A Directional Modifier-Adaptation Algorithm for Real-Time Optimization

Sean Costello* (*sean.costello@epfl.ch*),
Grégory François* (*gregory.francois@epfl.ch*),
Dominique Bonvin* (*dominique.bonvin@epfl.ch*).

* Laboratoire d'Automatique,
Ecole Polytechnique Fédérale de Lausanne.

December 28, 2014

Abstract

The steady advances of computational methods make 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. In addition, the resulting model may poorly represent the plant when there is structural mismatch between the two. The iterative optimization method called *Modifier Adaptation* overcomes these obstacles by directly incorporating plant measurements into the optimization framework, principally in the form of constraint values and gradients. However, the experimental cost (i.e. the number of experiments required) to estimate these gradients increases linearly with the number of process inputs, which tends to make the method intractable for processes with many inputs. This paper presents a new algorithm, called Directional Modifier Adaptation, that overcomes this limitation by only estimating the plant gradients in certain privileged directions. It is proven that plant optimality *with respect to these privileged directions* can be guaranteed upon convergence. A novel, statistically optimal, gradient estimation technique is developed. The algorithm is illustrated through the simulation of a realistic airborne wind-energy system, a promising renewable energy technology that harnesses wind energy using large kites. It is shown that Directional Modifier Adaptation can optimize in real time the path followed by the dynamically flying kite.

1 Introduction

Industrial processes have a certain number of degrees of freedom, the values of which are chosen by operators to meet safety requirements and operating constraints and to optimize process performance. Real-time optimization aims to determine the optimal values of these degrees of freedom, and then to continually update them in response to disturbances and process variations.

We start with a quick review of the field of Real-Time Optimization (RTO) over the past 50 years, with a particular focus on the development of a technique called Modifier Adaptation (MA). 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' in which to move. 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 decision variables, process constraints and complex nonlinear plant behavior. A more recent model-free technique is Self Optimizing Control (SOC) (Skogestad, 2000; Alstad and Skogestad, 2007), which uses 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 degrees of freedom, it is fairly computation 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 Agarwal (1997), Gao and Engell (2005b) and Marchetti (2009) showed that, in the presence of structural plant-model mismatch, parameter estimation is 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 an 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 algorithms, 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 this gradient modifier 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 a MA algorithm that has now 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). Many aspects of MA have been investigated further, such as approaches to deal with the estimation of gradients (Bunin et al., 2013a; 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 optimization and the convergence to the plant optimum (François and Bonvin, 2013), 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).

What are the challenges currently facing RTO? The following discussion is largely based on an excellent review of the challenges facing RTO in industry today (Darby et al., 2011). It is estimated that there are at least 250-300 RTO implementations in industrial plants. RTO is particularly beneficial for plants involving operational or economic trade-offs, or large product price differentials. It is estimated that, for a large plant, the benefits of RTO can be up to 50 % of the benefits obtained from implementing advanced process control. Increased global competition calls for more effective and easier-to-implement RTO algorithms than the current state-of-the-art. Improved RTO algorithms should address the following issues:

1. *Constraint satisfaction* is of paramount importance, as violating constraints often has harsh economic consequences. The two-step approach, which represents the industry standard, cannot guarantee the satisfaction of operational constraints as these may be poorly predicted by a structurally incorrect model.

2. *Online diagnostics.* It is important to know why the RTO algorithm takes certain steps, and whether it has in fact reached the plant optimum. Again, due to the possibility of a structurally incorrect model, the two-step approach may not satisfy any optimality measure for the plant (Forbes et al., 1994). It may even do worse than simply applying the nominal optimal solution! In contrast, MA supplies an estimate of the plant gradient, which can be used to verify the optimality of the current operating point.

3. *Convergence speed.* A certain settling time must be respected between

set-point changes. The typical assumption in RTO is that disturbances and parameter drifts occur slowly with respect to both this settling time and the time taken by the RTO algorithm to converge. For example, RTO can easily handle disturbances that vary on a daily basis for a plant with a 30-minute settling time.

4. *RTO synthesis.* The RTO design process should be fairly methodological and straightforward, as the person implementing RTO may not have a detailed knowledge of the process. This point is particularly important as the majority of the modern RTO algorithms are reportedly based on rigorous process models (Darby et al., 2011).

In addition, we claim that it is desirable to develop RTO methods that do not require frequent online parameter estimation. While RTO based on parameter estimation may work well in some situations, namely when there are few uncertain parameters, it is not always a good solution. Parameter estimation is certainly useful as it improves the quality of the rigorous model, which may then be used for other offline investigations. 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 optimization objective of the RTO layer. If the model contains many uncertain parameters, it is difficult to ensure sufficient excitation to estimate these parameters, and doing so will detract from reaching the optimization objective.

With the preceding motivation in mind, this paper presents a novel RTO method, called *Directional Modifier Adaptation*(D-MA), with 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 degrees-of-freedom is guaranteed upon convergence, despite the use of an inaccurate model.

3. *Rapid convergence* is enforced, even in the presence of significant measurement noise. The convergence speed of the RTO algorithm is independent of the number of plant degrees-of-freedom.

4. *Straightforward design procedure* using the available model.

The paper is structured as follows. Section 2 briefly reviews the MA family of algorithms, Section 3 presents the novel D-MA algorithm and examines its properties, while Section 4 presents a dual D-MA algorithm, which simultaneously estimates the plant gradient and searches for the plant optimum. This algorithm is applied to the challenging problem of optimizing the flight path of a power-generating kite in Section 5. Finally, Section 6 concludes the paper and provides a prospective outlook.

2 RTO via Modifier Adaptation

2.1 MA for steady-state optimization

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

$$\begin{aligned} &\mathbf{u}_\mathrm{p}^* := \arg\min_{\mathbf{u}} \phi_\mathrm{p}(\mathbf{u}) \\ &\text{subject to} \quad \mathbf{g}_\mathrm{p}(\mathbf{u}) \leq \mathbf{0}\,, \end{aligned} \tag{2.1}$$

where $\mathbf{u}$ is the n_u-dimensional vector of inputs, ϕ_p the cost function and $\mathbf{g}_\mathrm{p}$ the n_g-dimensional vector of process constraints. Here, the subscript $(\cdot)_\mathrm{p}$ indicates a quantity related to the plant, and we will refer to this as the plant optimization problem. We will assume in this paper that ϕ_p and $\mathbf{g}_\mathrm{p}$ are continuously differentiable, although we note that an extension to certain classes of discontinuous processes is given in Serralunga et al. (2014).

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) is obtained by solving the following model-based problem:

$$\begin{aligned} &\mathbf{u}^*(\boldsymbol{\theta}) := \arg\min_{\mathbf{u}} \phi(\mathbf{u}, \boldsymbol{\theta}) \\ &\text{subject to} \quad \mathbf{g}(\mathbf{u}, \boldsymbol{\theta}) \leq \mathbf{0}, \end{aligned} \tag{2.2}$$

where $\boldsymbol{\theta}$ is the n_θ-dimensional vector of uncertain model parameters. If the model matches the plant perfectly, solving Problem (2.2) provides a solution to Problem (2.1). Unfortunately, this is rarely the case, since the structure of the model functions ϕ and $\mathbf{g}$ as well as the nominal values for the uncertain model parameters $\boldsymbol{\theta}_0$ are likely to be incorrect, which implies that the nominal model-based optimal input $\mathbf{u}^*(\boldsymbol{\theta}_0)$ will not correspond to $\mathbf{u}_\mathrm{p}^*$.

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:

$$\tilde{\phi}_\mathrm{p}(\mathbf{u}_i) = \phi_\mathrm{p}(\mathbf{u}_i) + d_i^\phi \tag{2.3}$$

$$\tilde{g}_{\mathrm{p},j}(\mathbf{u}_i) = g_{\mathrm{p},j}(\mathbf{u}_i) + d_i^{g,j}, \forall j \in [0, \ldots, n_g] \tag{2.4}$$

where d_k^ϕ and $d_k^{g,j}$ are realizations of zero-mean random variables 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 disturbances affecting the plant. The plant measurements are used to iteratively modify the model-based problem (2.2) 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). 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: Modifier Adaptation (Marchetti et al., 2009)

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

for $k = 1 \to \infty$

1. Solve the modified model-based optimization problem

$$\begin{aligned} \mathbf{u}_k := \underset{\mathbf{u}}{\operatorname{argmin}} \quad & \phi_{\mathrm{m},k-1}(\mathbf{u}) \\ \text{subject to} \quad & \mathbf{g}_{\mathrm{m},k-1}(\mathbf{u}) \leq \mathbf{0}, \end{aligned} \tag{2.5}$$

   where the modified cost and constraints are given by

$$\phi_{\mathrm{m},k}(\mathbf{u}) := \phi(\mathbf{u}, \boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_k^{\phi})^T (\mathbf{u} - \mathbf{u}_k), \tag{2.6}$$

$$\mathbf{g}_{\mathrm{m},k}(\mathbf{u}) := \mathbf{g}(\mathbf{u}, \boldsymbol{\theta}_0) + \boldsymbol{\epsilon}_k + (\boldsymbol{\lambda}_k^{\mathrm{g}})^T (\mathbf{u} - \mathbf{u}_k). \tag{2.7}$$

   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. Compute the estimates of the plant cost gradient, $\nabla \phi_{\mathrm{E},k}$, and of the plant constraint gradients, $\nabla \mathbf{g}_{\mathrm{E},k}$, for the *current* operating point $\mathbf{u}_k$, where the subscript E denotes an estimate. The gradients must be estimated using measurements collected at no less than n_u different operating points close to $\mathbf{u}_k$ (see Section 2.3).

