plant optimum. Instead of trying to verify first order KKT condition details, it checks conditions to do with the reduced gradient and Hessian of the approximate model at the plant optimum.

This discussion underscores the importance of the model selection task in designing an RTO system. Unfortunately, an adequate model in the sense of Forbes *et al.* [1994] may not be available in the case of large-scale, complex industrial systems. This has motivated the development of alternative RTO paradigms, such and the Integrated System Optimization and Parameter Estimation (ISOPE) method and Modifier Adaptation (MA), which will be discussed in Sections 2.4 and 2.5 respectively.

2.3 Constraint Bias Updating

The constraint bias update approach (sometimes also called constraint adaptation) is another common model-based RTO technology. Detailed information can be found regarding the approach in Forbes and Marlin [1994] and Chachuat *et al.* [2008]. While the ISOPE and modifier adaptation approaches (discussed in the next two sections) are greatly concerned with optimality, the constraint bias update approach is more concerned with feasibility. Therefore it is especially useful when it is expected that a large number of the degrees of freedom of the optimization problem will be taken up by active constraints [Forbes and Marlin, 1994].

Instead of solving the parameter estimation problem each iteration (like in the two-step approach), in the constraint bias update approach a new parameter is introduced into the RTO algorithm to be updated every iteration. This parameter is computed as follows

[Chachuat *et al.*, 2008]:

$$\epsilon^b_{k+1} = (\mathbf{I} - \mathbf{K}_g)\,\epsilon^b_k - \mathbf{K}_g\left[\mathbf{g}\left(\mathbf{u}_{k+1}, \mathbf{y}^p_{k+1}\right) - \mathbf{g}\left(\mathbf{u}_{k+1}, \mathbf{y}^m_{k+1},\right)\right] \tag{2.4}$$

where ϵ^b is the constraint bias parameter and $\mathbf{K}_g$ contains filter parameters for each constraint which generally range between 0 and 1. The higher the filter parameters, the more aggressive the correction, but also the more likely the iterates are to diverge. This new parameter, ϵ^b, is then incorporated into the model-based optimization problem as follows:

$$\begin{aligned} \mathbf{u}_{k+1} \in \arg\min_{\mathbf{u}} \quad & \phi(\mathbf{u}, \mathbf{y}^m) \\ \text{s.t.} \quad & \mathbf{y} = \mathbf{f}(\mathbf{u}, \boldsymbol{\beta}) \\ & \mathbf{g}\left(\mathbf{u}, \mathbf{y}^m\right) + \epsilon^b_k \leq \mathbf{0} \\ & \mathbf{u}^{min} \leq \mathbf{u} \leq \mathbf{u}^{max} \end{aligned} \tag{2.5}$$

One of the core advantages of this approach is that upon convergence, it is easily proved that its iterates will arrive at a feasible point [Chachuat *et al.*, 2008]. This is valuable, especially for those industrial processes in which feasibility is crucial. It is also easier to run from a computational point of view than methods that require the solution of a parameter estimation problem. In addition, it is less sensitive to measurement noise than the methods that are presented next (ISOPE and modifier adaptation) which require the approximation of the plant output gradient. The trade-off is the fact that this method makes no effort to match the gradients of the model cost function or constraints to those of the plant (the gradients are taken as those of the nominal process model), therefore it may suffer in terms of optimality.

2.4 The ISOPE Method

The ISOPE method is detailed in Roberts [1979]. There have since been a variety of modifications made to improve the original formulation. Here the original method will be discussed, followed by a look at some of the modifications that are most relevant to this work. For a more complete review of early ISOPE work consult Roberts [1995].

The major difference between ISOPE and the traditional two-step approach is the inclusion of an extra parameter in the objective function. This parameter attempts to match the gradient of the objective function of the model-based problem with the gradient of the objective function of the plant. This parameter is updated every iteration in the following way [Roberts, 1979]:

$$\psi_{k+1} = \frac{\partial \phi}{\partial \mathbf{y}} \left(\mathbf{u}_{k+1}, \mathbf{y}^m_{k+1}\right) \left(\frac{\partial \mathbf{F}}{\partial \mathbf{u}} \left(\mathbf{u}_{k+1}, \mathbf{y}^p_{k+1}\right) - \frac{\partial \mathbf{f}}{\partial \mathbf{u}} \left(\mathbf{u}_{k+1}, \mathbf{y}^m_{k+1}, \boldsymbol{\beta}_{k+1}\right) \right) \tag{2.6}$$

where ψ is the value of the cost gradient parameter. Note that this update procedure requires knowledge of the plant output gradient $(\frac{\partial \mathbf{F}}{\partial \mathbf{u}})$. As previously discussed, since no true plant model is ever available in RTO, this term must be approximated. A discussion of plant gradient estimation procedures is deferred until Section 2.6.

As in the two-step approach, a parameter estimation problem must be solved to update the adjustable model parameters, $\boldsymbol{\beta}$, for use in both the calculation of the additional parameter (2.6) and the model-based optimization problem (2.8). The parameter estimation problem here is essentially the same as Equation 2.2, however it must be solved using an additional constraint:

$$\mathbf{y}^p_{k+1} = \mathbf{y}^m_{k+1} \tag{2.7}$$

The rationale behind the addition of this constraint is to ensure that $\frac{\partial \phi}{\partial \mathbf{y}}$ in the additional parameter update step (2.6) is evaluated using the plant outputs $(\mathbf{y}^p)$. Depending on the degree of mismatch between the plant and the approximate model, this may not be achievable in practice. This is a deficiency of the original ISOPE procedure which was later addressed in the literature [Tatjewski, 2002].

The updated value of the cost gradient parameter (ψ) is then sent to the optimization problem which is solved as follows [Roberts, 1979]:

$$\begin{aligned} \mathbf{u}^* \in \arg\min_{\mathbf{u}} \quad & \phi(\mathbf{u}, \mathbf{y}^m) + \psi_k^T \mathbf{u} \\ \text{s.t.} \quad & \mathbf{y}^m = \mathbf{f}(\mathbf{u}, \boldsymbol{\beta}_k) \\ & \mathbf{g}\left(\mathbf{u}, \mathbf{y}^m\right) \leq 0 \\ & \mathbf{u}^{min} \leq \mathbf{u} \leq \mathbf{u}^{max} \end{aligned} \tag{2.8}$$

where $\mathbf{u}^*$ denotes the optimal inputs. Note that the only alteration from Problem 2.3 is the inclusion of ψ_k in the cost function. A filter is sometimes placed on the computed inputs in order to improve the stability of the algorithm [Roberts and Williams, 1981]:

$$\mathbf{u}_{k+1} = (\mathbf{I} - \mathbf{K})\,\mathbf{u}_k + \mathbf{K}_u \mathbf{u}^* \tag{2.9}$$

where $\mathbf{K}_u$ is a diagonal matrix of filter parameters with values typically ranging between 0 and 1.

The original ISOPE procedure has several shortcomings, which have been addressed in the literature throughout the last thirty years. For instance, constraints were not addressed in the original formulation. Specifically, if process constraints are active at the plant optimum, correcting the cost function gradient of the model is not enough to match the KKT conditions of the plant.

An attempt to address constraint handling is made in Tatjewski *et al.* [2001], where the use of a constraint follow-up controller (CFC) is proposed. This controller ensures that certain key constraints on the outputs are satisfied. This essentially involves splitting the outputs into two categories, separating those outputs whose constraints are active at the plant optimum from the others. A subset of the inputs are then assigned to act as manipulated variables in the CFC. The main disadvantage of this method is that the active set at the plant optimum must be known before-hand, and the design of the RTO system must be based on this active set. Therefore, if the active set were to change online, the performance of the RTO system would likely suffer.

Another possible fix for the constraint handling problem was presented in Gao and Engell [2005]. Using plant measurements and an approximation of the plant output gradient, the following is written:

$$\begin{aligned} \mathbf{g}_k(\mathbf{u}) \;=\; & \mathbf{g}(\mathbf{u}, \mathbf{y}^m) + \left[\mathbf{g}(\mathbf{u}_k, \mathbf{y}_k^p) - \mathbf{g}(\mathbf{u}_k, \mathbf{y}_k^m)\right] + \\ & \frac{\partial \mathbf{g}}{\partial \mathbf{y}}(\mathbf{u}_k, \mathbf{y}_k^m)\left(\frac{\partial \mathbf{F}}{\partial \mathbf{u}}(\mathbf{u}_k, \mathbf{y}_k^p) - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{u}_k, \mathbf{y}_k^m, \beta_k)\right)(\mathbf{u} - \mathbf{u}_k) \end{aligned} \tag{2.10}$$

where the intention is that the modified constraint be comprised of the value of the constraint using the model outputs, with 0^{th} and 1^{st} order corrections added to account for

plant-model mismatch. The modified constraint is then implemented in the optimization problem (2.8) as follows:

$$\mathbf{g}_k(\mathbf{u}) \leq 0 \tag{2.11}$$

Finally, a modification is proposed in Tatjewski [2002], which eliminates the need to solve the parameter estimation problem as a part of the ISOPE algorithm. This is done by introducing an extra parameter into the calculation procedure for ψ:

$$\psi_{k+1} = \frac{\partial \phi}{\partial \mathbf{y}}\left(\mathbf{u}_{k+1}, \mathbf{y}^m_{k+1} + \mathbf{a}_{k+1}\right)\left(\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \mathbf{y}^p_{k+1}\right) - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \mathbf{y}^m_{k+1}, \boldsymbol{\beta}_k\right)\right) \tag{2.12}$$

where $\mathbf{a}$ is referred to as the model shift parameter. This parameter $\mathbf{a}$ is defined as follows:

$$\mathbf{a}_{k+1} = \mathbf{y}^p_{k+1} - \mathbf{y}^m_{k+1} \tag{2.13}$$

Examining Equations 2.12 and 2.13, it is clear that the purpose of the model shift parameter is to make it so that the cost function output derivative, $\frac{\partial \phi}{\partial \mathbf{y}}$, is essentially evaluated at $\mathbf{y}^p$. Therefore, this negates the need for the parameter estimation problem to enforce the plant/model output matching criteria of Equation 2.7.

This concludes the discussion on the ISOPE method and extensions. Although this method is not studied extensively in the remainder of this thesis, the importance of the developments of the previous section as a precursor to modifier adaptation, the central focus of this work, will become evident in the sections to follow.

2.5 Ideal Modifier Adaptation

Modifier adaptation (MA) was first presented in Chachuat *et al.* [2009] and Marchetti *et al.* [2009]. It utilizes many of the concepts presented in the previous two sections to arrive at a clear, concise RTO methodology. Specifically it introduces a set of extra parameters (referred to as *modifiers* here) into the model-based optimization problem which directly alter both the cost and constraint functions.

The cost function is altered by the addition of the following cost gradient modifiers, $\boldsymbol{\lambda}^{\phi}$ [Marchetti *et al.*, 2009]:

$$\boldsymbol{\lambda}^{\phi} = \left[\frac{d\phi}{d\mathbf{u}}(\mathbf{u}, \mathbf{F}(\mathbf{u})) - \frac{d\phi}{d\mathbf{u}}(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta}))\right]^T \tag{2.14}$$

where $\boldsymbol{\lambda}^{\phi}$ is a column vector of modifiers and the individual derivatives are computed as follows:

$$\begin{aligned} \frac{d\phi}{d\mathbf{u}}(\mathbf{u}, \mathbf{F}(\mathbf{u})) &= \frac{\partial\phi}{\partial\mathbf{u}}(\mathbf{u}, \mathbf{F}(\mathbf{u})) + \frac{\partial\phi}{\partial\mathbf{y}}(\mathbf{u}, \mathbf{F}(\mathbf{u})) \cdot \frac{\partial\mathbf{F}}{\partial\mathbf{u}}(\mathbf{u}) \\ \frac{d\phi}{d\mathbf{u}}(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta})) &= \frac{\partial\phi}{\partial\mathbf{u}}(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta})) + \frac{\partial\phi}{\partial\mathbf{y}}(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta})) \cdot \frac{\partial\mathbf{f}}{\partial\mathbf{u}}(\mathbf{u}, \boldsymbol{\beta}) \end{aligned} \tag{2.15}$$

Note that this modifier is similar but not necessarily identical to the ISOPE parameter ψ defined in Equation 2.6. Both quantities are only the same if the additional ISOPE constraint, Equation 2.7, holds. The advantage of this particular definition of the cost gradient modifier is it allows for the elimination of the model shift parameter, $\mathbf{a}$ (Equations 2.12 and 2.13). It can be eliminated here because the full cost function input derivatives for both the plant and model are computed separately.

The constraints are also altered in modifier adaptation by the introduction of the following two modifiers:

$$\begin{aligned} \epsilon^g &= \mathbf{g}(\mathbf{u}, \mathbf{F}(\mathbf{u})) - \mathbf{g}(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta})) \\ \boldsymbol{\lambda}^g &= \left[\frac{d\mathbf{g}}{d\mathbf{u}}(\mathbf{u}, \mathbf{F}(\mathbf{u})) - \frac{d\mathbf{g}}{d\mathbf{u}}(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta}))\right]^T, \ \forall i = 1, .., n_g \end{aligned} \tag{2.16}$$

where ϵ^g and $\boldsymbol{\lambda}^g$ are the constraint bias and gradient modifiers respectively. Note that the way modifier adaptation deals with constraints is similar to the method that was detailed in Gao and Engell [2005] (Equation 2.10), with the distinction that separate modifiers are declared to represent both the zeroth and first order constraint corrections. This provides the user with the option to choose not to update certain modifiers. For instance, the gradient modifiers may not be used if the plant output derivatives are too noisy. A parallel can also be drawn here with the constraint bias update approach. Examining Equation 2.16, it is apparent that the zeroth order correction modifier, ϵ^g, is also identical to the unfiltered version of the bias update parameter (ϵ^b) defined in Equation 2.4. The modifiers are then

used in the model based optimization problem in the following way [Marchetti *et al.*, 2009]:

$$\begin{aligned} \mathbf{u}_{k+1} \in \arg\min_{\mathbf{u}} \quad & \phi\left(\mathbf{u}, \mathbf{y}^m\right) + \boldsymbol{\lambda}_k^{\phi,T}\mathbf{u} \\ \text{s.t.} \quad & \mathbf{y}^m = \mathbf{f}\left(\mathbf{u}, \boldsymbol{\beta}\right) \\ & \mathbf{g}\left(\mathbf{u}, \mathbf{y}^m\right) + \boldsymbol{\epsilon}_k^g + \boldsymbol{\lambda}_k^{g,T}\left(\mathbf{u} - \mathbf{u}_k\right) \leq \mathbf{0} \\ & \mathbf{u}^{min} \leq \mathbf{u} \leq \mathbf{u}^{max} \end{aligned} \tag{2.17}$$

One of the advantages of the way that modifier adaptation is formulated is it allows for straightforward filtering of the modifiers defined in Equations 2.14 and 2.16. This filtering is similar to the filtering of the bias update parameter shown in Equation 2.4:

$$\begin{bmatrix} \boldsymbol{\epsilon}_{k+1}^g \\ \boldsymbol{\lambda}_{k+1}^{\phi} \\ \boldsymbol{\lambda}_{k+1}^{g,1} \\ \vdots \\ \boldsymbol{\lambda}_{k+1}^{g,n_g} \end{bmatrix} = \left(\mathbf{I} - \mathbf{K}_\Lambda\right) \begin{bmatrix} \boldsymbol{\epsilon}_k^g \\ \boldsymbol{\lambda}_k^{\phi} \\ \boldsymbol{\lambda}_k^{g,1} \\ \vdots \\ \boldsymbol{\lambda}_k^{g,n_g} \end{bmatrix} + \mathbf{K}_\Lambda \begin{bmatrix} \mathbf{g}\left(\mathbf{u}, \mathbf{F}(\mathbf{u})\right) - \mathbf{g}\left(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta})\right) \\ \left[\frac{d\phi}{d\mathbf{u}}\left(\mathbf{u}, \mathbf{F}(\mathbf{u})\right) - \frac{d\phi}{d\mathbf{u}}\left(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta})\right)\right]^T \\ \left[\frac{dg^1}{d\mathbf{u}}\left(\mathbf{u}, \mathbf{F}(\mathbf{u})\right) - \frac{dg^1}{d\mathbf{u}}\left(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta})\right)\right]^T \\ \vdots \\ \left[\frac{dg^{n_g}}{d\mathbf{u}}\left(\mathbf{u}, \mathbf{F}(\mathbf{u})\right) - \frac{dg^{n_g}}{d\mathbf{u}}\left(\mathbf{u}, \mathbf{f}(\mathbf{u}, \boldsymbol{\beta})\right)\right]^T \end{bmatrix} \tag{2.18}$$

where n_g is the number of constraints and $\mathbf{K}_\Lambda$ is a matrix of filter parameters. Note that in this thesis, it is assumed that $\mathbf{K}_\Lambda$ is a diagonal matrix of filter parameters, $\mathbf{k}$, which generally range from 0 to 1.

The preceding definition of the modifiers was given in part because it is the definition suggested in Marchetti *et al.* [2009] and also because it allowed for clear comparisons to be made with previous literature. However, this is not the definition that has been selected for this work. In Marchetti *et al.* [2009], an alternative scheme is given, where the modifiers are used directly to correct the model outputs ($\mathbf{y}^m$) as opposed to the cost and constraint functions. These alternate modifiers are defined in the following way:

$$\begin{bmatrix} \boldsymbol{\epsilon}_{k+1} \\ \boldsymbol{\lambda}_{k+1}^{1} \\ \vdots \\ \boldsymbol{\lambda}_{k+1}^{n_y} \end{bmatrix} = \left(\mathbf{I} - \mathbf{K}_\Lambda\right) \begin{bmatrix} \boldsymbol{\epsilon}_k \\ \boldsymbol{\lambda}_k^{1} \\ \vdots \\ \boldsymbol{\lambda}_k^{n_y} \end{bmatrix} + \mathbf{K}_\Lambda \begin{bmatrix} \mathbf{F}\left(\mathbf{u}_{k+1}\right) - \mathbf{f}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right) \\ \left[\frac{\partial F^1}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}\right) - \frac{\partial f^1}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right)\right]^T \\ \vdots \\ \left[\frac{\partial F^{n_y}}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}\right) - \frac{\partial f^{n_y}}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right)\right]^T \end{bmatrix} \tag{2.19}$$

where ϵ and λ are the output bias and gradient modifiers respectively and n_y is the number of outputs. With the alternate modifier definitions of Equation 2.19, the model-based optimization problem becomes:

$$\begin{aligned} \mathbf{u}_{k+1} \in \arg\min_{\mathbf{u}} \quad & \phi\left(\mathbf{u}, \hat{\mathbf{y}}^m\right) \\ \text{s.t.} \quad & \hat{\mathbf{y}}^m = \mathbf{f}\left(\mathbf{u}, \beta\right) + \epsilon_k + \boldsymbol{\lambda}_k^T \left(\mathbf{u} - \mathbf{u}_k\right) \\ & \mathbf{g}\left(\mathbf{u}, \hat{\mathbf{y}}^m\right) \leq \mathbf{0} \\ & \mathbf{u}^{min} \leq \mathbf{u} \leq \mathbf{u}^{max} \end{aligned} \tag{2.20}$$

where $\hat{\mathbf{y}}^m$ are the modified outputs.

After the optimization step, the new set of inputs are sent back to the update procedure (Equation 2.19) and a new set of modifiers are computed. This algorithm is illustrated in full on Figure 2.3. In this figure, $\boldsymbol{\Lambda}$ represents the full set of modifiers.

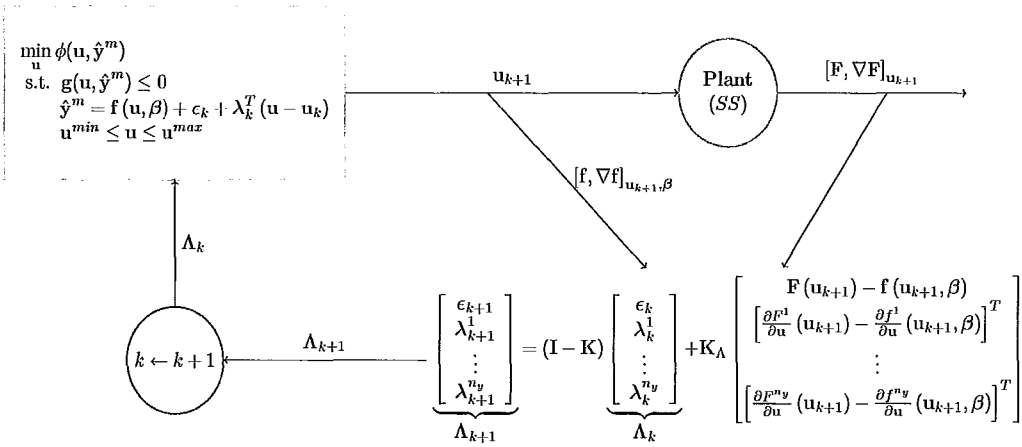

Figure 2.3: Ideal modifier adaptation diagram

One of the advantages of defining the modifiers in this manner is flexibility in which outputs are updated at each iteration. For instance, if a sensor stops working and one measurement is not available for an iteration, the modifiers pertaining to that output can be held constant while the rest of the algorithm proceeds as normal. Another example would be if one output measurement is very noisy, the decision could be made to not update all the modifiers related to it. This alternate scheme also makes it easier to combine modifier adaptation with other RTO algorithms.

One of the most beneficial properties of the modifier adaptation algorithm is that, upon convergence (in the absence of measurement noise), it will arrive at a local optimum of the plant. This is due to the fact that the KKT conditions of the model will mimic those of the plant at this point [Marchetti *et al.*, 2009]. At the converged point, $\mathbf{u}^*_\infty$, the modifiers can be computed as follows:

$$\begin{aligned} \boldsymbol{\epsilon}^*_\infty &= \mathbf{F}\left(\mathbf{u}^*_\infty\right) - \mathbf{f}\left(\mathbf{u}^*_\infty, \boldsymbol{\beta}\right) \\ \boldsymbol{\lambda}^*_\infty &= \frac{\partial \mathbf{F}}{\partial \mathbf{u}}\left(\mathbf{u}^*_\infty\right) - \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\left(\mathbf{u}^*_\infty, \boldsymbol{\beta}\right) \end{aligned} \tag{2.21}$$

Ignoring the input bound constraints (for simplicity), the KKT conditions of Problem 2.20 can be written:

$$\begin{aligned} \mathbf{g}\left(\mathbf{u}, \hat{\mathbf{y}}^m(\mathbf{u})\right) &\leq \mathbf{0} \\ \boldsymbol{\gamma}^T \mathbf{g}\left(\mathbf{u}, \hat{\mathbf{y}}^m(\mathbf{u})\right) &= \mathbf{0} \\ \boldsymbol{\gamma} &\geq \mathbf{0} \\ \frac{\partial \boldsymbol{\mathcal{L}}}{\partial \mathbf{u}} \triangleq \frac{\partial \phi}{\partial \mathbf{u}} + \frac{\partial \phi}{\partial \mathbf{y}} \frac{\partial \hat{\mathbf{y}}^m}{\partial \mathbf{u}} + \boldsymbol{\gamma}^T \left[\frac{\partial \mathbf{g}}{\partial \mathbf{u}} + \frac{\partial \mathbf{g}}{\partial \mathbf{y}} \frac{\partial \hat{\mathbf{y}}^m}{\partial \mathbf{u}} \right] &= \mathbf{0} \end{aligned} \tag{2.22}$$

where $\boldsymbol{\gamma}$ are the Lagrange multipliers for the inequality constraints $\mathbf{g}$ and $\boldsymbol{\mathcal{L}}$ is the Lagrangian. The first expression of Equation 2.21 can be re-arranged to yield: $\mathbf{F}\left(\mathbf{u}^*_\infty\right) = \boldsymbol{\epsilon}_\infty + \mathbf{f}\left(\mathbf{u}^*_\infty, \boldsymbol{\beta}\right)$. The right-hand side in this expression is also exactly what appears on the right-hand side of the expression for $\hat{\mathbf{y}}^m$ in Problem 2.20, because at the converged point $\mathbf{u} - \mathbf{u}_k = \mathbf{0}$. Therefore, at the converged point, $\mathbf{y}^p$ is equal to $\hat{\mathbf{y}}^m$ and the first condition in Equation 2.22 matches that of the plant.

Since the cost (ϕ) and constraint ($\mathbf{g}$) functions are identical for the plant and model and $\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\left(\mathbf{u}^*_\infty\right) = \frac{\partial \mathbf{f}}{\partial \mathbf{u}}\left(\mathbf{u}^*_\infty, \boldsymbol{\beta}\right) + \boldsymbol{\lambda}_\infty$, the fourth condition of (2.22) is the same for both. Since the constraints of the plant and model are identical, the second and third conditions of Equation 2.22 match those of the plant because the Lagrange multipliers of the model and plant are also the same.

Note that, as in the ISOPE algorithm, modifier adaptation requires some method of computing the plant output gradient, $\frac{\partial \mathbf{F}}{\partial \mathbf{u}}$. In the algorithm first presented by Marchetti *et al.*

[2009], it is assumed that these derivatives are known exactly. The algorithm that corresponds with this assumption will be referred to as *ideal* modifier adaptation from this point forward. Methods of estimating the plant output gradient are explored in the following section.

2.6 Methods for Estimating Plant Output Derivatives

The simplest and most widely known scheme for estimating the plant output gradient is forward finite differences [Mansour and Ellis, 2003]:

$$\left.\frac{\partial \mathbf{F}^i}{\partial \mathbf{u}}\right|_{\mathbf{u}_k} \approx \left[\frac{\mathbf{F}(\mathbf{u}_k+\omega_1)-\mathbf{F}(\mathbf{u}_k)}{\omega_1} \quad \frac{\mathbf{F}(\mathbf{u}_k+\omega_2)-\mathbf{F}(\mathbf{u}_k)}{\omega_2} \quad \cdots \quad \frac{\mathbf{F}(\mathbf{u}_k+\omega_{n_u})-\mathbf{F}(\mathbf{u}_k)}{\omega_{n_u}}\right] \tag{2.23}$$

where $\boldsymbol{\omega}$ is a vector of small input perturbations.

Examining Equation 2.23, it is evident that a plant perturbation must be made once every iteration for *each* input variable in the optimization problem. Therefore, in a large system, this would result in many plant perturbations. Not only would this result in a loss of profit due to operation at sub-optimal points (limiting its acceptance in industry), but it would also take a considerable amount of time to make all of the perturbations. For these reasons, the finite differences method is generally inefficient for large or slow processes [Mansour and Ellis, 2003]. Furthermore, if the process is contaminated by a large amount of noise, the finite difference approximations could potentially be very inaccurate. This loss of accuracy depends on the choice of $\boldsymbol{\omega}$. However, since no standard method exists for selecting $\boldsymbol{\omega}$, extensive trial and error may be required to find good perturbation sizes.

The dual ISOPE method, discussed in Brdys and Tatjewski [1994] and Mansour and Ellis [2003] is essentially an alternative way of applying finite differences, which requires no additional plant perturbations. The following expression is applied:

$$\mathbf{S}_k \cdot \left[\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\left(\mathbf{u}_k, \mathbf{y}_k^p\right)\right]^T \approx \begin{bmatrix} \mathbf{y}_{k-1}^p - \mathbf{y}_k^p \\ \mathbf{y}_{k-2}^p - \mathbf{y}_k^p \\ \vdots \\ \mathbf{y}_{k-n_u}^p - \mathbf{y}_k^p \end{bmatrix} \tag{2.24}$$

where n_u is the number of inputs and $\mathbf{S}_k = \Big[(\mathbf{u}_{k-1} - \mathbf{u}_k) \quad (\mathbf{u}_{k-2} - \mathbf{u}_k) \quad \dots \quad (\mathbf{u}_{k-n_u} - \mathbf{u}_k)\Big]$. Measurement noise will cause the matrix on the right-hand side of Equation 2.24 to be corrupted by errors, therefore it is important that $\mathbf{S}_k$ be well conditioned. The reciprocal condition number of $\mathbf{S}_k$ can be defined as κ_k in the following way [Gao and Engell, 2005]:

$$\kappa_k = \frac{\rho_{min}(\mathbf{S}_k)}{\rho_{max}(\mathbf{S}_k)} \tag{2.25}$$

where ρ denotes a singular value. The goal is to ensure that κ_k is large enough so that excessive corruption of the derivative approximations by measurement noise is prevented. This reciprocal condition number can be manipulated by adding an extra constraint to the model-based optimization problem, which can take the form of:

$$\kappa_k \geq \varepsilon \tag{2.26}$$

where ε is a minimum threshold on the reciprocal condition number. A range of 0.1-0.2 is suggested in Tatjewski *et al.* [2001].

This methodology is definitely a positive step, in that it eliminates the need to perturb the plant many times to get an accurate derivative estimate. However, it was not selected for a couple of reasons. First, the implementation of the additional constraint could cause a loss of optimality, depending on whether the model optimum falls inside the area restricted by the constraint. Furthermore, to the author's knowledge, there has been no design method suggested to tune ε specifically to fit the nature of any particular RTO problem. Also, the additional constraint (Equation 2.26) is typically non-convex, making the resulting model-based optimization problem more difficult to solve. This is illustrated for a two input problem in Tadej and Tatjewski [2001]. In some extreme situations, this extra constraint may even make the model-based optimization problem infeasible.

Dynamic perturbation methods, also detailed in Mansour and Ellis [2003], were not considered in this thesis due mainly to their inherent complexity. It is assumed for the case studies considered in this thesis that there is enough time to allow the plant to come to steady state before another RTO iteration is required. However, for very slow processes, dynamic perturbation methods could potentially be the best option.

Another possible method of plant output gradient estimation is Broyden's method. It was originally proposed in Broyden [1965] as a way of estimating function derivatives to be used in solving systems of non-linear equations. Specifically, Broyden's method was considered to be a good option when an analytical expression for the Jacobian of the non-linear functions could not be computed (or was very difficult to evaluate) and the functions themselves were also time-consuming to evaluate [Broyden, 1965]. This was the case because, in Newton's method, if using an analytical expression for the Jacobian was impractical, finite differences was a common method of estimating the Jacobian. However, if the underlying system of equations was difficult to evaluate in the first place, finite differences would have been very time-consuming since it required an extra function evaluation to be made for each independent variable. The advantage of Broyden's method was that no extra function evaluations had to be made (Equation 2.27), which in these particular cases saved a great deal of computation time. Methods of solving systems of non-linear equations which utilize forms of Broyden's method are typically called quasi-Newton methods [Broyden, 1965].

To the author's knowledge, Broyden's method is first discussed in reference to plant output gradient estimation for RTO in Mansour and Ellis [2003]. This method, like dual ISOPE, also avoids the need for making plant operating point perturbations. Instead of using many previous operating points to update the estimate at once, it uses information from only the last operating point (in addition to information at the current operating point) to update an existing derivative estimate. Therefore, it is an iterative update scheme in which only one direction, $(\mathbf{u}_{k+1} - \mathbf{u}_k)$, is updated at a time. It can be represented by the following rank-one update formula:

$$\mathbf{B}^i_{k+1} = \mathbf{B}^i_k + \left[(\mathbf{y}^{p,i}_{k+1} - \mathbf{y}^{p,i}_k) - \mathbf{B}^i_k(\mathbf{u}_{k+1} - \mathbf{u}_k)\right] \frac{(\mathbf{u}_{k+1} - \mathbf{u}_k)^T}{(\mathbf{u}_{k+1} - \mathbf{u}_k)^T(\mathbf{u}_{k+1} - \mathbf{u}_k)} \tag{2.27}$$

where $\mathbf{B}_{k+1}$ (the Broyden update matrix) is a first order approximation of $\frac{\partial \mathbf{F}}{\partial \mathbf{u}}\big|_{\mathbf{u}_{k+1}}$.

The initialization of the Broyden update matrix is an important issue. One simple and generally effective option is to set $\mathbf{B_0}$ to the model output gradient at the initial point $\mathbf{u}_0$. This is effective as long as the approximate model is a reasonable representation of the plant at this point. If this is not the case in practice, a different approximate model should probably be selected in the first place.

2.7 Williams-Otto Reactor Benchmark Problem

In the following section, the Williams-Otto Reactor test case is introduced. This is a benchmark RTO test case which has been studied in many papers, including Forbes and Marlin [1996] and Zhang and Forbes [2000]. The reactor has been isolated here from the overall Williams-Otto Plant (described in Govindarajan and Karunanithi [2004]) and the simplifications suggested in Forbes *et al.* [1994] have been made. It will be used throughout this thesis to make comparisons between RTO algorithms, as well as to illustrate new developments.

Problem Formulation

The process consists of an ideal CSTR where a set of three reactions involving six species (A, B, C, E, G and P) are taking place:

$$\begin{aligned} A + B &\longrightarrow C \\ B + C &\longrightarrow P + E \\ C + P &\longrightarrow G \end{aligned} \tag{2.28}$$

The CSTR has two feed streams of pure A and B (flowrates F_A and F_B) , and one exiting stream (flowrate F_R). The decision variables (inputs) are the temperature of the reactor, T_R [°K], and the flow rate of feed B, F_B [kg/s]. The lower bounds for these inputs are 333.15°K and 3 kg/s and the upper bounds are 403.15°K and 7 kg/s. Note that this represents a larger feasible region than the one defined in Forbes *et al.* [1994]. The reactor mass (M_R) is 2100 kg and F_A is 1.8275 kg/s. The objective function is related to plant economics. It assumes that E and P (mass fractions represented by X_E and X_P) are the only valuable products and that both feeds, A and B, must be purchased:

$$\phi = 76.23(F_A) + 114.34(F_B) - 1143.38(X_P)(F_A + F_B) - 25.92(X_E)(F_A + F_B) \tag{2.29}$$

Note that although the objective function for this case study was stated in terms of cost, its result will sometimes be referred to as *profit* here because it is positive throughout

the normal operating range of the plant. Furthermore, all costs/profits reported for the Williams-Otto Reactor test case are in thousands of dollars.

A constraint will sometimes be added to the test case. It is a limitation on the exit mass fraction of component B:

$$X_B \leq 0.35 \tag{2.30}$$

This constraint does not appear in the formulation in Forbes *et al.* [1994], but is included here in order facilitate the illustration of various concepts throughout this thesis. The material balances that can be derived from the description of the process (Equation 2.28) are given next:

$$\begin{aligned}
F_A - (F_A + F_B)X_A - k_1^r M_R X_A X_B &= 0 \\
F_B - (F_A + F_B)X_B - k_1^r M_R X_A X_B - k_2^r M_R X_B X_C &= 0 \\
-(F_A + F_B)X_C + 2k_1^r M_R X_A X_B - 2k_2^r M_R X_B X_C - k_3^r M_R X_C X_P &= 0 \\
-(F_A + F_B)X_E + 2k_2^r M_R X_B X_C &= 0 \\
-(F_A + F_B)X_P + k_2^r M_R X_B X_C - 0.5k_3^r M_R X_C X_P &= 0 \\
k_1^r &= 1.6599 * 10^6 e^{(-6666.7/T_R)} \\
k_2^r &= 7.2117 * 10^8 e^{(-8333.3/T_R)} \\
k_3^r &= 2.6745 * 10^{12} e^{(-11111/T_R)}
\end{aligned} \tag{2.31}$$

where k_i^r is the reaction rate constant for reaction i. Note how a material balance is not written for component G, as the overall material balance ($F_R = F_A + F_B$) is written directly into these component balances, using up an extra degree of freedom.

A different set of equations will be used as the process model for these simulations. This model consists of one less reaction and one less component than the full plant model. Therefore structural plant-model mismatch exists. The reaction sequence for the model is:

$$\begin{aligned}
A + 2B &\longrightarrow P + E \\
A + B + P &\longrightarrow G
\end{aligned} \tag{2.32}$$

It is assumed that the measurements of the mass fractions of each of the components

represented in Equation 2.32 are available to the RTO system. Therefore these quantities are considered to be the process outputs.

The material balances that comprise the process model are given next:

$$\begin{aligned}
F_A - (F_A + F_B)X_A - k_1^r V_R X_A X_B^2 - k_2^r V_R X_A X_B X_P &= 0 \\
F_B - (F_A + F_B)X_B - 2k_1^r V_R X_A X_B^2 - k_2^r V_R X_A X_B X_P &= 0 \\
-(F_A + F_B)X_E + 2k_1^r V_R X_A X_B^2 &= 0 \\
-(F_A + F_B)X_P + k_1^r V_R X_A X_B^2 - k_2^r V_R X_A X_B X_P &= 0 \\
k_1^r = \nu_1 e^{(-E_1^a/T_R)} & \\
k_2^r = \nu_2 e^{(-E_2^a/T_R)} & \qquad (2.33)
\end{aligned}$$

where ν_i is the pre-exponential factor of reaction i and E_i^a is the activation energy of reaction i. Note that there are no specific values provided here for the pre-exponential factors and activation energies. Instead it is the task of the individual modeler to provide estimates for these quantities. The effect of providing different model parameter values on the performance of the RTO system will be explored subsequently.

2.8 Comparison of the Two-Step Approach and Ideal Modifier Adaptation

In this section, ideal modifier adaptation is compared with the two-step approach. Similar comparisons have been made between RTO approaches in several works in the past. In Chachuat *et al.* [2009], the two-step approach is compared qualitatively and quantitatively to both ideal modifier adaptation and direct input adaptation methods (a class of model-free methods). Also, the performance of the two-step method was compared to the ISOPE method as well as both a linear and a quadratic adaptive on-line optimization approach in Zhang and Forbes [2006] (see the appropriate references for more details on the latter two approaches).

All of the comparisons in this section are made using the two reaction sequence for the

approximate model (Equation 2.32) and the three reaction sequence for the simulated plant (Equation 2.28). Both the two-step approach and modifier adaptation were started using the same initial model. The initial form of the model essentially represents the starting point of the algorithm, because to begin the first iteration, this model is optimized to compute the first operating point that is sent to the plant. The parameters that were used are the following: $[\nu_1, \nu_2, E_1^a, E_2^a] = [1.21 \times 10^7, 7.17 \times 10^{11}, 7207, 10249]$. These parameters will be used for the process model in all Williams-Otto Reactor simulations run in this thesis, unless otherwise stated. For ideal modifier adaptation, filter parameters of 0.25 ($\mathbf{k}$) were used for the simulation and the initial modifiers were set to zero ($\mathbf{\Lambda_0} = \mathbf{0}$). There was no filtering done for the two-step approach. Finally, in all of the figures presented for the Williams-Otto Reactor test case (unless otherwise noted), the profit contours shown are for the simulated plant, thereby representing the "true" plant profit attained by the RTO system.

The first plot, Figure 2.4, shows the noiseless convergence of both the two-step method, using the pre-exponential factors (ν_1, ν_2) as the adjustable parameters and ideal modifier adaptation. Note that the activation energies (E_1^a, E_2^a) could also have been used as the adjustable parameters in the two-step approach.

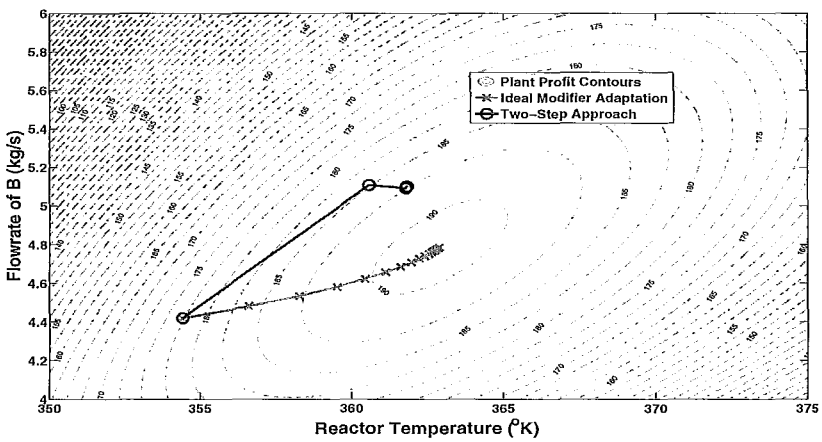

Figure 2.4: RTO methodology comparison - noiseless case

This comparison illustrates that updating the parameters of an approximate model is not enough to guarantee convergence to the optimum of a more rigorous model (in this case the

plant) [Biegler *et al.* [1985], Forbes *et al.* [1994]]. The algorithm does improve the initial operating point, however because of the structural plant-model mismatch in the reaction sequences, it converges to a sub-optimal point, realizing a loss of profit of just over 2% in this case. The ideal modifier adapatation simulation is consistent with the claim that the algorithm will, upon convergence, reach an optimum of the plant [Marchetti *et al.*, 2009]. It takes quite a few iterations to converge to the optimum, however this could likely be reduced by increasing the filter parameters.

The performance of both RTO approaches was also tested using different initial models. In addition to the initial model used in the first trial (Figure 2.4), two other initial models were tested here. In the first one, ν_1 was changed to 1.71×10^7 (model 2) and in the other, E_1^a was changed to 6707 (model 3). The results for the original model (model 1) as well as the two new models are shown in Figure 2.5:

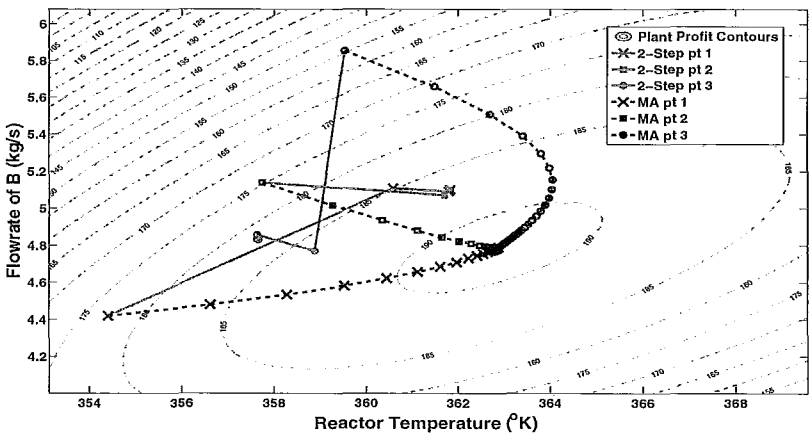

Figure 2.5: RTO methodology comparison - different initial models

It is evident from the simulations that the two-step approach does not always converge to the same point given different initial models. Only when the models differed solely by a change in one of the adjustable parameters, did two different models converge to the same point (models 1 and 2). This emphasizes the importance of model selection for the two-step approach.

The plant optimum is converged to in all the modifier adaptation simulations. This tends

to indicate that producing a very good initial model is not as important for modifier adaptation. However, if the initial model is too inaccurate, the algorithm may not converge at all. Furthermore, if the plant has multiple local optima, simulations involving different approximate models may not converge to the same point. Note that these problems are not exclusive to modifier adaptation, they exist for many other RTO schemes as well.

The noiseless convergence of both algorithms for the constrained Williams-Otto Reactor test case (recall Equation 2.30) is illustrated in Figure 2.6. The same settings that were used in the simulations for Figure 2.4 were used again here. These results demonstrate another possible drawback of the two-step method. Although the two algorithms converge to points that are very close together, the two-step method converges to an infeasible point of the plant. This could be unacceptable, depending on the consequences of violating the constraint.

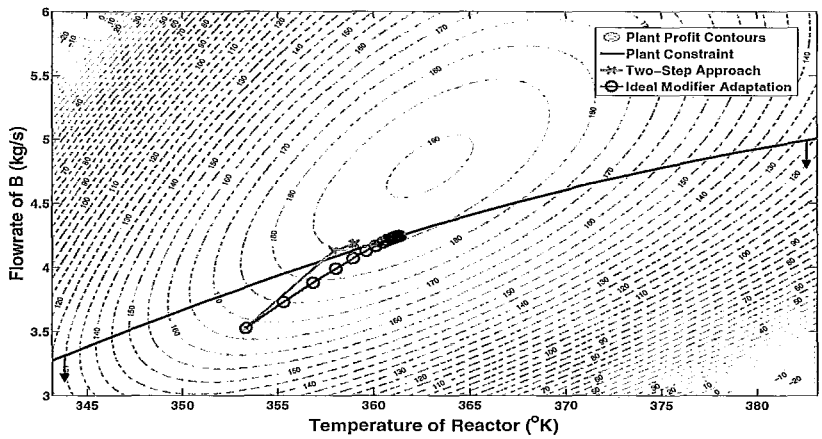

Figure 2.6: RTO methodology comparison for constrained problem - noiseless case

In the next comparison, a small amount of measurement noise is added to the mass fraction measurements (X_A, X_B, X_E, X_G and X_P). This noise has zero mean and standard deviation of approximately 2.5% of a typical set of mass fraction values. Noise was also applied to the individual plant output derivative terms in ideal modifier adaptation. This noise also had zero mean, and standard deviation of 5.0% of a typical set of values for the individual gradient elements. The noise was increased for the gradient elements in order to simulate

the fact that larger levels of noise are expected in the derivative estimates. Each algorithm was then run for 50 iterations with the results shown in Figure 2.7.

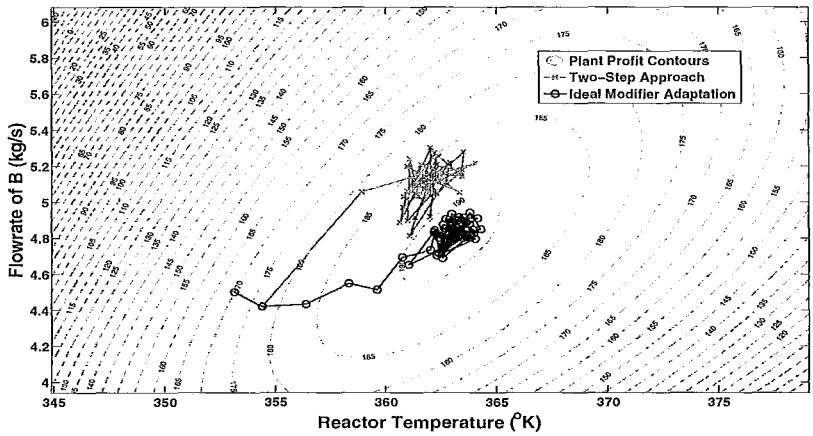

Figure 2.7: RTO methodology comparison - measurement noise added

Instead of converging to a point, both of the algorithms now move to the area surrounding the corresponding convergence point found in Figure 2.4. Both methods seem to be effected by the measurement noise to some extent. Modifier adaptation takes more iterations to arrive at the area of convergence, again because of the filtering of the modifiers. Due to the presence of measurement noise, the filter parameters cannot be increased without consequence here. The trade-off is, as the parameters of $\mathbf{K}_\Lambda$ increase, both the convergence speed and the sensitivity of the algorithm to measurement noise will increase.

The last comparison that is made evaluates the response of each approach to a fundamental change in the state of the process. Specifically, an unmeasured disturbance is artificially introduced in the inlet flow rate of component A after 100 iterations and the performance of the two RTO approaches on the unconstrained Williams-Otto Reactor process is tracked. The same noise levels that were considered for the simulations of Figure 2.7 are used again here. The results are given in Figure 2.8.

Note that the comparisons are made using the two manipulated variables, the temperature of the reactor (T_R) and the flowrate of component B (F_B). The ideal modifier adaptation algorithm tracks the step change relatively well, quickly moving toward the new plant

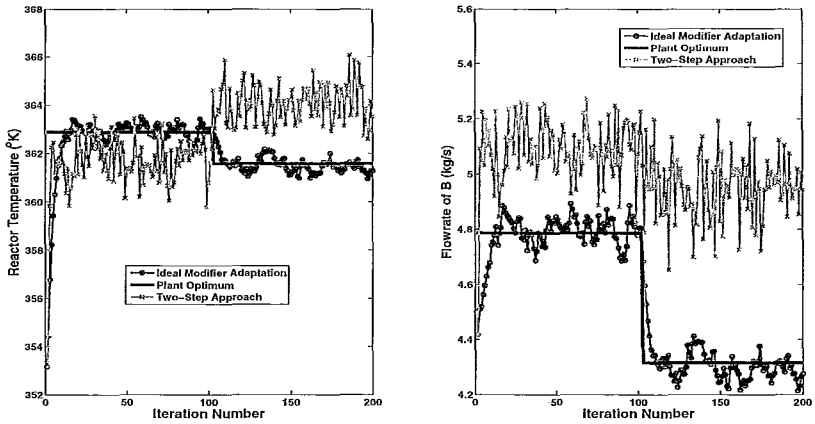

Figure 2.8: RTO methodology comparison - step change

optimum. In the two-step approach, while the flowrate of B moves toward its new plant optimal value, the temperature of the reactor actually moves away from its new plant optimal value. This illustrates the fact that when a disturbance occurs in the plant, if there is structural plant-model mismatch, the two-step approach is not guaranteed to move in the correct direction.

2.9 Chapter Summary

The goal of this chapter was to provide background information in order to better inform the reader as well as to motivate the research discussed in the remainder of the thesis. First, the different components of model-based RTO systems were discussed and references for further study were provided. This was followed by a description of four existing model-based RTO methods: the two-step approach, constraint bias updating, ISOPE and ideal modifier adaptation. In addition, different plant output-gradient estimation techniques were discussed and the Williams-Otto Reactor case study was introduced.

The last section of this chapter consisted of a comparison between the two-step approach and ideal modifier adaptation. Although ideal modifier adaptation appeared to be superior in the preceding discussion, in order to apply it in practice, the difficult task of approximat-

ing the plant output gradient still must be addressed. The formulation and analysis of a modifier adaptation scheme using Broyden's method to estimate the plant output gradient is conducted in the next chapter.

Chapter 3

Dual Modifier Adaptation

The goal of this chapter is to develop an approach to real-time optimization which not only can effectively identify the plant optimum, but also performs well in the presence of measurement noise. To begin, a modification is proposed to the ideal modifier adaptation algorithm, wherein Broyden's method is utilized to estimate the plant output gradient. A convergence analysis of this new algorithm is then carried out, followed by a discussion of the performance of this algorithm in the presence of measurement noise. Modifications are then introduced, giving the algorithm two distinct goals: seeking the optimum and achieving accurate Broyden derivative estimates. All of the concepts discussed in this chapter are demonstrated using the Williams-Otto Reactor test case.

3.1 Modifier Adaptation with Broyden's Method

The main difficulty with the online implementation of ideal modifier adaptation, as presented in Section 2.5, is that the plant output gradient is never precisely known in practice. Therefore, it must be estimated in some way from the available plant measurements. Several potential plant output gradient computation methods were discussed in Section 2.6.

