Real-Time Optimization via Directional Modifier Adaptation, with Application to Kite Control

THÈSE N^{O} 6571 (2015)

PRÉSENTÉE LE 20 MARS 2015

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR

LABORATOIRE D'AUTOMATIQUE

PROGRAMME DOCTORAL EN GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Sean COSTELLO

acceptée sur proposition du jury:

Prof. C. N. Jones, président du jury
Prof. D. Bonvin, Dr G. François, directeurs de thèse
Prof. C. de Prada Moraga, rapporteur
Prof. M.-O. Hongler, rapporteur
Prof. R. Smith, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2015

Essentially, all models are wrong, but some are useful.
— George E. P. Box

The bulk of mankind is as well equipped for flying as thinking.
— Jonathon Swift

To my parents.

Acknowledgements

This thesis developed hand in hand with myself over the past five years. Many people played a part, either by directly contributing to the thesis work, or by teaching and encouraging me.

Firstly, I thank Professor Dominique Bonvin for his unwavering conviction in me and a wonderfully run lab. Often, all I needed was encouragement, freedom, and a little money, which you always provided! Hopefully some of your professionalism rubbed off on me. I might very well have given the whole business up at some stage had it not been for the long conversations with my co-supervisor, Dr. Grégory François. It is a pleasure to work with such a friendly, tactful person. A special mention also goes to Gene Bunin, my moral compass in all matters concerning research, whose enthusiasm for RTO kept things interesting.

It is well known at EPFL that LA is a special place. I very much appreciated the support given by all the permanent staff members. It would be impossible to set up high-quality experiments without the support provided by Christophe and company, travel would become a nightmare without Françine, and without Ruth there simply would be no LA! Exercise sessions would fall apart without Sandra. Ali and Phillipe tirelessly explain and re-explain control theory, or any kind of theory in the case of Phillipe, to anyone who knocks on their door.

A special thanks goes to Professor Colin Jones for introducing me to the Swiss Kite Power team. This was a turning point in my thesis. Corey Houle, Rolf Luchsinger and the rest of the team inspired me with their passion. Thanks for letting me join in the fun. It was also great to collaborate with Ioannis Lymperopoulos, an apparently inexhaustible source of theoretical solutions to practical problems.

Many students worked with me on the kite project: Alex Mermoud, Enea Martinoli, Paul Bertusi, Damien Benoît, Raphaël Waldis, Yannick Poffet, José Afonso Aires Mesquita Jr. and Nikitas Rontsis. Thanks for your help, your extraordinary commitment to the project, your stamina and patience during the long outdoor tests, and your friendship. When everything went wrong during a testing session, as it often did, it was much easier to pick up the pieces and put it right as a team.

My father has been a collaborator in every engineering project I've undertaken, and

this thesis is no exception. From helping me build my own kite board 12 years ago, to helping me solder the kite prototype's heavy-duty ground screws together just six months ago, your practical know-how and encouragement is still as much in demand as ever!
Living abroad, friends become family. You know who you are! Who can tell where we'll all be in years to come, but I hope we'll still have time for a good singsong in the mountains.
My family is my rock. Thanks to my parents and siblings for your support, reminding me how to have a good laugh. Alice, you unwittingly arrived at just at the right moment to put things in perspective. Finally, a huge thanks to Giorgia for your infinite patience and encouragement.

Morges, 12 January 2015 Sean Costello

Abstract

The steady advance of computational methods makes model-based optimization an increasingly attractive method for process improvement. Unfortunately, the available models are often inaccurate. The traditional remedy is to update the model parameters, but this generally leads to a difficult parameter estimation problem that must be solved on-line, and the resulting model may still poorly predict the process optimum. An iterative real-time optimization method called Modifier Adaptation overcomes these obstacles by directly incorporating plant measurements into the optimization framework, in the form of constraint values and plant-gradient estimates.

Experimental gradient estimation is the main difficulty encountered when applying Modifier Adaptation. The experimental effort required to estimate plant gradients increases along with the number of plant inputs. This tends to make the method intractable for processes with many inputs. The main methodological contribution of this thesis is a new algorithm called 'Directional' Modifier Adaptation, which handles the gradient-estimation problem by estimating plant derivatives only in certain privileged directions. By spending less effort on gradient estimation, the algorithm can focus on optimizing the plant. A 'Dual' Directional Modifier Adaptation is proposed, which estimates these 'directional' derivatives using past operating points. This algorithm exhibits fast convergence to a neighborhood of the plant optimum, even for processes with many inputs.

Modifier Adaptation also makes use of an approximate process model. Another difficulty which may be encountered is that this model's inputs differ from those of the real process. The second methodological contribution is 'Generalized' Modifier Adaptation, a framework for dealing with the case where the model's inputs differ from those of the plant. This approach circumvents remodeling the system. For example, Generalized Modifier Adaptation allows an open-loop process model to be used to optimize a closed-loop plant, without having to model the controller.

The Dual Directional Modifier Adaptation method is applied to a purpose-built experimental kite system. Kites are currently being developed into a radical new renewable-energy technology. Large-scale applications include pulling ships and generating

electricity from wind at altitudes beyond the reach of conventional wind turbines. While kites were traditionally manually controlled, these new applications require autonomous operation. The first challenge is to design reliable control algorithms for kites, capable of dealing with noise, wind disturbances, and time delays. The control algorithm keeps the kite flying a periodic path, at very high speeds. The second challenge is to choose this path in order to maximize the energy extracted from the wind.
During this thesis, a small autonomous kite system was constructed. Thirty days of experimental testing were carried out, over the space of two years. A new modeling hypothesis was validated, linking steering deflections to a decrease in the kite's lift/drag ratio. A path-following controller was implemented, capable of achieving good, robust path-following performance, despite significant time delay. The only real-time measurement required by the control algorithm is the kite's position, which, in this work, was obtained simply by measuring the angle of the kite's tether.
A two-layer optimizing control scheme was implemented on the experimental kite system. Dual Directional Modifier Adaptation was used to periodically update the reference path tracked by the path-following controller, in order to maximize the kite's average tether tension. Despite extremely high noise levels, the algorithm was able to locate the optimal reference path in only 10 minutes, while ensuring that a minimum-altitude constraint was never violated. The resulting average tether tension is about 20% higher than that obtained following the optimal path computed using the model. An experimental study comparing the average tether tension obtained using different reference paths confirms the importance of path shape, and validates the optimal solution reached by the Dual Directional Modifier Adaptation algorithm.

Key words: Real-Time Optimization, Control, Airborne Wind Energy, Kite Power.

Résumé

Avec les progrès des méthodes computationnelles, l'optimisation numérique devient une méthode de plus en plus attractive pour l'amélioration des procédés. Malheureusement, les méthodes classiques d'optimisation nécessitent qu'un modèle du procédé soit disponible, lequel est souvent imprécis, notamment pour les procédés industriels. Ainsi, les variables optimales déterminées à l'aide du modèle ne sont généralement pas optimales pour le procédé réel. L'optimisation en temps réel est la famille de méthodes pour lesquelles on utilise les mesures disponibles sur le procédé étudié pour corriger, directement ou indirectement, les entrées optimales prédites par le modèle. La solution traditionnelle est de corriger les paramètres du modèle, et de ré-optimiser le modèle ainsi corrigé. Pour cela il est nécessaire d'estimer la valeur de ces paramètres en temps réel, ce qui est souvent difficile à implémenter. De plus, même si cette correction des paramètres est réalisable en pratique, la capacité du modèle à prédire les conditions d'optimalité du procédé réel peut toujours s'avérer insuffisante.

Une méthode itérative pour l'optimisation en temps réel appelée 'Modifier Adaptation' (ci-après MA) surmonte ces obstacles en utilisant des mesures, typiquement la valeur mesurée des contraintes et l'estimation des gradients du procédé, directement au niveau de la formulation du problème d'optimisation numérique.

Estimer des gradients expérimentaux est la difficulté principale rencontrée lors de l'application de MA à un procédé réel. La quantité de données expérimentales nécessaire pour estimer ces gradients croît avec le nombre d'entrées du procédé. Ceci rend la méthode difficilement applicable pour des procédés avec de nombreuses entrées. La contribution méthodologique principale de cette thèse est une nouvelle méthode appelée 'Directional' Modifier Adaptation (ci-après D-MA), qui propose de résoudre ce problème en n'estimant les gradients expérimentaux que selon certaines directions privilégiées. En dépensant moins d'effort sur l'estimation des gradients, l'accent est donc mis sur l'optimisation du procédé. Un algorithme appelé Dual Directional Modifier Adaptation (ci-après Dual D-MA) est développé. Dual D-MA optimise le procédée, tout en assurant que les points d'opération passées puissent servir pour estimer le gradient directionnelle avec fiabilité. Cet algorithme converge rapidement au voisinage de la solution optimale pour le procédé réel, même pour un procédé avec de nombreuses

entrées.
Même si MA utilise des mesures, il utilise néanmoins également un modèle du procédé. Une deuxième difficulté survient quand les entrées de ce modèle ne sont pas les mêmes que celles du procédé réel. La deuxième contribution méthodologique de cette thèse est 'Generalized' Modifier Adaptation (ci-après G-MA), une méthode qui permet de traiter ce cas. Il est démontré que G-MA permet d'éviter la ré-modélisation du procédé. Ainsi, G-MA permet, par exemple, qu'un modèle en boucle ouverte soit utilisé pour optimiser un procédé en boucle fermée, sans devoir modéliser la boucle de rétroaction.

L'algorithme Dual D-MA est appliqué à un système de cerf-volant expérimental. Les cerfs-volants, (i.e. les 'kites'), sont en voie de devenir une nouvelle technologie pour l'exploitation de l'énergie éolienne. Les applications à grande échelle incluent la traction de navires et la génération d'électricité en exploitant le vent à des altitudes hors de portée des éoliennes conventionnelles.
Alors que les cerfs-volants traditionnels, de loisir, sont dirigés manuellement, un fonctionnement complétement autonome est nécessaire pour produire de l'énergie avec des kites. Le premier défi est de concevoir des algorithmes de contrôle fiables pour les cerfs-volants, capables de fonctionner malgré des hauts niveaux de bruit, des retards purs, et des perturbations dues au vent. L'algorithme de contrôle doit maintenir la voile sur une trajectoire cyclique, malgré le fait qu'elle se déplace à très haute vitesse.
Un système de cerf-volant autonome de petite échelle a été construit pendant cette thèse. En deux ans, trente jours d'expériences à l'extérieur ont été effectués. Une nouvelle hypothèse de modélisation des kites a été validée, qui lie les déflections de pilotage à une diminution de la finesse (le rapport portance/trainée) du cerf-volant. Un contrôleur capable de suivre différentes trajectoires a été implémenté sur le système expérimental. Les expériences illustrent la fiabilité de ce contrôleur, en dépit de retards purs importants. La seule mesure utilisée par le contrôleur est la position de la voile. Dans ce travail, cette position est toute simplement dérivée au moyen de l'angle du câble reliant le cerf-volant à la terre.
Une approche à deux niveaux a été implémentée pour à la fois contrôler et optimiser le système expérimental. Dual D-MA met périodiquement à jour la trajectoire de référence qui sert de consigne au contrôleur, afin de maximiser la tension moyenne dans le câble. Malgré un niveau de bruit très élevé, l'algorithme a pu trouver la trajectoire de référence optimale pour la voile réelle en seulement 10 minutes, en assurant le respect de la contrainte sur l'altitude minimale. La tension moyenne résultante est environ 20% plus grande que celle obtenue en suivant la trajectoire optimale calculée avec le modèle du procédé. Une étude expérimentale comparant la tension produite par différentes trajectoires confirme l'importance de la forme de la trajectoire. Cette étude confirme en outre que la trajectoire à laquelle Dual D-MA converge correspond bien à l'optimum du

procédé réel.

Mots clefs: Optimisation en temps réel, Automatique, Airborne Wind Energy, Cerfs-Volants.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

1.1.1 Real-Time Optimization

The operators of industrial processes are continually faced with choices. For example, the operator in the control room of Lausanne's incineration plant asks himself: "Will we produce more electricity today if I increase the steam turbine's intermediate bleed pressure by 1 bar?" For the operator of an emulsion-polymerization batch reactor in Geneva, the question is: "If I maintain the reactor at its maximum temperature for 30 minutes longer, the reaction will take place faster, but will the average molecular weight of my polymer still be acceptable?" Researchers at EPFL studying fuel cells ask themselves: "For the current power demand, what hydrogen-to-oxygen fuel ratio will maximize the cell's electrical efficiency?" These questions are often tackled by constructing a mathematical model of the system. However, as engineers know, models are rarely perfect. In any case, processes evolve over time. The model of the incineration plant that was calibrated over the winter will no longer be correct during a mid-summer heat-wave. Likewise, raw materials obtained from a different supplier will behave differently in the polymerization reactor, throwing a carefully calibrated model off the mark. These processes need continuous monitoring and adjustment in order to keep them operating optimally; this is the aim of Real-Time Optimization (RTO). In a broad sense RTO refers to any means of actively adjusting a process in order to optimize its performance, in response to disturbances and process variations. This thesis is concerned with RTO using mathematical models and measurements.

Industry is increasingly turning to RTO for three reasons. Firstly, RTO improves process *efficiency* (and hence profitability). Globalization is leading to increasingly competitive markets. More competition requires processes to operate more efficiently in order to

remain profitable. For example, in Europe the classic thermal power plant is actively being replaced by far more efficient cogeneration (combined heat and power) power plants. Many new gas-turbine power plants in Europe are of the combined-cycle type, again in the interest of efficiency and, ultimately, profitability. Efficiency requires not only an efficient design, but also an efficient operating strategy, which is what RTO provides. Secondly, RTO is ideal for reacting to process *variability*. Process operating conditions are nowadays much more variable than in the past. In today's liberal markets, prices and product specifications change rapidly. A process may need to constantly adapt its mode of operation in response to its varying economic context. Electricity prices are a good example of this variability. When the market price is low, hydro-electric dams in Switzerland completely reverse their mode of operation, consuming electricity to pump water back uphill. Thirdly, RTO can ensure industrial processes satisfy operating constraints, and that their products satisfy a growing number of norms and safety regulations. For example, environmental regulations are much more stringent than in the past. Processes emissions must be actively monitored and any violation can result in severe financial penalties. Europe's largest steel works in Taranto, Italy, currently risks being shut down for violating emissions regulations.

There is potential for significant improvement in RTO algorithms. Increased global competitiveness, causing more frequent changes in plant operation, along with staffing reductions and tighter budgets, means industry requires more effective, easier to implement RTO algorithms than the current state-of-the-art (Darby et al., 2011). The current industry standard, the two-step approach, is to regularly update the parameters of the process model, and then to recompute the optimal mode of operation. However, this technique has a number of important drawbacks. Improved RTO algorithms should address the following issues:

1. *Constraint satisfaction* is paramount, as often, violating constraints can have harsh economic consequences (for example if emissions exceed regulatory levels). The industry standard for RTO, the two-step approach, cannot guarantee that operational constraints are satisfied (unless they are actively monitored by a lower level dynamic controller) as they may be poorly predicted by an incorrect model.

2. *Online diagnostics:* It is important to know why the RTO algorithm takes certain steps, and whether it has in fact reached an optimal solution for the plant. Due to a structurally incorrect model, the two-step approach may not satisfy any optimality measure for the plant (Forbes et al., 1994). It may even lead to a degradation in performance compared to not performing RTO!

3. *Convergence speed* of the RTO algorithm is crucial. This is determined by the

number of operational changes required before reaching a neighborhood of the plant optimum. In general, a settling time must be respected between operational changes. The assumption in RTO is that disturbance variations and parameter drift occur slowly with respect, not only to this settling time, but also with respect to the time the RTO algorithm takes to converge, allowing time-invariant models to be used. For example, disturbances may vary on a daily basis for a plant with a 30-minute settling time. Thus, the optimal operating conditions vary on a daily basis, and the RTO algorithm in this case should converge in no more than several hours, otherwise it cannot reject the optimality loss due to the daily variations affecting the plant.

4. *Design process:* The RTO design process should be relatively methodological and straightforward, as the person implementing RTO may not possess detailed insight into the process at hand. As the majority of modern RTO algorithms are reportedly based on rigorous process models (Darby et al., 2011), a methodological RTO design procedure should be based upon straightforward steps using this model.

In addition, it is desirable to develop RTO methods that do not require frequent online parameter estimation. Parameter estimation is certainly useful as it improves the quality of the process model, which may then be used for other off-line analyses. However, the accurate, automated estimation of many model parameters as required by the two-step approach is not only extremely complicated to implement, it is also at odds with the RTO layer's optimization objectives. If the model contains many uncertain parameters, it is difficult to ensure sufficient excitation in order to estimate these parameters, and doing so will detract from reaching the optimization objective.

An alternative RTO algorithm called Modifier Adaptation (MA) has emerged in response to the shortcomings of the two-step approach. This algorithm has been applied to a number of industrially relevant systems, with promising results. In particular MA provides an optimality guarantee for the *real process*. However, several barriers remain to widely applying MA to complex industrial processes. Firstly, MA requires the sensitivity of the plant to small changes in the operating conditions, i.e. experimental plant gradients, to be estimated. The estimation of these experimental plant gradients required by MA prohibits its application to processes with many inputs. Secondly, MA relies on the use of a model linking the plant inputs to performance. However, while the plant inputs (i.e. the variables available to optimize the plant) may be set-points, not all the control loops regulating a large plant can be accurately modeled, and it is possible that only an open-loop model of the process is available. Alternatively, a partial model of the process may be available. In both situations the model inputs will not be

the same as the plant inputs, a situation which, up until now, could not be handled by MA.

1.1.2 Kite Control

Currently, the gap between the actual energy-production capacity and the projected energy demand is one of the most serious global problems. It is predicted (International Energy Agency, 2014) there will be a 40% increase in global energy demand between 2012 and 2040. Electricity demand is expected to increase by 80% over the same period. Even this enormous increase in demand will not mean energy needs are met worldwide. It is predicted that in 2040, in sub-Saharan Africa alone, 530-million people will still remain without electricity.

The ever-increasing global thirst for energy is rapidly consuming the earth's fossil-fuel reserves. While energy production from renewable sources is expected to increase significantly between now and 2040, unfortunately it is predicted that fossil fuels will continue to dominate the power sector. This will obviously lead to a fuel shortage in the future, and in addition, the widespread use of fossil fuels pollutes the environment and is driving climate change. The renewable energy sector is expanding in response to the threats of an impending fuel crisis, environmental damage and climate change. While this is positive, it must be placed in the perspective of our increasing energy needs. Current renewable energy technologies are not expected to break our reliance on fossil fuels. Indeed, the current forecast, which takes into consideration the new national and international policies aimed at halting climate change, predicts that annual fossil-fuel consumption will *increase* between now and 2040.

Wind is one of the most promising renewable energy sources. A global study based on experimental data estimated that the world's energy demand could be entirely satisfied using conventional wind turbines by exploiting only 20% of the world's land sites with suitably strong (class 3 or greater) winds (Archer and Jacobson, 2005). As Archer and Jacobson (2005) estimate that only 13 % of the world's land area is class 3 or greater, this means that wind turbines installed on roughly only 2 % of the world's land area could produce an amount of electricity equal to the world's energy demand. However, in most locations the cost of wind energy is still significantly higher than that of energy produced from fossil sources. This is due to the high material and installation costs associated with wind turbines, and to the difficulty in identifying sites that are both remote enough and windy enough. This is why, despite the relative maturity of conventional wind-turbine technology, wind power only accounts for about 2% of global electricity production.

It is likely that only a breakthrough in renewable-energy production methods will put

an end to our reliance on fossil fuels. 'Airborne Wind Energy' is one of the radically different concepts currently being investigated. The aim is to exploit the fact that wind strength and regularity increases with altitude (Roberts et al., 2007; Archer, 2013; Van den Berg, 2005). Wind power density, which is proportional to the wind-speed squared, increases significantly with altitude close to the earth's surface. At a height of 500-1000 meters, the mean wind power density is, on average, roughly four times that at 50-150 meters (Fagiano and Milanese, 2012; Archer and Caldeira, 2009). The foremost concept for harnessing high-altitude wind power is by using kites. These are wings, ranging from flexible para-glider type designs to rigid composite aircraft wings, attached to the ground by a flexible tether. By flying perpendicular to the wind, similarly to the blades of a wind turbine, the wing experiences a large aerodynamic force. This force is transmitted to the ground via the tether, where it can be used to drive a generator, by slowly unreeling the tether. In a second phase, a small percentage of the generated electricity is used to reel the kite back in, while it remains static, generating very little tether force. This is known as *pumping-cycle* operation (Ruiterkamp and Sieberling, 2013). An alternative method is to place small, electricity producing wind turbines on the kite itself (Lind, 2013). Powerful kites have wider application than for electricity production; kite-propelled craft regularly break the world speed-sailing record and very large kites are even being used to propel cargo ships (Erhard and Strauch, 2013a).

A number of significant technical barriers must be overcome for kite generators to become a viable option. These range from designing suitable wings, to building robust all-weather ground stations. The automatic control of the kite is one of the most fundamental challenges. The type of kite used for power generation is inherently unstable; if it is not constantly steered, it will crash in a matter of seconds. An 'autopilot' must keep the kite flying in a wide variety of wind conditions. The control task is complicated to a large extent by the sensing challenges. Despite recent advances in compact inertial measurement units, it remains difficult to accurately determine the position and orientation of a kite flying at speeds in excess of 150 km/h and experiencing accelerations over 15 Gs. What is more, unlike the blades of a wind turbine which must move in a circle, a kite can follow many different flight paths. The path the kite flies determines how much power is produced. Thus, in addition to keeping the kite from crashing, the autopilot must ensure the kite follows a path that is efficient for power production.

1.2 State of the Art

1.2.1 Real-Time Optimization

The process industry is comprised of both continuous plants operating at near steady state, and transient plants. Batch, or semi-batch processes (François and Bonvin, 2013b) are examples of transient processes. These processes are typically multi variable, nonlinear and affected by measurement noise and process disturbances. In addition, they are subject to safety constraints (such as maximum temperatures or pressures), equipment constraints (such as actuator limits, compressor speeds or power saturation), quality constraints (such as product specifications) and environmental constraints (typically emissions levels). On-line computer-aided operation is widely used to manage these complex processes (Cutler and Perry, 1983; Darby et al., 2011). In addition to computer-aided multivariable control, computerized RTO can be used to track the plant's optimal operating conditions. These are likely to change over time due to gradual process change (for example, due to heat exchanger fouling or mechanical wear), low-frequency disturbances (such as climate variations or demand fluctuations), a changing economic context, or changes in production goals. To date, a large number of successful RTO application have been reported (Marlin and Hrymak, 1997; Darby et al., 2011), and it remains a very active research field.

Srinivasan et al. (2003a) gives a comprehensive review of RTO techniques and divides them into two categories: 'model-based' and 'model-free' techniques, depending on whether or not the process model is used explicitly for *on-line* calculations. Heuristic model-free evolutionary-search techniques were developed first (Box and Draper, 1969). These techniques use plant data to find 'improving directions' for the plant inputs. Since these techniques require no process model and only simple calculations, they can be implemented readily. However, evolutionary operation has difficulty handling large numbers of inputs, process constraints and complex nonlinear behavior. More recent model-free technique's are Self-Optimizing Control (SOC) (Skogestad, 2000; Alstad and Skogestad, 2007) and NCO tracking (François et al., 2005; Srinivasan and Bonvin, 2007), which use a process model off-line to select controlled variables that lead to near-optimal operation via multivariable feedback control.

Increased computational power led to the development of the original model-based algorithm, the so-called two-step approach (Chen and Joseph, 1987; Jang et al., 1987). Two steps are repeated online, namely, parameter estimation to update the model and optimization of the updated model to compute the optimal inputs. Although this approach can handle arbitrarily complex systems with many inputs, it is fairly computationally intensive. Despite the popularity of the two-step method, Forbes et al.

(1994) and Forbes and Marlin (1996) proved that the employed model must satisfy extremely stringent 'model adequacy' conditions for the RTO scheme to converge to the plant optimum. These conditions will almost never be satisfied in a practical setting, and in practice, there is no way of verifying them either. Agarwal (1997), Gao and Engell (2005b), and Marchetti (2009) showed that, in the presence of structural plant-model mismatch, parameter estimation may be ineffective and can even lead to worse performance than if no RTO was performed at all!

The pitfalls of the two-step approach are, for the most part, theoretical. In practice, it is likely, although this cannot be guaranteed, to perform well if a structurally accurate model with few uncertain parameters is available. It is the default RTO algorithm for industrial applications (Darby et al., 2011). However the two-step approach is unlikely to perform well if the model is quite inaccurate, or if the parameter estimation problem is difficult to solve, or if there are simply too many uncertain parameters in the model. For this reason, another class of model-based algorithm, which addresses the issues associated with the two-step approach, has developed in parallel. Roberts (1979) proposed a method called 'Integrated System Optimization and Parameter Estimation' (ISOPE), which uses measurements to update both the model parameters and the *gradient* of the cost function in the optimization problem to be solved on-line. It is thanks to these gradient modifiers that ISOPE can guarantee plant optimality in the presence of plant-model mismatch. A number of researchers have improved and extended the ISOPE algorithm over the next 20 years, and a good review of this development is given by Roberts (1995).

Tatjewski (2002) significantly simplified ISOPE by eliminating the parameter estimation step. This simpler algorithm was further refined to handle general plant constraints by Gao and Engell (2005a). Finally, Marchetti et al. (2009) provided a solid theoretical basis for the simplified ISOPE algorithm, by comprehensively dealing with tuning, convergence and optimality conditions. The result is the 'Modifier Adaptation' (MA) algorithm that has been successfully applied to a number of reasonably complex industrially relevant systems that include an experimental solid-oxide fuel-cell stack (Bunin et al., 2012), the simulated heat and power system of a sugar and ethanol plant (Serralunga et al., 2013), and a simulated oxygen-consumption plant (Navia et al., 2012). MA uses *modifier terms* to correct the values and the gradients of the cost and constraint functions in the model-based optimization problem. Many aspects of MA have been investigated further, such as approaches to deal with the estimation of gradients (Bunin et al., 2013; Marchetti, 2013; Rodger and Chachuat, 2011; Navia et al., 2013), extension to closed-loop systems (Costello et al., 2014), extension to discontinuous systems (Serralunga et al., 2014), use of convex models to ease the numerical optimization and enforce model

adequacy (François and Bonvin, 2013a), use of second-order modifiers (Faulwasser and Bonvin, 2014), and even promising preliminary results on sufficient conditions for global convergence (Bunin, 2014; Faulwasser and Bonvin, 2014).

Many RTO methods have primarily been developed for the more widespread continuous processes, but there is also a significant interest in applying RTO to transient processes, and the process control literature is rich with applications to semi-batch chemical processes (Ruppen et al., 1998; Filippi-Bossy et al., 1989; Ubrich et al., 1999), such as batch polymerization (Kadam et al., 2007; François et al., 2004; Zafiriou and Zhu, 1990; Clarke-Pringle and Mac Gregor, 1998), batch distillation (Welz et al., 2008), and fed-batch bio-processes (Visser et al., 2000; Bodizs et al., 2007). A review of RTO for transient processes is given by Bonvin et al. (2002). It is generally more difficult to apply RTO to a transient process, simply because it never reaches a steady state. The process model is dynamic, the model-based optimization problem is an optimal-control problem, and the measurements are distributed in time. Many of the RTO methods for continuous processes have at least to some extent been adapted to handle transient processes, such as the two-step approach for dynamic systems (Filippi-Bossy et al., 1989), dynamic ISOPE (Roberts, 1995), and dynamic SOC (de Oliveira et al., 2013; Jäschke et al., 2011). On the other hand, some RTO methods were specifically designed for the unique characteristics of transient processes, such as NCO tracking (Srinivasan and Bonvin, 2007), and optimizing variations of iterative learning control (Ge et al., 2000; Xiong and Zhang, 2005). Unfortunately none of these techniques are perfect, or even generally applicable. As is the case for continuous processes, the two-step approach may perform poorly if the model is structurally incorrect. In order to guarantee optimality, both dynamic ISOPE and optimizing iterative learning control rely on the repeated identification of a linear time-varying process model, yet this model is very difficult to obtain. Dynamic SOC, just like static SOC, requires an accurate disturbance model and is based on the assumption that the disturbances and the plant-model mismatch are small. NCO tracking assumes the optimal solution is comprised of a particular sequence of arcs, yet this sequence can change due to plant-model mismatch.

Scant attention has been paid to applying MA to transient processes. MA uses measurements to estimate the manner in which the process performance is locally influenced by the plant inputs, i.e. to estimate the plant gradients. For a transient process, the plant inputs are infinite-dimensional functions (usually of time), rather than finite-dimensional vectors (Deshpande et al., 2012), and it is unclear how the plant gradients should be obtained. One workaround is simply not to use gradient correction terms, conserving only the constraint bias (Marchetti et al., 2007). While this may work very well for certain processes, it may also perform poorly for others, as this approach cannot

provide any optimality guarantee for the plant upon convergence. Another approach is to parametrize the plant inputs using a finite-dimensional vector, which, at least in theory, allows plant gradients to be calculated with respect to the parameterizing vector. However, if the dimensionality of the parameterizing vector is large, it is unrealistic to estimate plant gradients, as the experimental cost would be prohibitive. The approach proposed by Chachuat et al. (2009) is to combine MA with the 'parsimonious'[1] parametrization from NCO tracking (Srinivasan and Bonvin, 2007) . This parsimonious parametrization exploits the fact that the solutions to dynamic optimization problems have a particular structure. Process intuition, or robustness analysis using the model, often confirms that this structure is unlikely to change for any 'likely' disturbance or plant-model mismatch scenario. The result is a small number of RTO decision variables. While attractive, this is a 'tailor-made' solution for each process, often requiring a high level of process insight.

1.2.2 Kite Control

A dynamically flying kite is a fast, unstable system influenced by unpredictable wind disturbances. It is a testament to the difficulty of stabilizing a kite during dynamic flight that the first successful account of experimental kite control was published in 2013 (Erhard and Strauch, 2013a), 33 years after research on kite power began (Loyd, 1980).

Initial development in the field of kite control was to a large extent focused on Nonlinear Model Predictive Control (NMPC) (Diehl, 2001; Ilzhöfer et al., 2007; Canale et al., 2010). This seemed a logical choice, given the complex nature of the control problem, the presence of numerous operating constraints, and the likelihood of significant time delays in a practical implementation. However, to date, there is no reported practical implementation of an NMPC controller for kites. This is probably in part due to the inaccuracy of existing kite models, particularly for kites made from flexible material, as NMPC relies heavily upon the model. While attempts have been made to accurately model the behavior of a flexible kite (Breukels et al., 2013; Gohl and Luchsinger, 2013), the result is a very complex, high-dimensional model, unsuitable for NMPC.

When it comes to practical implementation, simpler geometric approaches have so far proved more successful than NMPC. Indeed, a number of experimentally-validated geometric control laws have very recently been published. Baayen and Ockels (2012) observed that a simple control scheme should aspire to control the kite's direction of

[1] The parsimonious parametrization used in NCO tracking segments the optimal solution into different arcs. A control structure ensures that any active path constraints are tracked during the appropriate arcs. Typically, it is then sufficient to adapt the switching times between the different arcs from one batch to the next, in order to optimize the plant's performance and satisfy terminal constraints.

motion, referred to as the velocity angle (this is defined in Chapter 5). They combined an on-line system-identification algorithm with a Lyapunov-based control law. The control law attempts to choose the kite's velocity angle such that it smoothly attains the prescribed target trajectory. An additional contribution was to elegantly exploit the concept of geodesic curvature to simplify the problem of tracking on a sphere. Although the approach showed promise in simulation, in practice it proved unable to cope with time delays and actuator constraints. Erhard and Strauch (2013a) developed a simple, robust cascade controller for kites, which was tested by years of sea trials on large vessels. Essentially, a low-level proportional controller regulates the kite's orientation (i.e. the direction the kite is *pointing*), while a higher-level guidance controller chooses the bang-bang reference orientation signal, based on the kite's current position. This results in a horizontal figure-of-eight pattern, which is generally considered to be the most efficient type of path for extracting energy from the wind. The resulting controller has only a few tuning parameters, however the effect of these parameters on the kite's trajectory is difficult to determine a priori. A very similar cascade-control strategy was proposed and experimentally validated on a small prototype by Fagiano et al. (2014). The primary controlled variable was the kite's velocity angle in this case, which was again regulated by a simple low-level linear controller. The guidance strategy alternately directs the kite towards one of two points, producing the classic figure-of-eight pattern. The tuning parameters in this case can be used to choose the height, width and inclination of the figure-of-eight. The authors extended this control law to also handle the retraction phase for a pumping-cycle generator, and successfully implemented the algorithm on a power-producing prototype (Zgraggen et al., 2014).

Jehle and Schmehl (2014) proposed a more advanced path-following controller using a nonlinear guidance-law and successfully tested it on a 20kW pumping-cycle prototype. Feedback linearization is used to design a lower-level velocity-angle controller. The guidance law aims to minimize the cross-track error, taking into account the kite's current velocity angle, the path's direction, and the curvature of the path. Successful implementation of a path-following controller is also reported by Ruiterkamp and Sieberling (2013), this time via way-point tracking for rigid wings.

While several control solutions for kites now exist, and the most advanced of these are even capable of tracking relatively arbitrary paths, the path-planning problem is still very much unsolved. Intelligent path planning is important because, although the kite is free to follow almost any flight path, it is the flight path that directly determines the aerodynamic force the kite experiences, and hence the power generated. Experimental studies (Zgraggen et al., 2013) have confirmed that the path taken by the kite significantly affects the power it can generate. The path-planning problem

results in an interesting optimal control problem that has been studied by a number of authors (Diehl, 2001; Houska and Diehl, 2006, 2007; Williams et al., 2008; Argatov and Silvennoinen, 2010; Dadd et al., 2011). While these studies yield useful qualitative results, only an approximate optimal path can be calculated off-line using the simplified models these studies employ. Recently, more detailed models are being employed (Horn et al., 2013), particularly for rigid wings for which modeling is more straightforward. Nonetheless, modeling for kites is still quite approximate, and it is questionable whether a purely model-based approach can really calculate optimal paths for a real system. It would be difficult, if not impossible, to ensure a model could perfectly describe wind gradients (which vary depending on weather and location) and the kite's behavior in all flight conditions. Indeed, the only currently available experimental study uses a much more modest, ad-hoc approach to tune the path the kite follows; Zgraggen et al. (2013) proposed an algorithm that adjusts the height and lateral position of the kite's path in real time, using experimental data only. These two parameters are essentially optimized on-line using a primitive gradient-search algorithm.

1.3 Contributions of the Thesis

The main contributions of this thesis are a novel RTO methodology, and its experimental validation. However, the experimental work with kites became something of a passion, and a long and fruitful foray into the domain of kite modeling and control led to some secondary contributions in these domains.

1.3.1 Main Contributions

- A reformulation of the standard MA algorithm, called 'Generalized' MA, that allows it to be applied to a process whose inputs are not the same as the inputs of the available model.

- A novel 'dual directional-MA' (Dual D-MA) algorithm that can be used to optimize processes with many inputs, and in particular, transient processes.

- Experimental results demonstrating that a two-layer optimizing control scheme based on Directional MA can significantly increase the performance of a repetitive transient process, in this case a power kite flying a repetitive pattern.

1.3.2 Secondary Contributions

- A new experimentally validated modeling relationship for kites, linking the decrease in the kite's lift-to-drag ratio to the steering deflection.
- An experimentally validated path-following controller for kites, which is capable of accurately tracking arbitrary paths despite significant time delay. The controller only requires measurements of the kite's line angles, or alternatively, of the kite's position.

1.4 Organization of the Thesis

The next chapter presents a number of preliminary results upon which the rest of this thesis is built. This includes the MA algorithm and a dynamic model for kites.

Chapter 3 develops the Generalized MA method. Two alternative approaches are described, and they are illustrated and compared on a simulated continuous reactor.

Chapter 4 contains the main methodological contribution of this thesis, Directional MA. The basic principle is first explained, and some important theoretical properties are demonstrated. Then a practically applicable Dual D-MA algorithm, which integrates a novel gradient estimation method, is developed. The method is illustrated on a simulated dynamically flying kite.

Chapter 5 applies the Dual D-MA algorithm to an experimental small-scale kite system. In fact, this chapter describes the entire implementation of a two-layer optimizing control scheme, including system modeling, state estimation, delay compensation, path-following control and, finally, RTO via Dual D-MA.

Chapter 6 concludes the thesis, and discusses the perspectives for future work.

2 Preliminaries

RTO methods can be applied either to continuous processes, or to transient (discontinuous) processes. In the case of continuous processes, the RTO scheme aims to find the optimal *steady-state values* for the plant inputs, which often correspond to set-points for lower-level controllers. If the process is operated in transient mode, a steady state is intentionally never reached, and the RTO layer attempts to find the optimal *time-varying profiles* for the plant inputs. In this thesis, only *periodic* transient systems are considered, such as repeated batch or semi-batch reactors. It is only at the end of each period that the effect of the current input profile on the cost and constraints can be determined, so each RTO iteration corresponds to one period.

Static optimization theory provides a framework for characterizing a continuous process's optimal operating conditions, and the properties of an optimal solution are described in Section 2.1. Transient processes, on the other hand, are treated by the theory of dynamic optimization. However, in this thesis, dynamic optimization problems will be approximated by static optimization problems. The approximation procedure for a typical dynamic optimization problem is described in Section 2.2. If a model of the process is available, numerical optimization can be used to approximately compute the optimal operating conditions. It is an approximate solution because the model never perfectly matches the real process. The MA algorithm, which compensates for this mismatch using measurements, is described in Section 2.3. Finally, a model to describe a dynamically flying kite is described in Section 2.4.

2.1 Static Optimization

The problem of finding optimal steady-state operating conditions for a continuous process is typically expressed mathematically as:

$$\begin{aligned} \mathbf{u}_{\mathrm{p}}^{*} &:= \arg\min_{\mathbf{u}} \phi_{\mathrm{p}}(\mathbf{u}) \\ \text{subject to} \quad & \mathbf{g}_{\mathrm{p}}(\mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{2.1.1}$$

where $\mathbf{u}$ is the n_u-dimensional vector of inputs, ϕ_{p} is the cost function and $\mathbf{g}_{\mathrm{p}}$ is the n_g-dimensional vector of process constraints. Here, the subscript $(\cdot)_{\mathrm{p}}$ indicates a quantity related to the plant, and this will be referred to as the plant optimization problem. It is assumed throughout this thesis that ϕ_{p} and $\mathbf{g}_{\mathrm{p}}$ are twice continuously differentiable. Throughout this thesis, the 'inputs' are the degrees of freedom available to optimize the process. For example, these may be flow rates, feed rates, voltages or set-points for low-level controllers.

Theorem 2.1.1 (KKT Necessary Conditions)**.** *Let $\mathbf{u}_{\mathrm{p}}^{*}$ be a (local) optimum of Problem 2.1.1, and assume that $\mathbf{u}_{\mathrm{p}}^{*}$ is a regular point of the constraints, that is, the active constraints are linearly independent:*

$$\operatorname{rank}\left(\operatorname{diag}(\mathbf{g}_{\mathrm{p}})\frac{\partial \mathbf{g}_{\mathrm{p}}}{\partial \mathbf{u}}\right) = n_g. \tag{2.1.2}$$

Then, there exist unique values for the n_g-dimensional vector of Lagrange multipliers, $\boldsymbol{\nu}$, such that the following first-order Karush-Kuhn-Tucker (KKT) conditions hold at $\mathbf{u}_{\mathrm{p}}^{*}$:

$$\mathbf{g}_{\mathrm{p}} \leq 0, \tag{2.1.3}$$

$$\boldsymbol{\nu}^{T}\mathbf{g}_{\mathrm{p}} = 0, \tag{2.1.4}$$

$$\boldsymbol{\nu} \geq \mathbf{0}, \tag{2.1.5}$$

$$\frac{\partial \mathbf{L}_{\mathrm{p}}}{\partial \mathbf{u}} = \mathbf{0}, \tag{2.1.6}$$

with the Lagrangian function defined as: $\mathbf{L}_{\mathrm{p}}(\mathbf{u},\boldsymbol{\theta},\boldsymbol{\nu}) = \phi_{\mathrm{p}}(\mathbf{u},\boldsymbol{\theta}) + \boldsymbol{\nu}^{T}\mathbf{g}_{\mathrm{p}}(\mathbf{u},\boldsymbol{\theta})$.

Proof. See, for example, (Luenberger and Ye, 2008). □

The four KKT conditions are referred to as the primal feasibility, complementary slackness, dual feasibility and stationarity conditions, respectively (in the order given above). These conditions must hold at any local minimum *that is also a regular point of the constraints*. As these conditions are not sufficient, they may be satisfied by a point that is not a local minimum.

2.2 Dynamic Optimization

Many industrial processes never operate at steady state. This is the case for batch and semi-batch chemical process, which account for an important portion of the process industry. Robot manipulators performing repetitive tasks, and power-producing kites flying repetitive paths do not reach an equilibrium state either. These are periodic transient processes. The problem of finding optimal operating conditions for a transient process can be expressed mathematically as follows (Srinivasan et al., 2003b):

$$\begin{aligned} \mathbf{w}_{\mathrm{p}}^{*}(\cdot) := \arg\min_{\mathbf{w}(\cdot)} \quad & J_{\mathrm{p}}\big(\mathbf{w}(\cdot)\big) \\ \text{subject to} \quad & \mathbf{S}_{\mathrm{p}}(t, \mathbf{w}(\cdot)) \leq \mathbf{0} \qquad \forall\ t \in [0, t_f], \\ & \mathbf{T}_{\mathrm{p}}(\mathbf{w}(\cdot)) \leq \mathbf{0}, \end{aligned} \tag{2.2.1}$$

where J_{p} is the terminal cost (note that problems with a running cost can be reformulated into this form), $\mathbf{w}(t)$ is the n_w-dimensional time-varying vector of decision variables at time t, $\mathbf{S}_{\mathrm{p}}$ is the vector of path constraints, and $\mathbf{T}_{\mathrm{p}}$ is the vector of terminal constraints. The notation $\mathbf{w}(\cdot)$ is used to indicate the *function* mapping t to $\mathbf{w}$, for all $t \in [0, t_f]$. The theory of dynamic optimization deals with the solution to this problem, and a continuous-time equivalent of the KKT necessary conditions, called Pontryagin's Maximum Principle, exists. However, complex dynamic optimization problems are generally discretized and approximated by static optimization problems, because a static optimization problem is much easier to solve numerically.

The discretization process involves representing the (infinite-dimensional) input function $\mathbf{w}(\cdot)$ using a finite-dimensional input vector $\mathbf{u}$ (i.e. parameterizing the input profile). This is commonly done by first dividing the time horizon into n_s control stages:

$$t_0 < t_1 < t_2 < \cdots < t_{n_s} = t_f, \tag{2.2.2}$$

and then using low-order polynomials on each interval

$$\mathbf{w}(t) = \mathcal{P}(t, \hat{\mathbf{u}}^j), \qquad t_{j-1} \leq t < t_j \quad \forall\, j \in \{1, 2, \ldots n_s\}, \tag{2.2.3}$$

with the vector $\hat{\mathbf{u}}^j \in \mathbb{R}^{n_w \times (M+1)}$ and the polynomial function $\mathcal{P}$ of order M. The discrete decision variable is the vector:

$$\mathbf{u} = \begin{bmatrix} \hat{\mathbf{u}}^1 \\ \hat{\mathbf{u}}^2 \\ \vdots \\ \hat{\mathbf{u}}^{n_s} \end{bmatrix} \quad \in \quad \mathbb{R}^{n_u}, \tag{2.2.4}$$

with $n_u = n_s \times n_w \times (M+1)$.