4. Update the modifier terms using the measurements:

$$\boldsymbol{\epsilon}_k := (\mathbf{I}_{n_g} - \mathbf{K}^{\epsilon})\boldsymbol{\epsilon}_{k-1} + \mathbf{K}^{\epsilon}\left(\tilde{\mathbf{g}}_{\mathrm{p}}(\mathbf{u}_k) - \mathbf{g}(\mathbf{u}_k, \boldsymbol{\theta}_0)\right), \tag{2.8}$$

$$\boldsymbol{\lambda}_k^{\mathrm{g}} := (\mathbf{I}_{n_u} - \mathbf{K}^{\mathrm{g}})\boldsymbol{\lambda}_{k-1}^{\mathrm{g}} + \mathbf{K}^{\mathrm{g}}\left(\nabla \mathbf{g}_{\mathrm{E},k} - \nabla \mathbf{g}(\mathbf{u}_k, \boldsymbol{\theta}_0)\right)^T, \tag{2.9}$$

$$\boldsymbol{\lambda}_k^{\phi} := (\mathbf{I}_{n_u} - \mathbf{K}^{\phi})\boldsymbol{\lambda}_{k-1}^{\phi} + \mathbf{K}^{\phi}\left(\nabla \phi_{\mathrm{E},k} - \nabla \phi(\mathbf{u}_k, \boldsymbol{\theta}_0)\right)^T. \tag{2.10}$$

end

The exponential filter matrices $\mathbf{K}^{\epsilon}, \mathbf{K}^{\mathrm{g}}, \mathbf{K}^{\phi}$ are tuning parameters. In order for the MA algorithm to be stable and have a non-oscillatory response (although this cannot be guaranteed), they matrices are chosen with real, positive eigenvalues in the interval $(0, 1]$. The choice of the filter matrices is discussed in

Marchetti et al. (2009). As can be expected, with more filtering (smaller eigenvalues), the method is more likely to converge, but it will do so more slowly. These filters also partially eliminate the noise affecting the constraint measurements and the gradient estimates. If the MA scheme converges, then it will do so to the (local) plant optimum, provided the model satisfies some very relaxed adequacy conditions (Marchetti et al., 2009), which can be strictly enforced if ϕ and $\mathbf{g}$ are replaced by convex approximations (François and Bonvin, 2013). In reality, due to noise, the algorithm will converge to a neighborhood of the plant optimum.

2.2 MA for run-to-run optimization of dynamic processes

Static RTO methods can be applied to either continuous processes or semi-batch/periodic processes. In the case of a continuous process, the RTO scheme aims to find the optimal *steady-state values* for the plant inputs (which may actually correspond to set-points for lower-level controllers). If the process is operated in batch or semi-batch (i.e. transient) mode, there is obviously no steady state. However, one can parameterize the *time-varying inputs* and let the resulting input parameters become the decision variables. At the end of a batch run, the effect of the current input parameters (which act on the dynamic process via the corresponding input profiles) on the cost and constraints can be determined. This way, a batch run is assimilated to a RTO iteration, and static RTO can be used to compute optimal input parameters, which generate optimal input profiles.

While RTO methods have primarily been developed for the more widespread continuous processes, there is also a significant interest in applying RTO to transient processes, and the process engineering literature is rich with applications to batch and semi-batch chemical processes for fine chemicals (Ruppen et al., 1998; Filippi-Bossy et al., 1989; Ubrich et al., 1999), polymerization (Kadam et al., 2007; François et al., 2004; Zafiriou and Zhu, 1990; Clarke-Pringle and Mac Gregor, 1998), distillation (Welz et al., 2008), crystallization (Fiordalis and Georgakis, 2013), and bio-processes (Visser et al., 2000; Bodizs et al., 2007). A review of RTO for batch processes is given by Bonvin et al. (2002).

The problem of finding optimal operating conditions for a transient process can be expressed mathematically as follows (Srinivasan et al., 2003b):

$$\begin{aligned} \mathbf{w}_\mathrm{p}^*(\cdot) := \arg\min_{\mathbf{w}(\cdot)} \quad & J_\mathrm{p}\big(\mathbf{w}(\cdot)\big) \\ \text{subject to} \quad & \mathbf{S}_\mathrm{p}(t, \mathbf{w}(\cdot)) \leq \mathbf{0} \qquad \forall\, t \in [0, t_f], \\ & \mathbf{T}_\mathrm{p}(\mathbf{w}(\cdot)) \leq \mathbf{0}, \end{aligned} \tag{2.11}$$

where J_p is the terminal cost, $\mathbf{w}(t)$ 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 from 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 necessary KKT conditions, called Pontryagin's Maximum Principle, exists. However, these days, complex dynamic optimization problems are generally discretized and approximated by a static optimization problems, because static optimization problems are typically 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:

$$t_0 < t_1 < t_2 < \cdots < t_{n_s} = t_f \tag{2.12}$$

and then using low-order polynomials on each interval

$$\mathbf{w}(t) = \boldsymbol{\mathcal{P}}(t, \hat{\mathbf{u}}^j), \qquad t_{j-1} \leq t < t_j, \tag{2.13}$$

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

$$\mathbf{u} = \begin{bmatrix} \hat{\mathbf{u}}^1 \\ \hat{\mathbf{u}}^2 \\ \vdots \\ \hat{\mathbf{u}}^{n_s} \end{bmatrix} \quad \in \quad \mathbb{R}^{n_u}, \tag{2.14}$$

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 $\boldsymbol{\mathcal{W}}$ of the form:

$$\mathbf{w}(t) = \boldsymbol{\mathcal{W}}(t, \mathbf{u}) := \left\{ \boldsymbol{\mathcal{P}}(t, \hat{\mathbf{u}}^j) \quad \text{with } j \in [1, \ldots, n_s] \;\; s.t. \; t_{j-1} \leq t < t_j \right\}. \tag{2.15}$$

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:

$$\hat{\mathbf{g}}^i(\mathbf{u}) = \mathbf{S}\left(t_i, \boldsymbol{\mathcal{W}}(\cdot, \mathbf{u})\right), \quad i = 1, 2, \ldots, n_c. \tag{2.16}$$

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

$$\phi_\mathrm{p}(\mathbf{u}) = J_\mathrm{p}\left(\boldsymbol{\mathcal{W}}(\cdot, \mathbf{u})\right), \tag{2.17}$$

$$\mathbf{g}_\mathrm{p}(\mathbf{u}) = \begin{bmatrix} \hat{\mathbf{g}}^1(\mathbf{u}) \\ \hat{\mathbf{g}}^2(\mathbf{u}) \\ \vdots \\ \hat{\mathbf{g}}^{n_c}(\mathbf{u}) \\ \mathbf{T}_\mathrm{p}\left(\boldsymbol{\mathcal{W}}(\cdot, \mathbf{u})\right) \end{bmatrix}. \tag{2.18}$$

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:

$$J_\mathrm{p}\left(\boldsymbol{\mathcal{W}}(\cdot, \mathbf{u}_\mathrm{p}^*)\right) \simeq J_\mathrm{p}\left(\mathbf{w}_\mathrm{p}^*(\cdot)\right). \tag{2.19}$$

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

$$\mathbf{S}_{\mathrm{p}}\left(t, \boldsymbol{\mathcal{W}}(\cdot, \mathbf{u}_{\mathrm{p}}^{*})\right) \lessapprox \mathbf{0} \quad \forall\, t \in [0, t_f]. \tag{2.20}$$

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, n_u tends to be at least 20 for a typical dynamic optimization problem.

As will be seen in Section 2.3, the large dimension of the input vector $\mathbf{u}$ makes gradient estimation very difficult, if not intractable. An easy solution is simply to not use gradient correction terms, thus working only with the zeroth-order modifiers for the constraints (Marchetti et al., 2007). While this may work very well for processes, for which the optimal solution is mostly determined by active constraints, it may also perform poorly for others. The approach proposed by Chachuat et al. (2009) is to combine MA with the 'parsimonious' parametrization that is used in NCO tracking (Srinivasan and Bonvin, 2007). This parsimonious parametrization, with a relatively small number of decision variables, exploits the fact that the solutions to dynamic optimization problems have a particular structure. Process intuition, or robustness analysis involving optimization of the nominal model, can often confirm that this structure is unlikely to change for any expected disturbance or plant-model mismatch scenario. While attractive, this is a 'tailor-made' solution for each process, which requires a high level of process insight. The technique presented in the following section allows MA to be applied to transient processes without making any assumptions regarding the structure of the optimal solution.