Broyden's method (Equation 2.27) is chosen in this work for implementation with the

modifier adaptation algorithm for several reasons. First, it mitigates many of the drawbacks of the finite differencing method, especially the need to potentially make multiple plant perturbations in each iteration. It also proves to be fairly easy to initialize, as $\mathbf{B}_0$ can be set to the model output gradient at the initial point $\mathbf{u}_0$. This is effective as long as the approximate model gradient does not differ extensively from the true plant gradient at this point.

The update step of the ideal modifier adaptation algorithm, presented in Section 2.6 as Equation 2.19, can be rewritten to reflect the use of Broyden's method to estimate the plant output gradient:

$$\begin{bmatrix} \epsilon_{k+1} \\ \lambda^1_{k+1} \\ \vdots \\ \lambda^{n_y}_{k+1} \end{bmatrix} = (\mathbf{I} - \mathbf{K}_\Lambda) \begin{bmatrix} \epsilon_k \\ \lambda^1_k \\ \vdots \\ \lambda^{n_y}_k \end{bmatrix} + \mathbf{K}_\Lambda \begin{bmatrix} \mathbf{F}(\mathbf{u}_{k+1}) - \mathbf{f}(\mathbf{u}_{k+1}, \boldsymbol{\beta}) \\ \left[\mathbf{B}^1_{k+1} - \frac{\partial f^1}{\partial \mathbf{u}}(\mathbf{u}_{k+1}, \boldsymbol{\beta})\right]^T \\ \vdots \\ \left[\mathbf{B}^{n_y}_{k+1} - \frac{\partial f^{n_y}}{\partial \mathbf{u}}(\mathbf{u}_{k+1}, \boldsymbol{\beta})\right]^T \end{bmatrix} \tag{3.1}$$

where $\mathbf{B}^i$ is a row vector denoting the estimated derivative of output i with respect to the inputs. The algorithm diagram is given as Figure 3.1, where $\mathcal{R}$ is a non-linear map that represents the Broyden formula (Equation 2.27).

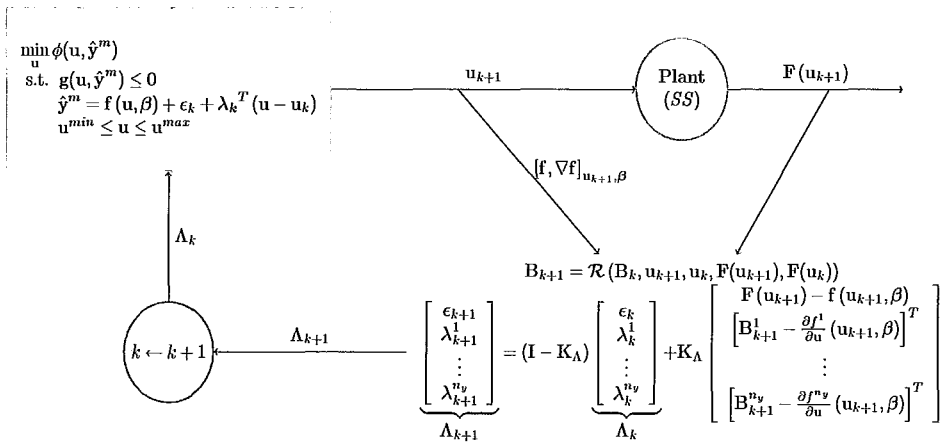

Figure 3.1: Modifier adaptation algorithm with Broyden's method

3.1.1 Williams-Otto Reactor Case Study

The Williams-Otto Reactor test case is now used to compare modifier adaptation with Broyden derivative estimates to ideal modifier adaptation. The results shown in Figure 3.2 are obtained with an absence of measurement noise and the filter parameters set to 0.25. The modifiers are also initialized to zero in this simulation ($\mathbf{\Lambda}_0 = \mathbf{0}$). In addition, the starting point ($\mathbf{u}_0$) is $[T_R, F_B] = [353.15, 4.5]$ and the initial Broyden derivative estimate is computed as follows: $\mathbf{B}_0 = \frac{df}{d\mathbf{u}}\big|_{\mathbf{u}_0}$. The next operating point ($\mathbf{u}_1$) is then calculated through the model-based optimization (using $\mathbf{\Lambda}_0$). This point is the optimum of the process model because ($\mathbf{\Lambda}_0 = \mathbf{0}$). Finally, the next plant output derivative estimate was computed as: $\mathbf{B}_1 = \mathcal{R}(\mathbf{B}_0, \mathbf{u}_1, \mathbf{u}_0, \mathbf{y}^p(\mathbf{u}_1), \mathbf{y}^p(\mathbf{u}_0))$. These same initial settings were used in all the simulations detailed in this chapter, unless otherwise noted.

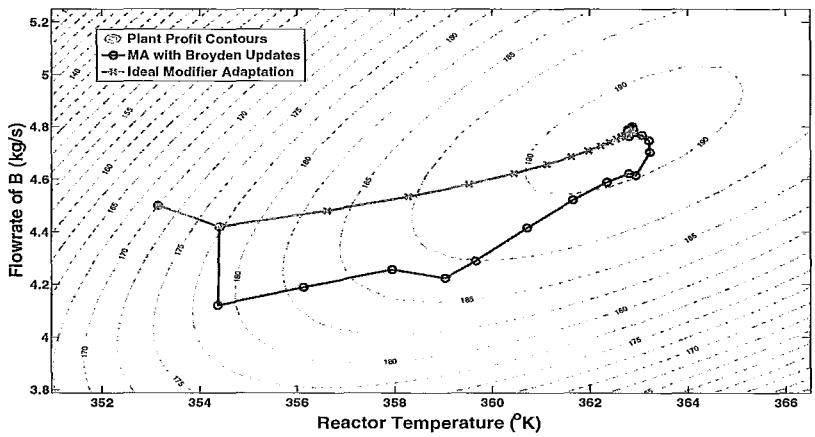

Figure 3.2: Comparison of MA with Broyden's method and ideal MA

Although the plant optimum is reached by both algorithms, it is immediately evident that it takes modifier adaptation with Broyden's method more iterations to converge very close to the plant optimum. This is because the algorithm does not move in a very direct route toward the optimum, due to inaccurate gradient information. The convergence of modifier adaptation with Broyden's method is analyzed in detail in Section 3.2. Also note that in all of the simulations in this chapter, both T_R and F_B are scaled linearly so that the interval

$[-2, 2]$ corresponds with the variable bounds outlined in Section 2.7:

$$\begin{aligned} u_1 &= 0.05\,(T_R) - 18.1575 \\ u_2 &= F_B - 5 \end{aligned} \tag{3.2}$$

Note that this was done to improve the performance of Broyden's method (see Section 3.2.3 for explanation).

Similar to ideal modifier adaptation, the performance of modifier adaptation with Broyden's method also depends on the approximate model used by the algorithm. The same three starting models that were used in Figure 2.5 of Section 2.8 are utilized again here. The filter parameters were again set to 0.25 and the same initial settings $(\mathbf{u}_0, \mathbf{\Lambda}_0)$ that were used for the simulation in Figure 3.2 were used again here. Note that the initial Broyden estimates $(\mathbf{B}_0)$ differed for each of the three initial models, as they were again computed using: $\mathbf{B}_0 = \frac{df}{du}\big|_{u_0}$. The results are shown in the Figure 3.3.

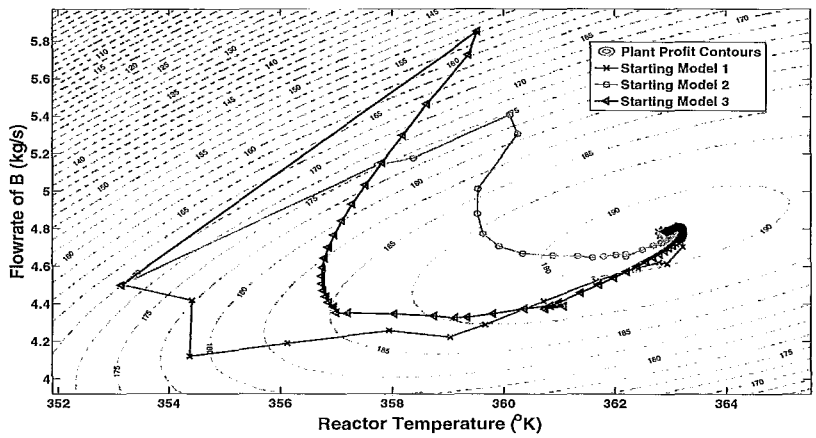

Figure 3.3: Modifier adaptation with Broyden's method using different starting models

Figure 3.3 illustrates the fact that the path taken to the convergence point and the number of iterations taken to get there both depend on the process model used in the algorithm. The algorithm converges quickly to the plant optimum in some simulations (model 1) and it gets stuck moving in the wrong direction for a considerable number of iterations before converging in other simulations (model 3). Note that no conclusions should be drawn here

about which process model provides better *overall* performance, as only a single simulation is provided as evidence here. Furthermore, although the algorithm converges to the plant optimum for all process models in this demonstration, there is no guarantee of this. This will be discussed further in Section 3.2.3.

In addition to the process model, a starting point, $\mathbf{u}_0$, must also be provided for the algorithm. It is at this point that the Broyden matrix ($\mathbf{B}$) is initialized (recall that $\mathbf{B}_0$ can be estimated as: $\mathbf{B}_0 = \frac{df}{d\mathbf{u}}\big|_{\mathbf{u}_0}$). In addition to being used in this initialization, this point will be utilized in the first Broyden update as $\mathbf{u}_k$ in Equation 2.27. It is therefore very important that this point is well chosen, as it will have a significant effect, especially on the early performance of the algorithm. To demonstrate this, simulations were run using three different starting points: $[353, 6]$, $[373, 6]$ and $[353, 3.5]$. The results are shown in Figure 3.4.

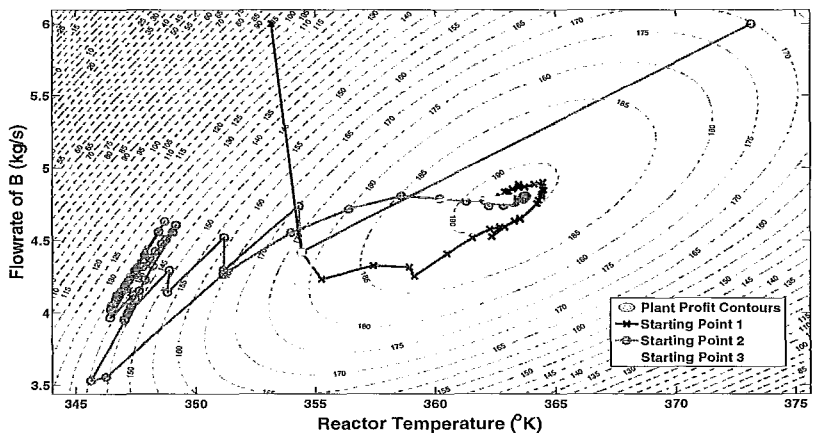

Figure 3.4: Modifier adaptation with Broyden's method using different starting points

Although all three of the simulations eventually converge to the plant optimum, the early performance of the algorithm seems to be strongly dependent on the starting point. The simulations beginning at points 1 and 3 converge quite quickly to the plant optimum. The simulation beginning at point 2, however, spends a fair number of iterations considerably far away from the plant optimum before finally moving towards it. This seems to indicate that it is important to choose a starting point ($\mathbf{u}_0$) that is reasonably close to the model

optimum ($\mathbf{u}_1$). This is not only for the benefit of the model-based optimizer, it also helps in obtaining a good first Broyden update ($\mathbf{B}_1$) (see Section 3.2.3).

The next comparison, Figure 3.5, demonstrates the effect that the filter parameter settings have on the convergence behaviour of modifier adaptation with Broyden updates. In each of the tests all of the filter parameters are set to the same value. While it is possible, even sometimes advantageous to run the algorithm with filter parameters set at different levels, investigation of this is beyond the scope of this thesis.

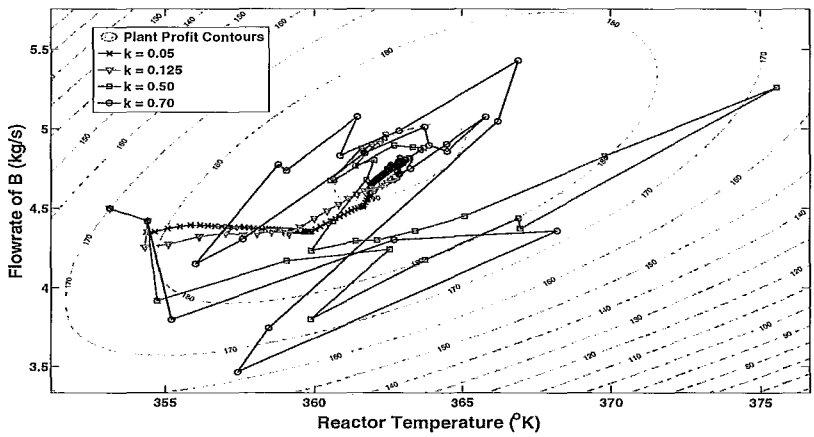

Figure 3.5: Modifier adaptation with Broyden's method with different $\mathbf{k}$ values

Upon examination of Figure 3.5, it is clear that the larger the filter parameters, the faster the movement toward the plant optimum. The downside is that, as the filter parameters increase, the path toward the optimum becomes more erratic. The trial with filter parameters of 0.05 moved very slowly, but in fairly a direct path towards the plant optimum. On the other hand, the trials with filter parameters of 0.50 and 0.70 moved around in a slightly erratic path, but managed to converge to the plant optimum fairly quickly.

Trials with $\mathbf{k} \geq 0.70$ proved to be more erratic, or in some cases did not converge at all. Therefore, for the Williams-Otto Reactor case study, a good range of $\mathbf{k}$ values appears to be $0.125 - 0.50$, however no *best* values can be identified without knowing more specific information about the RTO system in question, such as the nature and frequency of the disturbances expected in practice. Key information about the filter parameters can also be

garnered by performing a convergence analysis of the algorithm. This is addressed in detail in the next section of this thesis.

3.2 Convergence Analysis of Modifier Adaptation with Broyden's Method

The purpose of the following section is to analyze the convergence of modifier adaptation with Broyden's method, which was presented in Section 3.1. Analyzing the convergence of the algorithm is important because it should provide key information regarding the selection of the filter parameters. Most importantly, it should prevent filter parameters that lead to divergence from being chosen.

First, the algorithm will be stated formally to facilitate the analysis to follow. Then, the convergence of single input problems will be analyzed, using a modified formulation of the Williams-Otto Reactor process as a test case. Finally, the convergence of multi-input problems will be investigated.

3.2.1 Algorithm Statement and Linearization

In order to facilitate the calculation of various sensitivities, the algorithm for modifier adaptation with Broyden updates, that was presented in Section 3.1, is formally stated here. This statement is based on the one for ideal modifier adaptation that can be found in Marchetti *et al.* [2009]. The purpose of stating the algorithm in this way is to remove $\mathbf{u}_k$ from the model-based optimization problem. To begin, the following alternate modifiers are defined, which represent the constant terms of the output correction:

$$\overline{\epsilon}_k \equiv \epsilon_k - \lambda_k \mathbf{u}_k \tag{3.3}$$

Hence, the new vector of modifiers, $\overline{\Lambda} = \begin{bmatrix} \overline{\epsilon} \\ \lambda \end{bmatrix}$, is related to the previously defined set of

modifiers by [Marchetti *et al.*, 2009]:

$$\mathbf{\Lambda}_k = \mathbf{T}\left(\mathbf{u}_k\right)\overline{\mathbf{\Lambda}}_k. \tag{3.4}$$

where $\mathbf{T}\left(\mathbf{u}\right)$ is defined as follows [Marchetti *et al.*, 2009]:

$$\mathbf{T}\left(\mathbf{u}\right) \equiv \begin{bmatrix} 1 & & & \mathbf{u}^T & & \\ & \ddots & & & \ddots & \\ & & 1 & & & \mathbf{u}^T \\ & & & \mathbf{I}_{n_u} & & \\ & & & & \ddots & \\ & & & & & \mathbf{I}_{n_u} \end{bmatrix} \tag{3.5}$$

The optimization problem can then be rewritten using the alternate modifiers:

$$\begin{aligned} \mathbf{u}_{k+1} \in \arg\min_{\mathbf{u}} \quad & \phi\left(\mathbf{u}, \hat{\mathbf{y}}^m\right) \\ \text{s.t.} \quad & \hat{\mathbf{y}}^m = \mathbf{f}\left(\mathbf{u}, \boldsymbol{\beta}\right) + \overline{\boldsymbol{\epsilon}}_k + \boldsymbol{\lambda}_k^T \mathbf{u} \\ & \mathbf{g}\left(\mathbf{u}, \hat{\mathbf{y}}^m\right) \leq \mathbf{0} \end{aligned} \tag{3.6}$$

After making the appropriate substitutions, the modifier adaptation update step (Equation 3.1) appears as follows:

$$\mathbf{T}\left(\mathbf{u}_{k+1}\right)\overline{\mathbf{\Lambda}}_{k+1} = \left(\mathbf{I} - \mathbf{K}_\Lambda\right)\mathbf{T}\left(\mathbf{u}_k\right)\begin{bmatrix} \overline{\boldsymbol{\epsilon}}_k \\ \boldsymbol{\lambda}_k^1 \\ \vdots \\ \boldsymbol{\lambda}_k^{n_y} \end{bmatrix} + \mathbf{K}_\Lambda \begin{bmatrix} \mathbf{F}\left(\mathbf{u}_{k+1}\right) - \mathbf{f}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right) \\ \left[\mathbf{B}_{k+1}^1 - \frac{\partial f^1}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right)\right]^T \\ \vdots \\ \left[\mathbf{B}_{k+1}^{n_y} - \frac{\partial f^{n_y}}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right)\right]^T \end{bmatrix} \tag{3.7}$$

Lastly, due to the unique form of $\mathbf{T}\left(\mathbf{u}\right)$, the following can be written: $\mathbf{T}^{-1} = 2\mathbf{I} - \mathbf{T}\left(\mathbf{u}\right)$. This allows the update law to be restated as follows:

$$\overline{\mathbf{\Lambda}}_{k+1} = \left[2\mathbf{I} - \mathbf{T}\left(\mathbf{u}_{k+1}\right)\right]\left[\left(\mathbf{I} - \mathbf{K}_\Lambda\right)\mathbf{T}\left(\mathbf{u}_k\right)\overline{\mathbf{\Lambda}}_k + \mathbf{K}_\Lambda \begin{bmatrix} \mathbf{F}\left(\mathbf{u}_{k+1}\right) - \mathbf{f}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right) \\ \left[\mathbf{B}_{k+1}^1 - \frac{\partial f^1}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right)\right]^T \\ \vdots \\ \left[\mathbf{B}_{k+1}^{n_y} - \frac{\partial f^{n_y}}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right)\right]^T \end{bmatrix}\right] \tag{3.8}$$

This update law (Equation 3.8) can be represented by the non-linear map $\mathcal{A}$ in the following way:

$$\overline{\mathbf{\Lambda}}_{k+1} = \mathcal{A}\left(\mathbf{u}_{k+1}, \mathbf{u}_k, \overline{\mathbf{\Lambda}}_k, \mathbf{y}^p_{k+1}, \mathbf{y}^m_{k+1}, \mathbf{B}_{k+1}, \nabla\mathbf{y}^m_{k+1}\right) \tag{3.9}$$

where $\nabla\mathbf{y}^m_{k+1} = \frac{df}{d\mathbf{u}}\left(\mathbf{u}_{k+1}, \boldsymbol{\beta}\right)$. Similarly, the model-based optimization problem (Equation 3.6) can also be represented by a non-linear map:

$$\mathbf{u}_{k+1} = \mathbf{U}^*\left(\overline{\mathbf{\Lambda}}_k\right) \tag{3.10}$$

Note that $\mathbf{U}^*$ is differentiable at $\overline{\mathbf{\Lambda}}_k$ only if $\mathbf{u}_{k+1}$ is a unique optimizer of $\overline{\mathbf{\Lambda}}_k$. For this to be true, the solution of the optimization problem must satisfy the linear independence constraint qualification (LICQ), the 2nd order (sufficient) KKT conditions for a strict local minimum and the strict complementary slackness condition (see Appendix B). These two laws (3.9 and 3.10), along with the definition of $\mathbf{T}$ in Equation 3.5 and the Broyden update map ($\mathcal{R}$), make up the algorithm that will be used in the convergence analysis to follow. This overall algorithm is denoted by $\mathbf{Y}$ hereafter.

The algorithm for modifier adaptation with Broyden updates ($\mathbf{Y}$) can be thought of as a discrete-time, non-linear map. In order to accurately reflect the state of the RTO system at a given iteration (here at iteration k), three quantities need to be known: $\overline{\mathbf{\Lambda}}_k$, $\overline{\mathbf{\Lambda}}_{k-1}$ and $\mathbf{B}_k$. Therefore, the overall algorithm can be represented as follows:

$$\begin{bmatrix} \overline{\mathbf{\Lambda}}_{k+1} \\ \overline{\mathbf{\Lambda}}_k \\ \mathbf{B}_{k+1} \end{bmatrix} = \mathbf{Y}\left(\overline{\mathbf{\Lambda}}_k, \overline{\mathbf{\Lambda}}_{k-1}, \mathbf{B}_k\right) \tag{3.11}$$

Although the algorithm ($\mathbf{Y}$) is non-linear, the asymptotic behaviour of $\mathbf{Y}$ can be analyzed by considering a first order approximation (linearization) of $\mathbf{Y}$ in the vicinity of an equilibrium point, provided $\mathbf{Y}$ is differentiable at that equilibrium point [Khalil, 2001]. In practice, a stable equilibrium point for the algorithm can be identified by running it for many iterations and noting the point to which it converges (if such a point exists). Suppose that

$\left[\overline{\mathbf{\Lambda}}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty, \mathbf{B}^*_\infty\right]$ is an equilibrium point for the algorithm, $\mathbf{Y}$, that is:

$$\begin{bmatrix} \overline{\mathbf{\Lambda}}^*_\infty \\ \overline{\mathbf{\Lambda}}^*_\infty \\ \mathbf{B}^*_\infty \end{bmatrix} = \mathbf{Y}\left(\overline{\mathbf{\Lambda}}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty, \mathbf{B}^*_\infty\right) \tag{3.12}$$

A first order Taylor series expansion of the algorithm can then be written around the equilibrium point:

$$\mathbf{Y}\left(\overline{\mathbf{\Lambda}}_k, \overline{\mathbf{\Lambda}}_{k-1}, \mathbf{B}_k\right) \approx \mathbf{Y}\left(\overline{\mathbf{\Lambda}}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty, \mathbf{B}^*_\infty\right) + \nabla\mathbf{Y}\left(\overline{\mathbf{\Lambda}}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty, \mathbf{B}^*_\infty\right) \begin{bmatrix} \overline{\mathbf{\Lambda}}_k - \overline{\mathbf{\Lambda}}^*_\infty \\ \overline{\mathbf{\Lambda}}_{k-1} - \overline{\mathbf{\Lambda}}^*_\infty \\ \mathbf{B}_k - \mathbf{B}^*_\infty \end{bmatrix} \tag{3.13}$$

A new triplet is now defined, $\left[\delta\overline{\mathbf{\Lambda}}_k, \delta\overline{\mathbf{\Lambda}}_{k-1}, \delta\mathbf{B}_k\right]$, representing the distance between a particular iterate and the equilibrium point (i.e. $\delta\overline{\mathbf{\Lambda}}_k = \overline{\mathbf{\Lambda}}_k - \overline{\mathbf{\Lambda}}^*_\infty$). Using the preceding definition and Equations 3.11-3.13 the following can be written:

$$\begin{bmatrix} \delta\overline{\mathbf{\Lambda}}_{k+1} \\ \delta\overline{\mathbf{\Lambda}}_k \\ \delta\mathbf{B}_{k+1} \end{bmatrix} \approx \mathbf{\Upsilon}_\infty \begin{bmatrix} \delta\overline{\mathbf{\Lambda}}_k \\ \delta\overline{\mathbf{\Lambda}}_{k-1} \\ \delta\mathbf{B}_k \end{bmatrix} \tag{3.14}$$

where $\mathbf{\Upsilon}_\infty$ is equal to $\nabla\mathbf{Y}$ evaluated at the equilibrium point. It can be defined specifically as the following matrix of sensitivities:

$$\mathbf{\Upsilon}_\infty = \begin{bmatrix} \frac{d\overline{\mathbf{\Lambda}}_{k+1}}{d\overline{\mathbf{\Lambda}}_k} & \frac{d\overline{\mathbf{\Lambda}}_{k+1}}{d\overline{\mathbf{\Lambda}}_{k-1}} & \frac{d\overline{\mathbf{\Lambda}}_{k+1}}{d\mathbf{B}_k} \\ \mathbf{I} & \mathbf{0} & \frac{d\overline{\mathbf{\Lambda}}_k}{d\mathbf{B}_k} \\ \frac{d\mathbf{B}_{k+1}}{d\overline{\mathbf{\Lambda}}_k} & \frac{d\mathbf{B}_{k+1}}{d\overline{\mathbf{\Lambda}}_{k-1}} & \frac{d\mathbf{B}_{k+1}}{d\mathbf{B}_k} \end{bmatrix}_{\left[\overline{\mathbf{\Lambda}}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty, \mathbf{B}^*_\infty\right]} \tag{3.15}$$

Now, if the eigenvalues of $\mathbf{\Upsilon}_\infty$ are computed, the asymptotic behaviour of the algorithm in the neighbourhood of the equilibrium point can be characterized. The following theorem, adapted from Antsaklis and Michel [2007], describes how this is done:

Theorem 3.1 (Asymptotic stability of Y) The algorithm, $\mathbf{Y}$, is asymptotically stable in the vicinity of an equilibrium point if and only if the magnitudes of all of the eigenvalues of $\mathbf{\Upsilon}_\infty$ are less than one, that is, all of the eigenvalues of $\mathbf{\Upsilon}_\infty$ are located inside the unit circle of the complex plane.

Proof. See [Theorem 4.47, Antsaklis and Michel [2007]] for an explanation.

3.2.2 Convergence Analysis of Single Input Problems

Analyzing the convergence of single input problems is considerably easier than it is for multiple input problems. This simplicity comes from the fact that, for single input problems, the Broyden method formula (Equation 2.27), reduces to the following expression (a simple finite difference):

$$\mathbf{B}^i_{k+1} = \frac{\mathbf{y}^{p,i}_{k+1} - \mathbf{y}^{p,i}_{k}}{u_{k+1} - u_k} \tag{3.16}$$

Notice that in Equation 3.16 the dependence on the previous Broyden iterate, $\mathbf{B}_k$, vanishes. Therefore, the state of the system no longer depends on $\mathbf{B}_k$, and Equations 3.14 and 3.15 can be rewritten as follows:

$$\begin{bmatrix} \delta\overline{\mathbf{\Lambda}}_{k+1} \\ \delta\overline{\mathbf{\Lambda}}_{k} \end{bmatrix} \approx \begin{bmatrix} \frac{d\overline{\mathbf{\Lambda}}_{k+1}}{d\overline{\mathbf{\Lambda}}_{k}} & \frac{d\overline{\mathbf{\Lambda}}_{k+1}}{d\overline{\mathbf{\Lambda}}_{k-1}} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}_{[\overline{\mathbf{\Lambda}}^*_{\infty}, \overline{\mathbf{\Lambda}}^*_{\infty}]} \begin{bmatrix} \delta\overline{\mathbf{\Lambda}}_{k} \\ \delta\overline{\mathbf{\Lambda}}_{k-1} \end{bmatrix} \tag{3.17}$$

The set of sensitivities in (3.17) are not trivial to compute, as they require careful examination of the modifier adaptation algorithm. Considering $\frac{d\overline{\mathbf{\Lambda}}_{k+1}}{d\overline{\mathbf{\Lambda}}_{k}}$, there are a variety of paths through which the modifiers at iteration k can effect the modifiers at iteration $k+1$. First of all, this influence can come directly from the values of the previous modifiers through the first-order exponential filter (see Equation 3.8). However, the previous modifiers also effect the new input values determined by the optimizer (see Equation 3.10). These new inputs then effect the output information (values and gradients) which in turn influence the current modifiers. All six possible pathways are illustrated in Figure 3.6 and can be expressed mathematically as follows:

$$\begin{aligned} \frac{d\overline{\mathbf{\Lambda}}_{k+1}}{d\overline{\mathbf{\Lambda}}_{k}} = {} & \frac{\partial \mathcal{A}}{\partial \overline{\mathbf{\Lambda}}_k} + \left(\frac{\partial \mathcal{A}}{\partial u_{k+1}} + \frac{\partial \mathcal{A}}{\partial \mathbf{B}_{k+1}} \cdot \left(\frac{\partial \mathcal{R}}{\partial u_{k+1}} + \frac{\partial \mathcal{R}}{\partial y^p_{k+1}} \cdot \frac{d\mathbf{F}}{du}\Big|_{u_{k+1}} \right) + \frac{\partial \mathcal{A}}{\partial \mathbf{y}^p_{k+1}} \cdot \frac{d\mathbf{F}}{du}\Big|_{u_{k+1}} \right) \cdot \frac{d\mathbf{U}^*}{d\overline{\mathbf{\Lambda}}}\Big|_{\overline{\mathbf{\Lambda}}_k} \\ & + \left(\frac{\partial \mathcal{A}}{\partial \mathbf{y}^m_{k+1}} \cdot \frac{d\mathbf{f}}{du}\Big|_{u_{k+1},\beta} + \frac{\partial \mathcal{A}}{\partial \nabla \mathbf{y}^m_{k+1}} \cdot \frac{\partial^2 \mathbf{f}}{\partial u^2}\Big|_{u_{k+1},\beta} \right) \cdot \frac{d\mathbf{U}^*}{d\overline{\mathbf{\Lambda}}}\Big|_{\overline{\mathbf{\Lambda}}_k} \end{aligned} \tag{3.18}$$

where $\frac{d\mathbf{F}}{du}\big|_{u_{k+1}}$ can be approximated by $\mathbf{B}_{k+1}$. The most difficult set of sensitivities to compute are $\left(\frac{\partial \mathcal{R}}{\partial u_{k+1}} + \frac{\partial \mathcal{R}}{\partial y^p_{k+1}} \cdot \frac{d\mathbf{F}}{du}\big|_{u_{k+1}} \right)$. The derivation for this set of sensitivities gives the

following: $\frac{1}{2}\frac{d^2\mathbf{F}}{du^2}\Big|_{u_{k+1}}$. Full details on the calculation of each individual sensitivity appearing in Equation 3.18 can be found in Appendix C.

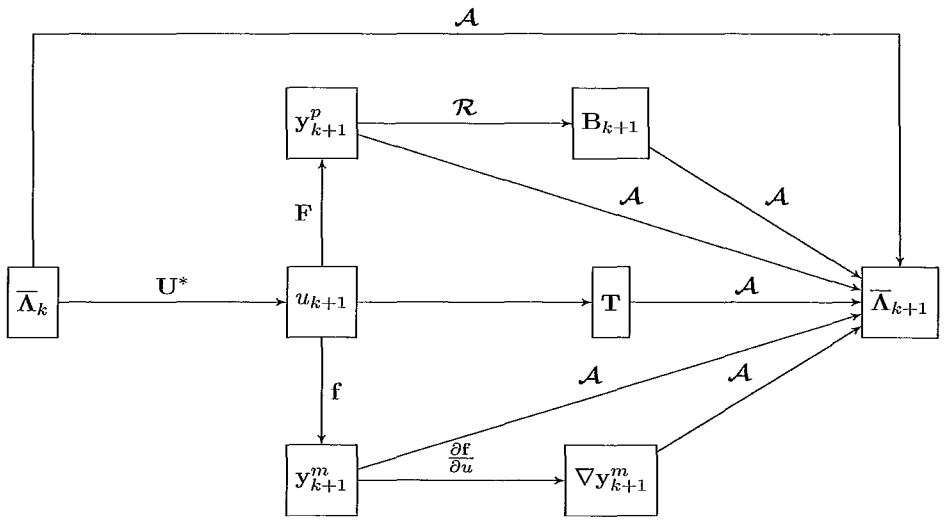

Figure 3.6: Flowchart illustrating pathways through which $\overline{\mathbf{\Lambda}}_k$ influences $\overline{\mathbf{\Lambda}}_{k+1}$

The other term in Equation 3.17 is easier to evaluate. The influence of $\overline{\mathbf{\Lambda}}_{k-1}$ on $\overline{\mathbf{\Lambda}}_{k+1}$ comes entirely through the modified update law (Equation 3.9). It can be represented by the following expression:

$$\frac{d\overline{\mathbf{\Lambda}}_{k+1}}{d\overline{\mathbf{\Lambda}}_{k-1}} = \frac{\partial \mathcal{A}}{\partial u_k} \cdot \frac{d\mathbf{U}^*}{d\overline{\mathbf{\Lambda}}}\Big|_{\overline{\mathbf{\Lambda}}_{k-1}} \tag{3.19}$$

Williams-Otto Reactor Case Study

In order to test the convergence analysis procedure, the Williams-Otto Reactor problem, originally presented in Section 2.7, is modified here. Specifically, the feed rate of component B (F_B) is fixed at 4.6 kg/s. This effectively makes the Williams-Otto Reactor a single-input problem, with the temperature of the reactor (T_R) being the only input.

A plot showing how the profit of the plant varies with changes in the temperature of the reactor (T_R) is given as Figure 3.7. This plot also shows the convergence of a modifier adaptation system using Broyden updates. In this simulation, filter parameters of 0.5 are

used. Convergence to the general area of the plant optimum is achieved in only a handful of iterations.

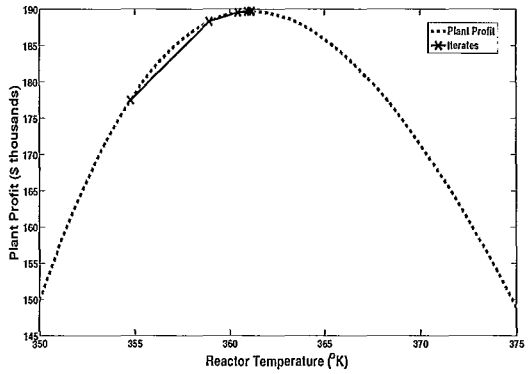

Figure 3.7: Plant profit plot for Williams-Otto Reactor single input system

In this case study, upon examination of Equation 2.29, two outputs need to be modified in order to match the KKT conditions of the model to those of the plant. They are the mass fraction of component E and the mass fraction of component P. This means that four modifiers must be updated, one bias and one gradient modifier per output. Therefore, there are a total of eight variables that are needed to describe the state of the system for this single input case study (both $\overline{\Lambda}_{k+1,i}$ and $\overline{\Lambda}_{k,i}$ for each updated modifier i).

Numerous tests were run to examine the convergence of modifier adaptation with Broyden's method for this case study. The analysis was done by looking at the eigenvalues of $\boldsymbol{\Upsilon}_{\infty}$ for different sets of filter parameters, $\mathbf{k}$. Note that only cases where all the filter parameters were the same were considered here.

Filter parameters of 0.50 are considered first. The non-zero eigenvalues ($\mathbf{E}^{val}$) of the $\boldsymbol{\Upsilon}_{\infty}$ matrix are given below:

$$\mathbf{E}^{val} = \left[0.5, 0.5, 0.5, 0.3272 \pm 0.4884i,\right] \tag{3.20}$$

Since all of the eigenvalues lie within the unit circle, the algorithm is asymptotically stable near equilibrium point [Antsaklis and Michel, 2007]. This means that the algorithm should move toward the equilibrium point when it is started "sufficiently" close to it. Since projections can only be taken on a two-dimensional axis, and the system has 8 states in total,

a couple of different plots are shown in Figure 3.8 to confirm that the system is indeed moving toward the equilibrium point.

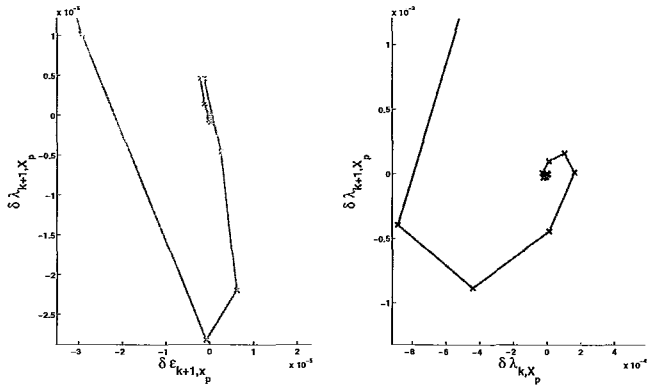

Figure 3.8: Modifiers approaching equilibrium point with $\mathbf{k} = 0.50$

To get a sense of which filter parameter values would support stable operation and conversely which filter parameter values would cause the algorithm to diverge, the gain matrix (Υ_∞) is computed for a range of filter parameters between 0.01 and 2. The eigenvalues of each gain matrix are then computed and the eigenvalue with the largest magnitude (dominant eigenvalue) for each filter parameter sample is isolated. These dominant eigenvalues are plotted against their corresponding filter parameter values in Figure 3.9.

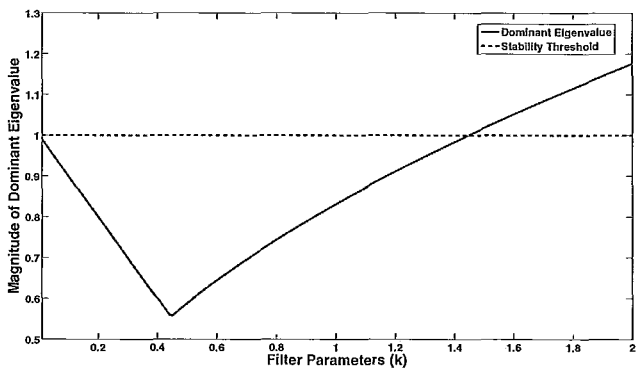

Figure 3.9: Dominant eigenvalue for various filter parameter values

Figure 3.9 illustrates the fact that any sets of filter parameters in the typical range of 0 to 1 should provide stable performance. When the magnitude of the dominant eigenvalue of the gain matrix is greater than one, this indicates that the algorithm will likely be unstable in

practice. To test this, filter parameter values of 1.5 were implemented for the single input Williams-Otto Reactor system. The algorithm was run using Broyden's method to estimate the plant output gradient, with no measurement noise. The convergence analysis prediction was confirmed as the algorithm quickly diverged.

3.2.3 Convergence Analysis of Multiple-Input Problems

Unfortunately, multiple-input problems cannot be analyzed as simply as single-input problems. The main difficulty stems from the fact that, in the multiple-input case, some of the derivative terms in Υ_∞ (Equation 3.15) are not defined at the convergence point $\left[\overline{\mathbf{A}}^*_\infty, \overline{\mathbf{A}}^*_\infty, \mathbf{B}^*_\infty\right]$. For instance, $\frac{\partial \mathbf{B}^i_{k+1}}{\partial \mathbf{B}^i_k}$ can be represented by the following, for a system with two inputs:

$$\frac{\partial \mathbf{B}^i_{k+1}}{\partial \mathbf{B}^i_k} = \begin{bmatrix} 1 - \frac{\Delta^2_{k+1,1}}{\Delta^2_{k+1,1}+\Delta^2_{k+1,2}} & -\frac{\Delta_{k+1,1}\Delta_{k+1,2}}{\Delta^2_{k+1,1}+\Delta^2_{k+1,2}} \\ -\frac{\Delta_{k+1,1}\Delta_{k+1,2}}{\Delta^2_{k+1,1}+\Delta^2_{k+1,2}} & 1 - \frac{\Delta^2_{k+1,2}}{\Delta^2_{k+1,1}+\Delta^2_{k+1,2}} \end{bmatrix} \tag{3.21}$$

where $\Delta_{k+1,i} = u_{k+1,i} - u_{k,i}$ (the difference in component i between the current and previous operating points). As the algorithm converges, the fractional expressions in this matrix approach an undefined form $(\frac{0}{0})$. Therefore, linearization of the algorithm is not possible at the convergence point.

To demonstrate the convergence behaviour of the algorithm, the Williams-Otto Reactor test case is used. The Broyden derivative for one of the outputs, X_p, will be examined. This output can be considered in isolation because in the Broyden formula (Equation 2.27) there is no interaction between outputs. The algorithm is run under noiseless conditions, with filter parameters of 0.25 and the two terms in the top row of Equation 3.21 are plotted. The term at position (1,1) is shown at the top of Figure 3.10 and the term at position (1,2) is shown at the bottom of the figure.

It is clear here that both Broyden derivative terms are not converging after 350 iterations. This confirms that the Broyden update procedure and by extension the entire algorithm is not differentiable at the convergence point. Therefore, linearization, as discussed in

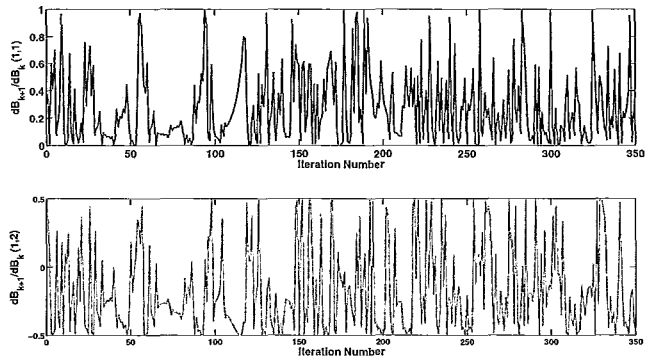

Figure 3.10: Broyden derivative values as the algorithm converges

Equations 3.14 and 3.15, is not possible. Note that a possible solution to this difficulty would be the application of Lyapunov methods, which do not require differentiability of the algorithm. This could potentially be a topic of future research.

There is another concept that needs to be discussed related to the use of Broyden's method to estimate the plant output gradient for multiple-input systems. Up until this point, nothing has been said about the nature of the point that modifier adaptation with Broyden's method converges to. It was said earlier that in the case of ideal modifier adaptation, as long as the algorithm converged, it would converge to a KKT point of the plant. This same guarantee *cannot* be made when Broyden's method is employed to estimate the plant output derivatives in multiple-input systems.

To be effective, Broyden's method needs consistent excitation in all directions. In the previous section, the comment was made that Broyden's method updates in only one direction, $(\mathbf{u}_{k+1} - \mathbf{u}_k)$, in each iteration. Therefore, any direction perpendicular to this one is completely ignored by the method. This can be confirmed by locating the set of directions, $\mathbf{M}$, such that the Broyden estimate does not change between two successive iterations:

$$\mathbf{B}_{k+1}\mathbf{m} = \mathbf{B}_k\mathbf{m}, \ \forall \mathbf{m} \in \mathbf{M} \tag{3.22}$$

By inspection of 2.27 it is easy to find that the set of directions, $\mathbf{M}$, is the $\Re^{n_u-1}$ dimensional subspace comprising every direction perpendicular to the update direction from the current

iteration, $(\mathbf{u}_{k+1} - \mathbf{u}_k)$. This property does not cause a severe problem in a single iteration. However, if the same set of directions is ignored iteration after iteration, the method may not ever identify the correct plant output gradient, causing the algorithm to fail to converge to a stationary point of the plant.

A more detailed mathematical analysis of this concept is now presented. The disparity between the Broyden plant output gradient and the true plant output gradient is referred to as the gradient offset subsequently. This gradient offset can be defined as follows:

$$\mathbf{e}^i_{k+1} = \mathbf{B}^i_{k+1} - \frac{\partial F^i}{\partial \mathbf{u}}(\mathbf{u}_{k+1}) \tag{3.23}$$

where $\mathbf{e}^i_{k+1}$ is a row vector for the gradient offset for output i at iteration $k+1$.

An expression is now developed that approximates this offset at any given iteration, as long as the offset at the previous iteration is already known. To begin, the direction of the last operating point update is normalized to give the following:

$$\boldsymbol{\zeta}_{k+1} = \frac{\mathbf{u}_{k+1} - \mathbf{u}_k}{\| \mathbf{u}_{k+1} - \mathbf{u}_k \|} \tag{3.24}$$

where $\boldsymbol{\zeta}_{k+1}$ is the unit direction representing the last operating point move. Note that 3.24 cannot be applied if the operating point did not change between iterations k and $k+1$.

The projection of the gradient offset in the direction $\boldsymbol{\zeta}_{k+1}$ can now be written:

$$\mathbf{e}^i_{k+1}\boldsymbol{\zeta}_{k+1} = \left(\mathbf{B}^i_{k+1} - \frac{\partial F^i}{\partial \mathbf{u}}(\mathbf{u}_{k+1})\right)\frac{[\mathbf{u}_{k+1} - \mathbf{u}_k]}{\| \mathbf{u}_{k+1} - \mathbf{u}_k \|} \tag{3.25}$$

Approximating $\frac{\partial F^i}{\partial \mathbf{u}}(\mathbf{u}_{k+1})$ using the Broyden update expression (Equation 2.27), the following expression can be obtained:

$$\mathbf{e}^i_{k+1}\boldsymbol{\zeta}_{k+1} \approx \frac{\mathbf{B}^i_{k+1}[\mathbf{u}_{k+1} - \mathbf{u}_k] - y^{p,i}_{k+1} + y^{p,i}_k}{\| \mathbf{u}_{k+1} - \mathbf{u}_k \|} \tag{3.26}$$

Now, writing a Taylor series expansion for $y^{p,i}_k \equiv F^i(\mathbf{u}_k)$ around $\mathbf{u}_{k+1}$ and ignoring terms greater than second order:

$$F^i(\mathbf{u}_k) \approx F^i(\mathbf{u}_{k+1}) + \frac{\partial F^i}{\partial \mathbf{u}}(\mathbf{u}_{k+1})[\mathbf{u}_k - \mathbf{u}_{k+1}] + \frac{1}{2}[\mathbf{u}_k - \mathbf{u}_{k+1}]^T \frac{\partial^2 F^i}{\partial \mathbf{u}^2}(\mathbf{u}_{k+1})[\mathbf{u}_k - \mathbf{u}_{k+1}] \tag{3.27}$$

The Taylor series expansion of Equation 3.27 is now substituted into Equation 3.26 to get the final expression for the projection of the gradient offset in the unit direction of the last operating point move:

$$\mathbf{e}^i_{k+1}\boldsymbol{\zeta}_{k+1} \approx \frac{1}{2}\boldsymbol{\zeta}^T_{k+1}\frac{\partial^2 F^i}{\partial \mathbf{u}^2}(\mathbf{u}_{k+1})\,\boldsymbol{\zeta}_{k+1} \parallel \mathbf{u}_{k+1} - \mathbf{u}_k \parallel \tag{3.28}$$

The gradient offset in directions normal to the direction of the last operating point change ($\boldsymbol{\zeta}_{k+1}$) is now considered. If $\mathbf{N} = [\mathbf{n}_1, \mathbf{n}_2, \ldots \mathbf{n}_{n_u-1}]$ represents an orthonormal basis of the $(n_u - 1)$-dimensional subspace orthogonal to $\boldsymbol{\zeta}_{k+1}$, then the projection of the gradient offset in any unit direction $\mathbf{n}_j$ gives:

$$\mathbf{e}^i_{k+1}\mathbf{n}_j = \left[\mathbf{B}^i_{k+1} - \frac{\partial F^i}{\partial \mathbf{u}}(\mathbf{u}_{k+1})\right]\mathbf{n}_j \tag{3.29}$$

Note that the Broyden update matrix is not changed in any direction orthogonal to the direction of the last operating point change, $\mathbf{B}^i_{k+1}\mathbf{n}_j = \mathbf{B}^i_k\mathbf{n}_j$. Using this, and the definition of the offset ($\mathbf{B}^i_k = \mathbf{e}^i_k + \frac{\partial F^i}{\partial \mathbf{u}}(\mathbf{u}_k)$) the following expression can be written:

$$\mathbf{e}^i_{k+1}\mathbf{n}_j = \left[\mathbf{e}^i_k + \frac{\partial F^i}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial F^i}{\partial \mathbf{u}}(\mathbf{u}_{k+1})\right]\mathbf{n}_j \tag{3.30}$$

Finally, writing a Taylor series expansion for $\frac{\partial F^i}{\partial \mathbf{u}}(\mathbf{u}_k)$ around $\mathbf{u}_{k+1}$ and ignoring terms greater than first order yields:

$$\mathbf{e}^i_{k+1}\mathbf{n}_j \approx \left[\mathbf{e}^i_k - \frac{\partial^2 F^i}{\partial \mathbf{u}^2}(\mathbf{u}_{k+1})\,\boldsymbol{\zeta}_{k+1} \parallel \mathbf{u}_{k+1} - \mathbf{u}_k \parallel\right]\mathbf{n}_j \tag{3.31}$$

To finish this derivation, the projections of the offset in the orthogonal subspaces $\boldsymbol{\zeta}_{k+1}$ and $\mathbf{N}$ can be combined using $\mathbf{e}^{i,T}_{k+1} = \boldsymbol{\zeta}_{k+1}\boldsymbol{\zeta}^T_{k+1}\mathbf{e}^{i,T}_{k+1} + \mathbf{N}\mathbf{N}^T\mathbf{e}^{i,T}_{k+1}$. Since $\mathbf{N}\mathbf{N}^T = \mathbf{I} - \boldsymbol{\zeta}_{k+1}\boldsymbol{\zeta}^T_{k+1}$, the following comprehensive expression can be written:

$$\mathbf{e}^{i,T}_{k+1} = \left[\mathbf{I} - \boldsymbol{\zeta}_{k+1}\boldsymbol{\zeta}^T_{k+1}\right]\mathbf{e}^{i,T}_k - \left[\mathbf{I} - \frac{3}{2}\boldsymbol{\zeta}_{k+1}\boldsymbol{\zeta}^T_{k+1}\right]\frac{\partial^2 F^i}{\partial \mathbf{u}^2}(\mathbf{u}_{k+1})\,\boldsymbol{\zeta}_{k+1} \parallel \mathbf{u}_{k+1} - \mathbf{u}_k \parallel \tag{3.32}$$

The gradient offset in Equation 3.32 appears to grow as the step-size is increased. This provides motivation for limiting the size of the steps taken by the modifier adaptation algorithm. For instance, if $\mathbf{u}_0$ and $\mathbf{u}_1$ are far apart, the gradient offset for the first few iterations of the algorithm will be large. More importantly, the first term of Equation 3.32

shows that the gradient offset may not vanish even when the algorithm is converging and the step-size, $\| \mathbf{u}_{k+1} - \mathbf{u}_k \|$, becomes very small. Mitigation of these issues will be discussed later in this chapter.