The function $\mathbf{w}(\cdot)$ is now *parametrized* by $\mathbf{u}$ through the relationship $\mathcal{W}$ of the form:

$$\mathbf{w}(t) = \mathcal{W}(t, \mathbf{u}) := \left\{ \mathcal{P}(t, \hat{\mathbf{u}}^j) \quad \text{for} \quad t_{j-1} \le t < t_j \quad \forall\, j \in \{1, 2, \dots n_s\} \right\}. \tag{2.2.5}$$

In a similar manner, the continuous (infinite-dimensional) path constraints $\mathbf{S}_\mathrm{p}$ can be approximated by *point-wise* constraints, i.e. they are only enforced at n_c time instants, called here collocation times:

$$\hat{\mathbf{g}}^i(\mathbf{u}) = \mathbf{S}_\mathrm{p}\left(t_i, \mathcal{W}(\cdot, \mathbf{u})\right), \quad i = 1, 2, \dots, n_c. \tag{2.2.6}$$

The cost and constraint function for the discretized problem then read:

$$\phi_\mathrm{p}(\mathbf{u}) = J_\mathrm{p}\left(\mathcal{W}(\cdot, \mathbf{u})\right), \tag{2.2.7}$$

$$\mathbf{g}_\mathrm{p}(\mathbf{u}) = \begin{bmatrix} \hat{\mathbf{g}}^1(\mathbf{u}) \\ \hat{\mathbf{g}}^2(\mathbf{u}) \\ \vdots \\ \hat{\mathbf{g}}^{n_c}(\mathbf{u}) \\ \mathbf{T}_\mathrm{p}\left(\mathcal{W}(\cdot, \mathbf{u})\right) \end{bmatrix}. \tag{2.2.8}$$

If the discretization is sufficiently dense (i.e. n_u and n_c are sufficiently large), then the optimal vector of decision variables for the discretized problem, $\mathbf{u}_\mathrm{p}^*$, results in near-optimal performance, that is,

$$J_\mathrm{p}\left(\mathcal{W}(\cdot, \mathbf{u}_\mathrm{p}^*)\right) \simeq J_\mathrm{p}\left(\mathbf{w}_\mathrm{p}^*(\cdot)\right). \tag{2.2.9}$$

In addition, the constraint violation in-between constraint collocation points is negligible:

$$\mathbf{S}_\mathrm{p}\left(t, \mathcal{W}(\cdot, \mathbf{u}_\mathrm{p}^*)\right) \lessapprox \mathbf{0} \quad \forall\ t \in [0, t_f]. \tag{2.2.10}$$

It is important to note that the dimensionality of $\mathbf{u}$ is invariably quite large after this discretization procedure. Even for $M = 0$, which corresponds to a piecewise-constant input parametrization, for a typical dynamic optimization problem it is necessary to choose $n_u > 20$ in order to ensure Conditions (2.2.9) and (2.2.10) are satisfied.

2.3 Modifier Adaptation

2.3.1 Basic Modifier Adaptation

The functions ϕ_p and $\mathbf{g}_\mathrm{p}$ are usually not known accurately, as only the models ϕ and $\mathbf{g}$ are available. Consequently, an approximate solution to the original problem (2.1.1) is obtained by solving the following model-based problem:

$$\begin{aligned} \mathbf{u}^*(\boldsymbol{\theta}) := \arg\min_{\mathbf{u}} \; & \phi(\mathbf{u},\boldsymbol{\theta}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{u},\boldsymbol{\theta}) \leq \mathbf{0}, \end{aligned} \tag{2.3.1}$$

where $\boldsymbol{\theta}$ is an n_θ-dimensional vector of uncertain model parameters. If the model matches the plant perfectly, solving Problem (2.3.1) provides a solution to Problem (2.1.1). Unfortunately, this is rarely the case, since the structure of the model functions ϕ and $\mathbf{g}$ as well as the nominal values, $\boldsymbol{\theta}_0$, for the uncertain model parameters are likely to be incorrect. This structural and parametric mismatch implies that the model-based optimal input $\mathbf{u}^*(\boldsymbol{\theta})$ will probably not correspond to $\mathbf{u}^*_\mathrm{p}$, not only for $\boldsymbol{\theta} = \boldsymbol{\theta}_0$, but for *any* values of the model parameters $\boldsymbol{\theta}$.

MA collects process information to correct for the differences between the plant and the model optimization problems. This is done by applying successively different values of $\mathbf{u}$ to the plant, each time waiting for the plant to settle to steady state and observing its performance. The measured cost and constraints corresponding to the input $\mathbf{u}_k$ at the kth iteration are:

$$\tilde{\phi}_\mathrm{p}(\mathbf{u}_k) = \phi_\mathrm{p}(\mathbf{u}_k) + d_k^\phi \tag{2.3.2}$$

$$\tilde{g}_{\mathrm{p},j}(\mathbf{u}_k) = g_{\mathrm{p},j}(\mathbf{u}_k) + d_k^{g,j}, \quad \forall\, j \in [1,\ldots,n_g] \tag{2.3.3}$$

where d_k^ϕ and $d_k^{g,j}$ are realizations of a zero-mean random variable for the cost and the j^{th} constraint, respectively, with the corresponding variances σ_ϕ^2 and $\sigma_{g,j}^2$. This stochastic component represents high-frequency noise due to measurement noise and high-frequency disturbances affecting the plant. The process measurements are used to iteratively modify the model-based problem (2.3.1) in such a way that, upon convergence, the necessary conditions of optimality (NCO) for the *modified* problem match those for the plant-based problem (2.1.1). This is made possible by using modifiers that, at each iteration, are computed as the differences between the measured and predicted values of the constraints and the measured and predicted cost and constraint gradients. This forces the cost and constraints in the model-based optimization problem to locally

match those of the plant. In its simplest form, the algorithm proceeds as follows:

Algorithm 2.3.1: Modifier Adaptation (Marchetti et al., 2009)

Initialize the modifier terms: the n_g-dimensional vector of zeroth-order constraint modifiers $\boldsymbol{\epsilon}_0 = \mathbf{0}$, the n_u-dimensional vector of first-order cost modifiers $\boldsymbol{\lambda}_0^{\phi} = \mathbf{0}$, and the $(n_u \times n_g)$ matrix of first-order constraint modifiers $\boldsymbol{\lambda}_0^{g} = \mathbf{0}$. Choose the modifier filter matrices $\mathbf{K}^{\epsilon}, \mathbf{K}^{\phi}, \mathbf{K}^{\mathrm{g}}$, typically diagonal matrices with eigenvalues in the interval $(0, 1]$. Also, choose arbitrarily $\mathbf{u}_0 = \mathbf{0}$.

for $k = 1 \rightarrow \infty$

1. Solve the modified model-based optimization problem

$$\begin{aligned} \mathbf{u}_k := \underset{\mathbf{u}}{\operatorname{argmin}} \quad & \phi_{\mathrm{m},k-1}(\mathbf{u}) \\ \text{subject to} \quad & \mathbf{g}_{\mathrm{m},k-1}(\mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{2.3.4}$$

   where the modified cost and constraints are given by

$$\phi_{\mathrm{m},k}(\mathbf{u}) := \phi(\mathbf{u}, \boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_k^{\phi})^T (\mathbf{u} - \mathbf{u}_k), \tag{2.3.5}$$

$$\mathbf{g}_{\mathrm{m},k}(\mathbf{u}) := \mathbf{g}(\mathbf{u}, \boldsymbol{\theta}_0) + \boldsymbol{\epsilon}_k + (\boldsymbol{\lambda}_k^{\mathrm{g}})^T (\mathbf{u} - \mathbf{u}_k). \tag{2.3.6}$$

   The subscript $(\cdot)_{\mathrm{m}}$ indicates a quantity that has been modified.

2. Apply the input $\mathbf{u}_k$ to the plant to obtain $\tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k)$ and $\tilde{\mathbf{g}}_{\mathrm{p}}(\mathbf{u}_k)$.

3. Obtain an estimate of the plant cost gradient, $\nabla\phi_{\mathrm{E},k}$, and the plant constraint gradient, $\nabla\mathbf{g}_{\mathrm{E},k}$ (these gradient estimates are for the *current* operating point $\mathbf{u}_k$). The gradients must be estimated using measurements collected at no less than n_u different operating points close to $\mathbf{u}_k$ (to be further discussed in Section 2.3.2).

4. Update the modifier terms using the following first-order filter equations[1]:

$$\boldsymbol{\epsilon}_k := (\mathbf{I}_{n_g} - \mathbf{K}^{\epsilon})\boldsymbol{\epsilon}_{k-1} + \mathbf{K}^{\epsilon}\left(\tilde{\mathbf{g}}_{\mathrm{p}}(\mathbf{u}_k) - \mathbf{g}(\mathbf{u}_k, \boldsymbol{\theta}_0)\right), \tag{2.3.7}$$

$$\boldsymbol{\lambda}_k^{\phi} := (\mathbf{I}_{n_u} - \mathbf{K}^{\phi})\boldsymbol{\lambda}_{k-1}^{\phi} + \mathbf{K}^{\phi}\left(\nabla\phi_{\mathrm{E},k} - \nabla\phi(\mathbf{u}_k, \boldsymbol{\theta}_0)\right)^T. \tag{2.3.8}$$

$$\boldsymbol{\lambda}_k^{\mathrm{g}} := (\mathbf{I}_{n_u} - \mathbf{K}^{\mathrm{g}})\boldsymbol{\lambda}_{k-1}^{\mathrm{g}} + \mathbf{K}^{\mathrm{g}}\left(\nabla\mathbf{g}_{\mathrm{E},k} - \nabla\mathbf{g}(\mathbf{u}_k, \boldsymbol{\theta}_0)\right)^T, \tag{2.3.9}$$

end

[1] There is also a variant of the MA algorithm which filters the inputs, rather than the modifier terms (Marchetti, 2009).

The MA algorithm's most attractive property is that, if the scheme converges, then, under ideal circumstances, it will do so to a KKT point *for the plant*.

Theorem 2.3.1 (KKT matching)**.** *Let the gain matrices* $\mathbf{K}^{\epsilon}, \mathbf{K}^{\phi}, \mathbf{K}^{\mathrm{g}}$ *be nonsingular. Assume no high-frequency noise and perfect gradient estimates, i.e.* $\nabla\phi_{\mathrm{E},k} = \nabla\phi_{\mathrm{p}}(\mathbf{u}_k), \nabla\mathbf{g}_{\mathrm{E},k} = \nabla\mathbf{g}_{\mathrm{p}}(\mathbf{u}_k)$*. If Algorithm 2.3.1 converges, with* $\mathbf{u}_\infty := \lim_{k\to\infty}\mathbf{u}_k$ *being a KKT point for the modified problem* (2.3.4)*, then* $\mathbf{u}_\infty$ *is also a KKT point for the plant optimization problem* (2.1.1)*.*

Proof. See Marchetti et al. (2009). □

Note that, while the KKT-matching property is a very desirable property for a RTO algorithm, it remains a theoretical result. In a real application, due to high-frequency noise, the algorithm will converge to a neighborhood of the plant optimum.

The KKT matching theorem guarantees that, *if* MA converges, it will converge to a plant KKT point. But *can* the algorithm converge to the plant optimum? In the field of RTO, this is referred to as the 'Model Adequacy' question (Forbes and Marlin, 1996). In the case of MA, the process model $\{\phi, \mathbf{g}\}$ is called adequate if values for the modifiers $\boldsymbol{\epsilon}, \boldsymbol{\lambda}^{\phi}$ and $\boldsymbol{\lambda}^{g}$ can be found such that a fixed point of the MA algorithm 2.3.1 coincides with the plant optimum $\mathbf{u}_{\mathrm{p}}^{*}$. This results in a (fairly relaxed) requirement that the model must satisfy.

Theorem 2.3.2 (Model Adequacy)**.** *Let* $\mathbf{u}_{\mathrm{p}}^{*}$ *be the unique plant optimum, which is assumed to be a regular point for the* n_g^a *active constraints. The process model is adequate for use in MA if the reduced Hessian of the cost function* ϕ *is positive definite at* $\mathbf{u}_{\mathrm{p}}^{*}$*:*

$$\mathbf{Z}^T \left(\nabla^2\phi\right)\mathbf{Z} > \mathbf{0}, \tag{2.3.10}$$

where the columns of $\mathbf{Z} \in \mathbb{R}^{n_u \times (n_u - n_g^a)}$ *are a set of basis vectors for the null space of the Jacobian of the* active *constraints in the model-based optimization problem* (2.3.1)*.*

Proof. See Marchetti et al. (2009). □

The Model Adequacy Condition is automatically satisfied if the model cost function is convex. Indeed, François and Bonvin (2013a) propose a method to enforce the Model

Adequacy Condition for a general non-linear model cost function by using convex approximations.

We now know that MA will *only* converge to a plant KKT point, and that, assuming the model-adequacy criterion is met, it *can* converge to the plant optimum. But *will* the MA scheme converge at all? This is a difficult question to answer. Both necessary and sufficient conditions have been proposed for the convergence of MA (Marchetti et al., 2009; Faulwasser and Bonvin, 2014; Bunin, 2014). Unfortunately, none of these conditions are very satisfactory from a practical point of view, as they are generally impossible to verify/enforce in practice. The exponential filter matrices $\mathbf{K}^{\epsilon}, \mathbf{K}^{\phi}, \mathbf{K}^{\mathrm{g}}$ are the tuning parameters that influence convergence. These matrices should be chosen with real, positive eigenvalues in the interval $(0, 1]$. Larger eigenvalues encourage more rapid convergence, but may also cause oscillating behavior, or a failure to converge at all. Smaller eigenvalues result in the MA algorithm taking more cautious steps, making convergence more likely, but slower. Currently, the only viable option is to tune these filter matrices through simulation or experimental trials.

Finally, note that an innovative method named'Nested' Modifier Adaptation has recently been proposed (Navia et al., 2014, 2013), which completely avoids the gradient estimation step. Rather, the gradient modifiers $\boldsymbol{\lambda}_k^{\phi}$ and $\boldsymbol{\lambda}_k^{g}$ are determined at each iteration by an unconstrained gradient-free optimization routine, such as the simplex method. This optimization routine adjusts the gradient modifiers at each RTO iteration in order to optimize the plant's measured performance. While this conveniently avoids gradient estimation, the drawback is that the gradient-free optimization algorithm must optimize the plant using $n_u(n_g + 1)$ decision variables. Due to this large number of decision variables, it may take many RTO iterations to converge to the plant optimum.

2.3.2 Gradient Estimation

In terms of implementation, gradient estimation is of more concern than theoretical convergence results. This is certainly the most difficult aspect of applying MA in practice. In the general context of RTO, gradient estimates can be obtained in many different manners (François et al., 2012; Mansour and Ellis, 2003; Bunin et al., 2013). Here, the discussion is limited to the techniques that have been most associated with MA. The basic method is to use finite differences. For example, using the forward finite-difference formula, the derivative of the plant cost[2] in the i^{th} direction of the input space, i.e. the

[2]Only the cost gradient is considered in this section. The procedure for estimating the constraint gradients is identical.

i^{th} element of $\nabla\phi_{\text{E},k}$, is estimated as:

$$\left(\frac{\partial\phi}{\partial u_i}\right)_{\text{E},k} = \frac{\tilde{\phi}_{\text{p}}(\mathbf{u}_k + \delta\mathbf{u}_i) - \tilde{\phi}_{\text{p}}(\mathbf{u}_k)}{\|\delta\mathbf{u}_i\|}, \tag{2.3.11}$$

where $\delta\mathbf{u}_i$ is a vector aligned with the i^{th} input direction. This generally requires n_u additional evaluations of the plant cost around each RTO point. Depending on the values of n_u and the plant settling time, the experimental cost may be unacceptable.

An alternative consists of computing the gradients solely from measurements collected at previously visited RTO points. For example, given $n_u + 1$ past input/measurement pairs, the cost gradient can be estimated by fitting a vector to the data (Marchetti et al., 2010):

$$\nabla\phi_{\text{E},k} = \begin{bmatrix} \tilde{\phi}_{\text{p}}(\mathbf{u}_k) - \tilde{\phi}_{\text{p}}(\mathbf{u}_{k-1}) \\ \tilde{\phi}_{\text{p}}(\mathbf{u}_k) - \tilde{\phi}_{\text{p}}(\mathbf{u}_{k-2}) \\ \vdots \\ \tilde{\phi}_{\text{p}}(\mathbf{u}_k) - \tilde{\phi}_{\text{p}}(\mathbf{u}_{k-n_u}) \end{bmatrix}^T \mathbf{U}_{k-1}^{-1}(\mathbf{u}_k), \quad . \tag{2.3.12}$$

with $\mathbf{U}_k(\mathbf{u}) = [\mathbf{u}-\mathbf{u}_k, \mathbf{u}-\mathbf{u}_{k-1}, \ldots, \mathbf{u}-\mathbf{u}_{k-n_u+1}]$. The matrix inverse in the above equation will become ill conditioned if the past points do not extend evenly in all directions of the input space. This ill conditioning can lead to very erroneous gradient estimates. Another technique, which does not require matrix inversion, is the rank-1 Broyden update (Rodger and Chachuat, 2011). In this case, the gradient estimate is updated in one direction only at each RTO iteration:

$$\nabla\phi_{\text{E},k} = \nabla\phi_{\text{E},k-1} + \frac{\tilde{\phi}_{\text{p}}(\mathbf{u}_k) - \tilde{\phi}_{\text{p}}(\mathbf{u}_{k-1}) - \nabla\phi_{\text{E},k-1}(\mathbf{u}_k - \mathbf{u}_{k-1})}{\|\mathbf{u}_k - \mathbf{u}_{k-1}\|^2}(\mathbf{u}_k - \mathbf{u}_{k-1})^T. \tag{2.3.13}$$

Finally, it is worth noting that there is no *redundancy* in any of the above gradient estimation methods: in the cases of the finite-differences technique and Equation (2.3.12), n_u measurements are used to estimate an n_u-dimensional gradient, while the Broyden update uses 1 measurement to preform a rank-1 update. This means that the methods are not well suited to dealing with significant high-frequency noise.

2.3.3 Dual Control

While it might appear that by using previously visited RTO points the gradients can be estimated 'for free', that is, without any additional experimental burden, in reality the steps taken by the RTO algorithm must be severely constrained to ensure good gradient estimates. 'Dual MA' algorithms attempt to guarantee accurate gradient estimates at

every iteration, by including constraints on the quality of the gradient estimates in the modified model-based optimization problem (Marchetti et al., 2010; Rodger and Chachuat, 2011; Marchetti, 2013)[3]. These Dual MA algorithms have two objectives: a) optimize the process, b) ensure accurate gradient estimates. Unfortunately, as we will see, these are usually conflicting objectives. A new 'dual' constraint $\mathbf{g}_{\mathrm{d},k-1}(\mathbf{u}) \leq \mathbf{0}$ is added to the model-based problem (2.3.4) in the MA algorithm. The aim of this constraint is to guarantee the accuracy of the gradient estimates. Two different dual constraint are discussed next, each one specific to the gradient-estimation method being used. In both cases, two distinct types of error are distinguished: the *truncation* error and the *noise* error. The truncation error occurs due to the curvature of the plant cost function, while all the gradient estimation equations in the preceding section assume it is locally linear. The noise error is due to the high-frequency noise affecting the measurements.

The first formulation was devised by Marchetti et al. (2010), and is specifically tailored to the case where the gradient is estimated using Equation (2.3.12). The authors proved that, at iteration k, the gradient estimation error is bounded as follows:

$$\|\nabla\phi_{\mathrm{E},k} - \nabla\phi_{\mathrm{p}}(\mathbf{u}_{\mathrm{k}})\| \leq \varepsilon_k^{\mathrm{t}}(\mathbf{u}) + \varepsilon_k^{\mathrm{n}}(\mathbf{u}), \tag{2.3.14}$$

where $\varepsilon_k^{\mathrm{t}}(\mathbf{u})$ is the truncation error bound, and $\varepsilon_k^{\mathrm{n}}(\mathbf{u})$ is the noise error bound.

The truncation error bound is given by:

$$\varepsilon_k^{\mathrm{t}}(\mathbf{u}) = \frac{\sigma_{\max}}{2} \left\| [(\mathbf{u}-\mathbf{u}_k)^T(\mathbf{u}-\mathbf{u}_k), \ldots, (\mathbf{u}-\mathbf{u}_{k-n_u+1})^T(\mathbf{u}-\mathbf{u}_{k-n_u+1})]\mathbf{U}_k^{-1}(\mathbf{u}) \right\|, \tag{2.3.15}$$

where $\sigma_{\max}$ is an upper bound on the spectral radius of the Hessian of the plant cost function. Roughly speaking, $\varepsilon_k^{\mathrm{t}}(\mathbf{u})$ increases along with the maximum of the distances between all pairs of points in the set $\{\mathbf{u}, \mathbf{u}_k, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-n_u+1}\}$. Hence, to keep the truncation error small, the past n_u+1 points must be sufficiently close to each other. The noise error bound is given by:

$$\varepsilon_k^{\mathrm{n}}(\mathbf{u}) = \frac{\delta}{l_{\min}(\mathbf{u})}, \tag{2.3.16}$$

where $l_{\min}(\mathbf{u})$ is the shortest distance between all possible pairs of complement affine subspaces that can be generated from $\mathbf{S} = [\mathbf{u}, \mathbf{u}_k, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-n_u+1}]$ (Marchetti et al., 2010), and δ is the maximum noise value that can occur. Thus, in this case, it is assumed that the noise affecting the cost measurement in Equation (2.3.2) is interval bounded.

[3] This is in analogy to the concept of 'dual control' in the field of adaptive control, whereby there is a dichotomy between more excitation for better identification (exploration) and less excitation for better control (exploitation).

Roughly speaking, $l_{\min}(\mathbf{u})$ is the minimum of the orthogonal distances between each individual point in the set $\{\mathbf{u}, \mathbf{u}_k, \mathbf{u}_{k-1}, \ldots, \mathbf{u}_{k-n_u+1}\}$ and the hyperplane passing through the remaining points. Hence, in order to keep the noise error small, the past $n_u - 1$ input moves should be approximately orthogonal to each other and no two points should be too close to each other. The additional (scalar) 'dual' constraint that is added to the modified model-based optimization problem is:

$$g_{\mathrm{d},k}(\mathbf{u}) = \varepsilon_k^{\mathrm{t}}(\mathbf{u}) + \varepsilon_k^{\mathrm{n}}(\mathbf{u}) - \varepsilon_{\max}, \tag{2.3.17}$$

which, as can be seen from Equation (2.3.14), should ensure that the maximum gradient error does not surpass the value $\varepsilon_{\max}$. Note that this constraint can be non-convex and, for that reason, Marchetti et al. (2010) also proposes a simple convex relaxation. Also, note that while only the error affecting the cost gradient estimate was discussed here, this constraint also ensures the gradient estimation error remains bounded.

A second formulation of the dual constraint, proposed by Rodger and Chachuat (2011), is specifically tailored to the case when the gradient is estimated using Equation (2.3.13). In this approach, two additional constraints are required. A first constraint aims to reduce the truncation error:

$$g_k^{\mathrm{t}}(\mathbf{u}) = 1 - (\mathbf{u} - \mathbf{u}_k)^T \mathbf{\Gamma}^T \mathbf{\Gamma} (\mathbf{u} - \mathbf{u}_k), \tag{2.3.18}$$

where $\mathbf{\Gamma}$ is typically a diagonal matrix. This simple constraint ensures that each new point is sufficiently distant from the preceding operating point. A second constraints aims to reduce the noise error:

$$g_k^{\mathrm{n}}(\mathbf{u}) = \min\left\{\left(\boldsymbol{\alpha}_k^T(\mathbf{u} - \mathbf{u}_k) + \sqrt{\boldsymbol{\alpha}_k^T \mathbf{\Sigma}^T \mathbf{\Sigma} \boldsymbol{\alpha}_k}\right), \left(-\boldsymbol{\alpha}_k^T(\mathbf{u} - \mathbf{u}_k) + \sqrt{\boldsymbol{\alpha}_k^T \mathbf{\Sigma}^T \mathbf{\Sigma} \boldsymbol{\alpha}_k}\right)\right\}, \tag{2.3.19}$$

where $\boldsymbol{\alpha}_k$ is any non-zero vector orthogonal to the previous $n_u - 1$ input moves, and $\mathbf{\Sigma}$ defines an ellipsoid around $\mathbf{u}_k$ outside which the next operating point must lie. Thus, reducing the noise error requires each input move to be both sufficiently large, and approximately orthogonal to the past $n_u - 1$ input moves. In this case, the dual constraint is a combination of the truncation-limiting constraint and the noise-limiting constraint:

$$\mathbf{g}_{\mathrm{d},k}(\mathbf{u}) = \begin{bmatrix} g_k^{\mathrm{t}}(\mathbf{u}) & g_k^{\mathrm{n}}(\mathbf{u}) \end{bmatrix}^T. \tag{2.3.20}$$

The feasible regions corresponding to both types of dual constraint for the $n_u = 2$ case are shown in Figure 2.1, for typical values of the tuning parameters in these constraints. Both constraints are similar: due to the trade off between truncation error and noise error, the next input move must neither be too large, nor too small. Also, the next input

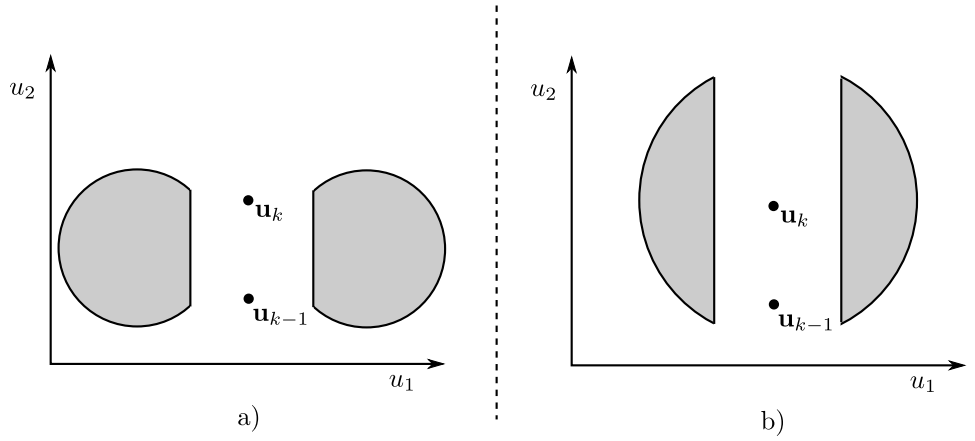

Figure 2.1: Feasible regions corresponding to the dual constraint: a) given by the convex relaxation of Equation (2.3.17), b) given by Equation (2.3.20). $\mathbf{u}_{k+1}$ is constrained to the shaded region.

move must be approximately orthogonal to the past $n_u - 1$ input moves. If past operating points are used for gradient computation, then there is no doubt these constraints are necessary to ensure the gradient estimate does not become very inaccurate. However, the dual constraints severely handicap the MA algorithm; by conflicting with the optimization objective, they negatively impact convergence towards the plant optimum. In order to optimize the process, the MA algorithm should move in the *one* direction that most improves performance without violating constraints. However, the dual constraints forbid moving in just one direction, instead requiring the exact opposite: that the MA algorithm should explore *all* directions of the input space. The larger n_u, the more directions must be explored in order to ensure accurate gradient estimates, and hence the more the dual constraints interfere with the optimization objective. Thus, the larger the number of input variables, the slower the plant optimum is reached.

2.4 Kite Dynamics

Dynamic models are the basic prerequisite for applying advanced control and optimization methods to kites. Flexible-kite modeling is still relatively immature compared to the modeling of airplanes. The modeling of *rigid* kites is actually much easier, as a mature aerodynamic theory already exists for rigid wings. The Airborne Wind Energy community is divided over the question of flexible vs. rigid wings. Flexible-wing kite designs for sports have been on the market for several decades. They are cheap, light, and very robust. However, they are not particularly efficient, although efforts are being made to improve their efficiency (Gohl and Luchsinger, 2013). Rigid wings, on the other hand, are expensive and delicate, but aerodynamically efficient. Given the focus on

flexible kites in the Swiss Kite Power group (SKP), flexible kites are the application in this thesis.

Flexible kites can be divided into two categories: ram-air kites, and tube kites. Although both types of kite behave in a qualitatively similar manner, more accurate modeling techniques are usually specific to one type or the other. Para-gliders and certain types of parachute are ram-air wings. The basic principle is that onrushing air enters an opening at the front of the wing, inflating it. The wing can be extremely light as it is the pressure of the onrushing air that provides structural rigidity, and the wing itself is made of light, flexible material. Although some of the modeling knowledge specific to para-gliders is now being applied to ram-air kites Dunker (2013), this has not been the traditional modeling approach in the Airborne-Wind-Energy field, as the addition of a fixed tether significantly modifies the dynamics of a ram-air wing. Tube kites use inflatable tube bladders to give the kite a semi-rigid structure. These kites were initially developed for kite surfing, where the kite often falls into the water and must float. The inflated bladders also act as structural support for the kite, which allows a greater variety of wing shapes to be considered. Modeling the fine details of how tube kites behave is extremely complex (Breukels, 2010; Breukels et al., 2013), as the wing shape will deform depending on the flight mode, leading to constantly changing aerodynamic properties.

Much progress has been made in modeling flexible-kite dynamics during the last decade (Diehl, 2001; Houska and Diehl, 2006; Dadd et al., 2010; Breukels, 2010; Erhard and Strauch, 2013b; Gohl and Luchsinger, 2013; Gros and Diehl, 2013; Breukels et al., 2013; Bosch et al., 2013; Paulig et al., 2013). Models can be constructed with anything from 3 to several-hundred states. However, the aerodynamics of a kite are a) very difficult to model precisely, and b) very dependent on the kite design. There are many complex and sometimes unpredictable factors affecting a flexible kite's flight: tether dynamics, moisture on the wing, wind gradients, wind gusting, stretching and deformation of the kite. For that reason even the most detailed models cannot claim to be perfect, and certainly for the purposes of controller design and optimization it is doubtful whether the use of very complex models is necessary, or even beneficial. The two simple models that are typically used for controller design and path optimization are the well-established point-mass model (Diehl, 2001; Fagiano et al., 2014) and the kinetic model recently proposed by Erhard and Strauch (2013a) (henceforth referred to as the 'Erhard Model'). Both models are general enough to apply to both ram-air kites and tube kites. If the correct parameters are used, both models will predict qualitatively similar behavior. However, the kinetic model is simpler and more intuitive. As opposed to the point-mass model, which is more geometric in nature, the kinetic model has only two aerodynamic parameters that can be calculated for a real system using straightforward experiments.

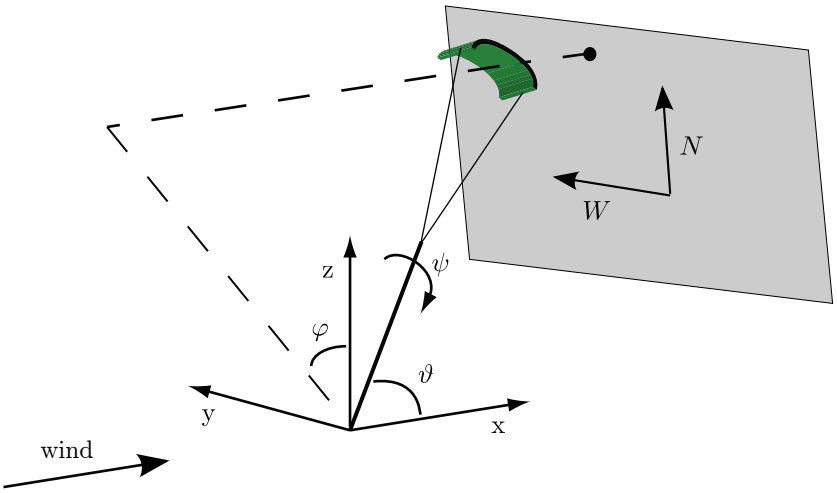

Figure 2.2: Spherical coordinate system for the kite position. The x and y axes are horizontal, while the z-axis points skywards. The kite is tethered to the origin. The wind is aligned with the x axis.

In addition, the experimentally validated Erhard Model has been successfully used in an industrial setting to design control algorithms for very large kites.

This section describes the Erhard Model; the equations are taken from Erhard and Strauch (2013a). While the system to be optimized may be a kite pulling a ship, or an electricity-producing 'pumping' kite, the basic model is for a kite attached to a fixed point by a fixed-length tether, as shown in Figure 2.2. This basic model can then be adapted to include the effects of a moving tether point (for a kite pulling a ship), or a varying tether length (for a 'pumping' kite).

The kite's fixed, inertial, right-hand coordinate system is depicted in Figure 2.2. The kite's position in Cartesian coordinates is given by:

$$\mathbf{p} = r \begin{bmatrix} \cos\vartheta \\ \sin\vartheta\sin\varphi \\ \sin\vartheta\cos\varphi \end{bmatrix}, \tag{2.4.1}$$

where r is the (constant) length of the kite's tether, and ϑ and φ are spherical coordinates for the kite's position, using the x-axis as the zenith. In this thesis, the kite's position is often represented as a projection onto the $\{N, W\}$ plane shown in Figure 2.2. The plane is defined by the two orthogonal vectors $\hat{\mathbf{e}}_W = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$ and $\hat{\mathbf{e}}_N = \begin{bmatrix} -\sin\bar{\vartheta} & 0 & \cos\bar{\vartheta} \end{bmatrix}^T$, which are tangent to the sphere upon which the kite can move at the point $\{\vartheta, \varphi\} = \{\bar{\vartheta}, 0\}$.

The dynamic equations for the model are:

$$\dot{\vartheta} = \frac{w_{\text{ap}}}{r}\left(\cos\psi - \frac{\tan\vartheta}{E}\right), \tag{2.4.2}$$

$$\dot{\varphi} = -\frac{w_{\text{ap}}}{r\sin\vartheta}\sin\psi, \tag{2.4.3}$$

$$\dot{\psi} = w_{\text{ap}} g_{\text{s}}\delta + \dot{\varphi}\cos\vartheta, \tag{2.4.4}$$

where ψ is the kite orientation, g_{s} is the turning constant, and E is the kite's lift-to-drag ratio. The steering deflection, δ, is the system's manipulated variable. w_{ap} is the magnitude of the apparent wind projected onto the plane that is normal to $\mathbf{p}$, and is given by:

$$w_{\text{ap}} = wE\cos\vartheta, \tag{2.4.5}$$

where w is the wind speed at the kite's current altitude. A number of different wind-shear models exist, which describe the variation of wind speed with altitude. One of the most common is the power law (Archer, 2013):

$$w = w_{\text{ref}}(z/z_{\text{ref}})^{a}, \tag{2.4.6}$$

where a is the surface friction coefficient, w_{ref} is the reference wind speed at the reference altitude z_{ref}, and z is the kite altitude. Finally, the line tension is given by

$$T = \left(\frac{1}{2}\rho A w^2\right)(E+1)\sqrt{E^2+1}\cos^2\vartheta. \tag{2.4.7}$$

3 Generalized Modifier Adaptation

In the previous chapter we saw that Modifier Adaptation (MA) uses measurements to implement affine corrections to the cost and constraint functions in the *model-based* optimization problem, while the uncertain parameters in the model are kept fixed. MA has been designed to resolve plant-model mismatch, yet the model must still satisfy two conditions:

1. have the same inputs as the plant,
2. predict a locally convex cost function at the plant optimum (Theorem 2.3.2).

Condition (2) tends to be satisfied by any reasonable model and its enforcement is discussed by François and Bonvin (2013a). On the other hand, Condition (1) may not necessarily be satisfied, and two examples of systems where this is the case are given in Section 3.1. Section 3.2 presents a 'Generalized MA' theory to deal with this issue. Finally, the algorithm is illustrated on a simulation case study in Section 3.3.

3.1 Motivating Examples

3.1.1 Incineration Plant

I was involved in developing a process model that did not satisfy Condition 1. The plant is the steam cycle of Lausanne's 80-MW incineration plant (Tridel), a combined heat-and-power regenerative Rankine cycle. Energy released by incinerating refuse is used to heat water to 400°C at 50 bar, which drives a turbine to generate electricity. Steam is bled from the turbine at two intermediate stages and passed through heat exchangers that heat water for district heating. A simplified diagram of the system is show in Figure 3.1. The optimization objective is to adjust the pressures, temperatures and mass flow

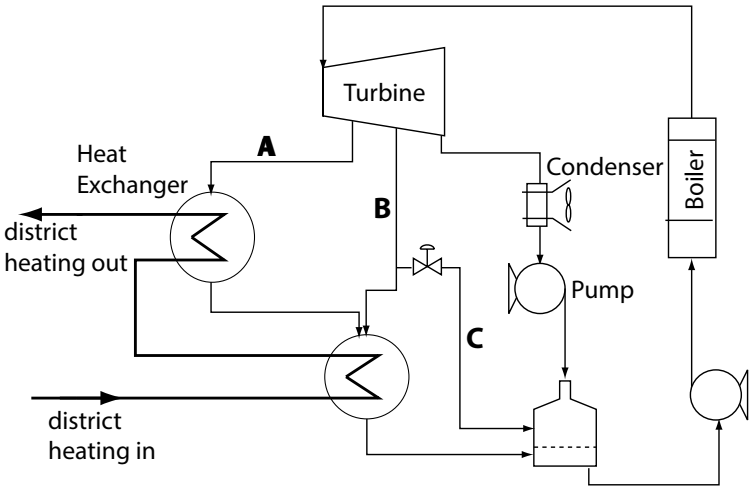

Figure 3.1: The steam cycle of the 80-MW incineration plant.

rates of the two intermediate bleeds from the turbine in order to maximize the electrical efficiency for a given district heating demand.

The available system model has the following five inputs: the temperature and mass flowrate at point A, T_A and w_A; the temperature and mass flowrate at point B, T_B and w_B; and the mass flowrate at point C, w_C. All the unknown variables in the steam cycle can be calculated if these five variables are specified first. These five variables were chosen as the model inputs, not necessarily because they correspond to the operator's inputs, but because they help solve the system equations for this complex cycle. In fact, it was later established that, from the operator's point of view, the plant has only two real inputs, the pressure at point A, p_A, and the pressure at point B, p_B. These are the only variables the operator can adjust in order to optimize the plant's performance. The block diagrams for the model and the plant are shown in Figure 3.2. The model has more inputs than the plant because certain relationships between variables are not modeled: 1) reliable equations for modeling the steam turbine are not available, and 2) it is not known how the control loop that adjusts w_C is implemented. As a result, the model is missing three equations, which results in three additional inputs. Furthermore, note that the true plant inputs are not among the model inputs. Although the model is useful for off-line numerical optimization and computation of $\mathbf{u} = [T_\mathrm{A}, w_\mathrm{A}, T_\mathrm{B}, w_\mathrm{B}, w_\mathrm{C}]$, it cannot be used for standard MA to compute the plant inputs $\mathbf{c} = [p_\mathrm{A}, p_\mathrm{B}]$ because Condition 1 is not satisfied.

Improving the model such that it encompasses the same set of inputs as the plant would require detailed models of the turbine and the controller for w_C, which, unfortunately, are not available. The manner in which the model equations are solved would also need

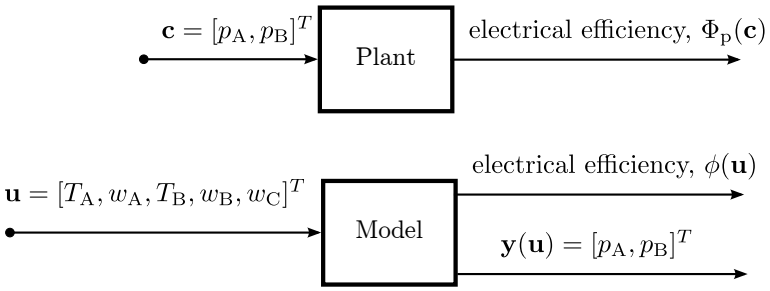

Figure 3.2: Model and plant inputs for the incineration plant.

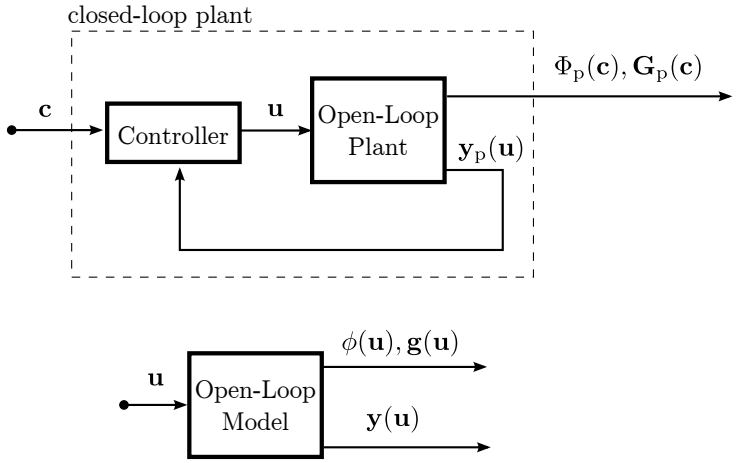

Figure 3.3: Closed-loop plant to be optimized and, for comparison, the open-loop model that is available.

to be changed. Hence, it is difficult to reformulate the model such that its inputs $\mathbf{u}$ are the same as those of the plant, $\mathbf{c}$. However, as we will see in this chapter, re-modeling is not necessary, and MA can be generalized such that the model can be used in its current form. This is particularly convenient because measurements (which are in abundance for this system) can be used to compensate for the missing modeling information.

3.1.2 Controlled Plant

Closed-loop systems, where only the open-loop plant has been modeled, will not satisfy Condition 1. As an example, consider the controlled plant shown in Figure 3.3. A model will allow the computation of the optimal *manipulated variables* $\mathbf{u}^*$. However, since the plant is operated in closed loop, there is no direct way of manipulating $\mathbf{u}$ to enforce optimality. Although the model can be used to predict the optimal values of the controlled variables $\mathbf{y}(\mathbf{u}^*)$, choosing $\mathbf{c} = \mathbf{y}(\mathbf{u}^*)$ as the *setpoint* for the closed-loop plant will not result in optimal operation. This is because a) $\mathbf{u}^*$ is not the plant optimum due to plant-model mismatch, and b) in any case $\mathbf{u}^*$ will not be applied to the open-loop

plant, as, even if the controller is perfect, $\mathbf{y}_\mathrm{p}(\mathbf{u}^*) \neq \mathbf{y}(\mathbf{u}^*)$.

In standard MA, the open-loop plant inputs are perturbed to estimate the gradients of the plant cost and constraints. This eventually leads to obtaining the optimal open-loop plant inputs $\mathbf{u}_\mathrm{p}^*$. For closed-loop systems, we are interested in determining the optimal set-points $\mathbf{c}_\mathrm{p}^*$, since optimality of the *closed-loop* system is sought. Furthermore, the plant gradients can be estimated with respect to these setpoints (and not $\mathbf{u}$). Fortunately, the fact that the model can predict the controlled variables, and thus also the setpoints required to achieve a certain performance, provides the link to the closed-loop plant.

The only way to apply standard MA is to re-model the closed-loop plant such that the setpoints become the inputs. This may be achieved by modeling the steady-state behavior of the controller with a law of the form:

$$\mathbf{u} = \mathbf{F_c}\left(\mathbf{y}(\mathbf{u}), \mathbf{c}\right). \tag{3.1.1}$$

For a given $\mathbf{c}$, these n_u equations can be solved for $\mathbf{u}$, allowing $\phi(\mathbf{u})$ and $\mathbf{g}(\mathbf{u})$ to be calculated. Alternatively, an ideal controller can be assumed, which ensures:

$$\mathbf{y}(\mathbf{u}) - \mathbf{c} = \mathbf{0}. \tag{3.1.2}$$

These n_c equations can be solved for $\mathbf{u}$ if $n_u = n_c$, where n_c is the number of controlled variables. Even if one of the above approaches can be applied (which is not always the case), it is likely to result in a closed-loop model that is difficult to evaluate in real time, as it will involve solving a system of n_u equations. This is likely to result in increased computation time per model evaluation, which can be problematic for on-line optimization.