2.3 Gradient estimation and dual MA

As we have already seen, gradient estimates are necessary for the implementation of RTO via MA. 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., 2013a). Here, we limit the discussion 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[1] in the i^{th} direction of the input space, i.e. the i^{th} element of $\nabla\phi_{\mathrm{E},k}$, is estimated as:

$$\left(\frac{\partial \phi}{\partial u_i}\right)_{\mathrm{E},k} = \frac{\tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k + \delta\mathbf{u}_i) - \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k)}{\|\delta\mathbf{u}_i\|}, \tag{2.21}$$

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 and constraints around each RTO point. Depending on the values of n_u and the plant settling time, the experimental cost may be unacceptable.

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

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

$$\nabla\phi_{\mathrm{E},k} = \begin{bmatrix} \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k) - \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_{k-1}) \\ \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k) - \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_{k-2}) \\ \vdots \\ \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k) - \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_{k-n_u}) \end{bmatrix}^T \left[\mathbf{u}_k - \mathbf{u}_{k-1}, \cdots, \mathbf{u}_k - \mathbf{u}_{k-n_u}\right]^{-1}. \quad (2.22)$$

The matrix inverse in the above equation will become badly conditioned if the past points do not extend evenly in all directions in the input space. This ill-conditioning can lead to very erroneous gradient estimates. Another technique, which does not suffer from ill-conditioning 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:

$$\nabla\phi_{\mathrm{E},k} = \nabla\phi_{\mathrm{E},k-1} + \frac{\tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k) - \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_{k-1}) - \nabla\phi_{\mathrm{E},k-1}(\mathbf{u}_k - \mathbf{u}_{k-1})}{\|\mathbf{u}_k - \mathbf{u}_{k-1}\|^2}(\mathbf{u}_k - \mathbf{u}_{k-1})^T. \quad (2.23)$$

While it might appear that, by using previously visited RTO points, the gradient 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 a good gradient estimate. 'Dual MA' algorithms have been proposed to deal with this problem by including the quality of the gradient estimates in the cost to be minimized at each step (Marchetti et al., 2010; Rodger and Chachuat, 2011; Marchetti, 2013) [2]. However, this conflict between the gradient estimation objective on the one hand, and the optimization objective on the other, negatively impacts convergence toward the plant optimum. Both of the above gradient estimate equations are based on the assumption that the plant cost function is locally linear. This assumption only holds if the past n_u RTO points are sufficiently close to each other, implying that the RTO input must evolve very gradually in order to ensure an acceptably accurate gradient estimate at each iteration. In general, the more decision variables in the optimization problem, the slower the plant optimum is reached.

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 the least-squares fit, 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 a significant amount of noise.

[2]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 and less excitation for better control.

3 Directional Modifier Adaptation

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 processes, and for 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.

3.1 Basic idea of directional MA

Definition 3.1 (Directional Derivative)**.** *The $(n_f \times n_r)$-dimensional directional derivative of a n_f-dimensional vector function* $\mathbf{f}$ *is:*

$$\nabla_{\mathbf{U}_\mathrm{r}} \mathbf{f}(\mathbf{u}) := \left. \frac{\partial \mathbf{f}(\mathbf{u} + \mathbf{U}_\mathrm{r}\mathbf{r})}{\partial \mathbf{r}} \right|_{\mathbf{r}=\mathbf{0}}, \tag{3.1}$$

where $\mathbf{U}_r = [\delta\mathbf{u}_1 \ \cdots \ \delta\mathbf{u}_\mathrm{r}]$ *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*.*

Property 3.1. *Applying the chain rule to Equation (3.1) yields:*

$$\nabla_{\mathbf{U}_\mathrm{r}} \mathbf{f}(\mathbf{u}) = \left. \frac{\partial \mathbf{f}(\mathbf{u} + \mathbf{U}_\mathrm{r}\mathbf{r})}{\partial \mathbf{r}} \right|_{\mathbf{r}=\mathbf{0}} = \nabla \mathbf{f}(\mathbf{u} + \mathbf{U}_\mathrm{r}\mathbf{r}) \left. \frac{\partial (\mathbf{u} + \mathbf{U}_\mathrm{r}\mathbf{r})}{\partial \mathbf{r}} \right|_{\mathbf{r}=\mathbf{0}} = \nabla \mathbf{f}(\mathbf{u}) \mathbf{U}_\mathrm{r}. \tag{3.2}$$

Property 3.2.

$$\nabla_{\mathbf{U}_\mathrm{r}} \mathbf{f}(\mathbf{u}) \mathbf{U}_\mathrm{r}^{+} \mathbf{x} = \begin{cases} \nabla \mathbf{f}(\mathbf{u})\mathbf{x} & \mathbf{x} \in C(\mathbf{U}_\mathrm{r}) \\ \mathbf{0} & \mathbf{x} \notin C(\mathbf{U}_\mathrm{r}) \end{cases}, \tag{3.3}$$

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

Property 3.2 follows from Property 3.1 by noting that

$$\mathbf{U}_\mathrm{r} \mathbf{U}_\mathrm{r}^{+} \mathbf{x} = \begin{cases} \mathbf{x} & \mathbf{x} \in C(\mathbf{U}_\mathrm{r}) \\ \mathbf{0} & \mathbf{x} \notin C(\mathbf{U}_\mathrm{r}) \end{cases}. \tag{3.4}$$

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 from Section 2:

Initialize: In addition, choose a matrix of 'privileged' input directions $\mathbf{U}_\mathrm{r}$, in which to estimate plant derivatives. Section 3.2 explains how to choose $\mathbf{U}_\mathrm{r}$.

4. 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. Estimate the cost gradient as:

$$\nabla\phi_{\mathrm{E},k} = \nabla\phi(\mathbf{u}_k)(\mathbf{I}_{n_u} - \mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+) + \nabla_{\mathbf{U}_\mathrm{r}}\phi_{\mathrm{E},k}\mathbf{U}_\mathrm{r}^+, \tag{3.5}$$

   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 (3.2), Equation (3.5) implies that:

$$\nabla\phi_{\mathrm{E},k}\delta\mathbf{u} = \begin{cases}\nabla\phi_\mathrm{p}(\mathbf{u}_k)\delta\mathbf{u} & \delta\mathbf{u}\in C(\mathbf{U}_\mathrm{r})\\ \nabla\phi(\mathbf{u}_k)\delta\mathbf{u} & \delta\mathbf{u}\notin C(\mathbf{U}_\mathrm{r})\end{cases}. \tag{3.6}$$

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 Problem (2.1), if it converges, it will do so to a point where the cost function cannot be improved in any of the privileged directions. This is formalized in the following theorem:

Theorem 3.1 (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:*

$$\begin{aligned}\min_{\mathbf{r}} \quad & \phi_\mathrm{p}\left(\mathbf{u}_\infty + \mathbf{U}_\mathrm{r}\mathbf{r}\right)\\ \text{s.t.} \quad & \mathbf{g}_\mathrm{p}\left(\mathbf{u}_\infty + \mathbf{U}_\mathrm{r}\mathbf{r}\right) \leq \mathbf{0}\,.\end{aligned} \tag{3.7}$$

Proof. Upon convergence of the D-MA algorithm

$$\boldsymbol{\epsilon}_\infty = \mathbf{g}_\mathrm{p}(\mathbf{u}_\infty^*) - \mathbf{g}(\mathbf{u}_\infty^*), \tag{3.8}$$

assuming noise-free constraint measurements. Also, according to Equations (2.9) and (2.10):

$$\left(\boldsymbol{\lambda}_\infty^\phi\right)^T = \nabla\phi_{\mathrm{E},\infty} - \nabla\phi(\mathbf{u}_\infty,\boldsymbol{\theta}_0), \tag{3.9}$$

$$\left(\boldsymbol{\lambda}_\infty^\mathrm{g}\right)^T = \nabla\mathbf{g}_{\mathrm{E},\infty} - \nabla\mathbf{g}(\mathbf{u}_\infty,\boldsymbol{\theta}_0). \tag{3.10}$$

Equation (3.8) implies that the constraint for the modified model-based optimization problem (2.5) matches that of the plant at $\mathbf{u}_\infty$:

$$\mathbf{g}_{\mathrm{m},\infty}(\mathbf{u}_\infty) = \mathbf{g}_{\mathrm{p}}(\mathbf{u}_\infty). \tag{3.11}$$