There is one more issue relating to modifier adaptation with Broyden updates, for multiple input systems, that needs to be discussed. In the denominator of the Broyden update definition (Equation 2.27) the following term appears: $[\mathbf{u}_{k+1} - \mathbf{u}_k]^T [\mathbf{u}_{k+1} - \mathbf{u}_k]$. For a two input system this can be rewritten as: $(u_{k+1,1} - u_{k,1})^2 + (u_{k+1,2} - u_{k,2})^2$. It is clear that this term closely relates changes in both inputs. Therefore, it is advantageous if both inputs are scaled so that they change by similar amounts. This is also important for systems with more than two inputs as well.

Instead of rescaling the inputs directly, a more systematic way to deal with the issue would be to use to restate the Broyden update formula (Equation 2.27) as follows:

$$\mathbf{B}^i_{k+1} = \mathbf{B}^i_k + \left[(\mathbf{y}^{p,i}_{k+1} - \mathbf{y}^{p,i}_k) - \mathbf{B}^i_k(\mathbf{u}_{k+1} - \mathbf{u}_k)\right] \frac{(\mathbf{u}_{k+1} - \mathbf{u}_k)^T \mathcal{M}}{(\mathbf{u}_{k+1} - \mathbf{u}_k)^T \mathcal{M} (\mathbf{u}_{k+1} - \mathbf{u}_k)} \tag{3.33}$$

where $\mathcal{M}$ is a (typically diagonal) scaling matrix.

Application to the Williams-Otto Reactor Test Case

The importance of input variable scaling, when Broyden's method is employed to estimate the plant output gradient, is demonstrated in Figure 3.11. Here, the Williams-Otto Reactor test case is run both with no input variable scaling and with the scaling described earlier in this chapter. The difference in performance is considerable, as the unscaled simulation does not even converge to the plant optimum after over 100 iterations.

3.3 Sensitivity to Measurement Noise

Another significant deficiency of Broyden's method, in addition to the gradient offset problem described in the previous section, is now discussed. Upon close examination of the

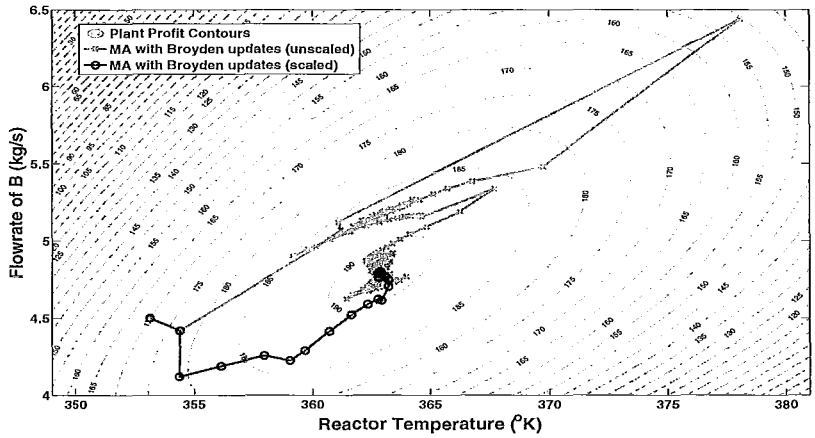

Figure 3.11: Williams-Otto Reactor test case with and without scaling

Broyden formula (Equation 2.27), it is clear that the output gradient approximation will become artificially large if in the presence of measurement noise, the new operating point ($\mathbf{u}_{k+1}$) is very close to the previous operating point ($\mathbf{u}_k$). This can be easily seen mathematically by considering the gradient estimate in the direction of the last operating point move, ζ_{k+1} :

$$\mathbf{B}^i_{k+1}\zeta_{k+1} = \frac{y^{p,i}_{k+1} - y^{p,i}_k}{\| \mathbf{u}_{k+1} - \mathbf{u}_k \|} \tag{3.34}$$

This problem is briefly alluded to in Mansour and Ellis [2003] and Gao and Engell [2005] and will be referred to as the peaking phenomenon hereafter.

Before specific solutions are explored, it is useful to consider the peaking problem from a more mathematical point of view. This will aid in confirming the root causes of the problem and may help in the evaluation of potential solutions. To this end, a first-order approximation of the variance in the Broyden derivative estimates is made. Examining Equation 2.27 again, there are three pathways through which uncertainty can propagate to the Broyden updates ($\mathbf{B}^i_{k+1}$). Both the current ($k+1$) and previous (k) measurements can be noisy and this noise can be easily propagated through to the Broyden derivative estimates via the corresponding terms in the update equation. In addition, variance in the previous Broyden estimates is propagated through to the new estimates via the $\mathbf{B}^i_k$ term in Equation 2.27.

To sum this up mathematically, the triplet $\left[\mathbf{B}_{k+1}^{i}, y_{k+1}^{p,i}, y_{k}^{p,i}\right]$ can be written as a function of the three quantities described in the previous paragraph:

$$\begin{bmatrix} \mathbf{B}_{k+1}^{i} \\ y_{k+1}^{p,i} \\ y_{k}^{p,i} \end{bmatrix} = \mathbf{R}\left(\mathbf{B}_{k}^{i}, y_{k+1}^{p,i}, y_{k}^{p,i}\right) \tag{3.35}$$

where $\mathbf{R}$ involves the Broyden update map, $\boldsymbol{\mathcal{R}}$ (Equation 2.27). $\mathbf{R}$ is then linearized around the mean values of the Broyden derivative estimates and plant measurement quantities:

$$\mathbf{R}\left(\mathbf{B}_{k}^{i}, y_{k+1}^{p,i}, y_{k}^{p,i}\right) \approx \mathbf{R}\left(\hat{\mathbf{B}}_{k}^{i}, \hat{y}_{k+1}^{p,i}, \hat{y}_{k}^{p,i}\right) + \nabla\mathbf{R}\left(\hat{\mathbf{B}}_{k}^{i}, \hat{y}_{k+1}^{p,i}, \hat{y}_{k}^{p,i}\right) \begin{bmatrix} \mathbf{B}_{k}^{i} - \hat{\mathbf{B}}_{k}^{i} \\ y_{k+1}^{p,i} - \hat{y}_{k+1}^{p,i} \\ y_{k}^{p,i} - \hat{y}_{k}^{p,i} \end{bmatrix} \tag{3.36}$$

where $\hat{\mathbf{B}}_{k}^{i}$ denotes the mean values of the Broyden derivative estimates for the output i, $\hat{y}_{k+1}^{p,i}$ and $\hat{y}_{k}^{p,i}$ are the mean values of the plant measurement i (at particular iterations) and $\nabla\mathbf{R}$ can be expressed as follows:

$$\nabla\mathbf{R} = \begin{bmatrix} \frac{\partial\boldsymbol{\mathcal{R}}}{\partial\mathbf{B}_{k}^{i}}^{T} & \mathbf{0} & \mathbf{0} \\ \frac{\partial\boldsymbol{\mathcal{R}}}{\partial y_{k+1}^{p,i}}^{T} & \mathbf{I} & \mathbf{0} \\ \frac{\partial\boldsymbol{\mathcal{R}}}{\partial y_{k}^{p,i}}^{T} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.37}$$

Now, using Equations 3.35-3.37, as well as the definitions of variance and covariance, the following expression for the variance of the triplet $\left[\mathbf{B}_{k+1}^{i}, y_{k+1}^{p,i}, y_{k}^{p,i}\right]$ can be written [Arras, 1998]:

$$\mathbf{V}\begin{pmatrix} \mathbf{B}_{k+1}^{i} \\ y_{k+1}^{p,i} \\ y_{k}^{p,i} \end{pmatrix} \approx \begin{bmatrix} \frac{\partial\boldsymbol{\mathcal{R}}}{\partial\mathbf{B}_{k}^{i}}^{T} & \mathbf{0} & \mathbf{0} \\ \frac{\partial\boldsymbol{\mathcal{R}}}{\partial y_{k+1}^{p,i}}^{T} & \mathbf{I} & \mathbf{0} \\ \frac{\partial\boldsymbol{\mathcal{R}}}{\partial y_{k}^{p,i}}^{T} & \mathbf{0} & \mathbf{I} \end{bmatrix}^{T} \mathbf{V}\begin{pmatrix} \mathbf{B}_{k}^{i} \\ y_{k+1}^{p,i} \\ y_{k}^{p,i} \end{pmatrix} \begin{bmatrix} \frac{\partial\boldsymbol{\mathcal{R}}}{\partial\mathbf{B}_{k}^{i}}^{T} & \mathbf{0} & \mathbf{0} \\ \frac{\partial\boldsymbol{\mathcal{R}}}{\partial y_{k+1}^{p,i}}^{T} & \mathbf{I} & \mathbf{0} \\ \frac{\partial\boldsymbol{\mathcal{R}}}{\partial y_{k}^{p,i}}^{T} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.38}$$

where $\mathbf{V}$ is a variance operator, $\frac{\partial\boldsymbol{\mathcal{R}}}{\partial\mathbf{B}_{k}^{i}} = \mathbf{I} - \boldsymbol{\zeta}_{k+1}\boldsymbol{\zeta}_{k+1}^{T}$ and $\frac{\partial\boldsymbol{\mathcal{R}}}{\partial y_{k+1}^{p,i}} = -\frac{\partial\boldsymbol{\mathcal{R}}}{\partial y_{k}^{p,i}} = \frac{\boldsymbol{\zeta}_{k+1}}{\|\mathbf{u}_{k+1}-\mathbf{u}_{k}\|}$. Note that the assumption is made here that the mean values of all involved quantities, $\hat{\mathbf{B}}_{k}^{i}$, $\hat{y}_{k+1}^{p,i}$, $\hat{y}_{k}^{p,i}$, $\hat{\mathbf{u}}_{k+1}$ and $\hat{\mathbf{u}}_{k}$, can be approximated by their respective values at the current iteration (i.e. $\mathbf{B}_{k}^{i}$, $y_{k+1}^{p,i}$, $y_{k}^{p,i}$, $\mathbf{u}_{k+1}$ and $\mathbf{u}_{k}$).

Next, the covariance matrix of the triplet $\left[\mathbf{B}_{k}^{i}, y_{k+1}^{p,i}, y_{k}^{p,i}\right]$ must be developed. First, the variances of the measurements themselves ($\sigma^2_{y_{k+1}^{p,i}}$ and $\sigma^2_{y_{k}^{p,i}}$) need to be estimated. This can

be done based on knowledge of the sensors that are being used to take the measurements. The variance due to the measurements is generally assumed to be constant throughout all the iterations of a given simulation. Also note the assumption is made that there is no correlation between $y^{p,i}_{k+1}$ and $y^{p,i}_{k}$, or $y^{p,i}_{k+1}$ and $\mathbf{B}^{i}_{k}$, and that the covariance between $y^{p,i}_{k}$ and $\mathbf{B}^{i}_{k}$ can be approximated as $\frac{\zeta_k}{\|\mathbf{u}_k-\mathbf{u}_{k-1}\|}\sigma^2_{y^{p,i}}$. Considering the full covariance matrices of the triplets $\left[\mathbf{B}^{i}_{k+1}, y^{p,i}_{k+1}, y^{p,i}_{k}\right]$ and $\left[\mathbf{B}^{i}_{k}, y^{p,i}_{k+1}, y^{p,i}_{k}\right]$, Equation 3.38 can be rewritten as follows:

$$\begin{bmatrix} \mathbf{V}\left(\mathbf{B}^{i}_{k+1}\right) & 0 & cov(\mathbf{B}^{i}_{k+1}, y^{p,i}_{k+1}) \\ 0 & \sigma^2_{y^{p,i}} & 0 \\ cov(\mathbf{B}^{i}_{k+1}, y^{p,i}_{k+1}) & 0 & \sigma^2_{y^{p,i}} \end{bmatrix} \approx \begin{bmatrix} \mathbf{I}-\zeta_{k+1}\zeta^T_{k+1} & 0 & 0 \\ \frac{\zeta^T_{k+1}}{\|\mathbf{u}_{k+1}-\mathbf{u}_k\|} & \mathbf{I} & 0 \\ -\frac{\zeta^T_{k+1}}{\|\mathbf{u}_{k+1}-\mathbf{u}_k\|} & 0 & \mathbf{I} \end{bmatrix}^T$$
$$\begin{bmatrix} \mathbf{V}\left(\mathbf{B}^{i}_{k}\right) & 0 & \frac{\zeta_k}{\|\mathbf{u}_k-\mathbf{u}_{k-1}\|}\sigma^2_{y^{p,i}} \\ 0 & \sigma^2_{y^{p,i}} & 0 \\ \frac{\zeta^T_k}{\|\mathbf{u}_k-\mathbf{u}_{k-1}\|}\sigma^2_{y^{p,i}} & 0 & \sigma^2_{y^{p,i}} \end{bmatrix} \begin{bmatrix} \mathbf{I}-\zeta_{k+1}\zeta^T_{k+1} & 0 & 0 \\ \frac{\zeta^T_{k+1}}{\|\mathbf{u}_{k+1}-\mathbf{u}_k\|} & \mathbf{I} & 0 \\ -\frac{\zeta^T_{k+1}}{\|\mathbf{u}_{k+1}-\mathbf{u}_k\|} & 0 & \mathbf{I} \end{bmatrix} \quad (3.39)$$

From Equation 3.39, the following first-order approximation for the variance of the Broyden estimates at iteration $k+1$ can be extracted:

$$\begin{aligned} \mathbf{V}\left(\mathbf{B}^{i}_{k+1}\right) \approx\ & \left[\mathbf{I}-\zeta_{k+1}\zeta^T_{k+1}\right]\mathbf{V}\left(\mathbf{B}^{i}_{k}\right)\left[\mathbf{I}-\zeta_{k+1}\zeta^T_{k+1}\right] + 2\sigma^2_{y^{p,i}}\frac{\zeta_{k+1}\zeta^T_{k+1}}{\|\mathbf{u}_{k+1}-\mathbf{u}_k\|^2} \\ & -\left[\mathbf{I}-\zeta_{k+1}\zeta^T_{k+1}\right]\frac{\zeta_k}{\|\mathbf{u}_k-\mathbf{u}_{k-1}\|}\sigma^2_{y^{p,i}}\frac{\zeta_{k+1}}{\|\mathbf{u}_{k+1}-\mathbf{u}_k\|} \\ & -\frac{\zeta^T_{k+1}}{\|\mathbf{u}_{k+1}-\mathbf{u}_k\|}\frac{\zeta_k}{\|\mathbf{u}_k-\mathbf{u}_{k-1}\|}\sigma^2_{y^{p,i}}\left[\mathbf{I}-\zeta_{k+1}\zeta^T_{k+1}\right] \end{aligned} \quad (3.40)$$

Examining Equation 3.40, it is clear that the variance is dependent on both the distance between the current and previous operating points ($\mathbf{u}_{k+1}$ and $\mathbf{u}_k$) and the distance between the previous two operating points ($\mathbf{u}_k$ and $\mathbf{u}_{k-1}$).

Application to the Williams-Otto Reactor Test Case

The Williams-Otto Reactor test case will be used here to illustrate the peaking problem. Measurement noise is added to the mass fraction measurements (zero mean, standard deviation of approximately 0.5% of a set of typical output mass fraction values) and modifier

adaptation with Broyden's method is run for 50 iterations. The following plot shows the true plant profit at each iteration:

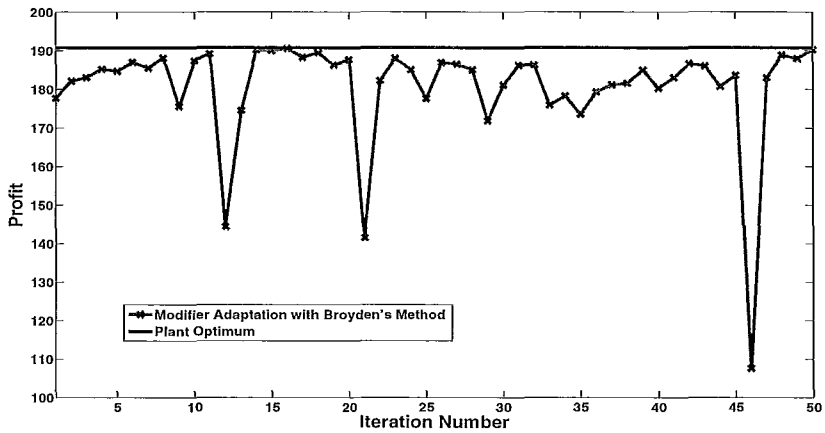

Figure 3.12: MA with Broyden's method, under the influence of measurement noise

It is easy to identify the peaking phenomenon in Figure 3.12 whenever the plant profit suddenly decreases. Since significantly large movements away from the plant optimum would clearly not be acceptable in practice, a modification will need to be made to the algorithm in order to mitigate this problem.

The dependence of the variance of the Broyden estimates on $\| \mathbf{u}_{k+1} - \mathbf{u}_k \|$ and $\| \mathbf{u}_k - \mathbf{u}_{k-1} \|$ (Equation 3.40) makes it clear that new operating points must be chosen in a clever way so as to make the distance between consecutive operating points as large as possible while still remaining close enough to the plant optimum so that good performance is achieved. To visualize the importance of the operating point choice, the following short example is given. For this example, it is assumed that the process has been running at the operated point $\mathbf{u}_k$ for some time. A new set of modifiers have been computed and it is time to choose $\mathbf{u}_{k+1}$. The goal here is to find out which operating points will in general provide the lowest variance of the Broyden derivative estimates $\mathbf{V}\left(\mathbf{B}^i_{k+1}\right)$ in the next Broyden update step.

Equation 3.40 has already been derived and can be tasked with making the estimation of $\mathbf{V}\left(\mathbf{B}^i_{k+1}\right)$ for any potential operating point. Since it is intended that this example be as general as possible, it is assumed that no information about the previous Broyden estimates

is available. In addition, it is assumed that only the current operating point ($\mathbf{u}_k$) is known. These two assumptions lead to a simplification of Equation 3.40:

$$\mathbf{V}\left(\mathbf{B}^i_{k+1}\right) \approx 2\sigma^2_{y^p,i} \frac{\zeta_{k+1}\zeta^T_{k+1}}{\| \mathbf{u}_{k+1} - \mathbf{u}_k \|^2} \tag{3.41}$$

Equation 3.41 is then applied to the Williams-Otto Reactor case study to estimate the effect of the operating point choice on the Broyden derivative estimates for the change in X_P with respect to both of the two inputs. Assuming measurement noise of a standard deviation of 5.38×10^{-4} (0.5% of a typical value for X_P), Figure 3.13 can be obtained, giving estimates of each of the four elements of $\mathbf{V}\left(\mathbf{B}^{x_p}_{k+1}\right)$ throughout the input space.

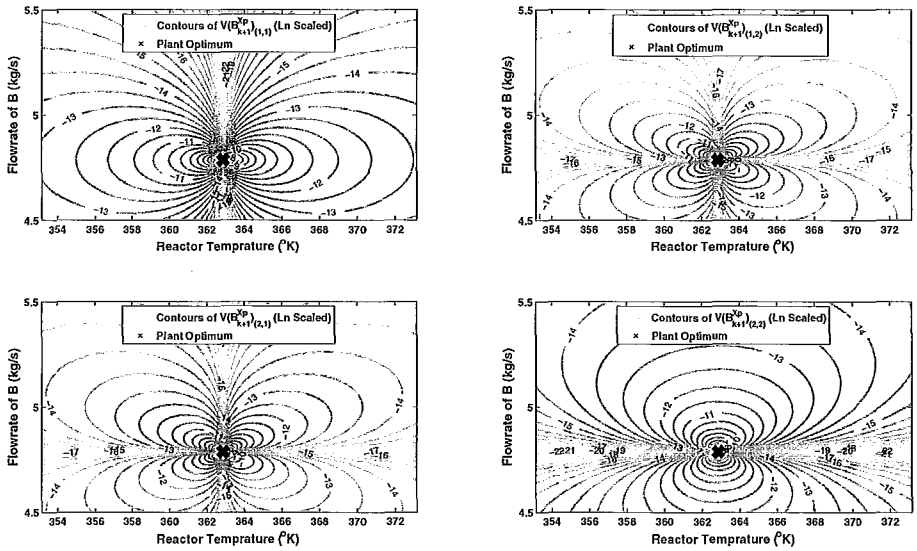

Figure 3.13: Broyden derivative estimate variance for the X_P measurement

Note that, as expected, points very close to the previous operating point, $\mathbf{u}_k$, have very high estimates for all the elements of the covariance matrix. Now, the variances of each of the diagonal terms of $\mathbf{B}^{x_p}_{k+1}$ are added together and the result is shown in Figure 3.14. Note that this neglects both the effect of the covariance of the two terms and also the fact that the variance of one of the terms of $\mathbf{B}^{x_p}_{k+1}$ might have a greater effect on the algorithm than the other.

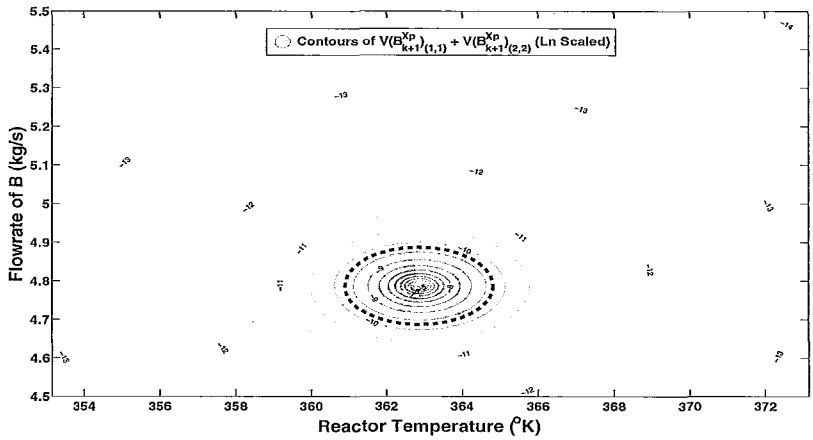

Figure 3.14: Broyden derivative estimate variance for X_P (variance terms added)

The shape of the contours in Figure 3.14 is an ellipse (this is expected given the form of Equation 3.41). It was important that the above example was as general as possible because when potential solutions are considered in Section 3.4, they must be effective in all possible situations, not just for a particular previous Broyden variance estimate or previous operating point ($\mathbf{u}_{k-1}$).

3.4 Dual Modifier Adaptation

In light of the both the gradient offset and peaking problems discussed in Section 3.2.3 and 3.3 respectively, it is clear that the algorithm for modifier adaptation with Broyden updates needs to be modified to achieve more consistently accurate plant output gradient estimates. Note that after any modification the algorithm would possess two distinct goals. The first being the original goal, to minimize the value of the cost function. The additional goal would be to achieve accurate Broyden derivative estimates, both in terms of offset and variance. This is why the new algorithm is called *dual* modifier adaptation.

In the following section, select ideas from dual control literature are examined for use with modifier adaptation. One of these ideas is adopted to address the peaking phenomenon. This idea is then extended to provide the Broyden update method with constant excitation

in all input directions, thereby helping to mitigate the offset problem as well.

3.4.1 Mitigating the Peaking Phenomenon

While there has not been much work done on dual RTO methods to this point, aside from the dual ISOPE method previously discussed in Section 2.4, the dual control problem has been studied extensively. This problem consists of attempting to achieve the best possible balance between controller performance and process knowledge when trying to control an unknown system. Essentially, the system must be perturbed in order to learn information about it so that better control can be realized in the future. An excellent overview on dual control was done in Wittenmark [1995]. Solution of the exact dual control problem, while theoretically possible, is not practical because of the computational time requirements of solving a stochastic dynamic program. Due to this, a number of approximate dual control methods have been developed over the years.

Two types of approximate methods mentioned in Wittenmark [1995] seemed to be potentially useful in modifier adaptation. The first is the addition of an extra term to the cost function to encourage movement away from the current operating point. This acts like a penalty term, which prevents new operating points from being positioned too close to the current one. A potential modified cost function is:

$$\min_{\mathbf{u}} \phi\left(\mathbf{u}, \hat{\mathbf{y}}^m\right) + \frac{1}{\left(\mathbf{u}_{k+1} - \mathbf{u}_k\right)^T \mathbf{C} \left(\mathbf{u}_{k+1} - \mathbf{u}_k\right)} \tag{3.42}$$

where the parameters in $\mathbf{C} = \mathsf{diag}(\mathbf{c})$ are chosen to achieve *good* performance. Note that there are other possible variations of this approach, for instance, adding one term to the cost function for each input:

$$\min_{\mathbf{u}} \phi\left(\mathbf{u}, \hat{\mathbf{y}}^m\right) + \sum_{i=1}^{n_u} \frac{c_i}{\left(u_i - u_{k,i}\right)^2} \tag{3.43}$$

which would allow the user to ensure movement in all input directions every iteration.

Another type of approximate method, discussed in Wittenmark [1995], is the addition of an extra constraint to the optimization problem. This could be done in a number of

different ways. One possible formulation would be to create an ellipsoid around the previous operating point:

$$(\mathbf{u} - \mathbf{u}_k)^T \boldsymbol{\mathcal{B}} (\mathbf{u} - \mathbf{u}_k) \geq 1 \tag{3.44}$$

where $\boldsymbol{\mathcal{B}}$ is a symmetric, positive definite matrix of ellipsoid size parameters. Instead of adding only a single constraint, separate ones could be included for each input:

$$(u_i - u_{k,i})^2 \geq \mu_i^2, \ \forall i = 1, .., n_u \tag{3.45}$$

where $\boldsymbol{\mu}$ are a set of tuning parameters, one for each input direction. Note that all four of the options outlined above involve introducing additional non-convexity into the optimization problem.

A check procedure, where the effect of the measurement noise on the input predictions is computed and checked against a reference value was also considered. Note that this is similar to the form of dual ISOPE suggested in Gao and Engell [2005]. In this case, if the check fails, the Broyden matrix is simply not updated during the current iteration. This approach was discarded however, because the nature of the conditional procedure made selection of a good check threshold for any given problem difficult. Trial and error, or a general rule of thumb, would likely have to have been employed instead.

Application to the Williams-Otto Reactor Test Case

In this section, the merits of each of the four dual approaches introduced in the last section are analyzed. Tests are conducted for each one of these methods by running modifier adaptation with Broyden updates and the appropriate addition to the model-based optimization problem. For each method, the Williams-Otto Reactor test case is simulated for 300 iterations, starting from a point relatively close to the plant optimum. White noise is added to the plant mass fraction measurements, with standard deviation of 0.5% of a set of typical mass fraction outputs for this system. The filter parameters (k) are all set at 0.25. Values are selected for the tuning parameters (c, $\boldsymbol{\mathcal{B}}$, or $\boldsymbol{\mu}$) in each of these methods after considerable testing. These values are chosen to achieve "good" performance of the

algorithm, however, they were by no means optimized. The results of the testing are shown in Figure 3.15.

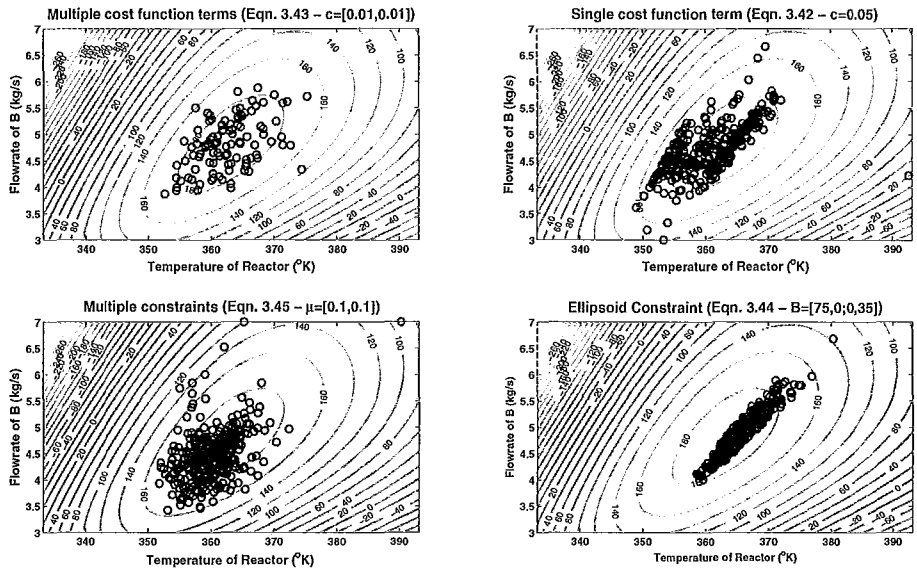

Figure 3.15: Demonstration of different dual control methods

Figure 3.15 shows that each approach appears to perform adequately for the specific set of tuning parameters chosen. No ranking of approaches is possible here, simply because no attempt has been made to optimize the respective tuning parameters of each approach. Therefore, the decision will have to be based mostly on qualitative factors. Finally, there seems to be some offset from the plant optimum present in each of the simulations in the figure. This deficiency is ignored for now, however, as none of the methods described earlier in this section have the ability to combat this. This issue will instead be addressed later in this chapter.

Note that the simulation employing the approach of adding multiple terms to the cost function, Equation 3.43, is shown in Figure 3.15 for only a run of 100 iterations. The reasoning for the reduced run length was the MATLAB optimizer, *fmincon*, frequently stalled during testing, so it was difficult to string together many successful iterations. The specific reason for the difficulty was not discovered, however this method was discarded because of this problem.

The algorithm which involved the addition of a single term to the cost function (Equation 3.42), performed much better in that there were never any problems with the MATLAB optimizer over the course of many simulation runs. The problem with cost function approaches in general, however, is that it is more difficult to devise a systematic procedure for choosing the tuning parameters. This is because it is hard to estimate the effect of the penalty term on the algorithm. Conversely, in the case of the constraint approaches, the restricted area itself can be used to approximate the effect of the constraint since its boundary naturally represents the maximum influence that it can have on a particular operating point selection.

For the approach involving multiple constraints (Equation 3.45), the model-based optimization problem had to be solved using multiple starting points. This was because of the fact that if the optimizer started in a specific feasible quadrant, it tended not to exit that particular quadrant, due to the non-convexity introduced by the extra constraints. To combat this, 4 starting points were selected from inside the input space for each optimization stage. These four points were the corners of the rectangular area enclosed by the variable bounds $\mathbf{u}^{min}$ and $\mathbf{u}^{max}$. The extra effort required to solve three additional optimization problems did not pose a problem for this relatively small two-input problem. However, since the number of separate areas would grow exponentially with the number of inputs, this method could become quite computationally expensive for larger problems.

In the remainder of this thesis, the emphasis is on the ellipsoid constraint method. This method is selected for a couple of reasons. First of all, it does not have any of the major drawbacks that each of the other three approaches have. There are no problems with the optimizer, as the algorithm runs smoothly and produces acceptable results. It also consists of a geometric constraint, making it easier to tune than the single term cost function approach, and it does not produce a model-based optimization problem that requires solutions from multiple starting points to be obtained. Also, recall the variance approximation study of Figure 3.14. The shape of the contours nominally resembled an ellipse, indicating that this might be a good shape to use for a constraint because the variance could be consistently kept below a certain level.

The ellipsoid parameters, denoted by $\mathbf{b}$, are assumed to be on the diagonal of the matrix $\mathcal{B}$ subsequently. In principle $\mathcal{B}$ does not have to be diagonal, it only needs to be symmetric and positive-definite. However, for the purpose of this thesis, the assumption is made that $\mathcal{B}$ is diagonal. A future research direction would be to extend the design procedure in Chapter 4 to address all $\mathcal{B}$ matrices that are symmetric and positive definite.

The effect of the introduction of the ellipsoid constraint on the performance of the algorithm is illustrated in Figure 3.16. A simulation consisting of 50 iterations, utilizing the same general settings that were used in creating Figure 3.12 was run. Ellipsoid parameters of ($\mathbf{b} = [40, 30]$) were chosen. These were chosen after a period of testing because they appeared to help achieve "good" algorithm performance. The improvement in performance is clear, as the large variations in the plant profit are eliminated.

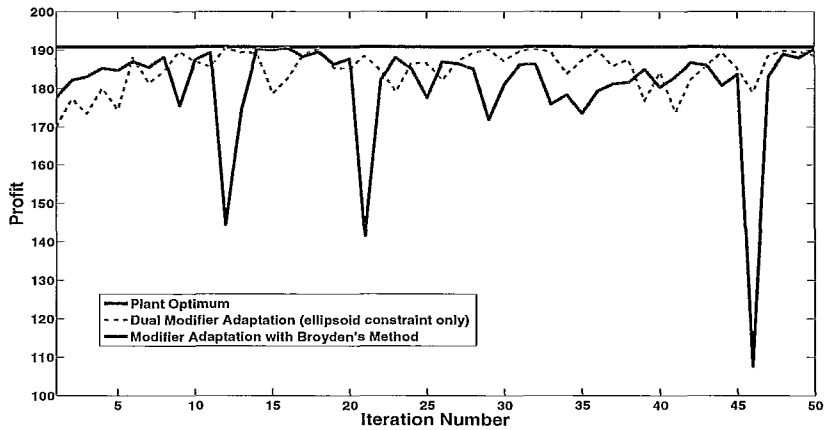

Figure 3.16: Dual modifier adaptation - ellipsoid constraint only

3.4.2 Mitigating the Gradient Offset Problem

The causes of the gradient offset problem are discussed in detail in Section 3.2.3. It is the cause of the bias from the plant optimum seen in the simulations of the previous section (Figure 3.15).

To reduce the effect of the gradient bias, two different approaches are possible (see Section 3.2.3). First, the step length (distance between consecutive operating points) can be re-

duced. This can be easily done by employing a trust region constraint which creates an area around the current operating point inside which the new operating point must be placed. This constraint can be written as follows:

$$[\mathbf{u} - \mathbf{u}_k]^T \mathcal{T} [\mathbf{u} - \mathbf{u}_k] \leq 1 \tag{3.46}$$

where $\mathcal{T}$ is a symmetric positive definite matrix of ellipsoid parameters which define the trust region. Note that since the trust region constraint and the ellipsoid constraint (Equation 3.44) have conflicting objectives, they must be designed in a consistent manner so that they do not cause the model-based optimization problem to be infeasible. In some extreme cases, in which the level of measurement noise is very high, it might not be possible to find an operating point at which both the offset and variance of the Broyden derivative estimates will be acceptable.

The second method of eliminating offset consists of ensuring, in some way, that the algorithm explores a variety of possible input directions. It is assumed here that an ellipsoid has already been sized offline based on the properties of the RTO problem to be solved. These include the cost function, constraints and model themselves as well as the level of expected measurement noise. A systematic design procedure is developed for this in Chapter 4. Suffice it to say that it is already sized at run-time, and this size is available.

For this derivation the assumption is made that there are n_u previous operating points available, which represent a set of $n_u - 1$ previous movement directions $\boldsymbol{\Delta}_k, ..., \boldsymbol{\Delta}_{k-n_u+1}$ (where $\boldsymbol{\Delta}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$). If n_u previous operating points are not available (i.e. if it is very early in the RTO execution period), the ellipsoid constraint itself can simply be used without considering the offset (as in the lower right panel of Figure 3.15). There is also the option of generating the missing points by conducting plant experiments.

Let $\mathbf{H}_k = \text{span}\,(\boldsymbol{\Delta}_k, ..., \boldsymbol{\Delta}_{k-n_u+1})$ denote a hyperplane which is defined by the $n_u - 1$ previous algorithm movement directions. Since the goal here is to ensure sufficient exploration of the input space, any offset elimination procedure should ensure that the algorithm moves off of this hyper-plane. One natural way to impart this goal on the dual modifier adaptation algorithm would be to build upon the existing ellipsoidal restriction region constraint that

was developed in Section 3.4.1. In such a case, the constraint(s) would have two distinct goals: moving the new iterate a significant distance away from $\mathbf{u}_k$ and moving a significant distance away from the hyper-plane $\mathbf{H}_k$. Figure 3.17 shows a pair of constraints which accomplish these goals. The advantage of such a design would be that the constraints would create only one exclusion area, which would be easier to handle for both the model-based optimizer and the RTO design procedure (see Chapter 4).

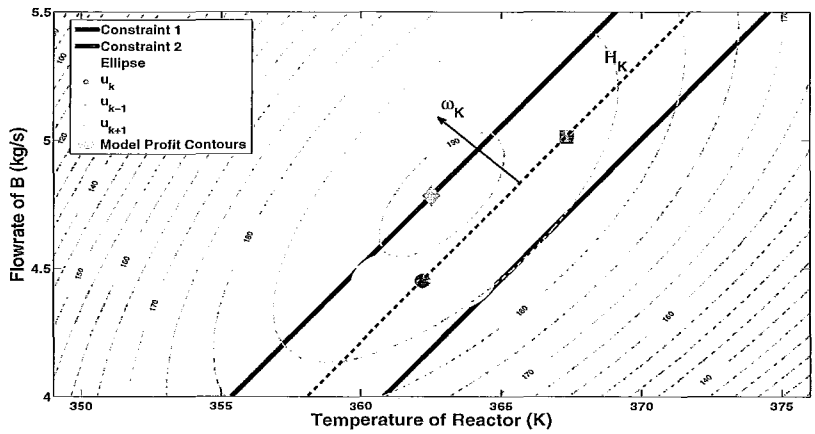

Figure 3.17: Illustration of the dual constraints

There are already a set of tuning parameters that need to be chosen ($\mathcal{B}$) in order to size the ellipsoidal restriction constraint itself. In the interest of design simplicity, it would be advantageous to utilize the same design parameters in satisfying the requirement of movement off of the hyper-plane. The following disjunctive expression does exactly this:

$$\bigvee \left\{ \begin{array}{l} \omega_k^T [\mathbf{u} - \mathbf{u}_k] \geq \sqrt{\omega_k^T \mathcal{B}^{-1} \omega_k} \\ -\omega_k^T [\mathbf{u} - \mathbf{u}_k] \geq \sqrt{\omega_k^T \mathcal{B}^{-1} \omega_k} \end{array} \right\} \tag{3.47}$$

where ω_k is a vector orthogonal to the hyperplane $\mathbf{H}_k$. One way to choose ω_k is as the first row of the adjugate of the matrix of previous movement directions:

$$\mathbf{U}_k = \left[\mathbf{u} - \mathbf{u}_k \quad \Delta_k \quad \Delta_{k-1} \quad \ldots \quad \Delta_{k-n_u+1} \right] \tag{3.48}$$

where $\Delta_k = \mathbf{u}_k - \mathbf{u}_{k-1}$. Even though the new operating point, $\mathbf{u}$, appears here, the first row of the adjugate of $\mathbf{U}_k$ is independent of the new operating point.

Figure 3.17 illustrates the constraints (3.47) and corresponding operating point decision which result from running the dual modifier adaptation algorithm for the Williams-Otto Reactor test case. Note that the contours shown here are for the process model at this specific iteration of the algorithm. This ensures that the contours reflect the operating point choice made in the figure.

Now an explanation of how the constraints of Equation 3.47 result in the construction that is illustrated in Figure 3.17 is given. Note first that the $\sqrt{\omega_k^T \mathcal{B}^{-1} \omega_k}$ term in Equation 3.47 must be positive. This is due to the fact that the matrix $\mathcal{B}$ is positive definite and therefore its inverse is also positive definite. If $[\mathbf{u} - \mathbf{u}_k]$ is on $\mathbf{H}_k$, then both the left sides of the disjunctive constraints in Equation 3.47 will be equal to zero (recall that ω_k is orthogonal to $\mathbf{H}_k$) and neither constraint will be satisfied. Therefore, the constraints of (3.47) ensure that $[\mathbf{u} - \mathbf{u}_k]$ is not on the hyperplane.

This is a good first step, however, the fact that the constraints in Equation 3.47 reside on or outside of the ellipsoid defined by $\mathcal{B}$ must still be established if the constraints (3.47) are to replace the ellipsoid constraint (3.44). The first disjunctive constraint of (3.47) can be interpreted as meaning that the projection of $[\mathbf{u} - \mathbf{u}_k]$ in the direction ω_k (orthogonal to the hyperplane $\mathbf{H}_k$) must be larger than a constant based on the size of the ellipsoid and the length of ω_k. Since the constraint is linear in terms of the inputs ($\mathbf{u}$), its contours can be represented by a series of hyper-planes. In addition, since the constraint represents a projection in a direction orthogonal to $\mathbf{H}_k$, these hyper-planes will be parallel to $\mathbf{H}_k$.

Now, consider the following operating point change:

$$[\mathbf{u} - \mathbf{u}_k] = \frac{\mathcal{B}^{-1}\omega_k}{\sqrt{\omega_k^T \mathcal{B}^{-1} \omega_k}} \tag{3.49}$$

Note that at this point, both the first disjunctive constraint of (3.47) and the ellipsoid constraint (3.44) are active. If it can be proved that at this point the first disjunctive constraint represents a hyper-plane tangent to the ellipsoid, then this is sufficient because a tangent hyper-plane is by nature an outer-approximation of the ellipsoid.

To prove that the point given in (3.49) is indeed a point where the disjunctive constraint is

tangent to the ellipsoid, the gradients of both are examined. The gradient of the ellipsoid constraint can be computed as follows:

$$\frac{d}{d\mathbf{u}}\left(\left[\mathbf{u}-\mathbf{u}_k\right]^T \boldsymbol{\mathcal{B}}\left[\mathbf{u}-\mathbf{u}_k\right]-1\right)=2\boldsymbol{\mathcal{B}}\left[\mathbf{u}-\mathbf{u}_k\right] \tag{3.50}$$

and when evaluated at $[\mathbf{u}-\mathbf{u}_k]=\frac{\boldsymbol{\mathcal{B}}^{-1}\omega_k}{\sqrt{\omega_k^T\boldsymbol{\mathcal{B}}^{-1}\omega_k}}$ it becomes: $2\frac{\omega_k}{\sqrt{\omega_k^T\boldsymbol{\mathcal{B}}^{-1}\omega_k}}$ On the other hand, the gradient of the disjunctive constraint $\frac{\omega_k^T[\mathbf{u}-\mathbf{u}_k]}{\sqrt{\omega_k^T\boldsymbol{\mathcal{B}}^{-1}\omega_k}}$ is simply $\frac{\omega_k}{\sqrt{\omega_k^T\boldsymbol{\mathcal{B}}^{-1}\omega_k}}$. Since these two gradients have the same direction, the point described in (3.49) must be tangent to the ellipsoid. Therefore, satisfaction of the first disjunctive constraint guarantees satisfaction of the ellipsoid constraint.

In the left panel of Figure 3.18 the contours of the first disjunctive constraint $\omega_k^T\left[\mathbf{u}-\mathbf{u}_k\right]-\sqrt{\omega_k^T\boldsymbol{\mathcal{B}}^{-1}\omega_k}\geq 0$ and the contours of the ellipsoid constraint $\left(\mathbf{u}-\mathbf{u}_k\right)^T\boldsymbol{\mathcal{B}}\left(\mathbf{u}-\mathbf{u}_k\right)-1\geq 0$ are illustrated for a fictitious system. The tangency point, given in Equation 3.49 as $\mathbf{u}$, is also shown in the figure as $\mathbf{u}_k^{t+}$.

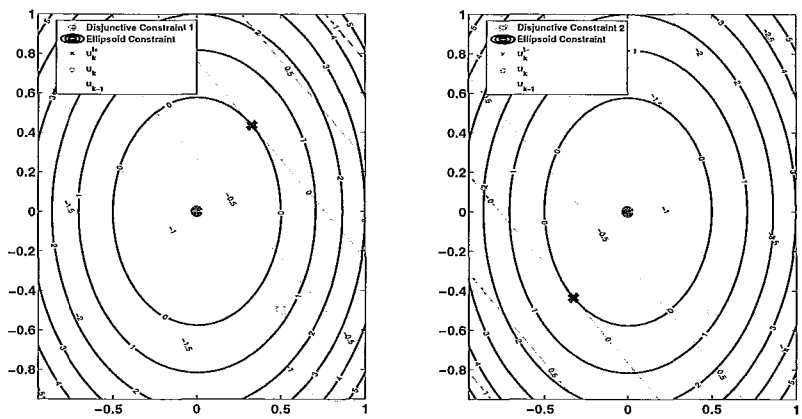

Figure 3.18: Illustration of the ellipsoid and disjunctive constraint contours

A similar argument can be made for the second disjunctive constraint of (3.47). This time, the point where the constraint is tangent to the ellipsoid is $\frac{-\boldsymbol{\mathcal{B}}^{-1}\omega_k}{\sqrt{\omega_k^T\boldsymbol{\mathcal{B}}^{-1}\omega_k}}$. The contours of the second disjunctive constraint $-\omega_k^T\left[\mathbf{u}-\mathbf{u}_k\right]-\sqrt{\omega_k^T\boldsymbol{\mathcal{B}}^{-1}\omega_k}\geq 0$ and the ellipsoid constraint $\left(\mathbf{u}-\mathbf{u}_k\right)^T\boldsymbol{\mathcal{B}}\left(\mathbf{u}-\mathbf{u}_k\right)-1\geq 0$ are illustrated in the right panel of Figure 3.18. The tangency point is represented here as $\mathbf{u}_k^{t-}$.

In practice, due to the disjunctive nature of the constraints, two different optimization problems must be solved at each iteration. These two problems are given as Problems 3.51 and 3.52 below:

$$\begin{aligned} \mathbf{u}_{k+1}^{+} \in \arg\min_{\mathbf{u}} \quad & \phi\left(\mathbf{u}, \hat{\mathbf{y}}^{m}\right) \\ \text{s.t.} \quad & \hat{\mathbf{y}}^{m} = \mathbf{f}\left(\mathbf{u}, \boldsymbol{\beta}\right) + \bar{\boldsymbol{\epsilon}}_{k} + \boldsymbol{\lambda}_{k}^{T}\mathbf{u} \\ & \mathbf{g}\left(\mathbf{u}, \hat{\mathbf{y}}^{m}\right) \leq \mathbf{0} \\ & \left[\mathbf{u} - \mathbf{u}_{k}\right]^{T} \boldsymbol{\mathcal{T}} \left[\mathbf{u} - \mathbf{u}_{k}\right] \leq 1 \\ & \boldsymbol{\omega}_{k}^{T} \left[\mathbf{u} - \mathbf{u}_{k}\right] \geq \sqrt{\boldsymbol{\omega}_{k}^{T} \boldsymbol{\mathcal{B}}^{-1} \boldsymbol{\omega}_{k}} \\ & \mathbf{u}^{min} \leq \mathbf{u} \leq \mathbf{u}^{max} \end{aligned} \tag{3.51}$$

$$\begin{aligned} \mathbf{u}_{k+1}^{-} \in \arg\min_{\mathbf{u}} \quad & \phi\left(\mathbf{u}, \hat{\mathbf{y}}^{m}\right) \\ \text{s.t.} \quad & \hat{\mathbf{y}}^{m} = \mathbf{f}\left(\mathbf{u}, \boldsymbol{\beta}\right) + \bar{\boldsymbol{\epsilon}}_{k} + \boldsymbol{\lambda}_{k}^{T}\mathbf{u} \\ & \mathbf{g}\left(\mathbf{u}, \hat{\mathbf{y}}^{m}\right) \leq \mathbf{0} \\ & \left[\mathbf{u} - \mathbf{u}_{k}\right]^{T} \boldsymbol{\mathcal{T}} \left[\mathbf{u} - \mathbf{u}_{k}\right] \leq 1 \\ & -\boldsymbol{\omega}_{k}^{T} \left[\mathbf{u} - \mathbf{u}_{k}\right] \geq \sqrt{\boldsymbol{\omega}_{k}^{T} \boldsymbol{\mathcal{B}}^{-1} \boldsymbol{\omega}_{k}} \\ & \mathbf{u}^{min} \leq \mathbf{u} \leq \mathbf{u}^{max} \end{aligned} \tag{3.52}$$

The set of inputs generated from these problems ($\mathbf{u}_{k+1}^{+}$ or $\mathbf{u}_{k+1}^{-}$) with the lowest objective function value is then adopted as the next operating point, $\mathbf{u}_{k+1}$.

The algorithm employing Problems 3.51 and 3.52 as well as the Broyden update map, $\boldsymbol{\mathcal{R}}$ (see Figure 3.1), the modifier update law map, $\boldsymbol{\mathcal{A}}$ (3.10), and the alternate modifier conversion map, $\mathbf{T}$ (3.5), will be referred to as dual modifier adaptation for the rest of this thesis. In addition, the disjunctive constraints (3.47) added to the original model-based optimization problem to produce Problems 3.51 and 3.52 are referred to as the dual constraints. To complement the algorithm a systemic constraint design procedure is required, which will be discussed in the next chapter.

Application to the Williams-Otto Reactor Test Case

The algorithm involving the disjunctive constraints is now tested by running 300 iterations of the Williams-Otto Reactor test case with filter parameters of 0.25, ellipsoid parameters of $\mathbf{b} = [40, 30]$ and trust region constraint parameters of $\mathcal{T} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Measurement noise with a standard deviation of 0.5% of a nominal set of mass fraction values is assumed. The results are shown in Figure 3.19.

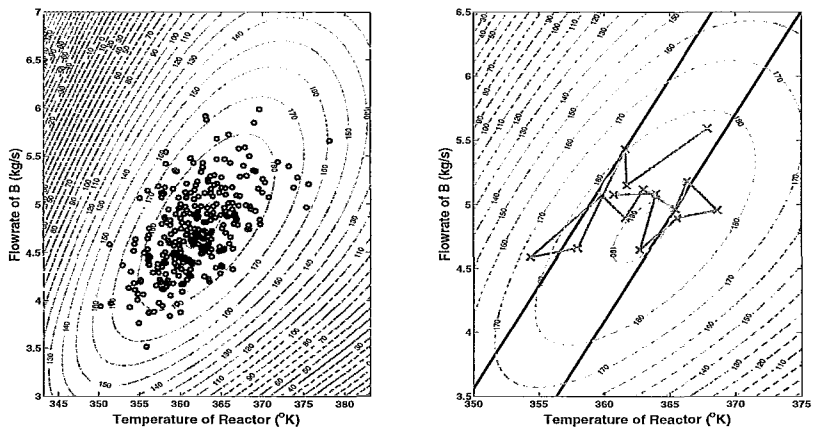

Figure 3.19: Performance of the dual constraints

It appears that the offset has been for the most part eliminated in the left plot of Figure 3.19 (compared with the bottom right plot of Figure 3.15). Examining the right panel, it seems that all directions in the input space are being explored. Figure 3.20 is also included to demonstrate how the algorithm using the disjunctive constraints responds to a disturbance. The scenario considered here is similar to the one considered in Figure 2.8. The only difference is the size of the unmeasured disturbance was increased so that the operating point change could be distinguished from the noise in the input estimates. The same parameter and noise settings that were applied in the simulation for Figure 3.19 were applied again here. The performance of the algorithm in Figure 3.19 is satisfactory, as it seems to move fairly quickly to the new plant optimum. It is noisy though, especially compared to Figure 2.8, which motivates the development of the systematic tuning approach discussed in the next chapter.