The “Generalized MA" framework presented in the next section avoids this onerous re-modeling, while nonetheless guaranteeing that the optimal plant setpoints are reached upon convergence.

3.2 Generalized Modifier Adaptation

We show next how standard MA can be altered to optimize a plant which does not have the same inputs as the available model. The aim is to avoid having to re-model the system. We consider the case where the plant cost function $\Phi_\mathrm{p}(\mathbf{c})$ and constraint functions $\mathbf{G}_\mathrm{p}(\mathbf{c})$ are expressed in terms of the n_c plant inputs. Generalized modifier adaptation can be applied under the following circumstances:

1. The model cost function $\phi(\mathbf{u})$ and constraint functions $\mathbf{g}(\mathbf{u})$ have n_u inputs $\mathbf{u}$, with $n_u \geq n_c$.

2. A model, $\mathbf{y}(\mathbf{u})$, which approximately models the mapping from $\mathbf{u}$ to $\mathbf{c}$ is available.

Two algorithms are introduced, each one with a different way of computing the gradient modifiers from the estimates of the plant gradients, $\frac{\partial \Phi_p}{\partial \mathbf{c}}$ and $\frac{\partial \mathbf{G}_p}{\partial \mathbf{c}}$, and the gradients computed from the open-loop model, $\frac{\partial \phi}{\partial \mathbf{u}}$ and $\frac{\partial \mathbf{g}}{\partial \mathbf{u}}$. Since these gradients are computed with respect to different variables, they cannot be compared directly. The first algorithm, Generalized Modifier Adaptation (G-MA), computes the modifiers in the space of the plant inputs $\mathbf{c}$. For this, the model gradients $\frac{\partial \phi}{\partial \mathbf{c}}$ and $\frac{\partial \mathbf{g}}{\partial \mathbf{c}}$ are computed by inverting the relationship $\frac{\partial \mathbf{y}}{\partial \mathbf{u}}$. This is the most obvious manner in which to adapt Standard MA. It is shown that if G-MA converges, it will do so to a KKT point for the plant. G-MA requires a nonlinear term to be inserted into the optimization problem to be solved on-line. This may introduce local minima, or make the problem more difficult to solve numerically. 'Linearized' G-MA (LG-MA) addresses this issue, by using a linearization of the problematic nonlinear term. The resulting algorithm computes the modifiers in the space of the model inputs $\mathbf{u}$ by expressing the plant gradients $\frac{\partial \Phi_p}{\partial \mathbf{c}}$ and $\frac{\partial \mathbf{G}_p}{\partial \mathbf{c}}$ in terms of $\mathbf{u}$. It is demonstrated that LG-MA also guarantees that convergence can occur to only a KKT point for the plant.

3.2.1 Basic Generalized Modifier Adaptation (G-MA)

The model gradients are, logically, calculated with respect to the model inputs $\mathbf{u}$. However, the plant gradients can only be evaluated with respect to the plant inputs $\mathbf{c}$. The method now described expresses the model gradients with respect to the *plant* inputs, by inverting the modeled relationship between $\mathbf{u}$ and $\mathbf{c}$, $\mathbf{y}(\mathbf{u})$. This allows the model and plant gradients to be compared to yield modifier terms.

Algorithm 3.2.1: Generalized MA

Initialize the modifier terms: the n_g-dimensional vector of zeroth-order modifiers $\boldsymbol{\epsilon}_0 = \mathbf{0}$, the n_c-dimensional vector of first-order cost modifiers $\boldsymbol{\lambda}_0^{\phi} = \mathbf{0}$, and the $(n_c \times n_g)$ matrix of first-order gradient modifiers $\boldsymbol{\lambda}_0^{g} = \mathbf{0}$. Also, choose arbitrarily $\mathbf{c}_0 = \mathbf{0}$.

for $k = 1 \rightarrow \infty$

1. Solve the modified model-based optimization problem (P1):

$$\begin{aligned} \mathbf{u}_k := \arg\min_{\mathbf{u}} \quad & \phi_{\mathrm{m},k-1}(\mathbf{u}), \\ \text{subject to} \quad & \mathbf{g}_{\mathrm{m},k-1}(\mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{3.2.1}$$

   where the modified cost and constraints are given by

$$\phi_{\mathrm{m},k}(\mathbf{u}) := \phi(\mathbf{u}) + \left(\boldsymbol{\lambda}_k^{\phi}\right)^T (\mathbf{y}(\mathbf{u}) - \mathbf{c}_k), \tag{3.2.2}$$

$$\mathbf{g}_{\mathrm{m},k}(\mathbf{u}) := \mathbf{g}(\mathbf{u}) + \boldsymbol{\epsilon}_k + \left(\boldsymbol{\lambda}_k^{g}\right)^T (\mathbf{y}(\mathbf{u}) - \mathbf{c}_k). \tag{3.2.3}$$

2. Update the plant inputs $\mathbf{c}_k := \mathbf{y}(\mathbf{u}_k)$, to obtain $\tilde{\Phi}_{\mathrm{p}}(\mathbf{c}_k)$ and $\tilde{\mathbf{G}}_{\mathrm{p}}(\mathbf{c}_k)$.

3. Estimate the plant cost gradient, $\nabla_{\mathbf{c}}\Phi_{\mathrm{E},k}$, and the plant constraint gradient, $\nabla_{\mathbf{c}}\mathbf{G}_{\mathrm{E},k}$, at the current operating point $\mathbf{c}_k$. Note that these gradients are with respect to the *plant inputs* $\mathbf{c}$.

4. Calculate the modifiers for the next iteration:

$$\boldsymbol{\epsilon}_k := \tilde{\mathbf{G}}_{\mathrm{p}}(\mathbf{c}_k) - \mathbf{g}(\mathbf{u}_k), \tag{3.2.4}$$

$$(\boldsymbol{\lambda}_k^{\phi})^T := \nabla_{\mathbf{c}}\Phi_{\mathrm{E},k} - \frac{\partial \phi}{\partial \mathbf{u}}(\mathbf{u}_k)\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^{+}, \tag{3.2.5}$$

$$(\boldsymbol{\lambda}_k^{g})^T := \nabla_{\mathbf{c}}\mathbf{G}_{\mathrm{E},k} - \frac{\partial \mathbf{g}}{\partial \mathbf{u}}(\mathbf{u}_k)\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^{+}, \tag{3.2.6}$$

   with $(\cdot)^+$ indicating the Moore-Penrose pseudo inverse. To encourage convergence, these modifiers may also be filtered using a low-pass filter before being incorporated into problem (P1). This is further discussed in Section 3.2.3.

end

We claim that, at least under ideal circumstances, all fixed points of this iterative procedure are KKT points for the plant.

Theorem 3.2.1. *[Plant optimality for G-MA]*
If G-MA converges, it will do so to a KKT point for the plant, provided there is no high-frequency noise and the gradient estimates are perfect.

Proof: Consider the iterative scheme upon convergence, i.e. $\lim_{k\to\infty} \mathbf{u}_k = \mathbf{u}_\infty$. We will first derive relationships between $\phi_{\mathrm{m},k}$ and $\mathbf{g}_{\mathrm{m},k}$ and the plant cost and constraints

Φ_p and $\mathbf{G}_p$.[1] Upon convergence, due to assumption of perfect gradient estimates, $\nabla_{\mathbf{c}}\Phi_{E,\infty} = \frac{\partial \Phi_p}{\partial \mathbf{c}}$, and thus:

$$(\boldsymbol{\lambda}_{\infty}^{\phi})^T = \frac{\partial \Phi_p}{\partial \mathbf{c}} - \frac{\partial \phi}{\partial \mathbf{u}} \left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ . \tag{3.2.7}$$

The gradient of the cost function $\phi_{m,\infty}$ in Problem (P1) is

$$\frac{\partial \phi_{m,\infty}}{\partial \mathbf{u}} = \frac{\partial \phi}{\partial \mathbf{u}} + (\boldsymbol{\lambda}_{\infty}^{\Phi})^T \frac{\partial \mathbf{y}}{\partial \mathbf{u}}, \tag{3.2.8}$$

which, using (3.2.7), gives:

$$\frac{\partial \phi_{m,\infty}}{\partial \mathbf{u}} = \frac{\partial \phi}{\partial \mathbf{u}} + \left(\frac{\partial \Phi_p}{\partial \mathbf{c}} - \frac{\partial \phi}{\partial \mathbf{u}} \left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+\right) \frac{\partial \mathbf{y}}{\partial \mathbf{u}}. \tag{3.2.9}$$

Multiplying both sides of this equation by $\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}}$ and using the identity $\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}} = \left(\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^2$ yields:

$$\frac{\partial \tilde{\phi}_{m,\infty}}{\partial \mathbf{u}} \left(\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right) = \frac{\partial \Phi_p}{\partial \mathbf{c}} \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}} \tag{3.2.10}$$

$$= \frac{\partial \Phi_p}{\partial \mathbf{c}} \frac{\partial \mathbf{y}}{\partial \mathbf{u}}. \tag{3.2.11}$$

The same argument can be used to show that

$$\frac{\partial \mathbf{g}_{m,\infty}}{\partial \mathbf{u}} \left(\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right) = \frac{\partial \mathbf{G}_p}{\partial \mathbf{c}} \frac{\partial \mathbf{y}}{\partial \mathbf{u}}. \tag{3.2.12}$$

From the definition of $\boldsymbol{\epsilon}_k$ (Equation (3.2.4)), and assuming $\tilde{\mathbf{G}}_p(\mathbf{c}_\infty) = \mathbf{G}_p(\mathbf{c}_\infty)$, one can write:

$$\begin{aligned} \tilde{\mathbf{g}}_{m,\infty}(\mathbf{u}_\infty) &= \mathbf{g}(\mathbf{u}_\infty) + \mathbf{G}_p(\mathbf{c}_\infty) - \mathbf{g}(\mathbf{u}_\infty) \\ &= \mathbf{G}_p(\mathbf{c}_\infty). \end{aligned} \tag{3.2.13}$$

Now, since by definition $\mathbf{u}_\infty$ is a KKT point for Problem (P1), $\exists\, \boldsymbol{\nu} \geq \mathbf{0}$ such that

$$\frac{\partial \phi_{m,\infty}}{\partial \mathbf{u}} + \boldsymbol{\nu}^T \frac{\partial \mathbf{g}_{m,\infty}}{\partial \mathbf{u}} = \mathbf{0}. \tag{3.2.14}$$

[1] The function arguments will be dropped in the following derivation as all functions are evaluated at the stationary point corresponding to $\mathbf{u}_\infty$ and $\mathbf{c}_\infty = \mathbf{y}(\mathbf{u}_\infty)$.

It follows from equations (3.2.11) and (3.2.12) and assuming $rank\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right) = n_c$ that

$$\frac{\partial \Phi_{\mathrm{p}}}{\partial \mathbf{c}} + \boldsymbol{\nu}^T \frac{\partial \mathbf{G}_{\mathrm{p}}}{\partial \mathbf{c}} = \mathbf{0}. \tag{3.2.15}$$

Hence the dual feasibility and the stationarity KKT conditions are satisfied for the plant. The KKT conditions for Problem (P1) also state that $\boldsymbol{\nu}^T \mathbf{g}_{\mathrm{m},\infty} = 0$. As we have shown that $\mathbf{g}_{\mathrm{m},\infty} = \mathbf{G}_{\mathrm{p}}$, it follows that

$$\boldsymbol{\nu}^T \mathbf{G}_{\mathrm{p}} = 0. \tag{3.2.16}$$

Thus, both the primal feasibility and the complementary slackness KKT conditions are satisfied for the plant. As all four KKT conditions are satisfied, $\mathbf{c}_\infty$ is a KKT point for the plant. ■

3.2.2 Linearized Generalized Modifier Adaptation

G-MA is the most immediately obvious way to adapt standard MA to deal with the plant and the model having different inputs. However, Problem (P1) contains additional nonlinear terms due to the introduction of $\mathbf{y}(\mathbf{u})$ into the modified cost and constraint functions (see Equations (3.2.2) and (3.2.3)). The addition of these nonlinear terms can make Problem (P1) more difficult to solve, or, even worse, introduce local minima. For that reason, a second method, LG-MA, is developed, which only introduces affine terms into the modified optimization problem.

A Taylor-series expansion of the second term in the modified cost function in Problem (P1) with respect to $\mathbf{u}$, around $\mathbf{u}_k$, gives:

$$\begin{aligned} \phi_{\mathrm{m},k}(\mathbf{u}) &= \phi(\mathbf{u},\boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_k^{\phi})^T(\mathbf{y}(\mathbf{u}) - \mathbf{c}_k) && (3.2.17)\\ &= \phi(\mathbf{u},\boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_k^{\phi})^T\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)(\mathbf{u}-\mathbf{u}_k) + O\left((\mathbf{u}-\mathbf{u}_k)^2\right). && (3.2.18)\end{aligned}$$

A new gradient modifier, $\boldsymbol{\lambda}^{\mathrm{L},\phi}$, is defined as:

$$\begin{aligned} \left(\boldsymbol{\lambda}^{\mathrm{L},\phi}\right)^T &:= (\boldsymbol{\lambda}_k^{\phi})^T\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)\\ &= \nabla_{\mathbf{c}}\Phi_{\mathrm{E},k}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial \phi}{\partial \mathbf{u}}(\mathbf{u}_k)\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^+\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k). && (3.2.19)\end{aligned}$$

Using this new modifier, a new modified cost function can be defined:

$$\phi^{\mathrm{L}}_{\mathrm{m},k}(\mathbf{u}) := \phi(\mathbf{u},\boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_k^{\mathrm{L},\phi})^T(\mathbf{u}-\mathbf{u}_k) \tag{3.2.20}$$

This is, by definition, a linear approximation of the modified cost function in G-MA:

$$\phi^{\mathrm{L}}_{\mathrm{m},k}(\mathbf{u}) = \phi_{\mathrm{m},k}(\mathbf{u}) + O\left((\mathbf{u}-\mathbf{u}_k)^2\right). \tag{3.2.21}$$

The same development may be carried out for the constraint function, yielding the following algorithm:

Algorithm 3.2.2: Linearized Generalized Modifier Adaptation (LG-MA)

Initialize the modifier terms: the n_g-dimensional vector of zeroth-order modifiers $\boldsymbol{\epsilon}_0 = \mathbf{0}$, the n_u-dimensional vector of first-order cost modifiers $\boldsymbol{\lambda}_0^{\mathrm{L},\phi} = \mathbf{0}$, and the $(n_u \times n_g)$ matrix of first-order gradient modifiers $\boldsymbol{\lambda}_0^{\mathrm{L},G} = \mathbf{0}$. Also, choose arbitrarily $\mathbf{u}_0 = \mathbf{0}$.

for $k = 1 \rightarrow \infty$

1. Solve the modified model-based optimization problem (P2):

$$\begin{aligned} \mathbf{u}_k := \arg\min_{\mathbf{u}} \quad & \phi^{\mathrm{L}}_{\mathrm{m},k-1}(\mathbf{u}), \\ \text{subject to} \quad & \mathbf{g}^{\mathrm{L}}_{\mathrm{m},k-1}(\mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{3.2.22}$$

   where the modified cost and constraints are given by

$$\phi^{\mathrm{L}}_{\mathrm{m},k}(\mathbf{u}) := \phi(\mathbf{u},\boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_k^{\mathrm{L},\phi})^T(\mathbf{u}-\mathbf{u}_k), \tag{3.2.23}$$

$$\mathbf{g}^{\mathrm{L}}_{\mathrm{m},k}(\mathbf{u}) := \mathbf{g}(\mathbf{u},\boldsymbol{\theta}_0) + \boldsymbol{\epsilon}_k + (\boldsymbol{\lambda}_k^{\mathrm{L,g}})^T(\mathbf{u}-\mathbf{u}_k). \tag{3.2.24}$$

2. Update the plant inputs $\mathbf{c}_k := \mathbf{y}(\mathbf{u}_k)$, to obtain $\tilde{\Phi}_{\mathrm{p}}(\mathbf{c}_k)$ and $\tilde{\mathbf{G}}_{\mathrm{p}}(\mathbf{c}_k)$.

3. Estimate the plant cost gradient, $\nabla_{\mathbf{c}}\Phi_{\mathrm{E},k}$, and the plant constraint gradient, $\nabla_{\mathbf{c}}\mathbf{G}_{\mathrm{E},k}$.

4. Calculate the modifiers for the next iteration:

$$\boldsymbol{\epsilon}_k := \tilde{\mathbf{G}}_p(\mathbf{c}_k) - \mathbf{g}(\mathbf{u}_k), \tag{3.2.25}$$

$$\left(\boldsymbol{\lambda}_k^{\mathrm{L},\phi}\right)^T := \nabla_{\mathbf{c}}\Phi_{\mathrm{E},k}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial \phi}{\partial \mathbf{u}}(\mathbf{u}_k)\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^+\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k), \tag{3.2.26}$$

$$\left(\boldsymbol{\lambda}_k^{\mathrm{L},g}\right)^T := \nabla_{\mathbf{c}}\mathbf{G}_{\mathrm{E},k}\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k) - \frac{\partial \mathbf{g}}{\partial \mathbf{u}}(\mathbf{u}_k)\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^+\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k). \tag{3.2.27}$$

end

One interpretation of LG-MA is that gradients of the model cost and constraints are

'corrected' only in those directions that locally influence $\frac{\partial \mathbf{y}}{\partial \mathbf{u}}$. To this end, the post multiplication by $\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)\right)^+ \frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)$ removes any components of $\frac{\partial \phi}{\partial \mathbf{u}}(\mathbf{u}_k)$ and $\frac{\partial \mathbf{g}}{\partial \mathbf{u}}(\mathbf{u}_k)$ in the null space of $\frac{\partial \mathbf{y}}{\partial \mathbf{u}}(\mathbf{u}_k)$.

Despite the linearization used in passing from G-MA to LG-MA, it can be shown that the attractive property of converging only to a plant KKT point is conserved.

Theorem 3.2.2. *[Plant optimality for LG-MA]*
If LG-MA converges, it will do so to a KKT point for the plant, provided there is no high-frequency noise and the gradient estimates are perfect.

Proof: Based on the definition of $(\boldsymbol{\lambda}^{\mathrm{L},\phi})^T$, it follows upon convergence that

$$\frac{\partial \phi^{\mathrm{L}}_{\mathrm{m},\infty}}{\partial \mathbf{u}} = \frac{\partial \phi}{\partial \mathbf{u}} + \left(\frac{\partial \Phi_{\mathrm{p}}}{\partial \mathbf{c}} - \frac{\partial \phi}{\partial \mathbf{u}}\left(\frac{\partial \mathbf{y}}{\partial \mathbf{u}}\right)^+\right)\frac{\partial \mathbf{y}}{\partial \mathbf{u}} = \frac{\partial \phi_{\mathrm{m},\infty}}{\partial \mathbf{u}}, \tag{3.2.28}$$

and by the same logic $\frac{\partial \mathbf{g}^{\mathrm{L}}_{\mathrm{m},\infty}}{\partial \mathbf{u}} = \frac{\partial \mathbf{g}_{\mathrm{m},\infty}}{\partial \mathbf{u}}$. Thus the gradient of the modified cost and constraints is identical for both Methods A and B. This means that the optimality proof for G-MA also applies to LG-MA, from Equation (3.2.9) onwards. ■

3.2.3 Filtering the Modifier Terms

The filtering of the modifiers has not been discussed so far in this chapter. In practice, this is an essential add-on to the algorithms presented here, as it tends to improve the chances of convergence. For example, in the case of G-MA (the procedure is identical for LG-MA) , the filtered modifiers $\bar{\boldsymbol{\lambda}}^{\phi}_k, \bar{\boldsymbol{\lambda}}^{g}_k$ and $\bar{\boldsymbol{\epsilon}}_k$ are obtained by adding an additional step after Step 4:

5. Filter the modifiers:

$$\bar{\boldsymbol{\epsilon}}_{k+1} = (\mathbf{I}_{n_g} - \mathbf{K}^{\epsilon})\bar{\boldsymbol{\epsilon}}_{k-1} + \mathbf{K}^{\epsilon}\,\boldsymbol{\epsilon}_k. \tag{3.2.29}$$

$$\bar{\boldsymbol{\lambda}}^{\phi}_{k+1} = (\mathbf{I}_{n_c} - \mathbf{K}^{\phi})\bar{\boldsymbol{\lambda}}^{\phi}_{k-1} + \mathbf{K}^{\phi}\,\boldsymbol{\lambda}^{\phi}_k, \tag{3.2.30}$$

$$\bar{\boldsymbol{\lambda}}^{g}_{k+1} = (\mathbf{I}_{n_c} - \mathbf{K}^{g})\bar{\boldsymbol{\lambda}}^{g}_{k-1} + \mathbf{K}^{g}\,\boldsymbol{\lambda}^{g}_k, \tag{3.2.31}$$

where $\mathbf{K}^{\epsilon}, \mathbf{K}^{\phi}$ and $\mathbf{K}^{g}$ are the filter matrices.

The filtered modifiers are then used to compute the modified cost and constraint functions in Step 1 of the next iteration:

$$\phi_{\mathrm{m},k}(\mathbf{u}) := \phi(\mathbf{u}) + (\bar{\boldsymbol{\lambda}}_k^{\phi})^T(\mathbf{y}(\mathbf{u}) - \mathbf{c}_k), \tag{3.2.32}$$

$$\mathbf{g}_{\mathrm{m},k}(\mathbf{u}) := \mathbf{g}(\mathbf{u}) + \bar{\boldsymbol{\epsilon}}_k + (\bar{\boldsymbol{\lambda}}_k^{g})^T(\mathbf{y}(\mathbf{u}) - \mathbf{c}_k). \tag{3.2.33}$$

Just as for standard MA, the filter matrices should be chosen with real positive eigenvalues in the interval $(0, 1]$.

3.3 Simulated Example: Williams-Otto Reactor

The method is illustrated on the Williams-Otto Reactor (Williams and Otto, 1960). We will use the model from Roberts (1979), which has become a standard test problem for real-time optimization techniques (Marchetti et al., 2010). Although the original problem is an open-loop reactor, here the aim is to optimize the reactor in closed loop. The open-loop plant is an ideal continuous stirred-tank reactor with the following reactions:

$$A + B \xrightarrow{k_1} C, \qquad k_1 = k_{10}e^{-E_1/(RT)}, \tag{3.3.1}$$

$$C + B \xrightarrow{k_2} P + E, \quad k_2 = k_{20}e^{-E_2/(RT)}, \tag{3.3.2}$$

$$C + P \xrightarrow{k_3} G, \qquad k_3 = k_{30}e^{-E_3/(RT)}. \tag{3.3.3}$$

The (open-loop) plant inputs are chosen as $\mathbf{u} = [F_A, F_B, T]^T$, that is, the feed rates of A and B, and the reactor temperature. However, the degrees of freedom (inputs) of the controlled plant are the controller set-points $\mathbf{c} = [X_{A,\mathrm{s}}, F_{B,\mathrm{s}}]^T$ for the mass fraction of A in the reactor and the feed rate of B. The desired products are P and E and the reactor mass holdup is 2105 kg.

A rather poor controller adjusts F_A, F_B and T in the following manner:

- $F_B = F_{B,\mathrm{s}} + 2$, that is, there is an offset between F_B and $F_{B,\mathrm{s}}$.
- $F_A = \frac{F_B}{2.4}$, that is, F_A is proportional to F_B.
- T is manipulated so as to meet the setpoint $X_{A,\mathrm{s}}$, however there is a large steady-state offset:

$$X_A = 1.5X_{A,\mathrm{s}}. \tag{3.3.4}$$

The block diagram of the controlled CSTR is shown in Figure 3.4.

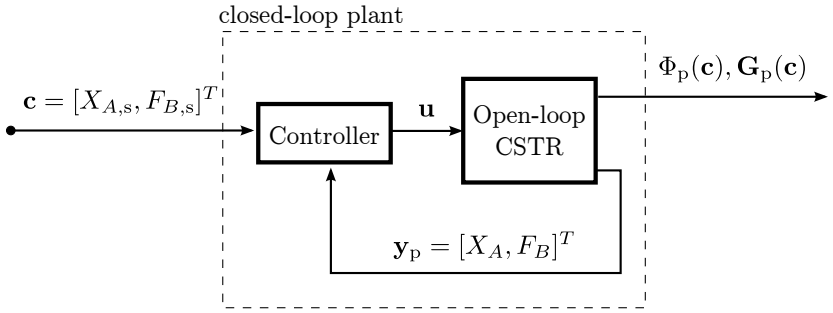

Figure 3.4: The controlled CSTR.

The available model is a two-reaction approximation of the open-loop plant:

$$A + 2B \xrightarrow{k_1^*} P + E, \quad k_1^* = k_{10}^* e^{-E_1^*/(RT)}, \tag{3.3.5}$$

$$A + B + P \xrightarrow{k_2^*} G, \quad k_2^* = k_{20}^* e^{-E_2^*/(RT)}, \tag{3.3.6}$$

with the parameters k_{10}^*, k_{20}^*, E_1^* and E_2^*. Three different nominal models will be considered, depending on the values taken by the parameters E_1^* and E_2^*, the parameters k_{10}^* and k_{20}^* being fixed. The material balance equations for both the plant and the model are given in Appendix A. From the implementation point of view, the controller is considered to be unknown. In particular, no knowledge is available regarding the manner in which F_A is manipulated.

The profit function to be maximized is

$$\begin{aligned}\text{Profit} =& 1143.38 X_P (F_A + F_B) + 25.92 X_E (F_A + F_B) \\ & - 76.23 F_A - 114.34 F_B,\end{aligned} \tag{3.3.7}$$

where X_P and X_E are the mass fractions of the products P and E. There are two operational constraints:

$$X_A \leq 0.09, \tag{3.3.8}$$

$$X_G \leq 0.6\,. \tag{3.3.9}$$

The cost and constraint functions $\Phi_\mathrm{p}(\mathbf{c}_\mathrm{s})$, $\mathbf{G}_\mathrm{p}(\mathbf{c}_\mathrm{s})$, $\phi(\mathbf{u})$ and $\mathbf{g}(\mathbf{u})$ are constructed by combining the above profit and constraint functions with the plant and model equations, respectively. Table 3.1 gives the numerical values of the plant parameters and the fixed model parameters. The input-output representation of the open-loop model is shown in Figure 3.5. The model can be used to *approximately* compute a) the values of the controlled variables $\mathbf{c}$, and thus the setpoints for the controlled reactor that would lead to particular inputs $\mathbf{u}$, and b) the resulting cost and constraint values.

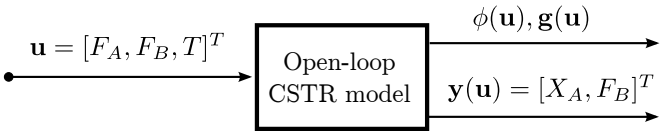

Figure 3.5: Open-loop model of the CSTR.

Table 3.1: Values of the plant parameters and the two fixed model parameters. The other model parameters are variable, as shown in Table 3.2, to generate the investigation cases I-III.

parameter	unit	value
k_{10}	s^{-1}	1.660×10^6
k_{20}	s^{-1}	7.212×10^8
k_{30}	s^{-1}	2.675×10^{12}
E_1	$kJ\,mol^{-1}$	5.5427×10^4
E_2	$kJ\,mol^{-1}$	6.9280×10^4
E_3	$kJ\,mol^{-1}$	9.2377×10^4
k_{10}^*	s^{-1}	6.7157×10^4
k_{20}^*	s^{-1}	1.0341×10^5

Figures 3.6-3.8 show the performance of G-MA and LG-MA with three different nominal models as given in Table 3.2. The RTO scheme is initialized at the nominal optimal solution and proceeds towards the plant optimum. The three trajectories labeled I, II and III correspond to the use of the three models in Table 3.2. Diagonal filter matrices, with eigenvalues of 0.2, were used in Equations (3.2.30) to (3.2.31). Both algorithms converge rapidly to the optimal solution for the plant, where the constraint on X_A is active ($X_A = 0.09$, which results in $X_{A,s} = 0.06$). The main observation to be made is that both algorithms behave very similarly. This is to be expected, as we proved in Section 3.2.2 that LG-MA is a linearized version of G-MA. Hence, either algorithm can be used, bearing in mind that LG-MA is computationally advantageous.

Note that, for this simulation study, the finite-difference method is used to estimate plant gradients. At the k^{th} iteration, three different values of $\mathbf{c}_s$ are applied to the plant, $\mathbf{c}_k$, $\mathbf{c}_k + [\Delta X_{A,s}, 0]^T$ and $\mathbf{c}_k + [0, \Delta F_{B,s}]^T$, where $\Delta X_{A,s}$ and $\Delta F_{B,s}$ are small perturbations. The gradient estimate is then computed as:

$$\nabla_{\mathbf{c}} \Phi_{E,k} = \begin{bmatrix} \frac{\Phi_p(\mathbf{c}_k + [\Delta X_{A,s}, 0]^T) - \Phi_p(\mathbf{c}_k)}{\Delta X_{A,s}} \\ \frac{\Phi_p(\mathbf{c}_{s,k} + [0, \Delta F_{B,s}]^T) - \Phi_p(\mathbf{c}_k)}{\Delta F_{B,s}} \end{bmatrix}. \tag{3.3.10}$$

Hence, each RTO iteration actually corresponds to application of *three* different sets of inputs to the plant.

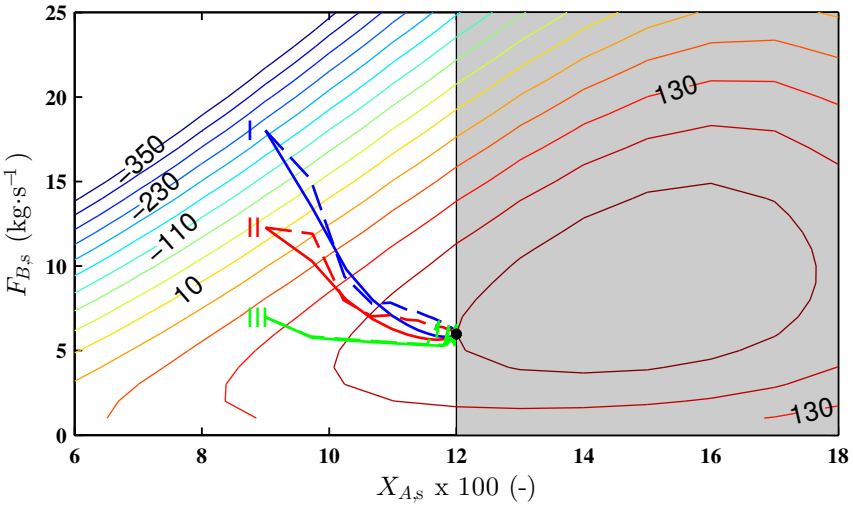

Figure 3.6: Evolution of the plant inputs **c** during the first 20 iterations of the generalized MA scheme for Cases I-III. Solid = G-MA, Dashed = LG-MA. In each case, the starting point, which is the nominal optimal solution, is marked by a roman numeral. The contour lines represent the plant profit. The shaded region is infeasible for the plant due to the constraint on X_A. Black dot = plant optimum.

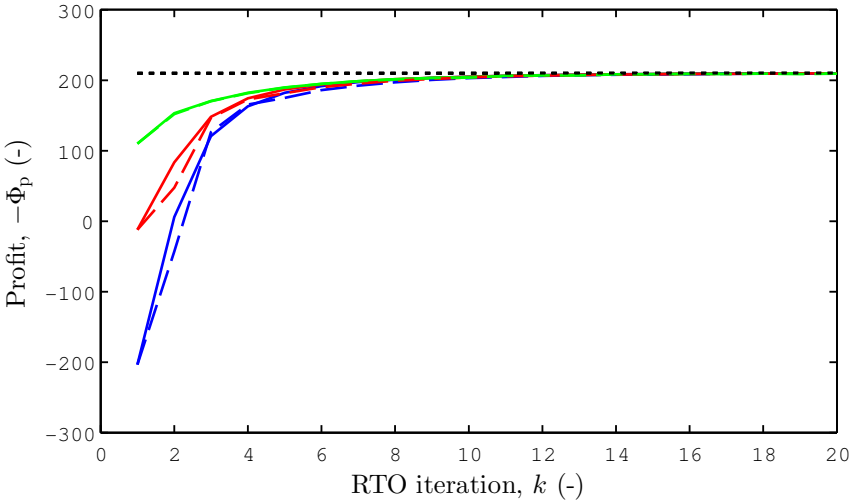

Figure 3.7: The profit as a function of the iteration number k. Blue/red/green = Cases I/II/III. Solid = G-MA, Dashed = LG-MA.

Table 3.2: Values of the variable model parameters for three different cases

Case	E_1^* (kJ mol^{-1})	E_2^* (kJ mol^{-1})
I	8050	12500
II	8100	12500
III	8100	12300

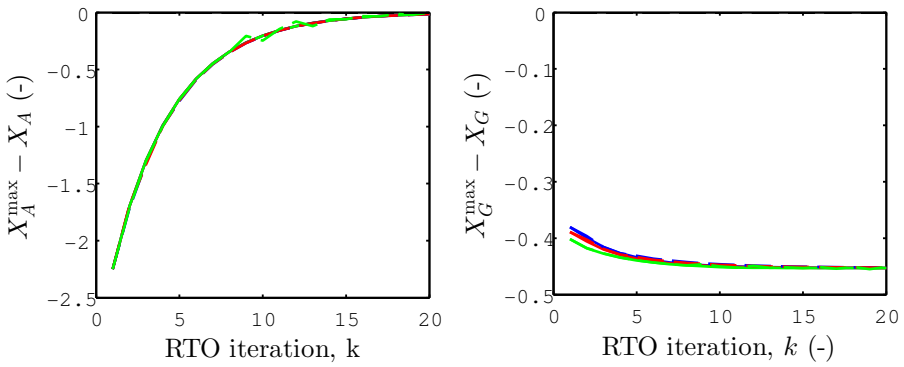

Figure 3.8: The constraints on X_A and X_G as a function of the RTO iteration number k. Blue/red/green = Cases I/II/III. Solid = G-MA, Dashed = LG-MA. The dotted line indicates $G_{p,1} = 0$.

3.4 Conclusions

Modifier Adaptation relies on a model of the process. It is typically assumed that the model and the plant have the same inputs. As the real-life example of an incineration plant showed, this assumption may not hold in practice. In addition, it was shown that closed-loop systems, for which only an open-loop model is available, will also violate this assumption. Obtaining a new model with the same inputs as the plant may not be feasible when the process is complex or the control system is not fully known. Generalized MA avoids remodeling the system, and this at no extra computational cost. It follows that a broader class of process-optimization problems can be tackled with MA.

This chapter has shown that Generalized MA conserves the valuable KKT matching property, i.e. convergence will only occur to a *plant* KKT point. The simulation results reveal that both methods presented in this Chapter perform very similarly. Thus, it is recommended that LG-MA be used, as it is computationally advantageous.

4 Directional Modifier Adaptation

In Chapter 1 we saw that a number of challenges currently face the field of RTO, namely: i) guaranteeing constraint satisfaction, ii) providing online diagnostics to understand if a point is in fact the plant optimum, iii) fast convergence, and iv) straightforward design procedures. The Modifier Adaptation (MA) method satisfies points i), ii) and iv), but not necessarily point iii). As discussed in Section 2.3.3, if the plant has many inputs, the speed of convergence of the RTO algorithm to the plant optimum will generally be very slow using the standard MA algorithm. This is because an accurate estimate of the plant gradients with respect to *all the plant inputs* must be available at each iteration. To obtain the gradient estimates, at each RTO iteration, there must be a number of nearby past operating points equal to or greater than the number of plant inputs. What is more, these past input moves must be linearly independent, i.e. all directions in the plant's input space must have been explored. This fundamental requirement applies even if a dual-control approach is used. As a result, the MA algorithm cannot progress directly towards the plant optimum, as this would imply moving in only one direction of the plant's input space. Instead, it must move slowly and cautiously, exciting all directions of the plant's input space. In many cases, such slow convergence towards the plant optimum is not acceptable. The plant optimum is constantly evolving due to slowly time-varying disturbances and process drift. If the RTO algorithm cannot converge to the plant optimum significantly faster than the rate at which the plant optimum itself changes, then it has little hope of tracking it.

This chapter proposes a Directional Modifier Adaptation (D-MA) method, which is designed to converge towards the plant optimum significantly faster than standard MA. The idea is to estimate the plant gradients only *in privileged directions* of the input space. This 'directional' gradient estimation requires far fewer past measurement points. As a result, D-MA can focus less on exploring the input space for gradient estimation, and more on optimizing the plant. The D-MA algorithm has the following characteristics:

1. *Constraint satisfaction* is ensured upon convergence, even for large numbers of complex constraints.
2. *Plant optimality with respect to a subset of the plant inputs* is guaranteed upon convergence, despite the use of an inaccurate model.
3. *Rapid convergence* is enforced, even in the presence of significant noise. The convergence speed is independent of the number of plant inputs.
4. *Straightforward design procedure* using the available model.

The chapter is structured as follows. Section 4.1 presents the novel D-MA algorithm and examines its properties. In addition, a theoretically motivated approach is presented for calculating the privileged directions in which to estimate plant derivatives. Section 4.2 presents a Dual D-MA algorithm, which simultaneously estimates the plant gradient and searches for the plant optimum. Finally, this algorithm is applied to the challenging problem of optimizing the flight path of a power-generating kite in Section 4.3.

4.1 Basic Idea

This section presents a very simple, novel method to circumvent the prohibitive experimental cost of estimating plant gradients when n_u is large, as is typically the case for complex continuous processes and discretized dynamic-optimization problems. The idea is that rather than estimating the full gradient of the plant cost and constraints, only a directional derivative (i.e. the gradient in certain directions) is estimated.

4.1.1 Directional Derivatives

This chapter makes extensive use of *directional derivatives*. While the directional derivative is sometimes defined as being the derivative of a vector function with respect to variations in *one* direction only, here it will be defined with respect to variations in a *subspace* of directions:

Definition 4.1.1 (Directional Derivative)**.** *The $(n_f \times n_r)$-dimensional directional derivative of a n_f-dimensional vector function* $\mathbf{f}$ *is:*

$$\nabla_{\mathbf{U}_r}\mathbf{f}(\mathbf{u}) := \left.\frac{\partial \mathbf{f}(\mathbf{u}+\mathbf{U}_r\mathbf{r})}{\partial \mathbf{r}}\right|_{\mathbf{r}=\mathbf{0}}, \tag{4.1.1}$$

where $\mathbf{U}_r = [\delta\mathbf{u}_1 \ \cdots \ \delta\mathbf{u}_r]$ *is an $n_u \times n_r$ matrix, the columns of which contain the $n_r < n_u$ directions in the input space that the directional derivative is evaluated in, and the dimension of* $\mathbf{r}$ *is n_r.*

Note that in the limiting case where $n_r = n_u$, the directional derivative thus defined is simply the gradient (usually referred to as the Jacobian if $n_f > 1$), of the vector function. Indeed, the directional derivative can be derived from the gradient:

Property 4.1.1. *Applying the chain rule to Equation* (4.1.1) *yields:*

$$\nabla_{\mathbf{U}_r}\mathbf{f}(\mathbf{u}) = \left.\frac{\partial \mathbf{f}(\mathbf{u}+\mathbf{U}_r\mathbf{r})}{\partial \mathbf{r}}\right|_{\mathbf{r}=\mathbf{0}} = \nabla\mathbf{f}(\mathbf{u}+\mathbf{U}_r\mathbf{r}) \left.\frac{\partial(\mathbf{u}+\mathbf{U}_r\mathbf{r})}{\partial \mathbf{r}}\right|_{\mathbf{r}=\mathbf{0}} = \nabla\mathbf{f}(\mathbf{u})\mathbf{U}_r. \tag{4.1.2}$$

The directional derivative can be considered a partial gradient. It embodies information about how the function varies (locally) in certain directions of the function's input space. A function's variability in a particular direction, $\mathbf{x}$, is typically calculated as: $\nabla\mathbf{f}(\mathbf{u})\mathbf{x}$. If $\mathbf{x} \in C(\mathbf{U}_r)$, then this can also be calculated using the directional derivative in place of the full gradient:

Property 4.1.2.

$$\nabla_{\mathbf{U}_r}\mathbf{f}(\mathbf{u})\mathbf{U}_r^+\mathbf{x} = \begin{cases} \nabla\mathbf{f}(\mathbf{u})\mathbf{x} & \mathbf{x} \in C(\mathbf{U}_r) \\ \mathbf{0} & \mathbf{x} \notin C(\mathbf{U}_r) \end{cases}, \tag{4.1.3}$$

where $C(\mathbf{U}_r)$ *is the column space of* $\mathbf{U}_r$.

Property 4.1.2 follows from Property 4.1.1 by noting that

$$\mathbf{U}_r\mathbf{U}_r^+\mathbf{x} = \begin{cases} \mathbf{x} & \mathbf{x} \in C(\mathbf{U}_r) \\ \mathbf{0} & \mathbf{x} \notin C(\mathbf{U}_r) \end{cases}. \tag{4.1.4}$$

The D-MA algorithm is essentially the standard MA algorithm, with gradients replaced by directional derivatives. The idea is that it is easier to obtain an experimental estimate of a directional derivative than a full gradient.

Algorithm 4.1.1: Algorithm: Directional Modifier Adaptation (D-MA)

For the basic D-MA algorithm, the following modifications are applied to the relevant steps in the standard MA algorithm (Algorithm 2.3.1):

Initialize: In addition, choose a matrix of 'privileged' input directions $\mathbf{U}_r$, in which to estimate plant derivatives. Section 4.1.2 explains how to choose $\mathbf{U}_r$.

3. This step is replaced by the following:

Estimate the *directional derivative* of the plant cost, $\nabla_{\mathbf{U}_\mathrm{r}}\phi_{\mathrm{E},k}$, and the plant constraints, $\nabla_{\mathbf{U}_\mathrm{r}}\mathbf{g}_{\mathrm{E},k}$, at the current operating point $\mathbf{u}_k$. These derivatives must be estimated using measurements collected at no less than n_r successive operating points close to $\mathbf{u}_k$. This can be done using finite differences or the novel approach proposed in Section 4.2. Estimate the cost gradient as:

$$\nabla\phi_{\mathrm{E},k} = \nabla\phi(\mathbf{u}_k)(\mathbf{I}_{n_u} - \mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+) + \nabla_{\mathbf{U}_\mathrm{r}}\phi_{\mathrm{E},k}\mathbf{U}_\mathrm{r}^+, \tag{4.1.5}$$

and likewise for the constraint gradient estimate.