Also, the KKT conditions for the modified optimization problem (2.5) must be satisfied upon convergence. Hence,

$$\mathbf{g}_{\mathrm{m},\infty}(\mathbf{u}_\infty) = \mathbf{g}_{\mathrm{p}}(\mathbf{u}_\infty) \leq \mathbf{0}, \tag{3.12}$$

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

$$\boldsymbol{\nu}_i \mathbf{g}_{\mathrm{m},\infty,i}(\mathbf{u}_\infty) = \boldsymbol{\nu}_i \mathbf{g}_{\mathrm{p},i}(\mathbf{u}_\infty) = 0, \quad \forall i = 1, \ldots, n_g, \tag{3.13}$$

and

$$\nabla \phi_{\mathrm{m},\infty}(\mathbf{u}_\infty) + \boldsymbol{\nu}^T \nabla \mathbf{g}_{\mathrm{m},\infty}(\mathbf{u}_\infty) = \mathbf{0} \tag{3.14}$$

$$\implies \nabla \phi(\mathbf{u}_\infty, \boldsymbol{\theta}_0) + \left(\boldsymbol{\lambda}^{\phi}_{\infty}\right)^T + \boldsymbol{\nu}^T \left(\nabla \mathbf{g}(\mathbf{u}_\infty, \boldsymbol{\theta}_0) + (\boldsymbol{\lambda}^{\mathrm{g}}_{\infty})^T\right) = \mathbf{0} \tag{3.15}$$

$$\implies \nabla \phi_{\mathrm{E},k} + \boldsymbol{\nu}^T \nabla \mathbf{g}_{\mathrm{E},k} = 0. \tag{3.16}$$

Post-multiplying Equation (3.16) by $\mathbf{U}_\mathrm{r}$ and using Equation (3.5) yields:

$$\nabla_{\mathbf{U}_\mathrm{r}} \phi_{\mathrm{E},\infty} + \boldsymbol{\nu}^T \nabla_{\mathbf{U}_\mathrm{r}} \mathbf{g}_{\mathrm{E},\infty} = \mathbf{0}. \tag{3.17}$$

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 3.1, yields:

$$\left.\frac{\partial \phi_{\mathrm{p}}(\mathbf{u}_\infty + \mathbf{U}_\mathrm{r}\mathbf{r})}{\partial \mathbf{r}} + \boldsymbol{\nu}^T \frac{\partial \mathbf{g}_{\mathrm{p}}(\mathbf{u}_\infty + \mathbf{U}_\mathrm{r}\mathbf{r})}{\partial \mathbf{r}}\right|_{\mathbf{r}=\mathbf{0}} = 0. \tag{3.18}$$

Since $\mathbf{u}_\infty + \mathbf{U}_\mathrm{r}\mathbf{r}$ obviously equals $\mathbf{u}_\infty$ when $\mathbf{r} = \mathbf{0}$, Equations (3.12)and (3.13) mean that the primal feasibility and dual feasibility conditions for Problem (3.7) are satisfied at $\mathbf{r} = \mathbf{0}$. Together with the fact that Equation(3.18) shows the satisfaction of the stationarity KKT condition for Problem (3.7) at $\mathbf{r} = \mathbf{0}$, this proves that $\mathbf{r} = \mathbf{0}$ is a KKT point for Problem (3.7). □

3.2 Choosing the gradient direction matrix

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 $\mathbf{u}$), we assume 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 3.2 (Optimal Gradient Directions for Small Parametric Uncertainty)**.** *In the event of small parametric plant-model mismatch, $\phi_{\mathrm{p}}(\mathbf{u}) = \phi(\mathbf{u}, \boldsymbol{\theta}_{\mathrm{p}})$ and $\mathbf{g}_{\mathrm{p}}(\mathbf{u}) = \mathbf{g}(\mathbf{u}, \boldsymbol{\theta}_{\mathrm{p}})$ with $\boldsymbol{\theta}_{\mathrm{p}} = \boldsymbol{\theta}_0 + \Delta\boldsymbol{\theta}$, 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:*

$$\mathbf{U}_{\mathrm{r}} = \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u} \partial \boldsymbol{\theta}}(\mathbf{u}^*(\boldsymbol{\theta}_0), \boldsymbol{\nu}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta}_0) \quad \in \mathbb{R}^{n_u \times n_\theta}, \tag{3.19}$$

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 (2.5), assuming it is not a non-minimum stationary point for this problem. The stationary KKT conditions mean $\exists\ \boldsymbol{\nu}$ such that:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\infty, \boldsymbol{\nu}, \boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_\infty^\phi)^T + \boldsymbol{\nu}^T (\boldsymbol{\lambda}_\infty^g)^T = \mathbf{0}, \tag{3.20}$$

with

$$\begin{aligned} (\boldsymbol{\lambda}_\infty^\phi)^T &= \nabla \phi_{\mathrm{E},\infty} - \nabla \phi(\mathbf{u}_\infty, \boldsymbol{\theta}_0) \\ &= (\nabla_{\mathbf{U}_{\mathrm{r}}} \phi_{\mathrm{E},\infty} - \nabla_{\mathbf{U}_{\mathrm{r}}} \phi(\mathbf{u}_\infty, \boldsymbol{\theta}_0)) \mathbf{U}_{\mathrm{r}}^+, \end{aligned} \tag{3.21}$$

where Equation (3.21) is obtained by combining the definition of the gradient estimate (3.5) with the definition of the gradient modifiers (2.10) upon convergence. In the same manner, the constraint gradient modifiers upon convergence are:

$$(\boldsymbol{\lambda}_\infty^g)^T = (\nabla_{\mathbf{U}_{\mathrm{r}}} \mathbf{g}_{\mathrm{E},\infty} - \nabla_{\mathbf{U}_{\mathrm{r}}} \mathbf{g}(\mathbf{u}_\infty, \boldsymbol{\theta}_0)) \mathbf{U}_{\mathrm{r}}^+. \tag{3.22}$$

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:

$$\mathbf{g}_{\mathrm{p}}(\mathbf{u}_\infty) \leq \mathbf{0}, \qquad \boldsymbol{\nu}^T \mathbf{g}_{\mathrm{p}}(\mathbf{u}_\infty) = \mathbf{0}. \tag{3.23}$$

We now show that $\mathbf{u}_\infty = \mathbf{u}_\mathrm{p}^*$ satisfies Conditions (3.20) and (3.23), with $\boldsymbol{\nu} = \boldsymbol{\nu}_\mathrm{p}^*$. Firstly, as $\mathbf{u}_\mathrm{p}^*$ is a KKT point for the plant, Conditions (3.23) are satisfied. Also, $\mathbf{u}_\mathrm{p}^*$ satisfies the stationary KKT condition for the plant optimization problem, which reads:

$$\nabla\phi_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) + (\boldsymbol{\nu}_\mathrm{p}^*)^T \nabla\mathbf{g}_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) = \mathbf{0}. \tag{3.24}$$

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:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p}) = \nabla\phi_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) + (\boldsymbol{\nu}_\mathrm{p}^*)^T \nabla\mathbf{g}_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) = \mathbf{0}. \tag{3.25}$$

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

$$\frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + \Delta\boldsymbol{\theta}^T \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}^T(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + \mathcal{O}(\Delta\boldsymbol{\theta}^2) = \mathbf{0}. \tag{3.26}$$

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):

$$\begin{aligned}
\frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) = & \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}} + \frac{\partial}{\partial \mathbf{u}}\left(\frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}\right)\frac{\partial \mathbf{u}^*}{\partial\boldsymbol{\theta}}\boldsymbol{\Delta\theta} + \\
& \left.\frac{\partial}{\partial \boldsymbol{\nu}}\left(\frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}\right)\frac{\partial \boldsymbol{\nu}^*}{\partial\boldsymbol{\theta}}\boldsymbol{\Delta\theta}\right|_{(\mathbf{u}^*(\boldsymbol{\theta}_0), \boldsymbol{\nu}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta}_0)} + \mathcal{O}(\Delta\boldsymbol{\theta}^2) && (3.27) \\
= & \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}(\mathbf{u}^*(\boldsymbol{\theta}_0), \boldsymbol{\nu}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta}_0) + \mathcal{O}(\Delta\boldsymbol{\theta}) && (3.28) \\
= & \mathbf{U}_\mathrm{r} + \mathcal{O}(\Delta\boldsymbol{\theta}), && (3.29)
\end{aligned}$$

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

$$\Delta\boldsymbol{\theta}^T \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}^T(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) = \Delta\boldsymbol{\theta}^T \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}^T(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0)\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+ + O(\Delta\boldsymbol{\theta}^2). \tag{3.30}$$