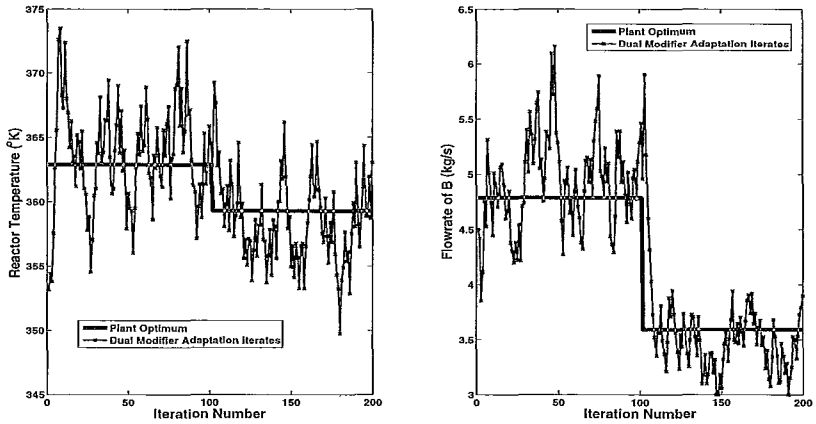

Figure 3.20: Step change test for dual constraints

3.5 Chapter Summary

In this chapter, Broyden's method was selected to estimate the plant output gradient in the modifier adaptation algorithm. The importance of several choices in regard to the new algorithm was illustrated though a series of simulations using the Williams-Otto Reactor test case. These choices included the starting point, the process model and the filter parameters. A convergence analysis of the algorithm for a single input formulation of the Williams-Otto Reactor test case was then carried out. This analysis proved to be helpful in identifying a good range of filter parameters for the problem.

The second half of the chapter detailed the development of the dual modifier adaptation algorithm. First, several difficulties in regard to the implementation of Broyden's method were discussed in detail, including offset in the gradient estimates and the peaking phenomenon. Ideas from dual control theory were then used to come up with a solution to the peaking phenomenon and this solution was then modified so that it could address the gradient offset problem as well. It consisted primarily of the addition of two disjunctive constraints to the model-based optimization problem, which restricted operating point placement. The new algorithm performed well on the Williams-Otto Reactor test case.

Chapter 4

Offline Design of Dual Modifier Adaptation

The implementation of an RTO system alone is not enough to guarantee process improvement. The system must also be well designed, based on the nature of the specific process in question, in order to achieve the greatest possible financial benefit. Proper design is especially important for the dual modifier adaptation algorithm introduced in Chapter 3. Both the trust region and dual constraints must be properly sized on their own, and also designed together in a consistent manner in order to avoid poor performance or in some cases even infeasibility.

The optimal design policy involves the selection of new values for the dual constraint parameters (and any other pertinent design parameters) as the algorithm is running in real time. This could be done as frequently as before every iteration. The problem with this idea in practice is the necessary computation time. As the size of the RTO problem increases, the computational effort required to solve the corresponding design problem would increase as well, likely in a dramatic manner. Therefore, for most processes, solving a design problem online would be computationally demanding, because the solution time is required to be shorter than the period between consecutive RTO iterations. In addition to the added

computation time, the complexity of this task would likely limit its acceptance in the process systems engineering community. It is for these reasons that an offline design approach is considered in this thesis. Online design would make for an interesting future research direction, especially if the computing power available continues to rapidly increase.

The main focus of Chapter 4 is therefore the offline design of dual modifier adaptation systems. The design procedure is based around the use of an adapted form of the design cost criterion, which was originally developed by Forbes and Marlin for the two-step approach of RTO [Forbes and Marlin, 1996]. The use of the criterion in the design of dual modifier adaptation systems is investigated first. Details on the calculation of this metric are given next. As previously, the developments of this chapter are then illustrated using the Williams-Otto Reactor test case.

4.1 Dual Modifier Adaptation Design Procedure

In general, the design cost criterion facilitates the comparison of different RTO design and technology options. The criterion is a quantitative estimate of the profit lost or additional cost incurred due to imperfections in the design of the RTO system. The use of this metric to arrive at the best possible dual modifier adaptation design is what is discussed in this section.

The basic design methodology is detailed first, in which both the sizing of the dual constraint as well as other design options are considered. These other design options can include the set of modifiers to be updated by the algorithm, the model to use and which measurements to take. Next, a multi-scenario optimization formulation is presented, which is designed to handle inaccuracies in the benchmark plant models and economic data used in the design procedure. This formulation can also handle the situation where the process is commonly operated at multiple different points as well. These concepts are then illustrated though an example involving the Williams-Otto Reactor test case.

4.1.1 RTO Design using the Design Cost Criterion

Forbes and Marlin [1996] formulated the design cost optimization problem as follows:

$$\min_{\varsigma} C(\varsigma) \tag{4.1}$$

where ς are a set of design choices and C is the design cost (the calculation of C is discussed in Sections 4.2 and 4.3). The selection of the optimal ellipsoid parameters, $\mathcal{B}$ (3.47), can easily be incorporated:

$$\min_{\varsigma,\mathcal{B}} C(\varsigma, \mathcal{B}) \tag{4.2}$$

Note that the selection of the trust region constraint parameters is not considered here (Equation 3.46). The reason for this is the inherent difficulty in estimating the effect of the trust region constraint on the model-based optimization problem. This problem stems from the fact that the trust region constraint will mainly be active as the system moves to react to fundamental changes in the state of the system (transients). The design cost approach, however, is concerned only with steady-state performance. Since it was not considered in the design procedure, the constraint was also not implemented in any of the simulations of this chapter. Incorporation of the trust region constraint into the design procedure could potentially be the subject of future research.

Most of the decisions that the design cost criterion has been previously applied to have been discrete in nature. For instance, previous published applications include model selection [Forbes and Marlin, 1996], adjustable parameter selection [Zhang and Forbes, 2000] and sensor selection and placement [Fraleigh *et al.*, 2003]. The dual constraint parameters are different however, because they can take any values as long as $\mathcal{B}$ is positive definite. Due to this property, the design cost problem to optimize $\mathcal{B}$ is a traditional NLP (instead of an MINLP). Realizing this, the following nested optimization formulation is given to represent the proposed method of solution:

$$\min_{\varsigma} \left(\min_{\mathcal{B}} C(\varsigma, \mathcal{B}) \right) \tag{4.3}$$

For this formulation, the intention is that the best set of dual constraint parameters are solved for at each unique combination of the discrete design variables, ς. Therefore, for

the problems addressed in this thesis, the inner optimization problem of (4.3) is solved using an NLP solver and the outer optimization problem is solved by enumeration. On the other hand, if there were many discrete design decisions, ς, to be made, Problem 4.2 could be addressed directly by employing an MINLP solution method, in order to potentially decrease the computational load.

4.1.2 Sources of Uncertainty in Dual Modifier Adaptation Systems

It is widely recognized that uncertainty plays an important role in RTO problems. Therefore, it is imperative that it be considered in the design phase of any new RTO system. A review of uncertainty specific to RTO systems was done in Zhang *et al.* [2002]. In this work, the authors consider four different types of uncertainty: model uncertainty, measurement uncertainty, process uncertainty and market uncertainty. It is useful to consider each of these types of uncertainty individually here to examine how they effect dual modifier adaptation systems and also how they will be dealt with in the dual modifier adaptation design procedure.

- Model uncertainty stems from the use of a process model which fails to represent plant operation. This can be either structural plant-model mismatch (discussed earlier) or parametric model mismatch, in which values of the parameters in the process model are inaccurate. This type of uncertainty is dealt with naturally through the modifiers used to alter the outputs in modifier adaptation.

- Measurement uncertainty occurs due to errors in the sensor system. It is assumed in this thesis that random measurement noise has a zero offset and some variance: $\sigma^2_{\mathbf{y}^p}$. This variance can typically be estimated based on factors such as the type, quality and age of the physical sensors being used. It is assumed here that gross errors have been eliminated by the data validation system.

  Measurement noise propagates through to the modifiers in several ways. It can have an effect during not only the RTO iteration in which it occurred, but also in future RTO iterations due to the first-order exponential filter and the Broyden update (which uses

the past two sets of plant measurements as well as the previous Broyden estimates). The effect of measurement noise is dealt with directly in the design cost metric itself.

- Process uncertainty is caused by disturbances in the plant [Zhang *et al.*, 2002]. These disturbances can include actual changes in the nature of the process (such as catalyst decay), variations in upstream conditions (such as reactant flowrates) and changes in surrounding conditions (for instance the outdoor temperature affecting a cooling water inlet temperature). This uncertainty manifests itself in the design procedure in the incorrect approximation of certain parameters in the benchmark plant model. This essentially means that the location of the true plant optimum is not known. This uncertainty is dealt with by using a scenario averaging approach along with the design cost criterion (Section 4.1.3).

- Market uncertainty is caused by the fact that market conditions (supply and demand) are constantly changing due to global events. This means that the sale and purchase prices of various process inputs and outputs may not be accurately known. This uncertainty can be dealt with in the same way as process uncertainty, using the multi-scenario approach of Section 4.1.3.

There is an additional source of variation considered in the design cost problem in this thesis that is unique to dual modifier adaptation. Recall that the dual constraint defines a restricted area around the previous operating point where the next operating point cannot be placed. The effect this constraint has on the solution to the model-based optimization problem varies from iteration to iteration. For the purpose of the design cost calculation, the effect of the dual constraint will be approximated and incorporated into the calculation procedure as a sort of endogenous disturbance, in this work called *dual excitation.*

4.1.3 Handling Process and Market Uncertainty

Often, there is uncertainty in the benchmark plant model upon which the RTO design is being based. After all, this is only an approximate representation of plant behaviour

in the first place. Fortunately, this uncertainty can be mitigated somewhat through the use of a scenario averaging approach. The approach that is adopted here requires that the uncertainty can be accounted for by varying the plant parameters. If the value of a parameter is uncertain, a reasonable range of values for this parameter must be known. Note that due to this requirement, structural mismatch between the plant and benchmark model cannot be accounted for by this approach.

A similar multiple-scenario approach, upon which the following approach is based, was developed in Forbes and Marlin [1996] for use with the two-step RTO approach. It considered the situation where there are multiple common plant operating modes. Therefore, this can also be dealt with using Equation 4.4 (given below) for modifier adaptation with Broyden updates, as long as the change in operating mode can be reflected by a change in the benchmark model parameters. Alternatively, the design cost procedure could be run individually for each operating mode, so as to avoid the potential profit loss incurred from adopting a "universal" design.

If α sets of uncertain plant parameter values (β^p) are sampled from the space of potential parameter values, then the optimization problem 4.3 can be recast:

$$C_{avg} = \frac{1}{\alpha}\left[\min_{\varsigma}\left(\min_{\mathcal{B}}\sum_{i=1}^{\alpha} C\left(\varsigma, \mathcal{B}, \beta_i^p\right)\right)\right] \tag{4.4}$$

where C_{avg} is the average design cost value for the scenarios considered. The $\frac{1}{\alpha}$ term is included so that the equation will give a plausible estimate for the average design cost at all the possible plant optima. Note that this approach can also handle uncertainty in the cost and constraint functions. This includes market uncertainty in purchase and sale prices in the cost function.

This extension increases the computing power required because it necessitates the solution of a more complicated design cost problem in order to identify the optimal dual constraint parameters. However, since all of the design cost computations are done offline this should not be a major problem. Note that Equation 4.4 can easily be extended to include the weighting of particular vectors of changing plant parameters, β^p, if one set of parameters is known to be more likely than another. Now that the use of the design cost criterion in

a dual modifier adaptation design procedure has been detailed, computation details for the design cost, C, will be discussed in the next section.

4.2 Design Cost Background

This section introduces the design cost criterion, which is a metric that can be used in the design of dual modifier adaptation systems, as described in Section 4.1. The criterion was first developed for the two-step method in Forbes and Marlin [1996]. For each proposed design, a total cost value, C, is calculated. This represents the extra cost (or loss of profit) incurred due to imperfections in the design and operation of the RTO system:

$$C = \mathsf{E}\left[\Phi^p(\mathbf{u}_\infty)\right] - \Phi^p(\mathbf{u}^{p,*}) \tag{4.5}$$

where $\mathbf{u}^{p,*}$ is the (unknown) plant optimum, $\mathbf{u}_\infty$ is the distribution of operating points, $\Phi^p(\mathbf{u}) := \phi(\mathbf{u}, \mathbf{F}(\mathbf{u}))$ is the plant cost function written only in terms of the inputs and E is an expectation operator. It is necessary to consider a distribution of operating points in Equation 4.5 because various sources of uncertainty (measurement noise, dual excitation) will cause the operating point computed by the RTO algorithm to change from iteration to iteration. These sources of uncertainty were discussed in Section 4.1.2.

In Equation 4.5, $\Phi^p(\mathbf{u}^{p,*})$ is computed from estimates of $\mathbf{u}^{p,*}$ and Φ^p. The other quantity in Equation 4.5 that needs to be determined is $\mathsf{E}\left[\Phi^p(\mathbf{u}_\infty)\right]$. The probability density function for the distribution $(\mathbf{u}_\infty)$ can be expressed as $\mathbf{h}(\vartheta)$, where ϑ is a specific instance of $\mathbf{u}_\infty$. Therefore the average value of the cost function for the iterates $(\mathbf{u}_\infty)$ can be approximated as follows [Forbes and Marlin, 1996]:

$$\mathsf{E}\left[\Phi^p(\mathbf{u}_\infty)\right] = \int_\chi \Phi^p(\vartheta)\mathbf{h}(\vartheta)\, d\vartheta \tag{4.6}$$

where χ denotes the space of potential operating points which the distribution $\mathbf{u}_\infty$ draws from. This is often defined in practice by $\mathbf{u}^{min}$ and $\mathbf{u}^{max}$. The next couple of steps of this derivation differ slightly from the derivation given in Forbes and Marlin [1996]. The major difference is that the derivation in Forbes and Marlin [1996] is made using reduced properties, where the one applied in this thesis is not.

The cost at ϑ can be approximated by the following second order Taylor series expansion:

$$\Phi^p(\vartheta) = \Phi^p(\mathbf{u}^{p,*}) + \nabla\Phi^p|_{\mathbf{u}^{p,*}}(\vartheta - \mathbf{u}^{p,*}) + \frac{1}{2}(\vartheta - \mathbf{u}^{p,*})^T \nabla^2\Phi^p|_{\mathbf{u}^{p,*}}(\vartheta - \mathbf{u}^{p,*}) + o\left(\|\vartheta - \mathbf{u}^{p,*}\|^3\right) \tag{4.7}$$

where $\nabla\Phi^p$ and $\nabla^2\Phi^p$ respectively denote the gradient and Hessian of the plant cost function. In writing Equation 4.7, the assumption is made that the profit function is at least twice continuously differentiable with respect to the set-points ($\mathbf{u}$).

Now, the expected value of the iterates, $\mathsf{E}[\mathbf{u}_\infty]$, is added and subtracted from each $(\vartheta - \mathbf{u}^{p,*})$ term of Equation 4.7 and the result rearranged to yield:

$$\begin{aligned}
\Phi^p(\vartheta) &= \Phi^p(\mathbf{u}^{p,*}) + \nabla\Phi^p|_{\mathbf{u}^{p,*}}(\mathsf{E}[\mathbf{u}_\infty] - \mathbf{u}^{p,*}) + \nabla\Phi^p|_{\mathbf{u}^{p,*}}(\vartheta - \mathsf{E}[\mathbf{u}_\infty]) \\
&+ \frac{1}{2}(\vartheta - \mathsf{E}[\mathbf{u}_\infty])^T \nabla^2\Phi^p|_{\mathbf{u}^{p,*}}(\vartheta - \mathsf{E}[\mathbf{u}_\infty]) \\
&+ \frac{1}{2}(\mathsf{E}[\mathbf{u}_\infty] - \mathbf{u}^{p,*})^T \nabla^2\Phi^p|_{\mathbf{u}^{p,*}}(\mathsf{E}[\mathbf{u}_\infty] - \mathbf{u}^{p,*}) \\
&+ (\mathsf{E}[\mathbf{u}_\infty] - \mathbf{u}^{p,*})^T \nabla^2\Phi^p|_{\mathbf{u}^{p,*}}(\vartheta - \mathsf{E}[\mathbf{u}_\infty]) + o\left(\|\vartheta - \mathbf{u}^{p,*}\|^3\right)
\end{aligned} \tag{4.8}$$

The Taylor series expansion (Equation 4.8) can then be manipulated in order to sub-divide the overall design cost into two separate parts. These two separate parts are the bias cost (C_B) and the variance cost (C_V):

$$C = C_B + C_V \tag{4.9}$$

- The bias cost, C_B, represents the performance loss due to offset from the plant optimum:

$$\begin{aligned}
C_B &= \Phi^p(\mathsf{E}[\mathbf{u}_\infty]) - \Phi^p(\mathbf{u}^{p,*}) \\
&= \nabla\Phi^p|_{\mathbf{u}^{p,*}}(\mathsf{E}[\mathbf{u}_\infty] - \mathbf{u}^{p,*}) \\
&+ \frac{1}{2}(\mathsf{E}[\mathbf{u}_\infty] - \mathbf{u}^{p,*})^T \nabla^2\Phi^p|_{\mathbf{u}^{p,*}}(\mathsf{E}[\mathbf{u}_\infty] - \mathbf{u}^{p,*})
\end{aligned} \tag{4.10}$$

  Note that the second line is made up of the terms from Equation 4.8 which did not depend on ϑ (they were taken directly out of the integral in Equation 4.6). This illustrates that C_B simply consists of a Taylor series expansion for $\Phi^p(\mathsf{E}[\mathbf{u}_\infty])$, evaluated at $\mathbf{u}^{p,*}$, subtracted from the profit at the plant optimum. If a good benchmark model

is known, or plant experiments can be cheaply done, the first line of Equation 4.10 can be used to more accurately estimate C_B.

- The variance cost (C_V) represents the performance loss because of the dispersion of the computed optima, $\mathbf{u}_\infty$, due to the presence of uncertainty. It can be expressed as follows [Forbes and Marlin, 1996]:

$$C_V = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\left.\frac{d^2\Phi^p}{du_i du_j}\right|_{\mathbf{u}^{p,*}} \cdot \mathbf{V}_{ij}\left(\mathbf{u}_\infty\right) \tag{4.11}$$

where $\mathbf{V}_{ij}\left(\mathbf{u}_\infty\right)$ is the covariance of inputs i and j. Note that the remaining terms of Equation 4.8 (that were not used in the bias cost definition) are used in this derivation. Further details can be found in Forbes and Marlin [1996]. If the individual variance and covariance terms, $\mathbf{V}_{ij}\left(\mathbf{u}_\infty\right)$ are arranged in covariance matrix form as $\mathbf{V}\left(\mathbf{u}_\infty\right)$, Equation 4.11 can be rewritten as follows [Forbes and Marlin, 1996]:

$$C_V = \frac{1}{2}\mathbf{1}^T\left(\nabla^2\Phi^p|_{\mathbf{u}^{p,*}} \circ \mathbf{V}\left(\mathbf{u}_\infty\right)\right)\mathbf{1} \tag{4.12}$$

where $\circ$ denotes the Hadamard product and $\mathbf{1}$ is a column vector of ones of length n_u.

Figure 4.1 illustrates the distinction between the two types of costs.

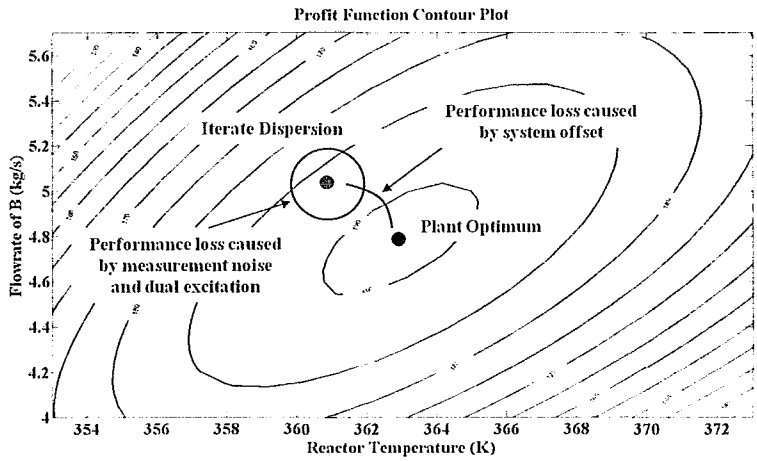

Figure 4.1: Design cost composition

Overall, the design cost is computed as the sum of both the performance loss caused by system offset (C_B) and the performance loss caused by measurement noise and dual excitation (C_V). Note that the true plant optimum ($\mathbf{u}^{p,*}$) and both the gradient and Hessian of the plant cost function evaluated at the plant optimum ($\nabla\Phi^p|_{\mathbf{u}^{p,*}}$ and $\nabla^2\Phi^p|_{\mathbf{u}^{p,*}}$) are required in order to compute C. Since true plant operation cannot be perfectly modeled, exact values for these quantities will never be known. However, they can be approximated in a number of different ways. One method is to utilize a detailed plant model, which might be too complex for online implementation, to make the approximation. If such a model is unavailable, plant experiments or even inherent process knowledge can be used to make satisfactory estimations. A more detailed discussion on this topic can be found in Forbes and Marlin [1996].

The design cost concept was extended by Zhang and Forbes, to include a transient term [Zhang and Forbes, 2000]. The idea here was to use the transient term to try to reflect the speed with which an RTO system responds to disturbances. This term, called the transient cost, was simply included in the design cost formulation along with both the bias and variance costs already defined in Forbes and Marlin [1996]. This extended design cost formulation is useful if the plant operation is expected to change frequently (every few RTO iterations) and if the RTO designer has *a priori* knowledge of the nature of the disturbances the RTO system is going to face [Zhang and Forbes, 2000]. Although the extended criterion will not be used in this work, it could be used in a future research study, possibly involving the selection of the best possible filter parameter matrix, $\mathbf{K}_\Lambda$, for a particular dual modifier adaptation implementation.

The design cost criterion can be used to make a variety of different design decisions. In the original paper by Forbes and Marlin [1996], and in the extended design cost paper by Zhang and Forbes [2000], it was used to select the best parameter set for updating the model of a particular system. In the extended design cost paper and in an RTO performance analysis paper [Zhang and Forbes, 2006], it was used to distinguish between different RTO methods. Other published applications include model selection [Zhang and Forbes, 2000] and sensor selection [Fraleigh *et al.*, 2003].

4.3 Design Cost for Dual Modifier Adaptation

In this section, specific details regarding the calculation of the design cost for dual modifier adaptation systems are discussed. The estimation of the variance cost is detailed first, followed by the bias cost.

4.3.1 Variance Cost Approximation

It is assumed that the Hessian matrix ($\nabla^2\Phi^p|_{\mathbf{u}^{p,*}}$) has already been estimated using one of the methods detailed in Section 4.2. Therefore, the only quantity left to compute in Equation 4.12 is the covariance matrix of the inputs, $\mathbf{V}(\mathbf{u}_\infty)$. One possible way would be to perform a post-optimal sensitivity analysis around the modifier adaptation optimum obtained with no noise or dual excitation. However, as $\mathcal{B}$ goes to infinity, the dual constraints (Equation 3.47) are non-differentiable, so no such analysis can be conducted. Instead, $\mathbf{V}(\mathbf{u}_\infty)$ will be approximated by considering two separate scenarios and then the covariance matrices calculated for each of these scenarios will be added together:

$$\mathbf{V}(\mathbf{u}_\infty) = \mathbf{V}_1(\mathbf{u}_\infty) + \mathbf{V}_2(\mathbf{u}_\infty) \tag{4.13}$$

The first scenario involves the modifier adaptation algorithm without the dual constraints, and $\mathbf{V}_1(\mathbf{u}_\infty)$ represents the effect that measurement noise has on the system. In the second scenario, $\mathbf{V}_2(\mathbf{u}_\infty)$ represents the effect of the dual constraints on the algorithm.

Scenario 1 (modifier adaptation without the dual constraint): The procedure begins by considering the covariance matrix of the modifiers. Then, a post-optimal sensitivity analysis is used to approximate the variance-covariance matrix of the inputs. This derivation is adapted from Forbes and Marlin [1996], where a similar derivation is made for the two-step approach.

Recall from Section 3.2.1 that a linear approximation of a closed-loop modifier adaptation system, using Broyden's method to estimate the plant output gradients, can be written as

follows:

$$\begin{bmatrix} \delta\overline{\Lambda}_{k+1} \\ \delta\overline{\Lambda}_{k} \\ \delta\mathrm{B}_{k+1} \end{bmatrix} \approx \Upsilon_{\infty} \begin{bmatrix} \delta\overline{\Lambda}_{k} \\ \delta\overline{\Lambda}_{k-1} \\ \delta\mathrm{B}_{k} \end{bmatrix} \tag{4.14}$$

where Υ_{∞} is defined as:

$$\Upsilon_{\infty} = \begin{bmatrix} \frac{d\overline{\Lambda}_{k+1}}{d\overline{\Lambda}_{k}} & \frac{d\overline{\Lambda}_{k+1}}{d\overline{\Lambda}_{k-1}} & \frac{d\overline{\Lambda}_{k+1}}{d\mathrm{B}_{k}} \\ \frac{d\overline{\Lambda}_{k}}{d\overline{\Lambda}_{k}} & \frac{d\overline{\Lambda}_{k}}{d\overline{\Lambda}_{k-1}} & \frac{d\overline{\Lambda}_{k}}{d\mathrm{B}_{k}} \\ \frac{d\mathrm{B}_{k+1}}{d\overline{\Lambda}_{k}} & \frac{d\mathrm{B}_{k+1}}{d\overline{\Lambda}_{k-1}} & \frac{d\mathrm{B}_{k+1}}{d\mathrm{B}_{k}} \end{bmatrix}_{[\overline{\Lambda}^{*}_{\infty},\overline{\Lambda}^{*}_{\infty},\mathrm{B}^{*}_{\infty}]} \tag{4.15}$$

The matrix, Υ_{∞}, cannot be computed in practice because of the non-differentiability of the Broyden update formula at the convergence point of the algorithm as $\mathcal{B}$ goes to infinity (Section 3.2.3). Instead, the behaviour of modifier adaptation with Broyden's method is approximated here by that of the ideal modifier adaptation algorithm. This gives rise to the following:

$$\begin{bmatrix} \delta\overline{\Lambda}_{k+1} \\ \delta\overline{\Lambda}_{k} \end{bmatrix} \approx \Upsilon_{\infty} \begin{bmatrix} \delta\overline{\Lambda}_{k} \\ \delta\overline{\Lambda}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{\partial\overline{\Lambda}_{k+1}}{\partial \mathrm{y}^{p}_{k+1}} & \frac{\partial\overline{\Lambda}_{k+1}}{\partial \mathrm{y}^{p}_{k}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathrm{y}^{p}_{k} \\ \delta\mathrm{y}^{p}_{k-1} \end{bmatrix} \tag{4.16}$$

where Υ_{∞} is now represented by:

$$\Upsilon_{\infty} = \begin{bmatrix} \frac{d\overline{\Lambda}_{k+1}}{d\overline{\Lambda}_{k}} & \frac{d\overline{\Lambda}_{k+1}}{d\overline{\Lambda}_{k-1}} \\ \frac{d\overline{\Lambda}_{k}}{d\overline{\Lambda}_{k}} & \frac{d\overline{\Lambda}_{k}}{d\overline{\Lambda}_{k-1}} \end{bmatrix}_{[\overline{\Lambda}^{*}_{\infty},\overline{\Lambda}^{*}_{\infty}]} \tag{4.17}$$

where $\overline{\Lambda}^{*}_{\infty}$ are the modifier values obtained by running the ideal modifier adaptation scheme, with no measurement noise, until convergence is reached (provided it is reached at all). The inputs at convergence are represented by $\mathbf{u}^{*}_{\infty}$.

Equation 4.16 can be rewritten as the following infinite sum, provided that the modifier adaptation system is point-wise stable:

$$\begin{bmatrix} \delta\overline{\Lambda}_{k+1} \\ \delta\overline{\Lambda}_{k} \end{bmatrix} \approx \sum_{i=0}^{\infty} (\Upsilon_{\infty})^{i} \begin{bmatrix} \frac{\partial\overline{\Lambda}_{k+1}}{\partial \mathrm{y}^{p}_{k+1}} & \frac{\partial\overline{\Lambda}_{k+1}}{\partial \mathrm{y}^{p}_{k}} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \delta\mathrm{y}^{p}_{k} \\ \delta\mathrm{y}^{p}_{k-1} \end{bmatrix} \tag{4.18}$$

If the modifier adaptation system is not point-wise stable, Equation 4.18 should not be used because as i increases, the expression on the right-hand side will go to infinity. If this

occurs, the design under consideration should be discarded. Now the expected value of the sum of squared deviations can be written:

$$\mathsf{E}\left[\begin{bmatrix}\delta\overline{\mathbf{\Lambda}}_{k+1}\\ \delta\overline{\mathbf{\Lambda}}_{k}\end{bmatrix}\begin{bmatrix}\delta\overline{\mathbf{\Lambda}}_{k+1}\\ \delta\overline{\mathbf{\Lambda}}_{k}\end{bmatrix}^{T}\right] = \sum_{i=0}^{\infty}(\mathbf{\Upsilon}_{\infty})^{i}\begin{bmatrix}\frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k+1}} & \frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k}}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\mathsf{E}\left[\begin{bmatrix}\delta\mathbf{y}^{p}_{k+1}\\ \delta\mathbf{y}^{p}_{k}\end{bmatrix}\begin{bmatrix}\delta\mathbf{y}^{p}_{k+1}\\ \delta\mathbf{y}^{p}_{k}\end{bmatrix}^{T}\right]\left[(\mathbf{\Upsilon}_{\infty})^{i}\begin{bmatrix}\frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k+1}} & \frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k}}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\right]^{T} \tag{4.19}$$

By definition, the covariance matrix of the modifiers $\left(\overline{\mathbf{\Lambda}}\right)$ for the first scenario, $\mathbf{V}_1\left(\overline{\mathbf{\Lambda}}_{\infty}\right)$, can be written as [Forbes and Marlin, 1996]:

$$\mathbf{V}_1\left(\overline{\mathbf{\Lambda}}_{\infty}\right) = \mathsf{E}\left[\begin{bmatrix}\delta\overline{\mathbf{\Lambda}}_{k+1}\\ \delta\overline{\mathbf{\Lambda}}_{k}\end{bmatrix}\begin{bmatrix}\delta\overline{\mathbf{\Lambda}}_{k+1}\\ \delta\overline{\mathbf{\Lambda}}_{k}\end{bmatrix}^{T}\right] - \mathsf{E}\begin{bmatrix}\delta\overline{\mathbf{\Lambda}}_{k+1}\\ \delta\overline{\mathbf{\Lambda}}_{k}\end{bmatrix}\left(\mathsf{E}\begin{bmatrix}\delta\overline{\mathbf{\Lambda}}_{k+1}\\ \delta\overline{\mathbf{\Lambda}}_{k}\end{bmatrix}\right)^{T} \tag{4.20}$$

It is assumed next that the measurement noise is white noise (not necessarily Gaussian), of variance $\sigma^2_{\mathbf{y}^p}$; that is,

$$\mathsf{E}\left[\begin{bmatrix}\delta\mathbf{y}^{p}_{k+1}\\ \delta\mathbf{y}^{p}_{k}\end{bmatrix}\begin{bmatrix}\delta\mathbf{y}^{p}_{k+1}\\ \delta\mathbf{y}^{p}_{k}\end{bmatrix}^{T}\right] = \begin{bmatrix}\sigma^2_{\mathbf{y}^p} & \mathbf{0}\\ \mathbf{0} & \sigma^2_{\mathbf{y}^p}\end{bmatrix} \tag{4.21}$$

From this assumption, it is clear that $\mathsf{E}\begin{bmatrix}\delta\overline{\mathbf{\Lambda}}_{k+1}\\ \delta\overline{\mathbf{\Lambda}}_{k}\end{bmatrix} = \mathbf{0}$ and it follows from Equation 4.19 that:

$$\mathbf{V}_1\left(\mathbf{\Lambda}_{\infty}\right) \approx \sum_{i=0}^{\infty}(\mathbf{\Upsilon}_{\infty})^{i}\begin{bmatrix}\frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k+1}} & \frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k}}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\sigma^2_{\mathbf{y}^p} & \mathbf{0}\\ \mathbf{0} & \sigma^2_{\mathbf{y}^p}\end{bmatrix}\left[(\mathbf{\Upsilon}_{\infty})^{i}\begin{bmatrix}\frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k+1}} & \frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k}}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\right]^{T} \tag{4.22}$$

There is another important assumption, in relation to Equation 4.22, that needs to be discussed here. Inside the $\frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k+1}}$ sensitivity the term $\frac{\partial\mathbf{B}_{k+1}}{\partial\mathbf{y}^{p}_{k+1}}$ appears and similarly, inside the $\frac{\partial\overline{\mathbf{\Lambda}}_{k+1}}{\partial\mathbf{y}^{p}_{k}}$ sensitivity the term $\frac{\partial\mathbf{B}_{k+1}}{\partial\mathbf{y}^{p}_{k}}$ is present. In practice, it is possible to approximate these terms for the n^{th} measurement and the i^{th} output (when $n = i$) as:

$$\frac{\partial\mathbf{B}^{i}_{k+1}}{\partial\mathbf{y}^{p,n}_{k+1}} = -\frac{\partial\mathbf{B}^{i}_{k+1}}{\partial\mathbf{y}^{p,n}_{k}} \approx \mathbf{s} \tag{4.23}$$

where $\mathcal{B} = \Omega^T\Omega$ and $s_i = \Omega_{(i,i)}$. See Appendix C for a derivation in the case that $\mathcal{B}$ is diagonal.

Post-optimal sensitivity analysis is carried out next, as detailed in Fiacco [1983], to obtain the covariance matrix of the inputs for scenario 1:

$$\mathbf{V}_1(\mathbf{u}_\infty) = \left[\frac{\partial \mathbf{U}^*}{\partial \overline{\mathbf{\Lambda}}_{k+1}} \frac{\partial \mathbf{U}^*}{\partial \overline{\mathbf{\Lambda}}_k}\right]_{\overline{\mathbf{\Lambda}}^*_\infty} \mathbf{V}_1(\overline{\mathbf{\Lambda}}_\infty) \left[\frac{\partial \mathbf{U}^*}{\partial \overline{\mathbf{\Lambda}}_{k+1}} \frac{\partial \mathbf{U}^*}{\partial \overline{\mathbf{\Lambda}}_k}\right]^T_{\overline{\mathbf{\Lambda}}^*_\infty} \tag{4.24}$$

To re-iterate, this post-optimal sensitivity analysis is conducted at the converged optimum of the ideal modifier adaptation scheme, when it is run without noise and dual excitation.

Scenario 2: In this scenario, the dual constraint is always active and causes variance in the optimal input values. The following is assumed as a first approximation:

$$\mathbf{V}_2(\mathbf{u}_\infty) = \frac{1}{2}\mathcal{B}^{-1} \tag{4.25}$$

This approximation is supported by experience with numerical simulations. See Appendix C for an explanation in the case that $\mathcal{B}$ is diagonal. A future research direction would be to develop some mathematical reasoning to support the approximation.

4.3.2 Bias Cost Approximation

Assuming that the gradient and Hessian of the plant profit function are well known, and a good estimate of the plant optimum is available, Equation 4.10 can be used to calculate the bias cost for unconstrained systems. The only value in Equation 4.10 specific to the dual modifier adaptation system in question is $\mathsf{E}[\mathbf{u}_\infty]$. It is possible to assume that $\mathsf{E}[\mathbf{u}_\infty] = \mathbf{u}^*_\infty$, however note this assumption ignores the effect of any gradient offset that Broyden's method may cause in practice.

In the case of constrained problems a modification needs to be made to the bias cost calculation procedure. This modification is introduced to account for a shift in the expected value of the iterates ($\mathsf{E}[\mathbf{u}_\infty]$) when the dual constraints are implemented. This shift occurs due to the tendency of the model-based optimizer to select points that are inside the feasible region.

Offset in Constrained Problems

The dual constraints have an additional effect on problems in which at least one process inequality constraint is active. Due to the desire of the model-based optimizer to arrive at feasible operating points, the expected value of the iterates may be pushed away from the constraint, toward the interior of the feasible region. This movement can be approximated by considering a fictitious shift in each active constraint in the direction normal to the hyperplane tangent to the constraint at the ideal modifier adaptation optimum. This direction can be represented, for constraint i, by $-\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)$.

An appropriate constraint shift can be approximated by the distance from the center of the ellipsoid to its outer edge in the direction $-\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)$. If $\mathbf{z}$ is a point vector centered at the ideal modifier adaptation optimum and the ellipsoid is also centered at this optimum, then the following can be written:

$$\mathbf{z}^T \boldsymbol{\mathcal{B}} \mathbf{z} = 1 \tag{4.26}$$

which requires that the point $\mathbf{z}$ is on the ellipsoid. Since $\mathbf{z}$ is also required to be in the direction $\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)$, it follows that:

$$\mathbf{z} = \theta \left(-\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty) \right) \tag{4.27}$$

where θ is a scalar parameter which fixes the length of the point vector $\mathbf{z}$. Substituting (4.27) into (4.26) and solving for θ:

$$\theta = \frac{1}{\sqrt{\left(\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)\right)^T \boldsymbol{\mathcal{B}} \left(\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)\right)}} \tag{4.28}$$

Equation 4.28 can now be substituted back into Equation 4.27, thereby giving the distance from the ideal modifier adaptation optimum $(\mathbf{u}^*_\infty)$ to the expected value of the iterates in the input space. To obtain a constraint shift, the result of Equation 4.28 has to be converted to a distance in the constraint space by multiplying it by $-\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)^T$:

$$\Delta g^i = \frac{\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)^T \frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)}{\sqrt{\left(\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)\right)^T \boldsymbol{\mathcal{B}} \left(\frac{\partial g^i}{\partial \mathbf{u}}(\mathbf{u}^*_\infty)\right)}} \tag{4.29}$$

Denoting the change in the constraint value (Δg^i) by the variable γ_i, the change in the ideal modifier adaptation optimum can then be approximated:

$$\Delta \mathbf{u} = \left.\frac{\partial \mathbf{U}^*}{\partial \gamma_i}\right|_{\overline{\mathbf{\Lambda}}^*_\infty} \gamma_i \tag{4.30}$$

where γ_i is considered to be a parameter in the optimization problem and $\frac{\partial \mathbf{U}^*}{\partial \gamma_i}$ is computed using post-optimal sensitivity analysis at $\overline{\mathbf{\Lambda}}^*_\infty$ [Fiacco, 1983]. If more than one constraint is active, the post-optimal sensitivity analysis is applied for all active constraints simultaneously. The expected value of the iterates can now be computed as:

$$\mathsf{E}\left[\mathbf{u}_\infty\right] = \Delta \mathbf{u} + \mathbf{u}^*_\infty \tag{4.31}$$

4.3.3 Constraint Backoff Calculation

In the case of problems where process constraints are active at the ideal modifier adaptation optimum, it is sometimes necessary to back off from the active constraint(s) in order to ensure that most of the operating points chosen by the model-based optimization problem are feasible points of the plant. The decision to do this is strictly up to the designer of the RTO system, as it will likely depend on the severity of the consequences if a plant process constraint is violated.

The root cause of this infeasibility problem is that measurement noise will inevitably cause the modifiers to be incorrectly estimated. Since these modifiers are often involved in the process constraints, specifically whenever a constraint involves an output variable, the errors in the modifiers will commonly cause the constraints to be incorrectly identified. These constraints will generally move in the area of the true plant constraints, however they may at times either enforce unnecessary back-off, or worse relax the true plant constraints. For example, for the constrained Williams-Otto Reactor test case, the dashed lines in Figure 4.2 show the true plant constraint and the corresponding process model constraints at ten different iterations during a simulation run.

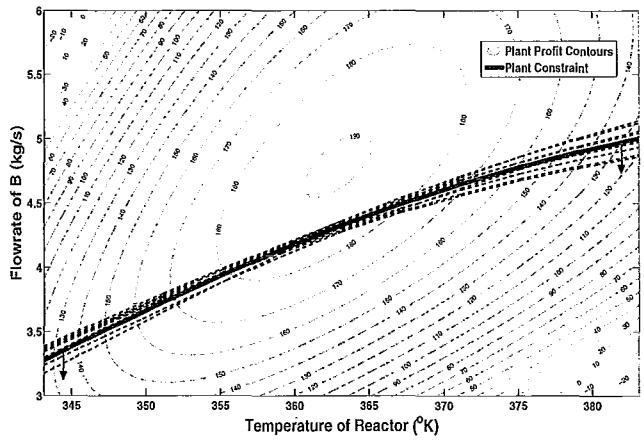

Figure 4.2: Illustration of variation of process constraints

An approach designed to combat uncertainty in constrained RTO problems was presented in Zhang *et al.* [2002]. This approach involved the online implementation of probabilistic constraints, which attempted to ensure that a feasible operating point in terms of the plant was chosen a certain percentage of the time. This percentage was a tuning parameter that was set by the user. The heart of this procedure involved converting the constraints written in probabilistic fashion, into constraints that could be implemented by the NLP solver tasked with solving the model-based optimization problem.

One potential drawback of this approach however is that since the probabilistic constraint calculation is done on-line, this constraint conversion procedure must be carried out every iteration, which could be computationally challenging for large-scale RTO systems. Instead, in this work, a method is proposed where the stochastic programming problem is solved off-line, which will yield an operating point to which the active constraints should be backed off according to the specified infeasibility tolerance. This back-off is then used as both a part of the design cost calculation and also implemented as a part of the online algorithm.

Back-off approaches have been used in the past to try to combat constraint violation in RTO systems. In the implementation in Loeblein and Perkins [1998], the back-off was computed to address uncertainty in both the plant measurements and the parameters. Additionally, an interesting example of the use of a back-off to address a kind of a RTO-control hybrid

problem can be found in Contreras-Dordelly and Marlin [2000].

The heart of the offline probabilistic programming approach involves the solution of the following optimization problem:

$$\begin{aligned} \mathbf{u}^{st} \in \arg\min_{\mathbf{u}} \quad & \phi\left(\mathbf{u}, \hat{\mathbf{y}}^m\right) \\ \text{s.t.} \quad & \hat{\mathbf{y}}^m = \mathbf{f}\left(\mathbf{u}, \beta\right) + \overline{\epsilon}^*_\infty + \boldsymbol{\lambda}^{*T}_\infty \mathbf{u} \\ & \mathsf{P}\left[g^i(\mathbf{u}, \hat{\mathbf{y}}^m) \leq 0\right] \geq p_i \quad i = 1, ..., n_g \end{aligned} \tag{4.32}$$

where $\mathbf{u}^{st}$ is the back-off point, P is a probability operator and p_i is the probability requirement for constraint i.

Recall that the probabilistic constraints, as shown in Equation 4.32, cannot be sent to a traditional NLP solver (such as *fmincon*). Instead, they must be approximated by constraints of a deterministic form first. Note also that the dual constraints do not appear in Problem 4.32. The assumption is being made here that the dual constraints are inactive at all times. This is due to the fact that the dual excitation plays no role in constraint violation, because it is known by the model-based optimizer.

Now the procedure originally discussed in Zhang *et al.* [2002] for online implementation will be adapted for use in the offline design of dual modifier adaptation systems. This procedure considers each constraint individually. Therefore, an overall feasibility level cannot be set, rather constraint satisfaction levels must be set on an individual basis. A method which considers all constraints simultaneously is also given in Zhang *et al.* [2002]. This method could be adapted for use in this design procedure in the future.

A first-order Taylor series expansion of the constraints in terms of the inputs and modifiers, around the ideal modifier adaptation optimum $\left(\mathbf{u}^*_\infty, \overline{\boldsymbol{\Lambda}}^*_\infty\right)$, gives the following:

$$G^i(\mathbf{u}, \overline{\boldsymbol{\Lambda}}) \approx G^i(\mathbf{u}^*_\infty, \overline{\boldsymbol{\Lambda}}^*_\infty) + \left.\frac{\partial G^i}{\partial \overline{\boldsymbol{\Lambda}}}\right|_{\mathbf{u}^*_\infty, \overline{\boldsymbol{\Lambda}}^*_\infty} \left(\overline{\boldsymbol{\Lambda}} - \overline{\boldsymbol{\Lambda}}^*_\infty\right) + \left.\frac{\partial G^i}{\partial \mathbf{u}}\right|_{\mathbf{u}^*_\infty, \overline{\boldsymbol{\Lambda}}^*_\infty} \left(\mathbf{u} - \mathbf{u}^*_\infty\right), \quad i = 1, ..., n_g \tag{4.33}$$

where G^i is the constraint function i written only in terms of the inputs and modifiers ($G^i(\mathbf{u}, \overline{\boldsymbol{\Lambda}}) := g^i(\mathbf{u}, \hat{\mathbf{y}}^m(\mathbf{u}, \overline{\boldsymbol{\Lambda}}))$).

The constant terms from Equation 4.33 can be collected and represented by the vector $\mathbf{d}$ with $d_i = G^i\left(\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty\right) - \left.\frac{\partial G^i}{\partial \overline{\mathbf{\Lambda}}}\right|_{\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty} \overline{\mathbf{\Lambda}}^*_\infty - \left.\frac{\partial G^i}{\partial \mathbf{u}}\right|_{\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty} \mathbf{u}^*_\infty,\ i = 1, ..., n_g$. The effect of the uncertain parameters $(\overline{\mathbf{\Lambda}})$ can then be represented by a new variable, $\boldsymbol{\xi}$, in the following way:

$$\xi_i = \left.\frac{\partial G^i}{\partial \overline{\mathbf{\Lambda}}}\right|_{\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty} \overline{\mathbf{\Lambda}} + d_i,\ i = 1, ..., n_g \tag{4.34}$$

Statistical properties of $\boldsymbol{\xi}$ are required. Its expected value can be approximated as:

$$\mathsf{E}\left[\xi_i\right] \approx \left.\frac{\partial G^i}{\partial \overline{\mathbf{\Lambda}}}\right|_{\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty} \mathsf{E}\left[\,\overline{\mathbf{\Lambda}}\,\right] + d_i,\ i = 1, ..., n_g \tag{4.35}$$

where the expected values of the modifiers are approximated as: $\mathsf{E}\left[\,\overline{\mathbf{\Lambda}}\,\right] = \overline{\mathbf{\Lambda}}^*_\infty$. This approximation is only valid as long as the backoff point, $\mathbf{u}^{st}$, is not too far from the ideal modifier adaptation optimum $(\mathbf{u}^*_\infty)$. If this is not the case in practice, alternative designs should be considered which better control the influence of measurement noise on the algorithm.

The variance of ξ_i can also be approximated as:

$$\mathbf{V}\left(\xi_i\right) = \left[\left.\frac{\partial G^i}{\partial \overline{\mathbf{\Lambda}}}\right|_{\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty}\right] \mathbf{V}_1\left(\mathbf{\Lambda}^*_\infty\right) \left[\left.\frac{\partial G^i}{\partial \overline{\mathbf{\Lambda}}}\right|_{\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty}\right]^T,\ i = 1, ..., n_g \tag{4.36}$$

Since each constraint is dealt with individually, separate variances for each element of $\boldsymbol{\xi}$ are isolated instead of considering the overall covariance matrix.

Using Equations 4.33 and 4.34, the overall probabilistic constraints can be written in the following way:

$$\mathsf{P}\left[-\left.\frac{\partial G^i}{\partial \mathbf{u}}\right|_{\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty} \mathbf{u} \geq \boldsymbol{\xi}_i\right] \geq p_i,\ i = 1, ..., n_g \tag{4.37}$$

The assumption is made that the random variable $\boldsymbol{\xi}_i$ can be represented by the normal distribution, which has the cumulative distribution function, N_i:

$$N_i\left(-\left.\frac{\partial G^i}{\partial \mathbf{u}}\right|_{\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty} \mathbf{u}; \mathsf{E}\left[\xi_i\right], \mathbf{V}\left(\xi_i\right)\right) \geq p_i,\ i = 1, ..., n_g \tag{4.38}$$

Finally, taking the inverse normal distribution of both sides of Equation 4.38:

$$-\left.\frac{\partial G^i}{\partial \mathbf{u}}\right|_{\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty} \mathbf{u} \geq N_i^{-1}\left(p_i; \mathsf{E}\left[\xi_i\right], \mathbf{V}\left(\xi_i\right)\right)\ \ i = 1, ..., n_g \tag{4.39}$$

This last set of constraints can replace the probabilistic constraints in Problem 4.32.

Once the problem (4.32) is solved to find an appropriate operating point ($\mathbf{u}^{st}$), the back-off from a constraint can be computed in the following way:

$$\eta^i = g^i(\mathbf{u}^*_\infty, \hat{\mathbf{y}}^m(\mathbf{u}^*_\infty, \overline{\mathbf{\Lambda}}^*_\infty)) - g^i(\mathbf{u}^{st}, \hat{\mathbf{y}}^m(\mathbf{u}^{st}, \overline{\mathbf{\Lambda}}^*_\infty)) \tag{4.40}$$

where η^i is the estimated back-off parameter for constraint i. Note that an additional assumption is made here, that the optimal modifiers are equal to $\overline{\mathbf{\Lambda}}^*_\infty$ at $\mathbf{u}^{st}$. This parameter is then added to the constraints in the model-based optimization of the online algorithm:

$$\mathbf{g}(\mathbf{u}, \hat{\mathbf{y}}^m) + \boldsymbol{\eta} \leq \mathbf{0} \tag{4.41}$$

In summary, Problem 4.32 is solved with the constraints of Equation 4.39 to obtain the back-off point, $\mathbf{u}^{st}$. Equations 4.40 and 4.41 are then used to compute and implement the required backoff, η_i, for each constraint, i, in the online algorithm.

If the backoff procedure is run, the point that the constraint(s) are backed off to ($\mathbf{u}^{st}$) is effectively the new ideal modifier adaptation optimum. Therefore, Equation 4.31 must be rewritten:

$$\mathsf{E}[\mathbf{u}_\infty] = \Delta\mathbf{u} + \mathbf{u}^{st} \tag{4.42}$$

A sample scenario is illustrated in Figure 4.3. In this figure, the constraint is backed-off, essentially moving the ideal modifier adaptation optimum from $\mathbf{u}_\infty$ to $\mathbf{u}^{st}$. Then, the new expected value of the iterates ($\mathsf{E}[\mathbf{u}_\infty]$) is computed through the offset calculation procedure of Section 4.3.2, along with Equation 4.42.