Note that if the estimated directional derivative is accurate, $\nabla_{\mathbf{U}_\mathrm{r}}\phi_{\mathrm{E},k} = \nabla_{\mathbf{U}_\mathrm{r}}\phi_\mathrm{p}(\mathbf{u}_k)$ and, according to property (4.1.2), Equation (4.1.5) implies that:

$$\nabla\phi_{\mathrm{E},k}\delta\mathbf{u} = \begin{cases} \nabla\phi_\mathrm{p}(\mathbf{u}_k)\delta\mathbf{u} & \delta\mathbf{u} \in C(\mathbf{U}_\mathrm{r}) \\ \nabla\phi(\mathbf{u}_k)\delta\mathbf{u} & \delta\mathbf{u} \notin C(\mathbf{U}_\mathrm{r}) \end{cases}. \tag{4.1.6}$$

The gradient estimate matches the plant gradient *in the n_r privileged directions*, and the model gradient in all the other directions. D-MA allows the user to *choose* which input directions the MA algorithm will pay particular attention to. Although D-MA will not, in general, reach a point satisfying the KKT conditions for the plant, if it converges, it will do so to a point where the *plant* performance cannot (locally) be improved by adapting $\mathbf{u}$ in any of the privileged directions. This is formalized in the following theorem:

Theorem 4.1.1 (Directional Optimality upon Convergence)**.** *If perfect plant directional-derivative estimates are available, and in the absence of noise, any point $\mathbf{u}_\infty$ that the D-MA algorithm converges to will be such that $\mathbf{r} = \mathbf{0}$ is a KKT point for the following problem:*

$$\begin{aligned} \min_{\mathbf{r}} \ & \phi_\mathrm{p}(\mathbf{u}_\infty + \mathbf{U}_\mathrm{r}\mathbf{r}) \\ \text{s.t.} \ & \mathbf{g}_\mathrm{p}(\mathbf{u}_\infty + \mathbf{U}_\mathrm{r}\mathbf{r}) \leq \mathbf{0}. \end{aligned} \tag{4.1.7}$$

Proof. Upon convergence of the D-MA algorithm

$$\boldsymbol{\epsilon}_\infty = \mathbf{g}_\mathrm{p}(\mathbf{u}_\infty^*) - \mathbf{g}(\mathbf{u}_\infty^*), \tag{4.1.8}$$

assuming noise-free constraint measurements. Also, according to Equations (2.3.9) and

(2.3.8):

$$\left(\boldsymbol{\lambda}_{\infty}^{\phi}\right)^{T} = \nabla\phi_{\mathrm{E},\infty} - \nabla\phi(\mathbf{u}_{\infty}, \boldsymbol{\theta}_0), \tag{4.1.9}$$

$$\left(\boldsymbol{\lambda}_{\infty}^{\mathrm{g}}\right)^{T} = \nabla\mathbf{g}_{\mathrm{E},\infty} - \nabla\mathbf{g}(\mathbf{u}_{\infty}, \boldsymbol{\theta}_0). \tag{4.1.10}$$

Equation (4.1.8) implies that the constraint for the modified model-based optimization problem (2.3.4) matches that of the plant at $\mathbf{u}_{\infty}$:

$$\mathbf{g}_{\mathrm{m},\infty}(\mathbf{u}_{\infty}) = \mathbf{g}_{\mathrm{p}}(\mathbf{u}_{\infty}). \tag{4.1.11}$$

Also, the KKT conditions for the modified optimization problem (2.3.4) must be satisfied upon convergence. Hence,

$$\mathbf{g}_{\mathrm{m},\infty}(\mathbf{u}_{\infty}) = \mathbf{g}_{\mathrm{p}}(\mathbf{u}_{\infty}) \leq \mathbf{0}, \tag{4.1.12}$$

and there exists a $\boldsymbol{\nu} \geq \mathbf{0}$ s.t.

$$\boldsymbol{\nu}_i \mathbf{g}_{\mathrm{m},\infty,i}(\mathbf{u}_{\infty}) = \boldsymbol{\nu}_i \mathbf{g}_{\mathrm{p},i}(\mathbf{u}_{\infty}) = 0, \quad \forall i = 1, \ldots, n_g, \tag{4.1.13}$$

and

$$\nabla\phi_{\mathrm{m},\infty}(\mathbf{u}_{\infty}) + \boldsymbol{\nu}^T \nabla\mathbf{g}_{\mathrm{m},\infty}(\mathbf{u}_{\infty}) = \mathbf{0} \tag{4.1.14}$$

$$\implies \nabla\phi(\mathbf{u}_{\infty}, \boldsymbol{\theta}_0) + \left(\boldsymbol{\lambda}_{\infty}^{\phi}\right)^{T} + \boldsymbol{\nu}^T \left(\nabla\mathbf{g}(\mathbf{u}_{\infty}, \boldsymbol{\theta}_0) + \left(\boldsymbol{\lambda}_{\infty}^{\mathrm{g}}\right)^{T}\right) = \mathbf{0} \tag{4.1.15}$$

$$\implies \nabla\phi_{\mathrm{E},k} + \boldsymbol{\nu}^T \nabla\mathbf{g}_{\mathrm{E},k} = 0. \tag{4.1.16}$$

Post-multiplying Equation (4.1.16) by $\mathbf{U}_{\mathrm{r}}$ and using Equation (4.1.5) yields:

$$\nabla_{\mathbf{U}_{\mathrm{r}}}\phi_{\mathrm{E},\infty} + \boldsymbol{\nu}^T \nabla_{\mathbf{U}_{\mathrm{r}}}\mathbf{g}_{\mathrm{E},\infty} = \mathbf{0}. \tag{4.1.17}$$

Assuming perfect gradient estimates, i.e. $\nabla_{\mathbf{U}_{\mathrm{r}}}\phi_{\mathrm{E},\infty} = \nabla_{\mathbf{U}_{\mathrm{r}}}\phi_{\mathrm{p}}(\mathbf{u}_{\infty})$ and $\nabla_{\mathbf{U}_{\mathrm{r}}}\mathbf{g}_{\mathrm{E},\infty} = \nabla_{\mathbf{U}_{\mathrm{r}}}\mathbf{g}_{\mathrm{p}}(\mathbf{u}_{\infty})$, and using Definition 4.1.1, yields:

$$\frac{\partial\phi_{\mathrm{p}}(\mathbf{u}_{\infty} + \mathbf{U}_{\mathrm{r}}\mathbf{r})}{\partial\mathbf{r}} + \boldsymbol{\nu}^T \left.\frac{\partial\mathbf{g}_{\mathrm{p}}(\mathbf{u}_{\infty} + \mathbf{U}_{\mathrm{r}}\mathbf{r})}{\partial\mathbf{r}}\right|_{\mathbf{r}=\mathbf{0}} = 0. \tag{4.1.18}$$

Since $\mathbf{u}_{\infty} + \mathbf{U}_{\mathrm{r}}\mathbf{r}$ obviously equals $\mathbf{u}_{\infty}$ when $\mathbf{r} = \mathbf{0}$, Equations (4.1.12)and (4.1.13) mean that the primal- and dual-feasibility conditions for Problem (4.1.7) are satisfied at $\mathbf{r} = \mathbf{0}$. Together with the fact that Equation(4.1.18) shows the satisfaction of the stationarity KKT condition for Problem (4.1.7) at $\mathbf{r} = \mathbf{0}$, this proves that $\mathbf{r} = \mathbf{0}$ is a KKT point for Problem (4.1.7). □

4.1.2 Choosing the Privileged Directions

The most important aspect of D-MA is the choice of the n_r privileged directions (the columns of $\mathbf{U}_\mathrm{r}$). D-MA acts at two levels. It will a) adapt the input in any directions necessary to ensure constraint satisfaction, and b) try to improve the cost by adapting the decision variables $\mathbf{u}$ in the privileged directions. It is important to note that, regardless of $\mathbf{U}_\mathrm{r}$, constraint satisfaction (upon convergence) is ensured. While the available model may be inaccurate (for example it may predict a cost value with 50% error for a given input $\mathbf{u}$), it is assumed that it describes the main optimization trade-offs, and gives a reasonable indication of the effect of uncertain parameters on the optimal solution. Simple tendency models (Filippi-Bossy et al., 1989), if well designed (i.e. with optimization and parametric analysis in mind), can fulfill these requirements. MA for constrained problems attempts to match the Lagrangian gradient for the modified model-based optimization problem with that of the plant-based problem. Hence, in the case of MA, parametric analysis of the model should be used to study the effect of parameter variations on the *Lagrangian's gradient*. If all likely parameter variations only cause notable change in the Lagrangian gradient *in a few directions*, then it will suffice to only estimate the gradient in these few directions. This is formalized in the following theorem.

Theorem 4.1.2 (Optimal Gradient Directions for Small Parametric Uncertainty)**.** *Consider small parametric plant-model mismatch, that is,* $\phi_\mathrm{p}(\mathbf{u}) = \phi(\mathbf{u},\boldsymbol{\theta}_\mathrm{p})$ *and* $\mathbf{g}_\mathrm{p}(\mathbf{u}) = \mathbf{g}(\mathbf{u},\boldsymbol{\theta}_\mathrm{p})$ *with* $\boldsymbol{\theta}_\mathrm{p} = \boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta}$*. Then, in the absence of noise and assuming perfect directional-derivative estimates, the plant optimal solution* $\mathbf{u}_\mathrm{p}^*$ *is a fixed point for the D-MA algorithm if the direction matrix is chosen as:*

$$\mathbf{U}_\mathrm{r} = \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u} \partial \boldsymbol{\theta}}(\mathbf{u}^*(\boldsymbol{\theta}_0), \boldsymbol{\nu}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta}_0) \quad \in \mathbb{R}^{n_u \times n_\theta}, \tag{4.1.19}$$

where $\mathbf{L}(\mathbf{u},\boldsymbol{\nu},\boldsymbol{\theta}) = \phi(\mathbf{u},\boldsymbol{\theta}) + \boldsymbol{\nu}^T\mathbf{g}(\mathbf{u},\boldsymbol{\theta})$ *is the Lagrangian,* $\mathbf{u}^*(\boldsymbol{\theta}_0)$ *is the nominal optimal solution, and* $\boldsymbol{\nu}^*(\boldsymbol{\theta}_0)$ *are the corresponding Lagrange multipliers for the model-based problem.*

Proof. A sufficient condition for $\mathbf{u}_\infty$ to be a fixed point for the D-MA algorithm is that it satisfies the first-order KKT conditions for the modified model-based optimization problem (2.3.4), assuming it is not a non-minimum stationary point for this problem. The stationary KKT conditions mean $\exists\, \boldsymbol{\nu}$ such that:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\infty, \boldsymbol{\nu}, \boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_\infty^\phi)^T + \boldsymbol{\nu}^T(\boldsymbol{\lambda}_\infty^g)^T = \mathbf{0}, \tag{4.1.20}$$

with

$$(\boldsymbol{\lambda}_{\infty}^{\phi})^T = \nabla\phi_{\mathrm{E},\infty} - \nabla\phi(\mathbf{u}_{\infty},\boldsymbol{\theta}_0)$$
$$= (\nabla_{\mathbf{U}_\mathrm{r}}\phi_{\mathrm{E},\infty} - \nabla_{\mathbf{U}_\mathrm{r}}\phi(\mathbf{u}_{\infty},\boldsymbol{\theta}_0))\mathbf{U}_\mathrm{r}^{+}, \tag{4.1.21}$$

where Equation (4.1.21) is obtained by combining the definition of the gradient estimate (4.1.5) with the definition of the gradient modifiers (2.3.8) upon convergence. In the same manner, the constraint gradient modifiers upon convergence are:

$$(\boldsymbol{\lambda}_{\infty}^{g})^T = (\nabla_{\mathbf{U}_\mathrm{r}}\mathbf{g}_{\mathrm{E},\infty} - \nabla_{\mathbf{U}_\mathrm{r}}\mathbf{g}(\mathbf{u}_{\infty},\boldsymbol{\theta}_0))\mathbf{U}_\mathrm{r}^{+}. \tag{4.1.22}$$

In addition, the primal and dual feasibility KKT conditions for the modified model-based problem must be satisfied. Due to the matching of the modified model constraints and the plant constraints upon convergence, these conditions are:

$$\mathbf{g}_\mathrm{p}(\mathbf{u}_{\infty}) \leq \mathbf{0}, \qquad \boldsymbol{\nu}^T\mathbf{g}_\mathrm{p}(\mathbf{u}_{\infty}) = \mathbf{0}. \tag{4.1.23}$$

We now show that $\mathbf{u}_{\infty} = \mathbf{u}_\mathrm{p}^*$ satisfies Conditions (4.1.20) and (4.1.23), with $\boldsymbol{\nu} = \boldsymbol{\nu}_\mathrm{p}^*$. Firstly, as $\mathbf{u}_\mathrm{p}^*$ is a KKT point for the plant, Conditions (4.1.23) are satisfied. Also, $\mathbf{u}_\mathrm{p}^*$ satisfies the stationary KKT condition for the plant optimization problem, which reads:

$$\nabla\phi_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) + (\boldsymbol{\nu}_\mathrm{p}^*)^T\nabla\mathbf{g}_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) = \mathbf{0}. \tag{4.1.24}$$

Since $\mathbf{L}(\mathbf{u}_\mathrm{p}^*,\boldsymbol{\nu}_\mathrm{p}^*,\boldsymbol{\theta}_\mathrm{p}) = \phi(\mathbf{u}_\mathrm{p}^*,\boldsymbol{\theta}_\mathrm{p}) + \boldsymbol{\nu}_\mathrm{p}^{*T}\mathbf{g}(\mathbf{u}_\mathrm{p}^*,\boldsymbol{\theta}_\mathrm{p}) = \phi_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) + \boldsymbol{\nu}_\mathrm{p}^{*T}\mathbf{g}_\mathrm{p}(\mathbf{u}_\mathrm{p}^*)$, it follows that:

$$\frac{\partial\mathbf{L}}{\partial\mathbf{u}}(\mathbf{u}_\mathrm{p}^*,\boldsymbol{\nu}_\mathrm{p}^*,\boldsymbol{\theta}_\mathrm{p}) = \nabla\phi_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) + (\boldsymbol{\nu}_\mathrm{p}^*)^T\nabla\mathbf{g}_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) = \mathbf{0}. \tag{4.1.25}$$

Developing this into a Taylor series around $\boldsymbol{\theta}_0$ leads to:

$$\frac{\partial\mathbf{L}}{\partial\mathbf{u}}(\mathbf{u}_\mathrm{p}^*,\boldsymbol{\nu}_\mathrm{p}^*,\boldsymbol{\theta}_0) + \Delta\boldsymbol{\theta}^T\frac{\partial^2\mathbf{L}}{\partial\mathbf{u}\partial\boldsymbol{\theta}}^T(\mathbf{u}_\mathrm{p}^*,\boldsymbol{\nu}_\mathrm{p}^*,\boldsymbol{\theta}_0) + O(\Delta\boldsymbol{\theta}^2) = \mathbf{0}. \tag{4.1.26}$$

Note that as $(\mathbf{u}_\mathrm{p}^* - \mathbf{u}^*(\boldsymbol{\theta}_0))$ and $(\boldsymbol{\nu}_\mathrm{p}^* - \boldsymbol{\nu}^*(\boldsymbol{\theta}_0))$ depend linearly on $\Delta\boldsymbol{\theta}$, which is a standard result from parametric sensitivity analysis (Fiacco, 1983):

$$\frac{\partial^2\mathbf{L}}{\partial\mathbf{u}\partial\boldsymbol{\theta}}(\mathbf{u}_\mathrm{p}^*,\boldsymbol{\nu}_\mathrm{p}^*,\boldsymbol{\theta}_0) = \frac{\partial^2\mathbf{L}}{\partial\mathbf{u}\partial\boldsymbol{\theta}} + \frac{\partial}{\partial\mathbf{u}}\left(\frac{\partial^2\mathbf{L}}{\partial\mathbf{u}\partial\boldsymbol{\theta}}\right)\frac{\partial\mathbf{u}^*}{\partial\boldsymbol{\theta}}\Delta\boldsymbol{\theta} +$$
$$\frac{\partial}{\partial\boldsymbol{\nu}}\left(\frac{\partial^2\mathbf{L}}{\partial\mathbf{u}\partial\boldsymbol{\theta}}\right)\frac{\partial\boldsymbol{\nu}^*}{\partial\boldsymbol{\theta}}\Delta\boldsymbol{\theta}\Big|_{(\mathbf{u}^*(\boldsymbol{\theta}_0),\boldsymbol{\nu}^*(\boldsymbol{\theta}_0),\boldsymbol{\theta}_0)} + O(\Delta\boldsymbol{\theta}^2) \tag{4.1.27}$$
$$= \frac{\partial^2\mathbf{L}}{\partial\mathbf{u}\partial\boldsymbol{\theta}}(\mathbf{u}^*(\boldsymbol{\theta}_0),\boldsymbol{\nu}^*(\boldsymbol{\theta}_0),\boldsymbol{\theta}_0) + O(\Delta\boldsymbol{\theta}) \tag{4.1.28}$$
$$= \mathbf{U}_\mathrm{r} + O(\Delta\boldsymbol{\theta}), \tag{4.1.29}$$

and using the matrix identity $\mathbf{X}^T = \mathbf{X}^T\mathbf{X}\mathbf{X}^+$, we can write:

$$\Delta\boldsymbol{\theta}^T \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}^T (\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) = \Delta\boldsymbol{\theta}^T \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}^T (\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0)\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+ + O(\Delta\boldsymbol{\theta}^2). \tag{4.1.30}$$

Equation (4.1.26) can now be written as:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + \Delta\boldsymbol{\theta}^T \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}^T (\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0)\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+ + O(\Delta\boldsymbol{\theta}^2) = \mathbf{0} \tag{4.1.31}$$

$$\implies \frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + \nabla\left(\phi(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p}) - \phi(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_0)\right)\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+ + (\boldsymbol{\nu}_\mathrm{p}^*)^T\nabla\left(\mathbf{g}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p}) - \mathbf{g}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_0)\right)\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+ + O(\Delta\boldsymbol{\theta}^2) = \mathbf{0} \tag{4.1.32}$$

$$\implies \frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + \nabla_{\mathbf{U}_\mathrm{r}}\left(\phi(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p}) - \phi(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_0)\right)\mathbf{U}_\mathrm{r}^+ + (\boldsymbol{\nu}_\mathrm{p}^*)^T\nabla_{\mathbf{U}_\mathrm{r}}\left(\mathbf{g}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p}) - \mathbf{g}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_0)\right)\mathbf{U}_\mathrm{r}^+ + O(\Delta\boldsymbol{\theta}^2) = \mathbf{0}. \tag{4.1.33}$$

If it is assumed that the gradient estimate is perfect, i.e. $\nabla_{\mathbf{U}_\mathrm{r}}\phi_{\mathrm{E},\infty} = \nabla_{\mathbf{U}_\mathrm{r}}\phi_\mathrm{p}(\mathbf{u}_\infty) = \nabla_{\mathbf{U}_\mathrm{r}}\phi(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p})$ (and likewise for the constraint gradient estimates), and that $O(\Delta\boldsymbol{\theta}^2) \approx 0$, this becomes:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\infty, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_\infty^\phi)^T + (\boldsymbol{\nu}_\mathrm{p}^*)^T(\boldsymbol{\lambda}_\infty^g)^T = \mathbf{0}, \tag{4.1.34}$$

with the modifier terms defined as in Equations (4.1.21) and (4.1.22) (recalling that $\mathbf{u}_\infty = \mathbf{u}_\mathrm{p}^*$). Hence, Condition (4.1.20) is satisfied, and $\mathbf{u}_\mathrm{p}^*$ is a fixed (stationary) point for the D-MA algorithm.

□

This result provides a theoretical motivation for using parametric sensitivity analysis to determine the adaptation directions. From the practical point of view, several simulation case studies have confirmed that, even when there is significant parametric mismatch, this approach systematically chooses very appropriate adaptation directions. Indeed, as shown in the example of Section 4.3, it can even yield nearly 'optimal' adaptation directions when there is structural plant-model mismatch. It will not usually be necessary to use all of the n_θ directions given by $\frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}(\mathbf{u}^*(\boldsymbol{\theta}_0), \boldsymbol{\nu}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta}_0)$. Marchetti (2013) proves that "*when the available cost and constraint gradients are estimated quantities, the loss in cost induced will be determined by the resulting error in the gradient of the Lagrangian function*".

Theorem 4.1.3 (Optimality Loss due to Lagrangian Gradient Error)**.** *The optimality loss*

due to a small Lagrangian gradient error is:

$$\phi_{\mathrm{p}}(\mathbf{u}_{\mathrm{p}}^{*}) - \phi_{\mathrm{p}}(\mathbf{u}^{*}(\boldsymbol{\theta}_0)) = -\boldsymbol{\epsilon}^T \mathbf{A} \boldsymbol{\epsilon} + O(\boldsymbol{\epsilon}^3) \tag{4.1.35}$$

$$\boldsymbol{\epsilon} = \frac{\partial \mathbf{L}_{\mathrm{p}}}{\partial \mathbf{u}}(\mathbf{u}, \boldsymbol{\nu}) - \frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\theta}_0) \tag{4.1.36}$$

where $\mathbf{L}(\mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\theta}) = \phi(\mathbf{u}, \boldsymbol{\theta}) + \boldsymbol{\nu}^T \mathbf{g}(\mathbf{u}, \boldsymbol{\theta})$ *and* $\mathbf{L}_{\mathrm{p}}(\mathbf{u}, \boldsymbol{\nu}) = \phi_{\mathrm{p}}(\mathbf{u}) + \boldsymbol{\nu}^T \mathbf{g}_{\mathrm{p}}(\mathbf{u})$ *are Lagrangians for the model-based and the plant-based problems, respectively, and* $\mathbf{A}$ *depends on the plant equations.*

Proof. See Marchetti (2013). □

Hence, the optimality loss is approximately proportional to a weighted norm of the Lagrangian gradient error, meaning larger Lagrangian gradient error will result in more optimality loss. Singular value decomposition(SVD) can be used to single out those directions in which the Lagrangian gradient will be most affected by parameter variations. If $\theta_i^{\max}$ and $\theta_i^{\min}$ are the maximum and minimum expected values of the uncertain parameter θ_i, the effect of a normalized parameter variation on the gradient of the Lagrangian is given by the following transformation:

$$\mathbf{U}\Sigma\mathbf{V}^T = \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u} \partial \boldsymbol{\theta}}(\mathbf{u}^{*}(\boldsymbol{\theta}_0), \boldsymbol{\nu}^{*}(\boldsymbol{\theta}_0), \boldsymbol{\theta}_0)\mathrm{diag}(\theta_1^{\max} - \theta_1^{\min}, \ldots, \theta_{n_\theta}^{\max} - \theta_{n_\theta}^{\min}), \tag{4.1.37}$$

where $\mathbf{U}$, Σ and $\mathbf{V}$ are the matrices of the ordered SVD, i.e. the elements (singular values) $\sigma_1, \sigma_2, \ldots$ on the diagonal of $\boldsymbol{\Sigma}$ descend in magnitude. $\mathbf{U}_{\mathrm{r}}$ can be chosen as the first $n_r < n_\theta$ columns of $\mathbf{U}$, which are those directions corresponding to the n_r largest singular values. The number of directions, n_r, should be chosen such that $\sigma_{n_r+1} << \sigma_{n_r}$. This ensures that the maximum variation of the Lagrangian gradient in the neglected directions due to parametric mismatch is relatively small (and thus the resulting optimality loss is negligible).

4.2 Dual Directional Modifier Adaptation

An efficient MA implementation should use all available information, for example all appropriate past measurements, to estimate experimental derivatives. This section develops a 'dual control' approach to D-MA that not only optimizes the plant, but also ensures an accurate directional-derivative estimate can be calculated using the past operating points. Firstly, a gradient estimation technique is proposed that combines information from all available measurements in the vicinity of the current RTO point.

The measurements are reconciled in a statistically optimal manner to maximally reject the effect of noise. A confidence interval is obtained for the gradient estimate, as its variance (which is minimized by the estimation procedure) is also calculated. Secondly, an excitation-rewarding term is added to the modified model-based optimization problem. This term incites the RTO algorithm to take steps that will improve the gradient estimate *in the privileged directions*.

4.2.1 Gradient Estimation using Previous Measurements

The method proposed here is iterative. At each RTO iteration, a reliable gradient estimate is constructed, starting with the nominal model gradient. The past measurements are integrated into the gradient estimate one at a time. Using the measured cost at the current RTO point $\mathbf{u}_k$ and that at a previous RTO point, $\mathbf{u}_j$, the directional derivative in the one direction $\delta\mathbf{u} = \frac{\mathbf{u}_j - \mathbf{u}_k}{\|\mathbf{u}_j - \mathbf{u}_k\|}$ can be estimated as

$$\nabla_{\delta\mathbf{u}}\phi_{\mathrm{E}} = \frac{\tilde{\phi}_{\mathrm{p}}(\mathbf{u}_j) - \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k)}{\|\mathbf{u}_j - \mathbf{u}_k\|} \tag{4.2.1}$$

$$= \nabla_{\delta\mathbf{u}}\phi_{\mathrm{p}}(\mathbf{u}_k) + \frac{d_j^{\phi} - d_k^{\phi}}{\|\mathbf{u}_j - \mathbf{u}_k\|} + O\left(\|\mathbf{u}_j - \mathbf{u}_k\|\right). \tag{4.2.2}$$

If $\|\mathbf{u}_j - \mathbf{u}_k\|$ is sufficiently small, the last term (the truncation error) can be neglected, and

$$\sigma_{\mathrm{E}}^2 = var\{\nabla_{\delta\mathbf{u}}\phi_{\mathrm{E}}\} = \frac{2\sigma_{\phi}^2}{\|\mathbf{u}_j - \mathbf{u}_k\|^2}. \tag{4.2.3}$$

This estimate of the directional derivative can be combined with an existing gradient estimate, $\nabla\phi_{\mathrm{old}}$, using a weighted rank-1 (Broyden) update to give the new gradient estimate:

$$\nabla\phi_{\mathrm{new}} = \nabla\phi_{\mathrm{old}} + \kappa(\nabla_{\delta\mathbf{u}}\phi_{\mathrm{E}} - \nabla\phi_{\mathrm{old}}\delta\mathbf{u})\delta\mathbf{u}^T, \tag{4.2.4}$$

with the variance matrix

$$\boldsymbol{\Sigma}_{\mathrm{new}} = (\mathbf{I}_{n_u} - \kappa\delta\mathbf{u}\delta\mathbf{u}^T)\boldsymbol{\Sigma}_{\mathrm{old}}(\mathbf{I}_{n_u} - \kappa\delta\mathbf{u}\delta\mathbf{u}^T) + \kappa^2\sigma_{\mathrm{E}}^2\delta\mathbf{u}\delta\mathbf{u}^T. \tag{4.2.5}$$

The variance of the new gradient estimate in the $\delta\mathbf{u}$ direction is $var\{\nabla\phi_{\mathrm{new}}\delta\mathbf{u}\} = \delta\mathbf{u}^T\boldsymbol{\Sigma}_{\mathrm{new}}\delta\mathbf{u}$. The optimal value of κ is given by the following theorem.

Proposition 4.2.1 (Optimal Weighted Broyden Update)**.** *The value of κ that minimizes*

the variance of the gradient estimate in the $\delta\mathbf{u}$ direction is:

$$\kappa = \frac{\delta\mathbf{u}^T\boldsymbol{\Sigma}_{\text{old}}\delta\mathbf{u}}{\delta\mathbf{u}^T\boldsymbol{\Sigma}_{\text{old}}\delta\mathbf{u} + \sigma_E^2} \tag{4.2.6}$$

Proof. The variance of the new gradient estimate in the $\delta\mathbf{u}$ direction is:

$$\delta\mathbf{u}^T\boldsymbol{\Sigma}_{\text{new}}\delta\mathbf{u} = (1-\kappa)^2\delta\mathbf{u}^T\boldsymbol{\Sigma}_{\text{old}}\delta\mathbf{u} + \kappa^2\sigma_{\text{E}}^2. \tag{4.2.7}$$

By differentiating the expression with respect to κ, it follows that the value of k given in Equation (4.2.6) minimizes this variance. □

If the nominal model gradient is used as the initial gradient estimate, the following algorithm is obtained (note that it is similar for the constraint gradient estimates):

Algorithm 4.2.1: Algorithm: Iterative weighted-Broyden-update gradient estimator

---

Initialize: Initialize $\nabla\phi_{\text{old}}$ and $\boldsymbol{\Sigma}_{\text{old}}$ with the model gradient $\nabla\phi(\mathbf{u}_k, \boldsymbol{\theta}_0)$ and the estimated model gradient covariance $\boldsymbol{\Sigma}_0^{\phi}$.

for $\forall\ j$ such that $\|\mathbf{u}_j - \mathbf{u}_k\| < \Delta_{\text{max}}^{\text{r}}$

1. $\delta\mathbf{u} = \frac{\mathbf{u}_j - \mathbf{u}_k}{\|\mathbf{u}_j - \mathbf{u}_k\|}$
2. Compute $\nabla_{\delta\mathbf{u}}\phi_{\text{E}}$ and σ_{E}^2 using Equations (4.2.1) and (4.2.3).
3. Compute κ according to Equation (4.2.6).
4. Compute $\nabla\phi_{\text{new}}$ and $\boldsymbol{\Sigma}_{\text{new}}$ using Equations (4.2.4) and (4.2.5).
5. $\nabla\phi_{\text{old}} = \nabla\phi_{\text{new}}$ and $\boldsymbol{\Sigma}_{\text{old}} = \boldsymbol{\Sigma}_{\text{new}}$.

end

$\nabla\phi_{\text{E},k} = \nabla\phi_{\text{old}}$
$\boldsymbol{\Sigma}_{\text{E},k}^{\phi} = \boldsymbol{\Sigma}_{\text{old}}$

Note that $\Delta_{\text{max}}^{\text{r}}$ ensures that only past measurements sufficiently close to the current RTO point are used for the gradient estimate. This limits truncation error.

4.2.2 Dual Directional-MA Algorithm

The following is the practically applicable RTO algorithm advocated in this chapter. It combines the concepts of directional derivatives, dual control, and statistically optimal gradient estimates with the existing MA technique. The algorithm has two objectives: 1) optimize the real process, 2) ensure the gradient estimate *in the privileged directions* is precise. The idea is to introduce an additional *reward* term into the cost function of the optimization problem to be solved on-line. The reward term encourages the RTO algorithm to move in any of the privileged directions for which only a poor gradient estimate is available.

Algorithm 4.2.2: Algorithm: Dual Directional Modifier Adaptation (Dual D-MA)

Initialize: Choose $\mathbf{U}_\mathrm{r}$ using the method in Section 4.1.2. Choose a positive *reward factor*, c_0, and set the initial reward coefficient $c = 0$. Initialize $\boldsymbol{\epsilon}_0 = \mathbf{0}$, $\boldsymbol{\lambda}_0^g = \mathbf{0}$, $\boldsymbol{\lambda}_0^\phi = \mathbf{0}$. Choose the modifier filter matrices $\mathbf{K}^\epsilon, \mathbf{K}^\mathrm{g}, \mathbf{K}^\phi$ as (typically) diagonal matrices with eigenvalues in the interval $(0, 1]$. Initialize $\mathbf{u}_0$ with a conservative input (one that is unlikely to violate the plant constraints). Select values for Δ_max and $\Delta_\mathrm{max}^\mathrm{r}$. Choose the desired gradient estimate variance in the privileged directions, σ_{TOL}^2 and set $\delta\bar{\mathbf{u}} = \mathbf{0}$.

for $k = 1 \rightarrow \infty$

1. Solve the modified model-based optimization problem

$$\begin{aligned} \mathbf{u}_k := &\underset{\mathbf{u}}{\operatorname{argmin}} \quad \phi_{\mathrm{m},k-1}(\mathbf{u}) \\ \text{s.t.} \quad &\mathbf{g}_{\mathrm{m},k-1}(\mathbf{u}) \leq \mathbf{0}, \qquad (4.2.8) \\ &\|\mathbf{u} - \mathbf{u}_{k-1}\| \leq \Delta_{max}. \qquad (4.2.9) \end{aligned}$$

   where the modified cost and constraints are given by

$$\phi_{\mathrm{m},k}(\mathbf{u}) := \phi(\mathbf{u}, \boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_k^\phi)^T(\mathbf{u} - \mathbf{u}_k) - c|\delta\bar{\mathbf{u}}^T(\mathbf{u} - \mathbf{u}_k)|^2, \qquad (4.2.10)$$

$$\mathbf{g}_{\mathrm{m},k}(\mathbf{u}) := \mathbf{g}(\mathbf{u}, \boldsymbol{\theta}_0) + \boldsymbol{\epsilon}_k + (\boldsymbol{\lambda}_k^\mathrm{g})^T(\mathbf{u} - \mathbf{u}_k). \qquad (4.2.11)$$

   The last term in the modified cost function is the aforementioned reward term. It rewards steps in the direction $\delta\bar{\mathbf{u}}$, which is decided in step 4.

2. Apply the input $\mathbf{u}_k$ to the plant to obtain $\tilde{\phi}_\mathrm{p}(\mathbf{u}_k)$ and $\tilde{\mathbf{g}}_\mathrm{p}(\mathbf{u}_k)$.

3. Use the gradient estimation algorithm in Section 4.2.1 to compute, from the previous RTO measurements, the cost gradient estimate at the current operating point $\nabla\phi_{\mathrm{E},k}$, and the estimate of the gradient of each constraint $\nabla g_{i,\mathrm{E},k}$. The

algorithm will also calculate the variance of the cost gradient estimate $\boldsymbol{\Sigma}^{\phi}_{\mathrm{E},k}$ and the variance of *each* constraint gradient estimate $\boldsymbol{\Sigma}^{g_i}_{\mathrm{E},k}$, $\forall\ i = 1,\ldots,n_g$.

4. Get the direction in the column space of $\mathbf{U}_\mathrm{r}$ that maximizes the estimated variance of the Lagrangian[1]:

$$\begin{aligned} \bar{\delta\mathbf{u}} \in\ & \arg\max_{\delta\mathbf{u}}\ \delta\mathbf{u}^T\boldsymbol{\Sigma}^{L}_{\mathrm{E},k}\delta\mathbf{u} \\ s.t.\quad & \|\delta\mathbf{u}\| = 1, \\ & \delta\mathbf{u} \in C(\mathbf{U}_\mathrm{r}), \end{aligned} \tag{4.2.12}$$

   where $\boldsymbol{\Sigma}^{L}_{\mathrm{E},k} = \left(\boldsymbol{\Sigma}^{\phi}_{\mathrm{E},k} + \sum_{i=1}^{n_g} \nu_i \boldsymbol{\Sigma}^{g_i}_{\mathrm{E},k}\right)$ is the variance matrix of the Lagrangian gradient estimate ($\boldsymbol{\nu}$ is the Lagrange multiplier obtained in Step 1).

5. **if** $\bar{\delta\mathbf{u}}^T\boldsymbol{\Sigma}^{L}_{\mathrm{E},k}\bar{\delta\mathbf{u}} > \sigma^2_{TOL}$
   $c = c_0$
   else
   $c = 0$
   end

6. Calculate the modifier terms for the next iteration according to Equations (2.3.7), (2.3.9) and (2.3.8).

end

Essentially the algorithm proceeds in the same manner as standard MA but uses the novel gradient estimation technique. However, if the accuracy of the gradient estimate in the privileged directions does not satisfy the required tolerance, a quadratic reward term is added to the model-based optimization problem to encourage the RTO algorithm to move in the direction that will most improve the gradient estimate. This is different to past dual MA approaches that used constraints to enforce sufficient excitation. While constraints are often approximated by additional cost terms in the field of optimization, the distinction is particularly important here, as, in our experience, excitation constraints can result in an infeasible optimization problem. Section 4.3.3 illustrates how the algorithm parameters can be chosen in a methodological fashion.

[1] Note that the solution to problem (4.2.12) is the (normalized) dominant eigenvector of $\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^T\boldsymbol{\Sigma}^{L}_{\mathrm{E},k-1}\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^T$.

4.3 Simulated Case Study: Large-Scale Power Kite

Chapter 1 discussed the motivations for optimally controlling a power-producing kite during dynamic flight. While an approximate optimal path can be calculated off-line using a simplified model, the problem of determining the optimal path for the real kite in real time is still an open problem. This section applies Dual D-MA to a simulated kite system, showing not only that the method could efficiently address this problem, but also that Dual D-MA can rapidly optimize an uncertain system with a large number of inputs.

4.3.1 Plant Description

The system is a large kite on a fixed-length tether. The objective is to maximize the average line tension by adjusting the repetitive path flown by the kite.

The (simulated) plant is described by the Erhard Model given in Section 2.4. The kite's lift-to-drag ratio, E, is modeled using the following law:

$$E = E_0 - c\delta^2. \tag{4.3.1}$$

where E_0 is the lift/drag ratio when $\delta = 0$, and c is the turning penalty factor. This law will be further justified in Chapter 5. The plant parameters are given in Table 4.1. They are similar to those of a number of Airborne Wind Energy prototypes currently under development (Ruiterkamp and Sieberling, 2013; Fritz, 2013; van der Vlugt et al., 2013). For plotting purposes in this chapter, the kite position is projected onto the $\{N, W\}$ plane defined in Section 2.4, with $\bar{\vartheta} = 0.3$ radians.

Table 4.1: Plant and model parameter values. The uncertain model parameters θ are highlighted.

Parameter	Plant value	Nominal model value	Unit
r	250	250	m
A	25	25	m^2
ρ	1.2	1.2	$\mathrm{kg}\cdot\mathrm{m}^{-3}$
E_0	6	4.5	-
g_s	5×10^{-3}	7×10^{-3}	$\mathrm{rad}\cdot\mathrm{m}^{-2}$
c	.06	.02	m^{-2}
z_{ref}	10	10	m
w_{ref}	8	8	$\mathrm{m}\cdot\mathrm{s}^{-1}$
a	.15		-
Δw		1×10^{-3}	s^{-1}

As the kites used for power generation are highly unstable, a controller must continuously adjust the steering deflection δ to ensure the kite does not crash. For the purpose of this simulation study, we assume that a 'perfect' path-following controller ensures that the kite follows a periodic reference path, $\{\vartheta_\mathrm{r}(l), \varphi_\mathrm{r}(l)\},\ l \in [0,1]$, where l is the normalized path length. This allows performance optimization to be focused on, without control errors biasing the results. The optimization variable is the reference path to be chosen. The aim is to maximize the average thrust, $\bar{T}$, obtained by following the reference path:

$$\bar{T} := \frac{1}{t_\mathrm{f} - t_0} \int_{t_0}^{t_\mathrm{f}} T \, dt, \tag{4.3.2}$$

where t_0 and t_f are the initial and final times for one cycle of the path. The average thrust $\bar{T}$ depends on the periodic reference path, which is a continuous *function* of the path length. Hence, this is in fact an optimal control problem, which must be discretized to apply RTO. To this end, the RTO decision variables are chosen as a finite set of points on the reference path:

$$\mathbf{u} = \left[\vartheta_\mathrm{r}(0) \quad \varphi_\mathrm{r}(0) \quad \vartheta_\mathrm{r}(\tfrac{1}{N}) \quad \varphi_\mathrm{r}(\tfrac{1}{N}) \quad \vartheta_\mathrm{r}(\tfrac{2}{N}) \quad \varphi_\mathrm{r}(\tfrac{2}{N}) \cdots \vartheta_\mathrm{r}(\tfrac{N-1}{N}) \quad \varphi_\mathrm{r}(\tfrac{N-1}{N})\right]^T, \tag{4.3.3}$$

where $N = n_u/2$ (for this simulation study $n_u = 40$ is used). The kite must respect a height constraint $z(l) := r \sin(\vartheta(l)) \cos\left(\phi(l)\right) \geq z_\mathrm{min}$ and a maximum steering-deflection constraint $|\delta(l)| \leq \delta_\mathrm{max}$, *at every point on the path.* These constraints are also discretized:

$$\mathbf{g}_z = \begin{bmatrix} 1 - z(0)/z_\mathrm{min} \\ 1 - z(\frac{1}{N})/z_\mathrm{min} \\ 1 - z(\frac{2}{N})/z_\mathrm{min} \\ \vdots \\ 1 - z\left(\frac{N-1}{N}\right)/z_\mathrm{min} \end{bmatrix}, \qquad \mathbf{g}_\delta = \begin{bmatrix} |\delta(0)|/\delta_\mathrm{max} - 1 \\ |\delta(\frac{1}{N})|/\delta_\mathrm{max} - 1 \\ |\delta(\frac{2}{N})|/\delta_\mathrm{max} - 1 \\ \vdots \\ |\delta\left(\frac{N-1}{N}\right)|/\delta_\mathrm{max} - 1 \end{bmatrix}. \tag{4.3.4}$$

The RTO layer aims to solve the following discretized plant optimization problem:

$$\begin{aligned} \mathbf{u}_\mathrm{p}^* = \underset{\mathbf{u}}{\mathrm{argmin}} \quad & \phi_\mathrm{p}(\mathbf{u}) := -\frac{\bar{T}}{c_T} \\ \text{s.t.} \quad & \mathbf{g}_\mathrm{p}(\mathbf{u}) := \begin{bmatrix} \mathbf{g}_z \\ \mathbf{g}_\delta \end{bmatrix} \leq \mathbf{0}, \end{aligned} \tag{4.3.5}$$

where $c_T = \left(\frac{1}{2}\rho A\right) r^2 w_\mathrm{ref}^2$ is a scaling factor to make the cost dimensionless. Note also that the input $\mathbf{u}$ is also dimensionless, as the spherical co-ordinates for the kite position

are in radians. While it is not explicitly stated in the above formulation, $\bar{T}, \mathbf{g}_z$ and $\mathbf{g}_\delta$ depend on $\mathbf{u}$ through the Erhard Model's dynamic equations. The manner in which these quantities are calculated for a given reference path is described in Appendix B. The parameters of the optimization problem are given in Table 4.2. The cost and constraint measurements are corrupted with about 3 % zero-mean noise.

Table 4.2: Optimization Parameters

Parameter	Value	Unit
$z_{\min}$	12.5	m
$\delta_{\max}$	7.5	m
σ_ϕ	0.2	-
σ_g	.002	-

4.3.2 Model of the Controlled Kite

The available model is also based upon the Erhard Model, however a different wind law is used, given by the simple linear law:

$$w = w_{\text{ref}} + (z - z_{\text{ref}})\Delta w, \tag{4.3.6}$$

where Δw is the rate of change of wind speed with altitude. Regardless of the value of z_{ref} and Δw chosen, this simplistic model cannot account for the plant nonlinear wind profile (i.e. there is structural plant-model mismatch). In addition, the nominal values of the model parameters (given in Table 4.1) are substantially different from the actual plant values (i.e. there is parametric plant-model mismatch).

4.3.3 RTO Design Procedure

The preferred directions $\mathbf{U}_{\text{r}}$ are chosen exactly as described in Section 4.1.2, with the parameter uncertainty intervals given in Table 4.3. The diagonal matrix of singular

Table 4.3: Uncertainty intervals for the uncertain model parameters.

Parameter	Minimum value	Maximum value	Unit
E_0	3	6	-
g_{s}	2×10^{-3}	11×10^{-3}	$\text{rad} \cdot \text{m}^{-2}$
c	.01	.08	m^{-2}
Δw	0	.025	s^{-1}

values $\mathbf{\Sigma}$ in Equation (4.1.37) contains two very dominant singular values (almost 100 times larger than the other singular values). Hence, this analysis reveals that likely parameter variations will overwhelmingly affect the gradient of the Lagrangian in these two directions. As the aim of the gradient modifiers in MA is to reject any error in the Lagrangian gradient (which is justified by more theoretical arguments in Section 4.1.2), $\mathbf{U}_\mathrm{r}$ was duly chosen as the directions (the columns of $\mathbf{U}$ in Equation (4.1.37)) corresponding to the two dominant singular values. The path variations corresponding to the two chosen directions are shown in Figure 4.1. Their 'orthogonality' can be observed as follows: roughly speaking, one variation makes the path fatter and lower, while the other makes it fatter and higher.