Equation (3.26) can now be written as:

$$\begin{aligned}
& \frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + \Delta\boldsymbol{\theta}^T \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u}\partial\boldsymbol{\theta}}^T(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0)\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+ + \mathcal{O}(\Delta\boldsymbol{\theta}^2) = \mathbf{0} && (3.31) \\
\Longrightarrow\ & \frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + \nabla\left(\phi(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p}) - \phi(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_0)\right)\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+ \\
& +(\boldsymbol{\nu}_\mathrm{p}^*)^T\nabla\left(\mathbf{g}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p}) - \mathbf{g}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_0)\right)\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^+ + \mathcal{O}(\Delta\boldsymbol{\theta}^2) = \mathbf{0} && (3.32) \\
\Longrightarrow\ & \frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + \nabla_{\mathbf{U}_\mathrm{r}}\left(\phi(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p}) - \phi(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_0)\right)\mathbf{U}_\mathrm{r}^+ \\
& +(\boldsymbol{\nu}_\mathrm{p}^*)^T\nabla_{\mathbf{U}_\mathrm{r}}\left(\mathbf{g}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_\mathrm{p}) - \mathbf{g}(\mathbf{u}_\mathrm{p}^*, \boldsymbol{\theta}_0)\right)\mathbf{U}_\mathrm{r}^+ + \mathcal{O}(\Delta\boldsymbol{\theta}^2) = \mathbf{0}. && (3.33)
\end{aligned}$$

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 $\mathcal{O}(\Delta\boldsymbol{\theta}^2) \approx 0$, this becomes:

$$\frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}_\infty, \boldsymbol{\nu}_\mathrm{p}^*, \boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_\infty^\phi)^T + (\boldsymbol{\nu}_\mathrm{p}^*)^T(\boldsymbol{\lambda}_\infty^g)^T = \mathbf{0}, \tag{3.34}$$

with the modifier terms defined as in Equations (3.21) and (3.22) (recalling that $\mathbf{u}_\infty = \mathbf{u}_\mathrm{p}^*$). Hence, Condition (3.20) 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 5, it can even yield nearly 'optimal' adaptation directions when there is significant 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 3.3 (Optimality Loss due to Lagrangian Gradient Error). *The optimality loss due to a small Lagrangian gradient error is:*

$$\phi_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) - \phi_\mathrm{p}(\mathbf{u}^*(\boldsymbol{\theta}_0)) = -\boldsymbol{\epsilon}^T \mathbf{A} \boldsymbol{\epsilon} + \mathcal{O}(\boldsymbol{\epsilon}^3) \tag{3.35}$$

$$\boldsymbol{\epsilon} = \frac{\partial \mathbf{L}_\mathrm{p}}{\partial \mathbf{u}}(\mathbf{u}, \boldsymbol{\nu}) - \frac{\partial \mathbf{L}}{\partial \mathbf{u}}(\mathbf{u}, \boldsymbol{\nu}, \boldsymbol{\theta}_0) \tag{3.36}$$

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:

$$\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^T = \frac{\partial^2 \mathbf{L}}{\partial \mathbf{u} \partial \boldsymbol{\theta}}(\mathbf{u}^*(\boldsymbol{\theta}_0), \boldsymbol{\nu}^*(\boldsymbol{\theta}_0), \boldsymbol{\theta}_0) \mathrm{diag}(\theta_1^{\max} - \theta_1^{\min}, \ldots, \theta_{n_\theta}^{\max} - \theta_{n_\theta}^{\min}), \tag{3.37}$$

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 $\sigma_{n_r+1} << \sigma_1$. 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 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 uses previously visited RTO 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.*

4.1 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

$$\nabla_{\delta\mathbf{u}}\phi_{\mathrm{E}} = \frac{\tilde{\phi}_{\mathrm{p}}(\mathbf{u}_j) - \tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k)}{\|\mathbf{u}_j - \mathbf{u}_k\|} \tag{4.1}$$

$$= \nabla_{\delta\mathbf{u}}\phi_{\mathrm{p}}(\mathbf{u}_k) + \frac{d_j^{\phi} - d_k^{\phi}}{\|\mathbf{u}_j - \mathbf{u}_k\|} + \mathcal{O}\left(\|\mathbf{u}_j - \mathbf{u}_k\|\right). \tag{4.2}$$

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

$$\sigma_{\mathrm{E}}^2 = var\{\nabla_{\delta\mathbf{u}}\phi_{\mathrm{E}}\} = \frac{2\sigma_{\phi}^2}{\|\mathbf{u}_j - \mathbf{u}_k\|^2}. \tag{4.3}$$

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:

$$\nabla\phi_{\mathrm{new}} = \nabla\phi_{\mathrm{old}} + \kappa(\nabla_{\delta\mathbf{u}}\phi_{\mathrm{E}} - \nabla\phi_{\mathrm{old}}\delta\mathbf{u})\delta\mathbf{u}^T, \tag{4.4}$$

with the variance matrix

$$\mathbf{\Sigma}_{\mathrm{new}} = (\mathbf{I}_{n_u} - \kappa\delta\mathbf{u}\delta\mathbf{u}^T)\mathbf{\Sigma}_{\mathrm{old}}(\mathbf{I}_{n_u} - \kappa\delta\mathbf{u}\delta\mathbf{u}^T) + \kappa^2\sigma_{\mathrm{E}}^2\delta\mathbf{u}\delta\mathbf{u}^T. \tag{4.5}$$

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

Proposition 4.1 (Optimal Weighted Broyden Update)**.** *The value of κ that minimizes the variance of the gradient estimate in the $\delta\mathbf{u}$ direction is:*

$$\kappa = \frac{\delta\mathbf{u}^T\mathbf{\Sigma}_{\text{old}}\delta\mathbf{u}}{\delta\mathbf{u}^T\mathbf{\Sigma}_{\text{old}}\delta\mathbf{u} + \sigma_E^2} \tag{4.6}$$

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

$$\delta\mathbf{u}^T\mathbf{\Sigma}_{\text{new}}\delta\mathbf{u} = (1-\kappa)^2\delta\mathbf{u}^T\mathbf{\Sigma}_{\text{old}}\delta\mathbf{u} + \kappa^2\sigma_{\text{E}}^2. \tag{4.7}$$

By differentiating the expression with respect to κ, it follows that the value of k given in Equation (4.6) 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: Iterative weighted Broyden-update gradient estimator

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

```
for ∀ $j$ such that $\|\mathbf{u}_j - \mathbf{u}_k\| < \Delta^{\text{r}}_{\text{max}}$
    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 $\sigma_{\text{E}}^2$ using Equations (4.1) and (4.3).
    3. Compute $\kappa$ according to Equation (4.6).
    4. Compute $\nabla\phi_{\text{new}}$ and $\mathbf{\Sigma}_{\text{new}}$ using Equations (4.4) and (4.5).
    5. $\nabla\phi_{\text{old}} = \nabla\phi_{\text{new}}$ and $\mathbf{\Sigma}_{\text{old}} = \mathbf{\Sigma}_{\text{new}}$.
end
```

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

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

4.2 Dual directional MA Algorithm

The following is the practically applicable RTO algorithm advocated in this paper. 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: Dual Directional Modifier Adaptation (Dual D-MA)

Initialize: Choose $\mathbf{U}_\mathrm{r}$ using the method in Section 3.2. 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 $\Delta^\mathrm{r}_\mathrm{max}$. Choose the desired gradient estimate variance in the privileged directions, σ^2_{TOL} and set $\bar{\delta\mathbf{u}} = \mathbf{0}$.

for $k = 1 \to \infty$

1. Solve the modified model-based optimization problem

$$\mathbf{u}_k := \underset{\mathbf{u}}{\operatorname{argmin}} \quad \phi_{\mathrm{m},k-1}(\mathbf{u})$$
$$\text{s.t.} \quad \mathbf{g}_{\mathrm{m},k-1}(\mathbf{u}) \leq \mathbf{0}, \tag{4.8}$$
$$\|\mathbf{u} - \mathbf{u}_{k-1}\| \leq \Delta_{max}. \tag{4.9}$$

   where the modified cost and constraints are given by

$$\phi_{\mathrm{m},k}(\mathbf{u}) := \phi(\mathbf{u}, \boldsymbol{\theta}_0) + (\boldsymbol{\lambda}_k^\phi)^T(\mathbf{u} - \mathbf{u}_k) - c|\bar{\delta\mathbf{u}}^T(\mathbf{u} - \mathbf{u}_k)|^2, \tag{4.10}$$
$$\mathbf{g}_{\mathrm{m},k}(\mathbf{u}) := \mathbf{g}(\mathbf{u}, \boldsymbol{\theta}_0) + \boldsymbol{\epsilon}_k + (\boldsymbol{\lambda}_k^\mathrm{g})^T(\mathbf{u} - \mathbf{u}_k). \tag{4.11}$$

   The last term in the modified cost function is the aforementioned reward term. It rewards steps in the direction $\bar{\delta\mathbf{u}}$, 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 4.1 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[3]:

$$\bar{\delta\mathbf{u}} \in \arg\max_{\delta\mathbf{u}} \ \delta\mathbf{u}^T\boldsymbol{\Sigma}^L_{\mathrm{E},k}\delta\mathbf{u}$$
$$s.t. \quad \|\delta\mathbf{u}\| = 1,$$
$$\delta\mathbf{u} \in C(\mathbf{U}_\mathrm{r}), \tag{4.12}$$

[3]Note that the solution to problem (4.12) is the (normalized) dominant eigenvector of $\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^T\boldsymbol{\Sigma}^L_{\mathrm{E},k-1}\mathbf{U}_\mathrm{r}\mathbf{U}_\mathrm{r}^T$.