4.4 Test Case: Williams-Otto Reactor

The concepts described in this chapter are now illustrated using the Williams-Otto Reactor test case. The section is split up into three subsections. First, a sample design cost

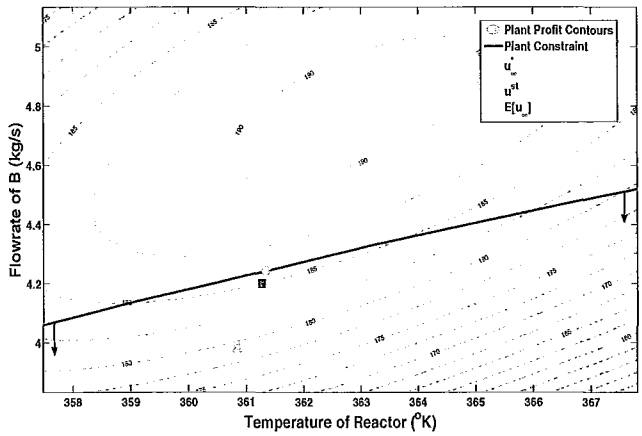

Figure 4.3: Illustration of the results of both the back-off and offset procedures

calculation is given. Next, the design of an RTO system for the unconstrained Williams-Otto Reactor formulation is investigated. Finally, the design of an RTO system for the constrained Williams-Otto Reactor process is examined.

4.4.1 Design Cost Computation

In this subsection, a sample design cost calculation is given to help reinforce the concepts discussed earlier in this chapter. To begin, a set of ellipsoid parameters, $\mathcal{B}$, must be chosen for the algorithm. For this purpose, a diagonal $\mathcal{B}$ matrix is chosen: $\mathcal{B} = \mathsf{diag}(\mathbf{b})$ where $\mathbf{b} = [25, 25]$. These values were proven through testing to provide reasonable performance, however, no attempt was made to optimize them.

The plant optimum for the Williams-Otto Reactor system is $\mathbf{u}^{p,*} = [362.885, 4.7864]$. Since there are no constraints active at the plant optimum, the gradient is equal to zero there. The Hessian at the plant optimum is:

$$\nabla^2 \Phi^p|_{\mathbf{u}^{p,*}} = \begin{bmatrix} 211.13 & -62.49 \\ -62.49 & 44.37 \end{bmatrix} \tag{4.43}$$

Note that this matrix is positive definite, which confirms that the plant optimum is indeed a minimum for the Williams-Otto Reactor cost function. Also note that, the scaling detailed

in Section 3.1.1 is used in all the calculations and simulations in this section.

To begin, the ideal modifier adaptation algorithm was run, under noiseless conditions, until it converged to $\mathbf{u}_{\infty}^{*} = \mathbf{u}^{p,*}$. Filter parameters of 0.25 were used and the true plant model, described in Section 2.7, was used to simulate plant operation. The algorithm converged along the path previously shown in Figure 2.4. Since the plant and ideal modifier adaptation optima were identical (this will always be the case since exact output gradients are assumed) and the problem was unconstrained, it was immediately clear that the bias cost approximation, C_B, was zero.

In order to compute the variance cost (C_V), the covariance matrix of the set-points, $\mathbf{V}(\mathbf{u}_{\infty})$, is required. It was computed, as in Equation 4.13, by summing two separate contributions: the effect of measurement noise on the iterates without the dual constraints and the direct effect of the dual constraints on the iterates. Measurement noise of 0.5% of a typical set of mass fraction values was assumed. The values of each of the contributions as well as the final covariance matrix are shown below:

$$\mathbf{V}(\mathbf{u}_{\infty}) = \mathbf{V}_1(\mathbf{u}_{\infty}) + \mathbf{V}_2(\mathbf{u}_{\infty}) = \begin{bmatrix} 0.0551 & 0.139 \\ 0.139 & 0.350 \end{bmatrix} + \begin{bmatrix} 0.02 & 0 \\ 0 & 0.02 \end{bmatrix} = \begin{bmatrix} 0.0751 & 0.139 \\ 0.139 & 0.370 \end{bmatrix} \tag{4.44}$$

Note that both the measurement noise and the dual constraint make important contributions to the overall covariance matrix approximation. This is in general the sign of an effective design, because one of the goals of the design procedure is to find a good trade-off between the effect of the measurement noise and the effect of the dual excitation on the algorithm. The variance cost was then be approximated using Equation 4.12:

$$\begin{aligned} C_V &= \frac{1}{2}\mathbf{1}^T\left(\nabla^2\Phi^p|_{\mathbf{u}^{p,*}} \circ \mathbf{V}(\mathbf{u}_{\infty})\right)\mathbf{1} \\ &= \frac{1}{2}\mathbf{1}^T\left(\begin{bmatrix} 211.13 & -62.49 \\ -62.49 & 44.37 \end{bmatrix} \circ \begin{bmatrix} 0.0751 & 0.139 \\ 0.139 & 0.370 \end{bmatrix}\right)\mathbf{1} \\ &= 7.463 \end{aligned} \tag{4.45}$$

Since the bias cost was zero (Equation 4.10), the total design cost (C), by Equation 4.9, was also equal to 7.463. Practically, this means that if many iterations were run implementing

the settings assumed in the design cost approximation, the average cost expected from the algorithm would be $\Phi^p(\mathbf{u}^{p,*}) + 7.463 = -190.7978 + 7.463 = -183.33$ (or in other words a profit of 183.33).

The design cost results were tested by running the algorithm online for 2000 iterations. The result was an average cost function value of -185.32, which is reasonably close to the design cost approximation of -183.33. Naturally, this simulation was done using the same settings that were used for the design cost calculation. The results of the simulation run are shown below in Figure 4.4.

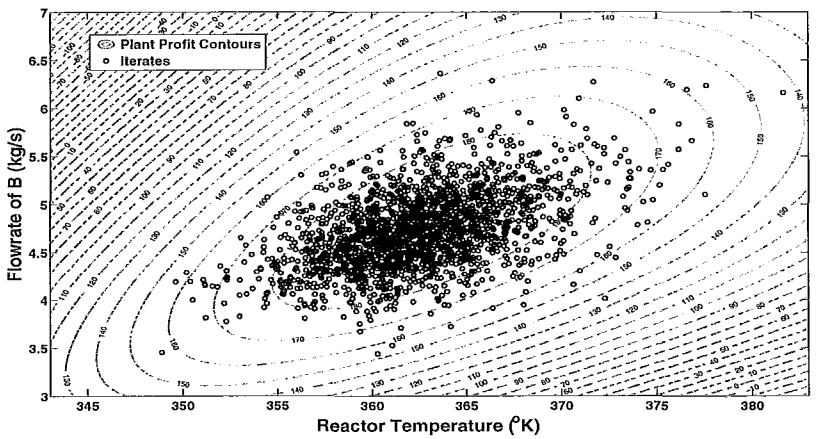

Figure 4.4: Online simulation results for $\mathbf{b} = [25, 25]$

Next, instead of testing one specific ellipsoid size, the whole range of possible sizes is sampled. For each size sampled, both the design cost calculation and an online simulation test were carried out. Each online simulation test consisted of taking the average cost value over 20,000 iterations. The same filter parameter and measurement noise settings that were used in the design cost example above were used here as well. The results are shown in Figure 4.5.

The comparison shows that the design cost predictions and the online simulation results are similar for some sets of dual constraint parameters and very different for others. Both contours have the same general bowl shape, with one distinct minimum. This is expected, because sizing the ellipsoid is a trade-off between the effect of the measurement noise, which

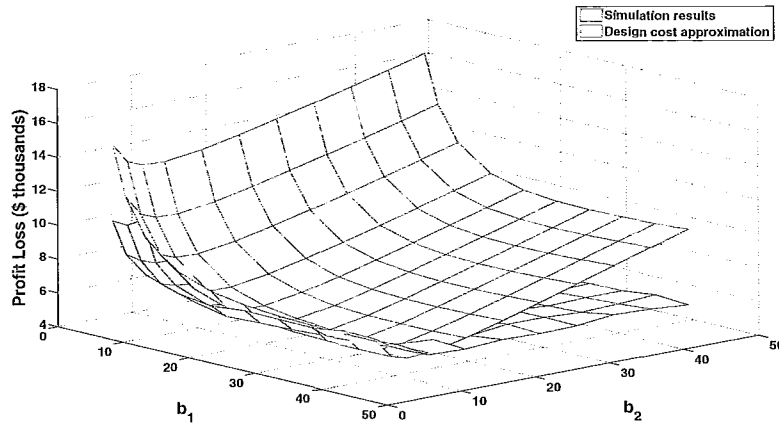

Figure 4.5: Comparison of design cost approximation and online simulation results

is dampened by a larger ellipsoid, and the effect of the dual excitation, which increases as the ellipsoid grows. As in design cost sample calculation given previously, the design cost approximation over-predicts the loss in profit for most of the sets of dual constraint parameters tested. Potential reasons for the over-estimation are discussed later on in the explanation of the results of Figure 4.6.

Out of the sampled ellipsoid sizes, the design procedure recommends $\mathbf{b} = [40, 10]$ with an average profit of 185.09. The average profit in the corresponding simulation was 184.945, which illuminates the fact that the design cost criterion can also under-estimate the profit loss caused by a particular design. On the other hand, the simulation study recommends $\mathbf{b} = [30, 15]$ with an average profit of 185.49. It is important to note that the design cost approximation results are not expected to exactly match the simulation results because of the many assumptions made in the calculation procedure. Instead, the goal is simply to select a good set of dual constraint parameters.

To get a closer look at the trends in Figure 4.5, a more detailed set of results for the special case when $b_1 = b_2$ are shown in Figure 4.6. To create each of the sets of simulation results in Figure 4.6, 20,000 iterations were again run with each set of dual constraint parameters, and the average profit over the iterations was computed.

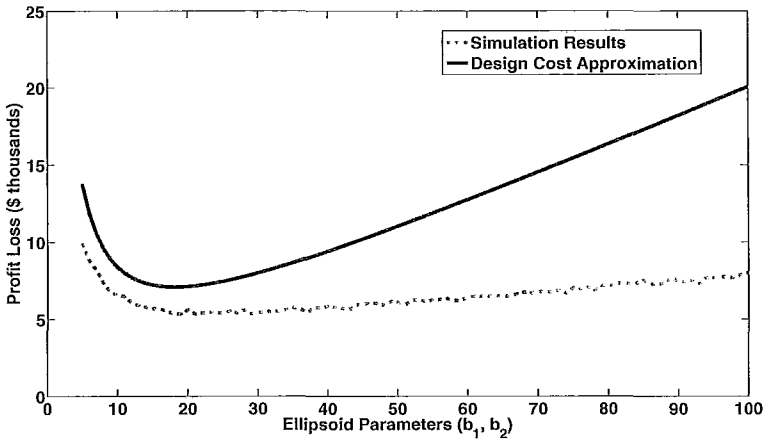

Figure 4.6: Comparison of design cost and online simulation results ($b_1 = b_2$)

Figure 4.6 shows good agreement between the design cost approximation and the simulation results in the vicinity of the optimal ellipsoid sizes. The slight over-estimation of the profit loss could be due to the fact that, in the design cost approximation, the interaction between the measurement noise and the dual excitation is ignored by treating them separately and then adding them up. The interaction is likely significant in this region because it is expected that at the optimal ellipsoid sizes there will be a balance struck between the effect of the measurement noise and the effect of the dual constraints.

There is a significant difference between the design cost and simulation results when the ellipsoid is either very small or very large. When the ellipsoid is small, the design cost over-predicts the actual profit loss by a significant amount. Again, this over-estimation could be due to the fact that the interaction between the two scenarios is being disregarded in the design cost approximation. Another potential cause of this over-estimation is the fact that the value of $\frac{\partial \mathbf{B}_{k+1}}{\partial \mathbf{y}_{k+1}^{p}}$ (Equation 4.23) that is used in the covariance matrix approximation for scenario 1 ($\mathbf{V}_1(\mathbf{u}_\infty)$) is actually an upper bound on the sensitivity (see Appendix C).

Similarly, the design cost criterion also over-estimates the profit lost when the ellipsoid is very large. To further investigate this, a closer look is taken at the simulation results when $\mathbf{b} = [5, 5]$. In this simulation, one of the dual constraints is active in 90% of the iterations. Therefore, while there is still some interaction between the two scenarios, ignoring this

interaction is unlikely to be the sole cause of the over-estimation. Another potential cause is the fact that the maximum effect of the dual constraint on the iterates is being used in the covariance matrix approximation for scenario 2, $\mathbf{V}_2(\mathbf{u}_\infty)$, (see Appendix C). For the simulation with $\mathbf{b} = [5, 5]$, the total profit loss was 9.94. On the other hand, in the corresponding variance cost approximation, the estimated profit loss due to scenario 2 alone was 12.775. Since this is already greater than the total profit loss in the simulation, this example supports the arguement that $\mathbf{V}_2(\mathbf{u}_\infty)$ is being over-estimated.

4.4.2 Design Cost - Unconstrained Formulation

An example constructed to illustrate how the design cost criterion can be practically applied to help design an RTO system for the unconstrained Williams-Otto Reactor process is now presented. In modifier adaptation it is possible to choose which modifiers are updated. The advantage to updating less than the full set of modifiers is that the effect of measurement noise on the RTO system may be significantly dampened. Of course the disadvantage is that the system will not converge to the plant optimum, $\mathbf{u}^{p,*}$, if the KKT conditions of the model-based optimization problem no longer match those of the plant.

For the Williams-Otto Reactor case study, upon examination of the cost function (2.29), there are two outputs that need to be modified, X_E and X_P, in order to converge to the plant optimum. Three different updating scenarios are therefore possible: modifying only X_E (design 1), modifying only X_P (design 2) and modifying both mass fractions (design 3). The trials assume filter parameters of 0.25 and white noise added to the mass fraction measurements with a standard deviation of 0.5% of a nominal set of their values. The inner optimization problem of Equation 4.3 was run for each of the three designs. Table 4.1 summarizes the results.

In Table 4.1 it is clear that the design cost criterion predicts that both designs 2 and 3 should produce nearly identical performance. Modifying only X_E (design 1) results in a large bias cost, indicating that the algorithm is moving to the area around a point far away from the plant optimum in this case. This trial also has by far the lowest variance cost.

Table 4.1: Breakdown of design cost estimates for different modifier combinations

	Design 1	Design 2	Design 3
C_B	12.88	0.0158	0
C_V	0.314	5.697	5.677
C	13.19	5.7128	5.677

The results of the design cost approximation can be heavily dependent on the expected level of measurement noise in the system. For instance, if this level increases, the variance costs of designs 2 and 3 would likely increase substantially. Conversely, since the variance cost of design 1 is quite low in Table 4.1, it would likely not increase by as much. Therefore, if the increase in measurement noise was large enough, design 1 could become the best option.

The recommended ellipsoid size parameters, obtained by solving (4.3) for each of the three design options are given in Table 4.2. Both designs 2 and 3 have very similar recommended ellipsoid sizes. This makes sense given that their bias and variance cost estimates are very similar as well. The method in which only X_E is modified (design 1) has a much smaller recommended ellipsoid size, likely because measurement noise does not effect the system as much. This is supported by the fact that it has a much lower variance cost prediction for its optimal design, despite a much smaller ellipsoid size. In general, a lower susceptibility to measurement noise dampens the potential effect of the peaking phenomenon, which the ellipsoid constraint is trying to prevent. Therefore the ellipsoid itself does not need to be as large to keep the effect of measurement noise in check.

Table 4.2: Ellipsoid sizes for different modifier combinations

	Design 1	Design 2	Design 3
b_1	789.6	37.04	36.97
b_2	123.3	7.90	7.97

Simulations were also run for each of these potential designs. In each simulation the appropriate online algorithm was run for 2000 iterations, in the presence of measurement noise, and the average profit lost was reported. In addition, approximations of both of the design

cost components were made using statistics ($\mathrm{E}[\mathbf{u}_\infty]$ and $\mathbf{V}(\mathbf{u}_\infty)$) collected during each simulation run. Note that C is not expected to match the "true" average profit lost due to the use of the gradient and Hessian to approximate the actual plant profit surface in the design cost approximation. The results are shown numerically in Table 4.3 and visually in Figure 4.7.

Table 4.3: Simulation results for different modifier combinations

	Design 1	Design 2	Design 3
Average Lost Profit	12.912	6.008	5.753
C_B	11.53	0.0793	0.0361
C_V	0.220	5.985	5.862
C	11.730	6.065	5.898

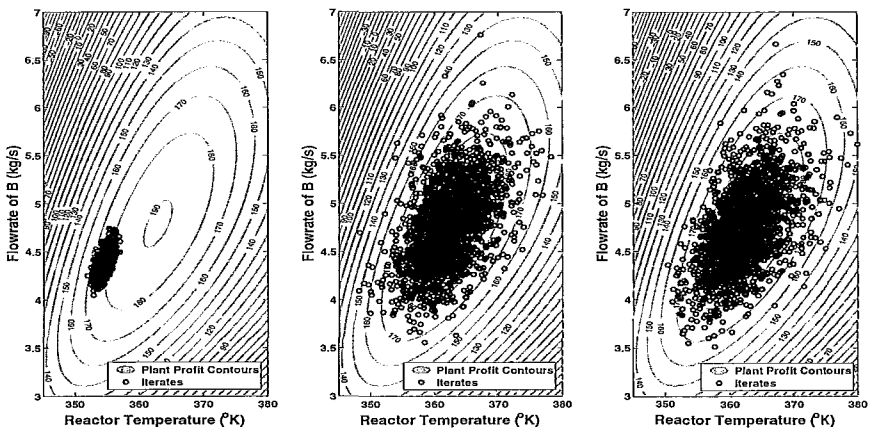

Figure 4.7: Simulations of designs 1-3 - unconstrained formulation

The left panel of Figure 4.7 verifies that the reason for the high bias cost for design 1 is that it fails to move to the vicinity of the plant optimum. The low variance cost prediction for this approach is also validated, in that the iterates are much more concentrated for this design than for the other two.

The central plot in Figure 4.7 demonstrates that if the mass fraction of P is modified instead of the mass fraction of E, the algorithm moves to an area much closer to the plant optimum.

The price that is paid for approaching the plant optimum, however, is a large increase in the variance of the iterates. This is consistent with the large increase in the variance cost approximation between design 1 and design 2.

The right panel of Figure 4.7 shows that design 3 produces results that are very similar to those of design 2. This is consistent with the design cost approximations. In summary, design 3 should be selected as its average profit loss is a little lower in both the design cost approximation and the simulation results.

A closer look is taken now at the differences between the design cost approximation and the simulation results for design 3, in which the modifiers for both X_E and X_P were updated. Table 4.4 illustrates some of the key differences.

Table 4.4: Design cost/simulation detailed comparison for dual modifier adaptation

	Design Cost Approximation	Simulation
$\mathsf{E}[\phi(\mathbf{u}_\infty, \mathbf{F}(\mathbf{u}_\infty))]$	-185.12	-185.05
$\mathsf{E}[\mathbf{u}_\infty]$	$\begin{bmatrix} -0.0134 \\ -0.214 \end{bmatrix}$	$\begin{bmatrix} -0.0374 \\ -0.247 \end{bmatrix}$
$\mathsf{V}[\mathbf{u}_\infty]$	$\begin{bmatrix} 0.0528 & 0.0830 \\ 0.0830 & 0.238 \end{bmatrix}$	$\begin{bmatrix} 0.04707 & 0.0494 \\ 0.0494 & 0.179 \end{bmatrix}$

The design cost approximation over-estimates the average profit by about 0.07 (or \$70). It is important to note from this that the design cost criterion again provides a good prediction of the average profit when "good" ellipsoid parameters are being considered. All of the components of the covariance matrix of the inputs are slightly over-predicted by the design cost criterion, however, since the covariance terms are over-predicted to the greatest extent, the variance cost is actually under-predicted. There is also a slight discrepancy between the expected value of the inputs in the design cost calculation and the average value of the inputs in the simulation. This could be due to a small lingering offset between the Broyden derivative estimates and the true plant output gradient.

In order to further evaluate the performance of the dual modifier adaptation algorithm

it is now compared to the two-step approach. Recall Figure 2.5, which illustrated the performance of the two-step approach when different process models were used. Since the noiseless performance of the two-step approach varied in these simulations with changes in the process model, it follows that the behaviour of the two-step approach under noisy conditions should vary as well depending on the process model. Therefore, the use of three different process models in this comparison should produce interesting results.

Model 1 is described by the following parameters: $[\nu_1, \nu_2, E_1^a, E_2^a] = [1.21 \times 10^7, 7.17 \times 10^{11}, 7207, 10249]$. Note that this is the model that has been used to this point in Chapter 4. Since the two-step approach is run in the following test using the pre-exponential factors as the adjustable parameters, changes in the activation energies, specifically that of reaction 1 (E_1^a), are considered. In Model 2, E_1^a is changed to 6707 and in Model 3 E_1^a is changed to 7707. Simulations of 500 iterations are run for each of the three models, and the results are displayed in Figure 4.8 and Table 4.5. Note that approximations of each of the design cost components are not made here for sake of brevity.

Table 4.5: Simulation results for two-step approach with different process models

	Design 1	Design 2	Design 3
Average Profit	187.34	182.01	186.66

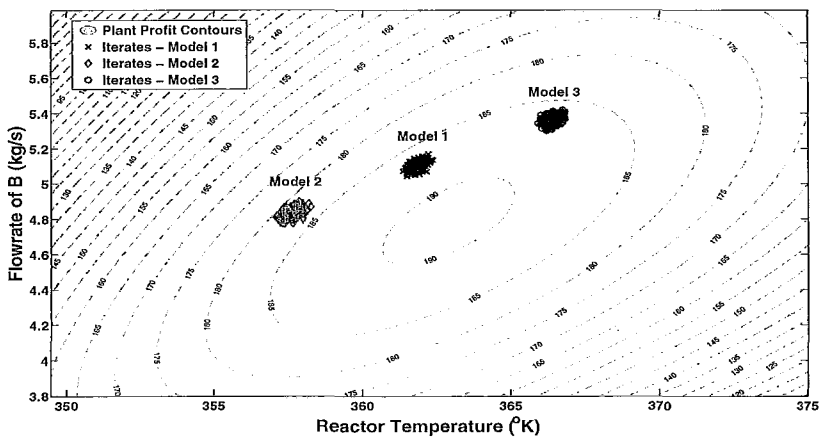

Figure 4.8: Simulations of the two-step approach with different process models

For comparison purposes, dual modifier adaptation simulations for Models 2 and 3 are run, consisting of 2000 iterations. The same settings that were used in the previous dual modifier adaptation simulations in this chapter were used again here. Note that before running each of these simulations, the design cost optimization procedure was run to select the values for the dual parameters. The results, along with those previously reported for Model 1, are given in Table 4.6 and Figure 4.9.

Table 4.6: Simulation results for dual modifier adaptation with different process models

	Model 1	Model 2	Model 3
b_1	36.97	42.98	30.49
b_2	7.97	11.87	5.33
Average Profit	185.04	186.05	180.53

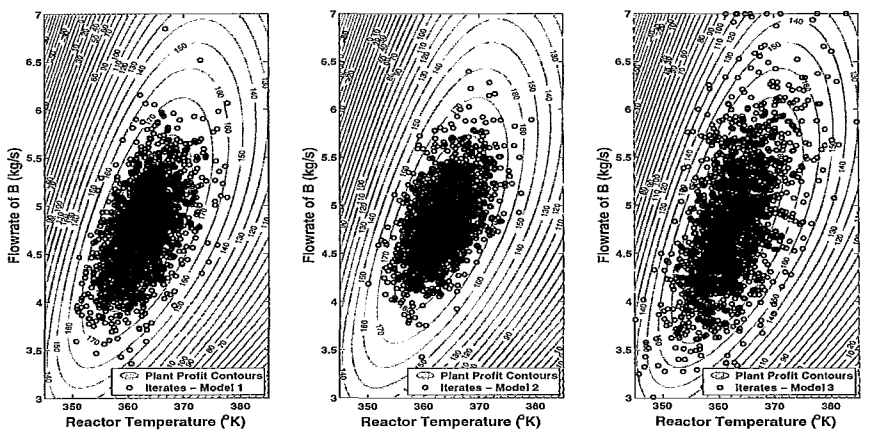

Figure 4.9: Simulations of dual modifier adaptation with different process models

First of all, the core difference between the two approaches is clear from Figures 4.8 and 4.9. The very compact set of iterates for each of the two-step approach simulations indicates that it is considerably more resistant to measurement noise than dual modifier adaptation is. However, while none of the two-step simulations in Figure 4.8 are able to move to the area of the plant optimum, each of the modifier adaptation simulations in Figure 4.9 are able to do so. The results of Tables 4.5 and 4.6 indicate that the two-step approach is

the best methodology if Models 1 and 3 are used and dual modifier adaptation is the best methodology if Model 2 is used.

The purpose of this comparison was not to draw conclusions as to which approach is better. Instead, from the results it should be clear that the choice of methodology should be dependent on the particular choice of process model for the RTO system, among other factors. This underscores the importance of a good RTO design procedure. Furthermore, the clear illustration of the core difference between the two approaches in Figures 4.8 and 4.9 motivates the possible combination of these methodologies in the future. This is discussed in more detail as a future research direction in Section 6.2.4.

Now, a situation is considered where, in addition to sizing the ellipsoid, variations in the activation energy of the second reaction in the plant are also considered. It is assumed that this reaction is catalyzed, and the catalyst is decaying over the course of the operating day. To capture these variations, three representative values for the activation energy of the reaction (E_{a_2}) are chosen: 8333, 8433 and 8533 J/mol. The same filter parameter and measurement noise settings that were used above are used again here. Only the modifiers for X_p are updated (as in design 2). Details of the design cost calculation for each individual parameter set are shown in Table 4.7.

Table 4.7: Design cost calculation details for each individual parameter set

	$E_2^a = 8333$ J/mol	$E_2^a = 8433$ J/mol	$E_2^a = 8533$ J/mol
$\Phi^p(\mathbf{u}^{p,*}, \mathbf{y}^{p,*})$	−190.798	−111.718	−35.362
C_B	0.0158	0.02076	0.0284
C_V	5.6974	5.40638	5.09848
C	5.7132	5.4271	5.1269
b_1	37.037	41.3182	45.7264
b_2	7.897	7.9423	8.1631

The change in the activation energy of the second reaction does in fact cause minor changes in the design cost results. First of all, the second reaction produces both E and P, the two

products with economic value. Therefore, increasing the activation energy of the reaction causes a significant loss of profit (increase in the cost function). The catalyst decay also causes a decrease in the variance cost of the system. The decrease in variance cost generally indicates a decrease in the effect of measurement noise on the optimal solution. Therefore, the ellipsoid does not need to be as large as the catalyst decays.

Using the scenario averaging approach of Equation 4.4 the following dual constraint parameters are recommended for implementation: $\mathbf{b} = [41.00, 8.00]$, with a corresponding design cost of $C = 5.410$. Note that equal weighting was given to each scenario here. To test out the effectiveness of the ellipsoid parameter choice, the plant profit is compared to the actual profit in Figure 4.10. The simulation consisted of 100 iterations performed at each of the 3 activation energy levels, with the same settings that were considered in the design cost calculation. The algorithm performs well at each activation energy setting, once it adjusts to the corresponding change in the plant. Note that in practice the activation energy would not change suddenly, rather it would decay over the course of the operating day. Therefore, the performance loss due to the abrupt changes in activation energy would not be a problem.

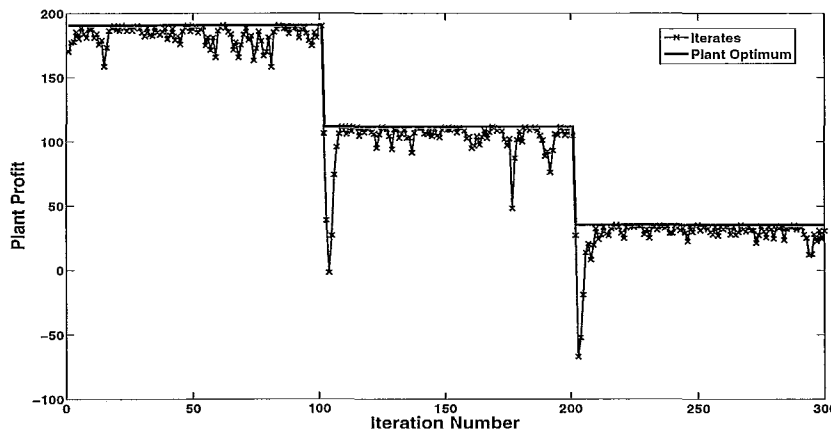

Figure 4.10: Simulation of dual MA performance at different activation energy settings

Since the dual parameters are very close to the optimal parameters recommended for each individual case (Table 4.7), the good performance seen in all scenarios in Figure 4.10 is expected. It would be interesting to see the performance if the scenarios involved vastly

different operation. This could be the topic of a future case study, possibly where larger changes are made in E_2^a.

4.4.3 Design Cost - Constrained Formulation

An example designed to illustrate how the design cost criterion can be practically applied to a constrained problem is now presented. The Williams-Otto Reactor case study is modified, as described in Section 2.7, by adding a requirement that the outlet mass fraction of component B (X_B) must be below 0.35. Note that due to the introduction of the constraint, X_B also needs to be modified in order for the system to converge to the plant optimum.

Three different updating scenarios will again be tested, modifying only X_E and X_B (design 1), modifying only X_P and X_B (design 2) and modifying all three mass fractions (design 3). The trials assume filter parameters of 0.25 and white noise in the composition measurements with a standard deviation of 0.5% of a nominal set of their values. Table 4.8 summarizes the design cost estimates for the 3 designs under consideration.

Table 4.8: Breakdown of design cost estimates (constrained system)

	Design 1	Design 2	Design 3
C_B	11.799	6.265	5.996
C_V	0.537	4.715	5.009
C_T	12.336	10.980	11.005

Table 4.8 indicates that, similar to the unconstrained case, the design cost criterion predicts that designs 2 and 3 will perform in a similar manner. Modifying X_E and X_B only (design 1) appears to cause the algorithm to move toward a point far away from the plant optimum (indicated by the large bias cost). The estimated ellipsoid size parameters are given in Table 4.9.

Both design 2 and design 3 have similar estimated ellipsoid sizes. This makes sense given that their bias and variance cost estimates are similar as well. Design 1 has a much smaller

Table 4.9: Ellipsoid sizes for different modifier combinations (constrained system)

	Design 1	Design 2	Design 3
b_1	2049	79.49	98.01
b_2	90.10	29.54	24.82

recommended ellipsoid size. This is likely because measurement noise does not effect design 1 as much as the others (as indicated by the much lower variance cost approximation), so the ellipsoid does not need to be as protective.

For each of the three approaches, as a part of the design cost optimization procedure, the probability requirement for the constraint was set at 95%. This necessitated the estimation of an appropriate back-off from the constraint. The resulting backoff parameters are given in Table 4.10.

Table 4.10: Backoff parameters for different modifier combinations (constrained system)

	Design 1	Design 2	Design 3
Backoff Parameter	0.00198	0.00369	0.00383

The estimated backoff parameter for design 1 is the smallest. This is consistent with the low variance cost prediction for this design, because a low variance cost tends to indicate that a design is not greatly effected by measurement noise, which is the uncertainty that the backoff is trying to compensate for in the first place. Designs 2 and 3 have similar backoff parameters, with the one for design 3 being slightly larger, possibly due to the extra modifiers causing an increase in the effect of measurement noise on the algorithm.

Simulations were run for each of these options. In each simulation run, the online algorithm was run for 2000 iterations, with the presence of measurement noise, and the average optimal cost was reported. The same settings that were used for the design cost approximation were used again here. In addition, the recommended backoff parameters, given in Table 4.10, were implemented for each design. The results shown in Table 4.11 and Figure 4.11 were obtained.

Table 4.11: Simulation results for different modifier combinations (constrained system)

	Design 1	Design 2	Design 3
Average Lost Profit	12.455	5.77	5.89
C_B	11.29	3.27	3.21
C_V	0.32	2.68	2.72
C	11.61	5.93	5.95
% Feasibility	90.6	95.5	95.6

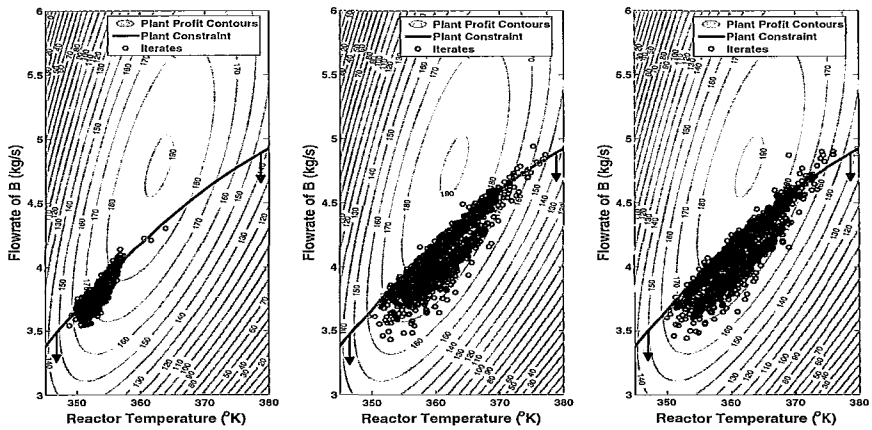

Figure 4.11: Simulations for designs 1-3 - constrained formulation

The left panel of Figure 4.11 verifies that the reason for the high bias cost approximation for design 1 is that it fails to move to the vicinity of the plant optimum. The low variance cost for this approach is also evident, in that the region of points is much smaller for this design than for the other two.

The central plot in Figure 4.11 illustrates that modifying the mass fractions of B and P instead of the mass fractions of B and E, causes a large increase in the variance of the iterates ($\mathbf{u}_\infty$). This is consistent with the large increase in variance cost predicted by the design cost formulation. For design 3, as the right panel of Figure 4.11 illustrates, modifying all three mass fractions performs very similarly to modifying the mass fraction of components B and P only.

The design cost formulation significantly over-estimates the overall profit lost for both designs 2 and 3 (Table 4.11). To identify potential causes of this discrepancy, a closer look is taken at the differences between the design cost approximation and the simulation results for design 3. Table 4.12 illustrates some of the key differences.

Table 4.12: Detailed design cost comparison for full dual MA (constrained system)

	Design Cost Approximation	Simulation
$\mathsf{E}[\phi(\mathbf{u}_\infty, \mathbf{y}^p(\mathbf{u}_\infty))]$	-174.93	-180.05
$\mathsf{E}[\mathbf{u}_\infty]$	$\begin{bmatrix} -0.1094 \\ -0.953 \end{bmatrix}$	$\begin{bmatrix} -0.1035 \\ -0.900 \end{bmatrix}$
$\mathsf{V}[\mathbf{u}_\infty]$	$\begin{bmatrix} 0.0702 & 0.0632 \\ 0.0632 & 0.0820 \end{bmatrix}$	$\begin{bmatrix} 0.0392 & 0.0385 \\ 0.0385 & 0.0496 \end{bmatrix}$
% Feasibility	95.0	95.6

The design cost approximation over-estimates the profit lost by about 5.12 ($5120). One reason for this is it over-estimates both the u_1 and u_2 components of the distance that the expected value of the iterates moves from the ideal modifier adaptation optimum ($\mathbf{u}^*_\infty$), due to the dual excitation constraint. This is demonstrated by the fact that $\mathsf{E}[\mathbf{u}_\infty]$ is farther away from the plant optimum for the design cost approximation that the simulations. Some of this error may also be due to an over-estimation of the required back-off, however, since the back-off distance is quite small relative to the distance of the offset (not shown in Table 4.12), most of the error is likely due to the over-estimation of the offset.

Each of the variance and covariance quantities are also widely over-estimated by the design cost procedure. One reason for this could be the fact that, as previously discussed, that the interaction between the measurement noise and dual excitation is ignored when they are treated separately and then their effects are summed.

The backoff parameter estimation seemed to be very effective. It was set to ensure that about 95% of iterates are feasible points of the plant, and in the simulation 95.6% of the iterates are feasible. To better visualize the effect of the backoff parameter on the iterates,

the approach where all three outputs were modified was implemented with and without the backoff parameter and simulations of 500 iterations were run.

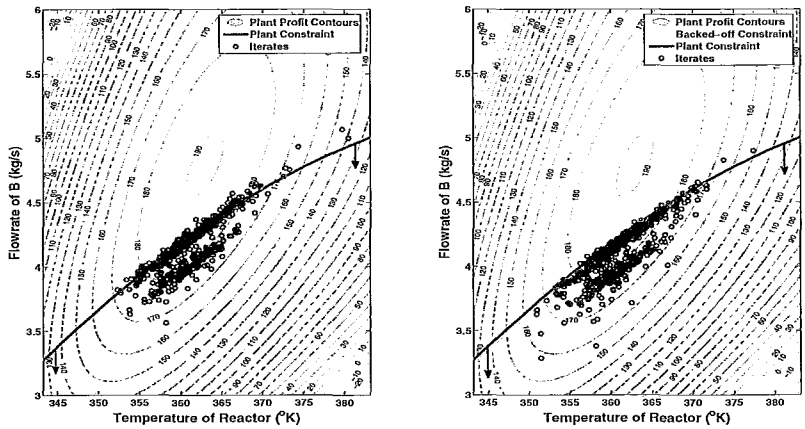

Figure 4.12: Demonstration of the effect of introducing a back-off

Shown in Figure 4.12 are the results without (left panel) and with (right panel) the backoff parameter. It is clear that the backoff causes an improvement in feasibility. In the run without the backoff (left panel of Figure 4.12), the constraint was violated 24.3% of the time, whereas in the run with the backoff applied (right panel of Figure 4.12), the constraint was only violated 4.6% of the time. As expected, this increase in feasibility came at the cost of a decrease in average profit. The average profit in the simulation without the backoff was 181.411, while the average profit in the simulation with the backoff was 180.093. This increase in profit came because of the selection of points that were infeasible in terms of the plant. Since such points can have profit values higher than the profit at $\mathbf{u}^{p,*}$, they can cause an increase the average profit. Once again the severity of the consequences if an infeasible operating point of the plant is implemented should determine whether the backoff approach is used, and how high the feasibility level (p_i) should be set for each constraint.

4.5 Chapter Summary

In this chapter, a methodology was presented for the design of dual modifier adaptation systems. This methodology utilized the design cost criterion along with an optimization

procedure to set the dual constraint parameters as well as any other design parameters that needed to be addressed. A multi-scenario approach was also presented for handling uncertainty in the benchmark plant model that is used in the design cost calculation.

Specifics of both the bias and variance cost calculation procedures have been discussed in detail. The variance cost calculation involved the consideration of two separate scenarios, one designed to capture the variance due to measurement noise and one designed to capture the variance caused by the dual constraint. The bias cost computation involved a procedure designed to estimate the change in the expected value of the dual modifier adaptation iterates due to the dual constraints. In addition, a method for the estimation of an appropriate level of back-off to ensure constraint feasibility was developed. The calculated back-off was also then applied in the online simulations.

All of these developments were tested using the Williams-Otto Reactor test case. These tests demonstrated the ability of the design cost approach to identify a good design for the system. There was significant error in some of the design cost approximations, however this was expected given the assumptions made in the calculation procedure. The back-off methodology also proved to be effective, greatly reducing constraint violation.

In the final chapter, the developments of the previous two chapters are illustrated using a more involved case study. This case study consists of the real-time optimization of a propane furnace.

Chapter 5

Case Study - Propane Pyrolysis Reactor

In this final chapter the dual modifier adaptation algorithm is implemented to perform the real-time optimization task for a propane furnace. At a basic level, this furnace takes a pure feed of propane, mixes it with steam and thermally cracks it to form a range of hydrocarbon products. Two of the major products are ethylene and propylene. Many of the concepts presented in the previous two chapters of this thesis will be illustrated using this case study. First, the performance of the dual modifier adaptation algorithm will be investigated, both under noiseless conditions and under the presence of measurement noise. Then the design of the an effective dual modifier adaptation system for this process will be investigated. As a part of this, a detailed comparison between the design cost approximation results and corresponding simulation results will be made.

5.1 Background and Process Description

The problem formulation and specific model for the propane furnace process considered in this chapter are taken from Contreras-Dordelly [1999]. The specific furnace model was

developed from industrial information provided to the authors of Contreras-Dordelly [1999] by M. Kutten, Combustion Engineering Simcom Inc., 1989. A simple process diagram is given as Figure 5.1 for reference.

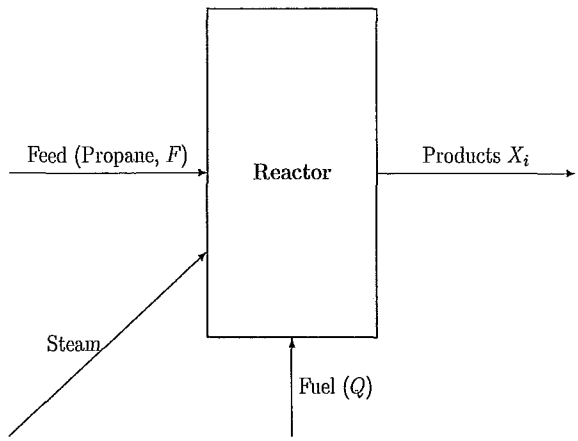

Figure 5.1: Propane furnace process schematic

The end goal of this propane cracking process is to maximize the profit from the sale of the hydrocarbons produced. In this model, there are ten different hydrocarbon products considered, all of which have an appropriate sale value. There are also costs associated with the process: the dilution steam, propane feed and the fuel used to heat the furnace all must be bought. The objective function for this process takes all of this into account:

$$\phi = \varpi_F \cdot F + \varpi_Q \cdot Q + \varpi_S \cdot SO \cdot F - \sum_{i=1}^{10} \varpi_i \cdot F \cdot X_i \tag{5.1}$$

where F is the feed flow of propane in [pounds], ϖ_F is the cost of the propane feed [\$/pound], Q is the energy consumption [MBTU], ϖ_Q is the cost of the energy [\$/MBTU], SO is the steam-to-oil dilution ratio, ϖ_S is the cost of the steam in [\$/pound], ϖ_i is the sale price of component i [\$/pound] and X_i is the weight fraction of component i in the outlet stream. Note that the time period considered in this problem is one month, therefore the feed flow, energy consumption and profit quantities are all given on a per month basis.

The objective function, ϕ, denotes the total cost of the propane cracking process. In the normal operating range of the process, however, the value of the cost function will be

negative, indicating that a profit is made off of the process. Due to this, economic values will commonly be reported in terms of profit in this chapter. **Also note that the feed rate of propane (F), the energy consumption (Q) and profit (ϕ) are assumed to be divided by 10^8, as was done in Contreras-Dordelly [1999].**

The values used for the sale prices as well as the costs of propane, energy and steam can be found in Appendix E. The subscripts for each of the ten components are given in Table 5.1.

Table 5.1: Components in the propane cracking process

Subscript	Component
1	Hydrogen
2	Methane
3	Ethylene
4	Ethane
5	Propylene
6	Propane
7	Butadiene
8	Butylene
9	Butane
10	Gasoline

Note that in the actual process there were three other products of the cracking reaction: acetylene, methyl-acetylene and propadiene. However, since downstream of this reactor in the actual process acetylene was converted to ethylene, and both methyl-acetylene and propadiene were converted to propylene, these three intermediate products are ignored in this case study.

The set-points that will be determined by the dual modifier adaptation system for this process are the feed rate of propane, F, and the conversion, denoted by C subsequently.

The conversion is defined as follows:

$$\mathcal{C} = \frac{X_6^{in} - X_6}{X_6^{in}} \tag{5.2}$$

where X_6^{in} is the weight fraction of propane in the inlet stream. Since the feed is pure propane, the following expression for conversion can be obtained:

$$\mathcal{C} = 1 - X_6 \tag{5.3}$$

It is assumed that $\mathcal{C}$ can be controlled online by adjusting the operating temperature of the furnace. Note that in the original formulation in Contreras-Dordelly [1999] there was a third input discussed, the steam-to-oil ratio SO. It is considered to be a fixed parameter in this work however. Since it was frequently at its lower bound of 0.3 in Contreras-Dordelly [1999], it is fixed at this value subsequently. A possible extension would therefore be to consider the full 3 input case study.

The model for this process considers 12 state variables, and is therefore made up of a set of 12 non-linear algebraic equations. The first ten equations of the model are empirical expressions which compute the weight fraction in the product stream (X) of each of the ten components in the furnace (Table 5.1). The other two equations calculate the energy consumption in the furnace, Q, and the average molecular weight of the product stream W. The ten empirical model equations each have the following form:

$$X_i = \varrho_{0i} + \varrho_{1i}SO + \varrho_{2i}SO^2 + \varrho_{3i}\mathcal{C} + \varrho_{4i}\mathcal{C}^2 + \varrho_{5i}(SO)\mathcal{C}, \quad i = 1, ..., 10 \tag{5.4}$$

where ϱ_{0i}, ϱ_{1i}, ϱ_{2i}, ϱ_{3i}, ϱ_{4i}, ϱ_{5i} are all parameters of the empirical model used to compute the outlet weight fraction of component i (see Appendix E). The empirical expression used to determine the furnace energy consumption (Q) is:

$$Q = 0.036\frac{F}{W} \tag{5.5}$$

and the expression used to determine the average molecular weight of the exiting stream (W) is:

$$W = \sum_{i=1}^{10} X_i MW_i \tag{5.6}$$

where MW_i is the molecular weight of pure component i [pound/pound-mol].

The model that will used as both the simulated plant and the benchmark plant model for the design cost calculations has the same form as the process model presented above. Only the set of parameters in the empirical expressions, ϱ_{0i}, ϱ_{1i}, ϱ_{2i}, ϱ_{3i}, ϱ_{4i}, ϱ_{5i} vary between them. There is therefore no structural plant-model mismatch present in this case study.

The original model parameters found in Contreras-Dordelly [1999] are used as the parameters of the simulated plant and benchmark plant models in this work and a small subset of the parameters in the process model is altered to introduce the mismatch. These alterations had to be done very carefully due to the empirical nature of the model. Specifically, care had to be taken to ensure that despite the changes, all the weight fractions determined by the model still sum up to one $\left(\sum_{i=1}^{10} X_i = 1\right)$. Due to this requirement, pairs of similar parameters were altered simultaneously. For instance, the ϱ_{43} (the parameter multiplying $\mathcal{C}^2$ for ethylene) was reduced by 0.5 in the process model and ϱ_{45} (the parameter multiplying $\mathcal{C}^2$ for propylene) was increased by 0.5. Three other similar changes were made, one more involving ethylene and propylene, and two others involving methane and ethane. Therefore, there are 8 parameters that vary between the simulated plant model and the process model.

There are a total of 11 outputs considered in this problem. They are the ten weight fractions ($\mathbf{X}$) and the energy consumption of the furnace, Q. Measurements of the weight fractions are assumed to be readily available through an online analyzer. It is assumed that an estimate of Q can be made in real-time through the rate of consumption of the fuel used to heat the furnace and a real-time analysis of the fuel gas heating value.

There are also a series of process constraints that apply to this process. First of all, there are bounds on both of the input variables. The lower bound on F is necessary due to equipment constraints and the upper bound exists due to upstream supply limitations. Both the lower and upper bounds on the conversion, $\mathcal{C}$, are necessary because the empirical model was only developed for a specific operating range. The set-points or inputs for this problem can be formally stated as: $\mathbf{u} = [F, \mathcal{C}]$. Using this, the variable bounds are then stated: $\mathbf{u}^{min} = [5, 0.70]$ and $\mathbf{u}^{max} = [15, 0.93]$.

In addition to the variable bounds, there are also two other constraints on this process. The first is an upper limit on the amount of energy that can be used by the furnace:

$$Q \leq 0.0147 \tag{5.7}$$

This constraint is required because the temperature of certain equipment in the furnace must be kept below a certain level. The last constraint is an upper bound of the amount of ethylene produced by the furnace. This constraint exists because there is a limit to the amount of ethylene that can be sold on the market. Different constraint levels are used in Contreras-Dordelly [1999], however, this constraint will be fixed in this thesis as follows:

$$X_3 F \leq 3 \tag{5.8}$$

For the actual implementation of the algorithm, some variable scaling had to be done. The reason for this is Broyden's method performs better when the general size of the movements of each of the inputs from iteration to iteration are similar (see Section 3.2.3). To this end, the feed rate of propane input ($\mathbf{u}_1$) was divided by 100 in the MATLAB code.

5.2 Dual Modifier Adaptation

In this following section, many of the concepts introduced and discussed in Chapter 3 are illustrated using the propane furnace reactor test case. This includes a comparison of the performance of ideal modifier adaptation and modifier adaptation with Broyden's method, a study of the behaviour of modifier adaptation using Broyden updates with different filter parameter settings and an illustration of the effect of the dual constraints on the performance of the algorithm.

First, before any specific tests are done, a contour plot of the plant profit surface is given in Figure 5.2. This plot shows all six constraints discussed in Section 5.1 as well as the contours of the plant profit function. The plant optimum is clearly marked with a circle. Note that this optimum lies on a constraint, specifically the ethylene demand constraint. This indicates that any change in the demand of ethylene will cause a change in the optimal

solution. Also note the feasible region of the plant (marked with the black arrows). This is the wedge-shaped area bordered by constraint 4 (the upper bound on conversion) at the top, constraint 3 on the left (the lower bound on the propane feed rate), constraint 5 at the bottom (the lower bound on conversion) and constraint 6 on the right (the ethylene demand constraint). Constraint 1 (the energy consumption limit) and constraint 2 (the upper bound on the feed rate of propane) lie completely outside the feasible region.