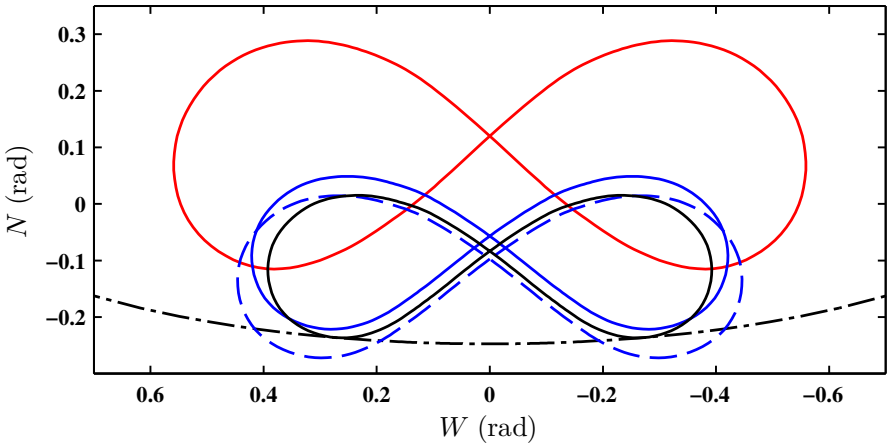

Figure 4.1: Kite optimal paths: path corresponding to $\mathbf{u}_\mathrm{p}^*$ (red); model optimal path corresponding to $\mathbf{u}^*(\boldsymbol{\theta_0})$ (black); path variations produced by steps in the privileged input directions, corresponding to $\mathbf{u}^*(\boldsymbol{\theta_0}) + \Delta_{\max}\mathbf{U}_{\mathrm{r},i}$, for $i = 1$ (dashed blue) and $i = 2$ (solid blue); height constraint (dot-dashed).

Table 4.4: Values of the design parameters for dual D-MA in the kite example.

Parameter	Value
n_r	2
$\Delta_{\max}$	0.03
$\Delta^{\mathrm{r}}_{\max}$	0.06
$\mathbf{\Sigma}_0^{\phi}$	$32^2 \times \mathbf{I}_{n_u}$
$\mathbf{\Sigma}_0^{g}$	$32^2 \times \mathbf{I}_{n_u}$
σ_{TOL}	3.5
c_0	1

The remaining parameters for the dual D-MA algorithm (given in Table 4.4) are chosen by performing a number of mock RTO simulations where the plant is approximated by

the model with different values for the uncertain model parameters. These simulations must generally be carried out to validate the RTO scheme before applying it to the real process. Nonetheless, it is also useful to study the effect of several parameters in a simplified analytic fashion. For example, to see the effect of $\Delta_{\max}$, consider the error when the cost directional derivative is estimated using Equation (4.2.1) and the two points $\mathbf{u}_k$ and $\mathbf{u}_j$. According to Equation (4.2.3) if $\Delta = \|\mathbf{u}_j - \mathbf{u}_k\|$, the standard deviation of the noise error is:

$$\zeta_d = \frac{\sqrt{2}\sigma_\phi}{\Delta}. \tag{4.3.7}$$

Also, the truncation error can be approximated as:

$$\zeta_T = \Delta \cdot H, \tag{4.3.8}$$

where H is the maximum curvature of the model cost function in the space of privileged directions at the nominal optimal solution, that is, the maximum eigenvalue of $\mathbf{U}_\mathrm{r}^+ \nabla^2 \phi(\mathbf{u}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta}_\mathbf{0}) \mathbf{U}_\mathrm{r}$. Figure 4.2 plots these two error terms as functions of Δ. There is a trade-off, namely, too large a value of Δ will result in an unacceptable truncation error, while too small a value of Δ increases the noise error. The maximum step size for the Dual D-MA algorithm was chosen as $\Delta_{\max} = 0.03$, i.e. the point at which the truncation error and the noise error are roughly equivalent. This ensures that, at each iteration, the last step taken by the Dual D-MA algorithm will provide a directional gradient estimate that is not overly contaminated by truncation error. Note that the reward factor c in Equation (4.2.10) will encourage the algorithm to take as large a step as is allowed by $\Delta_{\max}$, which helps reduce the noise error. The radius used to define 'close' points that can be used to estimate the current gradient is chosen as $\Delta^r_{\max} = 2 \times \Delta_{\max}$. Again, this choice is a trade-off, a smaller value of $\Delta^r_{\max}$ means that fewer points can be used by the gradient-estimation algorithm (reducing the quality of the gradient estimate), while a larger value increases the truncation error.

A relatively large value was chosen for the variance of the error affecting the nominal model gradients, $\boldsymbol{\Sigma}_0^\phi = \boldsymbol{\Sigma}_0^g = 32^2 \times \mathbf{I}_{n_u}$, i.e. this is three times the variance (neglecting truncation error) of a derivative calculated using only two points (Figure 4.2). Thus, the dual D-MA algorithm will tend to 'trust' experimental information more than the model.

4.3.4 RTO Results

Figure 4.1 shows that the model optimal solution (calculated with the nominal parameter values) is significantly different from the plant optimal solution, with the optimality

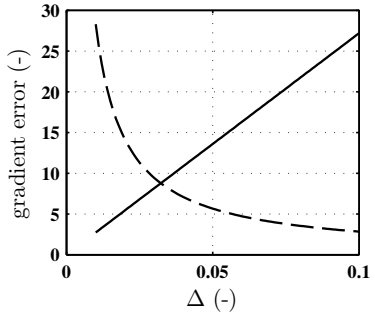

Figure 4.2: Noise error affecting the directional derivative estimate ζ_d (dashed), and truncation error ζ_{T} (solid) as a function of the distance between the points used to estimate Δ.

loss

$$\frac{\phi_{\mathrm{p}}(\mathbf{u}_{\mathrm{p}}^*) - \phi_{\mathrm{p}}(\mathbf{u}^*(\boldsymbol{\theta_0}))}{\phi_{\mathrm{p}}(\mathbf{u}_{\mathrm{p}}^*)} = 29\ \%. \tag{4.3.9}$$

After about 10 iterations, the Dual D-MA algorithm has reduced this optimality loss to about 5 % (Figure 4.3), despite a significant amount of noise. This is very fast, given that the kite takes roughly 15 seconds to complete one cycle of the path, with one RTO iteration per cycle.

As can be seen from Figures 4.4 and 4.5, since the desired gradient accuracy σ_{TOL} is not achieved within 60 iterations, the algorithm continues to take steps in the privileged directions to further improve the gradient estimate. These figures also show that the gradient error calculated in real time is quite accurate.

Figure 4.6 shows that the plant directional derivatives in the privileged directions are driven close to 0. This is particularly true for the $\mathbf{U}_{\mathrm{r},2}$ direction (see Figure 4.1), which is the main direction the algorithm needs to adapt in to reach the plant optimal solution. Hence, the Dual D-MA converges to the vicinity of a *directionally* optimal point for the plant, as predicted by Theorem 4.1.1. What is more, as can be seen from Figure 4.7, Dual D-MA not only achieves near-optimality for the plant, but also converges to the vicinity of the optimal path for the plant.

For the sake of comparison, the algorithm performance with $n_r = n_\theta = 4$ is shown in Figure 4.8. As could be expected, the convergence is slower, as the algorithm must excite the process in more directions (of which all are not necessarily improving directions) to maintain a good estimate of the plant directional derivative. This demonstrates the effectiveness of using the singular-value decomposition given in Equation (4.1.37) to select the privileged directions.

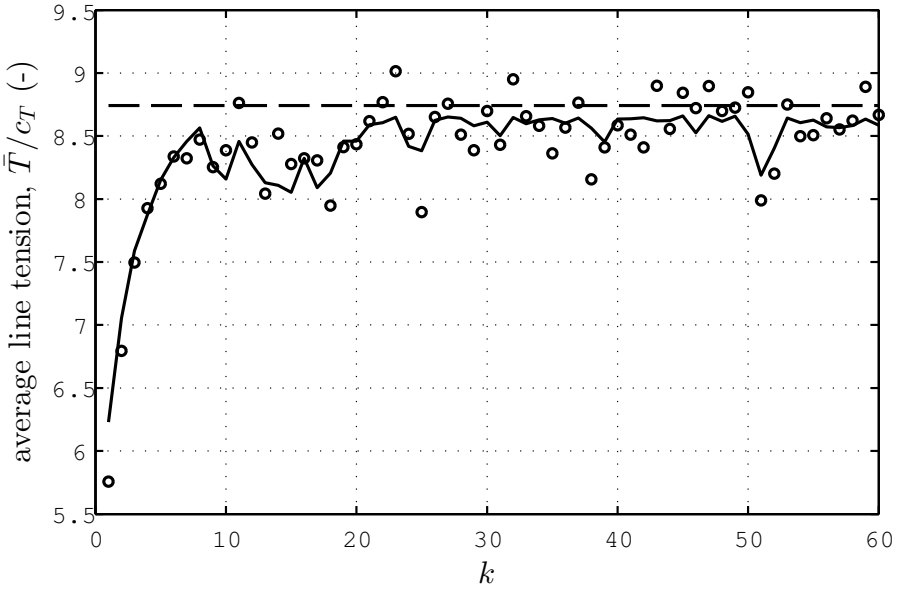

Figure 4.3: True noise-free (solid) and and measured noisy (dots) average line tension (equal to $-\phi_{\mathrm{p}}(\mathbf{u}_k)$ and $-\tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k)$, respectively) as functions of the RTO iteration number k for $n_r = 2$. The plant optimum (equal to $-\phi_{\mathrm{p}}(\mathbf{u}_{\mathrm{p}}^*)$) is also shown (dashed).

4.4 Conclusions

The gradient estimates used in MA represent a very logical diagnostic tool that allows the operator to assess whether the current operating point is optimal for the plant. In addition, if the current point is not optimal, gradient estimates provide an improving direction, and can eventually ensure that an optimal point for the plant is attained. However, for a process with many inputs, standard MA is crippled by the experimental cost of gradient estimation, which is likely to result in slow convergence to the plant optimum.

The solution put forward in this chapter is to estimate *directional derivatives* rather than full gradients. Compared to MA, the resulting D-MA algorithm devotes significantly less effort to gradient estimation, and hence converges much faster. The method, which was proven to guarantee constraint satisfaction and directional optimality upon convergence, has a straightforward design procedure using the available model. Furthermore, a novel way of optimally combining gradient estimates allows the model gradients to be reconciled with experimental data at each RTO iteration. The challenging case study of a dynamically flying power-generating kite has demonstrated rapid convergence to the vicinity of the plant optimal solution, despite significant high-frequency noise and both structural and parametric plant-model mismatch.

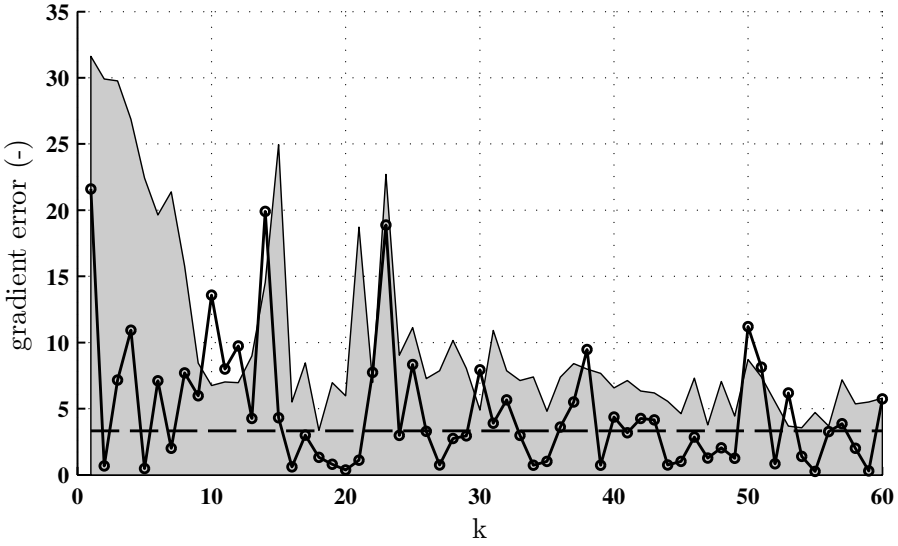

Figure 4.4: Gradient estimation error in the first privileged direction $|\nabla_{\mathbf{U}_{\mathrm{r},1}}\phi_{\mathrm{E},k} - \nabla_{\mathbf{U}_{\mathrm{r},1}}\phi_{\mathrm{p}}(\mathbf{u}_k)|$ (solid), with its standard deviation $\sqrt{\mathbf{U}_{\mathrm{r},1}^T \boldsymbol{\Sigma}_{\mathrm{E},k}^{\phi} \mathbf{U}_{\mathrm{r},1}}$ calculated online (shaded), along with the desired threshold value σ_{TOL} (dashed).

In summary, D-MA is specifically tailored to complex processes with many inputs, for which an approximate model containing a number of uncertain parameters is available. Can an the algorithm handle real-life conditions? The answer is given in the next chapter, which treats the application of the algorithm to an experimental kite-system.

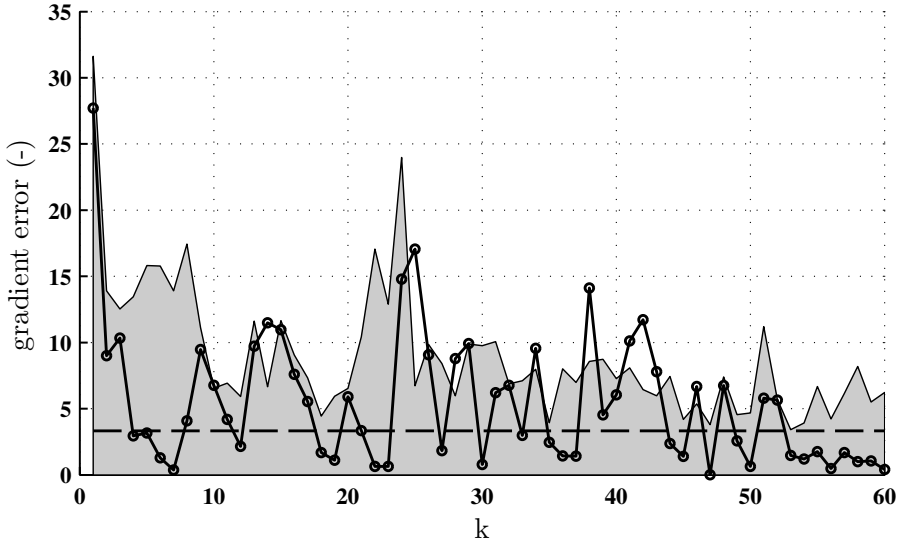

Figure 4.5: Gradient estimation error in the first privileged direction $|\nabla_{\mathbf{U}_{\mathrm{r},2}}\phi_{\mathrm{E},k} - \nabla_{\mathbf{U}_{\mathrm{r},2}}\phi_{\mathrm{p}}(\mathbf{u}_k)|$ (solid), with its standard deviation $\sqrt{\mathbf{U}_{\mathrm{r},2}^T\boldsymbol{\Sigma}_{\mathrm{E},k}^{\phi}\mathbf{U}_{\mathrm{r},2}}$ calculated online (shaded), along with the desired threshold value σ_{TOL} (dashed).

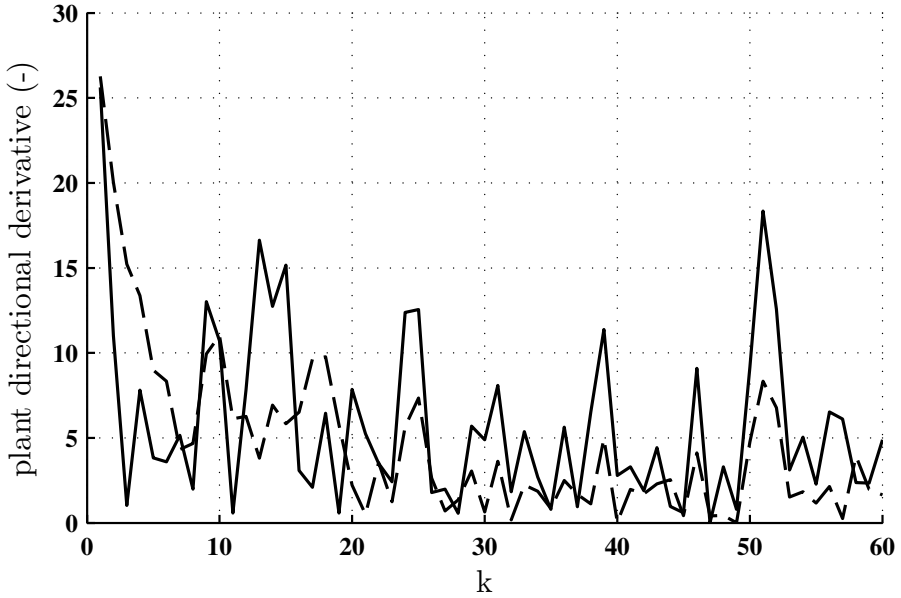

Figure 4.6: Directional derivatives for the plant cost $\nabla_{\mathbf{U}_{r,i}}\phi_{\mathrm{p}}(\mathbf{u}_k)$ for $i = 1$ (solid) and $i = 2$ (dashed) as functions of the RTO iteration number.

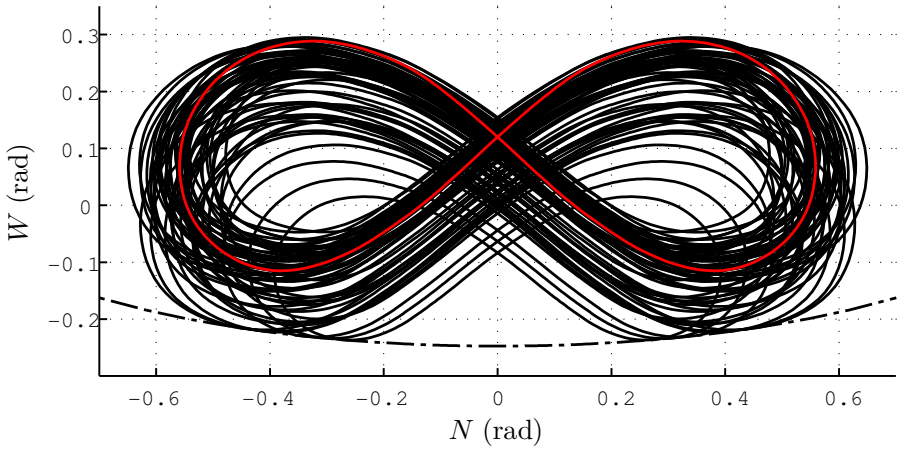

Figure 4.7: All the paths corresponding to $\mathbf{u}_k$, $k = 1,\ldots 60$, (black) for $r = 2$, as well as the plant optimal path $\mathbf{u}_k^*$ (red) and the height constraint (dot-dashed).

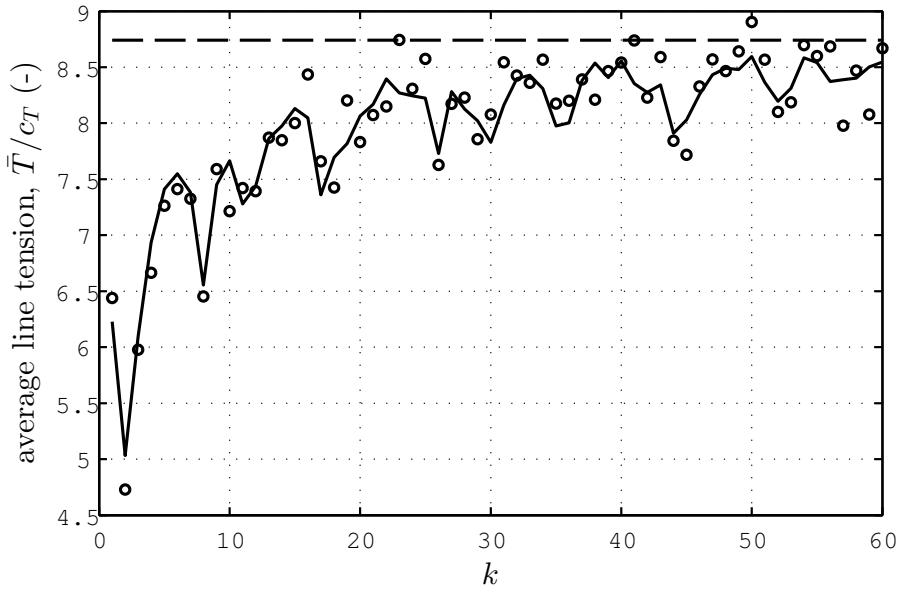

Figure 4.8: True noise-free (solid) and and measured noisy (dots) average line tension (equal to $-\phi_{\mathrm{p}}(\mathbf{u}_k)$ and $-\tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k)$, respectively) as functions of the RTO iteration number k for $n_r = 4$. The plant optimum (equal to $-\phi_{\mathrm{p}}(\mathbf{u}_{\mathrm{p}}^*)$) is also shown (dashed).

5 Application to a Small-Scale Experimental Kite Prototype

As recently as 2012, there were virtually no experimental results on modeling and control for kites. Thus, with the help of several students, I constructed a small-scale experimental kite system (in fact the system described here is the second prototype). The system was built to: a) validate modeling hypotheses, b) devise path-following controllers, c) optimize the path using RTO. All three aims were achieved over the course of three years, and the results are presented in this chapter. However, the field of kite control has changed significantly since 2012. Very recently, a number of publications experimentally validated simple laws to describe a kite's turning behavior and proposed experimentally-validated control-strategies for flying figure-of-eight's (Erhard and Strauch, 2013a; Fagiano et al., 2014; Jehle and Schmehl, 2014; Ruiterkamp and Sieberling, 2013). The results given here are presented in the context of these recent developments, as they are still a fair contribution to the current state-of-the art. In particular, a novel modeling law linking steering deflections to a reduction of the kite's lift-to-drag ratio is proposed and experimentally validated. A path-following controller is developed that requires only measurements of the kite's position, in contrast to other approaches which require the kite's attitude and velocity to be measured. This controller is specifically designed to cope with significant time delay, and is demonstrated experimentally to be extremely robust. Finally, the Dual D-MA RTO methodology developed in the previous chapter is successfully implemented. An experimental study on the effect of path shape on line tension confirms that the RTO algorithm reaches the plant optimum.

The chapter is structured as follows: Section 5.1 describes the experimental setup: the hardware, the software and the outdoor testing conditions. Section 5.2 develops a very simple dynamic model of the system, explains how the model states are calculated from the measurements, and experimentally identifies the model parameters. Section 5.3 presents a path-following controller for kites, encompassing an adaptive prediction

algorithm to compensate for delay, and a 'vector-field' path-following control algorithm specifically adapted for kites. Finally, Section 5.4 describes the experimental implementation of Dual D-MA, with particular emphasis on mitigating the effect of noise. Note that all graphs in this chapter display experimental data.

5.1 Experimental Setup: Small-Scale Kite Prototype

5.1.1 Motivation

A number of different control and optimization problems exist in kite power, just as there are several different kite-power concepts. The aim during this work was to develop generally applicable dynamic models, controllers and RTO strategies. The experimental setup was chosen to be a small kite on a fixed-length line, with a control objective of maximizing the average line tension. The justification was the following:

- Testing with the small-scale prototype can be carried out alone over the course of a day, in any small field. The number of potential test-sites within an hour's drive of EPFL are significant, including a number of beaches, which typically offer the steadiest winds (preferable for modeling experiments) in onshore conditions. In crosswind flight the kite's front line is straight as, due to its short length, it experiences little drag and its mass is negligible compared to the line tension. This is exploited to measure, from the ground, the kite's position with great accuracy, which further allows the kite's velocity and orientation to be estimated. In this manner, the delicate problem of placing sensors on the kite is avoided. The negligible line drag allows the kite's pitch and the steering deflection it experiences to be precisely controlled.
- Small kites with an almost identical design to those used for power generation are available (it is rather the power-generating kites that copied the small kites). As the Reynolds number of both a 3-m^2 kite and a 300-m^2 kite in crosswind flight ensures turbulent flow (Dadd et al., 2011), there are unlikely to be significant differences between their aerodynamic properties, and hence their behavior.
- Fixed line-length flying is perfectly sufficient to develop dynamic kite-models for crosswind flight. The kite's behavior is studied with respect to the apparent wind (the wind it experiences). By flying the kite on a fixed-length tether in a variety of wind speeds, the kite's behavior for a full range of apparent wind speeds and angles can be studied. For the same reason, a control algorithm that handles a kite on a fixed-length tether in a variety of wind conditions, could also be used for the traction phase of a pumping-cycle kite-power generator.

- As for the question of optimal paths, while there is no strict equivalence between the problems of maximizing the average line-tension for a fixed-line kite, and maximizing the component of the average line-tension in a particular direction (the objective with kites on boats (Skysails GmBH)), or maximizing the product of the line-tension and the reel-out speed (the objective during the traction phase of pumping-cycle generation), I have studied the solutions to these problems in detail (partly published in (Costello et al., 2013)) and the optimal solutions tend to be very similar. It can be expected that a RTO algorithm that works well for the average line-force maximization problem could be applied to the other problems with some adaptation.

The experimental system is designed to be complementary to the Swiss Kite Power group's large-scale testing platform. The size and complexity of the large-scale platform gives birth to additional engineering and human challenges. Testing with the large platform involves co-ordinating several people and transporting a lot of machinery to a very large, deserted test-site. Switzerland has very few such sites; most tests take place near the summit of the Chasseral in the Jura mountain range. The wind here is relatively gusty (irregular), and snow prevents testing for half of the year. Accurately measuring the kite's position, velocity and orientation is complicated by the long line-lengths, requiring (fragile) on-board sensors with their own power supply, and communication to a ground station. Actuation of the kite cannot always be precisely controlled due to the significant curvature in the long lines. All these issues (which are closely linked to design decisions, and hence platform dependent) must be addressed in a full-scale system. Using a small-scale system allows us to bypass these problems, and focus exclusively on dynamic modeling, control and path optimization for the kite.

5.1.2 Physical System

The ground station and the kite are shown in Figures 5.1 and 5.2.

Two standard commercial power kites were used: A 2.5-m^2 Flysurfer Viron and a 3.5-m^2 HQ Apex. Both are *three-line*[1] kites; one front line takes about 90% of the force generated by the kite, the two lightly-tensioned rear lines allow the kite to be maneuvered. There are two degrees of freedom to operate the kite: a) adjusting the *difference* between the lengths of the rear lines allows the kite to be steered left or right, b) adjusting the length of the front line allows the kite to be accelerated or decelerated by changing its angle-of-attack relative to the onrushing air. In this case, only the steering degree of

[1] This type of kite is often termed a 4-line kite also, as there are 4 attachment points on the kite. However, the two front lines join together, and there are only 3 attachment points on the control bar.

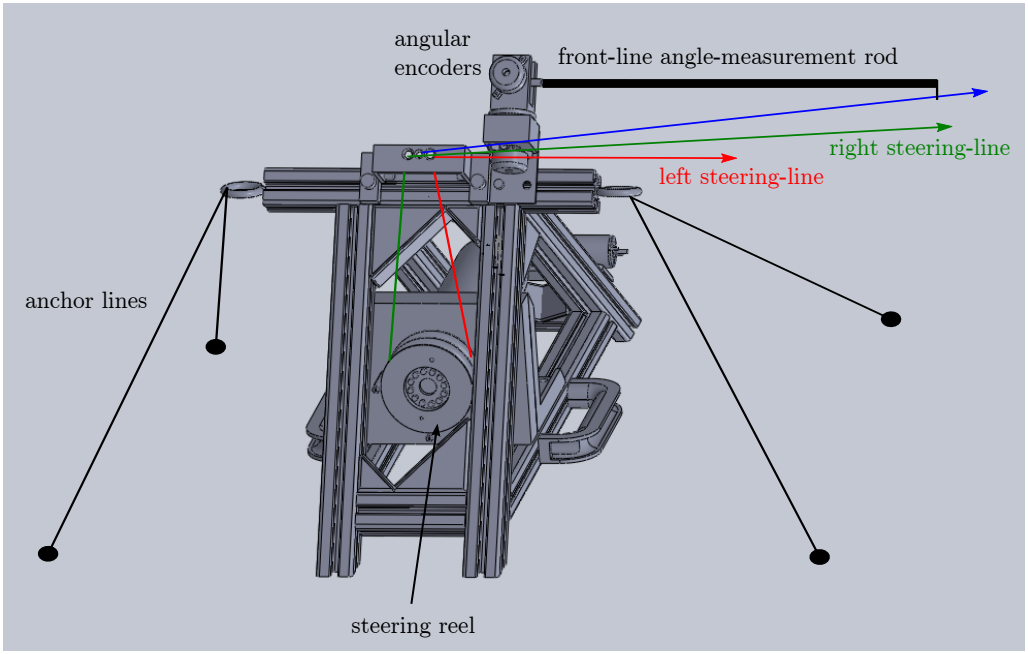

Figure 5.1: The ground station.

freedom is adjusted automatically. The length of the front line is maintained constant throughout each experiment. Unless the wind is extremely light, 3-line kites fly very well with the length of the front line kept constant.

The ground station is a wedge-shaped structure built from aluminium profiles. The lines coming from the kite are lead through three small eyes placed closely together. This ensures that as the kite moves around, and hence the angle of the lines changes, the relative lengths of the lines will not vary. The front line passes through an adjustable blocking device (a block followed by a clam cleat) which is attached to a load cell for measuring the tension in the front line. In this manner the length of the front line can be adjusted at the beginning of each experiment to suit the wind conditions and the geometry of the kite being tested. The angle of the front line between the station and the kite is measured by a 1-m long carbon-fiber rod with a small ring at the end, through which the front line passes. The angle of the rod, from which the line angle can be inferred, is measured by two incremental rotary encoders. The rear lines are wound *in opposite directions* around a reel, which is turned by a 400-Watt DC motor. Rotating the reel shortens one line, while lengthening the other. The interior of the structure houses the electronics for the motor, the load cell and the encoders. The total weight of the ground station is 30kg, allowing it to be displaced by a single person. As the ground station is very light relative to the load generated by the kite (up to 300 kg), it must be securely anchored to the ground. Four purpose-built steel ground screws

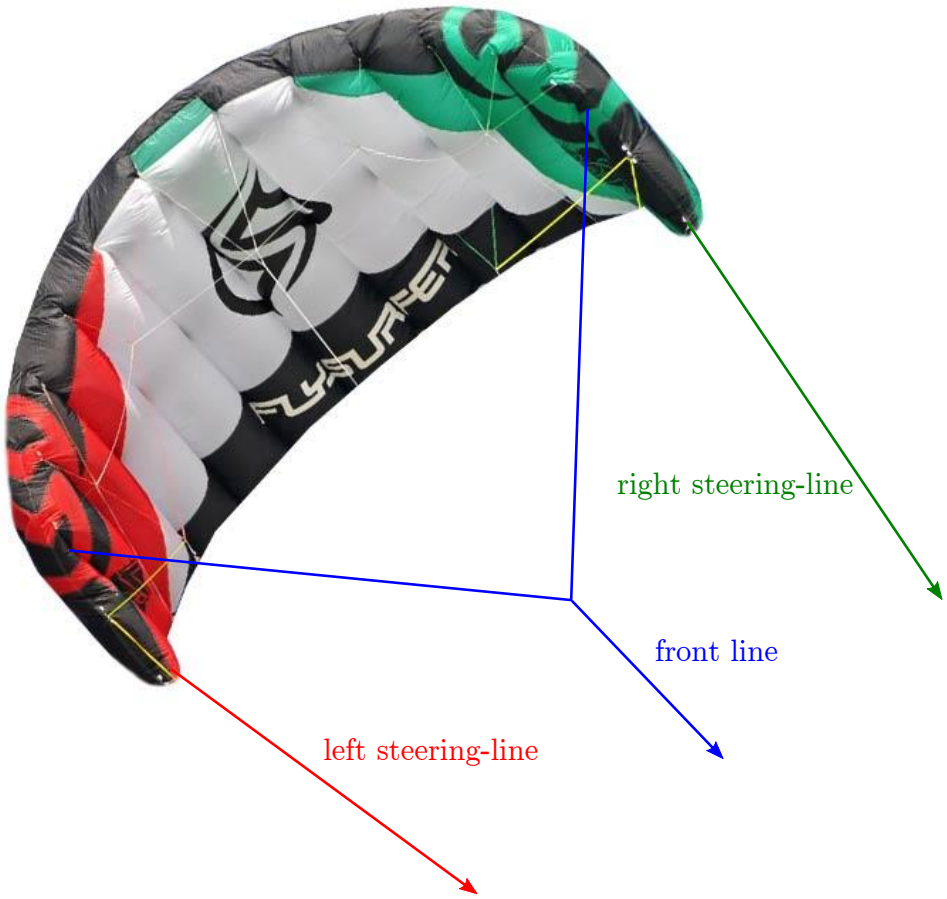

Figure 5.2: The kite (a Flysurfer Viron).

inserted 50 cm into the ground and a system of ratchet tie-downs are used. A number of additional items make up the experimental platform. A high-precision ultrasonic anemometer mounted on a 3 m pole measures the wind speed and direction. A laptop computer is connected (interfaced by appropriate electronics) to each measurement device and to the motor-control unit. The different devices are powered by a 220-V AC petrol generator, a 70-V DC power supply and a 5-V DC power supply.

The system is designed for loads of up to 200 kg. For the given kite sizes (which are the smallest 3-line power kites on the market), this allows testing in up to 15 $\mathrm{m \cdot s^{-1}}$ wind. Nonetheless, all the components are over-dimensioned and the station has survived peak loads in excess of 300 kg.

The DC motor actuating the rear lines can adjust the difference in line length very rapidly (at a nominal rate of 2 $\mathrm{m \cdot s^{-1}}$ with a tension difference of 20 kg, and several times faster for short periods of time). Position control is handled by a PID algorithm running on a dedicated micro-controller with a sampling frequency of 10 kHz. An incremental encoder attached to the motor's shaft measures the motor's position. The reel's position is inferred from this, with extremely high accuracy due to the motor's high gearbox ratio. The powerful motor means the position control loop is very fast, with a settling time of around 30 ms (this varies slightly depending on the kite being used and the wind strength).

The accuracy of the various sensors is paramount for an autonomous system. In theory, the encoders measuring the angle of front line are accurate enough to provide the kite's position to within ±5 cm, if 35 m lines are used. Furthermore, it is estimated that the slight curvature of the front line induced by aerodynamic drag is likely to introduce and error of at most ±10 cm. The load-cell measures the front-line tension with an accuracy of ±0.2 N. Given that the front line tension is typically at least 200 N, this error of ±0.1 % is negligible.

5.1.3 Software

Software running on the laptop reads and logs the sensor signals and calculates a position set point for the motor controller. The four main processes running *in parallel* are shown in Figure 5.3 (this is a simplification, as in reality a number of additional processes handle data retrieval from the anemometer, the load cell, and the joystick). Note, that a parallel division of tasks is essential, as a) different tasks must execute at different frequencies, and b) this way critical processes, such as the autopilot, do not get held up by other tasks.

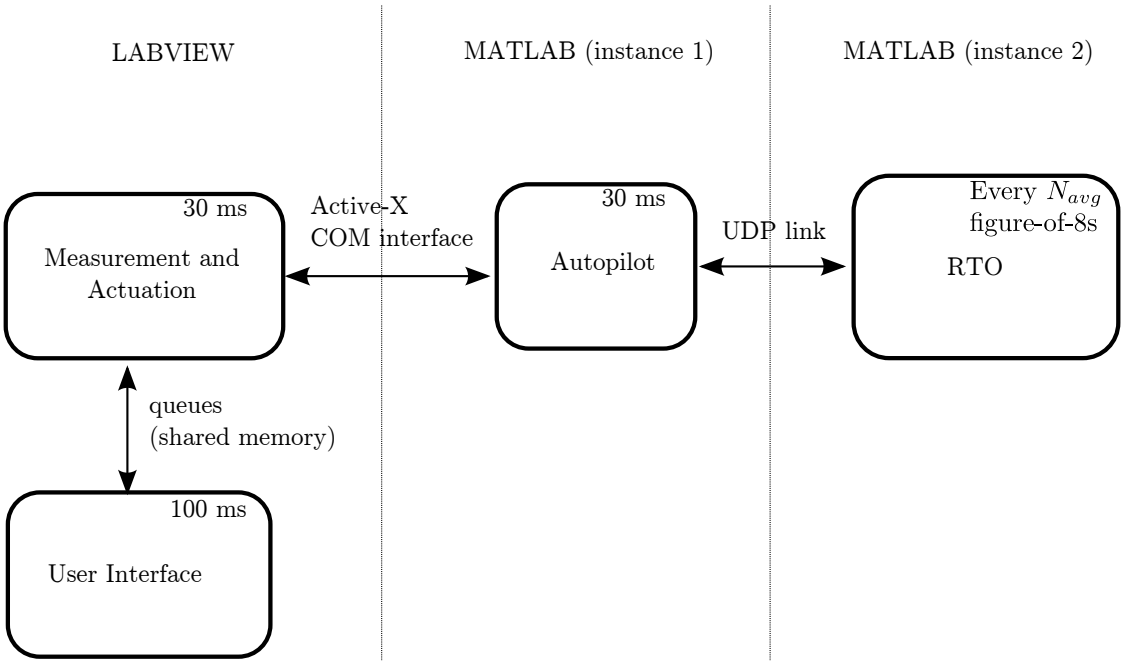

Figure 5.3: The *parallelized* structure of the software running on the laptop. Each box is a process, whose execution period is indicated in the upper right-hand corner. Arrows signify inter-process communication, and are annotated by the communication method.

- **User Interface** reads any user input from either the joystick or the keyboard, and graphs measurement data and other useful signals. With its time-consuming graphics to display, and given that it is interacting with a (relatively slow) human, this process is executed relatively infrequently.

- **Measurement and Actuation** updates the manipulated variable, namely the set point for the motor's position. At each iteration the sensors are read, state estimation is performed, and the system states are sent to the *Autopilot* process, which returns a desired steering input. This desired steering input is translated into a position set point for the motor and is sent to the motor controller. Alternatively, if the system is in 'manual' mode, the (filtered) joystick position is used to decide the motor's position set point. Finally, if the option to save data is activated, the current values of all measurements, state estimates and controller parameters are saved to file (in reality this data is placed in memory, and transfered to a file once the option to save data is deactivated). The execution frequency of this process is 33 Hz, roughly an order of magnitude larger than the bandwidth of the kite's states during flight.

- **Autopilot** embodies the cascade control algorithm to be described in Section 5.3. Based on the kite's current states, this algorithm calculates the necessary steering input in order for the kite to follow a reference path. This process also keeps track of the average values of several signals over the past N_{avg} repetitions of the reference path that the kite follows. Every N_{avg} repetitions, this data is transmitted to the *RTO* process, which returns a new value of the reference path.
- **RTO** periodically adjusts the reference path in order to optimize the kite's performance using the algorithm to be described in Section 5.4.

One important consequence of the software structure is the introduction of delay. The first such delay is the 'execution delay' τ_{e} due to the execution time of the *Measurement and Actuation* process: on average the updated motor position set point is communicated to the motor controller 20 ms after the beginning of this process. The second software-induced delay affects the RTO algorithm. By the time the *RTO* process has finished executing and sent a reply (a new reference path for the kite) to the *Autopilot* process, the kite has already partly completed another cycle of the previous reference path. The new reference path is only used once the kite has finished this cycle. The result is a delay of 1 cycle affecting the RTO algorithm.

5.1.4 Field Testing

During 2013 and 2014, the experimental system was used for 28 day-long tests, in 8 different locations. A further 7 tests were aborted upon reaching the test location due to insufficient wind. The testing locations along with the wind directions used for the tests, are shown in Figure 5.4. 30 % of the tests were dedicated to equipment troubleshooting and modeling, 50 % were spent on the control algorithm, and 20 % were spent on RTO.

The testing location has a large influence on the control and RTO algorithms' performances, as the wind is strongly influenced by the local terrain. For example, achieving robust control performance in the mountains was more difficult, as the wind strength and direction varied rapidly. The steadiest wind was encountered on lake sides with an onshore wind, however, unfortunately the lake sides in Switzerland are invariably either built-up or forested. Many tests were carried out at Evionnaz (VS), which sits on the floor of the deep Rhone valley, right at a narrowing. This narrowing results in a Venturi effect, producing strong thermal winds almost every second day in summer. Unfortunately, the wind here is extremely gusty (meaning it varies significantly over short periods of time, of the order of several minutes, or even tens-of-seconds). The most recent tests, accounting for the majority of the results presented here, were carried out on the plain above Lac Léman, either in the communes of Penthaz (VD) or Echichens (VD), during

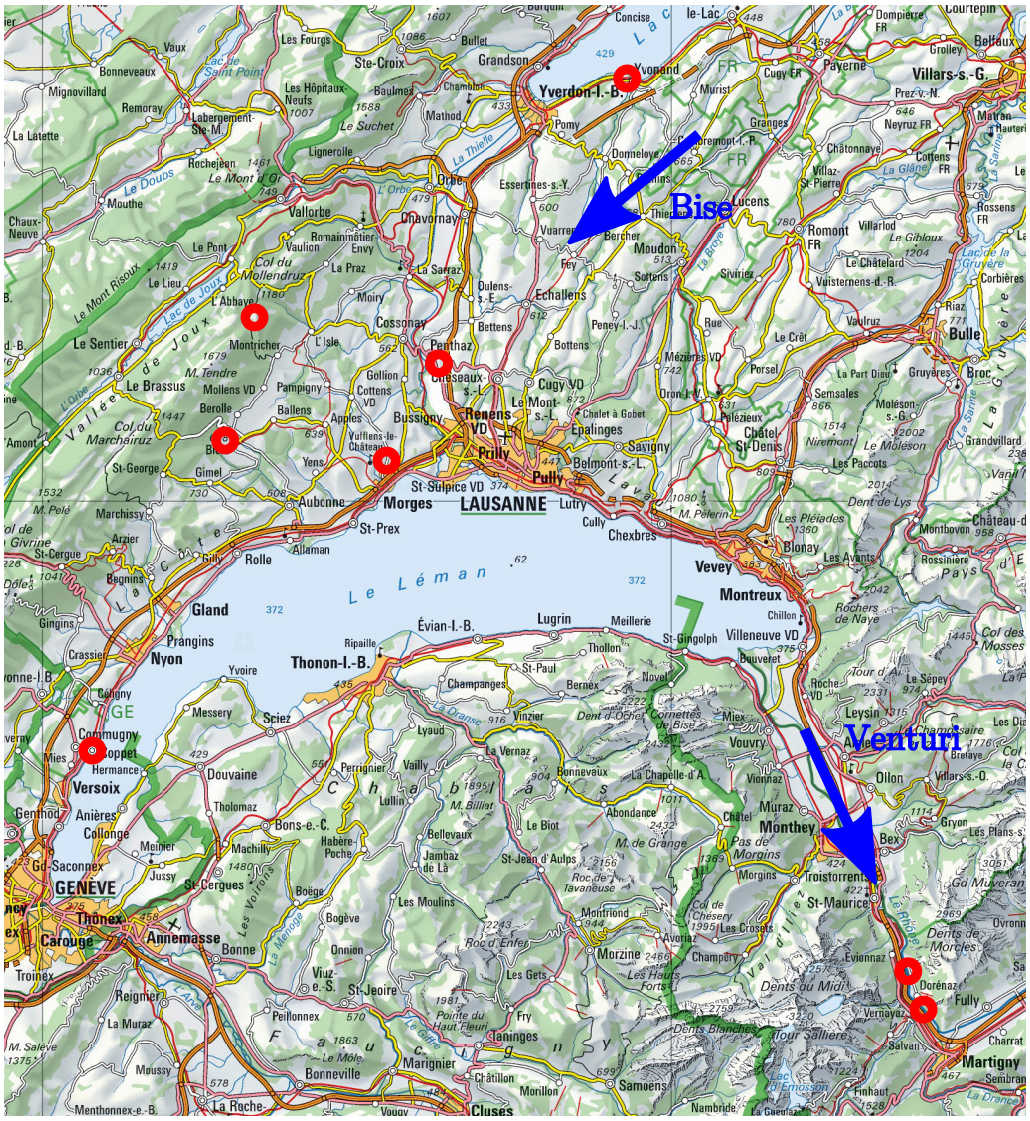

Figure 5.4: Testing locations (red circles), and the two winds used for testing.

episodes of north-easterly wind ('la Bise'). In both cases the tests were carried out in flat fields, with virtually no obstructions to windward for at least 500 m. The gustiness of the wind experienced by the kite in this case is medium: as it is coming from the land it is far more gusty than the wind experienced at sea or along a coastline, however it is steadier than the wind to be found in the mountains.