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

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

 6. Calculate the modifier terms for the next iteration according to Equations (2.8), (2.9) and (2.10).

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 5.3 illustrates how the algorithm parameters can be chosen in a methodological fashion.

5 Simulated Case Study

Airborne Wind Energy (otherwise known as kite power) is a promising emerging wind-power technology. It exploits the aerodynamic force generated by a kite (which can be imagined as an airplane on a string) to generate power, either by driving a generator (Ruiterkamp and Sieberling, 2013), or by pulling a boat (Erhard and Strauch, 2013). The open problem of optimally controlling a power-producing kite during dynamic flight (which can be imagined as an airplane on a string) is currently of great technological relevance. The kite is free to fly almost any path, provided it does not crash. However, experimental studies (Zgraggen et al., 2013) have confirmed that the path taken by the kite significantly affects the power it can generate. 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 shows that Dual D-MA can efficiently address this problem.

The simulation example considered here is based on industrial data, experimental studies from the literature, and one of the authors' own practical experience in experimental kite control.

5.1 Plant description

The kite dynamic equations are taken from Erhard and Strauch (2013). These experimentally validated equations have been successfully used in an industrial setting to design control algorithms for very large kites. An embellishment proposed by Costello et al. (2013), which has also been experimentally validated, accounts for the reduction of line tension caused by steering deflections. The kite fixed, inertial, right-hand co-ordinate system is depicted in Figure 1. Since

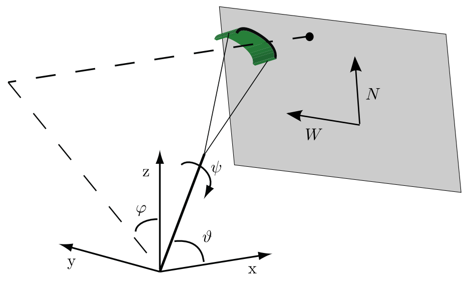

Figure 1: Spherical co-ordinate system for the kite position. The x and y axes are horizontal, while the z-axis points skywards. The kite is tethered to the origin.

the dynamic equations for the model are simple analytic expressions, they are merely stated here (the interested reader is invited to see Erhard and Strauch (2013) and Costello et al. (2013) for more details):

$$\dot{\vartheta} = \frac{w_{\mathrm{ap}}}{r}\left(\cos\psi - \frac{\tan\vartheta}{E}\right), \tag{5.1}$$

$$\dot{\varphi} = -\frac{w_{\mathrm{ap}}}{r\sin\vartheta}\sin\psi, \tag{5.2}$$

$$\dot{\psi} = w_{\mathrm{ap}} g_{\mathrm{s}} \delta + \dot{\varphi}\cos\vartheta, \tag{5.3}$$

where ϑ and φ are the kite spherical co-ordinates (see Figure 1), ψ is the kite orientation, r is the (constant) line length, g_{s} is the turning constant, and δ is the steering deflection. The lift/drag ratio, E, and the magnitude of the apparent wind projected onto the quarter sphere, w_{ap}, are given by

$$w_{\mathrm{ap}} = wE\cos\vartheta, \tag{5.4}$$

$$E = E_0 - c\delta^2, \tag{5.5}$$

where w is the wind speed at the kite current altitude, and c is the turning penalty factor. The wind speed is given by the classic power law (Archer, 2013):

$$w = w_{\mathrm{ref}}(z/z_{\mathrm{ref}})^a, \tag{5.6}$$

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. The line tension is given by

$$T = \left(\frac{1}{2}\rho A w_0^2\right) \cos^2 \vartheta (E+1)\sqrt{E^2+1}. \tag{5.7}$$

The plant parameters are given in Table 1. They were selected to correspond closely with the prototypes currently under development in this field (Ruiterkamp and Sieberling, 2013; Fritz, 2013; van der Vlugt et al., 2013). For plotting purposes in this paper, the kite position is projected onto the plane defined by the two orthogonal vectors $\hat{\mathbf{e}}_W = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T$ and $\hat{\mathbf{e}}_N = \begin{bmatrix} -\sin\bar{\vartheta} & 0 & \cos\bar{\vartheta} \end{bmatrix}^T$ (radians), which (as shown in Figure 1) are tangent to the sphere upon which the kite can move at the point $\{\bar{\vartheta}, \bar{\varphi}\} = \{0.3, 0\}$ rad.

Table 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	$\mathrm{kg \cdot m^{-3}}$
E_0	6	4.5	-
g_{s}	5×10^{-3}	7×10^{-3}	$\mathrm{rad \cdot m^{-2}}$
c	.06	.02	m^{-2}
z_{ref}	10	10	m
w_{ref}	8	8	$\mathrm{m \cdot s^{-1}}$
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 us to focus on performance optimization, 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:

$$\bar{T} := \frac{1}{t_{\mathrm{f}} - t_0} \int_{t_0}^{t_{\mathrm{f}}} T dt, \tag{5.8}$$

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 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:

$$\mathbf{u} = \left[\vartheta_{\mathrm{r}}(0) \quad \varphi_{\mathrm{r}}(0) \quad \vartheta_{\mathrm{r}}(\tfrac{1}{N}) \quad \varphi_{\mathrm{r}}(\tfrac{1}{N}) \quad \vartheta_{\mathrm{r}}(\tfrac{2}{N}) \quad \varphi_{\mathrm{r}}(\tfrac{2}{N}) \cdots \vartheta_{\mathrm{r}}(\tfrac{N-1}{N}) \quad \varphi_{\mathrm{r}}(\tfrac{N-1}{N})\right]^T, \tag{5.9}$$

where $N = n_u/2$ (for this simulation study $n_u = 40$ is used). The continuous reference path is obtained from $\mathbf{u}$ by fitting a spline to the points it contains. The spline is forced to be periodic (note that $\mathbf{u}$ does not specify the final point on the path), that is, both the values and the slope of the spline at the endpoint must match:

$$\vartheta_{\mathrm{r}}(0) = \vartheta_{\mathrm{r}}(1), \qquad \varphi_{\mathrm{r}}(0) = \varphi_{\mathrm{r}}(1) \tag{5.10}$$

$$\dot{\vartheta}_{\mathrm{r}}(0) = \dot{\vartheta}_{\mathrm{r}}(1), \qquad \dot{\varphi}_{\mathrm{r}}(0) = \dot{\varphi}_{\mathrm{r}}(1) \tag{5.11}$$

The kite must also respect a height constraint $z(l) := r \sin(\vartheta(l)) \cos(\phi(l)) \geq z_{\mathrm{min}}$ and a maximum steering-deflection constraint $|\delta(l)| \leq \delta_{\mathrm{max}}$, *at every point on the path.* These constraints are also discretized:

$$\mathbf{g}_z = \begin{bmatrix} 1 - z(0)/z_{\mathrm{min}} \\ 1 - z(\frac{1}{N})/z_{\mathrm{min}} \\ 1 - z(\frac{2}{N})/z_{\mathrm{min}} \\ \vdots \\ 1 - z\left(\frac{N-1}{N}\right)/z_{\mathrm{min}} \end{bmatrix}, \qquad \mathbf{g}_\delta = \begin{bmatrix} |\delta(0)|/\delta_{\mathrm{max}} - 1 \\ |\delta(\frac{1}{N})|/\delta_{\mathrm{max}} - 1 \\ |\delta(\frac{2}{N})|/\delta_{\mathrm{max}} - 1 \\ \vdots \\ |\delta\left(\frac{N-1}{N}\right)|/\delta_{\mathrm{max}} - 1 \end{bmatrix}. \tag{5.12}$$

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

$$\begin{aligned} \mathbf{u}_{\mathrm{p}}^* = & \operatorname*{argmin}_{\mathbf{u}} \quad \phi_{\mathrm{p}}(\mathbf{u}) := -\frac{\bar{T}}{c_T} \\ \text{s.t.} \quad & \mathbf{g}_{\mathrm{p}}(\mathbf{u}) := \begin{bmatrix} \mathbf{g}_z \\ \mathbf{g}_\delta \end{bmatrix} \leq \mathbf{0}, \end{aligned} \tag{5.13}$$

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 kite dynamic equations. The parameters of the optimization problem are given in Table 2. The cost and constraint measurements are corrupted with about 3 % zero-mean noise.