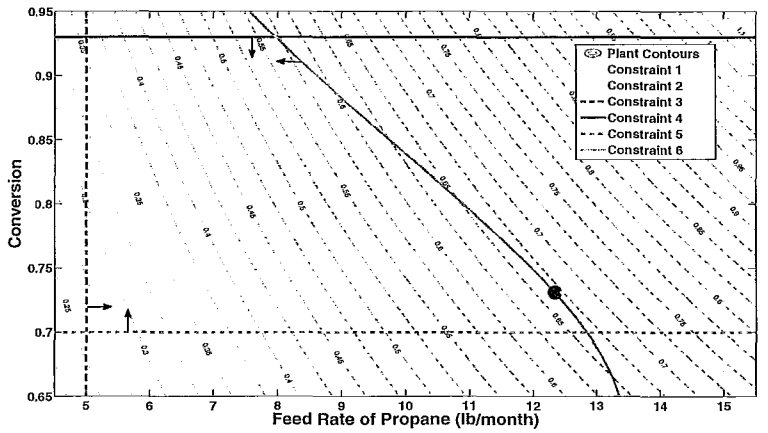

Figure 5.2: Plant profit contour plot

The behaviour of the ideal modifier adaptation algorithm, first presented in Section 2.5, is now compared to the behaviour of modifier adaptation with Broyden's method, first presented in Section 3.1. Both algorithms are run in the absence of measurement noise, utilizing the benchmark plant model given in Section 5.1 to represent the plant. The starting point for both algorithms is $\mathbf{u}_0 = [10, 0.8]$. Filter parameters of 0.4 are also used and the modifiers are initialized to zero ($\mathbf{\Lambda}_0 = \mathbf{0}$). To begin, the process model is optimized using $\mathbf{\Lambda}_0$ yielding the process model optimum: $\mathbf{u}_1 = [8.02, 0.857]$. The initial Broyden derivative matrix estimate is obtained using the derivative of the approximate model at the starting point, $\mathbf{B}_0^i = \left.\frac{\partial f^i}{\partial \mathbf{u}}\right|_{\mathbf{u}_0}$. Note that this initialization procedure will be the same for all the simulations in this chapter (unless otherwise noted). The performance of both algorithms is shown in Figure 5.3.

Both algorithms move initially at approximately the same speed. However, when the plant optimum is approached, Broyden's method takes more iterations to actually zero in on

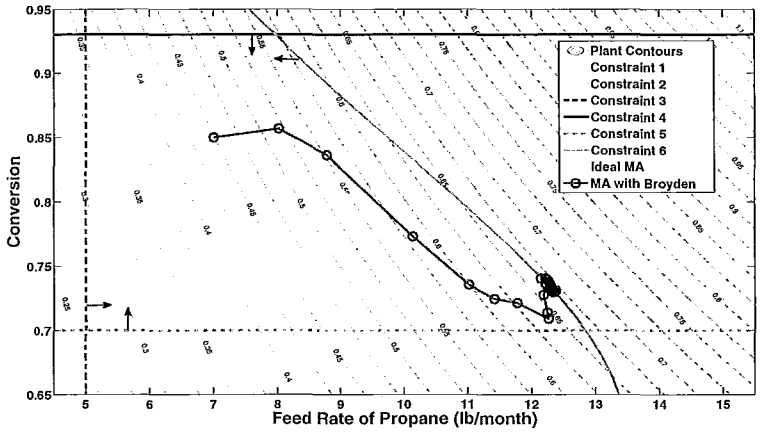

Figure 5.3: Comparison of ideal MA and MA with Broyden updates

the plant optimum. This is illustrated by the large amount of points close together in the general area of the plant optimum in Figure 5.3.

The effect of the filter parameters (**k**) on modifier adaptation with Broyden updates is examined next. For simplicity, the filter parameters for each of the 33 modifiers are always set to the same value. Note that there are 33 modifiers required because there are 11 process outputs to be modified, so there are 11 output bias modifiers and 22 output gradient modifiers that need to be updated. Figure 5.4, illustrates the performance of the algorithm with filter parameter values of 0.1, 0.4, 0.6 and 0.8. For each of the 3 lowest **k** values, the algorithm converges to the plant optimum. In those cases, as **k** increases, the algorithm takes a less direct route to the plant optimum, but also reaches it in fewer iterations. A less direct route does not by itself degrade algorithm performance, however, more erratic behaviour (larger step sizes) may increase the level of gradient offset (see Section 3.2.3).

The simulation with filter parameter values of 0.8 illustrates the problem of gradient offset that can occur in multi-input problems. In this case the algorithm reaches a point on constraint 5 very quickly. This is problematic however, because the algorithm gets stuck on this constraint, so that the Broyden derivative estimate is never updated in any direction perpendicular to this constraint. Therefore, the algorithm converges to the optimal point of the plant in the direction of the constraint (the intersection between constraints 5 and 6).

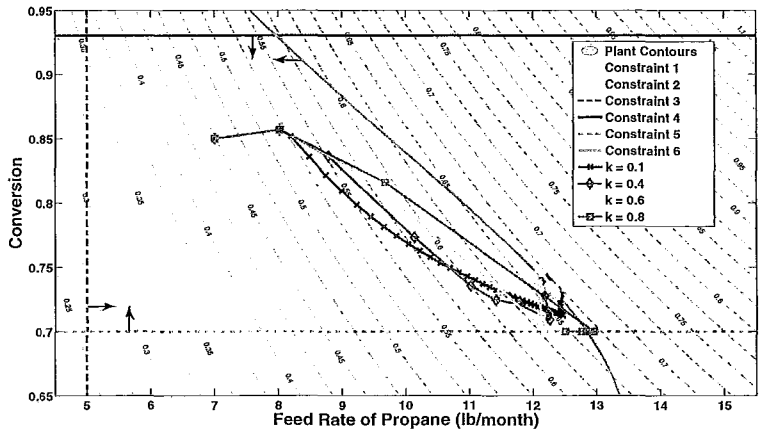

Figure 5.4: Effect of changes in **k** in modifier adaptation with Broyden updates (1)

This is clearly not the plant optimum however. Figure 5.5 is provided so that the behaviour of the algorithm near the plant optimum can be more easily seen.

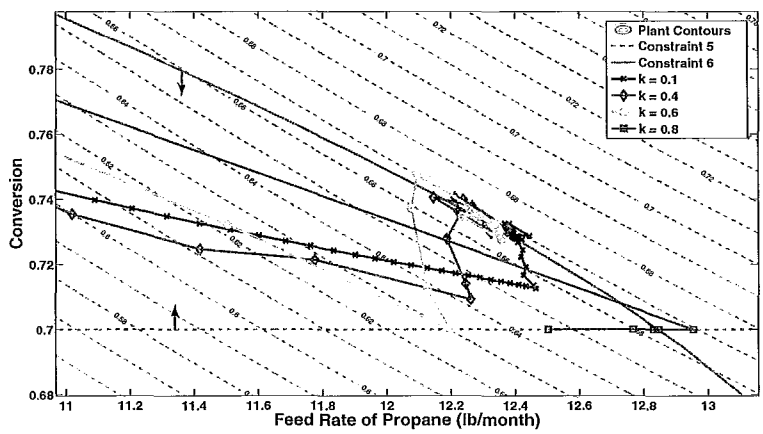

Figure 5.5: Effect of changes in **k** in modifier adaptation with Broyden updates (2)

The peaking phenomenon can occur when the algorithm is run under the presence of measurement noise, with no restriction on the placement of new operating points with reference to previous ones (see Section 3.3). To demonstrate this, modifier adaptation is run, using Broyden derivative estimates, under the presence of measurement noise. This is assumed to be white noise, with a standard deviation of 1% of a standard set of values for the weight fractions and energy consumption (Q). The filter parameters are set at 0.4 and the inputs,

modifiers and Broyden estimates are initialized as described in the first example of this section. The algorithm is run for 100 iterations and its performance is shown in Figure 5.6. It demonstrates the large jumps away from optimal operation that can occur when a new set-point is placed too close to a previous one.

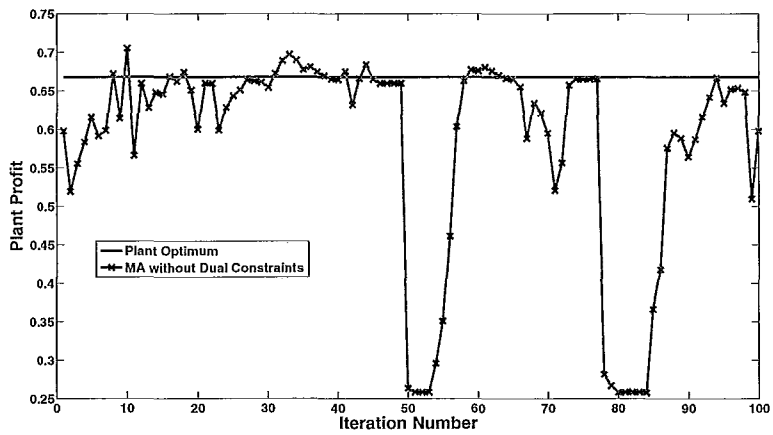

Figure 5.6: Performance of MA with Broyden updates (with measurement noise)

This motivated the development of dual modifier adaptation, which restricts the placement of a new operating point in a clever way so it is not located too close to the previous one (see Section 3.4). The dual modifier adaptation algorithm is now implemented and the simulation is rerun, with the results shown in Figure 5.7. Values of $\mathcal{B} = \text{diag}(25000, 15000)$ are chosen for the dual constraint parameters for Figure 5.7. Rough tests were done to ensure that these values provided reasonable performance, however no attempt was made to optimize them. Examining Figure 5.7 it is clear that with dual modifier adaptation there are no large jumps away from the plant optimum. This shows that the dual constraints can effectively mitigate the peaking phenomenon.

The reason that the optimal profit value can be exceeded in both Figures 5.6 and 5.7 is that the optimizer sometimes selects points that are infeasible in terms of the plant. This selection happens because measurement noise causes incorrect identification of the modifiers and since the modifiers are involved in the active process constraint, this causes it to be inadvertently relaxed in some iterations of the algorithm. This motivates the design of an

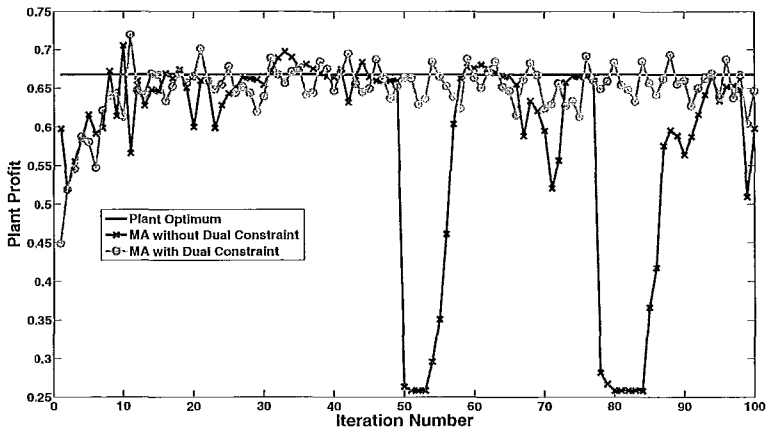

Figure 5.7: Performance of dual MA in the presence of measurement noise

appropriate constraint backoff, which is discussed in Section 5.3. Another potential cause of this, which is not addressed by the constraint backoff approach is offset in the plant output gradient estimates (see Section 3.2.3).

To close this section, an illustration and corresponding discussion of the effect of the sizing of the dual constraints on the iterates of dual modifier adaptation is included. This is meant to illustrate the trade off that exists in the sizing of the ellipsoid and also to motivate the use of a systematic dual modifier adaptation design procedure. Note that since the trust region constraint is not considered in the design procedure developed in this thesis, it will not be implemented for any of the simulations of this chapter.

The performance of a design with an ellipsoid that is too small is shown first. The parameters of this ellipsoid are $\mathcal{B} = \text{diag}(2 \times 10^6, 5 \times 10^4)$. The simulation shown in Figure 5.8 consists of 30 iterations, with measurement noise of a standard deviation of 1% of a typical set of output values and filter parameters of 0.4.

The algorithm selects operating points that are reasonably close to the plant optimum for most of the simulation in Figure 5.8, however, on a couple of occasions the iterates jump far away from the plant optimum. This is an indication that the ellipsoid is too small, because the peaking phenomenon is still occasionally occurring.

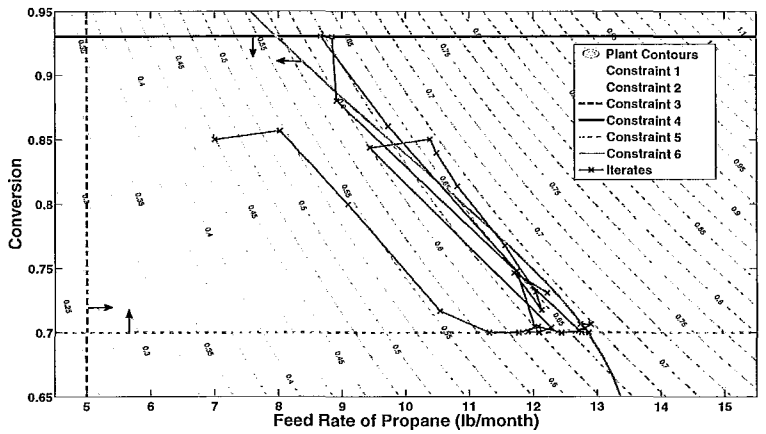

Figure 5.8: Performance of dual modifier adaptation with a small ellipsoid

If the dual constraint is too large, it can also have an adverse affect on the algorithm. To demonstrate this, another 30 iterate simulation was now run, this time with dual constraint parameters of $\mathcal{B} = \text{diag}(10000, 500)$. The results are shown in Figure 5.9. Notice that in this simulation some of the operating point moves seem to follow a similar pattern. This pattern consists of 3 operating point changes: a move directly along the constraint, followed by a move directly away from the constraint, followed finally by a move back towards the constraint. Examining Figure 5.9 closely, it can be seen a few times. This is okay from a stability point of view, however, since the dual constraint is always active, it is forcing larger than necessary input moves each iteration, which impacts the average profit of the RTO design.

The ideal ellipsoid size is one where the dual constraints are active in some of the iterations, and in others the measurement noise pushes the system far enough away from the previous operating point, so the dual constraints are inactive. Achieving this trade-off is one of the key goals of the design procedure presented in Chapter 4. An example of good performance is given in Figure 5.10 which shows a simulation utilizing the following dual constraint parameters: $\mathcal{B} = \text{diag}(25000, 15000)$.

In the simulation shown in Figure 5.10, after the first few iterations there are no movements which cause a large decrease in the plant profit (note the shape of the profit contours). Also

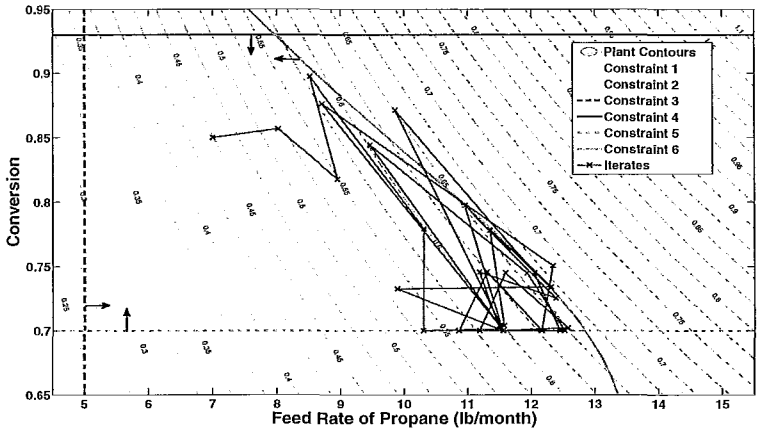

Figure 5.9: Performance of dual modifier adaptation with a large ellipsoid

note that the iterates do not follow a constant pattern and instead move somewhat randomly in the area close to $\mathbf{u}^{p,*}$. These are both indicators of good performance.

5.3 Design of Dual Modifier Adaptation Systems

The following section will discuss the design of a dual modifier adaptation system for the propane pyrolysis reactor case study. The main design decision in dual modifier adaptation systems is the size of the ellipsoid, which is the basis of the dual constraints.

First, a sample design cost calculation is made, to demonstrate how the criterion is implemented for this specific case study. The dual constraint parameters are chosen as: $\mathcal{B} = \text{diag}(25000, 15000)$, which result in relatively good, but by no means optimal system performance. Again, noise of a standard deviation of 1% of a common set of outputs, and filter parameters of 0.4 are considered. First, the Hessian of the plant cost function is computed at the plant optimum ($\mathbf{u}^{p,*} = [12.34, 0.7311]$):

$$\nabla^2 \Phi^p|_{\mathbf{u}^{p,*}} = \begin{bmatrix} 0 & -7.943 \\ -7.943 & -1.048 \end{bmatrix} \tag{5.9}$$

Note that the eigenvalues of the Hessian are -8.484 and 7.437, meaning it is indefinite. Since from the previous section it is known that the plant optimum lies on a constraint,

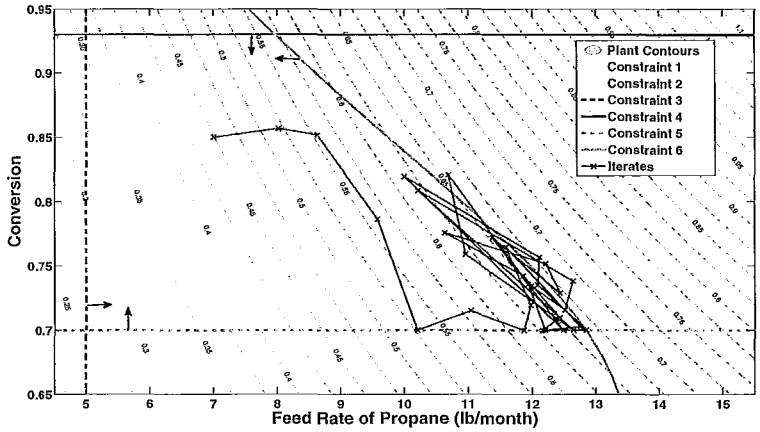

Figure 5.10: Performance of dual modifier adaptation with a well designed ellipsoid

this property of the full Hessian is not surprising. It is only the reduced Hessian that is required to be positive definite at a minimum of an optimization problem. Note that since this Hessian matrix will be used in the computation of the variance cost, it is possible that it may cause the variance cost of a particular design to be negative.

Although the ideal modifier adaptation optimum, $\mathbf{u}_{\infty}^{*}$ is equal to the plant optimum for this case study, the bias cost is not zero. This is due to the fact that both optima lie on a constraint. A procedure that estimates the change in the expected value of the iterates due to the tendency of the optimizer to select points inside the feasible region was developed in Section 4.3.2. This procedure increases C_B as it shifts the expected value of the iterates away from the ideal modifier adaptation optimum ($\mathbf{u}_{\infty}^{*}$). Another procedure that involves backing off from the ideal modifier adaptation optimum to ensure a certain level of feasibility was also developed (Section 4.3.3). This procedure also increases the bias cost as it shifts the effective ideal modifier adaptation optimum from $\mathbf{u}_{\infty}^{*}$ to $\mathbf{u}^{st}$.

A probabilistic programming approach is applied offline to compute the backoff, utilizing an estimate of $\mathbf{V}_1(\mathbf{\Lambda}_{\infty})$ giving, $\mathbf{u}^{st} = [12.06, 0.7091]$ for a 95% feasibility threshold. The required backoff in this case for constraint 6 (the constraint that is active at $\mathbf{u}_{\infty}^{*}$) is: $\eta_6 = 0.1576$.

The expected value of the iterates is then estimated, as detailed in Section 4.3.2, yielding: $\mathsf{E}[\mathbf{u}_\infty] = [11.41, 0.7091]$. The elements of the bias cost calculation procedure are shown in Figure 5.11. Note that both the calculations of $\mathbf{u}^{st}$ and $\mathsf{E}[\mathbf{u}_\infty]$ involved linearizations of parts of the algorithm. It is for this reason that $\mathbf{u}^{st}$ is not exactly on the backed-off constraint, the backed-off constraint is not on the ellipsoid and $\mathsf{E}[\mathbf{u}_\infty]$ is not on the second backed-off constraint.

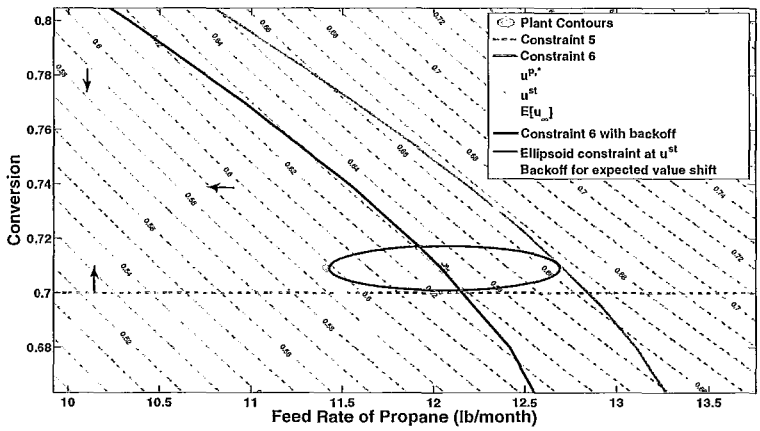

Figure 5.11: Illustration of elements of bias cost calculation

Using the benchmark plant model (the same model that was used to simulate the plant), the value of the profit at $\mathsf{E}[\mathbf{u}_\infty]$ was computed. This profit was then subtracted from the estimated profit at the plant optimum to approximate the bias cost:

$$C_B = -0.5976 - (-0.6677) = 0.0701 \tag{5.10}$$

The covariance matrix of the iterates is computed now, so that a comparison can then be made with simulation results that are presented later. An estimate of the covariance matrix is obtained as follows:

$$\begin{aligned} \mathbf{V}(\mathbf{u}_\infty) &= \mathbf{V}_1(\mathbf{u}_\infty) + \mathbf{V}_2(\mathbf{u}_\infty) \\ &= \begin{bmatrix} 1.125 \times 10^{-4} & -5.903 \times 10^{-4} \\ -5.903 \times 10^{-4} & 3.617 \times 10^{-3} \end{bmatrix} + \begin{bmatrix} 2 \times 10^{-5} & 0 \\ 0 & 3.33 \times 10^{-5} \end{bmatrix} \\ &= \begin{bmatrix} 1.325 \times 10^{-4} & -5.903 \times 10^{-4} \\ -5.903 \times 10^{-4} & 3.650 \times 10^{-3} \end{bmatrix} \end{aligned} \tag{5.11}$$

Note that the dual constraints seem to make a smaller contribution to the covariance matrix of the iterates (especially in the variance of u_2) than the measurement noise does. Intuitively this suggests that the size of the ellipsoid should be increased (especially in the direction of u_2) so there is more of an equal trade-off between the two effects.

The variance cost approximation is then computed using Equation 4.12:

$$\begin{aligned} C_V &= \frac{1}{2}1^T \left(\nabla^2 \Phi^p|_{\mathbf{u}^{p,*}} \circ \mathbf{V}(\mathbf{u}_\infty)\right) \mathbf{1} \\ &= \frac{1}{2}1^T \left(\begin{bmatrix} 0 & -7.9432 \\ -7.9432 & -1.0476 \end{bmatrix} \circ \begin{bmatrix} 1.325 \times 10^{-4} & -5.903 \times 10^{-4} \\ -5.903 \times 10^{-4} & 3.650 \times 10^{-3} \end{bmatrix} \right) \mathbf{1} \\ &= 2.78 \times 10^{-3} \end{aligned} \tag{5.12}$$

For comparison purposes, a simulation of 2000 iterations, with the same settings detailed in the design cost approximation is run. The recommended backoff is also implemented for this simulation. The numerical results are shown in Table 5.2 along with the corresponding design cost results. In addition, two plots of the iterates are given. Figure 5.12 shows the position of all 2000 iterates and Figure 5.13 shows a shorter simulation consisting of 50 iterates, in an effort to demonstrate the path traversed by the algorithm.

Table 5.2: Design cost/simulation detailed comparison

	Design Cost Approximation	Simulation
$\mathsf{E}[\phi(\hat{\mathbf{u}}_\infty, \mathbf{F}(\mathbf{u}_\infty))]$	-0.5948	-0.6184
$\mathsf{E}[\mathbf{u}_\infty]$	$\begin{bmatrix} 11.41 \\ 0.7091 \end{bmatrix}$	$\begin{bmatrix} 11.24 \\ 0.7442 \end{bmatrix}$
$\mathsf{V}[\mathbf{u}_\infty]$	$\begin{bmatrix} 1.325 * 10^{-4} & -5.903 * 10^{-4} \\ -5.903 * 10^{-4} & 3.650 * 10^{-3} \end{bmatrix}$	$\begin{bmatrix} 6.754 * 10^{-5} & -3.157 * 10^{-4} \\ -3.157 * 10^{-4} & 1.968 * 10^{-3} \end{bmatrix}$
% of Feasible Iterates	95	95.7

First note that the overall design cost approximation in Table 5.2 is fairly close to the actual average cost of the iterates in the simulation. Also, in the simulation, 95.7% of the iterates were feasible, which is close to the 95% feasibility requirement. However, upon examing the

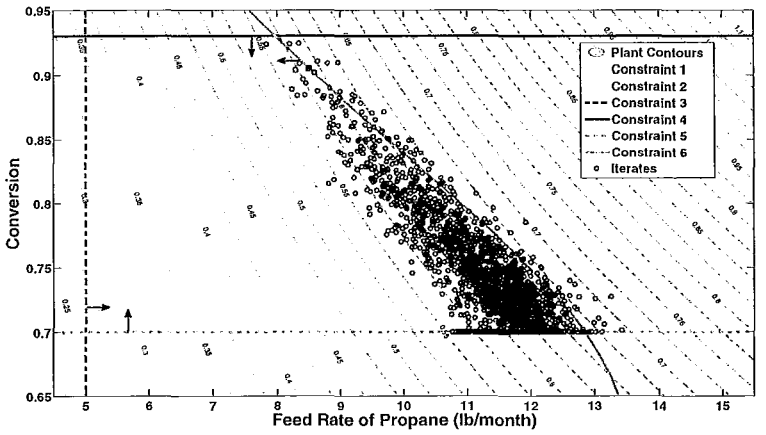

Figure 5.12: Performance of dual modifier adaptation with $\mathbf{b} = [25000, 15000]$

underlying results in Table 5.2 more closely, a couple of discrepancies between design cost approximation and the simulation become clear.

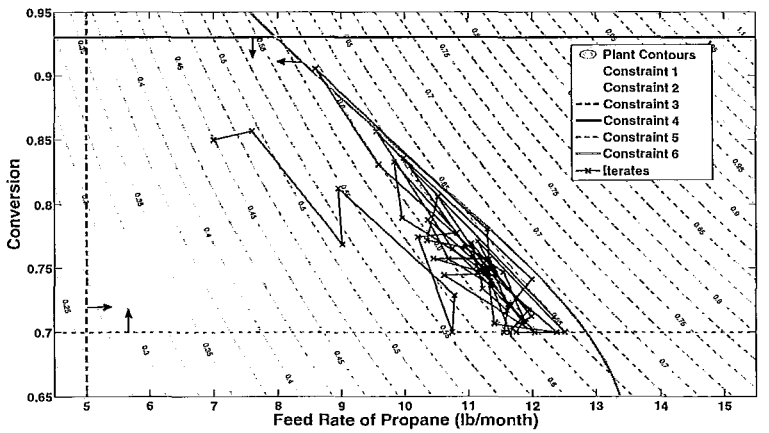

Figure 5.13: Path of dual modifier adaptation iterates with $\mathbf{b} = [25000, 15000]$

The average value of the feed rate of propane (u_1) in the simulation is reasonably close to the design cost prediction. On the other hand, the average value of the conversion (u_2) differs by 0.0351 (or 15% of the feasible region for u_2). One possible cause of the discrepancy is the presence of constraint 5 (the lower bound on conversion) in the simulation, which is not considered in the design cost approximation. This constraint, because it is very close to $\mathsf{E}[\mathbf{u}_\infty]$ does not allow for large decreases in u_2. On the other hand, constraint 4, the

upper bound on the conversion, is well above $\mathsf{E}[\mathbf{u}_\infty]$, so it rarely restricts the movement of the algorithm.

The design cost formulation over-predicts the value of each of the variance and covariance terms of $\mathbf{V}[\mathbf{u}_\infty]$. Some reasons for this over-estimation were discussed in Section 4.4.2. A further potential source of error is that the active set assumed in the variance cost calculation is not correct for some of the iterates in the simulation. This is due to the fact that constraint 5, the lower bound on the conversion, is sometimes active (Figure 5.12). This can be a major issue because the assumed active set is used to perform the post-optimal sensitivity analysis, which is used in the input covariance matrix calculation (Equation 4.24). Any error in the post-optimal sensitivity analysis can thus result in an inaccurate input covariance matrix approximation. Since constraint 5 restricts the movement of the inputs in practice, ignoring this constraint in the variance cost calculation procedure may have been part of the cause of the over-estimation.

To examine the performance of the design cost criterion at a variety of different ellipsoid sizes, pairs of design cost approximations and corresponding simulations were compiled. For simplicity, only the case where b_1 is equal to $2 \times b_2$ is considered. This is done because the algorithm was found to perform better in practice when $b_1 > b_2$. 16 different sets of parameters were considered, from $\mathcal{B} = \mathsf{diag}(5000, 2500)$ to $\mathcal{B} = \mathsf{diag}(80000, 40000)$. For each pair, the design cost calculation is made first, and then the corresponding simulation is run (5000 iterations). This is done so that the appropriate backoff can be computed through the design cost calculation and applied in the corresponding simulation. For these trials, the same settings that were used for the sample design cost calculation and simulation are used again. The results of the comparison are shown in Figure 5.14.

Examining Figure 5.14, it is clear that the design cost approximation performs quite well for this set of dual constraint parameters. Poor performance is correctly predicted for the small dual constraint parameters (large ellipsoid sizes) considered on the left and the slow decrease in performance is also well predicted as the dual constraint parameters grow larger on the right side of the plot. Most importantly, the optimum of the design cost approximation curve is fairly close to the optimum of the simulation curve. This means

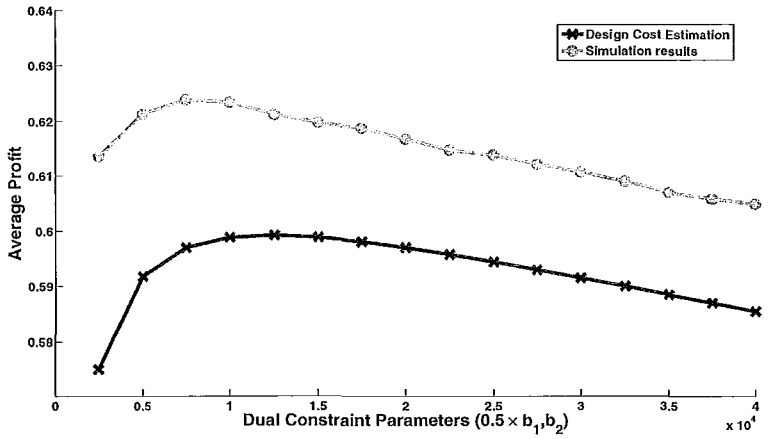

Figure 5.14: Design cost/simulation comparison

that if the optimization problem was solved for this set of dual constraint parameters, an effective set of parameters would be chosen. The under-estimation of the actual values of the average profit for all the dual constraint parameters pairs is likely due to the same reasons discussed previously for the sample calculation.

The final study of this chapter examines the effect of the level of measurement noise on the accuracy of the design cost estimates. Two new noise levels were tested: noise with a standard deviation of 0.5% of a common set of outputs and noise with a standard deviation of 2% of a common set of outputs. Design cost estimates and corresponding simulation results are shown in Figure 5.15 for the same range of dual constraint parameter values that was used in the study detailed in Figure 5.14. The results from Figure 5.14 are also included for comparison purposes.

First of all, both the design cost estimates and simulation results of Figure 5.15 illustrate that as the noise level is increased, the average profit will decrease. This makes sense, since a noise level increase would typically increase the variance of the iterates as well as the backoff required to achieve the 95% feasibility requirement. Also note that, for each individual noise level, although there is offset between the design cost estimates and the simulation results, the shape of both curves is fairly similar. This indicates that an effective set of dual constraint parameters would likely be chosen for each noise level. Upon close

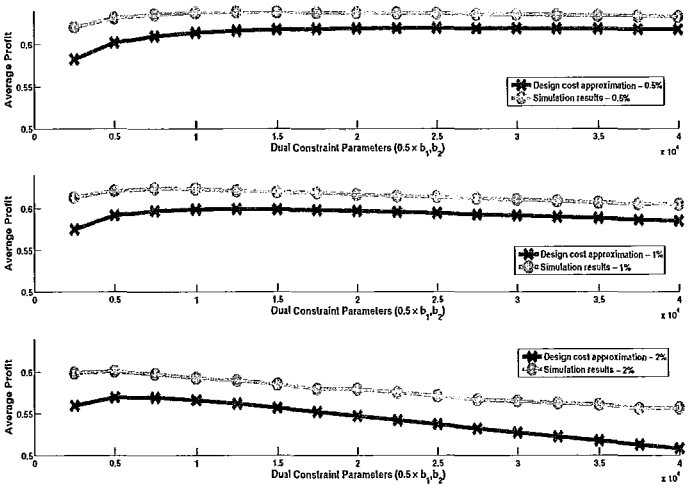

Figure 5.15: Design cost/simulation comparisons at different noise levels

examination of Figure 5.15, it is also clear that as the noise level increases, the error in the design cost estimates also increases. This is likely due to the fact that since the noise level (specifically the variance of the measurements) enters linearly into the calculation of the covariance matrix of the modifiers (Equation 4.22), any increase in noise level would likely magnify any existing errors in the modifier covariance matrix approximation. Potential sources of error have already been discussed earlier in this section.

5.4 Chapter Summary

In this chapter, the concepts introduced in Chapters 3 and 4 of this thesis were applied to a propane furnace process. This was a two input system, similar to the Williams-Otto Reactor, however the models were more complicated which caused an increase in the number of outputs to be modified. First, several investigations were carried out to demonstrate some of the properties of the dual modifier adaptation algorithm. These included studies on the effect of the filter parameters and the dual constraint parameters on algorithm performance. The design cost calculation was demonstrated for a sample ellipsoid size. Comparisons were then made with simulation results and reasons for discrepancies were suggested.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

The main goal of this research was to develop an RTO system which was both optimum seeking and resistant to measurement noise. The development of said technology was carried out in two phases. First, the algorithm itself had to be developed. The previously developed modifier adaptation approach [Marchetti *et al.*, 2009] was used as a basis, and it was made practical by employing Broyden's method to estimate the plant output gradients. Additional constraints were then added to the model-based optimization problem in order to increase the accuracy of the Broyden derivative estimates. The final algorithm was referred to as dual modifier adaptation because of its two distinct goals: optimality and the quality of the Broyden derivative estimates.

The second phase consisted of the development of a design methodology so that various tuning parameters in dual modifier adaptation could be chosen in order to achieve acceptable online performance of the algorithm. The design methodology was based around the design cost criterion, which was introduced in Forbes and Marlin [1996] for the two-step approach. This original methodology was modified to make it applicable to dual modifier adaptation and specifically so that it could address the selection of the dual constraint parameters. The

concepts presented in this thesis were illustrated using both the Williams-Otto Reactor test case and a propane furnace case study.

To begin this chapter, conclusions are drawn based on the research that has been presented in this thesis. Then, recommendations of future research directions are given.

6.1.1 Dual Modifier Adaptation Scheme

- Broyden's method was shown to be effective in estimating the plant output derivatives for the modifier adaptation RTO algorithm. This method was chosen because of the fact it requires no plant experiments to be carried out and it is easy to initialize. Its performance was demonstrated using the Williams-Otto Reactor test case (Figure 3.5).

- It was established that modifications to modifier adaptation with Broyden's method were necessary in order to achieve accurate plant output gradient estimates (see Figure 3.12). Both gradient variance and offset had to be controlled. The algorithm was analyzed to determine the causes of each of these phenomena, and appropriate solutions were presented.

  - The gradient variance can be very high because of the peaking phenomenon. The peaks occur when the algorithm is under the influence of measurement noise and two consecutive operating points are placed too close together. A constraint consisting of an ellipsoidal exclusion region, placed around the previous operating point, was added to the model-based optimization problem. This constraint proved to be very effective in eliminating the peaking problem (Figure 3.16).
  - The gradient offset was controlled in two ways. A trust-region constraint was introduced in order to limit the size of the steps taken by the algorithm in the input space. Additionally, the ellipsoidal exclusion region constraint was modified so that it forced the algorithm to explore a variety of input directions. These methods proved to be effective in reducing the offset in the Williams-Otto Reactor test case (Figure 3.19).

6.1.2 Offline Design of Dual Modifier Adaptation

- A methodology was developed in which an adapted form of the design cost criterion could be used to help design a dual modifier adaptation RTO system. This design procedure was shown to effectively address the selection of the dual constraint parameters in addition to a variety of other external design variables (such as the model to use in the RTO system or the specific outputs to modify). The approach was also successfully extended to account for uncertainty in the benchmark plant model. The methodology was demonstrated using the Williams-Otto Reactor test case (Sections 4.4.2 and 4.4.3).

- The design cost criterion was used to effectively design dual modifier adaptation systems. Changes were made to both the variance and bias cost approximation procedures to address unique aspects of dual modifier adaptation systems. The redesigned metric proved to be fairly accurate in estimating the design cost for the unconstrained Williams-Otto Reactor test case (Section 4.4.1) and the propane furnace case study (Section 5.3).

  - In the variance cost calculation, two separate scenarios were considered: one in which both of the dual constraints were always inactive, and one in which one of the dual constraints was always active. The contributions of these two scenarios were then added up to comprise the overall variance cost approximation. For the unconstrained formulation of the Williams-Otto Reactor case study the variance cost calculation procedure was reasonably effective, especially at good ellipsoid sizes. Significant error was seen in the results of the constrained Williams-Otto Reactor formulation. Potential reasons for this were discussed.

  - The bias cost calculation was modified so it could effectively address constrained problems. Specifically, a procedure was developed to compute the expected value of the iterates of a constrained problem. This was necessary because of the desire of the model-based optimizer to select feasible operating points. In addition, a constraint backoff approach was developed to try to ensure that the dual modifier adaptation iterates do not violate the process constraints of the plant. This back-

off approach proved to be fairly adept at maintaining the level of feasibility above a particular value (Figure 4.12).

6.2 Recommendations for Further Work

The following section outlines a number of promising research directions based on the work that has been presented in this thesis.

6.2.1 Convergence Analysis for Multiple Input Problems

In Section 3.2.3 it was found that convergence analysis of the modifier adaptation algorithm with Broyden updates for multiple-input problems, using a sensitivity analysis approach, is not possible. This is due to the fact that some of the derivatives of the Broyden update step are not defined as the algorithm converges. Lyapunov theory does not require differentiability, therefore it would be an ideal candidate for use in carrying out this analysis.

6.2.2 Improving and Extending the Design Cost Predictions

There is definite room for improvement in the accuracy of the design cost predictions. Several of the approximations used in the variance cost estimation procedure are fairly rough. These include the estimate of the $\frac{\partial \mathrm{B}^{i}_{k+1}}{\partial \mathrm{y}^{p,n}_{k+1}}$ sensitivity in Equation 4.23 and the estimate of $\mathbf{V}_2\left(\mathbf{u}_\infty\right)$ in Equation 4.25. If these approximations were refined, the performance of the design cost approach could be significantly improved.

A multi-scenario approach was suggested in Section 4.1.3 to handle uncertainty in the benchmark plant parameters. This method works well if there are only a few uncertain parameters. If there are a large number of scenarios to consider however, this method will become computationally challenging. Therefore, it would be ideal if there was an efficient, robust optimization method available to handle such problems. For instance, some logic could be implemented to intelligently choose a subset of the potential combinations of

parameters in order to decrease the computational requirements of the problem. It also may be advantageous to develop a procedure that could consider probability distributions of uncertain parameters, instead of only discrete sets of parameter values.

A couple of extensions to the design cost formulation are also possible. In this thesis no attempt was made to tune the trust region constraint parameters (Equation 3.46) or the filter parameters (Equation 3.1). The tuning of both of these quantities involve clear trade-offs, which motivates the need to carefully consider their respective settings. For instance, a larger trust region constraint would allow quicker movement to a new optimum, if the plant optimum for an RTO system were to change. However, the smaller the trust region constraint is, the more it restricts large, unwanted movements away from the current plant optimum, caused by measurement noise. The trade-off in the filter parameter values is quite similar. Higher filter parameter values would also allow quicker responses to disturbances causing a change in the optimal operating point of the system, but on the other hand would magnify the effect of measurement noise, possibly leading to instability of the system.

Consideration of both of these quantities would necessitate extending the design cost approximation method so it could address the transient behaviour of an RTO system (i.e. when an unmeasured disturbance causes the plant optimum to change). The extended design cost metric, developed in Zhang and Forbes [2000], could potentially be used as a starting point. The original formulation would likely have to be modified somewhat though to address dual modifier adaptation specifically.

6.2.3 Results Analysis for Dual Modifier Adaptation

The concept of results analysis was discussed briefly in Section 2.1. Recall that it is an optional part of an RTO implementation which is generally tasked with deciding whether or not to implement a newly computed operating point in the plant. This decision is made immediately following the solution of the model-based optimization problem. In making this decision, the results analysis system is trying to distinguish between common cause variation (e.g. measurement noise) and actual process disturbances.

Miletic and Marlin [1998a] proposed a statistical test for results analysis that was designed to be used within the two-step approach. This method could potentially be adapted in order to carry out the results analysis task for dual modifier adaptation systems. Such a procedure, if well designed, could prevent a great deal of unnecessary algorithm movement once the vicinity of the plant optimum is reached.

To further motivate this potential research direction, a first implementation of the results analysis procedure detailed in Miletic and Marlin [1998a] was carried out for a dual modifier adaptation system. The unconstrained Williams-Otto Reactor case study was considered, with filter parameter settings of 0.25 and measurement noise with a standard deviation of 0.5% of a typical set of mass fraction values. The modifiers for both X_E and X_P were updated and the optimal dual constraint parameters of $\mathbf{b} = [36.97, 7.97]$ were used. The algorithm was started far away from the plant optimum on three occasions and each time it was run for 250 iterations. The results are shown in Figure 6.1.

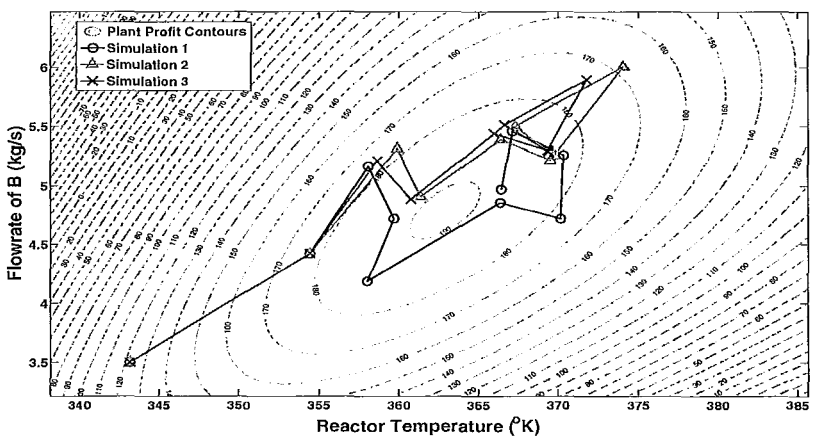

Figure 6.1: Performance of results analysis implementation

In each trial in Figure 6.1, the algorithm moved quickly toward the plant optimum in the first few iterations and then made very few additional operating point changes. The benefit of the results analysis procedure is therefore clearly seen, in that many unnecessary operating point changes were prevented. Note again that this is just meant as a motivating example. Further research is required to not only tune the results analysis procedure itself,

but also study its effect on both the steady-state behaviour of the dual modifier adaptation algorithm, as well as its ability to respond to disturbances.

6.2.4 Hybrid RTO Technologies

The relevant advantages and disadvantages of both the two-step approach and dual modifier adaptation have been detailed at length throughout this report. To summarize, the two-step approach can fail to locate the plant optimum if structural model mismatch exists [Biegler *et al.*, 1985], but is on the other hand fairly resistant to measurement noise. Dual modifier adaptation on the other hand is optimum seeking, but can be very susceptible to measurement noise due to the need to estimate the plant output gradient. It is these two key differences that motivates the investigation of hybrid approaches. A comparison that illustrates these differences was provided in Section 4.4.2.

It was established in Sections 4.4.2 and 4.4.3 that dual modifier adaptation can be effectively carried out with only a subset of the outputs being modified. The two-step approach can be altered in the same way, as removing outputs just means that the number off terms in the objective function of the parameter estimation problem (Equation 2.2) will decrease. Since both approaches can function without the full set of possible outputs, it makes sense to hybridize them. In this way, some of the outputs would be used to update a set of parameters and others would be used to update a set of modifiers. These new parameters and modifiers could then both be used to simultaneously alter the model-based optimization problem.

The ability to combine the two-step approach and modifier adaptation in this way was one of the motivations for choosing to modify the outputs directly instead of both the cost and constraint functions in modifier adaptation (see Section 2.5 for more details). Design of these hybrid RTO systems could easily be carried out by adapting the design cost approach detailed in Chapter 4 and combining it with the original design cost procedure for the two-step approach [Forbes and Marlin, 1996].

A short example intended to demonstrate the potential of these hybrid RTO systems is given hereafter. The unconstrained formulation of the Williams-Otto Reactor case study is considered and it is assumed that measurements of the mass fractions of components A, B, E, G and P are available. Four different hybrid methods are considered, which are detailed in Table 6.1.

Table 6.1: Hybrid method details

	β	Outputs for Λ	Outputs for β	Dual Constraint Params
Method 1	ν_1, ν_2	X_E, X_P	X_A, X_B, X_G	[36.97, 7.97]
Method 2	E_{a_1}, E_{a_2}	X_E, X_P	X_A, X_B, X_G	[36.97, 7.97]
Method 3	ν_1, ν_2	X_P	X_A, X_B, X_E, X_G	[37.04, 7.90]
Method 4	E_{a_1}, E_{a_2}	X_E	X_A, X_B, X_P, X_G	[789.6, 123.3]

Each hybrid methodology is run for 2000 iterations, in the presence of measurement noise of standard deviation of 0.5% of a typical set of mass fraction values and with filter parameters of 0.25. The ellipsoids sizes used are given in Table 6.1. These sizes are the ones estimated with the design cost optimization procedure for the corresponding dual modifier adaptation algorithm. These are likely not optimal, however, since the design cost criterion has not been adapted to consider hybrid approaches, there is currently no systematic approach available for estimating the best sizes. The average cost values are shown in Table 6.2 and plots of the iterates for each method are shown as part of Figure 6.2.

Table 6.2: Hybrid method results

	Method 1	Method 2	Method 3	Method 4
Average Lost Profit	4.855	5.149	4.790	4.23

Examining Table 6.2, it is evident that each of the hybrid methods are viable RTO approaches. Although none of these methods were tuned precisely, they still all out-performed the dual modifier adaptation designs detailed in Tables 4.1-4.3 (although none of the methods were able to out-perform the two-step method application for model 1 - Table 4.5).

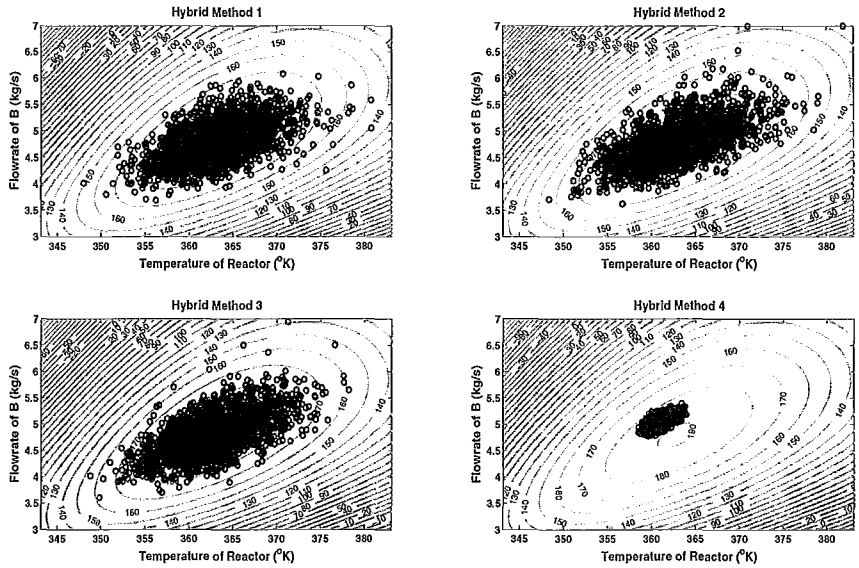

Figure 6.2: Performance of hybrid approaches

Method 4 performs especially well compared to its counterparts. This makes sense because modifying X_P has been shown to amplify the effect of measurement noise on the system, much more than modifying X_E does. So in this effective design, the output that caused dual modifier adaptation to be very susceptible to measurement noise (X_P) was used in the two-step approach and the other output (X_E) was used in dual modifier adaptation.

Another benefit of hybridization is that it might greatly improve acceptance of the modifier adaptation technology by those industries where the two-step approach is already well accepted. This is due to the fact that if modifier adaptation is introduced in this way, it can be seen as a small deviation from the already trusted two-step approach. For instance, modifier adaptation could be slowly introduced, by at first only updating the modifiers for a single output.

In closing, the good performance of all of the hybrid methods on the Williams-Otto Reactor test case motivates further research in this area. Although the technology itself is easy to implement, a design procedure would need to be developed, potentially based on the design cost approach detailed in Chapter 4 of this thesis. In addition, depending on the number

of measurements available, there could be hundreds or even thousands of discrete design options for a larger problem. Thus, a systematic optimization method to choose the best discrete design may need to be developed, to avoid considering all possible discrete designs.

List of References

Antsaklis, P. and Michel, A. (2007). *A Linear Systems Primer.* Birkhauser.

Arras, K. (1998). An introduction to error propagation: Derivation, meaning and examples of equation $\mathbf{C}_y = \mathbf{F}_x \mathbf{C}_x \mathbf{F}_x^T$. Technical report, École Polytechnique Fédérale de Lausanne.

Bailey, J., Hrymak, A., Treiber, S., and Hawkins, R. (1993). Nonlinear optimization of a hydrocracker fractionation plant. *Computers and Chemical Engineering*, **17**, 123–138.

Biegler, L., Grossman, I., and Westerberg, A. (1985). A note on approximation techniques used for process optimization. *Computers and Chemical Engineering*, **9**, 201–206.

Brdys, M. and Tatjewski, P. (1994). An algorithm for steady-state optimizing dual control of uncertain plants. In *Proc. 1st IFAC Workshop: New Trends in Design of Control Systems, Smolenice, Slovakia*, pp. 249–254.