5.2 Modeling and State Reconstruction

In order to employ the Erhard model introduced in Section 2.4, it is necessary to a) measure (or estimate) the states used by this model in real time, b) experimentally identify the free model parameters for the system. The states used in this model are the *position* and the *orientation* of the kite. Although the position of the kite is measured, our experimental setup does not measure the orientation of the kite (this would require an on-board inertial sensor). From the model equations it would seem essential to measure the kite's orientation accurately, as this is the only state that can be directly influenced by (i.e. whose first derivative depends upon) the steering deflection. Indeed, the control algorithm proposed by Erhard and Strauch (2013a) uses the kite's orientation as the basic controlled variable. While the kite's orientation is no doubt a useful measurement in any situation, it is not actually essential for control during crosswind flight. During crosswind flight the kite is moving rapidly, and the aim is to control the kite's *direction of motion*, rather than its orientation. Thus we define an alternative state variable, the *velocity angle*, which is closely related to the kite's orientation. In fact, we will show that under normal conditions, the kite's orientation is approximately the same as its velocity angle. In addition, the resulting model is even simpler than that proposed by Erhard and Strauch (2013a). The velocity angle can be inferred from a sequence of past position measurements, albeit at the cost of introducing delay.

5.2.1 The Velocity Angle

The assumptions upon which the model in Section 2.4 is based are only valid if the kite is flying roughly crosswind (when $\frac{\tan\vartheta}{E} << 1$). During crosswind flight the speed of the kite is several times that of the wind speed. We will now show that assuming crosswind flight allows the model to be further simplified. We begin by introducing the velocity angle:

$$\gamma = \tan^{-1}\left(\frac{\dot{\varphi}\sin\vartheta}{\dot{\vartheta}}\right). \tag{5.2.1}$$

This is the angle of the kite's velocity in the plane that is tangent to the sphere at the kite's current position. Developing the expression for the velocity angle (using Equations 2.4.2 and 2.4.3) gives:

$$\gamma = \tan^{-1}\left(\frac{-\sin\psi}{\cos\psi - \frac{\tan\vartheta}{E}}\right) \tag{5.2.2}$$

As $\frac{\tan\vartheta}{E} << 1$ in crosswind flight, we have that:

$$\gamma \simeq -\psi. \tag{5.2.3}$$

In other words, we have shown that, not surprisingly, the kite flies crosswind in roughly the same direction as it is pointing. Next we examine the kite's speed $\|\dot{\mathbf{p}}\|$ when traveling crosswind (we recall that $\mathbf{p}$ is the kite's position in x, y, z coordinates, defined by Equation 2.4.1) . The kite's speed relative to the tether length (i.e. in $\text{rad}\cdot\text{s}^{-1}$) is:

$$\omega_k := \frac{\|\dot{\mathbf{p}}\|}{r} = \sqrt{\left(\dot{\varphi}\sin\vartheta\right)^2 + \left(\dot{\vartheta}\right)^2}. \tag{5.2.4}$$

Now inserting the differential Equations 2.4.2 and 2.4.3 gives:

$$\omega_k = \sqrt{\left(-\frac{w_{ap}}{r}\sin\psi\right)^2 + \left(\frac{w_{ap}}{r}\left(\cos\psi - \frac{\tan\vartheta}{E}\right)\right)^2}, \tag{5.2.5}$$

$$= \frac{w_{ap}}{r}\sqrt{\left(\sin\psi\right)^2 + \left(\cos\psi - \frac{\tan\vartheta}{E}\right)^2}. \tag{5.2.6}$$

Finally, as $\frac{\tan\vartheta}{E} << 1$ in crosswind flight:

$$\omega_k \simeq \frac{w_{ap}}{r} = \frac{w}{r}E\cos\vartheta. \tag{5.2.7}$$

Thus, during crosswind flight the kite's speed approximately depends only on the position, and not the orientation, of the kite.

Using the definition of γ (Equation 5.2.1) and the kite's angular velocity (Equation (5.2.4)) we can now write a simplified dynamic model for the kite:

$$\dot{\vartheta} = \omega_k\cos\gamma, \tag{5.2.8}$$

$$\dot{\varphi} = \frac{\omega_k}{\sin\vartheta}\sin\gamma, \tag{5.2.9}$$

$$\dot{\gamma} = -\left(\omega_k r g_s\delta + \dot{\varphi}\cos\vartheta\right), \tag{5.2.10}$$

$$\omega_k = \frac{w}{r}E\cos\vartheta. \tag{5.2.11}$$

This model will henceforth be referred to as the 'Cart Model'. The dynamics are now

exactly those of a cart driving on a sphere, whose speed we cannot directly control. If unlimited steering action is assumed, any path can be achieved, but in reality the steering deflection is bounded. Note that, if a constant E is assumed (which is reasonable if the model is to be used to design steering controllers, but not for tether-force optimization) the input required to follow any (smooth) path $[\vartheta(l), \varphi(l)]$ can be easily computed, although it may not satisfy the maximum-steering-deflection constraint. Also, as $\dot{\varphi}\cos\vartheta$ is usually small relative to $\omega_k r g_s \delta$ in Equation (5.2.10), the turn rate is roughly proportional to the steering deflection. This observation has also been made by Fagiano et al. (2014) and Jehle and Schmehl (2014).

5.2.2 State Reconstruction

The first step towards controlling the system is to be able to reconstruct the states of the Cart Model, $\{\vartheta, \varphi, \gamma\}$, from measurements. It is much easier to reconstruct the states *retrospectively*, after an experiment, using both past and future data to reconstruct the states at a particular time instant. This *non-causal* approach probably yields quite a good estimate of the states, and in this work it will be assumed that the states can be reconstructed perfectly in this fashion. During real-time operation, only current and past measurements are available, making non-causal reconstruction impossible. The result is a less-than-perfect estimate of the states. Luckily, this estimate can be compared with the non-causal reconstruction in order to understand exactly how imperfect it is.

Non-causal reconstruction

The kite's estimated position at each sampling instant, $\hat{\mathbf{p}}[n]$, is calculated from the encoder signals. The position signal is filtered with a non-causal filter with a cut-off frequency of 5 Hz[2]. The logic is that the kite's movement is visually observed to be smooth. The kite, although light, has a certain inertia. Regardless of the steering deflections applied, it cannot change its direction of motion instantaneously. Therefore, any high frequency components present in $\hat{\mathbf{p}}[n]$ must be due to measurement noise, and should be removed by filtering. The resulting non-causal estimate of the kite's position is $\mathbf{p}^{\mathrm{NC}}$, where 'NC' stands for non-causal. Converting the non-causally filtered position to spherical coordinates yields ϑ^{NC} and φ^{NC}.

[2] This is achieved by taking the Discrete Fourier Transform of the signal, multiplying the resulting frequency-domain signal by an appropriately sized Hann window, then moving back into the time domain with the Inverse Discrete Fourier Transform

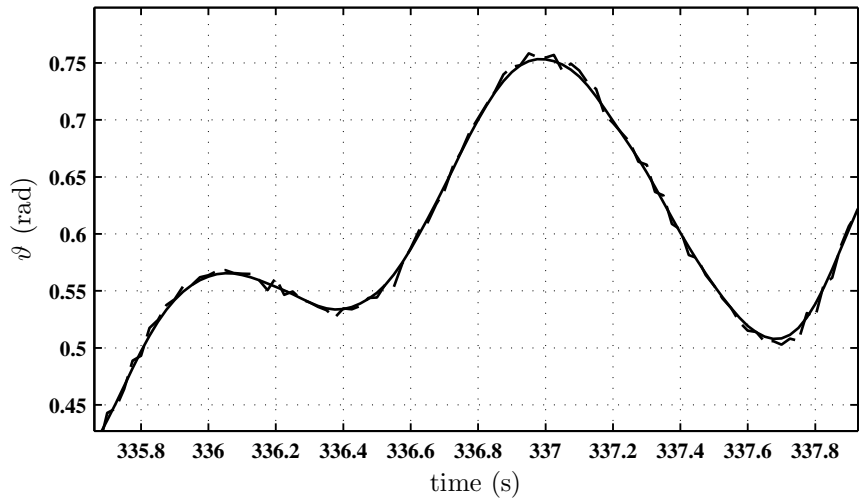

Figure 5.5: The estimate of the kite's spherical position coordinate $\hat{\vartheta}$, inferred directly from measurements in real-time (dashed). The non causal reconstruction ϑ^{NC} (solid).

Applying a centered derivative formula to $\mathbf{p}^{\mathrm{NC}}$ yields:

$$\dot{\mathbf{p}}^{\mathrm{NC}}[n] = \frac{\mathbf{p}^{\mathrm{NC}}[n+1] - \mathbf{p}^{\mathrm{NC}}[n-1]}{2T_{\mathrm{s}}}, \tag{5.2.12}$$

from which we can obtain $\dot{\vartheta}^{\mathrm{NC}}$, $\dot{\varphi}^{\mathrm{NC}}$, and $\omega_k^{\mathrm{NC}} = \frac{\|\dot{\mathbf{p}}^{\mathrm{NC}}\|}{r}$. Note that the notation $[n-i]$ represents a delay of a discrete signal by i sampling periods. The velocity angle can now be calculated as:

$$\gamma^{\mathrm{NC}} = \tan^{-1}\left(\frac{\dot{\varphi}^{\mathrm{NC}} \sin\vartheta^{\mathrm{NC}}}{\dot{\vartheta}^{\mathrm{NC}}}\right). \tag{5.2.13}$$

Applying the centered derivative formula to γ^{NC} yields $\dot{\gamma}^{\mathrm{NC}}$. Finally, $w_{\mathrm{ap}}^{\mathrm{NC}}$ is estimated by projecting $(\dot{\mathbf{p}}^{\mathrm{NC}} - \bar{\mathbf{w}})$ onto the plane orthogonal to $\mathbf{p}^{NC}$, where $\bar{\mathbf{w}}$ is the *average* measured wind vector during the experiment in question.

Real-time reconstruction

The position of the kite is inferred from the angle of the front line, which is measured by the line-angle measurement rod shown in Figure 5.1. As this line is under great tension during crosswind flight, and the length of the line is relatively short, the line is extremely straight. If the only source of error were the very slight curvature of the line due to gravity and aerodynamic drag, then the position of the kite could be calculated with an accuracy of ±15 cm. Unfortunately, a further source of error, vibrations, complicates matters slightly. The spherical position coordinate $\hat{\vartheta}$ reconstructed in real time during

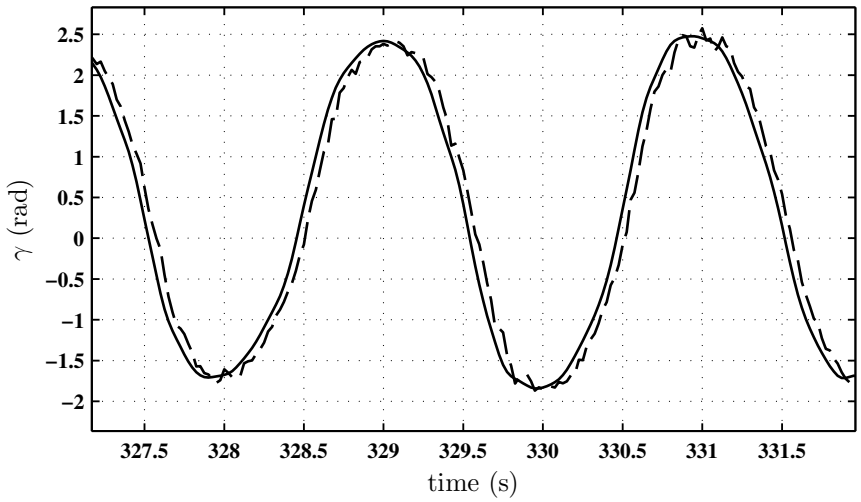

Figure 5.6: The real-time estimate of the kite's velocity $\hat{\gamma}$, calculated according to Equation 5.2.14 (dashed), compared with the non-causal reconstruction γ^{NC} (solid).

an experiment is compared with the non causally reconstructed signal in Figure 5.5. The real-time signal is slightly 'jittery'; this is due to noise caused by vibrations of the kite line and the line-angle measurement rod. The kite's actual position, which is assumed to correspond to the non-causal reconstruction, will vary quite smoothly, due to the kite's inertia. The standard deviation of the 'measurement noise' affecting both $\hat{\vartheta}$ and $\hat{\varphi}$ can be estimated as $\sigma_\vartheta = \sigma_\varphi \simeq .01$ rad. The standard deviation of the resulting error affecting the estimate of the kite's absolute position is ±50 cm.

A very simple method is used to estimate γ from the kite's measured position. At the sampling instant n, an estimate of the kite's velocity in cartesian coordinates, $\widehat{\dot{\mathbf{p}}}$, is obtained by solving the following least-squares problem:

$$\{\widehat{\dot{\mathbf{p}}},\ \hat{\mathbf{a}}\} = \arg\min_{\{\dot{\mathbf{p}},\ \mathbf{a}\}} \sum_{j=0}^{2d_\mathrm{d}} \left\| \left(\mathbf{a} - j\,T_\mathrm{s}\dot{\mathbf{p}}\right) - \hat{\mathbf{p}}[n-j] \right\|^2 \tag{5.2.14}$$

where $\hat{\mathbf{a}}$ is an estimate of the kite's current position which is not used. Solving this problem fits a line to the last $2d_\mathrm{d}+1$ position estimates, and the velocity vector associated with the fit is $\widehat{\dot{\mathbf{p}}}$. Note that the estimate of the kite's velocity obtained in this manner has a delay of $\tau_\mathrm{d} = d_\mathrm{d}\,T_\mathrm{s}$. Given that the bandwidth of the velocity-angle signal is about 3 Hz, the truncation error that occurs from using a linear fit over several points at intervals of 30 ms is relatively small compared to the error induced by the position estimates' measurement noise. The velocity angle is then calculated as:

$$\hat{\gamma} = \tan^{-1}\left(\frac{\hat{\dot{\varphi}}\sin\left(\hat{\vartheta}\right)}{\hat{\dot{\vartheta}}}\right). \tag{5.2.15}$$

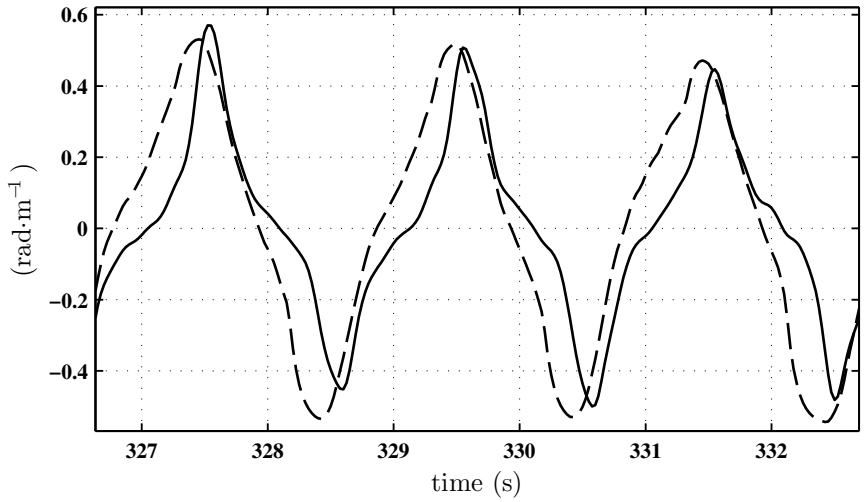

Figure 5.7: The scaled steering input, $\delta \times g_s$ (dashed), with $g_s = 1.1$ rad·m^{-2}, and $-\frac{\dot{\gamma}_c^{NC}}{\omega_k^{NC} r}$ (solid), while the kite flies regular figure-of-eights.

The velocity angle estimated in this manner during an experiment is shown in Fig. 5.6. For comparison, the non-causal reconstruction is also shown. It can be seen that the velocity angle estimated in real time lags the 'ideal' non-causal estimate by about $d_d T_s$ =60 ms, as in this case $d_d = 2$. In addition the signal obtained in real time is contaminated by high-frequency noise. The standard deviation of the noise affecting $\hat{\gamma}$ can be estimated as $\sigma_\gamma = var\{\hat{\gamma} - \gamma\} \simeq .05$ rad.

5.2.3 Experimental Characterization of the Kite's Turning Behavior

Understanding the turning behavior of the kite is clearly fundamental if the kite's path is to be controlled. Let the 'corrected' turning rate be defined as:

$$\dot{\gamma}_c = \dot{\gamma} + \dot{\varphi} \cos\vartheta. \tag{5.2.16}$$

The cart model (Equation (5.2.10)), establishes the following proportional relationship:

$$\frac{\dot{\gamma}_c}{\omega_k r} = -g_s \delta. \tag{5.2.17}$$

Although a demonstration is not given here, note that this implies that the *curvature* of the kite's path is proportional to the steering deflection, δ. The constant g_s can be estimated from experimental data as shown in Figure 5.7, by finding the value of g_s that results in the two signals overlaying each other. Two important phenomena can be observed from this figure. Firstly, $\frac{\dot{\gamma}_c}{\omega_k r}$ is only approximately proportional to the steering

input δ. Secondly, there is a significant delay between a change in the steering deflection and the resultant change in the turning rate. This delay is due to the inertia of the kite, which is neglected in the kinetic cart model, and it shall be referred to as the inertia delay, τ_I.

From a path-optimization viewpoint, it is important to understand the effect turning has upon line tension, as the optimization objective is to maximize line tension. In the Erhard Model, the line tension depends on the angle between the tether and the wind, ϑ, and the lift-to-drag ratio, E, where E is constant. Erhard and Strauch (2013a) demonstrated that this is a reasonable approximation if the model is used for controller design. Indeed, from a control point of view, it is important to precisely control the kite's direction of motion. The lift-to-drag ratio mostly influences the kite's speed (and the ensuing line tension), which is of secondary importance for the control algorithm. However, Equation (2.4.7) from the Erhard Model implies that the line tension depends *only* on the kite's *position*. This equation also implies that there is an optimal position (or positions), which depends on the wind gradient and the model parameters, that maximizes the line tension. If the Erhard Model is to be believed, the kite should be kept close to this optimal position by aggressively steering the kite. However, during experiments, it was observed that steering deflections cause a reduction in tether tension. This is particularly noticeable for very large steering deflections, which will almost cause the kite to stall, drastically reducing the apparent wind speed, and hence the tether tension. Thus, the tether tension does not depend on the kite's position only, but also on the steering deflection currently being applied. The following empirical law is proposed to model this behavior:

$$E = E_0 - c\delta^2, \tag{5.2.18}$$

where c is a constant that determines how much the kite's lift-to-drag ratio is penalized for a steering deflection. The basic idea is that a kite steers either by banking (like an airplane), or by increasing the angle of attack on one side of the kite, or through a combination of both. Banking misaligns the kite's aerodynamic-force vector and the tether, causing a reduction in tether tension. Increasing the angle of attack on one side of the kite will tend to decrease the kite's overall lift-to-drag ratio. Both of these effects can be modeled as a decrease in E which depends on the steering deflection δ. A quadratic penalty term was chosen as it best fit the experimental data. So-called 'bang-bang' experiments were carried out, during which the magnitude of the input was maintained constant, while an operator commuted the sign (steering direction) between positive and negative (left and right) to keep the kite flying fast through the air. Based on Equation (2.4.5),the lift-to-drag ratio corresponding to each value of $|\delta|$ was

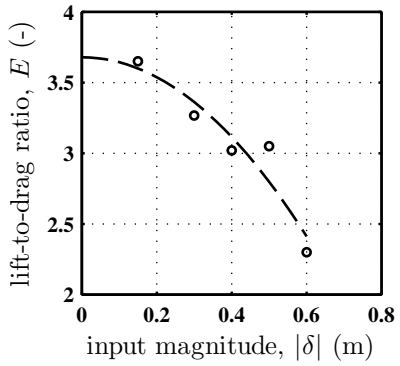

Figure 5.8: The lift-to-drag ratio vs. the magnitude of the steering-deflection set-point, estimated from experimental data using Equation (5.2.19) (circles), and then fitted to these points according to Equation (5.2.18) (dashed).

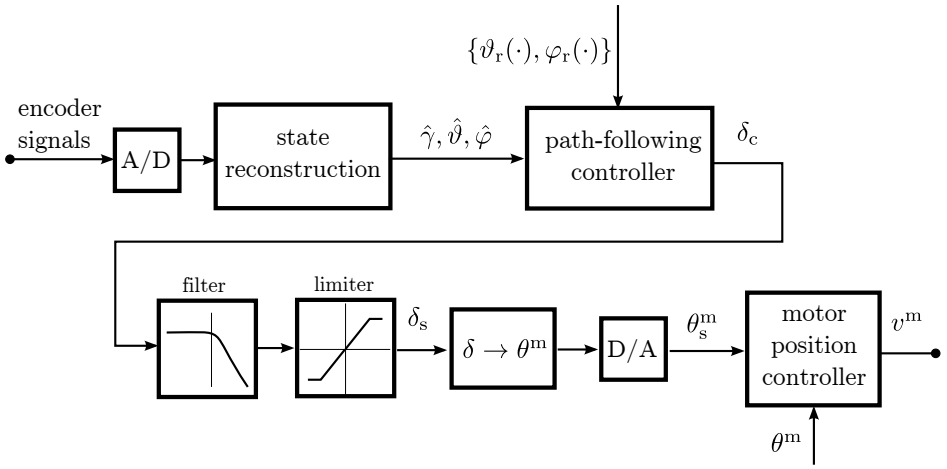

Figure 5.9: Block diagram of the control structure.

then estimated as:

$$E = \frac{1}{t_f - t_0} \int_{t_0}^{t_f} \frac{w_{ap}^{NC}}{\bar{w} \cos \vartheta^{NC}} dt. \tag{5.2.19}$$

For each value of $|\delta|$, a time period, $t_f - t_0$, of at least 1 minute (corresponding to more than 30 turns of the kite) was used, and $\bar{w}$ is the average measured wind speed between t_0 and t_f. The results, along with the fitted curve for E, is shown in Fig. 5.8

5.3 Path-Following Control

The control structure, from the encoder signals to the voltage that is sent to the motor, is shown in Figure 5.9. The state-reconstruction block was already described in Section 5.2.2. It estimates the kite's position and velocity angle from the encoder signals. The

estimates are contaminated by noise, and in the case of the velocity angle, are delayed. The path-following controller, which is described in this section, computes a desired steering input δ_c in order to follow the reference path $\{\vartheta_r(\cdot), \varphi_r(\cdot)\}$. This reference path is periodically updated by the RTO algorithm, to be described in Section 5.4.

The path-following controller's output is low-pass filtered before being converted into a position set-point for the motor. This filter, the only one in the entire control structure, protects the motor (and the entire setup, as the motor is very powerful) by smoothing any high-frequency components in δ_c. Abrupt changes in δ_c can either be caused by the logic in the path-following controller, or by the high-frequency noise contaminating the state estimates. This filter is essential, in fact removing it was observed to cause unstable behavior. Very violent changes in the rear-line lengths can cause oscillations in the front line, which in turn introduces even more high-frequency noise into the state estimates. However, filtering has the unavoidable consequence of introducing delay, due to the filter's negative phase response, and it is important to characterize this delay. Strictly speaking, filter delay is variable, as it depends not only on the order of the filter and the cut-off frequency, but also on the frequency content of the signal being filtered. However, as δ_c tends to have a bandwidth of about 2Hz, it is reasonable to assume a constant value, τ_f, taken as the value of the filter's group delay at a frequency of 1 Hz.

After δ_c is filtered, it is also limited to ensure the steering deflection applied to the system is lower than a preset maximum value, δ_{max}. The filtered, limited steering set point, δ_s, is converted into a corresponding motor position set-point, θ_s^m, for the motor controller. The motor controller regulates the voltage applied to the motor, v^m, to ensure that the motor position, θ^m, matches the set-point. Again, due to the time response of the position control loop, a small time delay, τ_m, is introduced between the steering deflection set-point δ_s, and the actual steering deflection δ. The model parameters, including those estimated experimentally, are given in Table 5.1.

From the point of view of the path-following controller, the system to be controlled incorporates not only the physical system approximated by the cart model, but also the delays introduced at the software level and the state-reconstruction error. This 'overall system' is depicted in Fig. 5.10.

5.3.1 Adaptive Prediction

A number of different time delays affect the system to be controlled, as shown in Figure 5.10, and the combined effect of these delays is non-negligible. There is a delay of about 260 ms before a change in δ_c causes a change in $\hat{\gamma}$. Given that the kite reaches speeds of 40 m·s^{-1}, in 260 ms the kite can travel up to 10 m. With a line length of 35 m, the span

Table 5.1: Parameter values for the small-scale experimental kite system (for the Flysurfer Viron kite)

Parameter	Units	Value
r	m	35
g_s	$\mathrm{rad}\cdot\mathrm{m}^{-2}$	1.1
c	m^{-2}	3.52
$\delta_{\max}$	m	0.4
E_0	-	3.68
T_s	ms	30
τ_e	ms	20
τ_f	ms	50
τ_d	ms	60
τ_m	ms	30
τ_I	ms	120

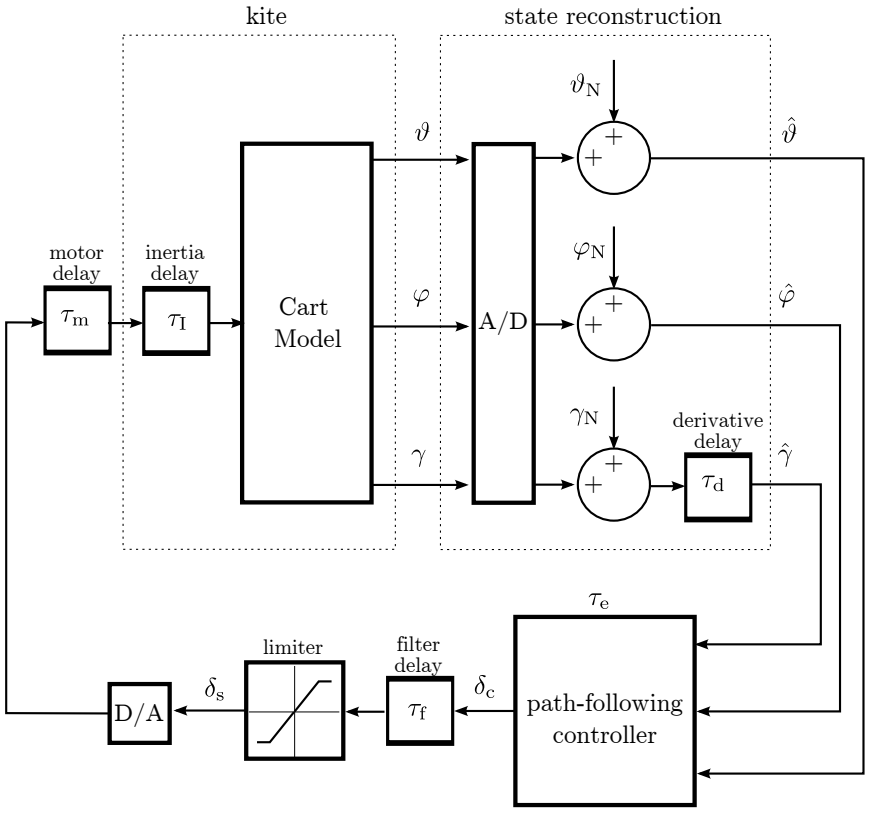

Figure 5.10: The system viewed from the path-following controller's point of view.

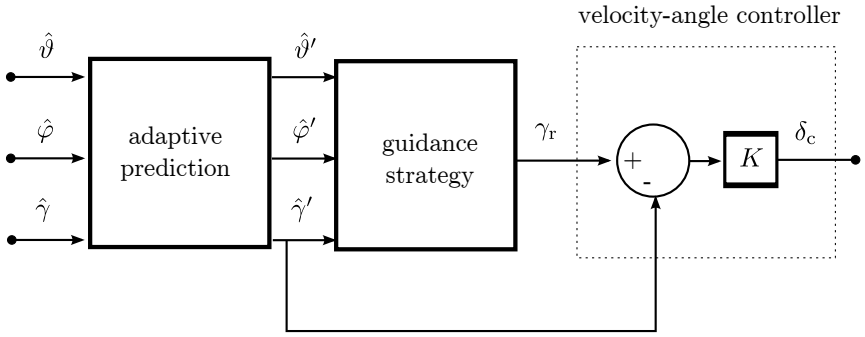

Figure 5.11: The structure of the path-following controller.

of the entire path to be followed may be only 15 m! Not surprisingly, it was found that satisfactory control performance could not be achieved without delay compensation.

If the model equations were completely accurate, delay compensation would simply involve integrating the model equations forward in time. Unfortunately, as shown in Figure 5.7, the dynamic equation accounting for the kite's turning behavior is only a very approximate relationship. Worse still, the turning coefficient g_s varies significantly with the wind speed and is very sensitive to adjustments of the length of the kite's front line (which are necessary to adapt to different wind conditions). For this reason, an adaptive least mean squares (LMS) filter was used to obtain an improved relationship between δ_s and $\widehat{\dot{\gamma}}$, capable of adapting to different flight conditions.

The LMS filter models the *estimate* of the kite's corrected turning rate (defined in Equation (5.2.16)) using the following relationship:

$$\widehat{\dot{\gamma}}_c[n] = (\mathbf{h}_n)^T \boldsymbol{\delta}_s[n], \tag{5.3.1}$$

where $\boldsymbol{\delta}_s[n] = \begin{bmatrix} \delta_s[n-d] & \delta_s[n-d-1] & \cdots \delta_s[n-d-n_{LMS}] \end{bmatrix}^T$ is the vector of past inputs, delayed by d sampling periods, n_{LMS} is the filter length, and $\mathbf{h}_n \in \mathbb{R}^{n_{LMS}}$ is the vector of filter weights. The value of d is selected such that $d T_s \simeq \tau_m + \tau_I + \tau_d$, that is, the number of sampling periods delay between δ_s and $\hat{\dot{\gamma}}$ (see Figure 5.10). The estimation error at each sampling period is:

$$e_n = \dot{\gamma}_c[n] - (\mathbf{h}_n)^T \boldsymbol{\delta}_s[n]. \tag{5.3.2}$$

The filter weights are adapted at each sampling period in order to minimize the mean-square estimation error, $E[e_n^2]$. The Least-Mean-Squares gradient algorithm (LMS) was used to find the optimal weights (Proakis and Manolakis, 2006). The filter coefficients are recursively computed in order to minimize $E[e_n^2]$, using the following simple update

equation:

$$\mathbf{h}_{n+1} = \mathbf{h}_n + \mu e_n \boldsymbol{\delta}_\mathrm{s}[n], \tag{5.3.3}$$

where μ is the step size.

Using this relationship linking δ_s to $\widehat{\dot{\gamma}}_\mathrm{c}$, it is possible to *predict* what the kite states will be several sampling periods into the future. This is necessary because, due to the time delays, changes in δ_c will only begin to affect the kite's behavior several sampling periods later. Thus, decisions made by the path-following controller should be based upon the states estimates in the future, rather than the current estimates. At sampling instant n_0, corresponding to time t_0, the aim is to obtain the following predictions:

$$\hat{\vartheta}'[n_0] \simeq \hat{\vartheta}[n_0 + d'] = \hat{\vartheta}(t_0 + \tau_\mathrm{f} + \tau_\mathrm{m} + \tau_\mathrm{I}), \tag{5.3.4}$$

$$\hat{\varphi}'[n_0] \simeq \hat{\varphi}[n_0 + d'] = \hat{\varphi}(t_0 + \tau_\mathrm{f} + \tau_\mathrm{m} + \tau_\mathrm{I}), \tag{5.3.5}$$

$$\hat{\gamma}'[n_0] \simeq \hat{\gamma}[n_0 + d' + d_\mathrm{d}] = \hat{\gamma}(t_0 + \tau_\mathrm{f} + \tau_\mathrm{m} + \tau_\mathrm{I} + \tau_\mathrm{d}), \tag{5.3.6}$$

where $d' T_\mathrm{s} \simeq \tau_\mathrm{f} + \tau_\mathrm{m} + \tau_\mathrm{I}$, and recalling that $d_\mathrm{d} = T_\mathrm{s}\tau_\mathrm{d}$ is the number of sampling periods derivative delay. Essentially, obtaining these prediction will negate the effect of all time delays except the execution delay, which is neglected. It may help to refer to Figure 5.10 to understand this. The prediction is calculated in two steps. First, the following equation is integrated forward in time by d_d sampling periods, from $n = n_0$ to $n = n_0 + d_\mathrm{d}$:

$$\dot{\gamma}[n] = \mathbf{h}_{n_0}^T \boldsymbol{\delta}_\mathrm{s}[n] - \widehat{\dot{\varphi}}[n_0] \cos\left(\hat{\vartheta}[n - d_\mathrm{d}]\right), \quad \gamma[n_0] = \widehat{\gamma}[n_0]. \tag{5.3.7}$$

This yields $\hat{\gamma}_{+d_\mathrm{d}}$, which is actually an estimate of the kite's velocity angle at the current sampling instant n_0, as this first step compensates for the derivative delay. The next step predicts what the kite's states will be d' sampling periods into the future. This is achieved by integrating the following equations forward in time by d' sampling periods, from $n = n_0$ to $n = n_0 + d'$:

$$\dot{\vartheta}[n] = \left(\frac{\bar{w}}{r} E_0 \cos\left(\vartheta[n]\right)\right) \cos\left(\gamma[n]\right), \qquad \vartheta[n_0] = \hat{\vartheta}[n_0], \tag{5.3.8}$$

$$\dot{\varphi}[n] = \left(\frac{\bar{w}}{r} E_0 \cos\left(\vartheta[n]\right)\right) \frac{\sin\left(\gamma[n]\right)}{\sin\left(\vartheta[n]\right)}, \qquad \varphi[n_0] = \hat{\varphi}[n_0], \tag{5.3.9}$$

$$\dot{\gamma}[n] = \mathbf{h}_{n_0}^T \boldsymbol{\delta}_\mathrm{s}[n + d_\mathrm{d}] - \dot{\varphi}[n] \cos\left(\vartheta[n]\right), \quad \gamma[n_0] = \hat{\gamma}_{+d_\mathrm{d}}, \tag{5.3.10}$$

to yield ϑ', φ' and γ', where $\bar{w}$ is the average measured wind speed during the past 15 minutes[3]. Note that the differential equations for ϑ and φ are from the Cart Model

[3] Note that this integration requires *future* values of δ_s, up until $\delta_\mathrm{s}[n_0 + \tau_\mathrm{f}]$. These can be calculated

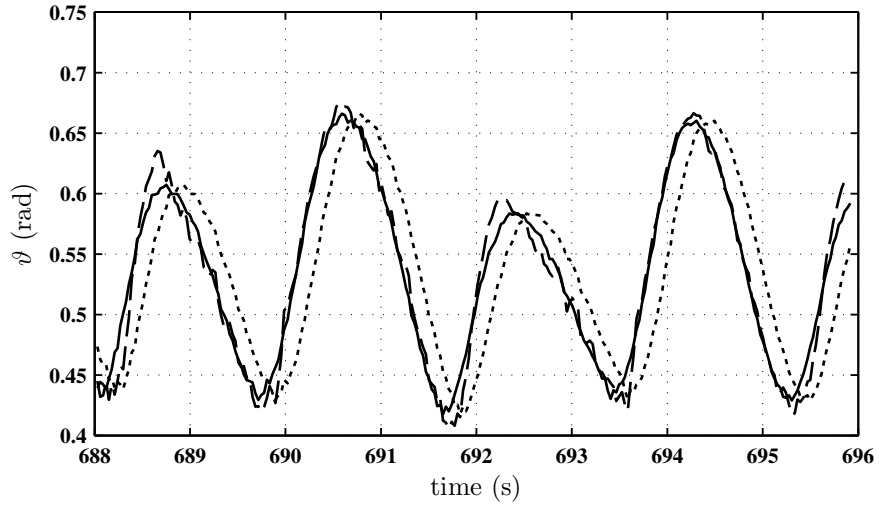

Figure 5.12: Performance of the adaptive prediction algorithm during an experiment: the prediction $\hat{\vartheta}'(t)$ (dashed), the signal we are trying to predict $\hat{\vartheta}(t + d'T_s)$ (solid), and the non-delay-compensated estimate $\hat{\vartheta}(t)$ (dotted).

(Equations 5.2.8 and 5.2.9), with the simplifying assumption that $E = E_0$. Both integration steps are performed using the simple Euler Method, which integrates a differential equation for a variable x forward in time according to: $x[n+1] = x[n] + T_s\dot{x}[n]$.

The performance of this prediction algorithm during an experiment is shown in Figures 5.12, 5.13 and 5.14. The prediction is not absolutely perfect, but it is certainly far more accurate than using the non-delay-compensated state estimates. Indeed, the next section demonstrates the importance of the prediction on control performance.

5.3.2 Velocity-Angle Control

The velocity-angle controller is a simple proportional feedback loop. The performance during an experiment is shown in Figure 5.15. The closed-loop response is very good, as, thanks to the adaptive prediction, Equation (5.2.17) holds, and the relationship between the manipulated variable δ_c and the controlled variable $\hat{\gamma}'$ is roughly that of a first-order dynamical system.

The performance obtained during an early experiment *without the adaptive prediction* is shown in Fig. 5.16. The oscillations around the reference signal are typical of proportional control applied to a first-order system with significant time delay. An important peculiarity of path-following control can be observed: oscillations in the controlled variable cause oscillations in the reference signal! This is because the reference signal

from the current values of δ_c.

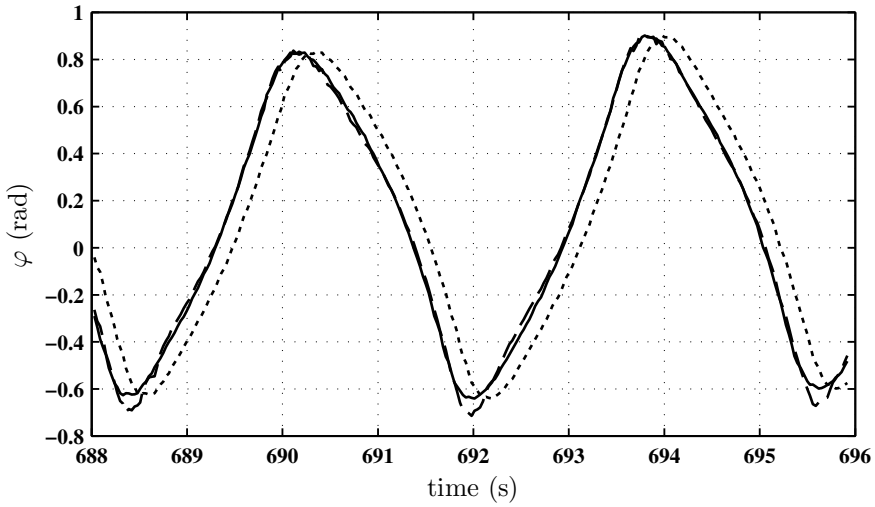

Figure 5.13: Performance of the adaptive prediction algorithm during an experiment: the prediction $\hat{\varphi}'(t)$ (dashed), the signal we are trying to predict $\hat{\varphi}(t + d' T_\mathrm{s})$ (solid), and the non-delay-compensated estimate $\hat{\varphi}(t)$ (dotted).

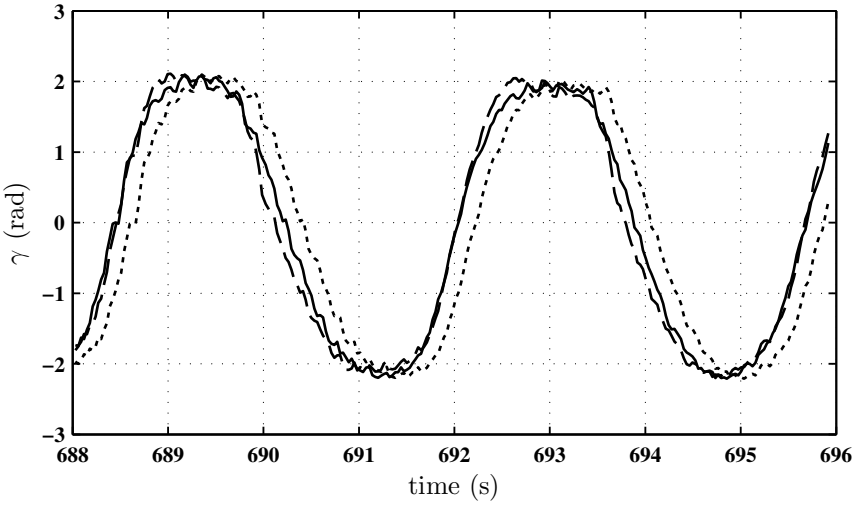

Figure 5.14: Performance of the adaptive prediction algorithm during an experiment: the prediction $\hat{\gamma}'(t)$ (dashed), the signal we are trying to predict $\hat{\gamma}(t + d' T_\mathrm{s} + d_\mathrm{d} T_\mathrm{s})$ (solid), and the non-delay-compensated estimate $\hat{\gamma}(t)$ (dotted).

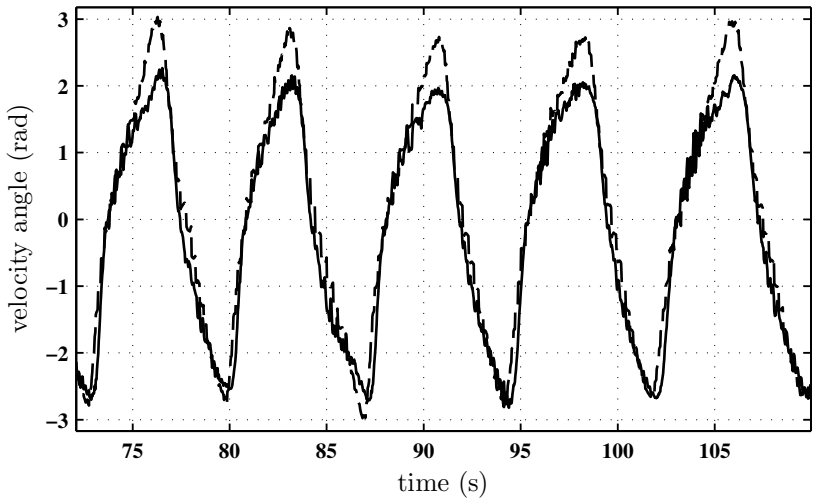

Figure 5.15: Performance of the velocity-angle control loop during autonomous figure-of-eights with adaptive prediction. The reference signal γ_r (dashed), and the controlled variable $\hat{\gamma}'$ (solid).

depends on the kite's position, so if the kite's position oscillates around the desired position on the path due to control error, oscillations will tend to be introduced into the reference signal also. These variations in the reference signal are likely to cause further oscillations in the controlled variable, as the poorly performing controller tries to track them. The overall effect is very poor path following, and for this reason, it was found that the response of the velocity control loop must absolutely not overshoot or oscillate. A similar observation regarding the detrimental effect of delay has been made by Jehle and Schmehl (2014).