5.2 Available model

The available model of the closed-loop system is based on the same equations as the plant, with the exception of the wind law, which for the model is given by the simple linear law:

$$w = w_{\mathrm{ref}} + (z - z_{\mathrm{ref}})\Delta w, \tag{5.14}$$

Table 2: Optimization Parameters

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

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 1) are substantially different from the actual plant values (i.e. there is parametric plant-model mismatch).

5.3 RTO design procedure

The preferred directions $\mathbf{U}_{\text{r}}$ are chosen exactly as described in Section 3.2, with the parameter uncertainty intervals given in Table 3. The diagonal matrix of

Table 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}

singular values $\mathbf{\Sigma}$ in Equation (3.37) 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 3.2), $\mathbf{U}_{\text{r}}$ was duly chosen as the directions (the columns of $\mathbf{U}$ in Equation (3.37)) corresponding to the two dominant singular values. The path variations corresponding to the two chosen directions are shown in Figure 2. 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.

The remaining parameters for the dual D-MA algorithm (given in Table 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

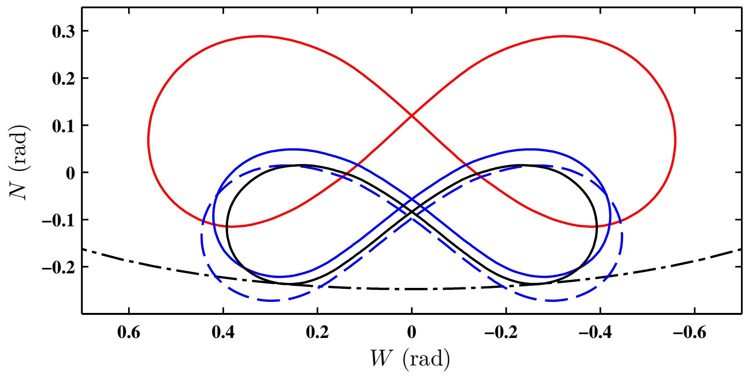

Figure 2: Kite optimal paths: path corresponding to $\mathbf{u}_\mathrm{p}^*$ (red); model optimal path corresponding to $\mathbf{u}^*(\boldsymbol{\theta_0})$ (black); path variations produced by steps in the privileged input directions, corresponding to $\mathbf{u}^*(\boldsymbol{\theta_0}) + \Delta_\mathrm{max}\mathbf{U}_{\mathrm{r},i}$,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

Table 4: Values of the design parameters for dual D-MA in the kite example.

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

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 Δ_max, consider the error when the cost directional derivative is estimated using Equation (4.1) and the two points $\mathbf{u}_k$ and $\mathbf{u}_j$. According to Equation (4.3) if $\Delta = \|\mathbf{u}_j - \mathbf{u}_k\|$, the standard deviation of the noise error is:

$$\zeta_d = \frac{\sqrt{2}\sigma_\phi}{\Delta}. \tag{5.15}$$

Also, the truncation error can be approximated as:

$$\zeta_T = \Delta \times H, \tag{5.16}$$

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 3 plots these two error terms as fucntions 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 $\Delta_\mathrm{max} = 0.03$, 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 (4.10) will encourage the algorithm to take as large a step as is allowed by Δ_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 $\Delta^r_\mathrm{max} = 2 \times \Delta_\mathrm{max}$. Again, this choice is a trade-off, a smaller value of Δ^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.

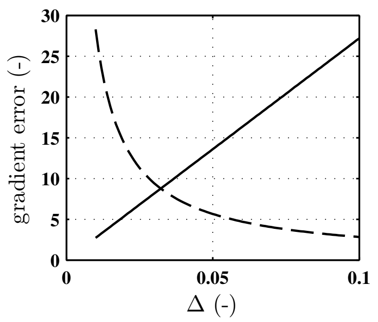

Figure 3: Noise error affecting the directional derivative estimate ζ_d (dashed), and truncation error ζ_T (solid) as a function of the distance between the points used to estimate Δ.

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 3). Thus, the dual D-MA algorithm will tend to 'trust' experimental information more than the model.

5.4 RTO results

Figure 2 shows that the model optimal solution (calculated with the nominal parameter values) is significantly different from the plant optimal solution, with the optimality loss

$$\frac{\phi_\mathrm{p}(\mathbf{u}_\mathrm{p}^*) - \phi_\mathrm{p}(\mathbf{u}^*(\boldsymbol{\theta}_\mathbf{0}))}{\phi_\mathrm{p}(\mathbf{u}_\mathrm{p}^*)} = 29\ \%. \tag{5.17}$$

After about 10 iterations, the dual D-MA algorithm has reduced this optimality loss to about 5 % (Figure 4), despite a significant amount of measurement noise. This is very fast, given that the kite takes roughly 15 s to complete one cycle of the path, with one RTO iteration per cycle.

As can be seen from Figures 5 and 6, since the desired gradient accuracy $\sigma_{\rm TOL}$ 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.

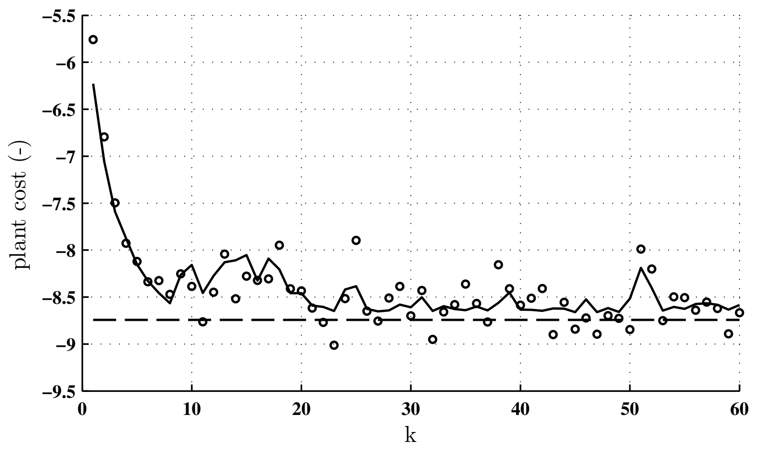

Figure 4: True (noise-free) and and measured (noisy) plant costs $\phi_{\rm p}(\mathbf{u}_k)$ (solid) and $\tilde{\phi}_{\rm p}(\mathbf{u}_k)$ (dots) as functions of the RTO iteration number k for $n_r = 2$. The optimal plant cost $\phi_{\rm p}(\mathbf{u}_{\rm p}^*)$ is also shown (dashed).

Figure 7 shows that the plant directional derivatives in the privileged directions are driven close to 0. This is particularly true for the $\mathbf{U}_{\rm r,2}$ direction (see Figure 2), which is the main direction the algorithm needs to adapt in to reach the plant optimal solution. Hence,s dual D-MA converges to the vicinity of a *directionally* optimal point for the plant, as predicted by Theorem 3.1. What is more, as can be seen from Figure 8, 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 9. 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 (3.37) to select the privileged directions.

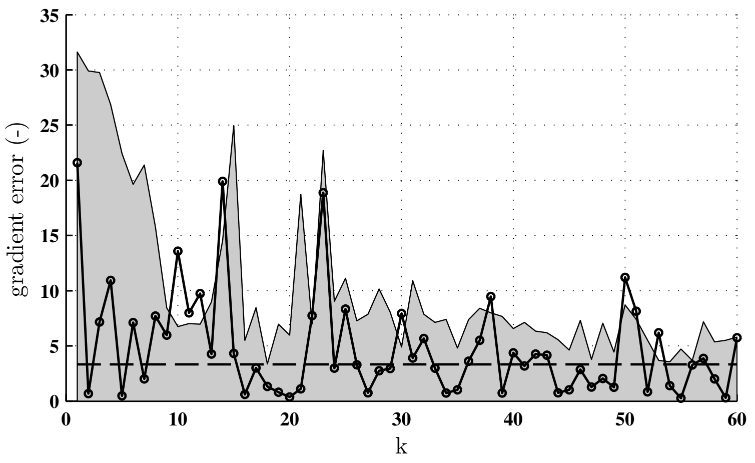

Figure 5: Gradient estimation error in the first privileged direction $|\nabla_{\mathbf{U}_{\mathrm{r},1}}\phi_{\mathrm{E},k} - \nabla_{\mathbf{U}_{\mathrm{r},1}}\phi_{\mathrm{p}}(\mathbf{u}_k)|$ (solid), with its standard deviation $\sqrt{\mathbf{U}_{\mathrm{r},1}^T \boldsymbol{\Sigma}_{\mathrm{E},k}^{\phi} \mathbf{U}_{\mathrm{r},1}}$ calculated on-line (shaded), along with the desired threshold value σ_{TOL} (dashed).