Broyden, C. (1965). A class of methods for solving nonlinear simultaneous equations. *Mathematics of Computation*, **19**, 577–593.

Chachuat, B., Marchetti, A., and Bonvin, D. (2008). Process optimization via constraints adaptation. *Journal of Process Control*, **18**, 244–257.

Chachuat, B., Srinivasan, B., and Bonvin, D. (2009). Adaptation strategies for real-time optimization. *Computers and Chemical Engineering*, **33**, 1557–1567.

CHAPRA, S. C. (2008). *Applied Numerical Methods with MATLAB - 2nd Edition.* McGraw-Hill.

CHEN, C. AND JOSEPH, B. (1987). On-line optimization using a two-phase approach: An application study. *Industrial Chemical Engineering Research*, **26**, 1924–1930.

CONTRERAS-DORDELLY, J. AND MARLIN, T. (2000). Control design for increased profit. *The Canadian Journal of Chemical Engineering*, **24**, 267–272.

CONTRERAS-DORDELLY, J. L. (1999). Steady-state process optimization under uncertainty. Master's Thesis, Department of Chemical Engineering, McMaster University.

CUTLER, C. R. AND PERRY, R. T. (1983). Real time optimization with multivariable control is required to maximize profits. *Computers and Chemical Engineering*, **7**, 663–667.

DARBY, M. AND WHITE, D. (1988). On-line optimization of complex process units. *Chemical Engineering Progress*, **84**, 51–59.

FIACCO, A. V. (1983). *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming.* Academic Press.

FORBES, J. AND MARLIN, T. (1994). Model accuracy for economic optimizing controllers: The bias update case. *Industrial Engineering and Chemistry Research*, **33**, 1919–1929.

FORBES, J., MARLIN, T., AND MACGREGOR, J. (1994). Model adequacy requirements for optimizing plant operations. *Computers and Chemical Engineering*, **18**, 497–510.

FORBES, J. F. AND MARLIN, T. E. (1996). Design cost: A systematic approach to technology selection for model-based real-time optimization systems. *Computers and Chemical Engineering*, **20**, 717–734.

FRALEIGH, L., GUAY, M., AND FORBES, J. (2003). Sensor selection for model-based real-time optimization: relating design of experiments and design cost. *Journal of Process Control*, **13**, 667–678.

FRANCOIS, G., SRINIVASAN, B., AND BONVIN, D. (2005). Use of measurements for enforcing the necessary conditions of optimality in the presence of constraints and uncertainty. *Journal of Process Control*, **15**, 701–712.

GAO, W. AND ENGELL, S. (2005). Iterative set-point optimization of batch chromatography. *Computers and Chemical Engineering*, **29**, 1401–1409.

GOVINDARAJAN, L. AND KARUNANITHI, T. (2004). Design optimization of process plants using real coded genetic algorithm. *Chemical and biochemical engineering quarterly*, **18**, 353–361.

KADAM, J., SCHLEGEL, M., SRINIVASAN, B., BONVIN, D., AND MARQUARDT, W. (2007). Dynamic optimization in the presence of uncertainty: From off-line nominal solution to measurement-based implementation. *Journal of Process Control*, **17**, 389–398.

KHALIL, H. (2001). *Nonlinear Systems*. Prentice-Hall.

KRISHNAN, S., BARTON, G., AND PERKINS, J. (1992a). Robust parameter estimation in on-line optimization - part 2. application to an industrial process. *Computers and Chemical Engineering*, **16**, 545–562.

KRISHNAN, S., BARTON, G., AND PERKINS, J. (1992b). Robust parameter estimation in on-line optimization - part i. methodology and simulated case study. *Computers and Chemical Engineering*, **16**, 545–562.

LOEBLEIN, C. AND PERKINS, J. (1998). Economic analysis of different structures of on-line process optimization systems. *Computers and Chemical Engineering*, **22**, 1257–1269.

MANSOUR, M. AND ELLIS, J. E. (2003). Comparison of methods for estimating real process derivatives in on-line optimization. *Applied Mathematical Modelling*, **27**, 275–291.

MARCHETTI, A., CHACHUAT, B., AND BONVIN, D. (2009). Modifier-adaptation methodology for real-time optimization. *Industrial and Engineering Chemistry Research*, **48**, 6022–6033.

MARCHETTI, A. G. (2009). *Modifier-Adaptation Methodology for Real-Time Optimization*. PhD Thesis, École Polytechnique Fédérale de Lausanne.

MARLIN, T. AND HRYMAK, A. (1997). Real-time operations optimization of continuous processes. fifth international conference on process control. In *American Institute of Chemical Engineering Symposium Series*, 93, p. 156.

MILETIC, I. AND MARLIN, T. (1998a). On-line statistical results analysis in real-time operations optimization. *Industrial Engineering and Chemistry Research*, **37**, 3670–3684.

MILETIC, I. AND MARLIN, T. (1998b). Results diagnosis for real-time process operations optimization. *Computers and Chemical Engineering*, **22**, S475–S482.

PETERSEN, K. B. AND PEDERSEN, M. S. (2008). The matrix cookbook. Version 20081110.

RARDIN, R. L. (1998). *Optimization in Operations Research.* Prentice-Hall.

ROBERTS, P. (1979). Algorithm for steady-state system optimization and parameter-estimation. *International Journal of Systems Science*, **10**, 719–734.

ROBERTS, P. (1995). Coping with model-reality differences in industrial process optimisation - a review of integrated system optimisation and parameter estimation (ISOPE). *Computers in Industry*, **26**, 281–290.

ROBERTS, P. AND WILLIAMS, W. (1981). On an algorithm for combined system optimisation and parameter estimation. *Automatica*, **17**, 199–209.

SEBORG, D. E., EDGAR, T. F., AND MELLICHAMP, D. A. (2004). *Process Dynamics and Control.* Wiley.

SKOGESTAD, S. (2000). Self-optimizing control: the missing link between steady-state optimization and control. *Computers and Chemical Engineering*, **24**, 569–575.

SRINIVASAN, S. AND BONVIN, D. (2007). Real-time optimization of batch processes by tracking the necessary conditions of optimality. *Industrial Engineering and Chemistry Research*, **46**, 492–504.

STANLEY, G. AND MAH, R. (1981). Observability and redundancy in process data estimation. *Chemical Engineering Science*, **36**, 259–272.

TADEJ, W. AND TATJEWSKI, P. (2001). Analysis of an isope-type dual algorithm for optimizing control and nonlinear optimization. *International Journal of Applied Mathematics and Computer Science*, **11**, 429–457.

TATJEWSKI, P. (2002). Iterative optimizing set-point control - the basic principle redesigned. In *15th Triennial World Congress.*

TATJEWSKI, P., BRDYS, M., AND DUDA, J. (2001). Optimizing control of uncertain plants with constrained feedback controlled outputs. *International Journal of Control*, **74**, 1510–1526.

WITTENMARK, B. (1995). Adaptive dual control methods: an overview. In *Adaptive Systems in Control and Signal Processing 1995*, pp. 67–72.

YIP, W. AND MARLIN, T. (2002). Multiple data sets for model updating in real-time optimization. *Computers and Chemical Engineering*, **26**, 1345–1362.

YIP, W. AND MARLIN, T. (2003). Designing plant experiments for real-time optimization systems. *Control Engineering Practice*, **11**, 837–845.

ZHANG, Y. AND FORBES, J. (2006). Performance analysis of perturbation-based methods for real-time optimization. *The Canadian Journal of Chemical Engineering*, **84**, 209–218.

ZHANG, Y. AND FORBES, J. F. (2000). Extended design cost: a performance criterion for real-time optimization systems. *Computers and Chemical Engineering*, **24**, 1829–1841.

ZHANG, Y., MONDER, D., AND FORBES, J. (2002). Real-time optimization under parametric uncertainty: a probability constrained approach. *Journal of Process Control*, **12**, 373–389.

Appendix A

Nomenclature

Table A.1: Definitions of Latin symbols

Symbol	Definition
$\mathcal{A}$	modifier update map
$\mathbf{a}$	ISOPE model shift parameter
$\mathbf{B}$	matrix of Broyden derivative estimates
$\hat{\mathbf{B}}^i$	average values of Broyden derivative estimates
$\mathcal{B}$	matrix of dual constraint parameters
$\mathbf{b}$	vector of dual constraint parameters
C	conversion (propane furnace)
C	design cost
C_B	bias cost
C_V	variance cost
C_{avg}	average design cost
$\mathbf{c}$	vector of cost function penalty terms
$\mathbf{d}$	vector of constant terms in constraint back-off calculation
E	expected value operator
$\mathbf{E}^{val}$	eigenvalues
$\mathbf{E}_i^a$	activation energy of reaction i, ^{o}K
$\mathbf{e}$	gradient offset
$\mathbf{F}$	explicit functions describing plant operation
$\mathcal{F}$	feed rate of propane, lb
F_i	feed rate of component i, kg/s (Williams-Otto Reactor test case)
$\mathbf{f}$	explicit process model equations
$\mathbf{G}$	constraints in terms of inputs and modifiers only
$\mathcal{G}^p$	constraints in terms of inputs and plant outputs only
$\mathcal{G}^{p,a}$	active constraints in terms of inputs and plant outputs only
$\mathbf{g}$	process constraints

Table A.1: Definitions of Latin symbols (continued)

Symbol	Definition
$\mathbf{H}$	hyper-plane
$\mathcal{H}$	intermediate variable in calculation of sensitivity of $\mathcal{A}$ to previous inputs $\mathbf{u}_k$
$\mathbf{h}$	probability density function for the iterates
$\mathbf{I}$	identity matrix
$\mathbf{K}_g$	matrix used to filter newly computed constraint values
$\mathbf{K}_u$	matrix used to filter newly computed input values
$\mathbf{K}_\Lambda$	matrix used to filter newly computed modifier values
$\mathbf{k}$	vector of filter parameters
k_i^r	reaction rate constant for reaction i
$\mathcal{L}$	Lagrangian
$\mathcal{L}^m$	Lagrangian of a model-based optimization problem
$\mathcal{L}^p$	Lagrangian of a plant optimization problem
$\mathbf{M}$	subspace containing all directions in which the Broyden matrix does not change
$\mathbf{MW}_i$	molecular weight of pure component i, lb/lb-mol
M_R	mass of reactor, kg
$\mathbf{m}$	a direction in which the Broyden matrix does not change
$\mathbf{N}$	orthonormal basis for subspace orthogonal to direction of last movement
N_i	cumulative distribution function for random variable ξ_i
$\mathbf{n}$	unit direction of $\mathbf{N}$
n_u	number of inputs
n_y	number of outputs
n_g	number of constraints
n_Λ	number of modifiers

Table A.1: Definitions of Latin symbols (continued)

Symbol	Definition
P	probability operator
$\mathbf{p}$	vector of feasibility requirements for constraints
Q	energy consumption, MBTU
$\mathbf{R}$	non-linera map involving the Broyden update map
$\mathcal{R}$	Broyden update map
$\mathbf{S}$	matrix of differences between previous operating points and the current one
$\mathbf{SO}$	steam to oil dilution ratio
$\mathbf{T}$	transition function between original and alternate sets of output modifiers
T_R	temperature of reactor, K
$\mathcal{T}$	matrix of trust region constraint parameters
$\mathbf{t}$	vector representation of $\mathbf{T}(\mathbf{u}_{k+1})$
$\mathbf{U}$	matrix of vectors defining previous movement directions
$\mathbf{U}^*$	non-linear map describing model-based optimization step
$\mathcal{U}$	matrix involved in SVD of gradient of active constraints
$\mathbf{u}$	manipulated inputs or operating points
$\hat{\mathbf{u}}$	average values of manipulated inputs or operating points
$\mathbf{u}^{p,*}$	plant optimum
$\mathbf{u}^{st}$	solution to backoff optimization problem
$\mathbf{u}_\infty$	dual modifier adaptation iterates
$\mathbf{u}^*$	solution to model-based optimization problem for ISOPE
$\mathbf{u}^*_\infty$	convergence point of ideal modifier adaptation algorithm
$\mathbf{u}'$	new operating point ignoring the effect of the dual constraints

Table A.1: Definitions of Latin symbols (continued)

Symbol	Definition
$\mathbf{u}_{a_i}$	point at maximum distance the dual constraints can move the new optimum in direction i
$\mathbf{V}$	variance operator
$\mathcal{V}$	matrix involved in SVD of gradients of active constraints
$\mathbf{v}$	random vector in derivative orientation definition
$\mathbf{W}$	average molecular weight of product stream
$\mathbf{w}$	intermediate variable in calculation of sensitivity of $\mathcal{A}$ to current inputs $\mathbf{u}_{k+1}$
X_i	mass fraction of component i, (Williams-Otto Reactor case study)
$\mathcal{X}$	matrix involved in non-linear sensitivity analysis
$\mathbf{x}$	random vector in derivative orientation definition
$\mathbf{Y}$	overall modifier adaptation algorithm with Broyden updates
$\mathcal{Y}$	matrix involved in non-linear sensitivity analysis
$\mathbf{y}^m$	model outputs
$\hat{\mathbf{y}}^m$	modified model outputs
$\mathbf{y}^p$	plant outputs
$\hat{\mathbf{y}}^p$	average values of plant outputs
$\mathbf{Z}$	null space of the active constraint set
$\mathbf{z}$	point vector from the steady-state optimum to the edge of the ellipsoid constraint, normal to an active constraint

Table A.2: Definitions of Latin symbols

Symbol	Definition
α	number of unique sets of uncertain parameter values
β	adjustable model parameters
β^p	uncertain plant benchmark model parameters
$\mathbf{\Gamma}^*$	map relating modifiers to the Lagrange multipliers of the corresponding model-based optimization problem
γ	Lagrange multipliers
$\mathbf{\Delta}$	difference between or change in operating points
δ	difference between an operating point and the steady-state optimum
ϵ	output bias modifier
$\overline{\epsilon}$	alternate output bias modifier
ϵ^b	constraint bias parameter (constraint bias updating)
ϵ^g	constraint bias modifier
ε	reciprocal condition number threshold
ζ	unit vector representing last operating point move
η_i	estimated back-off from constraint i
$\mathbf{\Theta}^t$	non-linear map between the inputs and the vector representation of $\mathbf{T}(\mathbf{u}_{k+1})$
$\mathbf{\Theta}^\tau$	non-linear map between the inputs and the vector representation of $\mathbf{T}(\mathbf{u}_k)$
θ	scalar parameter which fixes the length of point vector $\mathbf{z}$
ϑ	a specific instance of the iterates ($\mathbf{u}_\infty$)
κ	reciprocal condition number of the matrix $\mathbf{S}$
$\mathbf{\Lambda}$	full set of output modifiers
$\overline{\mathbf{\Lambda}}$	full set of alternate output modifiers
λ	output gradient modifier

Table A.2: Definitions of Greek symbols (continued)

Symbol	Definition
$\boldsymbol{\lambda}^g$	constraint gradient modifier
$\boldsymbol{\lambda}^\phi$	cost gradient modifier
$\boldsymbol{\mu}$	tuning parameters for multiple constraint approach
ν_i	pre-exponential factor of reaction i
$\boldsymbol{\xi}$	uncertainty variable in backoff approach
ϖ_i	cost or sale value of component i
$\boldsymbol{\Sigma}$	matrix involved in SVD of gradient of the active constraints with respect to the inputs
σ	singular value
ς	external design variables
τ	vector representation of $\mathbf{T}(\mathbf{u}_k)$
$\boldsymbol{\Upsilon}$	matrix linearizing relationship between previous and current key algorithm values
υ	vector of small input perturbations
$\boldsymbol{\Phi}^p$	plant cost function written only in terms of the inputs
$\boldsymbol{\Phi}^m$	model cost function written only in terms of the inputs
$\nabla\boldsymbol{\Phi}^p$	plant cost function gradient
$\nabla^2\boldsymbol{\Phi}^p$	plant cost function Hessian
ϕ	cost function
χ	space of potential input values
ψ	ISOPE parameter used to modify the cost function
$\boldsymbol{\Omega}$	approximation of the sensitivity of the Broyden derivative estimates to changes in the measurements
ω	vector orthogonal to hyper-plane $\mathbf{H}$

Table A.3: Definitions of select superscripts

Symbol	Definition
$*$	optimum
min	minimum
max	maximum
p	plant
m	model

Table A.4: Definitions of select subscripts

Symbol	Definition
k	iteration number

Appendix B

Mathematical Background

The purpose of this appendix is to describe in detail some of the mathematical concepts applied throughout this thesis. The format of this appendix is based on a similar section in Marchetti [2009].

B.1 The Plant NLP in RTO Systems

The first few demonstrations of this appendix are made in regards to the following non-linear plant optimization problem, which in general is the uncertain problem that the RTO system is trying to solve:

$$\begin{aligned} \mathbf{u}^{p,*} \in \arg\min_{\mathbf{u}} \quad & \phi(\mathbf{u}, \mathbf{y}^p) \\ \text{s.t.} \quad & \mathbf{y}^p = \mathbf{F}(\mathbf{u}) \\ & \mathbf{g}(\mathbf{u}, \mathbf{y}^p) \leq 0 \\ & \mathbf{u}^{min} \leq \mathbf{u} \leq \mathbf{u}^{max} \end{aligned} \tag{B.1}$$

In this NLP, ϕ represents the objective function, which is defined in terms of both the inputs ($\mathbf{u}$) and the plant outputs ($\mathbf{y}^p$), $\mathbf{F}$ denotes the explicit plant model, which relates the inputs to the plant outputs, $\mathbf{g}$ represents the plant inequality constraints and $\mathbf{u}^{min}$ and $\mathbf{u}^{max}$ represent the input variable bounds. Problem B.1 can easily deal with equality constraints as well, because they are simply inequality constraints that are always active.

The optimization problem (B.1) can be restated by redefining the objective function and the constraints in terms of only the inputs. This is done as follows:

$$\begin{aligned} \mathbf{u}^{p,*} \in \arg\min_{\mathbf{u}} \quad & \Phi^p(\mathbf{u}) \\ \text{s.t.} \quad & \mathcal{G}^p(\mathbf{u}) \leq 0 \end{aligned} \tag{B.2}$$

where $\Phi^p(\mathbf{u}) := \phi(\mathbf{u}, \mathbf{y}^p)$ and $\mathcal{G}^p(\mathbf{u}) := \mathbf{g}(\mathbf{u}, \mathbf{y}^p)$. Note that in Problem B.2 the input variable bounds have been removed, as the bound constraints can simply be rewritten as inequality constraints and incorporated in $\mathcal{G}^p$. For the developments in the remainder of this appendix, it is assumed that the objective function (Φ^p) and constraints ($\mathcal{G}^p$) are twice continuously differentiable with respect to the inputs.

B.2 Taylor Series Expansion

Taylor's theorem, upon which the Taylor series itself is based, essentially states that any function with continuous derivatives can be approximated by a polynomial [Chapra, 2008]. From a practical point of view, a Taylor series expansion provides an approximation of the value of a function at a certain operating point. This estimate is produced using the value of that same function at another operating point as well as some derivative information at the point where the function value is known. For instance, say the value of the constraint g is known at the current operating point, u_k. The value of that same constraint at a potential new operating point, u_{k+1} can then be approximated in the following way:

$$g(u_{k+1}) = g(u_k) + \sum_{i=1}^{n_t} \frac{1}{(i)!} \nabla_u^{(i)} g\Big|_{u_k} (u_{k+1} - u_k)^i + o\left(\| u_{k+1} - u_k \|^{n_t+1}\right) \tag{B.3}$$

where n_t is the highest order of the terms that the Taylor series expansion is going to consider [Chapra, 2008]. Note also that the approximation of $g(u_{k+1})$ is more accurate when u_{k+1} is closer to u_k. This is true because in Equation B.3, as $(u_{k+1} - u_k)$ shrinks, so does the size of the remainder term $O\left(\| u_{k+1} - u_k \|^{n_t+1}\right)$. Conversely, if the two points are considerably far apart, the Taylor series approximation may be very inaccurate.

B.3 LICQ

A minimum of the plant ($\mathbf{u}^{p,*}$) is only required to satisfy the KKT conditions of the plant optimization problem (Problem B.2) if the linear independence constraint qualification (LICQ) is satisfied. The LICQ states that the gradients of the active constraints (denoted by $\mathcal{G}^{p,a}$) must be linearly independent at the plant optimum $\mathbf{u}^{p,*}$ [Rardin, 1998]. If this requirement is not satisfied, $\mathbf{u}^{p,*}$ may not satisfy the proper KKT conditions, while still being a minimum of the plant.

B.4 KKT Conditions

If the first order (necessary) KKT conditions are satisfied at a particular operating point, $\mathbf{u}^{p,*}$, it means that this point is a KKT point or stationary point of the plant. The first order (necessary) KKT conditions for the plant optimization problem (Problem B.2) are:

$$\begin{aligned} \mathcal{G}^p(\mathbf{u}) &\leq \mathbf{0} \\ \gamma^T \mathcal{G}^p &= \mathbf{0} \\ \gamma &\geq \mathbf{0} \\ \frac{\partial \mathcal{L}^p}{\partial \mathbf{u}} = \frac{d\Phi^p}{d\mathbf{u}} + \gamma^T \frac{d\mathcal{G}^p}{d\mathbf{u}} &= \mathbf{0} \end{aligned} \tag{B.4}$$

where the Lagrangian is defined as: $\mathcal{L}^p = \Phi^p + \gamma^T \mathcal{G}^p$ and γ are the Lagrange multipliers [Rardin, 1998]. In (B.4), the first expression is called the primal feasibility condition, the second is referred to as the complementary slackness condition and the third and fourth expressions are together called the dual feasibility conditions [Marchetti, 2009]. Note that satisfaction of the LICQ and first order KKT conditions does not guarantee that $\mathbf{u}^{p,*}$ is a minimum of the plant. In addition, satisfaction of second order conditions is required.

The second order (sufficient) KKT conditions are used to determine what type of stationary point a particular operating point is. Note that in order to correctly apply the second order conditions, the particular operating point must satisfy the first-order KKT conditions (B.4).

To apply the second order conditions, information about the curvature of Φ^p at the stationary point, $\mathbf{u}^{p,*}$, must be known. If one or more constraints are active at $\mathbf{u}^{p,*}$, the reduced Hessian, $\nabla^2_{\mathbf{u},r}\Phi^p$, must be obtained. Note that the following derivation for the calculation of the reduced Hessian is taken from Forbes and Marlin [1996].

In order to compute the reduced Hessian, the active set ($\mathcal{G}^{p,a}$) at the stationary point the must be known. Based on this, the gradients of the active constraints with respect to the inputs at the stationary point, $\frac{d\mathcal{G}^{p,a}}{d\mathbf{u}}\Big|_{\mathbf{u}^{p,*}}$ can be calculated. A singular value decomposition is then carried out as follows:

$$\frac{d\mathcal{G}^{p,a}}{d\mathbf{u}}\bigg|_{\mathbf{u}^{p,*}} = \mathcal{U}\Sigma\mathcal{V} \tag{B.5}$$

A set of basis vectors for the null space of the gradients of the active constraints is given by the columns of $\mathcal{V}$ which correspond to the zero singular values of $\mathbf{\Sigma}$. Calling this set of basis vectors $\mathbf{Z}$, the reduced Hessian at the plant stationary point can be computed:

$$\nabla^2_{\mathbf{u},r}\Phi^p|_{\mathbf{u}^{p,*}} = \mathbf{Z}^T\nabla^2_{\mathbf{u}}\mathcal{L}^p|_{\mathbf{u}^{p,*}}\mathbf{Z} \tag{B.6}$$

If the reduced Hessian is strictly positive definite at the stationary point ($\nabla^2_{\mathbf{u},r}\Phi^p|_{\mathbf{u}^{p,*}} \succ \mathbf{0}$) then that point is a minimum of the plant optimization problem. Conversely, if the reduced Hessian at that point is strictly negative definite ($\nabla^2_{\mathbf{u},r}\Phi^p|_{\mathbf{u}^{p,*}} \prec \mathbf{0}$), then the point ($\mathbf{u}^{p,*}$) is a maximum. If it is positive semi-definite then it can be a minimum or a saddle point, and similarly, if it is negative semi-definite, it can be a maximum or a saddle point. If the reduced Hessian is indefinite, then $\mathbf{u}^{p,*}$ is a saddle point [Rardin, 1998].

B.5 Sensitivity Analysis Theory

The purpose of conducting a sensitivity analysis is to find out how the solution to an optimization problem would change if certain parameters in the optimization problem itself were to change. This is especially important in modifier adaptation (and RTO in general) because the system tends to be subject to measurement noise which can easily cause the modifiers, which are essentially parameters in the model-based optimization problem, to be incorrectly estimated. Non-linear sensitivity analysis theory is explained in detail in Fiacco [1983].

The following model-based optimization problem is considered for the explanation of non-linear sensitivity analysis:

$$\begin{aligned} \mathbf{u}^* \in \arg\min_{\mathbf{u}} \quad & \Phi^m\left(\mathbf{u}, \overline{\mathbf{\Lambda}}\right) \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{u}, \overline{\mathbf{\Lambda}}) \leq \mathbf{0} \end{aligned} \tag{B.7}$$

where the cost function and constraints of the process model are redefined in terms of the inputs and the modifiers ($\Phi^m(\mathbf{u}, \overline{\mathbf{\Lambda}}) := \phi(\mathbf{u}, \hat{\mathbf{y}}^m(\mathbf{u}, \overline{\mathbf{\Lambda}}))$ and $\mathbf{G}(\mathbf{u}, \overline{\mathbf{\Lambda}}) := \mathbf{g}(\mathbf{u}, \hat{\mathbf{y}}^m(\mathbf{u}, \overline{\mathbf{\Lambda}}))$). Note that this is just a reformulation of Problem 3.6. Also, both Φ^m and $\mathbf{G}$ must be twice

continuously differentiable with respect to the inputs and once continuously differentiable with respect to the modifiers. In addition, the following non-linear map is also defined: $\boldsymbol{\gamma}^* = \boldsymbol{\Gamma}^* \left(\overline{\boldsymbol{\Lambda}}\right)$, which relates the modifiers to the optimal values of the Lagrange multipliers.

To carry out the post-optimal sensitivity analysis, both $\mathbf{U}^*$ and $\boldsymbol{\Gamma}^*$ must be differentiable at $\overline{\boldsymbol{\Lambda}}$. Note that the non-linear map $\mathbf{U}^*$ (defined in Equation 3.10) can still be used here because the nature of the model-based optimization here is still the same as in Problem 3.6 (it has only been reformulated). The differentiability of $\mathbf{U}^*$ and $\boldsymbol{\Gamma}^*$ at $\overline{\boldsymbol{\Lambda}}$ is dependent on $\mathbf{u}^*$ being a unique optimizer of $\overline{\boldsymbol{\Lambda}}$. This requires that the LICQ as well as the first and second order KKT conditions for a local minimum are satisfied. In addition, the Lagrange multipliers corresponding to every active constraint must be greater than zero (strict complementary slackness) [Marchetti, 2009]. For reference, the first order KKT conditions for the reformulated model-based optimization problem (B.7) are:

$$\begin{aligned} \mathbf{G}(\mathbf{u}, \overline{\boldsymbol{\Lambda}}) &\leq \mathbf{0} \\ \boldsymbol{\gamma}^T \mathbf{G} &= \mathbf{0} \\ \boldsymbol{\gamma} &\geq \mathbf{0} \\ \frac{\partial \mathcal{L}^m}{\partial \mathbf{u}} = \frac{d\Phi^m}{d\mathbf{u}} + \boldsymbol{\gamma}^T \frac{d\mathbf{G}}{d\mathbf{u}} &= \mathbf{0} \end{aligned} \tag{B.8}$$

where the Lagrangian is defined as: $\mathcal{L}^m = \Phi^m + \boldsymbol{\gamma}^T \mathbf{G}$ and $\boldsymbol{\gamma}$ are the Lagrange multipliers [Rardin, 1998].

Note that since the analysis involves a set of assumptions and computations at a particular operating point ($\mathbf{u}^*$), it is only locally valid. Therefore, it should not be used to make assumptions about a potential change in the solution for a large parameter change. Furthermore, to carry out the analysis, an active set is assumed for the problem. Therefore, if a given parameter variation causes the active set of a problem to change in reality, the sensitivity analysis will be no longer be valid [Fiacco, 1983].

Two matrices, $\boldsymbol{\mathcal{X}}$ and $\boldsymbol{\mathcal{Y}}$ are required to complete the sensitivity analysis. They are defined

as follows [Fiacco, 1983]:

$$\mathcal{X} = \begin{bmatrix} \frac{d^2\mathcal{L}}{d\mathbf{u}^2} & -\frac{d\mathbf{g}}{d\mathbf{u}}^T \\ \gamma_1 \frac{dg_1}{d\mathbf{u}} & \\ \vdots & \mathbf{g} \\ \gamma_{n_g} \frac{dg_{n_g}}{d\mathbf{u}} & \end{bmatrix} \tag{B.9}$$

$$\mathcal{Y} = \begin{bmatrix} \frac{d^2\mathcal{L}}{d\mathbf{u}d\overline{\mathbf{\Lambda}}} \\ \gamma_1 \frac{dg_1}{d\overline{\mathbf{\Lambda}}} \\ \vdots \\ \gamma_{n_g} \frac{dg_{n_g}}{d\overline{\mathbf{\Lambda}}} \end{bmatrix} \tag{B.10}$$

Note that $\mathcal{X}$ represents the Jacobian of both the complementary slackness condition and the second dual feasibility KKT condition in (B.8) with respect to the inputs ($\mathbf{u}$) and the Lagrange multipliers (γ). Conversely, $\mathcal{Y}$ represents the Jacobian of the complementary slackness and second dual feasibility KKT conditions in (B.8) with respect to the modifiers, $\overline{\mathbf{\Lambda}}$ [Marchetti, 2009]. These matrices are then put together to obtain an estimate of how a change in the modifiers can effect the solution of the model-based optimization problem [Fiacco, 1983]:

$$\begin{bmatrix} \frac{d\mathbf{U}^*}{d\overline{\mathbf{\Lambda}}}\Big|_{\overline{\mathbf{\Lambda}}} \\ \frac{d\mathbf{\Gamma}^*}{d\overline{\mathbf{\Lambda}}}\Big|_{\overline{\mathbf{\Lambda}}} \end{bmatrix} = \mathcal{X}^{-1}\mathcal{Y} \tag{B.11}$$

Appendix C

Detailed Sensitivity Calculations

The purpose of this appendix is to detail the computation of the sensitivities used in the convergence analysis of single input problems (Section 3.2.2) and the variance cost approximation (Section 4.3.1). The sensitivities to be addressed are mainly those that are used to compute $\boldsymbol{\Upsilon}_\infty$ in both cases.

Specifically, the sensitivities making up Equations 3.18 and 3.19 are addressed. Note that although these equations are written for single input problems specifically, the derivations are written for multi-input problems in this appendix (unless otherwise stated), so they can also apply to the variance cost approximation for multiple input problems (Section 4.3.1). Before any specific derivations are made, note that the following assumption for the form of the derivatives:

$$\left[\frac{\partial \mathbf{v}}{\partial \mathbf{x}}\right]_{(i,j)} = \frac{\partial v_i}{\partial x_j} \tag{C.1}$$

where $\mathbf{v}$ and $\mathbf{x}$ are two random vectors [Petersen and Pedersen, 2008].

The first sensitivity from Equation 3.18, $\frac{\partial \boldsymbol{\mathcal{A}}}{\partial \overline{\boldsymbol{\Lambda}}_k}$, is now addressed. Examining Equation 3.8 (the full form of the update law $\boldsymbol{\mathcal{A}}$), $\overline{\boldsymbol{\Lambda}}_k$ appears linearly and therefore the following can be written:

$$\frac{\partial \boldsymbol{\mathcal{A}}}{\partial \overline{\boldsymbol{\Lambda}}_k} = \left(2\mathbf{I} - \mathbf{T}(\mathbf{u}_{k+1})\right)\left(\mathbf{I} - \mathbf{K}_\Lambda\right)\mathbf{T}(\mathbf{u}_k) \tag{C.2}$$

The sensitivity of the optimal inputs to the modifiers, $\frac{d\mathbf{U}^*}{d\overline{\boldsymbol{\Lambda}}}\Big|_{\overline{\boldsymbol{\Lambda}}_k}$, is computed using a non-linear post-optimal sensitivity analysis, as described in Fiacco [1983] and Appendix B of this thesis.

$\frac{\partial \boldsymbol{\mathcal{A}}}{\partial \mathbf{u}_{k+1}}$ is now considered. To begin, the following vector is defined for the purpose of this derivation:

$$\mathbf{w} = \left[\left(\mathbf{I} - \mathbf{K}_\Lambda\right)\mathbf{T}(\mathbf{u}_k)\overline{\boldsymbol{\Lambda}}_k + \mathbf{K}_\Lambda \left[\begin{array}{c} \mathbf{y}^p\left(\mathbf{u}_{k+1}\right) - \mathbf{f}\left(\mathbf{u}_{k+1}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right) \\ \left[\mathbf{B}^1_{k+1} - \frac{\partial f^1}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right)\right]^T \\ \vdots \\ \left[\mathbf{B}^{n_y}_{k+1} - \frac{\partial f^{n_y}}{\partial \mathbf{u}}\left(\mathbf{u}_{k+1}, \boldsymbol{\alpha}, \boldsymbol{\beta}\right)\right]^T \end{array}\right]\right] \tag{C.3}$$

The elements of $\mathbf{T}(\mathbf{u}_{k+1})$ are placed into a vector, $\mathbf{t}$, which can be defined as follows:

$$\mathbf{t} = \left[\mathbf{T}(\mathbf{u}_{k+1})_{(1,:)} \quad \mathbf{T}(\mathbf{u}_{k+1})_{(2,:)} \quad \dots \quad \mathbf{T}(\mathbf{u}_{k+1})_{(n_\Lambda,:)}\right] \tag{C.4}$$

The purpose of this change is that the derivative of the algorithm ($\mathcal{A}$) with respect to a matrix $\mathbf{T}(\mathbf{u}_{k+1})$ needs to be considered here. Since for both representation and implementation purposes it is advantageous if the derivative is two-dimensional, $\mathbf{T}(\mathbf{u}_{k+1})$ is stretched into the vector $\mathbf{t}$. The derivative, $\frac{\partial \mathcal{A}}{\partial \mathbf{t}}$ can be stated as follows:

$$\frac{\partial \mathcal{A}}{\partial \mathbf{t}} = \begin{bmatrix} -\mathbf{w} & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & -\mathbf{w} \end{bmatrix} \tag{C.5}$$

where n_Λ is the number of modifiers.

Now, the change in $\mathbf{T}(\mathbf{u}_{k+1})$ with respect to the operating point $\mathbf{u}_{k+1}$ ($\frac{\partial \mathbf{T}}{\partial \mathbf{u}}\big|_{\mathbf{u}_{k+1}}$), is required. Using the vector $\mathbf{t}$ the following non-linear map is written: $\mathbf{t} = \mathbf{\Theta}^t(\mathbf{u}_{k+1})$. Now, the corresponding derivative can be written:

$$\frac{\partial \mathbf{\Theta}^t}{\partial u_i} = \begin{bmatrix} \left.\begin{matrix} \mathbf{0}_{(n_y,1)} \\ \mathbf{0}_{(n_u(j-1),1)} \\ \mathbf{0}_{(i-1,1)} \\ 1 \\ \mathbf{0}_{(n_u-i,1)} \\ \mathbf{0}_{(n_u(n_y-j),1)} \end{matrix}\right\} j = 1...n_y \\ \mathbf{0}_{(n_y n_u n_\Lambda,1)} \end{bmatrix}, \forall\ i = 1...n_u \tag{C.6}$$

Now, using Equations C.5 and C.6, the term $\frac{\partial \mathcal{A}}{\partial \mathbf{u}_{k+1}}$ can be calculated as $\frac{\partial \mathcal{A}}{\partial \mathbf{t}} \frac{\partial \Theta^t}{\partial \mathbf{u}_{k+1}}$.

Four more sensitivities with respect to the non-linear map $\boldsymbol{\mathcal{A}}$ are computed as shown below:

$$\begin{aligned}
\frac{\partial \boldsymbol{\mathcal{A}}}{\partial \mathbf{y}^p_{k+1}} &= [2\mathbf{I} - \mathbf{T}(\mathbf{u}_{k+1})]\,\mathbf{K}_\Lambda \begin{bmatrix} \mathbf{I}_{n_y} \\ \mathbf{0}_{(n_y n_u, n_y)} \end{bmatrix} \\
\frac{\partial \boldsymbol{\mathcal{A}}}{\partial \mathbf{y}^m_{k+1}} &= [2\mathbf{I} - \mathbf{T}(\mathbf{u}_{k+1})]\,\mathbf{K}_\Lambda \begin{bmatrix} -\mathbf{I}_{n_y} \\ \mathbf{0}_{(n_y n_u, n_y)} \end{bmatrix} \\
\frac{\partial \boldsymbol{\mathcal{A}}}{\partial \mathbf{B}_{k+1}} &= [2\mathbf{I} - \mathbf{T}(\mathbf{u}_{k+1})]\,\mathbf{K}_\Lambda \begin{bmatrix} \mathbf{0}_{(n_y, n_y n_u)} \\ \mathbf{I}_{n_y n_u} \end{bmatrix} \\
\frac{\partial \boldsymbol{\mathcal{A}}}{\partial \nabla \mathbf{y}^m_{k+1}} &= [2\mathbf{I} - \mathbf{T}(\mathbf{u}_{k+1})]\,\mathbf{K}_\Lambda \begin{bmatrix} \mathbf{0}_{(n_y, n_y n_u)} \\ -\mathbf{I}_{n_y n_u} \end{bmatrix}
\end{aligned} \tag{C.7}$$

The four sensitivities in (C.7) were obtained by directly considering the definition of $\boldsymbol{\mathcal{A}}$ (Equations 3.8 and 3.9).

$\frac{\partial \boldsymbol{\mathcal{M}}}{\partial \mathbf{u}}$ is simply $\frac{\partial^2 \mathbf{f}}{\partial u^2}$ for the single input case. The same idea is applied for multi-input systems, however, the final matrix must be arranged to fit with the 4th expression of Equation C.7:

$$\frac{\partial \boldsymbol{\mathcal{M}}}{\partial \mathbf{u}} = \begin{bmatrix} \frac{\partial^2 f^1}{\partial \mathbf{u}^2} \\ \frac{\partial^2 f^2}{\partial \mathbf{u}^2} \\ \vdots \\ \frac{\partial^2 f^{ny}}{\partial \mathbf{u}^2} \end{bmatrix} \tag{C.8}$$

The last expression that must be developed in Equation 3.18 is $\left(\frac{\partial \mathcal{R}}{\partial u_{k+1}} + \frac{\partial \mathcal{R}}{\partial y^p_{k+1}} \cdot \frac{d\mathbf{F}}{du}\Big|_{u_{k+1}}\right)$. Its derivation for single input problems is considered first. Computing the individual derivative terms, $\frac{\partial \mathcal{R}}{\partial u_{k+1}} = -\frac{\mathbf{y}^p_{k+1} - \mathbf{y}^p_k}{(u_{k+1}-u_k)^2}$ and $\frac{\partial \mathcal{R}}{\partial y^p_{k+1}} = \frac{1}{u_{k+1}-u_k}$, the following can be written:

$$\begin{aligned}
\left(\frac{\partial \mathcal{R}}{\partial u_{k+1}} + \frac{\partial \mathcal{R}}{\partial y^p_{k+1}} \cdot \frac{d\mathbf{F}}{du}\Big|_{u_{k+1}}\right) &= -\frac{\mathbf{y}^p_{k+1} - \mathbf{y}^p_k}{(u_{k+1}-u_k)^2} + \frac{1}{u_{k+1}-u_k} \cdot \frac{d\mathbf{F}}{du}\Big|_{u_{k+1}} \\
&= \frac{1}{u_{k+1}-u_k}\left[\frac{d\mathbf{F}}{du}\Big|_{u_{k+1}} - \frac{\mathbf{y}^p_{k+1} - \mathbf{y}^p_k}{u_{k+1}-u_k}\right]
\end{aligned} \tag{C.9}$$

Now, a Taylor series expansion is written for $\mathbf{y}^p_k$ around the point $\mathbf{y}^p_{k+1}$:

$$\mathbf{y}^p_k \approx \mathbf{y}^p_{k+1} + \frac{d\mathbf{F}}{du}\Big|_{u_{k+1}} (u_k - u_{k+1}) + \frac{1}{2}\frac{d^2\mathbf{F}}{du^2}\Big|_{u_{k+1}} (u_k - u_{k+1})^2 \tag{C.10}$$

where terms greater than second order are ignored. This expansion is then rearranged to yield the following:

$$\frac{\mathbf{y}_{k+1}^p - \mathbf{y}_k^p}{u_{k+1} - u_k} \approx \left.\frac{d\mathbf{F}}{du}\right|_{u_{k+1}} - \frac{1}{2}\left.\frac{d^2\mathbf{F}}{du^2}\right|_{u_{k+1}} (u_{k+1} - u_k) \tag{C.11}$$

Equation C.11 is then substituted into the second expression of (C.9) to yield:

$$\left(\frac{\partial \mathcal{R}}{\partial u_{k+1}} + \frac{\partial \mathcal{R}}{\partial y_{k+1}^p} \cdot \left.\frac{d\mathbf{F}}{du}\right|_{u_{k+1}}\right) \approx \frac{1}{2}\left.\frac{d^2\mathbf{F}}{du^2}\right|_{u_{k+1}} \tag{C.12}$$

which completes the derivation. Note that $\left.\frac{d^2\mathbf{F}}{du^2}\right|_{u_{k+1}}$ can be estimated in a number of ways. First, if a benchmark model for the plant is available (a model that is too complicated for online implementation), it could be used here to approximate $\mathbf{F}(\mathbf{u})$. Conversely, it could also be estimated through plant experimentation or inherent process knowledge.

For multi-input problems, the development is much easier, since Υ_∞ is being developed for only the variance cost approximation. In the variance cost approximation (Section 4.3.1) the assumption is made that the ideal modifier adaptation algorithm approximates the behaviour of the dual modifier adaptation algorithm. It is assumed in ideal modifier adaptation, that an exact representation of the plant, $\mathbf{F}(\mathbf{u})$, is available. Note that in practice, the benchmark plant model is typically used in the ideal modifier adaptation algorithm. Alternatively, $\mathbf{F}(\mathbf{u})$ can also be approximated by conducting plant experiments or through inherent process knowledge. $\mathbf{F}(\mathbf{u})$ is then differentiated exactly as the process model was earlier (C.8):

$$\frac{\partial^2 \mathbf{F}}{\partial \mathbf{u}^2} = \begin{bmatrix} \frac{\partial^2 F^1}{\partial \mathbf{u}^2} \\ \frac{\partial^2 F^2}{\partial \mathbf{u}^2} \\ \vdots \\ \frac{\partial^2 F^{ny}}{\partial \mathbf{u}^2} \end{bmatrix} \tag{C.13}$$

Now, $\frac{\partial^2 \mathbf{F}}{\partial \mathbf{u}^2}$ replaces $\left(\frac{\partial \mathcal{R}}{\partial u_{k+1}} + \frac{\partial \mathcal{R}}{\partial y_{k+1}^p} \cdot \left.\frac{d\mathbf{F}}{du}\right|_{u_{k+1}}\right)$ in Equation 3.18 for the design cost approximation of multiple input problems.

Considering 3.19, $\frac{\partial \mathcal{A}}{\partial \mathbf{u}_k}$ can be addressed in much the same way that $\frac{\partial \mathcal{A}}{\partial \mathbf{u}_{k+1}}$ was earlier. To simplify the computation, the following matrix is defined: $\mathcal{H} = [2\mathbf{I} - \mathbf{T}(\mathbf{u}_{k+1})]\,(\mathbf{I} - \mathbf{K}_\Lambda)$. A

vector representation for $\mathbf{T}(\mathbf{u}_k)$ is defined as follows:

$$\tau = \begin{bmatrix} \mathbf{T}(\mathbf{u}_k)_{(1,:)} & \mathbf{T}(\mathbf{u}_k)_{(2,:)} & \dots & \mathbf{T}(\mathbf{u}_k)_{(n_\Lambda,:)} \end{bmatrix} \tag{C.14}$$

$\frac{\partial \mathcal{A}}{\partial \tau}$ can then be written as follows:

$$\frac{\partial \mathcal{A}}{\partial \tau} = \begin{bmatrix} \mathcal{H}_{(:,1)}\overline{\Lambda}_k^T & \mathcal{H}_{(:,2)}\overline{\Lambda}_k^T & \dots & \mathcal{H}_{(:,n_y(n_u+1))}\overline{\Lambda}_k^T \end{bmatrix} \tag{C.15}$$

This was written by repeatedly applying the following for a random vector $\mathbf{x}$ and a random matrix $\mathbf{A}$: $\frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{A}} = \mathbf{x}\mathbf{x}^T$ [Petersen and Pedersen, 2008]. Now, writing $\tau = \Theta^\tau(\mathbf{u}_k)$ and realizing that $\frac{\partial \Theta^\tau}{\partial \mathbf{u}_k} = \frac{\partial \Theta^t}{\partial \mathbf{u}_{k+1}}$ it is evident that $\frac{\partial \mathcal{A}}{\partial \mathbf{u}_k}$ in Equation 3.19 can be replaced by $\frac{\partial \mathcal{A}}{\partial \tau}\frac{\partial \Theta^\tau}{\partial \mathbf{u}_k}$.

The computation of $\left.\frac{d\mathbf{U}^*}{d\overline{\Lambda}}\right|_{\overline{\Lambda}_{k-1}}$ just consists of carrying out the non-linear sensitivity analysis described previously at a new set of modifiers, $\overline{\Lambda}_{k-1}$. Now, all the sensitivities required to write Υ_∞ for both the single input problem convergence analysis of Section 3.2.2 and the variance cost approximation of Section 4.3.1 have been developed.

There is one other pair of sensitivities used in the variance cost approximation that need to be estimated. These are $\frac{\partial \overline{\Lambda}_{k+1}}{\partial \mathbf{y}_{k+1}^p}$ and $\frac{\partial \overline{\Lambda}_{k+1}}{\partial \mathbf{y}_k^p}$, which are required for the approximation of the covariance matrix of the modifiers due to measurement noise (Equation 4.22). $\frac{\partial \overline{\Lambda}_{k+1}}{\partial \mathbf{y}_{k+1}^p}$ and $\frac{\partial \overline{\Lambda}_{k+1}}{\partial \mathbf{y}_k^p}$ are expressed as follows:

$$\begin{aligned} \frac{\partial \overline{\Lambda}_{k+1}}{\partial \mathbf{y}_{k+1}^p} &= \frac{\partial \mathcal{A}}{\partial \mathbf{y}_{k+1}^p} + \frac{\partial \mathcal{A}}{\partial \mathbf{B}_{k+1}} \cdot \frac{\partial \mathbf{B}_{k+1}}{\partial \mathbf{y}_{k+1}^p} \\ \frac{\partial \overline{\Lambda}_{k+1}}{\partial \mathbf{y}_k^p} &= \frac{\partial \mathcal{A}}{\partial \mathbf{B}_{k+1}} \cdot \frac{\partial \mathbf{B}_{k+1}}{\partial \mathbf{y}_k^p} \end{aligned} \tag{C.16}$$

where $\frac{\partial \mathcal{A}}{\partial \mathbf{y}_{k+1}^p}$ and $\frac{\partial \mathcal{A}}{\partial \mathbf{B}_{k+1}}$ were computed in (C.7). Note that the expression for $\frac{\partial \overline{\Lambda}_{k+1}}{\partial \mathbf{y}_k^p}$ is slightly different because there is no direct dependence of the non-linear map ($\mathcal{A}$) on $\mathbf{y}_k^p$.

An explanation is now included for the approximation given in Section 4.3.1 for $\frac{\partial \mathbf{B}_{k+1}}{\partial \mathbf{y}_{k+1}^p}$ and $\frac{\partial \mathbf{B}_{k+1}}{\partial \mathbf{y}_k^p}$ (Equation 4.23) for the special case where $\mathcal{B}$ is diagonal. $\frac{\partial \mathbf{B}_{k+1}}{\partial \mathbf{y}_{k+1}^p}$ can be expressed for

the n^{th} measurement and the i^{th} output (when $i = n$) as:

$$\frac{\partial \mathbf{B}_{k+1}^{i}}{\partial \mathbf{y}_{k+1}^{p,n}} \approx \begin{bmatrix} \frac{\Delta u_1}{\sum_{j=1}^{nu} \Delta u_j^2} \\ \frac{\Delta u_2}{\sum_{j=1}^{nu} \Delta u_j^2} \\ \vdots \\ \frac{\Delta u_{n_u}}{\sum_{j=1}^{nu} \Delta u_j^2} \end{bmatrix} \tag{C.17}$$

where $\Delta \mathbf{u}_{k+1} = \mathbf{u}_{k+1} - \mathbf{u}_k$. Note that if $i \neq n$ then there is no relationship between the n^{th} measurement and $\mathbf{B}_{k+1}^{i}$.

It is clear that as the algorithm converges, each term of Equation C.17 goes to infinity. In practice though, the dual constraints prevent this singularity from ever happening. By closely examining the shape of the ellipsoid itself, it is possible to estimate the values of the terms in Equation C.17. The maximum absolute values of each of the individual terms in Equation C.17 are inferred by considering the case, for term i, when $\Delta u_j = 0 \; \forall \; j \neq i$. This assumption is applied to Equation C.17 to obtain the following:

$$\frac{\partial \mathbf{B}_{k+1}^{i}}{\partial \mathbf{y}_{k+1}^{p,n}} \approx \begin{bmatrix} \frac{1}{\Delta u_1} \\ \frac{1}{\Delta u_2} \\ \vdots \\ \frac{1}{\Delta u_{n_u}} \end{bmatrix} \tag{C.18}$$

Hence, Equation C.18 represents an upper bound on the magnitude of the effect that the measurement noise can have on the Broyden update.