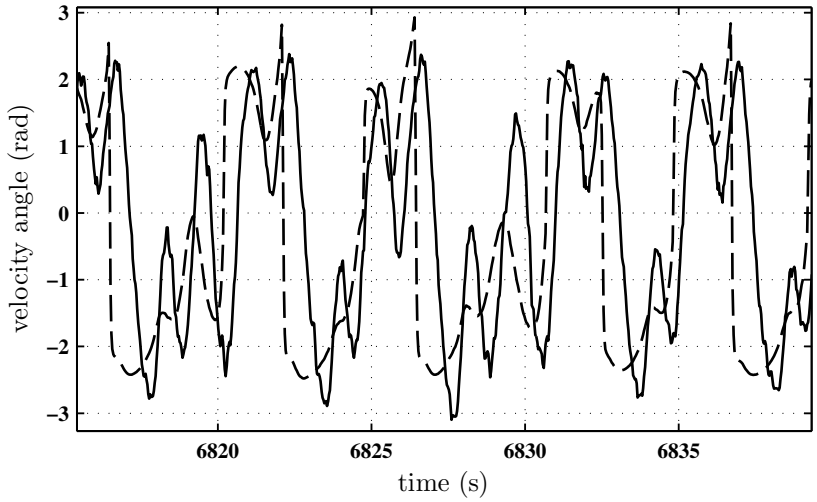

Figure 5.16: Performance of the velocity-angle control loop during autonomous figure-of-eights without adaptive prediction. The reference signal γ_r (dashed), and the controlled variable $\hat{\gamma}$ (solid).

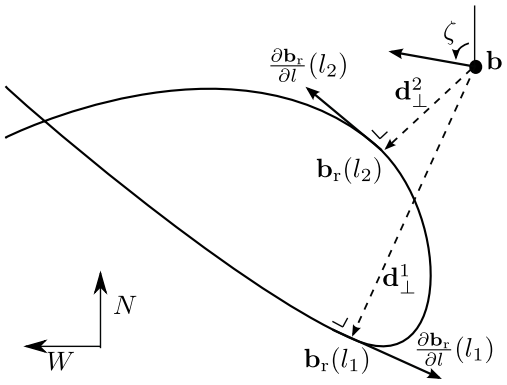

Figure 5.17: Path-following controller: illustration of the kite's position relative to the reference path (all projected onto the $\{N,W\}$ plane, shown in Figure 2.2). $\mathbf{b}$ is the kite's position, and $\mathbf{b}_r(l_1)$ and $\mathbf{b}_r(l_2)$ are the two points on the path at which the path's tangent is perpendicular to the line joining the kite position to that point.

5.3.3 Guidance Strategy

This section describes a simple 'vector-field' path-following controller for kites (or any vehicle moving on a plane). This type of controller is popular in the unmanned-aerial-vehicle (UAV) community, where it was recently developed to follow circular and straight-line paths (or composites of these) (Nelson et al., 2007). Here, it is adapted to follow arbitrary smooth paths (including paths that intersect themselves).

Firstly, the controller aims to control the position of the kite on the $\{N,W\}$ plane, as defined in Section 2.4. Recall that this plane is tangent to the sphere upon which the kite can move, at the point $\{\bar{\vartheta},0\}$. This point is chosen as the intersection point of the figure-of-eight reference path, i.e. the reference path's center. Essentially, this plane is a local approximation of the sphere, that allows us to consider the path-following problem on a flat 2-D surface.

The kite's position on the $\{N,W\}$ plane is defined as:

$$\mathbf{b} = \mathbf{T}\mathbf{p}, \quad \text{with} \quad \mathbf{T} = \begin{bmatrix} -\sin\bar{\vartheta} & 0 & \cos\bar{\vartheta} \\ 0 & 1 & 0 \end{bmatrix}. \tag{5.3.11}$$

The kite's velocity in $\{x,y,z\}$ coordinates is given by:

$$\dot{\mathbf{p}} = r\mathbf{C}\begin{bmatrix} \dot{\vartheta} \\ \dot{\varphi}\sin\vartheta \end{bmatrix}, \quad \text{with} \quad \mathbf{C} = \begin{bmatrix} -\sin\vartheta & 0 \\ \cos\vartheta\sin\varphi & \cos\varphi \\ \cos\varphi\cos\vartheta & -\sin\varphi \end{bmatrix}. \tag{5.3.12}$$

Now, using Equations 5.2.8 and 5.2.9, a relationship between γ and the kite's velocity

vector projected onto the $\{N, W\}$ plane can be obtained:

$$\dot{\mathbf{b}} = r\mathbf{TC}\begin{bmatrix} \dot{\vartheta} \\ \dot{\varphi}\sin\vartheta \end{bmatrix}, \quad \text{with} \quad \begin{bmatrix} \dot{\vartheta} \\ \dot{\varphi}\sin\vartheta \end{bmatrix} = \omega_{\mathrm{k}}\begin{bmatrix} \cos\gamma \\ \sin\gamma \end{bmatrix} \tag{5.3.13}$$

where we recall that ω_{k} is the kite's speed in rad·s^{-1}. We define the kite's velocity angle on the $\{N, W\}$ plane as:

$$\zeta = \angle\dot{\mathbf{b}} = \tan^{-1}\left(\frac{\dot{\mathbf{b}}_W}{\dot{\mathbf{b}}_N}\right). \tag{5.3.14}$$

The reference path (on the $\{N, W\}$ plane) is denoted $\mathbf{b}_{\mathrm{r}}(l)$, where l is the path length. The points on the path, l_i, at which the path's tangent is perpendicular to the kite position are $\mathbf{b}_{\mathrm{r}}(l_i)$. The angle of the path at each point is denoted as:

$$\zeta^i = \angle\frac{\partial\mathbf{b}_{\mathrm{r}}}{\partial l}(l_i), \tag{5.3.15}$$

while the vector pointing from the kite to each point is:

$$\mathbf{d}_\perp^i = \mathbf{b}_{\mathrm{r}}(l_i) - \mathbf{b}. \tag{5.3.16}$$

A *desired* velocity angle corresponding to each point is obtained with the classic vector-field law (Nelson et al., 2007):

$$\zeta_{\mathrm{d}}^i = \zeta^i + \zeta_{\mathrm{e}}\left(\frac{\|\mathbf{d}_\perp^i\|}{d_{\max}}\right)^\beta \times \mathrm{sgn}\left(\angle\left(\mathbf{d}_\perp^i - \frac{\partial\mathbf{b}_{\mathrm{r}}}{\partial l}(l_i)\right)\right) + \alpha\frac{\partial^2\mathbf{b}_{\mathrm{r}}}{\partial l^2}(l_i), \tag{5.3.17}$$

where the entry velocity angle ζ_{e} and the coefficient $\beta > 1$ are tuning parameters. The final term is a new (at least in the context of the vector-field controller) *curvature-compensation* term that allows the controller to effectively follow a curved path. The curvature of the path indicates the rate of change of the path's angle (direction). Thanks to the curvature compensation (which can be varied by adjusting α), the controller anticipates curves in the path. Finally, the reference velocity angle is selected as the ζ_{d}^i that is closest to the kite's current velocity angle.

$$\zeta_{\mathrm{r}} = \zeta_{\mathrm{d}}^{i_{\mathrm{r}}}, \qquad i_{\mathrm{r}} = \arg\min_i \quad |\zeta - \zeta_{\mathrm{d}}^i|. \tag{5.3.18}$$

The reference velocity angle in $\{N, W\}$ coordinates is translated back into a reference signal for the velocity-angle control loop by inverting Equation 5.3.13:

$$\gamma_{\mathrm{r}} = \tan^{-1}\left(\frac{x_2}{x_1}\right), \quad \text{with} \quad \mathbf{x} = (\mathbf{TC})^{-1}\begin{bmatrix} \cos\zeta_{\mathrm{r}} \\ \sin\zeta_{\mathrm{r}} \end{bmatrix}. \tag{5.3.19}$$

The path-following controller's performance while following different reference paths is shown in Figures 5.18, 5.19 and 5.20 . Each of these graphs represents 10 minutes of experimental data, during which the kite flies roughly 150 loops. It can be seen that the controller is very consistent, i.e. the path followed by the kite is very similar from one loop to the next. The path followed by the kite is not exactly the reference path, and there is a small but consistent error between the two. This is to be expected, and is also the case for the other account of path-following control for flexible kites to be found in the literature (Jehle and Schmehl, 2014). As long as the controller is consistent and robust to wind variations, a small path-following error is not actually a problem, as there is no practical motivation to control the kite's path *exactly*.

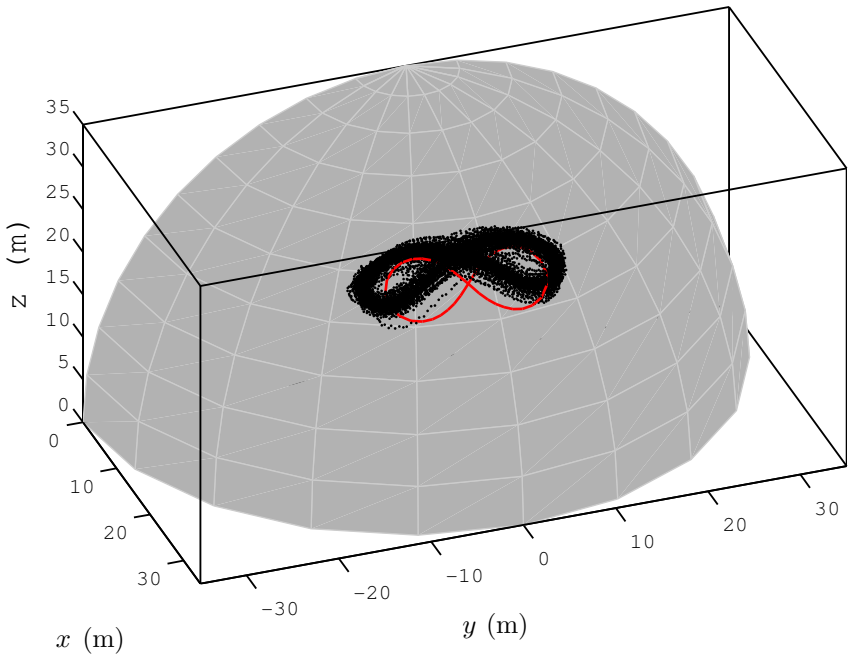

Figure 5.18: The kite's position (dots) during 10 minutes of autonomous flight, tracking the *high, narrow* reference path shown in red. The kite is restrained to flying on the gray quarter sphere.

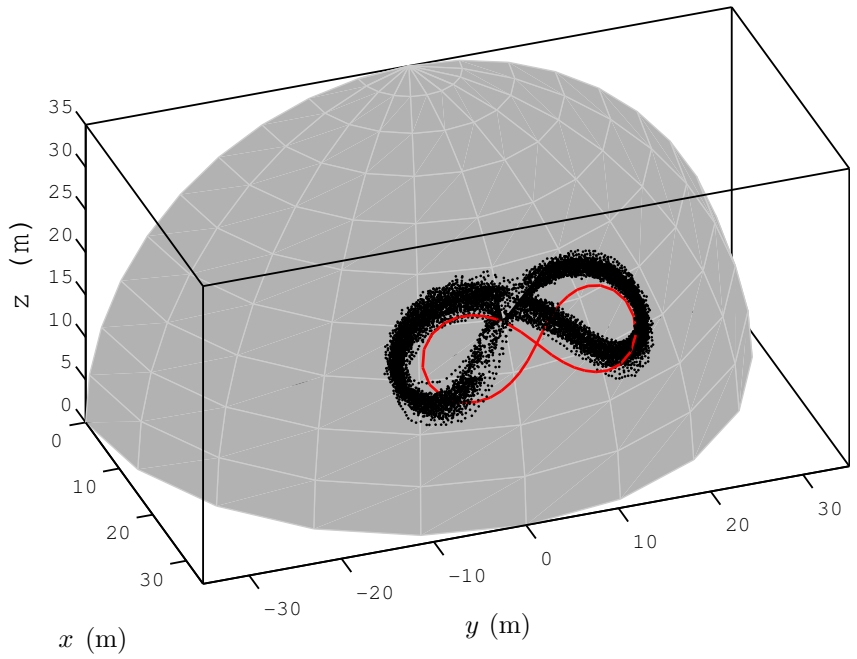

Figure 5.19: The kite's position (dots) during 10 minutes of autonomous flight, tracking the reference path shown in red.

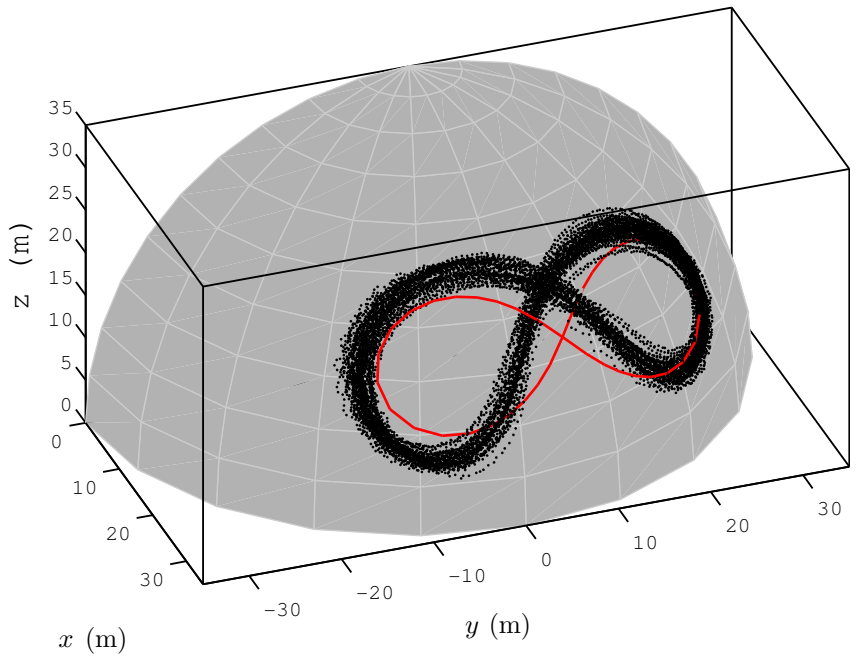

Figure 5.20: The kite's position (dots) during 10 minutes of autonomous flight, tracking the *wide* reference path shown in red.

5.4 Real-Time Optimization

The periodic reference path used by the path-following controller is modified while the kite is flying. The RTO algorithm executes periodically (every N_{avg} cycles of the reference path), each time specifying a new reference path. A slightly simplified version of the Dual D-MA algorithm described in Chapter 4 was implemented in the RTO layer.

5.4.1 RTO Algorithm

The RTO scheme is designed using 'full' (unsimplified) model-based cost and constraint functions, whose argument is a 'full', high-dimensional, vector of inputs, $\mathbf{u}$. However, on-line optimization is actually only carried out with respect to a lower-dimensional vector of RTO inputs, $\mathbf{u}'$. The 'full' model inputs are defined, identically to in Chapter 4, as a finite set of points on the reference path:

$$\mathbf{u} = \left[\vartheta_{\mathrm{r}}(0) \quad \varphi_{\mathrm{r}}(0) \quad \vartheta_{\mathrm{r}}(\tfrac{1}{N}) \quad \varphi_{\mathrm{r}}(\tfrac{1}{N}) \quad \vartheta_{\mathrm{r}}(\tfrac{2}{N}) \quad \varphi_{\mathrm{r}}(\tfrac{2}{N}) \cdots \vartheta_{\mathrm{r}}(\tfrac{N-1}{N}) \quad \varphi_{\mathrm{r}}(\tfrac{N-1}{N})\right]^T, \tag{5.4.1}$$

where $N = n_u/2 = 20$. The 'full' optimization functions, ϕ and $\mathbf{g}$, are defined as:

$$\phi(\mathbf{u}) := -\frac{\bar{T}}{c_T}, \qquad \mathbf{g}(\mathbf{u}) := 1 - \frac{z_{\mathrm{L,r}}}{z_{\min}} \leq \mathbf{0}, \tag{5.4.2}$$

where $\bar{T}$ is the average line tension attained when following the reference path, calculated from $\mathbf{u}$ using the Erhard Model, as described in Appendix B, and $z_{\mathrm{L,r}}$ is the *lowest* altitude attained by the kite's reference path, i.e.

$$z_{\mathrm{L,r}} = \min_{j=0,1,\ldots N-1} r \sin\left(\vartheta_{\mathrm{r}}\left(\frac{j}{N}\right)\right) \cos\left(\varphi_{\mathrm{r}}\left(\frac{j}{N}\right)\right) \tag{5.4.3}$$

By fitting a spline to the points specified by $\mathbf{u}$, the continuous reference path can be expressed in terms of $\mathbf{u}$:

$$\{\vartheta_{\mathrm{r}}(\cdot), \varphi_{\mathrm{r}}(\cdot)\} = \boldsymbol{\mathcal{W}}(\mathbf{u}). \tag{5.4.4}$$

The sensitivity analysis described in Section 4.1.2 was applied to the 'full' cost and constraint functions, ϕ and $\mathbf{g}$. The uncertain model parameters, and their uncertainty intervals, for the experimental system are given in Table 5.2. The resulting privileged directions are very similar to those obtained for the simulated example in Chapter 4(this is to be expected, as the large-scale simulation study in Chapter 4 was designed as a scaled-up version of the small-scale prototype). Again, adaptation in two privileged

Table 5.2: Uncertainty intervals for the uncertain model parameters.

Parameter	Nominal value	Minimum value	Maximum value	Unit
E_0	3.83	3	5	-
g_s	1.1	.5	2	$\mathrm{rad}\cdot\mathrm{m}^{-2}$
c	3.52	1.5	6	m^{-2}
Δw	0.05	0	.1	s^{-1}

directions is necessary to reject the optimality loss introduced by parameter variations, leading to a matrix of privileged directions $\mathbf{U}_\mathrm{r} \in \mathbb{R}^{n_u \times 2}$. At this point, a simplification is introduced with respect to the Dual D-MA algorithm presented in Chapter 4. The reference path, which is provided for the path-following controller by the RTO layer, is parametrized in the following manner:

$$\{\vartheta_\mathrm{r}(\cdot), \varphi_\mathrm{r}(\cdot)\} = \boldsymbol{\mathcal{W}}(\mathbf{u}^*(\boldsymbol{\theta}_0) + \mathbf{U}_\mathrm{r}\mathbf{R}\mathbf{u}'), \tag{5.4.5}$$

where $\mathbf{u}' \in \mathbb{R}^2$ is the vector of RTO inputs, and $\mathbf{R}$ is a 2×2 rotation matrix that is chosen such that u'_1 principally affects the *height* of the reference path, while u'_2 affects its width. The reference path for different values of $\mathbf{u}'$ is shown in Figures 5.21 and 5.22. From the practical point of view, allowing the RTO layer to adjust the path in these two directions makes perfect sense. The height of the trajectory can be adjusted to suit the wind shear. Increasing the width of the trajectory tends to increase the path's curvature at each point. Thus, the curvature can be adapted to suit the kite's turning behavior.

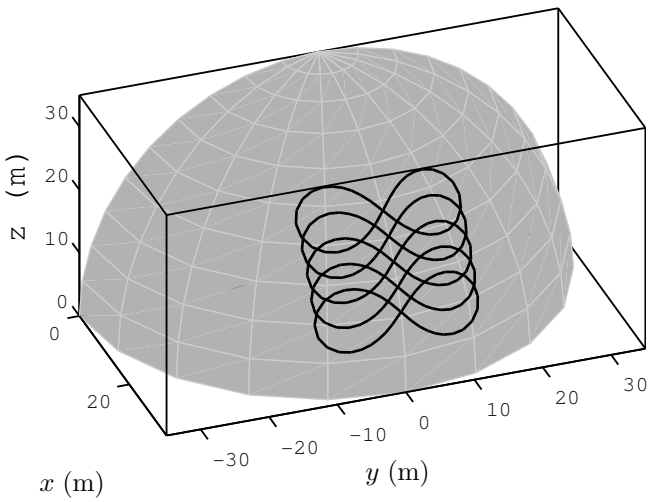

Figure 5.21: Reference path for different values of $u'_1 = \{-0.2, -0.1, 0, 0.1, 0.2\}$, with $u'_2 = 0$.

The (simplified) model-based optimization problem that is modified and solved online

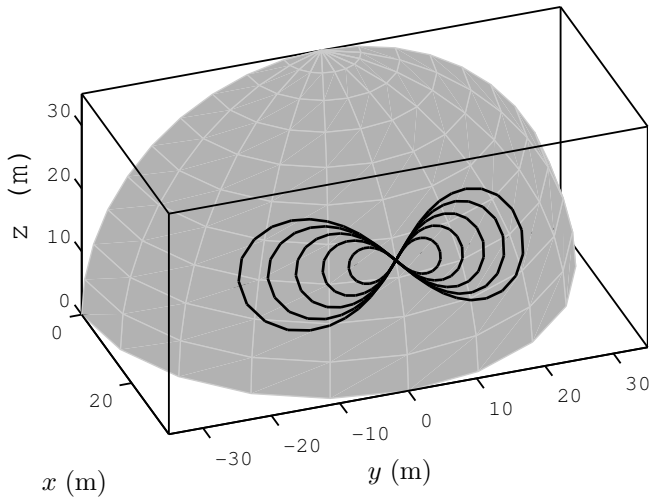

Figure 5.22: Reference path for different values of $u_2' = \{-0.2, -0.1, 0, 0.1, 0.2\}$, with $u_1' = 0$.

by the Dual D-MA algorithm uses $\mathbf{u}'$ as a vector of decision variables. This (simplified) model-based problem is:

$$\begin{aligned} &\min_{\mathbf{u}'} \quad \phi'(\mathbf{u}') := \phi\left(\mathbf{u}^*(\boldsymbol{\theta}_0) + \mathbf{U}_\mathrm{r}\mathbf{R}\mathbf{u}',\, \boldsymbol{\theta}_0\right) \\ &\text{subject to} \quad \mathbf{g}'(\mathbf{u}') := \mathbf{g}\left(\mathbf{u}^*(\boldsymbol{\theta}_0) + \mathbf{U}_\mathrm{r}\mathbf{R}\mathbf{u}',\, \boldsymbol{\theta}_0\right) \leq \mathbf{0}. \end{aligned} \tag{5.4.6}$$

This is the model-based optimization problem that is used (and modified at each iteration) by the Dual D-MA algorithm. The matrix of privileged directions to be used with the simplified problem is $\mathbf{U}_\mathrm{r}' = \mathbf{I}_2$, i.e. experimental gradients are estimated in all directions of the simplified problem's decision-variable space. If translated into words, the optimization aim is to adjust $\mathbf{u}'$ in order to maximize the tether tension, while ensuring the kite's altitude is always above a minimum value.

The measured cost and constraint functions are defined as:

$$\tilde{\phi}_\mathrm{p}'(\mathbf{u}') := -\frac{\bar{T}}{c_T}, \qquad \tilde{\mathbf{g}}_\mathrm{p}'(\mathbf{u}) := 1 - \frac{z_\mathrm{L}}{z_\mathrm{min}}, \tag{5.4.7}$$

where in this case $\bar{T}$ is the average line tension measured during N_avg cycles of the reference path, $c_T = \left(\frac{1}{2}\rho A\right) r^2 w_\mathrm{ref}^2$ is a scaling factor to make the cost dimensionless, z_L is the lowest altitude attained by the kite during N_avg cycles, and $z_\mathrm{min} = 1.8$ m is the minimum permissible altitude. The RTO scheme is represented using a block diagram in Figure 5.23. First, the measured average line tension and minimum altitude is converted into cost- and constraint-function values for the Dual D-MA algorithm, according to Equation 5.4.7. The Dual D-MA algorithm computes the next [4] RTO inputs. These are

[4] Note that upon receiving the measurements from iteration $k-1$, the Dual D-MA algorithm actually

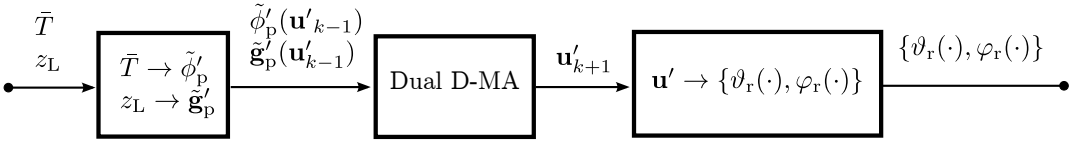

Figure 5.23: The block diagram of the implemented RTO scheme. This is executed every N_{avg} figure-of-eights to update the reference path.

then converted into a reference path, according to Equation 5.4.5.

It is important to characterize the noise affecting the measured cost- and constraint-functions. The average line tension and the minimum altitude per path cycle are shown in Figure 5.24 , while the kite followed the same reference path for 6 minutes. The minimum height is relatively consistent (the variations are generally ±1 m), which can be explained by the consistency of the path-following controller. However, the average line tension is extremely variable, ranging from 60 kg to 120 kg, i.e. up to 35 % noise. This would be considered 'extreme' noise in the RTO literature, where simulation studies generally assume at most 10% measurement noise, and usually less. This noise is not due to measurement error, as the load cell measuring the front line's tension has an error of ±.02 kg. Neither is it caused by variable controller performance, as the kite follows almost exactly the same path during each cycle. Rather, the noise is caused by wind variations. The wind speed measured at the ground station during the same experiment is shown in Figure 5.25. The measured wind speed varies significantly over time, and there is a rough correlation between the variations in the measured wind speed and the line tension.

The wind measured over a longer period of time is shown in Figure 5.26, and the spectrum of the wind speed estimated from this data is shown in Figure 5.27. Unfortunately, there is a significant amount of low-frequency wind speed variation. This results in low-frequency variations affecting the average line tension per cycle, which can only be removed to a limited extent with averaging or low-pass filtering. The measured wind speed cannot be used to associate tension variations with wind speed variations, as on a short time scale, there is poor correlation between the average tension produced by the kite and the wind speed measured at the ground station. This does not mean the wind-speed measurement on the ground is completely inaccurate, it is certainly a good *indication* of the average wind speed, and the magnitude of the wind gusts, experienced by the kite. The same rough trend in the wind-speed measurements in Figure 5.25 can also be observed in the line tension in Figure 5.24. However, it suggests that the rapid wind variations experienced by the kite are not necessarily the same, and do not occur

calculates the RTO inputs for iteration $k+1$. This is due to the delay caused by the Dual D-MA algorithm's calculation time.

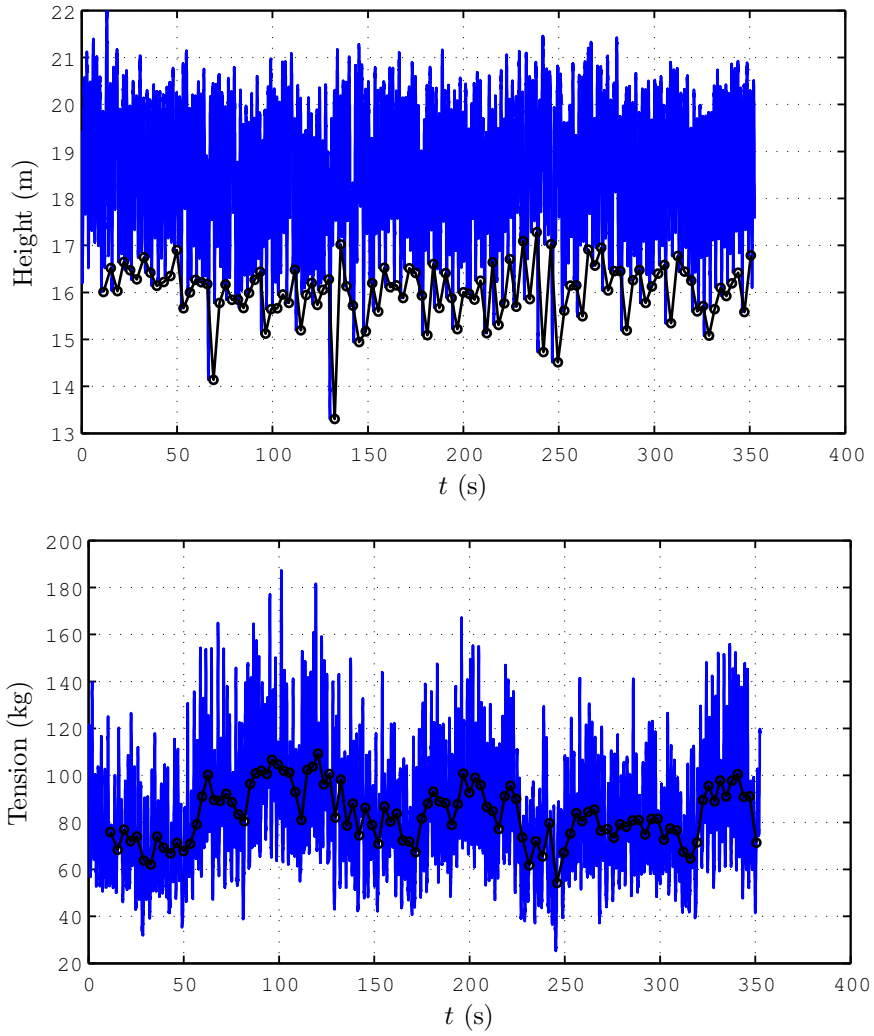

Figure 5.24: The kite's altitude and the measured line tension during 6 minutes following a constant reference path (blue). The minimum attained altitude and the average line tension per path cycle (black).

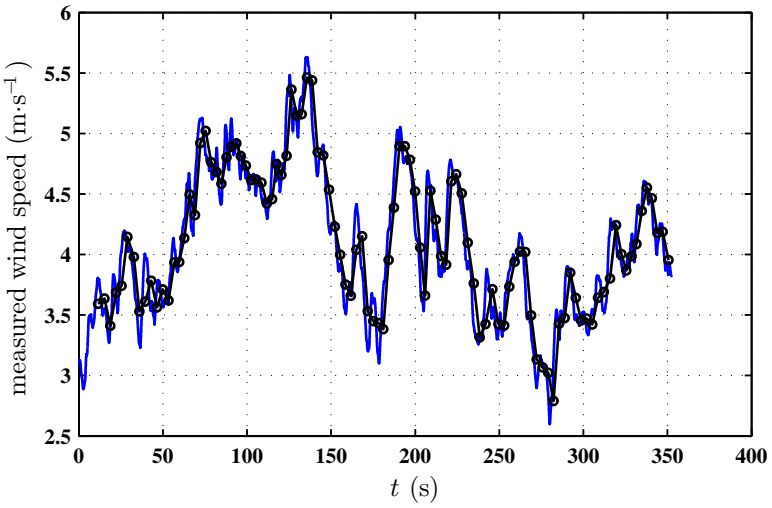

Figure 5.25: The wind speed measured at the ground station during the experiment shown in Figure 5.24 (blue), and the average wind speed per reference-path cycle (black dots).

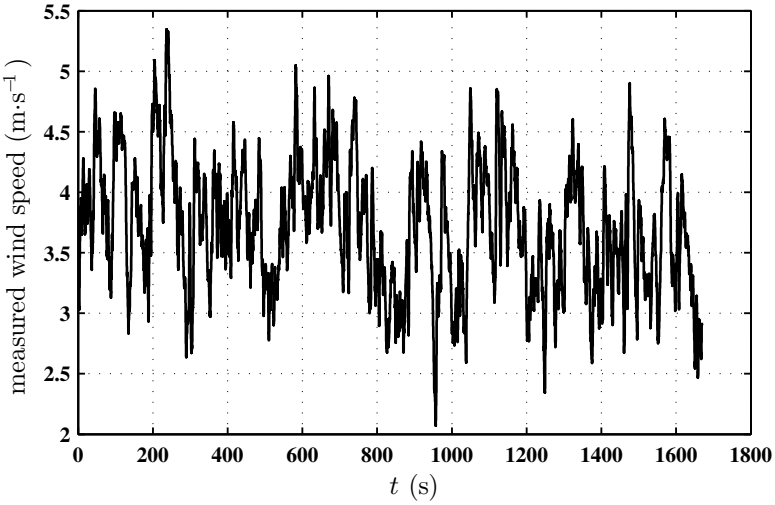

Figure 5.26: The wind speed measured at the ground station over a 30-minute period.

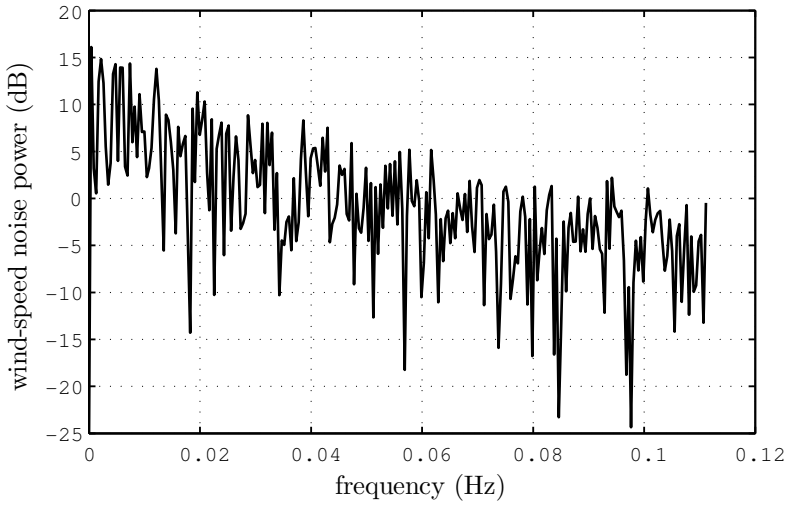

Figure 5.27: The spectrum of the wind-speed variations, estimated from the signal in Figure 5.26.

at exactly the same time, as those measured on the ground.

The simple solution to dealing with noise, which is the one adopted here, is to partially remove it via averaging. The effect of averaging the line tension over several cycles is apparent in Figure 5.28. The (scaled) standard deviation of the measured line tension vs. N_{avg} is shown in Figure 5.29. Essentially, averaging the line tension over several path cycles, as opposed to just one cycle, progressively reduces the noise to a manageable (if still very high) level.

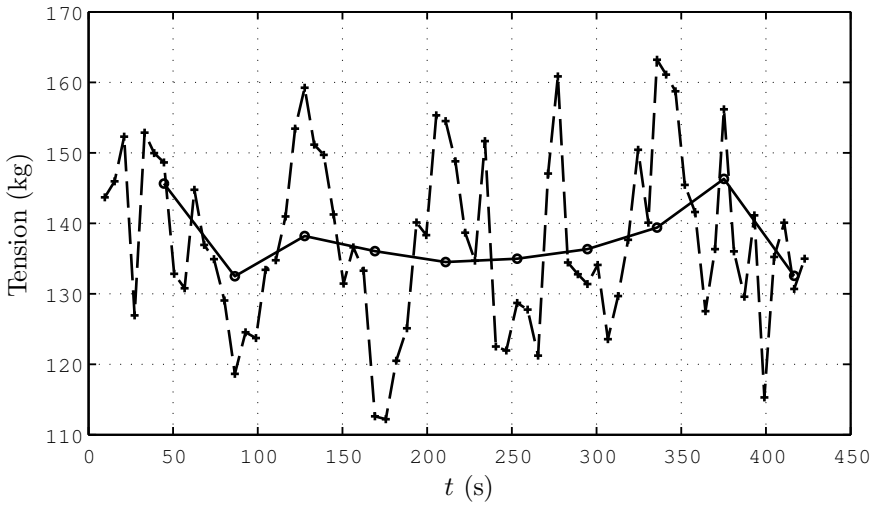

Figure 5.28: $N_{\text{avg}} = 1$ (dashed with crosses), $N_{\text{avg}} = 7$ (solid with circles). During this experiment the kite followed a constant reference path.

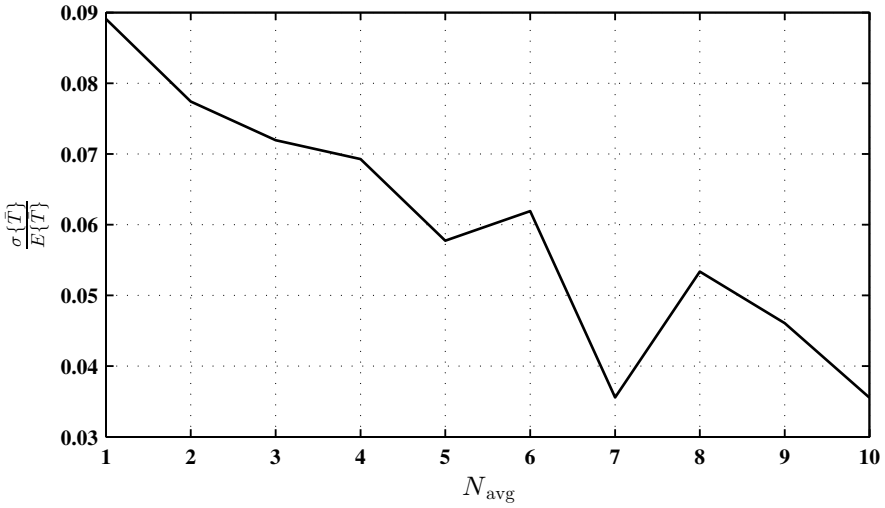

Figure 5.29: Standard deviation of the process noise affecting the average line-tension measurement vs. N_{avg}, based on the data shown in Figure 5.28.

Table 5.3: Values of the design parameters for the Dual D-MA algorithm.

Parameter	Value
n_r	2
$\Delta_{\max}$	0.03
$\Delta^{\text{r}}_{\max}$	0.06
$\mathbf{\Sigma}_0^{\phi}$	$20^2 \times \mathbf{I}_2$
$\mathbf{\Sigma}_0^{g}$	$5^2 \times \mathbf{I}_2$
σ_{TOL}	.5
c_0	1

$N_{\text{avg}} = 7$ was chosen. The other parameters for the Dual D-MA algorithm were hand tuned using a simulator of the system. These are given in Table 5.3.

5.4.2 RTO Results

The RTO algorithm's performance was tested over several day's of experiments. In general it performed reasonably well, each time converging to roughly the same reference path. In order to verify the algorithm's performance, it is useful to test it many times under near-identical conditions. Unfortunately this is complicated by the ever-changing wind conditions, which often require slight adjustments to the experimental setup. Increasing the kite's angle of attack, or tuning the controller gain to ensure good performance when the wind drops slightly, can modify the plant's optimal solution. Nonetheless, at one point several hours of continuous experiments were carried out without any adjustments being made to the experimental setup, and these results are presented next.

Figure 5.30 shows the tension and the kite's altitude during 30 minutes of autonomous flight. During the first 7 minutes (13 iterations) and the last 5 minutes (10 iterations), the kite follows an unchanging reference path that is rather high and narrow, resulting in a low average line tension, about 80 kg. The RTO algorithm markedly improves the average line tension, increasing it to about 135 kg. It is interesting to note that the minimum altitude constraint, which is very low, does not become active. The RTO inputs during this experiment are shown in Figure 5.31. The zig-zagging behavior is in part caused by the Dual D-MA algorithm exciting the process in order to estimate experimental gradients, and in part due to the noisiness of the cost gradient estimate, shown in Figure 5.32. Theoretically, the estimate of the cost gradient should tend towards zero. In reality, the estimate is extremely noisy despite the significant averaging ($N_{\text{avg}} = 7$) used during this experiment to reduce the effect of noise. So how does the Dual D-MA algorithm markedly improve the line tension using such a noisy gradient estimate? While the noise contaminating the gradient estimate will cause the Dual D-MA algorithm to zig-zag rather than move directly towards the plant optimum, it is the *average* value of the gradient estimate which will determine the overall direction in which the algorithm adapts the RTO inputs. It can be observed that during the first 15 RTO iterations (from $k = 15$ to $k = 30$), the estimate of the cost gradient is on average *positive* in the u_1' direction, and *negative* in the u_2' direction. Thus the Dual D-MA algorithm *reduces* u_1' and *increases* u_2' in order to reduce the cost function. Between $k = 30$ and $k = 40$, the average value of the gradient estimate is approximately zero, which explains why the RTO algorithm stays at roughly the same place.

In order to verify that the RTO algorithm did indeed converge to a neighborhood of the plant optimum, an experimental study was carried out to see how the average line tension varies with respect to the path followed by the kite. This consisted of measuring the average line tension over 10 minutes for a variety of different paths. The

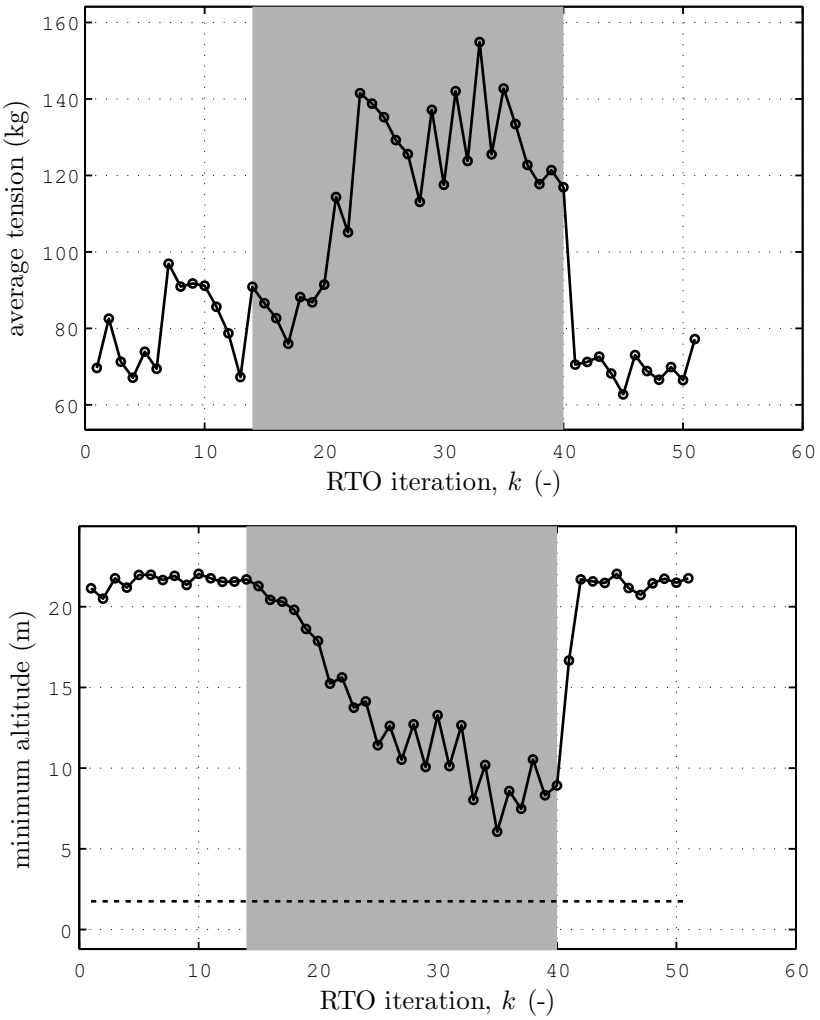

Figure 5.30: Performance of the RTO algorithm with $N_{\text{avg}} = 7$. Each circle is the average/minimum value for the tension/altitude during N_{avg} path cycles. The dotted line indicates the minimum height constraint. The RTO algorithm was activated during the shaded iterations. The total experiment lasted 29 minutes, and the RTO algorithm was active for 17 minutes.