6 Conclusions

RTO using process models is regularly implemented in industry, with significant performance improvement (Darby et al., 2011). Ideally, RTO algorithms provide constraint satisfaction, on-line diagnostics (including optimality guarantees for the plant), and rapid convergence. However, the current industry standard, the two-step approach, cannot detect whether the plant optimum has been reached. In addition, the two-step approach requires a parameter-estimation problem to be solved on-line, which may become intractable if there are many uncertain model-parameters. In contrast, the MA family of techniques uses measurements to estimate the plant gradients rather than to estimate model parameters. Gradient estimates 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. However, for a process with many inputs, standard MA is crippled by the experimental cost of gradient estimation.

The solution put forward in this paper 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

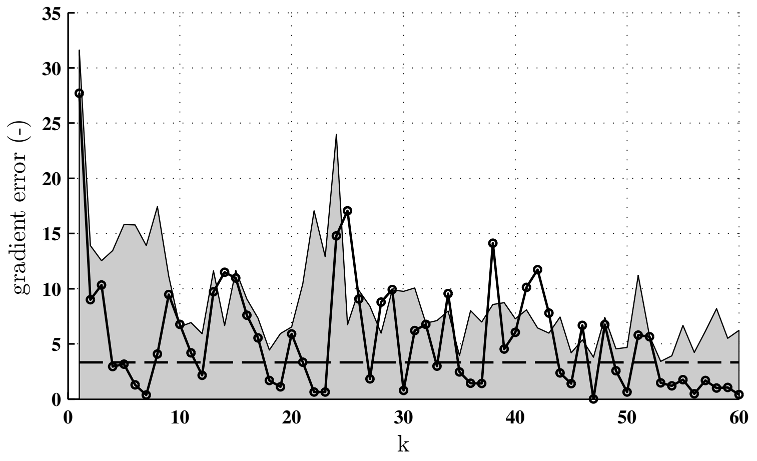

Figure 6: Gradient estimation error in the first privileged direction $|\nabla_{\mathbf{U}_{\mathrm{r},2}}\phi_{\mathrm{E},k} - \nabla_{\mathbf{U}_{\mathrm{r},2}}\phi_{\mathrm{p}}(\mathbf{u}_k)|$ (solid), with its standard deviation $\sqrt{\mathbf{U}_{\mathrm{r},2}^T \mathbf{\Sigma}_{\mathrm{E},k}^{\phi} \mathbf{U}_{\mathrm{r},2}}$ calculated on-line (shaded), along with the desired threshold value σ_{TOL} (dashed).

of the plant optimal solution, despite significant measurement noise and both structural and parametric plant-model mismatch. In summary, D-MA is specifically tailored to complex processes with many degrees-of-freedom, for which an approximate model containing a number of uncertain parameters is available.

A number of interesting research directions could improve this work. Firstly, as is also the case for MA and the two-step approach, there is no rigorous theoretical guarantee that D-MA will converge. Neither can it be proven that constraints will not be violated *prior* to convergence, although by using constraint back-offs and by tuning conservatively the filters for the zeroth-order constraint modifiers, this can generally be achieved in practice. It is likely that such theoretical guarantees could be made for MA schemes in general, if the ideas in Bunin (2014) can be extended to constrained problems, or alternatively, if an intelligent algorithm such as that suggested by Bunin et al. (2013b) were used to filter the steps taken by the MA algorithm. Further work will also investigate the theoretical properties of the Iterative Weighted Broyden-update gradient estimation algorithm proposed in this paper.

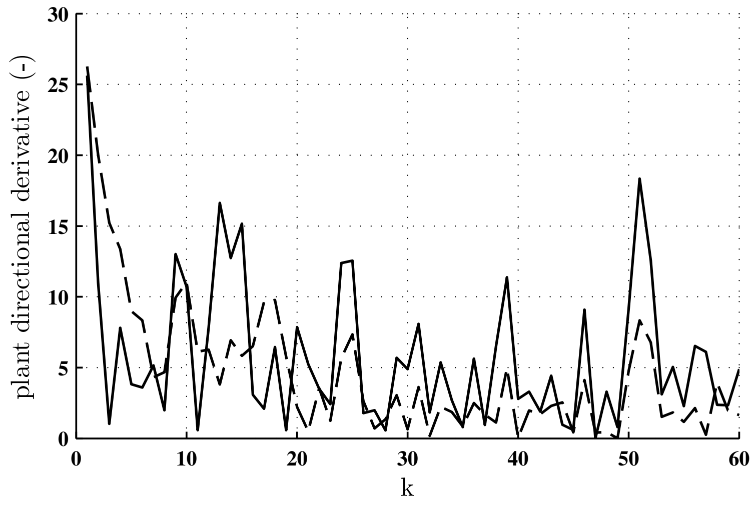

Figure 7: Directional derivatives for the plant cost $\nabla_{\mathbf{U}_{r,i}}\phi_{\mathrm{p}}(\mathbf{u}_k)$ for $i = 1$ (solid) and $i = 2$ (dashed) as functions of the RTO iteration number.

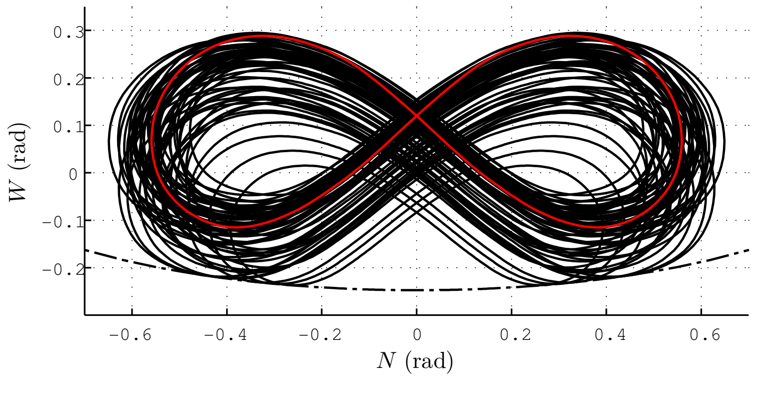

Figure 8: All the paths corresponding to $\mathbf{u}_k$, $k = 1, \ldots 60$, (black) for $r = 2$, as well as the plant optimal path $\mathbf{u}_k^*$ (red) and the height constraint (dot-dashed).

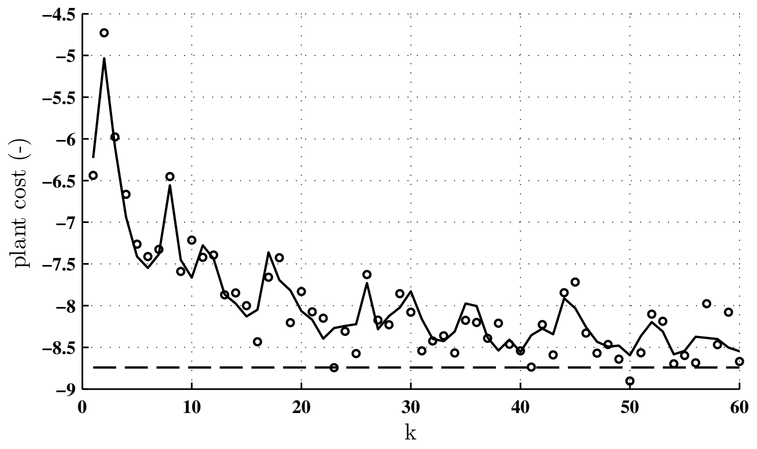

Figure 9: True (noise-free) and and measured (noisy) plant costs $\phi_{\mathrm{p}}(\mathbf{u}_k)$ (solid) and $\tilde{\phi}_{\mathrm{p}}(\mathbf{u}_k)$ (dots) as functions of the RTO iteration number k for $n_r = 4$. The optimal plant cost $\phi_{\mathrm{p}}(\mathbf{u}_{\mathrm{p}}^*)$ is also shown (dashed).

References

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.

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.

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

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, 2013a.

G. A. Bunin, G. François, and D. Bonvin. Sufficient conditions for feasibility and optimality of real-time optimization schemes - I. theoretical foundations. 2013b. ArXiv:1308.2620.

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 *Proceedings 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.

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.

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

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.

A. Fiordalis and C. Georgakis. Data-driven, using design of dynamic experiments, versus model-driven optimization of batch crystallization processes. *J. Process Control*, 23(2):179–188, 2013.

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, 2013.

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. Comparison of six implicit real-time optimization schemes. *Journal 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.

S. Jang, B. Joseph, and H. Mukai. On-line optimization of constrained multivariable chemical processes. *AIChE J.*, 33(1):26–35, 1987.

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.

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.

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.

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(12):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.

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. *Journal of Loss Prevention in the Process Industries*, 12(6):485–493, 1999.

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.

E. Zafiriou and J. Zhu. Optimal control of semi-batch processes in the presence of modeling error. In *American Control Conference*, pages 1644–1649, May 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.