The dual constraint itself is difficult to generalize from an online point of view, due to varying previous movement directions causing different restricted areas. Therefore, only the ellipsoid portion of this constraint is considered in this offline approximation. Now, in the separate cases for each of the n_u terms in Equation C.18, the minimum distance that the ellipsoid itself will allow in any input direction, u_i, is $1/\sqrt{b_i}$. This is accurate in each separate case since the assumption $\Delta u_j = 0 \; \forall \; j \neq i$ reduces the equation of the ellipsoid to $b_i \left(\Delta u_i^2\right) = 1$. Therefore, the maximum absolute value of each of the terms in the sensitivity

in Equation C.18 can be estimated:

$$\frac{\partial \mathbf{B}_{k+1}^{i}}{\partial \mathbf{y}_{k+1}^{p,n}} \approx \begin{bmatrix} \sqrt{b_1} \\ \sqrt{b_2} \\ \vdots \\ \sqrt{b_{n_u}} \end{bmatrix} \tag{C.19}$$

Since this derivation was predicated on the assumption that each of the individual terms of Equation C.17 takes on its maximum value, this will lead to an over-estimation of the effect of measurement noise on the Broyden estimates. Note that this development also applies to $\frac{\partial \mathbf{B}_{k+1}}{\partial \mathbf{y}_k^p}$ which appears as a part of the $\frac{\partial \mathbf{\Lambda}_{k+1}}{\partial \mathbf{y}_k^p}$ sensitivity. The sign of the result is flipped due to the difference between $\mathbf{y}_k^p$ and $\mathbf{y}_{k+1}^p$ in Equation 2.27.

To demonstrate this approximation in practice, a small example is now given. A two input system is assumed, with an ellipsoid defined by $\mathbf{b} = [4, 4]$. The algorithm has just moved to an operating point 0.4 units in the u_1 direction and 0.3 units in the u_2 direction from the previous operating point. The new point, previous point, and ellipse are shown as parts of Figure C.1.

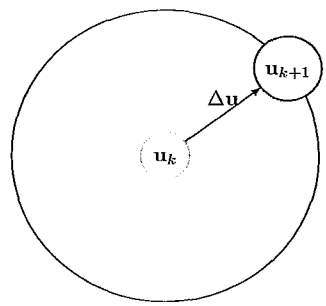

Figure C.1: Diagram for Broyden sensitivity calculation example

Equation C.19 gives the approximation of the sensitivity of the Broyden update for output i to the measurement n:

$$\frac{\partial \mathbf{B}_{k+1}^{i}}{\partial \mathbf{y}_{k+1}^{p,n}} = \begin{bmatrix} \sqrt{\mathbf{b}_1} \\ \sqrt{\mathbf{b}_2} \end{bmatrix} = \begin{bmatrix} \sqrt{4} \\ \sqrt{4} \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \tag{C.20}$$

The value of the sensitivity in the case of the move specified for this particular example can be calculated using Equation C.17:

$$\frac{\partial \mathbf{B}^i_{k+1}}{\partial \mathbf{y}^{p,n}_{k+1}} \approx \begin{bmatrix} \frac{\Delta u_1}{\sum_{j=1}^{n_u} \Delta u_j^2} \\ \frac{\Delta u_2}{\sum_{j=1}^{n_u} \Delta u_j^2} \end{bmatrix} = \begin{bmatrix} \frac{0.4}{0.4^2+0.3^2} \\ \frac{0.3}{0.4^2+0.3^2} \end{bmatrix} = \begin{bmatrix} 1.6 \\ 1.2 \end{bmatrix} \tag{C.21}$$

Note that the approximation does indeed provide an upper bound on the magnitude of the true value of the sensitivity in question.

Finally, a short explanation of the assumption of Equation 4.25 is given for the special case in which $\mathcal{B}$ is diagonal. Recall that this assumption deals with approximating the effect of the dual constraints on the algorithm when one of them is always active. Consider the fictitious scenario situation in which the algorithm is only moving between three operating points, $\mathbf{u}^*_\infty$, $\mathbf{u}^{i-}$ and $\mathbf{u}^{i+}$. Both $\mathbf{u}^{i-}$ and $\mathbf{u}^{i+}$ are defined as the closest points that the ellipsoid constraint will allow to $\mathbf{u}^*_\infty$ if movement is only allowed in the input direction i. These points are shown in Figure C.2 for the u_1 input direction.

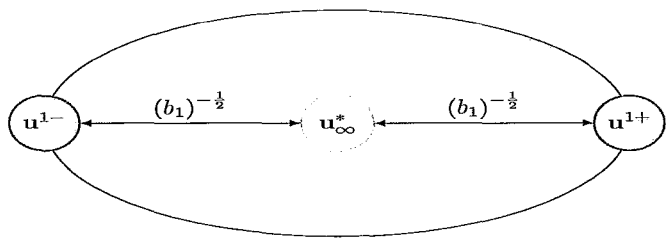

Figure C.2: Assumed algorithm movement for input direction 1 in scenario 2

The distance between $\mathbf{u}^*_\infty$ and both $\mathbf{u}^{i-}$ and $\mathbf{u}^{i+}$ is easily computed as $\frac{1}{\sqrt{b_i}}$, because the assumption of movement only in the input direction i reduces the equation of the ellipsoid to $b_i\left(\Delta u_i^2\right) = 1$. The following 4-point pattern is now assumed from experience for this scenario: starting at $\mathbf{u}^{i-}$, moving to $\mathbf{u}^*_\infty$, then on to $\mathbf{u}^{i+}$ and then back to $\mathbf{u}^*_\infty$. Assuming that this pattern is constantly repeated, the variance of input i is easily computed: $\mathbf{V}(u_{\infty,i}) = \frac{1}{2b_i}$. Now applying this scenario in all of the other input directions, the following is obtained:

$$\mathbf{V}_2\left(\mathbf{u}_\infty\right) = \begin{bmatrix} \frac{1}{2b_1} & & \\ & \ddots & \\ & & \frac{1}{2b_{n_u}} \end{bmatrix} \tag{C.22}$$

which is equivalent to (4.25) if $\mathcal{B}$ is diagonal. Since in reality the constraint can only force the computed operating point to be pushed the maximum distance ($\frac{1}{\sqrt{b_i}}$) in one direction at a time, this represents an over-approximation of the effect of the dual constraints on the iterates.

Appendix D

Listing of the Source Code

In the following appendix, a selection of the source code for the work presented in this thesis is shown. It is written in MATLAB and the version shown here corresponds to the propane furnace problem. The basic dual modifier adaptation code is shown first, followed by the high-level functions involved in the design of dual modifier adaptation systems.

D.1 Dual Modifier Adaptation Algorithm

```
function MainModAdapt()

%this function is the main function that can be used to test a particular
%dual modifier adaptation design. It produces a graph of the movement of
%the system as well as writes out some performance statistics to a file

%reads in modifiers to be updated
methodology_details;

%creates all scripts containing system properties (e.g. models, objective
%function) as well as corresponding derivative info
[nu, nyp, nym, nxp, nxm, ng] = model_gen_sim(param_var_num, yparam_var_num, mods_var_num);

%reads in problem parameters
problem_parameters_real;

%computing noise level (standard deviation) for each output
rhov = noise_sd*opt_noise_values;

%setting initial values
u_tan_pos = zeros(nu+nym,1);
u_tan_neg = zeros(nu+nym,1);

%performing a plant experiment at the starting point
[zold, zn, c] = plant_measurements(u_matrix(:,1), pp, rhov, nyp, nu, nxp);

%computing model outputs at starting point
w = ones(nxm,1);
for i = 1:(length(u_matrix)-nu)
    w(i) = u_matrix(nu+i);
end
xum = fsolve(@(w)model_eqns(w, u_matrix(:,1), p), w);
model_states;
output_model_equations;

%computing model derivative estimate at starting point
dydu = model_calcs(u_matrix(:,1), lambdacost(:,1), xum, yum, p);

%computing first Broyden derivative estimate
BR = dydu;

%solving first optimization problem (no dual constraints)
u = optimizer3(u_matrix(:,1), p, lambdacost(:,1), ng, nym, ustart, uL, uU, max_it);

```

```
%loop to move through RTO iterations
for j=1:num_it

    %updating and tracking values through RTO iterations
    zold = zn;
    cost_tracker(j) = c;
    u_matrix(:,j+1) = u;

    %taking measurements from the fictitious plant
    [z, zn, c] = plant_measurements(u_matrix(:,j+1), pp, rhov, nyp, nu, nxp);

    %calculating the new dual constraint implementation parameters
    if j >= (nu-1)
        [p, u_tan_pos, u_tan_neg] = const_parameter_calc4(u_matrix, p, nu, j, u, g_vector, nxm);
    end

    %updating the Broyden derivative estimates
    BRold = BR;
    BR = broydon(u_matrix(1:nu,j+1), u_matrix(1:nu,j), zn, zold, BRold);

    %computing the model derivatives
    opt_states;
    output_model_equations;
    dydu = model_calcs(u_matrix(:,j+1), lambdacost(:,j),  u_matrix((1+nu):(nu+nxm),j+1), yum, p);

    %comparing plant and model information
    C1 = zn - yum';
    C2 = [];
    for i = 1:nym
        C2 = [C2 ; (BR(i,:)' - dydu(i,:)')];
    end

    %computing new modifier values
    lambdacost(:,j+1) = modifier_calc_bar(C1, C2, nym, nu, lambdacost(:,j), u_matrix(1:nu,j+1),...
        u_matrix(1:nu,j), mods_var_num, b, q);

    %solving disjunctive optimization problem
    best = inf;
    [uset2, fopt, exitflag] = optimizer2(u_matrix(:,j+1), p, lambdacost(:,j+1), ng, nym,...
        u_tan_pos, uL, uU, max_it);
    if fopt < best && exitflag > 0;
        u = uset2;
        best = fopt;
    end
    [uset3, fopt, exitflag] = optimizer3(u_matrix(:,j+1), p, lambdacost(:,j+1), ng, nym,...
        u_tan_neg, uL, uU, max_it);
    if fopt < best && exitflag > 0;
        u = uset3;
    end

end

%creating statistics vectors
lambdacost_stat = lambdacost(:,stat_it:end);
u_matrix_stat = u_matrix(:,stat_it:end);
for i = 1:length(u_matrix_stat)
    u_matrix_stat2(1,i) = (u_matrix_stat(1,i)+18.1575)*20;
    u_matrix_stat2(2,i) = u_matrix_stat(2,i)+5;
```

```
end
cost_stat = cost_tracker(stat_it:end);

%plotting results
contour_plot_sim(pp, nxp, ng, zeros(nym*(nu+1),1), u, u_matrix_stat, g_backoff)
hold on
plot(u_matrix_stat2(1,:),u_matrix_stat2(2,:),'-x')

%calculating and printing statistics to a file
u_averages(1,:) = mean(u_matrix_stat(1:nu,:)');
u_variances(1,:) = var(u_matrix_stat(1:nu,:)');
cost_average(1) = mean(cost_stat);
cost_variance(1) = var(cost_stat);

data1 = [];
for i = 1:nu
    data1(:,i) = (g_vector(i));
end
data1(:,nu+1) = noise_sd;

for i = 1:nu
    data1(:,nu+1+1+2*(i-1)) = u_averages(:,i);
    data1(:,nu+1+2+2*(i-1)) = u_variances(:,i);
end
data1(:,3*nu+2) = (cost_average);
data1(:,3*nu+3) = (cost_variance);

varnames1 = [];
for i = 1:(3*nu+3)
    varnames1 = [varnames1 ; 'Data____'];
end
casenames1 =[];
for l = 1:1
    casenames1 = [casenames1 ;'_____'];
end

tblwrite(data1,varnames1,casenames1,'case_t1.dat')

end
```

```
%contains information on which modifiers should be updated (other two
%variables are for a later extension to hybrid methods)

param_var_num = [];
yparam_var_num = [];
mods_var_num = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23...
        24 25 26 27 28 29 30 31 32 33];
```

```
function [nu, nyp, nym, nxp, nxm, ng] = model_gen_sim()

%this function generates a set of files (or scripts) which contain
%information about the system (objective function, benchmark and normal
%process models, constraints) and theri respective derivatives

%defining symbolic variables
syms U1 U2 real
```

```
syms U01 U02 real
syms X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12 real
syms Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9 Y10 Y11 real
syms TH1 TH2 TH3 TH4 TH5 TH6 TH7 TH8 TH9 TH10 TH11 TH12 TH13 TH14 TH15 TH16 TH17 TH18 TH19...
 TH20 TH21 TH22 TH23 TH24 TH25 TH26 TH27 TH28 TH29 TH30 TH31 TH32 TH33 TH34 TH35 TH36 TH37...
 TH38 TH39 TH40 TH41 TH42 TH43 TH44 TH45 TH46 TH47 TH48 TH49 TH50 TH51 TH52 TH53 TH54 TH55...
 TH56 TH57 TH58 TH59 TH60 TH61 TH62 TH63 TH64 TH65 TH66 TH67 TH68 TH69 TH70 TH71 TH72 TH73...
 TH74 TH75 TH76 TH77 TH78 TH79 TH80 TH81 TH82 TH83 TH84 TH85 TH86 TH87 TH88 TH89 real
syms L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14 L15 L16 L17 L18 L19 L20 L21 L22 L23 L24...
 L25 L26 L27 L28 L29 L30 L31 L32 L33 real

%defining vectors of various quantities
U = [U1 U2];
U0 = [U01 U02];
XM = [X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12];
XP = [X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12];
YM = [Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9 Y10 Y11];
YP = [Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9 Y10 Y11];
param_set = [TH1 TH2 TH3 TH4 TH5 TH6 TH7 TH8 TH9 TH10 TH11 TH12 TH13 TH14 TH15 TH16 TH17...
        TH18 TH19 TH20 TH21 TH22 TH23 TH24 TH25 TH26 TH27 TH28 TH29 TH30 TH31 TH32 TH33 TH34...
        TH35 TH36 TH37 TH38 TH39 TH40 TH41 TH42 TH43 TH44 TH45 TH46 TH47 TH48 TH49 TH50 TH51...
        TH52 TH53 TH54 TH55 TH56 TH57 TH58 TH59 TH60 TH61 TH62 TH63 TH64 TH65 TH66 TH67 TH68...
        TH69 TH70 TH71 TH72 TH73 TH74 TH75 TH76 TH77 TH78 TH79 TH80 TH81 TH82 TH83 TH84 TH85...
        TH86 TH87 TH88 TH89];
W = [U XM];
P = [param_set XM];
L = [L1 L2 L3 L4 L5 L6 L7 L8 L9 L10 L11 L12 L13 L14 L15 L16 L17 L18 L19 L20 L21 L22 L23 L24...
        L25 L26 L27 L28 L29 L30 L31 L32 L33];
TH = [TH1 TH2 TH3 TH4 TH5 TH6 TH7 TH8 TH9 TH10 TH11 TH12 TH13 TH14 TH15 TH16 TH17 TH18...
        TH19 TH20 TH21 TH22 TH23 TH24 TH25 TH26 TH27 TH28 TH29 TH30 TH31 TH32 TH33 TH34 TH35...
        TH36 TH37 TH38 TH39 TH40 TH41 TH42 TH43 TH44 TH45 TH46 TH47 TH48 TH49 TH50 TH51 TH52...
        TH53 TH54 TH55 TH56 TH57 TH58 TH59 TH60 TH61 TH62 TH63 TH64 TH65 TH66 TH67 TH68 TH69...
        TH70 TH71 TH72 TH73 TH74 TH75 TH76 TH77 TH78 TH79 TH80 TH81 TH82 TH83 TH84 TH85 TH86...
        TH87 TH88 TH89];

%setting counters for numbers of inputs, states and outputs
nu = length(U);
nxp = length(XP);
nxm = length(XM);
nyp = length(YP);
nym = length(YM);

%writing various scripts
write_inputs(U,'U','inputs.m');
write_inputs_old(U0,'U0','inputs_old.m');
write_states_plant(XP,'X','plant_states.m');
write_states_model(XM,'X','model_states.m');
write_outputs_plant(YP,'Y','plant_outputs.m');
write_outputs_model(YM,'Y','model_outputs.m');
write_parameters(TH,'TH','parameters.m');
write_modifiers(L,'L','modifiers.m');
write_states_opt(XM,'X','opt_states.m',length(U));

%*****************************************************************

%cost function
f = -1*((U1*100)*TH61*(Y1 + L1 + L12*U1 + L13*U2) + (U1*100)*TH62*(Y2 + L2 + L14*U1 + L15*U2)...
 + (U1*100)*TH63*(Y3 + L3 + L16*U1 + L17*U2) + (U1*100)*TH64*(Y4 + L4 + L18*U1 + L19*U2)...
```

```
 + (U1*100)*TH65*(Y5 + L5 + L20*U1 + L21*U2) + (U1*100)*TH66*(Y6 + L6 + L22*U1 + L23*U2)...
 + (U1*100)*TH67*(Y7 + L7 + L24*U1 + L25*U2) + (U1*100)*TH68*(Y8 + L8 + L26*U1 + L27*U2)...
 + (U1*100)*TH69*(Y9 + L9 + L28*U1 + L29*U2) + (U1*100)*TH70*(Y10 + L10 + L30*U1 + L31*U2)...
 - TH71*TH85*(U1*100) - TH72*(Y11 + L11 + L32*U1 + L33*U2) - (U1*100)*TH73);
write1D(f,'f','cost.m');

%gradient of cost function wrt the inputs
dfdu = jacobian(f,U);
write2D(dfdu,'dfdu','cost_input_derivs.m');

%gradient of cost function wrt the plant outputs
dfdyp = jacobian(f,YP);
write2D(dfdyp,'dfdyp','cost_output_plant_derivs.m');

%gradient of cost function wrt the model outputs
dfdym = jacobian(f,YM);
write2D(dfdym,'dfdym','cost_output_derivs.m');

%gradients of cost function wrt optimization variables
dfdw = jacobian(f, W);
write2D(dfdw,'dfdw','cost_opt_derivs.m');

%*************************************************************

%plant model
xup(1) = TH1 + TH2*TH85 + TH3*TH85^2 + TH4*U2 + TH5*U2^2 + TH6*TH85*U2 - X1;
xup(2) = TH7 + TH8*TH85 + TH9*TH85^2 + TH10*U2 + TH11*U2^2 + TH12*TH85*U2 - X2;
xup(3) = TH13 + TH14*TH85 + TH15*TH85^2 + TH16*U2 + TH17*U2^2 + TH18*TH85*U2 - X3;
xup(4) = TH19 + TH20*TH85 + TH21*TH85^2 + TH22*U2 + TH23*U2^2 + TH24*TH85*U2 - X4;
xup(5) = TH25 + TH26*TH85 + TH27*TH85^2 + TH28*U2 + TH29*U2^2 + TH30*TH85*U2 - X5;
xup(6) = TH31 + TH32*TH85 + TH33*TH85^2 + TH34*U2 + TH35*U2^2 + TH36*TH85*U2 - X6;
xup(7) = TH37 + TH38*TH85 + TH39*TH85^2 + TH40*U2 + TH41*U2^2 + TH42*TH85*U2 - X7;
xup(8) = TH43 + TH44*TH85 + TH45*TH85^2 + TH46*U2 + TH47*U2^2 + TH48*TH85*U2 - X8;
xup(9) = TH49 + TH50*TH85 + TH51*TH85^2 + TH52*U2 + TH53*U2^2 + TH54*TH85*U2 - X9;
xup(10) = TH55 + TH56*TH85 + TH57*TH85^2 + TH58*U2 + TH59*U2^2 + TH60*TH85*U2 - X10;
xup(11) = TH74*((U1*100)/(X12*100)) - X11;
xup(12) = TH75*X1 + TH76*X2 + TH77*X3 + TH78*X4 + TH79*X5 + TH80*X6 + TH81*X7 + TH82*X8...
        + TH83*X9 + TH84*X10 - (X12*100);
write1D(xup,'xup','plant_model.m');

%gradient of functions wrt the inputs for the plant
dhdup = jacobian(xup,U);
write2D(dhdup,'dhdup','in_func_plant_derivs.m');

%gradient of functions wrt the states for the plant
dhdxp = jacobian(xup,XP);
write2D(dhdxp,'dhdxp','st_func_plant_derivs.m');

%*************************************************************

%plant outputs
yup(1) = X1;
yup(2) = X2;
yup(3) = X3;
yup(4) = X4;
yup(5) = X5;
yup(6) = X6;
yup(7) = X7;
```

```
125 yup(8) = X8;
126 yup(9) = X9;
127 yup(10) = X10;
128 yup(11) = X11;
129 write1D(yup,'yup','output_plant_equations.m');
130
131 %gradient of plant outputs wrt the inputs
132 dydup = jacobian(yup,U);
133 write2D(dydup,'dydup','output_input_plant_derivs.m');
134
135 %gradient of plant outputs wrt the plant states
136 dydxp = jacobian(yup,XP);
137 write2D(dydxp,'dydxp','output_state_plant_derivs.m');
138
139 %*****************************************************************
140
141 %approximate model
142 xum(1) = TH1 + TH2*TH85 + TH3*TH85^2 + TH4*U2 + TH5*U2^2 + TH6*TH85*U2 - X1;
143 xum(2) = TH7 + TH8*TH85 + TH9*TH85^2 + TH10*U2 + TH11*U2^2 + TH12*TH85*U2 - X2;
144 xum(3) = TH13 + TH14*TH85 + TH15*TH85^2 + TH16*U2 + TH17*U2^2 + TH18*TH85*U2 - X3;
145 xum(4) = TH19 + TH20*TH85 + TH21*TH85^2 + TH22*U2 + TH23*U2^2 + TH24*TH85*U2 - X4;
146 xum(5) = TH25 + TH26*TH85 + TH27*TH85^2 + TH28*U2 + TH29*U2^2 + TH30*TH85*U2 - X5;
147 xum(6) = TH31 + TH32*TH85 + TH33*TH85^2 + TH34*U2 + TH35*U2^2 + TH36*TH85*U2 - X6;
148 xum(7) = TH37 + TH38*TH85 + TH39*TH85^2 + TH40*U2 + TH41*U2^2 + TH42*TH85*U2 - X7;
149 xum(8) = TH43 + TH44*TH85 + TH45*TH85^2 + TH46*U2 + TH47*U2^2 + TH48*TH85*U2 - X8;
150 xum(9) = TH49 + TH50*TH85 + TH51*TH85^2 + TH52*U2 + TH53*U2^2 + TH54*TH85*U2 - X9;
151 xum(10) = TH55 + TH56*TH85 + TH57*TH85^2 + TH58*U2 + TH59*U2^2 + TH60*TH85*U2 - X10;
152 xum(11) = TH74*((U1*100)/(X12*100)) - X11;
153 xum(12) = TH75*X1 + TH76*X2 + TH77*X3 + TH78*X4 + TH79*X5 + TH80*X6 + TH81*X7 + TH82*X8...
154         + TH83*X9 + TH84*X10 - (X12*100);
155 write1D(xum,'xum','model.m');
156
157 %gradient of functions wrt the inputs
158 dhdum = jacobian(xum,U);
159 write2D(dhdum,'dhdum','in_func_model_derivs.m');
160
161 %gradient of functions wrt the model states
162 dhdxm = jacobian(xum,XM);
163 write2D(dhdxm,'dhdxm','st_func_model_derivs.m');
164
165 %*****************************************************************
166
167 %approximate model outputs
168 yum(1) = X1;
169 yum(2) = X2;
170 yum(3) = X3;
171 yum(4) = X4;
172 yum(5) = X5;
173 yum(6) = X6;
174 yum(7) = X7;
175 yum(8) = X8;
176 yum(9) = X9;
177 yum(10) = X10;
178 yum(11) = X11;
179 write1D(yum,'yum','output_model_equations.m');
180
181 %gradient of model outputs wrt the inputs
182 dydum = jacobian(yum,U);
```

```
183 write2D(dydum,'dydum','output_input_model_derivs.m');
184
185 %gradient of model outputs wrt the model states
186 dydxm = jacobian(yum,XM);
187 write2D(dydxm,'dydxm','output_state_model_derivs.m');
188
189 %*****************************************************************
190 %section has not been updated to cover recent changes
191
192 %first set of constraints and their derivatives
193 g2(1) = Y11 + L11 + L32*U1 + L33*U2 - 0.0147;
194 g2(2) = (U1*100) - 15;
195 g2(3) = 5 - (U1*100);
196 g2(4) = U2 - 0.93;
197 g2(5) = 0.70 - U2;
198 g2(6) = ((Y3 + L3 + L16*U1 + L17*U2)*(U1*100) - 3) + TH86;
199 g2(7) = TH89 - (TH87*(U1-U01)+TH88*(U2-U02));
200 write1D(g2,'g','constraints2.m');
201 dg2 = jacobian(g2, W);
202 write2D(dg2,'dgdw','ineq_const_derivs2.m');
203 dg2dym = jacobian(g2, YM);
204 write2D(dg2dym,'dgdym','const_output_derivs2.m');
205 dg2dl = jacobian(g2, L);
206 write2D(dg2dl,'dgdl','const_lam_derivs2.m');
207 dg2du = jacobian(g2, U);
208 write2D(dg2du,'dgdu','const_input_derivs2.m');
209
210 %second set of constraints and their derivatives
211 g3(1) = Y11 + L11 + L32*U1 + L33*U2 - 0.0147;
212 g3(2) = (U1*100) - 15;
213 g3(3) = 5 - (U1*100);
214 g3(4) = U2 - 0.93;
215 g3(5) = 0.70 - U2;
216 g3(6) = ((Y3 + L3 + L16*U1 + L17*U2)*(U1*100) - 3) + TH86;
217 g3(7) = TH89 + (TH87*(U1-U01)+TH88*(U2-U02));
218 write1D(g3,'g','constraints3.m');
219 dg3 = jacobian(g3, W);
220 write2D(dg3,'dgdw','ineq_const_derivs3.m');
221 dg3dym = jacobian(g3, YM);
222 write2D(dg3dym,'dgdym','const_output_derivs3.m');
223 dg3dl = jacobian(g3, L);
224 write2D(dg3dl,'dgdl','const_lam_derivs3.m');
225 dg3du = jacobian(g3, U);
226 write2D(dg3du,'dgdu','const_input_derivs3.m');
227
228 %setting counter for number of constraints
229 ng = length(g2);
230
231 %*****************************************************************
232
233 %model constraints written for the optimizer
234 h(1) = TH1 + TH2*TH85 + TH3*TH85^2 + TH4*U2 + TH5*U2^2 + TH6*TH85*U2 - X1;
235 h(2) = TH7 + TH8*TH85 + TH9*TH85^2 + TH10*U2 + TH11*U2^2 + TH12*TH85*U2 - X2;
236 h(3) = TH13 + TH14*TH85 + TH15*TH85^2 + TH16*U2 + TH17*U2^2 + TH18*TH85*U2 - X3;
237 h(4) = TH19 + TH20*TH85 + TH21*TH85^2 + TH22*U2 + TH23*U2^2 + TH24*TH85*U2 - X4;
238 h(5) = TH25 + TH26*TH85 + TH27*TH85^2 + TH28*U2 + TH29*U2^2 + TH30*TH85*U2 - X5;
239 h(6) = TH31 + TH32*TH85 + TH33*TH85^2 + TH34*U2 + TH35*U2^2 + TH36*TH85*U2 - X6;
240 h(7) = TH37 + TH38*TH85 + TH39*TH85^2 + TH40*U2 + TH41*U2^2 + TH42*TH85*U2 - X7;
```

```
h(8) = TH43 + TH44*TH85 + TH45*TH85^2 + TH46*U2 + TH47*U2^2 + TH48*TH85*U2 - X8;
h(9) = TH49 + TH50*TH85 + TH51*TH85^2 + TH52*U2 + TH53*U2^2 + TH54*TH85*U2 - X9;
h(10) = TH55 + TH56*TH85 + TH57*TH85^2 + TH58*U2 + TH59*U2^2 + TH60*TH85*U2 - X10;
h(11) = TH74*((U1*100)/(X12*100)) - X11;
h(12) = TH75*X1 + TH76*X2 + TH77*X3 + TH78*X4 + TH79*X5 + TH80*X6 + TH81*X7 + TH82*X8...
        + TH83*X9 + TH84*X10 - (X12*100);
write1D(h,'h','eq_model.m');

%gradients of process model wrt optimization variables
dh = jacobian(h, W);
write2D(dh,'dh','eq_const_derivs.m');

end
```

Note that this is intended as an example of these *write_...* functions. These were adapted from similar functions provided by Dr. Chachuat.

```
function write_inputs(Deps_variables,Var_name,File_name)

fid = fopen(File_name,'wt');
for i=1:length(Deps_variables)
    Var_name_i = strcat(Var_name,num2str(i));
    Equation_i = strcat(Var_name_i,'_=_u(',num2str(i),');');
    fprintf(fid,'%s\n',Equation_i);
end
fclose(fid);

end
```

```
%This script contains a bunch of system specific settings.  It is read
%directly into the main dual modifier adaptation function.  Its main
%purpose is to group together all the problem specific settings in one
%place.

%setting backoff and dual constraint paramters
g_backoff = 0.136;
g_vector = [25000;15000];

%setting plant and model parameters
p = [0.00494724; -0.03664829; 0.04069737; 0.019859935; -0.000969887; -0.002677888;...
        0.10800515; -0.052572383; 0.078125; -0.164172769; 0.377328861; -0.046034175;...
        0.492783395; 1.669458174; -1.875; -1.466407509; 1.076528013; 0.060010201;...
        0.007192032; -0.29770971; 0.318080357; 0.341778885; -0.253919555; 0.001326192;...
        -0.233759303; -4.104523391; 4.553571429; 2.831994285; -1.728076697; 0.011782708;...
        0.478230614; 2.608846932; -2.898718813; -1.0; -1.28684*10^(-11); -7.42328*10^(-13);...
        0.043096185; 0.520875137; -0.602678571; -0.427566842; 0.316262353; 0.03427697;...
        -0.026571494; -0.313838283; 0.3515625; 0.268080883; -0.174337183; -0.001683244;...
        -0.000404145; 0.002154272; -0.003395081; -1.58084*10^(-5); 0.000258392; 0.001683244;...
        0.053389638; 0.369410985; -0.368303571; -0.40355106; 0.386925703; -0.058684009;...
        0.24; 0.06; 0.25; 0.08; 0.20; 0.08; 0.20; 0.14; 0.08; 0.14; 0.03; 3; 0.08; 0.036;...
        2; 16; 28; 30; 42; 44; 54; 56; 58; 62; 0.3; g_backoff ; zeros(nu+1,1)];
pp = [0.00494724; -0.03664829; 0.04069737; 0.019859935; -0.000969887; -0.002677888;...
        0.10800515; -0.052572383; 0.078125; -0.114172769; 0.327328861; -0.046034175;...
        0.492783395; 1.669458174; -1.875; -1.966407509; 1.576528013; 0.060010201;...
```

```
        0.007192032; -0.29770971; 0.318080357; 0.291778885; -0.203919555; 0.001326192;...
        -0.233759303; -4.104523391; 4.553571429; 3.331994285; -2.228076697; 0.011782708;...
        0.478230614; 2.608846932; -2.898718813; -1.0; -1.28684*10^(-11); -7.42328*10^(-13);...
        0.043096185; 0.520875137; -0.602678571; -0.427566842; 0.316262353; 0.03427697;...
        -0.026571494; -0.313838283; 0.3515625; 0.268080883; -0.174337183; -0.001683244;...
        -0.000404145; 0.002154272; -0.003395081; -1.58084*10^(-5); 0.000258392; 0.001683244;...
        0.053389638; 0.369410985; -0.368303571; -0.40355106; 0.386925703; -0.058684009;...
        0.24; 0.06; 0.25; 0.08; 0.20; 0.08; 0.20; 0.14; 0.08; 0.14; 0.03; 3; 0.08; 0.036;...
        2; 16; 28; 30; 42; 44; 54; 56; 58; 62; 0.3; g_backoff ; zeros(nu+1,1)];

%initializing noise level and typical output values
noise_sd = 0.01;
opt_noise_values = [0.011029851237480 0.180665660061833 0.243067841956929 0.051120128815393...
 0.192389524065852 0.268872263828000 0.009087264974922 0.013358263483604 0.000432344129713...
 0.029976857627441 0.012959475225053];

%setting system start point and modifiers as well as the point at which the
%optimizer will be started in its first run
u_matrix(:,1) = [0.07 0.85 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.01 20/100];
lambdacost = zeros(nym*(nu+1),1);
for i = 1:length(u_matrix(:,1))
    ustart(i,1) = 0.97*u_matrix(i,1) + 0.06*u_matrix(i,1)*rand();
end

%optimizer settings
max_it = 50;
for i = 1:nu
    uL(i) = -inf;
    uU(i) = inf;
end
for i = 1:nym
    uL(i+nu) = -inf;
    uU(i+nu) = inf;
end

%filter parameters
b = 0.4;
q = 0.4;

%information about how many RTO iterations to run and at which iteration
%statistics should start being tabulated
num_it = 50;
stat_it = 1;
```

```
function [z, zn, f] = plant_measurements(u, p, rhov, nyp, nu, nxp)

%the purpose of this function is twofold: to generate plant measurements
%from the simulated plant model (adding in measurement noise) and
%computing the true plant cost function value

%solving simulated plant equations
w = ones(nxp,1);
for i = 1:(length(u)-nu)
    w(i) = u(nu+i);
end
options = optimset('TolX', 1e-10, 'TolFun', 1e-10);
xup = fsolve(@(w)plant_model_eqns(w, u, p), w, options);
```

```
14
15 %computing the plant cost function value (noiseless)
16 inputs;
17 parameters;
18 plant_states;
19 output_plant_equations;
20 plant_outputs;
21 lambdacost = zeros(nyp*(nu+1),1);
22 uold = zeros(nu,1);
23 inputs_old;
24 modifiers;
25
26 cost;
27
28 %adding measurement noise to the simluated plant outputs
29 z = yup';
30
31 for i = 1:nyp
32     inc = normrnd(0,rhov(i));
33     zn(i) = z(i) + inc;
34 end
35 zn = zn';
36
37 end
```

```
1  function xup = plant_model_eqns(xup, u, p)
2
3      %solves the implicit plant model equations
4
5      inputs;
6      parameters;
7      plant_states;
8
9      plant_model;
10 end
```

```
1  function xum = model_eqns(xum, u, p)
2
3      %solves the implicit process model equations
4
5      inputs;
6      parameters;
7      model_states;
8
9      model;
10 end
```

```
1  function dydum_tot = model_calcs(u, lambdacost, xum, yum, p)
2
3  %computes the derivatives of the model outputs wrt the inputs
4
5  modifiers;
6  parameters;
7  inputs;
8  model_outputs;
```

```
model_states;

in_func_model_derivs;
st_func_model_derivs;
output_input_model_derivs;
output_state_model_derivs;

dydum_tot = dydum + dydxm*(-1*inv(dhdxm)*dhdum);

end
```

```
function [xopt, fopt, exitflag, output, lambda] = optimizer2(u0, p, lambdacost, ng, ustart,...
        uL, uU, max_it)

%this function solves one of the model-based optimization problems

%providing settings to the optimizer
options = optimset('Display', 'iter', 'GradObj', 'on', 'GradConstr', 'on', ...
                   'LargeScale', 'off', 'HessUpdate', 'bfgs', 'Diagnostics', 'on', ...
                   'TolX', 1e-7, 'TolFun', 1e-7, 'TolCon', 1e-7, 'MaxFunEval', 3*max_it, ...
                   'MaxIter', max_it);

%carrying out the optimization
[xopt, fopt, exitflag, output, lambda] = fmincon(@(u)cost_fun(u, u0, p, lambdacost), ustart,...
        [], [], [], [], uL, uU, @(u)nonlin_const2(u, u0, p, lambdacost, ng), options);

end
```

```
function [xopt, fopt, exitflag, output, lambda] = optimizer3(u0, p, lambdacost, ng, ustart,...
        uL, uU, max_it)

%this function solves one of the model-based optimization problems

%providing settings to the optimizer
options = optimset('Display', 'iter', 'GradObj', 'on', 'GradConstr', 'on', ...
                   'LargeScale', 'off', 'HessUpdate', 'bfgs', 'Diagnostics', 'on', ...
                   'TolX', 1e-7, 'TolFun', 1e-7, 'TolCon', 1e-7, 'MaxFunEval', 3*max_it, ...
                   'MaxIter', max_it);

%carrying out the optimization
[xopt, fopt, exitflag, output, lambda] = fmincon(@(u)cost_fun(u, u0, p, lambdacost), ustart,...
        [], [], [], [], uL, uU, @(u)nonlin_const3(u, u0, p, lambdacost, ng), options);

end
```

```
function [f, df] = cost_fun(u, uold, p, lambdacost)

%this function calculates the value of the cost function and its
%derivatives for the optimizer

    parameters;
    modifiers;
    inputs;
    inputs_old;
    opt_states;
```

```
    output_model_equations;
    model_outputs;

    cost;

    %only run if analytical derivatives are asked for
    if nargout > 1
        cost_opt_derivs;
        cost_output_derivs;
        output_input_model_derivs;
        output_state_model_derivs;

        df = dfdw + dfdym*[dydum dydxm];
    end

end
```

```
function [g, h, dg, dh] = nonlin_const2(u, uold, p, lambdacost, ng)

    parameters;
    modifiers;
    inputs;
    inputs_old;
    opt_states;
    output_model_equations;
    model_outputs;

    g = [];

    %evaluating inequality constraints
    if ng > 0
        constraints2;
    end

    %evaluating equality constraints
    eq_model;

    %computing constraint derivatives if required
    if nargout > 2
        dg = [];
        if ng > 0
            ineq_const_derivs2;
            output_state_model_derivs;
            output_input_model_derivs;
            const_output_derivs2;
            dg = dgdw + dgdym*[dydum dydxm];
        end
        dg = dg';
        eq_const_derivs;
        dh = dh';
    end

end
```

```
function [g, h, dg, dh] = nonlin_const3(u, uold, p, lambdacost, ng)

```

```
    parameters;
    modifiers;
    inputs;
    inputs_old;
    opt_states;
    output_model_equations;
    model_outputs;

    g = [];

    %evaluating inequality constraints
    if ng > 0
        constraints3;
    end

    %evaluating equality constraints
    eq_model;

    %computing constraint derivatives if required
    if nargout > 2
        dg = [];
        if ng > 0
            ineq_const_derivs3;
            output_state_model_derivs;
            output_input_model_derivs;
            const_output_derivs3;
            dg = dgdw + dgdym*[dydum dydxm];
        end
        dg = dg';
        eq_const_derivs;
        dh = dh';
    end

end
```

```
function [p, u_tan_pos, u_tan_neg] = const_parameter_calc4(u_matrix, p, nu, j, u, g_vector, nxm)

%this function computes the vector orthogonal to the hyperplane of previous
%movement directions as well as the rhs of the disjunctive constraints

%calculating the orthogonal vector using the adjugate of the matrix of
%previous movement directions
vect_matrix = zeros(nu,nu);
for i = 1:(nu-1)
    vect_matrix(1:nu,i+1) = u_matrix(1:nu,j+2-i)-u_matrix(1:nu,j+1-i);
end
for i = 1:nu
    vect_matrix(i,1) = u_matrix(i,j+1)*rand();
end
determ = det(vect_matrix);
adj_vect = determ*inv(vect_matrix);
for i = 1:nu
    p(end-nu-1+i) = adj_vect(1,i);
end

%calculating the rhs of the disjunctive constraint
B = zeros(nu,nu);
```

```
for i = 1:nu
     B(i,i) = g_vector(i);
end
p(end) = sqrt(adj_vect(1,:)*inv(B)*adj_vect(1,:)');

%computing the tangent points
u_tan_pos = zeros(nu+nxm,1);
u_tan_neg = zeros(nu+nxm,1);
u_tan_pos(1:nu) = u(1:nu) + inv(B)*adj_vect(1,:)'/sqrt(adj_vect(1,:)*inv(B)*adj_vect(1,:)');
u_tan_neg(1:nu) = u(1:nu) - inv(B)*adj_vect(1,:)'/sqrt(adj_vect(1,:)*inv(B)*adj_vect(1,:)');
u_tan_pos(nu+1:end) = u(nu+1:end);
u_tan_neg(nu+1:end) = u(nu+1:end);

end
```

```
function BRnew = broydon(u, uold, y, yold, BRold)

%computes new Broyden derivative estimates

BRnew = BRold + (((y - yold)-BRold*(u - uold))*(u - uold)')/((u - uold)'*(u - uold));

end
```

```
function [lambdacost, C, K] = modifier_calc_bar(C1, C2, nym, nu, lambdacostold, u0, u02,...
 mods_var_num, b, q)

%this function computes the new values of the modifiers

%computes matrices so alternate modifiers can be used
Tuk1 = eye(nym*(nu+1),nym*(nu+1));
Tuk = eye(nym*(nu+1),nym*(nu+1));
for i = 1:nym
    for j = 1:nu
        Tuk1(i,nym+j+nu*(i-1))=u0(j);
        Tuk(i,nym+j+nu*(i-1))=u02(j);
    end
end

%putting together filter parameter matrix as well as old modifier vector
K = zeros(nym*(nu+1),nym*(nu+1));
for i = 1:nym*(nu+1)
    if i <= nym
        C(i) = C1(i);
        K(i,i) = b;
        modold(i)=lambdacostold(i);
    else
        C(i)=C2(i-nym);
        K(i,i) = q;
        modold(i)=lambdacostold(i);
    end
end

%defining identity matrix
I = eye(length(C));

%calculating new modifier values
```

```
lambdacost = (2*I-Tuk1)*((I-K)*Tuk*modold' + K*C');

%erasing update from any modifiers that arent being considered
count = 1;
for i = 1:nym*(nu+1)
   if i == mods_var_num(count)
       count = count + 1;
       if count > length(mods_var_num)
           count = 1;
       end
   else
       lambdacost(i) = 0;
   end
end

end
```

```
function contour_plot_sim(p, nxp, ng, lambdacost, uold, u_matrix, g_backoff, zn)

%defines range to create the contour plot (in scaled values)
range = [0.04 0.16 0.65 0.95];

%spreads input grid evenly across range
u(1) = range(1) - (range(2)-range(1))/10;
u(2) = range(3) - (range(4)-range(3))/10;

%computes plant cost function value for each input grid point (as well as
%corresponding constraint values)
for i=1:11
    u(2) = u(2) + (range(4)-range(3))/10;

    for j=1:11
        u(1) = u(1) + (range(2)-range(1))/10;

        x = 0.5*ones(nxp,1);
        x(1:length(zn),1) = zn;
        inputs;
        inputs_old;
        parameters;
        modifiers;
        [xup, fval, exitflag] = fsolve(@(x)plant_model_eqns(x, u, p), x);
        plant_states;
        output_plant_equations;
        plant_outputs;

        %collecting constraint information
        constraints2;
        gans1(i,j) = g(1);
        gans2(i,j) = g(2);
        gans3(i,j) = g(3);
        gans4(i,j) = g(4);
        gans5(i,j) = g(5);
        gans6(i,j) = g(6);

        %tracking plant cost function
        cost;
        fans(i,j)=-1*f;
```

```
41
42          %tracking inputs
43          u1(i,j) = (u(1)*100);
44          u2(i,j) = (u(2));
45
46
47      end
48
49      u(1) = u(1) - 11*(range(2)-range(1))/10;
50
51  end
52
53  %creating contour plots for cost function and constraints
54  [C, f] = contour(u1,u2,fans);
55  set(f,'LevelStep',0.025)
56  set(f,'ShowText','on','TextStep',get(f,'LevelStep')*2)
57  colormap cool
58  hold on
59
60  [C, f] = contour(u1,u2,gans1);
61  set(f,'LevelStep',500000)
62  set(f,'ShowText','on','TextStep',get(f,'LevelStep')*2)
63  colormap cool
64  hold on
65
66  [C, f] = contour(u1,u2,gans2);
67  set(f,'LevelStep',500000)
68  set(f,'ShowText','on','TextStep',get(f,'LevelStep')*2)
69  colormap cool
70  hold on
71
72  [C, f] = contour(u1,u2,gans3);
73  set(f,'LevelStep',500000)
74  set(f,'ShowText','on','TextStep',get(f,'LevelStep')*2)
75  colormap cool
76  hold on
77
78  [C, f] = contour(u1,u2,gans4);
79  set(f,'LevelStep',500000)
80  set(f,'ShowText','on','TextStep',get(f,'LevelStep')*2)
81  colormap cool
82  hold on
83
84  [C, f] = contour(u1,u2,gans5);
85  set(f,'LevelStep',500000)
86  set(f,'ShowText','on','TextStep',get(f,'LevelStep')*2)
87  colormap cool
88  hold on
89
90  [C, f] = contour(u1,u2,gans6);
91  set(f,'LevelStep',500000)
92  set(f,'ShowText','on','TextStep',get(f,'LevelStep')*2)
93  colormap cool
94  hold on
95
96  end
```

D.2 Offline Design of Dual Modifier Adaptation

Only a couple of functions are shown here for brevity. They are intended to demonstrate the basic framework for the calculation.

```
function combo_search()

%This function starts a dual modifier adaptation design run by designating
%which modifiers should be updated (note the other vectors are for the
%implementation of the two-step approach).

param_var_num = [];
yparam_var_num = [];
mods_var_num = [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27...
        28 29 30 31 32 33];
[best bbest, bias_best, var_best, final_backoff] = problem_coordinator...
        (param_var_num, yparam_var_num, mods_var_num);

disp(best)
disp(bbest)
disp(bias_best)
disp(var_best)
disp(final_backoff)

end
```

```
function [best_cost, bbest_real, best_bias, best_var, final_backoff] = problem_coordinator...
        (param_var_num, yparam_var_num, mods_var_num)

%This function coordinates the design procedure run. It starts with setting
%up the different uncertain parameter scenarios that will be used in the
%design cost optimization. Then it runs the model generation function
%which sets up all the cost and constraint functions and models and their
%derivatives. Next it computes the plant optimum (using the benchmark
%model) and then runs ideal modifier adaptation using the benchmark model.
%Finally, it carries out the design cost optimization.

%creating uncertain parameter scenarios (shown here for maximum 2 uncertain parameters)
count = 1;
params_to_vary = [7];
p1_vect = [0];
p2_vect = [0];
for i = 1:length(p1_vect)
    for j = 1:length(p2_vect)
        p_vect(count,:) = [p1_vect(i) ; p2_vect(j)];
        count = count + 1;
    end
end

%running model generation function
[nu, nyp, nym, nxp, nxm, ng] = model_gen_final(param_var_num, yparam_var_num, mods_var_num);

%running the benchmark plant model optimization and ideal modifier
%adaptation
```

```
[row,col] = size(p_vect);
for i = 1:row
    [u_plant_opt(:,i), lagrange_plant(:,i)] = MainModAdapt_new_plant(params_to_vary, p_vect(i,:));
    [u(:,i), lambdacost(:,i), zn(:,i), lagrange(:,i), BR(:,((1+nu*(i-1)):nu*i)), C(:,i), K, p,...
        pp(:,i), dydp(:,((1+nu*(i-1)):nu*i)), rhov, max_it, viol] = MainModAdapt_new_final...
        (param_var_num, yparam_var_num, mods_var_num, p_vect(i,:), params_to_vary, nu, nyp, nym,...
        nxp, nxm, ng);
end

%running design cost optimization to obtain estimates of optimal dual
%constraint parameters (dual constraint parameters are LN-scaled)
bstart = [log(80) log(30)];
[bbest, best] = optimizer_b(param_var_num, yparam_var_num, mods_var_num, nym, nu, u, K, rhov, p,...
        lambdacost, ng, nxm, bstart, BR, dydp, C, lagrange, u_plant_opt, pp, nyp, nxp, row, zn,...
        max_it, viol)

%running design cost calculation function to generate bias and variance
%cost and constraint backoff level at optimal dual constraint parameters
[best_cost, best_bias, best_var, final_backoff] = cost_fun_b(bbest, param_var_num,...
        yparam_var_num, mods_var_num, nym, nu, u, K, rhov, p, lambdacost, ng, nxm,...
        BR, dydp, C, lagrange, u_plant_opt, pp, nyp, nxp, row, zn, max_it, viol);

bbest_real = exp(bbest);

end
```

Appendix E

Data for the Propane Furnace Case Study

Table E.1: Sale values for products

Product	$ per lb
Hydrogen	0.24
Methane	0.06
Ethylene	0.25
Ethane	0.08
Propylene	0.20
Propane	0.08
Butadiene	0.20
Butylene	0.14
Butane	0.08
Gasoline	0.14

Table E.2: Process costs

Process Input	Cost ($)
Steam	0.03 per lb
Energy	3.00 per MBTU
Feed	0.08 per lb

Table E.3: Empirical model parameters (simulated and benchmark plant models)

Component	ϱ_0	ϱ_1	ϱ_2	ϱ_3	ϱ_4	ϱ_5
Hydrogen	0.00494724	-0.03664829	0.04069737	0.01985993	-0.00096988	-0.00267788
Methane	0.10800515	-0.05257238	0.078125	-0.11417276	0.32732886	-0.04603417
Ethylene	0.49278340	1.66945817	-1.875	-1.96640751	1.57652801	0.06001020
Ethane	0.007192032	-0.29770971	0.31808035	0.29177888	-0.20391955	0.00132619
Propylene	-0.23375930	-4.10452339	4.55357143	3.33199429	-2.22807667	0.01178271
Propane	0.478230614	2.60884693	-2.89871881	-1.0000	-1.2868E-11	-7.4232E-13
Butadiene	0.043096185	0.52087513	-0.60267857	-0.42756684	0.31626235	0.03427697
Butylene	-0.02657149	-0.31383828	0.3515625	0.26808088	-0.17433718	-0.00168324
Butane	-0.00040414	0.00215427	-0.00339508	-1.5808E-5	0.00025839	0.00168324
Gasoline	0.053389638	0.36941098	-0.36830357	-0.40355106	0.38692570	-0.05868400

Table E.4: Empirical model parameters (process model)

Component	ϱ_0	ϱ_1	ϱ_2	ϱ_3	ϱ_4	ϱ_5
Hydrogen	0.00494724	-0.03664829	0.04069737	0.01985993	-0.00096988	-0.00267788
Methane	0.10800515	-0.05257238	0.078125	**-0.16417276**	**0.37732886**	-0.04603417
Ethylene	0.49278340	1.66945817	-1.875	**-1.46640751**	**1.07652801**	0.06001020
Ethane	0.007192032	-0.29770971	0.31808035	**0.34177888**	**-0.25391955**	0.00132619
Propylene	-0.23375930	-4.10452339	4.55357143	**2.83199429**	**-1.72807667**	0.01178271
Propane	0.478230614	2.60884693	-2.89871881	-1.0000	-1.2868E-11	-7.4232E-13
Butadiene	0.043096185	0.52087513	-0.60267857	-0.42756684	0.31626235	0.03427697
Butylene	-0.02657149	-0.31383828	0.3515625	0.26808088	-0.17433718	-0.00168324
Butane	-0.00040414	0.00215427	-0.00339508	-1.5808E-5	0.00025839	0.00168324
Gasoline	0.053389638	0.36941098	-0.36830357	-0.40355106	0.38692570	-0.05868400

Note that the parameters that were altered between the simulated plant and process models are indicated in bold.