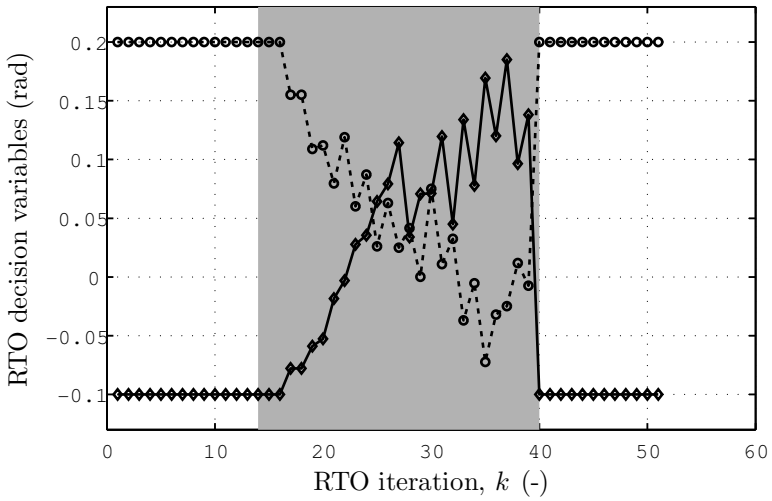

Figure 5.31: The RTO decision variable, u_1' (dashed), and u_2' (solid) during the experiment shown in Figure 5.30.

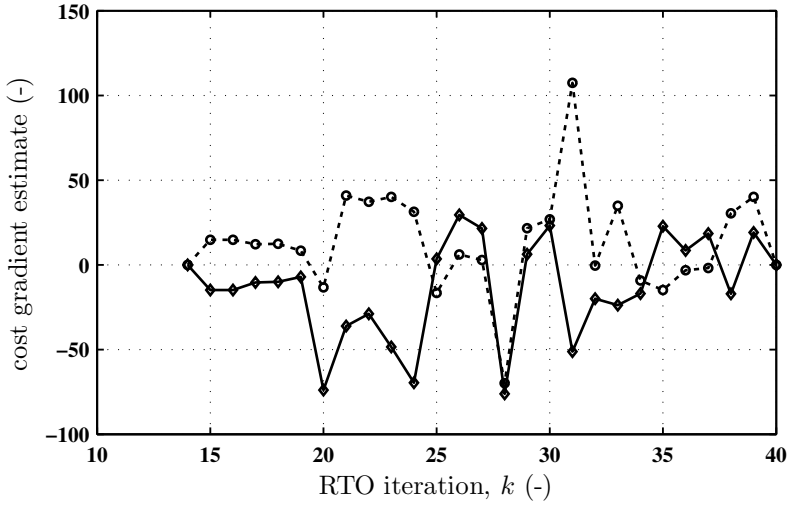

Figure 5.32: Estimate of the plant cost gradient, $\nabla\phi'_{\mathrm{E},k}$ during the experiment shown in Figure 5.30. Component in the u_1' direction (dashed) and in the u_2' direction (solid).

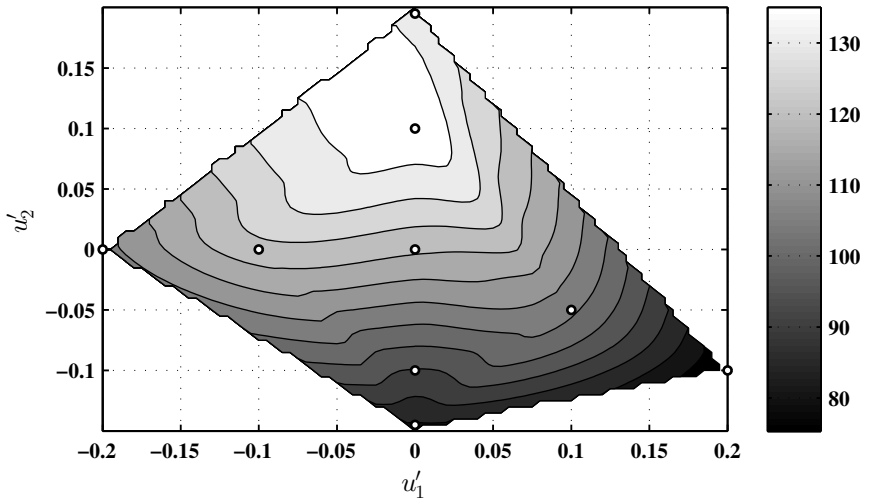

Figure 5.33: Contour plot of the average line tension in kg (shading) per figure-of-eight vs. $\mathbf{u}'$. At each of the data points (circles), the average line tension during 10 minutes of experimental data was recorded. The surface was estimated by performing a piecewise-cubic interpolation of these data points.

resulting surface is shown in Figure 5.33. Note that the average measured wind speed was relatively constant during the entire experimental study. Also, as it was carried out immediately after the experiment shown in Figure 5.30, the conditions were essentially the same as for the RTO experiment. It can be observed that the maximum attainable average line tension is about 130 - 140 kg. It is interesting to note that the nominal optimal solution that was calculated using the model, corresponding to $u_1' = u_2' = 0$, results in an average line tension of about 115 kg. Thus, following the nominal optimal path would result in an optimality loss of about 15-20 %. Figure 5.34 superimposes the Dual D-MA algorithm's path upon this contour. It can be seen that despite the zig-zagging behavior, the algorithm converges to the neighborhood of the plant optimum.

5.5 Conclusions

This chapter described not only the experimental application of the Dual D-MA algorithm developed in Chapter 4 to an autonomous kite prototype, but also the design choices involved in constructing the prototype, the testing conditions, the modeling process and the design of a path-following controller.

Based on this work, a number of interesting conclusions can be drawn regarding the control and optimization of power kites. Firstly, while additional measurements are certainly useful, good path-following control (of a kite flying roughly crosswind) can

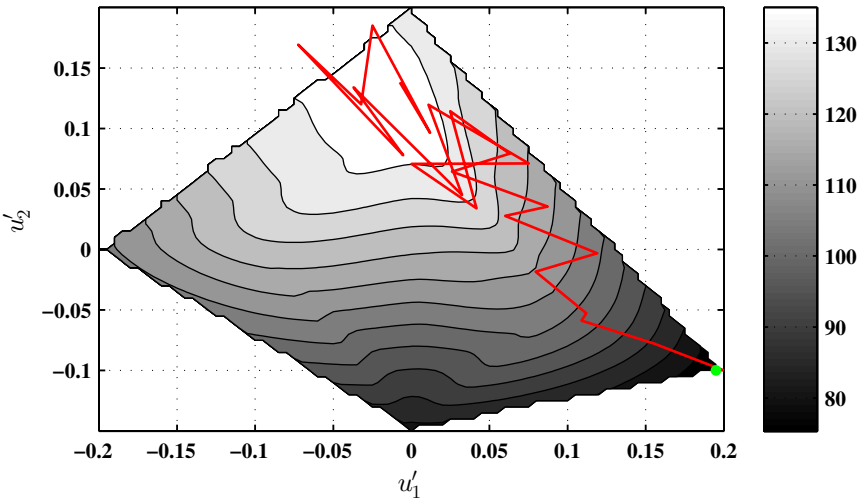

Figure 5.34: Contour plot of the attainable average line tension, as shown in Figure 5.33. The path taken by the Dual D-MA algorithm (red), and the algorithm's initial point (green dot).

be achieved using only measurements of the kite's position. This is the case even if there is significant system delay. The path-following performance could probably be improved using techniques such as Iterative Learning Control, but it is doubtful whether this is necessary. Secondly, the average line tension varies significantly depending on the path followed by the kite, to the extent that serious attention should be given to the kite's path if efficiency is to be maximized. It is doubtful whether model-based optimization alone can calculate an efficient path. Indeed, in this study, using the model's optimal path would have resulted in a 15-20 % reduction of the average line tension, if compared to the plant's optimal path, despite calibrating the model using experimental data. It appears that a better solution would be to either fix the kite's path based on experimental studies, or to perform RTO if the path needs to be constantly updated in response to process variations.

The Dual D-MA algorithm applied here is admittedly a slightly simplified version of the full algorithm presented in Chapter 4. Simulations predicted that this simplification might lead to a slight optimality loss compared to the full algorithm, but would render it more robust to the extremely high level of noise present during the experiments. In addition the computational power required by the simplified algorithm is roughly an order of magnitude lower than that required by the full algorithm. The RTO method was found to perform surprisingly well, given the high noise levels. This is one of the first measurement-based experimental RTO studies in the literature to rely entirely upon gradient estimates to optimize the plant, as the minimum-altitude constraint was not active upon convergence.

Finally, the information given here, if combined with some basic engineering knowledge, should be sufficient to construct a basic autonomous kite system. The main *practical* observation that was made during the entire process, is that the field-testing is at least as difficult and time consuming as constructing the physical system. A large number of elements must fall into place for a successful field-test: good weather conditions, access to a suitable terrain, no equipment failures, no software failures, no algorithmic 'bugs'. It was particularly important to ensure no algorithmic 'bugs' occurred, as debugging the controller code in the middle of a field, on a tiny laptop screen, in sub-zero temperatures while exposed to howling wind, is almost impossible. As the control and RTO algorithms were being changed before every field test, the only solution was to rigorously verify all code on a realistic simulator prior to each test.

6 Conclusions and Perspectives

6.1 Conclusions

6.1.1 Methodology

This thesis has demonstrated, both theoretically and experimentally, that the Modifier-Adaptation (MA) concept can be applied to a process with many inputs, such as a repetitive, transient process.

The first methodological development proposed here allows MA to be applied to processes where the inputs of the plant and the model differ. The motivating examples of an incineration plant and a general closed-loop system, show that this situation may often occur. The proposed 'Generalized' MA algorithm preserves the attractive property of achieving plant optimality upon convergence.

The second methodological development allows the MA concept to be applied to processes with many inputs. Full MA for a system with many inputs is probably unrealistic. Firstly, during the time that would be spent estimating the experimental gradients, the process may evolve. Secondly, the difficulty of estimating experimental gradients increases along with the number of inputs. In order to deal with the gradient estimation problem, it is proposed to estimate gradients only in certain 'privileged' directions. The resulting practically-applicable algorithm, Dual Directional-MA, is designed to converge rapidly and to cope with process noise.

Experimental application to a complex system shows that the algorithm is implementable and can perform well in practice, despite extremely high noise levels. It appears that this is the second experimental application of a Modifier Adaptation methodology, and the first to a transient process. The application considered here is also one of the first in the MA literature where experimental gradients are crucial

for RTO. No constraints are active at the plant optimal solution, meaning that classic constraint-control approaches would fail[1]. In the presence of noise, experimental gradient estimation, even with respect to a low number of inputs, is very challenging. While the situation considered here was rather extreme (up to 35% noise), it cannot be called abnormal, as very few other experimental studies have characterized noise from a RTO standpoint.

6.1.2 Application

Based on this work, a number of conclusions can be drawn relating to the control and optimization of kites. Firstly, this thesis demonstrates that it is possible to accurately control a kite during dynamic flight *using only measurements of the kite's position.* This is because, as demonstrated in this thesis, it is possible to determine the kite's attitude from its position with reasonable accuracy during crosswind flight. This distinguishes our approach from other control algorithms for kites that have very recently been published; they require the kite's attitude to be directly measured. Using only position measurements is significant because, in some cases, it may eliminate the need for sensors on the kite. This is because the kite's position can be estimated, with an accuracy dependent on the line length, from the line angles measured at the ground station.

As recently as 2012, there were no published accounts of experimental kite control, and it was often believed that a very advance control strategy such as NMPC would be necessary to control a dynamically flying kite. Here, it was demonstrated that good, robust path-following performance can be achieved with a relatively simple guidance strategy. Although the experiments never lasted more than a day, and the aim was not to achieve endless autonomous flight, at times the kite flew autonomously for 4 hours continuously. Indeed, in reasonable wind conditions, there is no reason the control algorithm would not keep the kite flying indefinitely. Due to the high speed of kites, delay negatively impacts control performance. It is thus essential to compensate for delay if the kite's direction of motion is to be controlled accurately. Prediction using a simple model was found to be sufficient for delay compensation; this made the difference between robust path following and constant crashing.

Although theoretical analyses already exist, here the effect of path geometry upon the average line tension was studied experimentally. It was shown that the average line tension can vary by a factor of about 2, depending on the figure-of-eight followed by the kite. RTO did a good job of optimizing the average line tension, despite the

[1] Some lower-level constraints that are handled by the control layer, such as input bounds for the steering deflection, may be active at times. However, these constraints are not indicative of an optimal solution.

challenging, gusty conditions. When initialized at a poor initial path, it converged to the neighborhood of the optimal path in roughly 10 min. Overall, RTO resulted in a 15-20 % improvement compared to using the optimal solution calculated using the model.

6.2 Perspectives

6.2.1 Methodology

The MA family of methods is relatively developed from a theoretical perspective. At this point, it is necessary to apply the methodology to experimental systems. On the one hand, this will reveal where the real challenges lie, and provide motivation for further improving the method. For example, the experimental application in this thesis illustrates the difficulty of dealing with noise when estimating gradients. On the other hand, industrial practitioners will be more likely to adopt these methods if they have already been shown to work on real systems.

Experimental gradient estimation is most definitely the main difficulty facing MA, and a great many other RTO algorithms. For processes with many inputs, the challenge is to estimate gradients from sparse, noisy data points. Another challenge is estimating gradients while the underlying process is changing over time. For example, in the case of the kite, the wind speed may gradually increase. Using additional measurements might radically improve gradient estimates in this case. In the case of the kite, wind-speed measurements should indicate rising wind speeds. Essentially, it is necessary to attribute the observed effect to a cause: is the process performance improving due to the recent change in the RTO inputs? or is it due to a change in the disturbances affecting the process? If an approximate model of the effect of disturbances upon the system is available, it could probably be used to at least partially distinguish the effect of disturbance variations from that of variations in the RTO inputs.

The D-MA methodology has proven both in simulation and experimentally to be a capable RTO algorithm. Nonetheless, a number of methodological improvements could certainly be made. The privileged directions for experimental gradient estimation are currently calculated using a local sensitivity analysis around the nominal optimal solution. While this was found to yield good results on the case studies considered, there may be more 'global' ways of doing this. For example, in the field of SOC, simulation studies are used to determine the effect of parameter variations over a wide range of values (Skogestad, 2000). Another open question is how to translate operator experience into a set of privileged directions.

6.2.2 Application

The field of kite power is in a phase of intense evolution, and there are certainly very interesting perspectives for control and RTO in this field.

Control

This thesis experimentally validates a path-following controller for kites. Although the control law is validated for a small-scale system, there is no obvious reason it could not be applied to a much larger kite with longer lines. The delay affecting the small-scale system is considerable with respect to the dynamics of the kite (the dynamics of a small kite are essentially faster, as it takes less time for the kite to travel a distance equal to its tether length). The delay compensation solution proposed here should be equally applicable to large kite systems, where, due to transmission times from on-board sensors, delay is known to be an issue.

Even the approach of using line-angle measurements alone for control may be applicable to larger systems. This depends on how straight the kite's tether is, as only a straight tether gives a good estimation of the kite's position. While I suspect that this only applies to lines shorter than about 500 m, Jehle and Schmehl (2014) claim that even for a 1 km long tether, "during powered flight, the tether is an almost perfectly straight line".

Several other experimentally validated kite controllers have recently been published. Notwithstanding the technical issues unique to each particular system, it is fair to say that the basic problem of autonomously flying figure-of-eights has essentially been solved at this point. However, it is undoubtedly considerably more difficult to keep a kite in the air for very long periods of time, during a wide variety of wind conditions, without ever losing control due to a lull in the wind, or suffering a system failure due to excess forces during a gust. A number of interesting issues remain to be tackled:

- **Pitch control:** kites typically have two degrees of freedom for control. Up until now, development has focused on the 'steering' input, which is usually the difference between the lengths of the rear lines. However, the kite's pitch angle relative to the lines can also be adjusted, usually by changing the length of the front line relative to that of the rear lines. This additional degree of freedom is already used when manually controlling kites; it directly influences the kite's angle of attack, which has a profound effect upon the line tension, the kite's speed, and the kite's turning behavior. The first step to exploiting this degree of freedom is to establish experimentally-validated models describing the influence of the kite's angle of attack, which are currently not available.

- **Robustness** is no-doubt a key characteristic of a kite controller. A large kite simply cannot be permitted to crash. Even small sports kites are banned on many beaches in Europe due to the damage a crash can cause. Ideally a robust control law should guarantee that, at all times, the kite can recover from any possible wind variation. There are certain configurations in which the kite is more vulnerable to such disturbances. For example, when flying at the edge of the wind window (when the tether makes a large angle with the wind vector), a sudden drop in the wind can result in the kite entering an unexpected dive. Alternatively, a high angle of attack combined with a large steering deflection can cause the kite to stall. One approach would be to follow the lead of aircraft designers and specify a 'flight envelope', which defines an acceptable operating region inside which the kite must operate.

- **Pumping-cycle control:** full pumping-cycle operation of a kite introduces further control challenges: the kite is now coupled to a winch. The kite controller should take the winch's dynamics and operational limits into account. In addition to the traction phase, a completely different control strategy is required for the reel-in phase, launching and landing.

Real-Time Optimization

The RTO algorithm presented here could straight forwardly be applied to a kite operating on a fixed-length tether, pulling a boat. Applying RTO to a pumping-kite generator would be considerably more difficult, as the line length is constantly changing. Several improvements could be envisaged, which would enhance the RTO algorithm's performance for fixed line-length flight, and hopefully make it applicable for variable line-length flight.

The theoretical analysis of the kite model revealed two optimization trade-offs which must be considered when choosing the kite's path: a) aggressive turning reduces the tether tension, yet a path with too gentle a curvature involves the kite flying through low-power regions of the wind window, b) flying higher takes advantage of the wind shear (increase with altitude), yet flying lower allows the kite to fly at a more favorable angle to the wind. An improved RTO scheme could be based upon experimentally identifying the wind shear and the kite's turning behavior during flight. For example, the kite's speed at different altitudes could, in theory, be used to estimate the difference in wind speed between those altitudes.

Finally, a host of interesting optimization problems must be solved for a pumping-cycle generator. What is the optimal path to ensure a maximally-steady power production?

What is the ideal length of each pumping cycle? The optimal mode of operation may change significantly when the winch efficiency, the conversion electronics, and the demands of the electricity distribution grid are taken into account.

Small-scale autonomous kites

The field of kite power is currently completely focused on the commercialization of *large* power-producing kites. To the best of my knowledge, the prototype described in this thesis is the smallest, simplest autonomous kite system constructed to date. It demonstrates that only a motor, two angular encoders, and a modest amount of processing power, would be sufficient to control a small kite. The system described here was a prototype, and no particular effort was made to save money or materials during the design. If more attention was payed to the design, it would no-doubt be possible to build a very compact 10 kg ground station, capable of autonomously flying kites up to 10 m^2, at a cost of several thousand euros. This size kite is cheap and extremely wide-spread among the sport-kiting community. Given the very high ratio of traction force to surface area compared to a traditional sail, and the fact that it does not require a supporting mast, it is possible that some time in the future sailing yachts, or motor boats, will carry a small autonomous kite for downwind sailing!

A CSTR Balance Equations

The 3-reaction simulated reality (plant) is governed by (Marchetti, 2009; Zhang and Forbes, 2000):

$$0 = F_A - (F_A + F_B)X_A - Wr_1, \tag{A.0.1}$$

$$0 = F_B - (F_A + F_B)X_B - \frac{M_B}{M_A}Wr_1 - Wr_2, \tag{A.0.2}$$

$$0 = -(F_A + F_B)X_C + \frac{M_C}{M_A}Wr_1 - \frac{M_C}{M_B}Wr_2 - Wr_3, \tag{A.0.3}$$

$$0 = -(F_A + F_B)X_P + \frac{M_P}{M_B}Wr_2 - \frac{M_P}{M_C}Wr_3, \tag{A.0.4}$$

$$0 = -(F_A + F_B)X_G + \frac{M_G}{M_C}Wr_3, \tag{A.0.5}$$

$$X_E = \frac{M_E}{M_P}X_P + \frac{M_E}{M_G}X_G, \tag{A.0.6}$$

with

$$r_1 = k_1 X_A X_B, \tag{A.0.7}$$

$$r_2 = k_2 X_B X_C, \tag{A.0.8}$$

$$r_3 = k_3 X_C X_P. \tag{A.0.9}$$

The model equations encompassing two reactions are:

$$0 = F_A - (F_A + F_B)X_A - Wr_1 - Wr_2, \tag{A.0.10}$$

$$0 = F_B - (F_A + F_B)X_B - \frac{M_B}{M_A}2Wr_1 - \frac{M_B}{M_A}Wr_2, \tag{A.0.11}$$

$$0 = -(F_A + F_B)X_P + \frac{M_P}{M_A}Wr_1 - \frac{M_P}{M_A}Wr_2, \tag{A.0.12}$$

$$0 = -(F_A + F_B)X_E + \frac{M_E}{M_A}Wr_1, \tag{A.0.13}$$

$$X_G = \frac{M_G}{M_E}X_E + \frac{M_G}{M_P}X_P, \tag{A.0.14}$$

with

$$r_1 = k_1 X_A X_B^2, \tag{A.0.15}$$

$$r_2 = k_2 X_A X_B X_P. \tag{A.0.16}$$

By assuming $M_A = M_B = M_P$, all the molecular weight ratios X_i are defined from the stoichiometry of the reactions.

B Closed-loop Kite Model

The dynamic equations of the Erhard Model allow the evolution of the kite's states, and the path it follows, to be calculated based on the steering deflection, i.e. the steering deflection is specified a priori. It is now shown how to calculate the kite's states, and the steering deflection, based on the path followed by the kite, i.e. the kite's path is specified a priori. Doing so involves the use of some approximations, however these remain reasonable, and it cannot be said that they introduce inaccuracies as, in any case, the Erhard Model is a simple approximate model in the first place.

The case considered here, which corresponds to the optimization decision variables used in Chapters 4 and 5, is that N points on the kite's path, $\{\vartheta(l), \varphi(l)\}$ for $l \in [0,1]$ are specified by the N pairs $\{\vartheta_j, \varphi_j\}$, where

$$\vartheta_j = \vartheta\left(\frac{j}{N}\right), \quad j = 0, 1, \ldots, N-1. \tag{B.0.1}$$

The average values of ϑ and φ between points j and $j+1$ are:

$$\vartheta'_j = \frac{\vartheta_j + \vartheta_{j+1}}{2}, \qquad \varphi'_j = \frac{\varphi_j + \varphi_{j+1}}{2}. \tag{B.0.2}$$

The velocity angle between points j and $j+1$ is given by:

$$\gamma'_j = \tan^{-1}\left(\frac{(\varphi_{j+1} - \varphi_j)\sin\vartheta'_j}{\vartheta_{j+1} - \vartheta_j}\right), \tag{B.0.3}$$

and the distance from each point to the next is:

$$d'_j = r\sqrt{\left((\varphi_{j+1} - \varphi_j)\sin\vartheta'_j\right)^2 + \left(\vartheta_{j+1} - \vartheta_j\right)^2}. \tag{B.0.4}$$

The curvature of the path, κ_j, i.e. the rate-of-change of γ, in rad·m^{-1} is given by:

$$\kappa_j = \frac{\gamma'_j - \gamma'_{j-1}}{\frac{d'_j + d'_{j-1}}{2}}. \tag{B.0.5}$$

Now, as $\dot{\gamma} \simeq w_{\text{ap}} g_{\text{s}} \delta$, and w_{ap} is approximately equal to the kite's velocity during cross-wind flight, $\kappa \simeq \frac{\dot{\gamma}}{w_{\text{ap}}} = g_{\text{s}}\delta$. The steering deflection can thus be deduced from the path curvature

$$\delta_j = \frac{\kappa_j}{g_{\text{s}}}. \tag{B.0.6}$$

The resulting lift/drag ratio is

$$E_j = E_0 - c\delta_j^2, \tag{B.0.7}$$

where E_0 is the kite's lift/drag ratio when $\delta = 0$ (*not* the lift/drag ratio at the point $j = 0$!).

The kites' altitude is

$$z_j = r \sin\vartheta_j \cos\varphi_j, \tag{B.0.8}$$

which allows the wind speed at each point, w_j, to be calculated using the wind-power law being used. The apparent wind projected onto the plane orthogonal to the tether is

$$w'_{\text{ap},j} = w'_j E'_j \cos\vartheta'_j, \tag{B.0.9}$$

which, being approximately equal to the kite's velocity during crosswind flight, allows the time taken by the kite to travel from point j to point $j+1$ to be calculated:

$$t'_j = \frac{d'_j}{w'_{\text{ap},j}}. \tag{B.0.10}$$

The average line tension between points j and $j+1$ is approximately

$$T'_j = \left(\frac{1}{2}\rho A\right)\left(w'_j E'_j \cos^2(\vartheta'_j)\right)^2 \tag{B.0.11}$$

Finally, the average line tension over the entire path is

$$\bar{T} = \frac{\sum_{j=0}^{j=N-1} T'_j t'_j}{\sum_{j=0}^{j=N-1} t'_j} \tag{B.0.12}$$

Note, that in the above derivation, any values of $0 > j > N-1$ should be replaced by

$\mathrm{mod}(j, N-1)$, which enforces periodicity of the path.

Bibliography

M. Agarwal. Feasibility of on-line reoptimization in batch processes. *Chem. Eng. Communications*, 158(1):19–29, 1997.

U. Ahrens, M. Diehl, and R. Schmehl, editors. *Airborne Wind Energy*. Springer, Berlin, 2013.

V. Alstad and S. Skogestad. Null space method for selecting optimal measurement combinations as controlled variables. *Ind. Eng. Chem. Res.*, 46(3):846–853, 2007.

C. L. Archer. An introduction to meteorology for airborne wind energy. In Ahrens et al. (2013), pages 81–94.

C. L. Archer and K. Caldeira. Global assessment of high-altitude wind power. *Energies*, 2 (2):307–319, 2009.

C. L. Archer and M. Z. Jacobson. Evaluation of global wind power. *J. Geophysical Research: Atmospheres*, 110(D12), 2005.

I. Argatov and R. Silvennoinen. Energy conversion efficiency of the pumping kite wind generator. *Renewable Energy*, 35(5):1052–1060, 2010.

J. Baayen and W. Ockels. Tracking control with adaption of kites. *Control Theory App., IET*, 6(2):182–191, 2012.

L. Bodizs, M. Titica, N. Faria, B. Srinivasan, D. Dochain, and D. Bonvin. Oxygen control for an industrial pilot-scale fed-batch filamentous fungal fermentation. *J. Process Control*, 17(7):595–606, 2007.

D. Bonvin, B. Srinivasan, and D. Ruppen. Dynamic optimization in the batch chemical industry. In *Proc. of the CPC-VI Conference: AIChE Symposium Series N. 326*, pages 255–273, 2002.

A. Bosch, R. Schmehl, P. Tiso, and D. Rixen. Nonlinear aeroelasticity, flight dynamics and control of a flexible membrane traction kite. In Ahrens et al. (2013), pages 307–323.

G. E. Box and N. R. Draper. *Evolutionary Operation: A Statistical Method for Process Improvement*. Wiley, NY, 1969.

J. Breukels. *An Engineering Methodology for Kite Design*. PhD thesis, Delft Universtiy of Technology, 2010.

J. Breukels, R. Schmehl, and W. Ockels. Aeroelastic simulation of flexible membrane wings based on multibody system dynamics. In Ahrens et al. (2013), pages 287–305.

G. A. Bunin. On the equivalence between the modifier-adaptation and trust-region frameworks. *Comp. Chem. Eng.*, 71:154–157, 2014.

G. A. Bunin, Z. Wuillemin, G. François, A. Nakajo, L. Tsikonis, and D. Bonvin. Experimental real-time optimization of a solid oxide fuel cell stack via constraint adaptation. *Energy*, 39(1):54–62, 2012.

G. A. Bunin, G. Francois, and D. Bonvin. From discrete measurements to bounded gradient estimates: A look at some regularizing structures. *Ind. Eng. Chem. Res.*, 52 (35):12500–12513, 2013.

M. Canale, L. Fagiano, and M. Milanese. High altitude wind energy generation using controlled power kites. *IEEE Tran. on Control Systems Tech.*, 18(2):279–293, 2010.

B. Chachuat, B. Srinivasan, and D. Bonvin. Adaptation strategies for real-time optimization. *Comp. Chem. Eng.*, 33(10):1557–1567, 2009.

C. Y. Chen and B. Joseph. On-line optimization using a two-phase approach: An application study. *Ind. Eng. Chem. Res.*, 26(9):1924–1930, 1987.

T. L. Clarke-Pringle and J. F. Mac Gregor. Optimization of molecular weight distribution using batch-to-batch adjustments. *Ind. Eng. Chem. Res.*, 37:3660–3669, 1998.

S. Costello, G. François, and D. Bonvin. Real-time optimization for kites. In *Proc. of the 5th IFAC Workshop on Periodic Control Systems (PSYCO)*, pages 64–69, 2013.

S. Costello, G. François, D. Bonvin, and A. G. Marchetti. Modifier adaptation for constrained closed-loop systems. In *Proc. IFAC World Congress*, volume 19, pages 11080–11086, 2014.

C. Cutler and R. Perry. Real time optimization with multivariable control is required to maximize profits. *Comp. Chem. Eng.*, 7(5):663–667, 1983. ISSN 0098-1354.

G. M. Dadd, D. A. Hudson, and R. A. Shenoi. Comparison of two kite force models with experiment. *J. of Aircraft*, 47:212–224, 2010.

G. M. Dadd, D. A. Hudson, and R. A. Shenoi. Determination of kite forces using three-dimensional flight trajectories for ship propulsion. *Renewable Energy*, 36(10):2667–2678, 2011.

M. L. Darby, M. Nikolaou, J. Jones, and D. Nicholson. RTO: An overview and assessment of current practice. *J. Process Control*, 21(6):874 –884, 2011.

V. de Oliveira, J. Jäschke, and S. Skogestad. Dynamic online optimization of a house heating system in a fluctuating energy price scenario. In *Proc. IFAC Symp. DYCOPS*, Mumbai, 2013.

S. Deshpande, D. Bonvin, and B. Chachuat. Directional input adaptation in parametric optimal control problems. *SIAM Journal on Control and Optimization*, 50(4):1995–2024, 2012.

M. Diehl. *Real Time Optimization for Large Scale Nonlinear Processes*. PhD thesis, Ruprecht-Karls-Universität Heidelberg, 2001.

S. Dunker. Ram-airwing design considerations for airborne wind energy. In Ahrens et al. (2013), pages 517–545.

M. Erhard and H. Strauch. Control of towing kites for seagoing vessels. *IEEE Tran. on Control Systems Tech.*, 21(5):1629–1640, 2013a.

M. Erhard and H. Strauch. Theory and experimental validation of a simple comprehensible model of tethered kite dynamics used for controller design. In Ahrens et al. (2013), pages 141–165.

L. Fagiano and M. Milanese. Airborne wind energy: An overview. In *Proc. American Control Conference (ACC)*, pages 3132–3143, 2012.

L. Fagiano, A. Zgraggen, M. Morari, and M. Khammash. Automatic crosswind flight of tethered wings for airborne wind energy: Modeling, control design, and experimental results. *IEEE Tran. on Control Systems Tech.*, 22(4):1433–1447, 2014.

T. Faulwasser and D. Bonvin. On the use of second-order modifiers for real-time optimization. In *Proc. IFAC World Congress*, volume 19, 2014.

A. V. Fiacco. *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*. Academic Press, NY, 1983.

C. Filippi-Bossy, J. Bordet, J. Villermaux, S. Marchal-brassely, and C. Georgakis. Batch reactor optimization by use of tendency models. *Comp. Chem. Eng.*, 13(1-2):35–47, 1989.

J. Forbes and T. Marlin. Design cost: A systematic approach to technology selection for model-based real-time optimization systems. *Comp. Chem. Eng.*, 20(6-7):717–734, 1996.

J. Forbes, T. Marlin, and J. MacGregor. Model adequacy requirements for optimizing plant operations. *Comp. Chem. Eng.*, 18(6):497–510, 1994.

G. François and D. Bonvin. Use of convex model approximations for real-time optimization via modifier adaptation. *Ind. Eng. Chem. Res.*, 52(33):11614–11625, 2013a.

G. François and D. Bonvin. Measurement-based real-time optimization of chemical processes. In S. Pushpavanam, editor, *Advances in Chemical Engineering*, Identification, Control and Optimisation of Proc. Sys. 43, pages 1–50. Academic Press, Waltham, 2013b.

G. François, B. Srinivasan, D. Bonvin, J. Hernandez Barajas, and D. Hunkeler. Run-to-run adaptation of a semiadiabatic policy for the optimization of an industrial batch polymerization process. *Ind. Eng. Chem. Res.*, 43(23):7238–7242, 2004.

G. François, B. Srinivasan, and D. Bonvin. Use of measurements for enforcing the necessary conditions of optimality in the presence of constraints and uncertainty. *J. Process Control*, 15(6):701–712, Sept. 2005.

G. François, B. Srinivasan, and D. Bonvin. Comparison of six implicit real-time optimization schemes. *J. Européen des Systemes Automatisés*, 46(2-3):291–305, 2012.

F. Fritz. Application of an automated kite system for ship propulsion and power generation. In Ahrens et al. (2013), pages 359–372.

W. Gao and S. Engell. Iterative set-point optimization of batch chromatography. *Comp. Chem. Eng.*, 29(6):1401–1409, 2005a.

W. Gao and S. Engell. Comparison of iterative set-point optimisation strategies under structural plant-model mismatch. In *Proc. IFAC World Congress*, volume 16, pages 401–401, 2005b.

M. Ge, Q. Wang, M. Chiu, T. Lee, C. Hang, and K. Teo. An effective technique for batch process optimization with application to crystallization. *Chem. Eng. Res. and Des.*, 78 (1):99–106, Jan. 2000.

F. Gohl and R. H. Luchsinger. Simulation based wing design for kite power. In Ahrens et al. (2013), pages 325–338.

S. Gros and M. Diehl. Modeling of airborne wind energy systems in natural coordinates. In Ahrens et al. (2013), pages 181–203.

G. Horn, S. Gros, and M. Diehl. Numerical trajectory optimization for airborne wind energy systems described by high fidelity aircraft models. In Ahrens et al. (2013), pages 205–218.

B. Houska and M. Diehl. Optimal control of towing kites. In *Proc. 45th IEEE Conference on Decision and Control (CDC)*, pages 2693–2697, 2006.

B. Houska and M. Diehl. Optimal control for power generating kites. In *Proc. 9th European Control Conference*, pages 3560–3567, 2007.

A. Ilzhöfer, B. Houska, and M. Diehl. Nonlinear MPC of kites under varying wind conditions for a new class of large-scale wind power generators. *Int. J. Robust and Nonlinear Control*, 17(17):1590–1599, 2007.

International Energy Agency. *World Energy Outlook 2014*. IEA Publications, Paris, 2014.

S. Jang, B. Joseph, and H. Mukai. On-line optimization of constrained multivariable chemical processes. *AIChE J.*, 33(1):26–35, 1987.

J. Jäschke, M. Fikar, and S. Skogestad. Self-optimizing invariants in dynamic optimization. In *Proc. 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 7753–7758, 2011.

C. Jehle and R. Schmehl. Applied tracking control for kite power systems. *J. Guidance, Control, and Dynamics*, 37(4):1211–1222, 2014.

J. V. Kadam, W. Marquardt, B. Srinivasan, and D. Bonvin. Optimal grade transition in industrial polymerization processes via NCO tracking. *AIChE J.*, 53(3):627–639, 2007.

D. V. Lind. Analysis and flight test validation of high performance airborne wind turbines. In Ahrens et al. (2013), pages 473–491.

M. L. Loyd. Crosswind kite power. *J. Energy*, 4(3):106–111, May 1980.

D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. Springer, 2008.

M. Mansour and J. E. Ellis. Comparison of methods for estimating real process derivatives in on-line optimization. *Applied Mathematical Modelling*, 27(4):275–291, 2003.

A. Marchetti, B. Chachuat, and D. Bonvin. Batch process optimization via run-to-run constraints adaptation. In *Proc. of the European Control Conference*, 2007.

A. Marchetti, B. Chachuat, and D. Bonvin. Modifier-adaptation methodology for real-time optimization. *Ind. Eng. Chem. Res.*, 48(13):6022–6033, 2009.

A. Marchetti, B. Chachuat, and D. Bonvin. A dual modifier-adaptation approach for real-time optimization. *J. Process Control*, 20(9):1027–1037, 2010.

A. G. Marchetti. *Modifier-Adaptation Methodology for Real-Time Optimization*. PhD thesis, # 4449, EPFL, Lausanne, 2009.

A. G. Marchetti. A new dual modifier-adaptation approach for iterative process optimization with inaccurate models. *Comp. Chem. Eng.*, 59:89–100, 2013.

T. E. Marlin and A. N. Hrymak. Real-time operations optimization of continuous processes. In *AIChE Symposium Series*, volume 93, pages 156–164, 1997.

D. Navia, R. Martí, D. Sarabia, G. Gutirrez, and C. de Prada. Handling infeasibilities in dual modifier-adaptation methodology for real-time optimization. In *Proc. 8th IFAC Symposium on Advanced Control of Chemical Processes*, pages 537–542, 2012.

D. Navia, G. Gutiérrez, and C. de Prada. Nested modifier-adaptation for RTO in the otto williams reactor. In *Proc. IFAC Symp. DYCOPS*, pages 123–128, 2013.

D. Navia, G. Gutierrez, and C. de Prada. Mixed modifier-adaptation for RTo in a continuous bioreactor. In *Proc. IFAC World Congress*, volume 19, pages 7635–7640, 2014.

D. Nelson, D. Barber, T. McLain, and R. Beard. Vector field path following for miniature air vehicles. *IEEE Tran. on Robotics*, 23(3):519–529, 2007.

X. Paulig, M. Bungart, and B. Specht. Conceptual design of textile kites considering overall system performance. In Ahrens et al. (2013), pages 547–562.

J. Proakis and D. K. Manolakis. *Digital Signal Processing*. Prentice Hall, New Jersey, 4$^{\text{th}}$ edition, 2006.

B. W. Roberts, D. H. Shepard, K. Caldeira, M. E. Cannon, D. G. Eccles, A. J. Grenier, and J. F. Freidin. Harnessing high-altitude wind power. *IEEE Tran. on Energy Conversion*, 22(1):136–144, 2007.

P. D. Roberts. An algorithm for steady-state system optimization and parameter estimation. *Int. J. Systems Sci.*, 10(7):719–734, 1979.

P. D. Roberts. Coping with model-reality differences in industrial process optimisation. a review of integrated system optimisation and parameter estimation (ISOPE). *Computers in Industry*, 26(3):281–290, 1995.

E. A. Rodger and B. Chachuat. Design methodology of modifier adaptation for on-line optimization of uncertain processes. In *Proc. IFAC World Congress*, pages 4113–4118, 2011.

R. Ruiterkamp and S. Sieberling. Description and preliminary test results of a six degrees of freedom rigid wing pumping system. In Ahrens et al. (2013), pages 443–458.

D. Ruppen, D. Bonvin, and D. Rippin. Implementation of adaptive optimal operation for a semi-batch reaction system. *Comp. Chem. Eng.*, 22(1–2):185–199, 1998.

F. J. Serralunga, M. C. Mussati, and P. A. Aguirre. Model adaptation for real-time optimization in energy systems. *Ind. Eng. Chem. Res.*, 52(47):16795–16810, 2013.

F. J. Serralunga, P. A. Aguirre, and M. C. Mussati. Including disjunctions in real-time optimization. *Ind. Eng. Chem. Res.*, 53(44):17200–17213, 2014.

S. Skogestad. Plantwide control: The search for the self-optimizing control structure. *J. Process Control*, 10:487–507, 2000.

SKP. http://www.swisskitepower.ch.

Skysails GmBH. http://www.skysails.info.

B. Srinivasan and D. Bonvin. Real-time optimization of batch processes by tracking the necessary conditions of optimality. *Ind. Eng. Chem. Res.*, 46(2):492–504, 2007.

B. Srinivasan, D. Bonvin, E. Visser, and S. Palanki. Dynamic optimization of batch processes II. role of measurements in handling uncertainty. *Comp. Chem. Eng.*, 27(1): 27–44, 2003a.

B. Srinivasan, S. Palanki, and D. Bonvin. Dynamic optimization of batch processes: I. characterization of the nominal solution. *Comp. Chem. Eng.*, 27(1):1–26, 2003b.

P. Tatjewski. Iterative optimizing set-point control-the basic principle redesigned. In *Proc. IFAC World Congress*, pages 992–992, 2002.

O. Ubrich, B. Srinivasan, P. Lerena, D. Bonvin, and F. Stoessel. Optimal feed profile for a second order reaction in a semi-batch reactor under safety constraints: Experimental study. *J. of Loss Prevention in the Process Industries*, 12(6):485–493, 1999.

G. P. Van den Berg. Wind gradient statistics up to 200 m altitude over flat ground. In *Proc. 1st International Meeting on Wind Turbine Noise*, 2005.

R. van der Vlugt, J. Peschel, and R. Schmehl. Design and experimental characterization of a pumping kite power system. In Ahrens et al. (2013), pages 403–425.

E. Visser, B. Srinivasan, S. Palanki, and D. Bonvin. A feedback-based implementation scheme for batch process optimization. *J. Process Control*, 10(5):399–410, 2000.

C. Welz, B. Srinivasan, and D. Bonvin. Measurement-based optimization of batch processes: Meeting terminal constraints on-line via trajectory following. *J. Process Control*, 18(3-4):375–382, 2008.

P. Williams, B. Lansdorp, and W. Ockesl. Optimal crosswind towing and power generation with tethered kites. *J. Guidance, Control, and Dynamics*, 31(1):81–93, 2008.

T. J. Williams and R. E. Otto. A generalized chemical processing model for the investigation of computer control. *Trans. of the American Inst. of Elec. Engineers, Part I: Communication and Electronics*, 79(5):458–473, 1960.

Z. Xiong and J. Zhang. A batch-to-batch iterative optimal control strategy based on recurrent neural network models. *J. Process Control*, 15(1):11–21, 2005.

E. Zafiriou and J. Zhu. Optimal control of semi-batch processes in the presence of modeling error. In *American Control Conference*, pages 1644–1649, 1990.

A. Zgraggen, L. Fagiano, and M. Morari. Real-time optimization and adaptation of the crosswind flight of tethered wings for airborne wind energy. 2013. arXiv:1310.0586.

A. U. Zgraggen, L. Fagiano, and M. Morari. Automatic retraction and full cycle operation for a class of airborne wind energy generators. 2014. ArXiv:1409.6151.

Y. Zhang and J. Forbes. Extended design cost: A performance criterion for real-time optimization systems. *Comp. Chem. Eng.*, 24(8):1829–1841, 2000.

Curriculum Vitae

Sean Costello
Nationality: Irish
D.O.B: 11.04.1987
Address: Avenue de Marcelin 7, Morges
Tel: +41 774546 2444
Email: sean.c.costello@gmail.com

EDUCATION

2009 to date — **PHD** in Optimization and Automation, Automatic Control Lab, EPFL.

2005-2009: — **BEng(Hons) in Electronic and Electrical Engineering,** at University College Dublin. S3 gold medal for the highest GPA of 2009 (4.16/4.20).

2007-2008: — Erasmus exchange year at EPFL.

EXPERIENCE

2010-2012: — **Project Leader for the Optimization of an Incineration Plant,** Tridel, Lausanne (Academia/Industry partenership project)**.**
- I lead a team of 4 students to propose measures to increase Tridel's revenue from energy sales by 5%.

2010 to date: — **Leader of the Autonomous Kite Power interdisciplinary project,** EPFL.
- Supervising a team of 2-3 students and interns over 2 years, with tasks ranging from mechanical design, automation, aerodynamic modelling and experimental testing.

2009: — **3-month Internship**, Nonlinear Optics Group, EPFL.

LANGUAGES

English: Native language.
French: Fluent
Italian: Fluent.
German: Basic.